소프트웨어공학 활용

아키텍처 설계

한국기술교육대학교
온라인평생교육원

학습내용

- 아키텍처 개요
- 분산 시스템과 응용 시스템 아키텍처

학습목표

- 소프트웨어 아키텍처의 기본 개념을 설명할 수 있다.
- 분산 시스템과 응용 시스템 아키텍처의 차이점을 이해하고 설명할 수 있다.

아키텍처 개요

1 아키텍처 설계 개요

1 소프트웨어 아키텍처란

1 정의

시스템을 구성하는 서브 시스템들을 식별하고,
서브시스템의 제어와 통신을 위한 프레임워크를 설정하는
설계 프로세스의 산출물을 기술한 것

- 구성요소
- 구성요소들 사이의 관계
- 구성요소들이 외부에 드러내는 속성
- 구성요소들과 주변 환경 사이의 관계
- 구성요소들이 제공하는 인터페이스
- 구성요소들의 협력 및 조립 방법

아키텍처 개요

1 아키텍처 설계 개요

1 소프트웨어 아키텍처란

2 아키텍처 설계

1. 시스템 설계 프로세스의 초기 단계
2. 명세화 및 설계 프로세스 사이의 연결을 나타냄
3. 특정 명세화 활동과 동시에 진행되는 것이 보통
4. 주요 시스템 컴포넌트들과 이들 간의 통신을 식별하는 것을 포함

3 장점

1. 프로젝트 참여자 의사소통 수단으로 사용 용이
2. 시스템 분석에 용이
3. 대규모 재사용(Large-Scale Reuse)이 가능

아키텍처 개요

1 아키텍처 설계 개요

2 아키텍처와 시스템 특성

1 특성

성능 (Performance)	• 서브시스템 통신 최소화를 위해 오퍼레이션을 지역화 • 작은 컴포넌트보다는 큰 컴포넌트가 좋음
보안 (Security)	• 계층형 아키텍처를 사용하고 주요한 요소는 내부 레이어에 놓는 것이 좋음
안전성 (Safety)	• 안전성이 중요한 기능은 적은 수의 서브시스템에 위치하는 것이 좋음
가용성 (Availability)	• 컴포넌트를 중복시키고 결함내성 메커니즘을 사용
유지보수성 (Maintainability)	• 작고 대체 가능한 컴포넌트를 사용

2 아키텍처 측면에서 충돌

- 큰 컴포넌트를 사용하면 성능은 개선되나 유지보수성이 안 좋아짐
- 중복된 데이터는 가용성을 개선시키나 보안성 유지가 어려워짐
- 안전성과 관련 있는 기능들을 지역화시키면 보다 많은 통신이 필요함 → 성능이 나빠짐

아키텍처 개요

1 아키텍처 설계 개요

3 아키텍처 품질 속성

개념적 무결성 (Conceptual Integrity)	정확성과 완전성 (Correctness and Completeness)	개발 용이성 (Buildability)
• 일관성 • 전체 시스템과 시스템 구성 요소가 일관되도록 아키텍처를 결정	• 사용자가 요구하는 기능을 충족시키는 정도 • 요구 분석 명세서와 일치하는 정도	• 전체 시스템을 적절한 모듈로 분할한 후 개발 팀에 알맞게 분배하여 개발 • 정해진 기간 내 완성 • 개발 과정 중 쉽게 변경할 수 있는 능력

4 아키텍처 설계 시 기술 방법

- 이해하기 쉽게 작성
- 명확하게 기술
- 표준화된 형식 사용
- 문서 버전 명시

아키텍처 개요

2 아키텍처 구축 절차 및 모델

1 아키텍처 구축 절차

요구사항 분석 → 아키텍처 분석 → 아키텍처 설계 → 검증 및 승인

1 요구사항 분석

- 소프트웨어 개발의 요구사항 분석 단계와 같음
- 품질 속성과 같은 비기능적인 요구사항에 더 많은 관심을 둠
- 요구사항 취득, 식별, 명세, 분류, 검증
- 기능적/비기능적 요구사항 분류 및 명세

2 아키텍처 분석

품질 속성 식별 → 우선순위 결정 → 반영 방법 개발

아키텍처 개요

2 아키텍처 구축 절차 및 모델

1 아키텍처 구축 절차

3 아키텍처 설계

관점 정의	이해 관계자 파악, 이해 관계자별 관점 정의
아키텍처 스타일 선택	Pipe-Filter, Mvc, Layer 등의 스타일 혼용 적용 가능
후보 아키텍처 도출	배경도 및 각 관점별 다이어그램 작성, 아키텍처 명세서 기술

4 검증 및 승인

아키텍처 평가	아키텍처 요구사항 만족도, 적합성, 품질 속성 간 절충 관계 등 평가
아키텍처 상세화	설계 방법 도출, 설계 패턴 고려(반복적 실행)
아키텍처 승인	이해 관계자들이 최종 승인

아키텍처 개요

2 아키텍처 구축 절차 및 모델

2 아키텍처 스타일

시스템의 아키텍처 모델은
일반적 아키텍처 모델, 아키텍처 스타일에 기초하게 됨

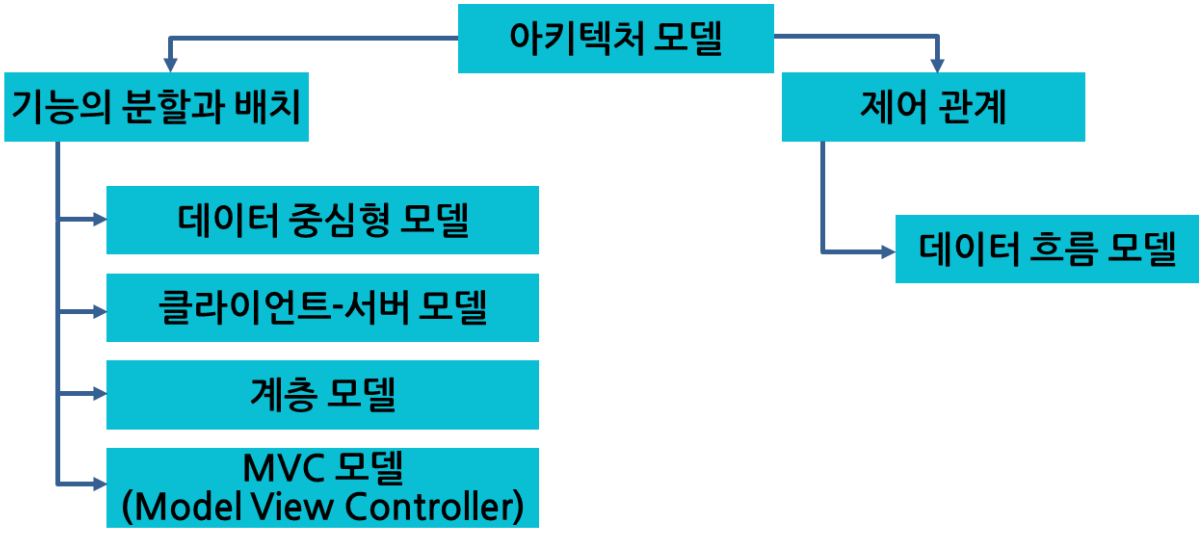
아키텍처 스타일이란?

클라이언트-서버 구성이나 계층형 아키텍처와 같은 시스템 구성에 관한 패턴을 말함

아키텍처 스타일들과 응용분야 및 장·단점 등을
잘 아는 것이 중요

대부분의 대형 시스템은 하나의 스타일로 만들지 못하고
부분마다 다른 아키텍처 스타일을 적용하게 됨

3 아키텍처 모델



아키텍처 개요

2 아키텍처 구축 절차 및 모델

3 아키텍처 모델

1 데이터 중심형 모델 - Repository Model

특징

- 주요 데이터가 Repository에서 중앙 관리

구성

- Repository와 여기에 접근하는 서브 시스템
  - Repository : 공동으로 활용하는 데이터 보관
  - 서브 시스템 : Repository에 접근하여 정보를 저장, 검색, 변경하는 역할

장점

- 데이터가 한군데에 모여 있기 때문에 데이터를 모순되지 않고 일관성 있게 관리 가능
- 새로운 서브시스템의 추가 용이

단점

- Repository의 병목 현상 발생 가능
- 서브 시스템과 Repository 사이의 강한 결합으로 변경 시 서브시스템에 영향을 줌

아키텍처 개요

2 아키텍처 구축 절차 및 모델

3 아키텍처 모델

2 클라이언트 서버 모델

네트워크를 이용하는 분산 시스템 형태

데이터, 처리 기능을 클라이언트와 서버에 분할하여 사용

분산 아키텍처에 유용

- 서버 : 클라이언트(서브시스템)에 서비스 제공
- 클라이언트 : 서버가 제공하는 서비스를 요청하는 서브시스템

3 계층 모델

기능을 몇 개의 계층으로 나누어 배치

하위 계층은 서버, 상위 계층은 클라이언트 역할

사용자 인터페이스	• 프레젠테이션 계층
애플리케이션	• 비즈니스 로직 계층
데이터베이스	• 데이터 계층

아키텍처 개요

2 아키텍처 구축 절차 및 모델

3 아키텍처 모델

4 Model View Controller 모델

1. 중앙 데이터 구조
2. 같은 모델의 서브시스템에 대하여 여러 뷰 서브 시스템을 필요로 하는 시스템에 적합
3. 3가지 서브 시스템 분리 이유 : 변경에 대한 영향을 덜 미치도록 하기 위함

Model 서브 시스템

- 뷰/제어 서브시스템과 독립되어 모든 데이터 상태와 로직 처리
- 특정 입출력 방식에 영향을 받지 않고 호출에만 응답함

View 서브 시스템

- 사용자와 직접 대화가 이루어지는 부분으로 데이터를 사용자에게 보여주는 역할

Controller 서브 시스템

- 뷰를 통한 사용자의 요청을 적절한 모델 쪽으로 넘겨줌
- 모델로부터 받은 응답을 다시 뷰를 통해 사용자에게 돌려줌

아키텍처 개요

2 아키텍처 구축 절차 및 모델

3 아키텍처 모델

4 Model View Controller 모델

장점

- 데이터를 화면에 표현하는 디자인과 로직을 분리함으로써 느슨한 결합 가능
- 구조 변경 요청 시 수정 용이

단점

- 기본 기능 설계로 인한 클래스 수의 증가로 복잡도 증가
- 속도가 중요한 프로젝트에 부적합

5 데이터 흐름 모델 - Pipe and Filter 구조

Filter

- Data Stream을 한 개 이상 입력 받아 처리 후 Data Stream 하나를 출력

Pipe

- Filter를 거쳐 생성된 Data Stream 하나를 다른 Filter의 입력에 연결

분산 시스템과 응용 시스템 아키텍처

1 분산 시스템 아키텍처

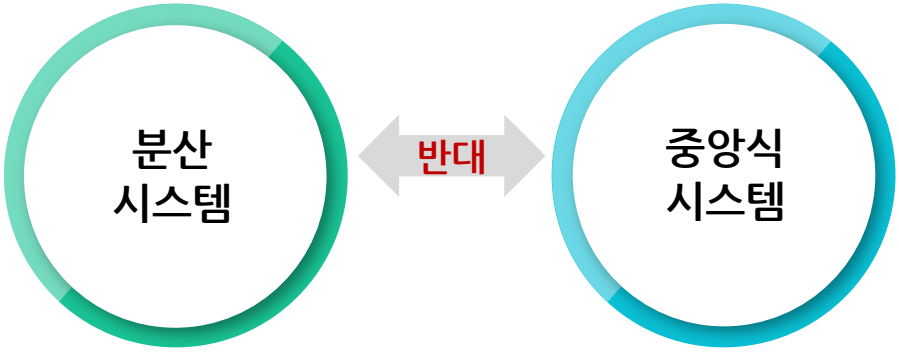
1 분산 시스템

1 분산 시스템이란

상호 연결하여 **특정 작업을**
공동으로 처리할 수 있도록 만들어진 시스템

네트워크 기반

독립적으로 운영될 수 있는 컴퓨터 시스템들



분산 시스템과 응용 시스템 아키텍처

1 분산 시스템 아키텍처

1 분산 시스템

2 특성 및 장점

▪ 가격 대비 성능 우수

- 중·대형 시스템을 기반으로 여러 사용자들이 공유하는 것보다 컴퓨터 여러 개의 네트워크를 연결 후 사용하는 것이 더 비용이 절감됨

▪ 분산된 자원 공유 가능

- 다른 기종의 여러 자원이 서로 연결되어 있다면 서로의 자원을 공유 및 사용 가능

▪ 연산 속도 향상

- 하나의 연산이 동시에 처리될 수 있는 여러 개의 부분 연산으로 분할된다면, 분산 시스템에 분산하여 처리할 수 있기 때문에 연산이 분산되어 속도가 향상됨

▪ 시스템 확장 용이

- 네트워크 하부 구조에 시스템을 추가 연결함으로써 간단하게 확장 가능

▪ 가용성 높음

- 네트워크가 연결된 곳이면 어디서든지 연결 가능

▪ 신뢰성 높음

- 일부 자원의 고장, 문제가 발생해도 분산된 시스템을 이용하면 동작을 계속하여 시스템 유지가 가능

분산 시스템과 응용 시스템 아키텍처

1 분산 시스템 아키텍처

1 분산 시스템

3 단점

- 중앙 시스템에 비해 설계가 복잡하고 어려움
- 분산된 컴퓨터의 데이터 정합성이나 통일성을 잃기 쉬움

4 분산 시스템 모델

Mini Computer | Workstation Server | Processor-Pool

- 몇 대의 중형급 컴퓨터들을 네트워크로 연결
- 사용자들은 시스템 각각에 연결된 단말기로 작업 수행
- 시분할 시스템을 네트워크 환경으로 확장한 형태

Mini Computer | **Workstation Server** | Processor-Pool

- 개인 PC들을 네트워크로 연결하고 이에 필요한 Server급 시스템을 추가로 연결하여 구성
- 사용자들은 작업을 자신의 PC에서 하지만 필요에 따라 서버로 접근해서 작업

분산 시스템과 응용 시스템 아키텍처

1 분산 시스템 아키텍처

1 분산 시스템

4 분산 시스템 모델

Mini Computer | Workstation Server | Processor-Pool

- 저가의 호스트(단순 시스템으로 구성)들을 몇몇 서버와 함께 연결하여 구성
- 요구한 작업에 대해 시스템은 프로세서 풀에서 필요한 만큼 프로세서를 할당하고 실행

5 계층화된 애플리케이션 아키텍처

표현 계층
- Presentation Layer
- 사용자에게 계산 결과를 제공하는 것과 사용자 입력을 수집하는 것과 관련

애플리케이션 처리 계층
- Application Processing Layer
- 예약 시스템의 예약 및 취소 등과 같은 애플리케이션의 특정 기능과 관련

데이터 관리 계층
- Data Management Layer
- 시스템 데이터 베이스 관리와 관련

분산 시스템과 응용 시스템 아키텍처

1 분산 시스템 아키텍처

2 마스터 / 슬레이브 아키텍처

1 개념

실시간 시스템에서 많이 사용되는 아키텍처

서로 다른 프로세서에서 실행되는 복수의 프로세스들로 구성됨

2 마스터 프로세스와 슬레이브 프로세스

마스터 프로세스

- 계산, 조정, 통신을 담당하고 슬레이브 프로세스를 제어함

슬레이브 프로세스

- 특정 동작을 수행함
- 예) 센서 제어, 기기 제어

3 실시간 시스템 모델의 구성요소

- 데이터 획득
- 데이터 처리와 계산
- 작동장치 관리 프로세서로 분리

분산 시스템과 응용 시스템 아키텍처

1 분산 시스템 아키텍처

3 클라이언트 서버 아키텍처

1 개념

애플리케이션은 서버가 제공하는 서비스의 집합과
이 서비스를 이용하는 클라이언트 집합으로 모델링 됨

- 서버 및 다른 클라이언트들의 존재를 알지 못함

클라이언트와 서버는 논리적 프로세스

프로세서와 프로세스 매칭은 항상 1:1일 필요 없음

분산 시스템과 응용 시스템 아키텍처

1 분산 시스템 아키텍처

3 클라이언트 서버 아키텍처

2 Thin Client Model

Thin Client Model이란?

- 모든 응용처리 및 데이터 관리가 서버에서 수행
- 클라이언트는 표현 계층만 구현

활용	예	단점
• 레거시 시스템(Legacy Systems)을 클라이언트 서버 아키텍처로 바꿀 때 사용	• 데이터 관리가 거의 필요 없는 컴파일러 같은 계산 중심의 애플리케이션 • 응용처리가 거의 필요 없는 데이터 중심의 애플리케이션 (웹 검색 등)	• 서버와 네트워크에 걸리는 부하가 큼

분산 시스템과 응용 시스템 아키텍처

1 분산 시스템 아키텍처

3 클라이언트 서버 아키텍처

3 Fat Client Model

Fat Client Model이란?

- 서버는 오직 데이터 관리만을 책임 짐
- 클라이언트 상의 소프트웨어는 애플리케이션 논리와 사용자와의 상호작용을 구현함

활용

- 클라이언트 컴퓨터의 사양이 응용 처리를 하기에 충분할 경우에 사용

예

- 응용처리가 클라이언트 상에서 상용제품으로 제공되는 애플리케이션
- 데이터를 계산하거나 처리하는 것의 비중이 큰 애플리케이션

단점

- 관리적인 면에서 Thin Client Model 보다 복잡
- 새로운 버전의 애플리케이션이 출시되면 모든 클라이언트에 배포 및 설치

분산 시스템과 응용 시스템 아키텍처

1 분산 시스템 아키텍처

4 분산 객체 아키텍처

1 개념

클라이언트, 서버 간 차이를 없애고 동일하게 분산시킨 객체로 아키텍처 구성

각각의 분산 요소는 다른 객체들에게 서비스를 제공 및 서비스를 받기도 함

2 특징

객체 간의 통신은 미들웨어 시스템을 이용

- ORB(Object Request Broker)

클라이언트 서버 시스템보다 설계가 어려움

유연성 제공 : 시스템 설계자는 서비스 제공에 대한 판단을 미룰 수 있음

새로운 리소스가 추가되기 쉬운 열린 시스템

시스템 확장성이 좋음

네트워크를 통해 객체를 주고 받으면서 시스템을 동적 재구성이 가능함

분산 시스템과 응용 시스템 아키텍처

1 분산 시스템 아키텍처

4 분산 객체 아키텍처

3 활용

시스템을 구조화하고 조직할 수 있도록 논리적 모델로 사용가능

클라이언트 서버 시스템의 구현을 위한 유연성 있는 접근법으로 사용 가능

분산 시스템과 응용 시스템 아키텍처

2 응용 시스템 아키텍처

1 애플리케이션 유형

데이터 처리 시스템	과금, 급여 시스템
트랜잭션 처리 시스템	전자 상거래, 예약 시스템
이벤트 처리 시스템	워드프로세서, 실시간 시스템
언어처리 시스템	컴파일러, 인터프리터

2 데이터 처리 시스템

1 정의

이용된 데이터베이스가 소프트웨어 자체보다도
훨씬 더 규모가 큰 데이터 중심 시스템

2 일괄처리 입력과 출력

입력

- 고객 번호 및 이와 관련된 전력 사용량 집합

출력

- 각 고객 번호에 대응하는 청구서 집합

분산 시스템과 응용 시스템 아키텍처

2 응용 시스템 아키텍처

2 데이터 처리 시스템

3 구조

Input	파일, 데이터베이스로부터 데이터를 읽고 데이터의 유효성을 점검 및 처리 후 데이터 큐로 내보냄
Process	큐로부터 데이터를 가져와 계산 수행, 계산 결과와 함께 새로운 레코드를 생성
Output	레코드들을 읽고 출력 형식에 따라 서식 처리, 데이터베이스에 기록

3 트랜잭션 처리 시스템

1 정의

데이터베이스에 관한 사용자 요구나 데이터베이스 갱신 요구를 처리

2 사용자 관점에서 트랜잭션

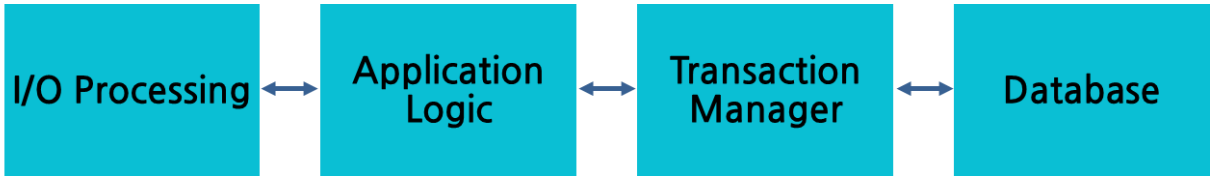
목표를 만족하는 일련의 응집성이 있는 오퍼레이션

분산 시스템과 응용 시스템 아키텍처

2 응용 시스템 아키텍처

3 트랜잭션 처리 시스템

3 구조



4 이벤트 처리 시스템

1 정의

시스템 환경의 이벤트에 반응함

2 특성

이벤트 타이밍을 예측할 수 없으므로 아키텍처는 이것을 해결하도록 구성

워드프로세서, 게임 등과 같은 다수의 공통적인 시스템이 이벤트 처리 시스템

분산 시스템과 응용 시스템 아키텍처

2 응용 시스템 아키텍처

5 언어 처리 시스템

1 정의

자연어나 인공언어를 입력으로 받아들여 출력으로
다른 언어 표현을 생성

2 특성

인터프리터를 포함할 수도 있음

문제 해결의 쉬운 방법으로 알고리즘, 시스템 데이터를
표현하는 상황에 이용

학습정리

1. 아키텍처 개요

- 시스템을 구성하는 서브 시스템들을 식별하고 서브 시스템의 제어와 통신을 위한 프레임워크를 설정하는 설계 프로세스의 산출물을 기술한 것
- 시스템 설계 프로세스의 초기 단계에서 아키텍처를 설계하며 명세화 및 설계 프로세스 사이의 연결을 나타냄
- 프로젝트 참여자의 의사소통 수단으로 사용이 용이하며 시스템 분석에도 용이함
- 아키텍처의 품질의 속성은 개념적 무결성, 정확성과 안전성, 개발이 용이한 속성을 가짐
- 아키텍처는 요구사항을 분석하여 아키텍처를 분석하고 이를 설계하여 최종적으로 검증 및 승인의 절차를 통해 구축함
- 아키텍처 모델은 기능의 분할과 배치 형태의 데이터 중심형 모델, 클라이언트 서버 모델, 계층 모델, MVC 모델이 있고, 제어 관계의 형태로는 데이터 흐름 모델이 있음

학습정리

2. 분산 시스템과 응용 시스템 아키텍처

- 독립적으로 운영될 수 있는 컴퓨터 시스템들을 네트워크 기반으로 상호 연결하여 특정 작업을 공동으로 처리할 수 있게 만들어진 시스템
- 분산 시스템은 가격 대비 성능이 우수하며 분산된 자원을 공유할 수 있고 하나의 연산을 분산 시스템에 분산하여 처리할 수 있기 때문에 연산 속도가 향상됨
- 분산 시스템 모델
  - Minicomputer 모델, Workstation Server 모델, Processor Pool 모델
- 실시간 시스템에서는 마스터/슬레이브 아키텍처를 적용
- 클라이언트 서버 아키텍처 : 애플리케이션은 서버가 제공하는 서비스의 집합과 이 서비스를 이용하는 클라이언트 집합으로 모델링
- 분산 객체 아키텍처는 클라이언트와 서버 간의 차이를 없애고 동일하게 분신시킨 객체로 아키텍처를 구성
- 응용 시스템 아키텍처의 애플리케이션 유형
  - 데이터 처리 시스템, 트랜잭션 처리 시스템, 이벤트 처리 시스템, 언어처리 시스템