소프트웨어 공학개론

Introduction to Software Engineering

2022-1학기 (Spring)
선문대학교 AI소프트웨어학과

5월 3일 (화)

강의 목차

선문대학교

- 7. 아키텍처설계
  - 7.1 아키텍처설계란?
  - 7.2 아키텍처설계공정
  - 7.3 아키텍처스타일
- 연습 문제

7.1 아키텍처설계란?

선문대학교

목차

- 7.1.1 아키텍처란 무엇인가?
- 7.1.2 소프트웨어아키텍처에 관련되는 품질특성
- 7.1.3 아키텍처의 역할

7.1 아키텍처설계란?

선문대학교

7.1.1 아키텍처란 무엇인가?

- 소프트웨어아키텍처
  - 소프트웨어의 골격이 되는 기본구조
  - 소프트웨어에 대해서 관계자들이 시스템 전체의 성질을 이해하기 위한 체계를 제공
  - 아키텍처를 결정하면, 설계 및 개발의 기본사항에 영향을 줌
    - 소프트웨어의 분할과 구조화
    - 프로그램 및 도구의 재이용
    - 성능과 신뢰성 등 품질특성의 평가

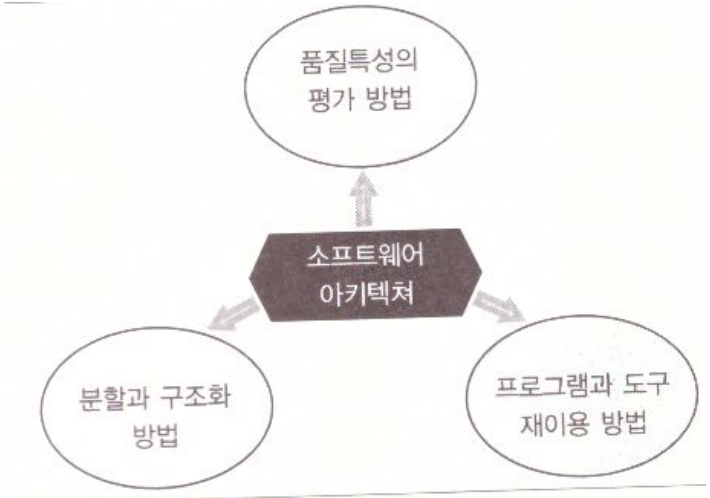


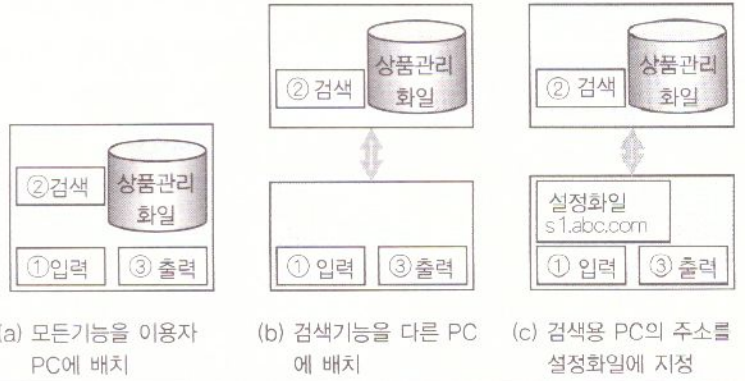
그림 7.3 | 아키텍쳐의 결정에 의해 규정되는 기본사항들

7.1 아키텍처설계란?

선문대학교

7.1.2 소프트웨어아키텍처에 관련되는 품질특성

- (1) 실행시의 성능에 미치는 영향
  - 품질특성 중 “효율성”의 부특성 “시간적 효율성”
    - 짧은 응답시간
    - 빠른 처리속도
    - 지정된 단위시간당 처리량 확보
  - 그림 예: (a) “상품검색”의 기능 ① ~ ③을 1개의 모듈로 작성하고, 상품관리파일과 함께 이용자의 컴퓨터에서 동작시킴
    - 상품관리파일의 검색시간이 짧다면, 이용자가 키보드로 상품검색키워드를 입력하여 결과가 표시될 때까지의 시간(응답시간)은 짧아짐



(a) 모든기능을 이용자 PC에 배치 (b) 검색기능을 다른 PC에 배치 (c) 검색용 PC의 주소를 설정화일에 지정

그림 7.4 “상품검색”의 기능분할 및 배치의 사례

7.1.2 소프트웨어아키텍처에 관련되는 품질특성

- (1) 실행시의 성능에 미치는 영향
  - 그림 예: (b) ① ~ ③의 처리를 각각 별개의 모듈들로 작성하여 ①과 ③을 이용자의 컴퓨터에서 동작시키고 ②를 다른 컴퓨터에서 동작시킴
    - 상품관리 파일의 검색시간이 짧더라도 ①과 ③의 기능과 ②의 기능 사이에서 컴퓨터들간의 통신이 필요하므로 응답시간이 길어지게 됨
    - 그러나 상품관리파일이 대규모화되고 검색시간이 매우 길어지게 되면, 이용자의 컴퓨터에 비해서 매우 속도가 빠른 컴퓨터로 ②를 동작시키는 것이 응답시간이 짧게 될 가능성이 있음

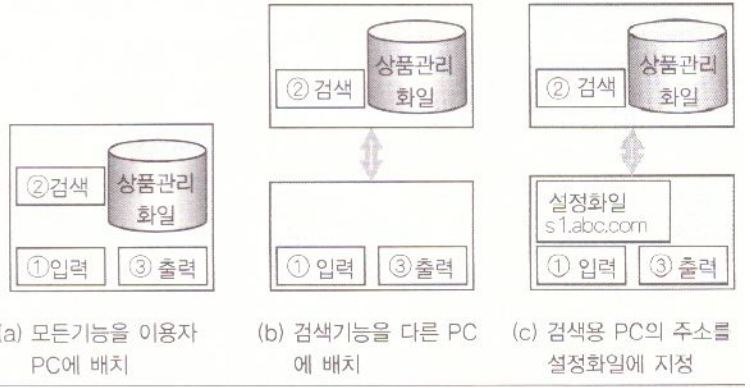


그림 7.4 "상품검색"의 기능분할 및 배치의 사례

7.1 아키텍처설계란?

7.1.2 소프트웨어아키텍처에 관련되는 품질특성

- (1) 실행시의 성능에 미치는 영향
  - 소프트웨어의 모듈 분할/할당, 모듈간 통신에 따라서 응답성이 다름
  - Web 서비스 등과같이 네트워크에 접속된 여러 개의 컴퓨터들에 처리기능을 분산한 경우
    - 기능의 분할과 배치 방법이 응답성과 단위시간당 처리량 등과 같은 시간효율성에 미치는 영향이 커짐
  - 통신지연시간, 접속단말기의 개수, 서버와 단말기의 처리능력 차이 등을 고려하여 적절한 아키텍처 검토 필요

7.1 아키텍처설계란?

선문대학교

7.1.2 소프트웨어아키텍처에 관련되는 품질특성

- (2) 개발시의 품질특성에 미치는 영향
  - 변경이 예상되는 기능을 가능한 한 모아두거나, 다른 기능과 분리시키는 구조인 경우
    - 변경작업을 수행시 영향범위를 줄일 수 있음
  - 그림 예: (b) 온라인 판매업무에서는 "상품검색"과 "상품구입"은 상품관리파일을 사용
    - 상품관리파일의 데이터구조와 참조방법을 변경시킬 가능성이 예상되는 경우
      - "상품검색"기능 ②와 ①, 그리고 ③으로 분리시킨 구조로 함
      - ②를 "상품구입"의 상품관리파일을 참조하는 기능으로 모아 정리
    - 상품관리파일에 관한 기능을 변경했을 경우
      - ①과 ③에 영향을 주지 않으면서 ②의 기능만 변경하면 됨
  - 품질특성 "유지보수성"의 부특성 "변경용이성" 향상

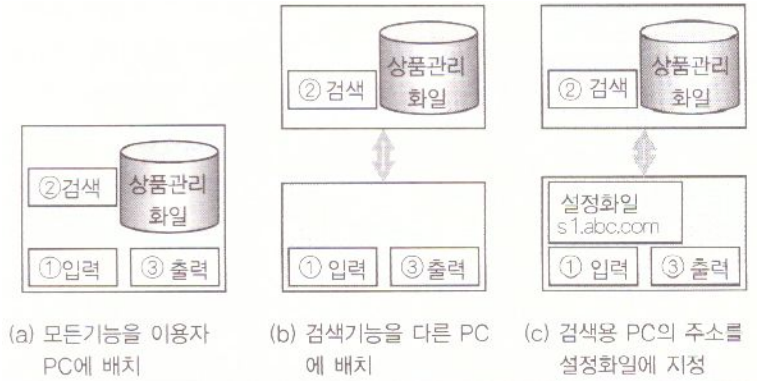


그림 7.4 "상품검색"의 기능분할 및 배치의 사례

7.1.2 소프트웨어아키텍처에 관련되는 품질특성

- (3) 운용시의 품질특성에 미치는 영향
  - 가동환경을 변경할 때마다 프로그램을 변경해야 하는 문제점 발생
    - 네트워크 및 컴퓨터 하드웨어, 그리고 운영체제에 제약을 주는 경우
    - 가동환경에 의존하는 정보가 프로그램 내부에 직접 기술되어 있는 경우

7.1.2 소프트웨어아키텍처에 관련되는 품질특성

- (3) 운용시의 품질특성에 미치는 영향
  - 그림 예: "상품검색"에서 ②를 설치해 놓는 컴퓨터의 주소를 ①의 프로그램 내부에 기술함
    - 위와 같은 문제점을 회피하기 위해 (c)와 같이
      - ②를 설치해 놓은 컴퓨터의 주소를 설정파일에 기술해두고
      - ①의 프로그램에서 설정파일을 읽어들여 해당주소를 사용하여 ②와 통신
    - 설정파일을 사용하여 가동환경을 동적으로 설정할 수 있는 구조로 만들면
      - 소프트웨어를 다른 환경에 이식할 때의 수고를 줄일 수 있음
      - 소프트웨어를 지정된 환경에 설치하기 수월해짐
  - 품질특성 "이식성"의 부특성 "순응성"과 "설치성" 향상

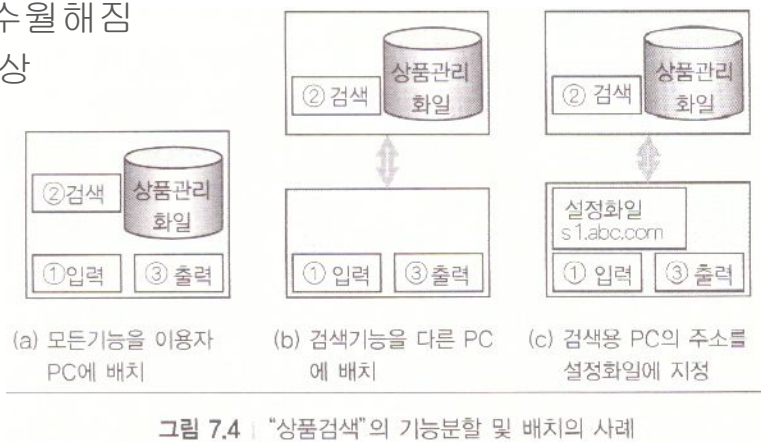


그림 7.4 "상품검색"의 기능분할 및 배치의 사례

7.1.2 소프트웨어아키텍처에 관련되는 품질특성

- (3) 운용시의 품질특성에 미치는 영향
  - 품질특성에 대한 요구조건을 아키텍처에 대한 요구조건으로 수용 필요
  - 신뢰성 관점에서 고장내성을 향상(고장이 발생하더라도 회복)시키려는 경우
    - 중복되는 구성요소(컴포넌트)들을 포함시켜두고
    - 시스템을 정지시키지 않고 컴포넌트를 교환/갱신가능한 구조로 만드는 것을 고려

표 7.2 | 품질특성의 목표와 소프트웨어아키텍쳐에 대한 요구사항

품질특성	목표(예)	아키텍쳐에 대한 요구사항(예)
효율성	응답성을 향상시킨다.	성능에 밀접하게 관련있는 조작들을 통신량이 적은 소수의 서브시스템들에게 국소화시킨다.
안전성	보안관리를 수월하게 한다.	계층화하여 보호해야할 기능을 가장 안쪽에 국소화하여 보안검사를 수월하게 한다.
신뢰성	고장내성을 향상시킨다.	중복적인 컴포넌트들을 포함시켜서, 시스템을 정지시키지 않고 컴포넌트를 변경·갱신가능한 구조로 만든다.
유지 보수성	확장성, 변경용이성, 테스트용이성을 향상시킨다.	비교적 작은 독립적인 컴포넌트들을 사용하여, 데이터공유를 회피할 수 있는 구조로 만든다.

7.1.3 아키텍처의 역할

- 개발대상이 되는 소프트웨어의 비기능적인 성질을 검토하여 기본구조를 정함
- 이에 따라 다음과 같은 시스템전체에 관련된 성질들을 이해하기 위한 체계를 제공함
  - (1) 전체의 흐름
  - (2) 통신패턴
  - (3) 처리규모와 성능
  - (4) 실행제어의 구조
  - (5) 확장성(이용자 수와 프로세서 수가 증가했을 때, 유연하게 확장/대응할 수 있는 정도)
  - (6) 소프트웨어전체에 관련된 일관성
  - (7) 장래의 발전에 대한 전망
  - (8) 입수가능한 부품과 부품의 적합성

7.1.3 아키텍처의 역할

- 소프트웨어아키텍처 설계 및 문서화의 잇점
  - (1) 관여자들 사이의 의사소통
    - 관여자들에게 있어 소프트웨어의 상세사항을 이해할 수 없는 경우에도 각각의 입장에서 소프트웨어의 성질에 관한 의사소통을 시도하는데 있어 유용
      - 소프트웨어의 성질: 성능과 변경용이성 등과 같은 비기능적 품질특성
  - (2) 시스템의 해석
    - 양립되는 특성(trade-off)에 대해서 시스템 개발의 기본방침에 견주어보면서 조절해서 타협점을 결정
      - trade-off: 변경용이성과 응답성, 보안과 속도성과 같은 이율배반적인 관계
      - trade-off를 맞춘다(또는 균형을 맞춘다): 설계상의 판단에 의해 타협점을 정하는 것
    - 시스템의 초기단계에서 trade-off를 맞추기 위해 시스템의 해석이 필요
      - 아키텍처상의 오류를 개발초기단계에서 발견할 수 있도록 하는데 유용
  - (3) 소프트웨어의 재이용
    - 유사한 요구사항에 대응하도록 만들려는 소프트웨어시스템의 경우
      - 같은 아키텍처를 채용하여 광범위하게 재이용할 수 있음

7.2 아키텍처설계공정

목차

7.2 아키텍처설계공정

선문대학교

7.2.1 아키텍처 설계상의 과제

- (1) 설계의 기본방침
  - 설계 전에 설계시 판단을 내릴 때 필요한 평가기준을 명확히 해야 함
    - 평가기준: 개발기간, 개발비용, 개발형태, 이용기간, 이용형태, 개발 후의 취급 등
    - 설계시의 판단이 크게 다른 경우의 예
      - 특정한 환경에서 특정한 목적으로 여러 사람들이 단기간동안 사용하고 역할을 끝내는 소프트웨어
      - 장기간 다수의 이용자들이 다양한 환경에서 사용하는 것이 예상되는 소프트웨어
  - 확장성을 높이거나, 성능을 중시하는 등 가치기준을 기본방침으로 하여 미리 규정해야 함

- (2) 기본구조를 평가하는 관점
  - 목적에 따라 다양한 구조를 가짐
    - 시스템의 응답성
    - 수정용이성
    - 신뢰성

7.2 아키텍처설계공정

7.2.1 아키텍처 설계상의 과제

- (3) 기본구조에 관련된 평가의 추적가능성
  - Trade-off를 맞추어야 함
    - 서로 상반되고 충돌되는 요구사항들에 대해 기본방침에 따라 제반 조건들을 고려하여 타협점을 찾아내서 기본구조를 정하는 작업을 반복 수행
    - 고려해 본 조건, 토대가 된 기본방침, trade-off를 맞춘 결과 등을 명확하게 기록하여 참조
      - 상세설계를 수행하는 경우 또는 수정/확장하는 경우에 부주의하여 기본구조를 훼손하는 일이 없도록 하기 위해

표 7.3 | 요구사항의 충돌과 Trade-Off

요구사항1	요구사항2
일정	예산
비용	강건성
고장내성	규모
보안	성능

7.2.2 아키텍처에 대한 관점

- 뷰(View): 관점에 따라서 아키텍처를 표현
  - 논리View
    - 시스템이 어떤 기능을 가지고 있고, 각 기능들이 논리적으로 어떤 구조로 구현되는지 나타냄
  - 실행View
    - 시스템이 공정과 업무 등의 실행단위들에 의해 어떻게 구성되고 있는지를 나타냄
    - 통신의 종류에 대해서 명시적으로 나타냄
    - 성능 등과 같은 실행 시의 품질특성과 관련
  - 개발View
    - 시스템이 어떤 단위로 파일들에 정리되어, 어떤 디렉토리구조로 관리되는지를 나타냄
    - 유지보수성과 수정용이성 등의 품질특성에 영향
  - 배치View
    - 시스템을 배치했을 때의 구조
      - 프로세스 등과 같은 실행시의 단위를 네트워크상에 있는 어떤 기계의 어떤 프로세스에 배치하는지 등
    - 성능뿐아니라 신뢰성, 안전성 등 다양한 품질특성과 관련

7.2.3 아키텍처의 설계기법

- 표준화된 설계기법이 있는 것이 아니며, 대다수의 설계자들에게 이용되어 안정화된 기법 미존재
  - 설계자들마다 각각 서로 다른 방법으로 아키텍처설계공정을 수행함
  - 아키텍처설계자가 가지고 있는 어플리케이션에 대한 지식과 설계자의 기술력 및 직감에 의존
  - 경험이 풍부한 설계자는 지금까지 개발했던 유사한 소프트웨어들을 참고하여 기본구조 결정
    - 상세설계와는 다르게 설계순서를 명확하게 의식하지 않는 경향이 있음
    - 무엇이 기본구조인지 또는 기본설계가 품질특성을 만족하고 있는지 등을 명시하지 않는 경우가 많음

- 최근 아키텍처 설계시 기법 제안됨

7.2.3 아키텍처의 설계기법

- Jan Bosch의 기법
  - 품질측면의 요구사항에 대해서는 충분히 주의하지 않는다는 점을 지적
    - 객체지향분석기법 등 기존의 설계기법에서는 기능적 측면의 요구사항에 대해서만 중시
  - 그림과 같은 아키텍처 설계기법 제안

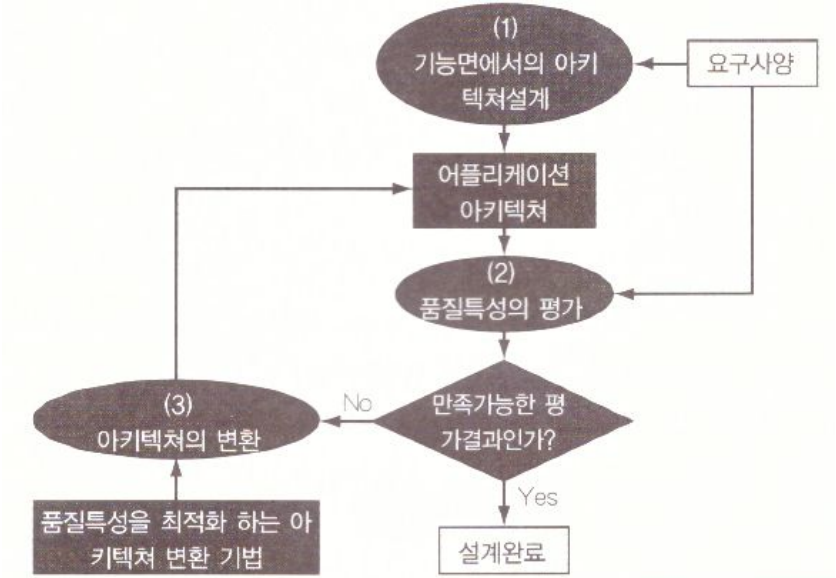


그림 7.5 | Bosch의 아키텍쳐설계기법

7.2.3 아키텍처의 설계기법

- Jan Bosch(얀 보쉬)의 기법
  - (1) 기능적 측면에서의 아키텍처설계
    - 요구사항에 지정된 기능적 요구사항에 대응할 수 있는 소프트웨어아키텍처를 설계
      - 애플리케이션 아키텍처의 첫 번째 버전
  - (2) 품질특성의 평가
    - 정성적 및 정량적 평가기술을 사용하여 품질특성값을 산출하고 요구사항에 규정되어 있는 값과 비교평가
      - 산출한 값이 모두 양호하여 만족할 수 있는 평가결과로 판정되는 경우, 아키텍처 설계가 완료됨
      - 그렇지 않은 경우, 아키텍처의 변환단계 실시
  - (3) 아키텍처의 변환
    - 아키텍처의 선택과 변경
      - 설계한 아키텍처가 요구된 품질특성을 만족하지 못하는 경우, 아키텍처 변환
      - 소프트웨어 전체의 구조에 관련된 기법으로서 아키텍처스타일(Architecture Style)을 사용한 변환이 있음

7.3 아키텍처스타일

개요

- 아키텍처 패턴(Architecture Pattern) 또는 아키텍처스타일 (Architecture Style)
  - 다수의 소프트웨어에 반복적으로 나타나는 구조를 카탈로그에 규정해 둔 것

목차

7.3 아키텍처스타일

7.3.1 아키텍처의 유형화

- Mary Shaw와 David Garlan이 제시하고 있는 스타일(실행View 토대)을 중심으로 다른 view와 관련되어 있는 스타일을 기술함
  - 모델: 설계자가 기본구조를 표현할 때에 공통적으로 수행하는 작업을 기반으로 스타일을 기술/분류하고 그 결과로 얻어지는 스타일
    - (1) 기능분할의 모델화
      - 소프트웨어시스템을 여러 개의 독립적인 기본요소로 분할하고, 각 기본요소 사이의 관계에 의해 전체의 구조를 표현
      - 요소들 사이의 통신을 명시
      - 기능의 분할과 배치를 토대로 구조를 분류
    - (2) 제어관계의 모델화
      - 소프트웨어시스템을 여러 개의 독립적인 기본요소로 분할하고, 각 기본요소 사이의 제어관계를 모델화
      - 데이터와 제어의 흐름을 토대로 구조를 분류

7.3.2 기능분할의 모델화

- (1) 계층모델
  - 기능들을 상위에서 하위에 이르기까지 계층적으로 나열하여 배치한 모델
  - 그림(a): 3계층 모델
    - (하위)데이터베이스
    - (중간)애플리케이션(Business Logic)
    - (상위)유저인터페이스
    - Web 서비스 등과 같이 데이터베이스를 다루는 비즈니스프로그램에서 많이 사용

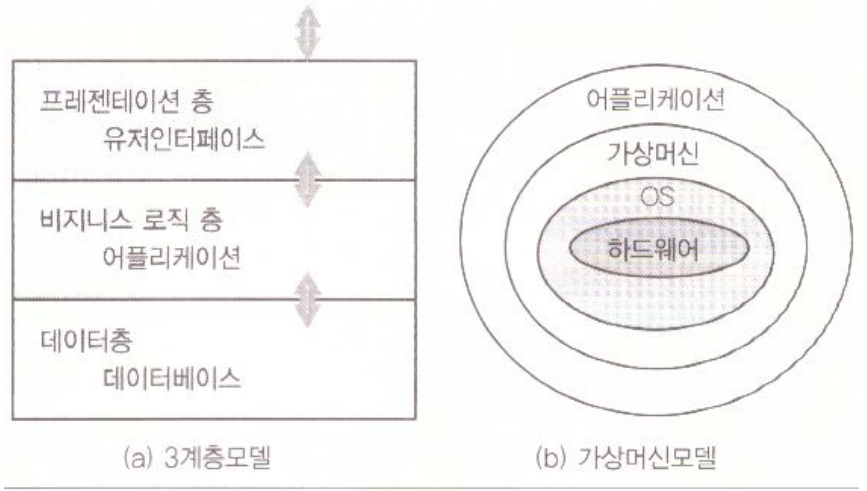


그림 7.6 계층모델

7.3.2 기능분할의 모델화

- (1) 계층모델
  - 그림(b): 가상머신모델
    - 시스템을 계층화하여 각 계층에서 각각 서비스를 제공하도록 구성
    - 특징
      - 가상머신: 애플리케이션을 위한 추상도가 높은 독자적인 명령어집합을 제공
      - 애플리케이션: 하드웨어와 OS에 의존하지 않는 애플리케이션 구현 가능
        - 유저인터페이스가 서로 다른 다양한 애플리케이션의 구현이 수월해짐

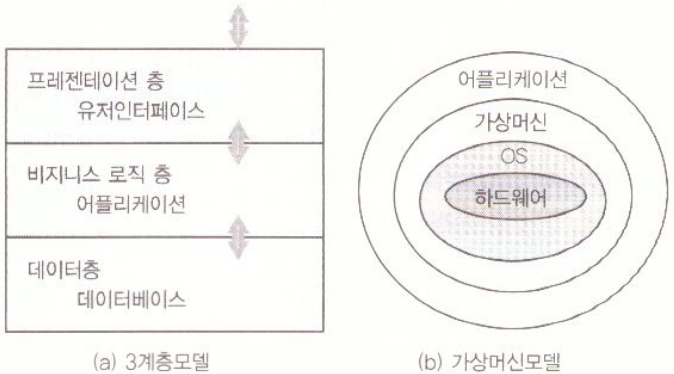


그림 7.6 계층모델

7.3.2 기능분할의 모델화

- (1) 계층모델
  - 그림(b): 가상머신모델
    - 장점
      - 각 계층마다 단계적으로 개발할 수 있음
      - 어떤 계층을 변경했을 때 발생하는 영향범위가 변경이 발생한 계층에 인접해 있는 계층에 한정됨
    - 단점
      - 모델 적용 여부가 문제에 의존
      - 이 방법으로 시스템을 구조화하는 것이 어려운 경우가 있음

7.3.2 기능분할의 모델화

- (2) 클라이언트서버모델
  - 데이터와 처리기능을 클라이언트와 서버에 분할시켜서 운용하는 분산시스템모델
  - 네트워크를 통해서 서비스를 제공하는 분산시스템의 기본모델로서 널리 사용됨
  - 구성
    - 독립된 서버들의 집합: 데이터검색 등과 같이 특정한 서비스를 제공
    - 클라이언트들의 집합: 서버들의 서비스를 호출

7.3.2 기능분할의 모델화

- (2) 클라이언트서버모델
  - 그림 예: 클라이언트와 서버의 부담비율의 다양화
    - 씬 클라이언트(Thin Client)
      - 클라이언트의 부담을 줄여 필요최소한의 기능들만 탑재한 클라이언트
    - 팻 클라이언트(Fat Client)
      - 애플리케이션과 데이터베이스 등의 기능 및 환경을 갖춘 클라이언트

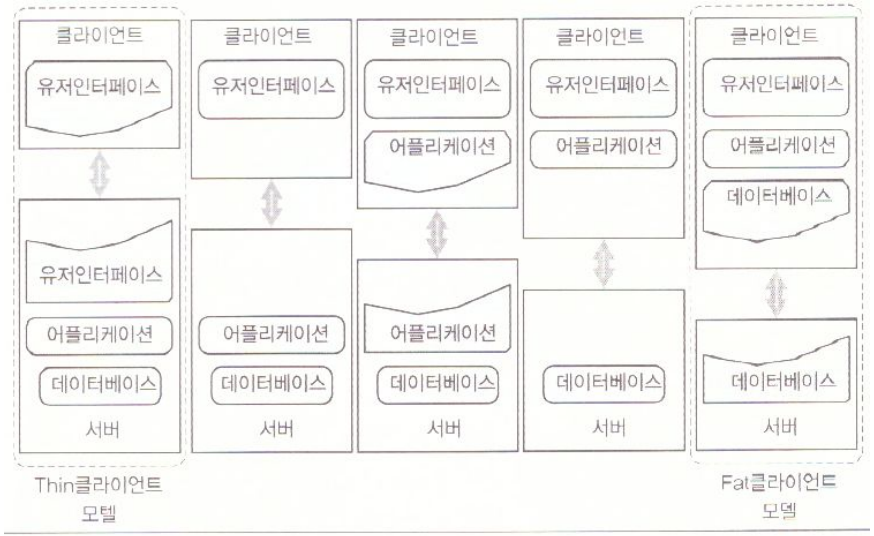


그림 7.7 | 클라이언트서버모델

7.3.2 기능분할의 모델화

- (3) 데이터중심형 모델(또는 리파지토리 모델(Repository Model))
  - 여러 개의 서브시스템들이 공유데이터를 통해서 데이터를 교환하면서 처리를 수행하는 모델
  - 중앙의 데이터베이스(Repository)에 공유데이터를 보존하고, 모든 서브시스템이 액세스함(그림 예)
  - 대량의 데이터를 공유하는 경우 널리 사용됨
  - 장점: 공유하는 데이터를 집중관리함으로써 데이터의 무모순성과 일관성을 쉽게 유지
  - 단점: 공유데이터에 대한 액세스가 집중함으로써 성능저하문제 발생 → 물리적 분산을 검토 필요

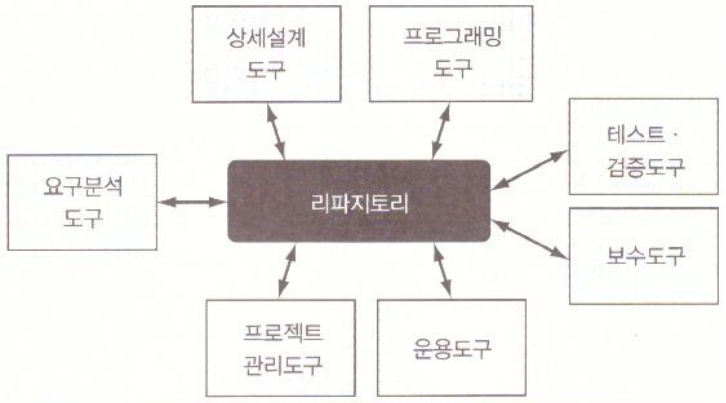


그림 7.8 | 데이터중심형 모델(Repository Model)

7.3.3 제어관계의 모델화

- (1) 데이터흐름모델
  - 입력데이터를 처리하여 출력을 생성하는 변환기능에 의해 구성되는 모델
    - 그림 (a): 컴파일러의 순차적인 변환처리
    - 그림 (b): UNIX의 shell 파이프와 필터를 조합하여 만들어지는 아키텍처
      - 필터: 1개 이상의 데이터스트림을 입력으로 하고, 이를 변환하여 1개의 데이터스트림을 출력
      - 파이프: 1개의 필터 출력을 다른 필터의 입력에 연결시킨 것
  - 상호작용을 지원하는(Interactive한) 시스템에 적합

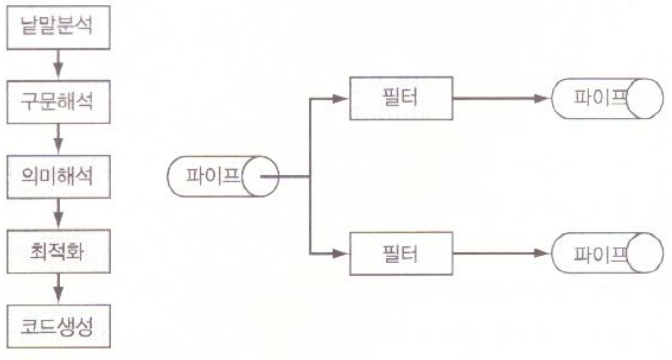


그림 7.9 데이터흐름모델

7.3.3 제어관계의 모델화

- (2) 제어모델
  - 서브시스템 사이의 제어에 주목한 모델
  - 집중형 제어모델
    - 1개의 서브시스템이 다른 서브시스템의 시작과 정지 등의 제어를 모두 관장
    - Goal-Return Model: 상위의 서브루틴이 하위의 서브루틴을 호출함으로써 제어가 이동함
      - 순차형 시스템에 적용가능
    - Manager Model: 1개의 구성요소가 다른 요소들의 시작과 정지, 협조를 제어
      - 병행처리시스템에 적용가능

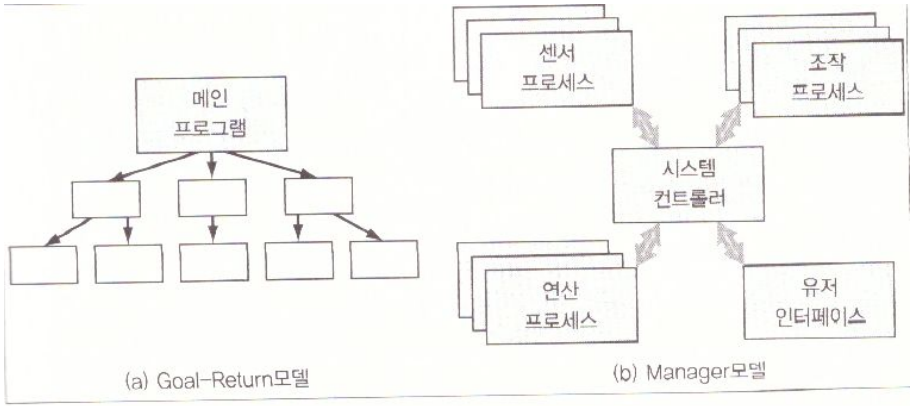


그림 7.10 | 집중형제어모델

7.3 아키텍처스타일

7.3.3 제어관계의 모델화

- (2) 제어모델
  - 이벤트구동형 제어모델
    - 외부에서 발생한 이벤트에 의해 처리가 구동
    - 이벤트 발생 타이밍은 해당 이벤트를 처리하는 서브시스템의 외부에 있으므로 각 서브시스템에서는 언제 이벤트가 발생할 지 알 수 없음
    - Broadcast Model
      - 그림(a)와 같이, 서로 다른 컴퓨터에서 서브시스템을 네트워크로 통합하는 경우 효과적
      - 서브시스템에 각각 특정한 이벤트에 대해서 등록해 두면, 이벤트가 발생했을 때 제어가 해당 서브시스템으로 이동되어 이벤트처리 시작할 수 있음

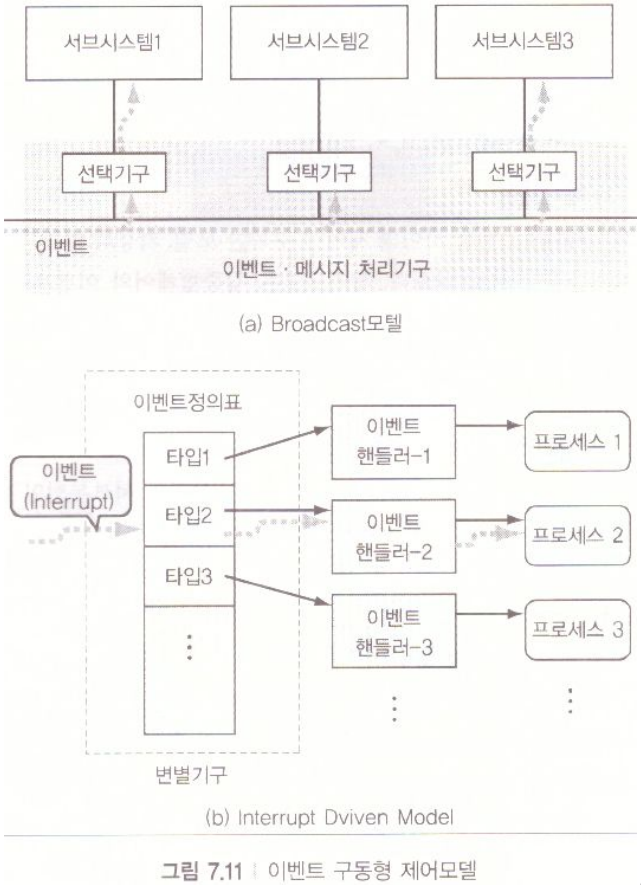


그림 7.11 | 이벤트 구동형 제어모델

7.3 아키텍처스타일

7.3.3 제어관계의 모델화

- (2) 제어모델
  - 이벤트구동형 제어모델
    - Interrupt Driven Model
      - 이벤트에 대한 빠른 응답이 중요시되는 실시간 처리에 적용가능
      - 그림(b)와 같이, 이벤트의 각 유형마다 이벤트처리루틴(Event Handler)이 정의됨
      - 이벤트의 발생이 interrupt에 의해 통지되면, 변별기구가 이벤트정의표를 조사하여 이벤트의 유형에 알맞은 이벤트처리루틴에게 제어를 이동시킴
      - 변별기구를 하드웨어 스위치로 구현하면, 실시간처리 성능이 뛰어난 시스템 구현가능

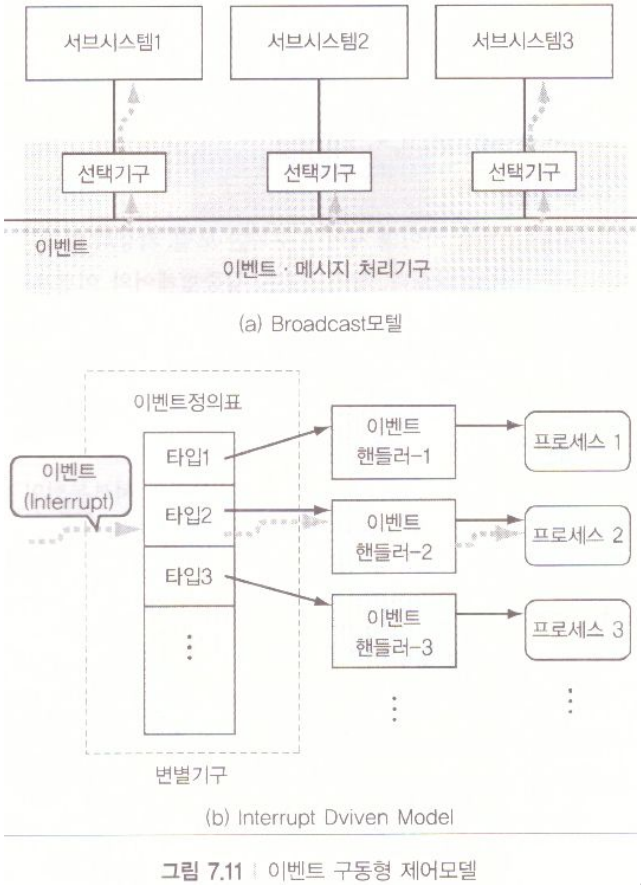


그림 7.11 | 이벤트 구동형 제어모델

연습문제 (객관식)

각 질문에 대한 알맞은 답의 번호(숫자)를 괄호안에 표시하시오.

- 1. 아키텍처설계의 기본방침으로서의 기준 중 명확히 하거나 미리 규정해야하는 기준을 2가지 고르시오. (,)
  (1) 평가기준 (2) 역할기준 (3) 가치기준 (4) 형태기준

- 2. 관점에 따라서 아키텍처를 표현하는 View 중 시스템이 어떤 단위로 파일들에 정리되어, 어떤 디렉토리구조로 관리되는지를 나타내며 유지보수성과 수정용이성 등의 품질특성에 영향을 주는 View는 무엇인가? ( )
  (1) 논리View (2) 실행View (3) 개발View (4) 배치View

- 3. 기능의 분할과 배치를 토대로 분류한 아키텍처 모델에 속하지 <u>않는</u> 것을 2가지 고르시오. (,)
  (1) 계층모델 (2) 클라이언트서버모델 (3) 데이터중심형모델
  (4) 데이터흐름모델 (5) 제어모델

객관식 문제 (답안지)

- 1. (1, 3) → 평가기준, 가치기준
- 2. (3) → 개발View
- 3. (4, 5) → 소프트웨어의 제어관계를 데이터와 제어의 흐름을 중심으로 분류한 모델

연습문제 (객관식)

각 질문에 대한 알맞은 답의 번호(숫자)를 괄호안에 표시하시오.

❑ 4. 데이터중심형(또는 리파지토리)모델에 대한 설명으로 알맞은 것을 고르시오.
( )
(1) 기능들을 상위에서 하위에 이르기까지 계층적으로 나열하여 배치한 모델
(2) 1개의 서브시스템이 다른 서브시스템의 시작과 정지 등의 제어를 모두 관장하는 모델
(3) 데이터와 처리기능을 클라이언트와 서버에 분할시켜서 운용하는 분산시스템모델
(4) 여러 개의 서브시스템들이 공유데이터를 통해서 데이터를 교환하면서 처리를 수행하는 모델
(5) 입력데이터를 처리하여 출력을 생성하는 변환기능에 의해 구성되는 모델

객관식 문제 (답안지)

❑ 4. (4) →
(1) 계층모델
(2) 집중형제어모델
(3) 클라이언트서버모델
(5) 데이터흐름모델

연습문제 (주관식)

선문대학교

주관식 문제 (괄호안에 알맞은 단어를 입력하시오.)

- 1. 소프트웨어아키텍처를 결정에 의해 규정되는 설계 및 개발의 기본사항 3가지를 쓰시오.
  ([1]) 방법,
  ([2]) 방법,
  ([3]) 방법
- 2. 소프트웨어아키텍처는 시간적 효율성과 같은 ([1])시의 성능, 변경용이성과 같은 ([2])시의 품질특성, 순응성, 설치성 등과 같은
  ([3])시의 품질특성에 영향을 준다.
- 3. 변경용이성과 응답성, 보안과 속도성과 같은 이율배반적인 관계를 ( )라고 한다.
- 4. 다수의 소프트웨어에 반복적으로 나타나는 구조를 카탈로그에 규정해 둔 것을 ( )이라고 한다.

주관식 문제 (답안지)

- 1. (1) 소프트웨어의 분할과 구조화
  (2) 프로그램과 도구의 재이용
  (3) 품질특성의 평가
- 2. (1) 실행
  (2) 개발
  (3) 운용
- 3. trade-off
- 4. 아키텍처 패턴 (Architecture Pattern) (또는 아키텍처스타일 (Architecture Style))

연습문제 (주관식)

선문대학교

주관식 문제 (괄호안에 알맞은 단어를 입력하시오.)

- 5. 아키텍처스타일 중 Web 서비스 등과 같이 비즈니스프로그램에서 많이 사용하는 3계층모델에서 3계층을 하위층부터 상위층 순서대로 쓰시오.
  (하위층)([1]),
  (중간층)([2]),
  (상위층)([3])

- 6. 클라이언트서버모델 중 클라이언트의 부담을 줄여 필요최소한의 기능들만 탑재한 클라이언트를 ([1]) 클라이언트,
  애플리케이션과 데이터베이스 등의 기능 및 환경을 갖춘
  클라이언트를 ([2]) 클라이언트라고 한다.

- 7. 여러 개의 서브시스템들이 공유데이터를 통해서 데이터를 교환하면서 처리를 수행하는 모델을 ( )
  모델이라고 한다.

주관식 문제 (답안지)

- 5. (1) 데이터베이스
  (2) 애플리케이션
  (3) 유저인터페이스

- 6. (1) 씬(Thin)
  (2) 팻(Fat)

- 7. 데이터중심형 (또는 리파지토리 (Repository))

연습문제 (주관식)

선문대학교

주관식 문제 (괄호안에 알맞은 단어를 입력하시오.)

- 8. 집중형제어모델 중 ([1]) Model은 상위의 서브루틴이 하위의 서브루틴을 호출함으로써 제어가 이동하며, 순차형 시스템에 적용가능하고, ([2]) Model은 1개의 구성요소가 다른 요소들의 시작과 정지, 협조를 제어하며, 병행처리시스템에 적용가능하다.

- 9. 외부에서 발생한 이벤트에 의해 처리가 구동되는 이벤트구동형제어모델은 ([1]) Model과 ([2]) Model이 있다.

주관식 문제 (답안지)

- 8. (1) Goal-Return
  (2) Manager

- 9. (1) Broadcast
  (2) Interrupt Driven

Any questions?