Chapter 12. 정리

1) 설계의 목표

2) 설계 완성에 필요한 설계모델의 종류및 특징

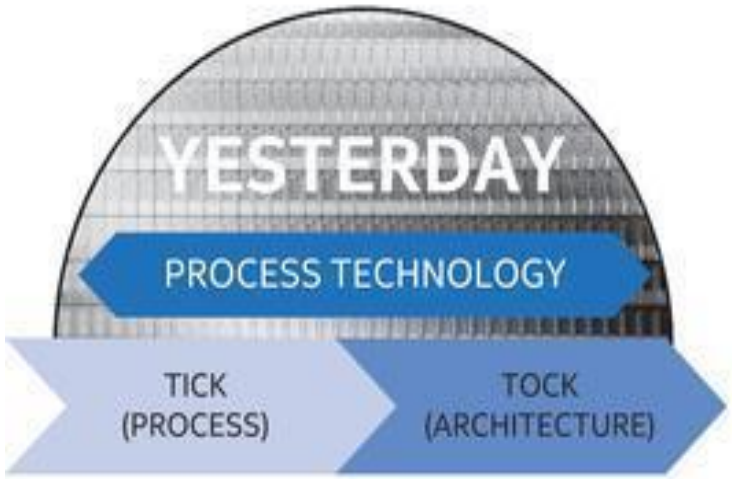
3) 분석 모델에서 설계 모델의 변화 관계

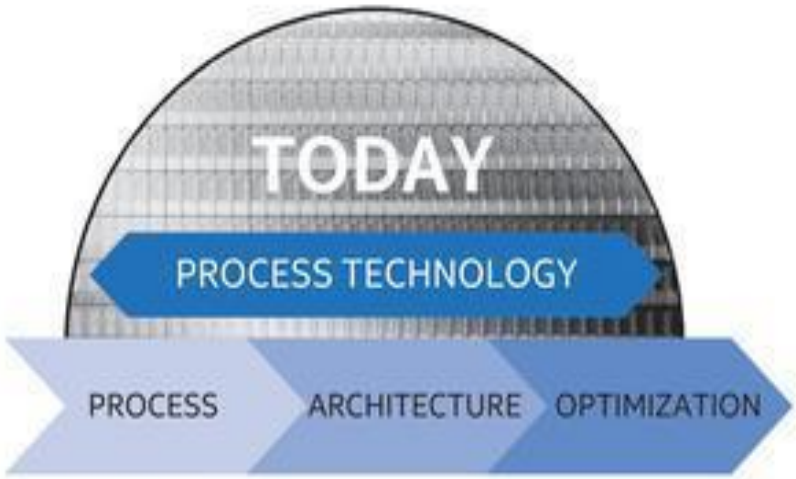
4) 좋은 설계 평가 와 품질 가이드라인

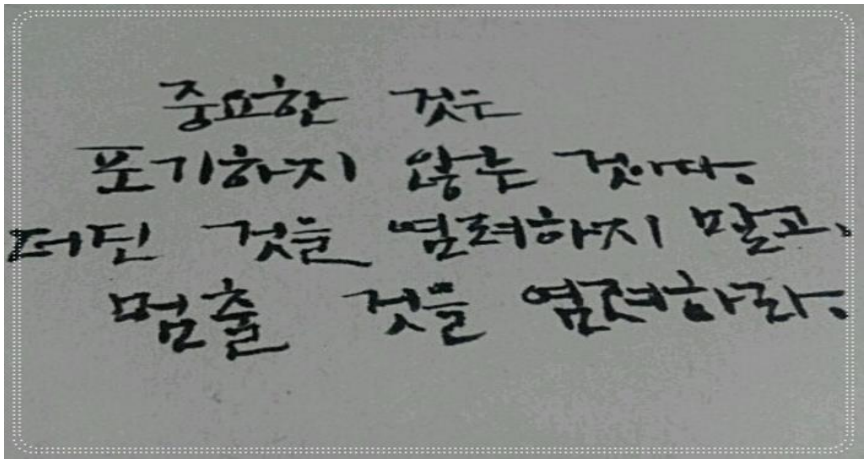
5) 설계 작업 프로세스

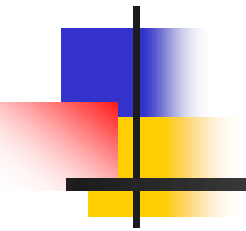
6) 설계 개념과 특징

최적화


YESTERDAY
PROCESS TECHNOLOGY
TICK
(PROCESS)
TOCK
(ARCHITECTURE)


TODAY
PROCESS TECHNOLOGY
PROCESS
ARCHITECTURE
OPTIMIZATION


중요한 것은
포기하지 않는 것이다.
버린 것을 염려하지 말고,
얻을 것을 염려하라.



제 13장. 아키텍처 개념

May. 2018
Young-gon, Kim
ykkim@kpu.ac.kr
Department of Computer Engineering
Korea Polytechnic University

Topics covered

- 소프트웨어 아키텍처
- 아키텍처 장르
- 아키텍처 스타일
- 아키텍처 고려사항
- 아키텍처 결정
- 아키텍처 설계
- 아키텍처 설계에 대한 평가

1.소프트웨어 아키텍처

◆ 건물 아키텍처

- 가장 단순한 수준에서는 **물리적 구조물**의 **전체적인 모양**
- 건물의 다양한 컴포넌트들을 통합하여

  응집력 있는 하나를 형성하는 방법
- 건물을 **환경에 부합**시키고 **주변의 다른 건물과**

  조화롭게 만들어주는 방법
- 건물이 갖는 **목적을 충족** 시키고 **소유자의 필요성을 만족**시키는 방법
- 구조물의 **심미적 느낌 , 건물의 시각적 영향 , 질감 , 색깔 ,**

  바닥재 종류 , 벽 부착물 배치.

1.소프트웨어 아키텍처

◆ 소프트웨어 아키텍처(설계는 아키텍처 인스턴스)

- **시스템 구조**를 **모델로 표현**하고, **데이터와 절차적 컴포넌트**들이 어떻게 **협력하는지**를 **모델로 표현**한 것
- 소프트웨어 **컴포넌트**, 이들 컴포넌트의 **외부적으로 보이는 속성** 및 이들 사이의 **관계**로 이루어진 **시스템의 구조**
- 아키텍처는 운용 소프트웨어가 아니라, 아래를 가능케 하는 표현
  - 기술한 **요구사항을 충족**시키는 데 있어서 **설계의 효과성을 분석**
  - **설계 변경**이 여전히 상대적으로 **용이한 단계**에서 **아키텍처 대안**을 고려
  - **소프트웨어 구축**에 관련된 **위험을 감소**.

1.소프트웨어 아키텍처

◆ 소프트웨어 아키텍처가 **중요한** 3 가지 **이유**

1) 소프트웨어 아키텍처는 **모든 이해당사자들 사이의 의사소통을 용이하게 하는 표현**을 제공

2) 아키텍처는 이어지는 **모든 소프트웨어 엔지니어링 작업에 중대한 영향을 미치는 초기 설계** 를 강조

3) 아키텍처는 " **시스템을 어떻게 구성**하고 그 **컴포넌트들이 함께 어떻게 작동**하는지에 관한 **모델을 쉽게 이해**할 수 있도록 제공".

1.소프트웨어 아키텍처

◆ 소프트웨어 아키텍처 용어를 이해하는 관점별 비유

- 언어 비유 : 이해당사자 집단에 걸친 **소통의 촉진자**
  - 높은 고객 초점을 가진 이해당사자 (관리자/마케팅전문가):**경계 결정시 협상기준**
- 문헌 비유 : **설계자와 소프트웨어 유지보수자**
  - **지식전달과 산출물 구축과 지원**
    - 과거에 구축된 **아키텍처 솔루션을 문서화** 하는데 사용
  - 컴포넌트와 설계를 **재사용** 하는데 관심
- 청사진 비유 : **프로그램 작성자**
  - 개발자 관점 : 아키텍트, 설계자, SW 엔지니어 (아키텍처 기술->**정보 전달 수단**)
- 결정 비유 : **프로젝트 관리자**
  - 아키텍처 : 비용 , 사용성 , 보수성 , 성능 같은 **속성의 결정의 산물.**

1.소프트웨어 아키텍처

◆ 아키텍처 결정

- 자체로 아키텍처의 한 **뷰로 간주**
  - **시스템구조와 이해당사자 관점에 부응**하기위한 나름의 **통찰력이 결정**을 내린 이유

◆ 아키텍처 결정 기술서 양식

1. 설계 문제
2. 해결책
3. 범주
4. 가정사항
5. 제한사항
6. 대안
7. 주장
8. 의미
9. 관련결정
10. 관련우려
11. 작업산출물
12. 유의사항

1. 다뤄야 할 아키텍처 설계를 기술
2. 설계를 다루기 위해 선택한 접근법 기술범주
3. 문제/해결책의 설계 범주 지정(데이터설계,컴포넌트구조)
4. 결정을 구체화 하는데 도움을 준 가정사항
5. 결정 구체화에 도움준 환경적 가정사항(기술표준, 패턴)
6. 고려한 아키텍처 설계 대안을 배제한 이유 기술
7. 다른 대안 중에서 이 해결책을 선택한 이유
8. 결정을 내리는 설계 결과(해결책이 영향도,설계방법 제한)
9. 이 결정에 관련된 다른 결정은 무엇인가 ?
10. 이 결정에 관련된 다른 요구사항은 무엇인가 ?
11. 이 결정을 반영해야 할 아키텍처 기술을 표시
12. 결정을 하는데 사용된 팁 유의사항/그 밖의 문서인용.

2.아키텍처 genres

◆ 아키텍처 장르

- 구축할 구조에 적용할 수 있는 **특정 아키텍처 접근법**
  - 아키텍처 설계의 기본 원칙 : **모든 유형**의 아키텍처에 적용 가능

- 아키텍처 **설계 장르**
  - 전체 소프트웨어 영역 안에서 **특정한 분류**
    - 건물 안에 세부 분류 : 주택 , 콘도 , 창고 , 건물(아파트, 사무용, 산업용)
  - 일반 스타일 안에 **좀 더 구체적인 스타일** 적용
    - 각 스타일 : 예측 가능한 패턴을 사용하여 기술할 수 있는 구조
  - 인공지능, 통신, 장치, 재무, 게임, 산업, 법률, 의료 구사, 운영체제, 운송, 유틸리티를 포함하는 **소프트웨어 기반 시스템**.

3.아키텍처 스타일

◆ 아키텍처 스타일 : 구축을 위한 **틀(Template)**

- 각 스타일의 **시스템 분류 기술**

1) 시스템이 필요하는 기능을 수행하는 **컴포넌트 집합**

(예; 데이터베이스, 계산모듈)

2) 컴포넌트 중에서 " **통신, 협조, 협력** "을 가능 케하는 **커넥터집합**

3) 컴포넌트를 통합하여 **시스템을 형성 할수 있는 방법을 정의**하는 **제한사항**

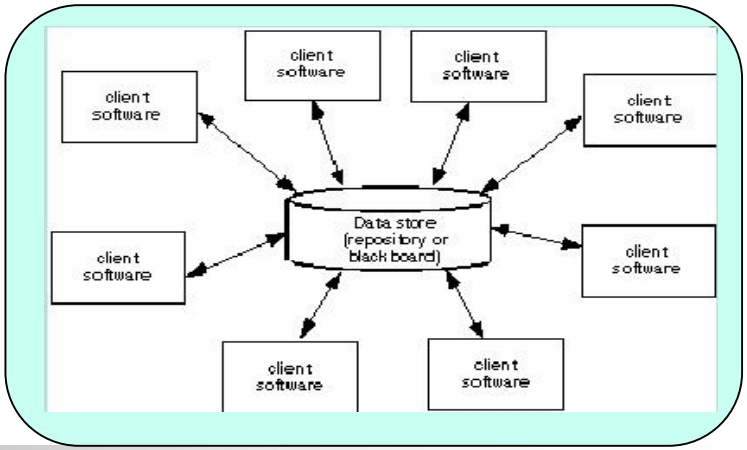
4) 설계자가 시스템 구성 부분의 알려진 속성을 분석하여

시스템의 전체적인 속성을 이해 하도록 해주는 **의미론적 모델.**

3.아키텍처 스타일

◆ 1) 데이터 중심 아키텍처

- **데이터 저장소**(파일, DB)가 **아키텍처 중심**을 차지
  - 다른 컴포넌트들은 데이터를 갱신 , 추가 , 삭제 , 수정하기 위해 저장소에 접근
- **클라이언트 소프트웨어**가 중앙저장소 접근
  - 데이터 저장소(수동적), 클라이언트 소프트웨어 (데이터 접근)
- 클라이언트 프로세스
  - **독립적**으로 수행
- 접근법 변형 : 저장소를 **"칠판"형태** 로 변환
  - 데이터 변경 발생시 -> 클라이언트 소프트웨어에 **알려주는 방식**
- **통합성 (Integrability)**을 강하게 함
  - 기존 컴포넌트 변경
  - 새로운 컴포넌트 아키텍처에 추가



3.아키텍처 스타일

◆2) 데이터 흐름 아키텍처

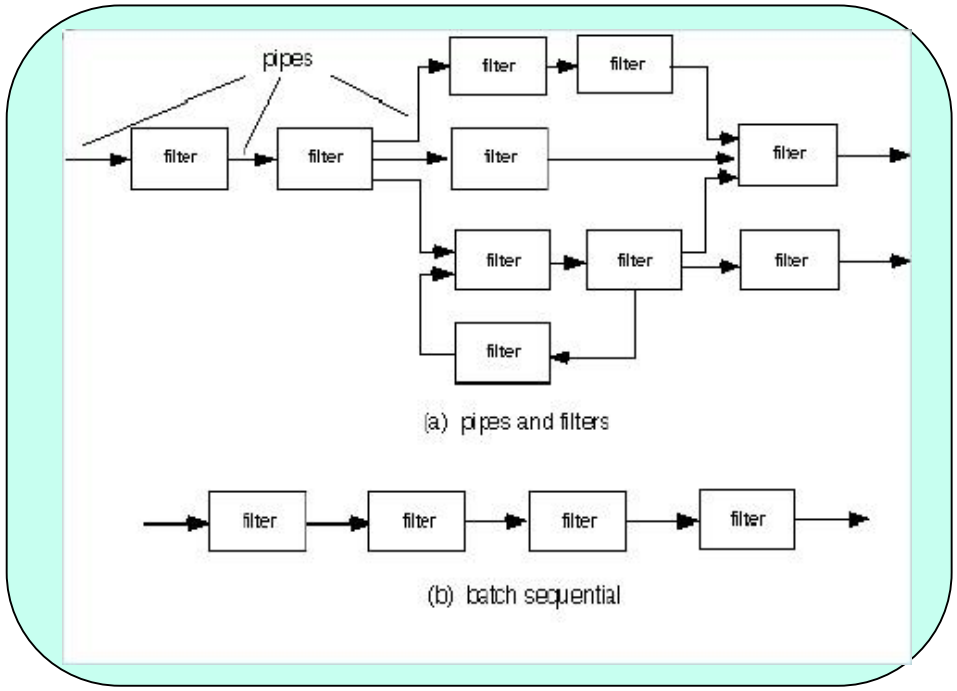
- 데이터가 일련의 계산/조작 컴포넌트 통해 **출력데이터 변경시 적용**
- **파이프 엔 필터** 패턴
  - **필터(컴포넌트)**: 특정 데이터 입력 받아, 특정 형태 **데이터 산출**토록 설계, **독립적**
  - **파이프** : 필터(컴포넌트)**간 데이터를 전송**
- **배치 순서**
  - 데이터 흐름이 **변형의 한 흐름**

    . **데이터 묶음**을 받아서

    일련의 연속 컴포넌트

    (필터)를 적용하여

    변형.



3.아키텍처 스타일

◆3) 호출 및 복귀아키텍처

- 상대적으로 **수정과 축소 및 확대가 용이**한 프로그램 구조를 가능
- **세부** 스타일

1) **주 프로그램/하부 프로그램** 아키텍처

- 구조는 기능을 **제어 계층** 으로 분할

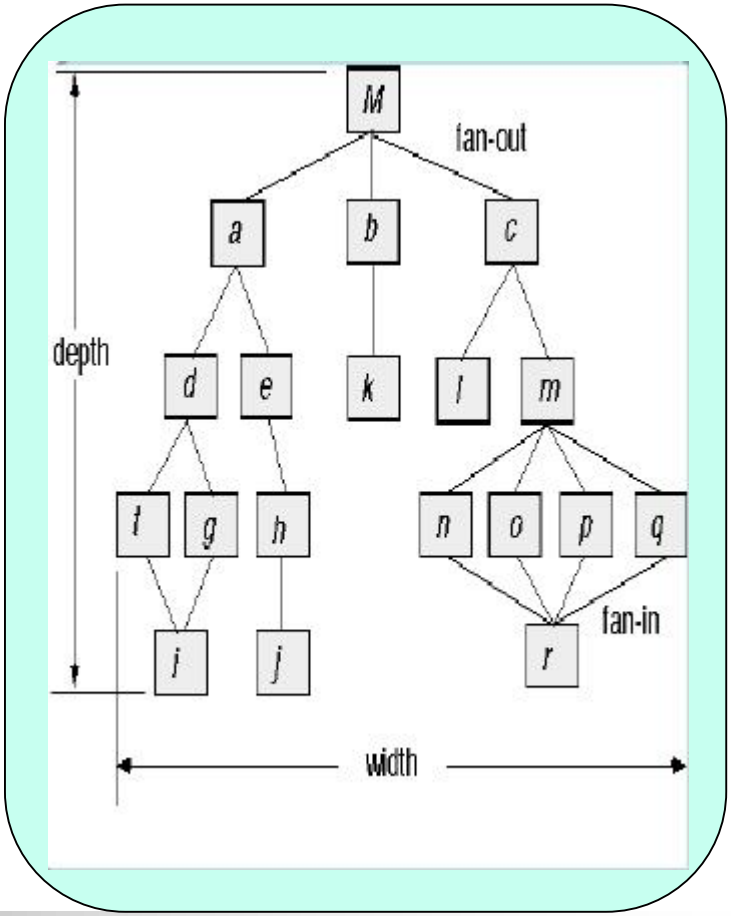
. **주 프로그램** : 다수 프로그램 컴포넌트 호출

. **프로그램 컴포넌트** : 다른 컴포넌트 호출

2) **원거리 프로시저 호출 아키텍처**

- 주 / 하부 프로그램 아키텍처구성요소

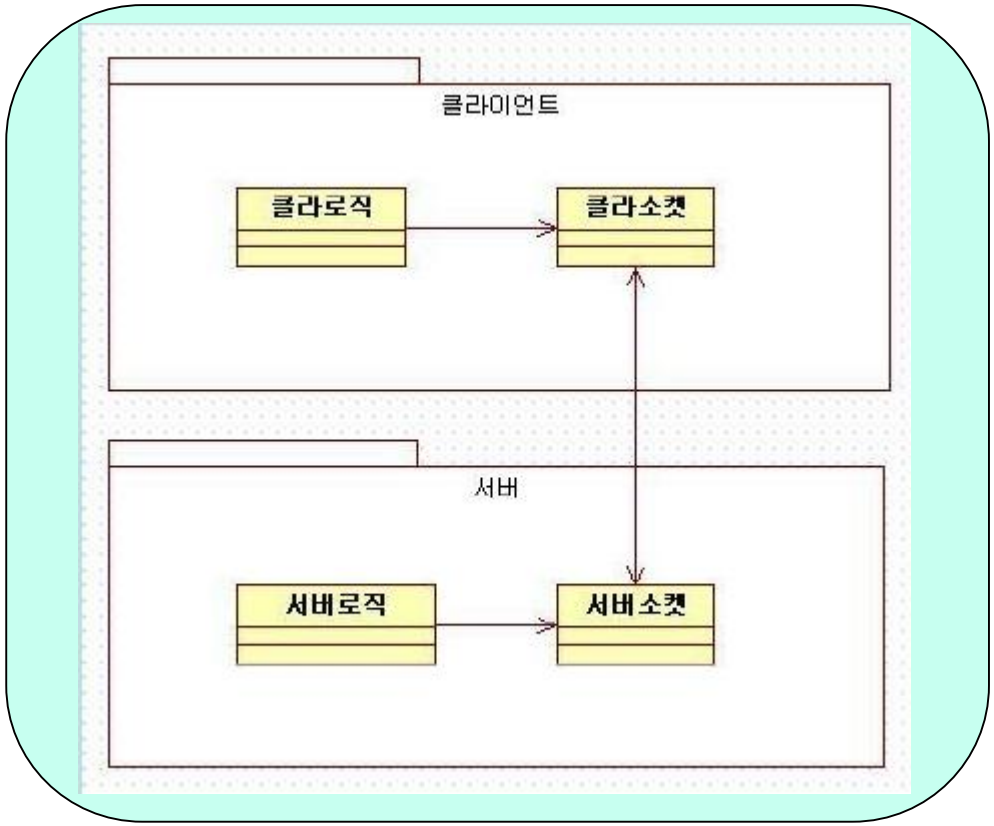
. 네트워크 상의 다수 컴퓨터에 분산.



3.아키텍처 스타일

◆ 4) 객체지향 아키텍처

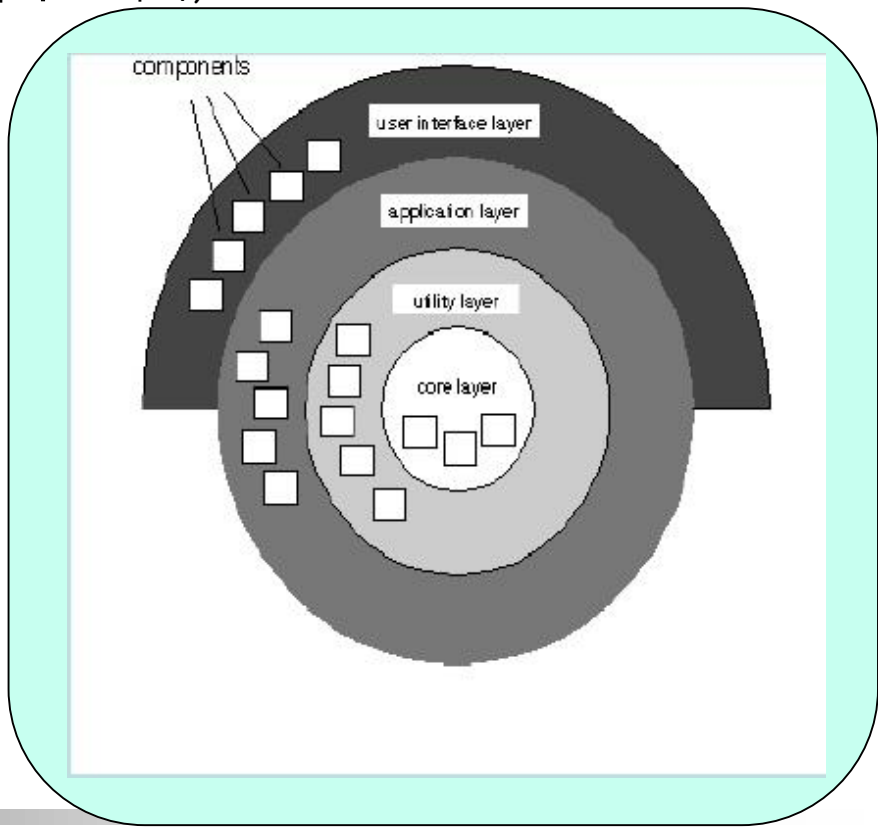
- 시스템 **컴포넌트** : **데이터**와 데이터 **조작 연산** 모두 포함
- 컴포넌트들 사이의 **교류와 협력** : **메시지** 전달 활용.



3.아키텍처 스타일

◆ 5) 층위구조 (Layered) 아키텍처

- 다수 층을 정의 , 각각은 점진적으로 기계어에 가까워지는 연산수행
- 계층
  - 사용자층 : 컴포넌트 (**사용자 인터페이스** 처리)
  - 중간층 : 유틸리티 서비스와 응용소프트웨어 기능 제공
  - 내부층 : **운영체제** 인터페이스.



3.아키텍처 스타일 :패턴

◆ 아키텍처 패턴

- **특정한 상황 , 특정 제한과 제약 하**에서 응용프로그램이 갖는 **고유한 문제**를 다룸
- 패턴
  - ➢ 아키텍처 설계의 기초적 역할을 할 수 있는 **아키텍처 해결방안** 제안.
- 응용프로그램 : **특정한 도메인** 또는 **장르**에 속하지만,
  해당 장르에 **하나 이상의 아키텍처 스타일**을 사용
  - ➢ 스타일 안에서 직면하는 **문제** : 특정한 아키텍처 **패턴을 이용하여 해결**.

3.아키텍처 스타일

◆ 아키텍처 스타일과 패턴 차이점 : 아키텍처 설계 변환 동일

- 패턴의 범위는 덜 광범위하므로, **아키텍처의 특정 측면**에 초점
  - **전체적**으로 아키텍처스타일
- 패턴은 **규칙을 제시** , 인프라 수준에서 **기능성의 일부 측면을 다루는 방법**을 소프트웨어가 어떻게 다루어야 하는지를 설명
  - 예) 동시성
- 패턴은 **아키텍처 상황 안에서 특정한 동작 측면**을 다루는 경향
  - 예) 실시간 응용프로그램이 동기화 또는 인터럽트 취급 방법.
- 패턴은 **문제와 컨텍츠, 해결책**에 초점.

3.아키텍처 스타일

◆ 아키텍처 스타일과 패턴

● 등장 배경

- ➢ 구현 단계가 아닌 분석/설계 단계에서의 생산성 향상 요구 증가
- ➢ 소프트웨어 재사용성과 개발 향상성의 효율화에 대한 대안 요구

● 차이점

	패턴	아키텍처 스타일
정의	- 자주 발생하는 설계상의 문제를 해결하기 위한 반복적인 해법 - 반복되며 유용한 구조 및 경험을 정리하여 일정한 형식으로 저장한 패턴	- 구조적 조직관점에서 문제점을 해결하고 제약성을 제시한 상세화 솔루션 - 컴포넌트와 커넥터 유형의 어휘를 정의하고 이들이 결합되는 방식에 의한 제약 조건들 정의
특징	- 구현 Class가 아닌 인터페이스 기준 Program - 상속 대신 위임 사용 - Coupling 최소화 - 빠른 설계와 유지보수가 용이하며 시스템 안정성 우수	- 중요한 아키텍처 설명 - 밀접한 설계 결정사항들의 패키지화 - 재사용성을 허용하는 알려진 속성 중심 - 데이터중심 (Data Flow, Data Centered),
관점	- 개발자(소프트웨어엔지니어)의 경험 중심	- 전이해관계자의 경험 중심

3.아키텍처 스타일 : 패턴

◆ 객체지향 설계에 관련된 **패턴**

카테고리	설계패턴	내용
창의적패턴	추상팩토리패턴	무슨요소를 인스턴스화할지의 결정을 중심으로
	팩토리유형패턴	다수구현중 하나를 선택하여 특정유형의 객체의생성중심으로
	빌터패턴	동일한 구성프로세스가 다른 표현을 생성토록 복잡한객체의생성을분리
구조패턴	어뎁터패턴	한인터페이스를 클라이언트가 기대하는인터페이스적응
	집단화패턴	자식클래스의 집단화를 위한방법을 가진 복합패턴의버전
	합성화파턴	모든객체가 동일한 인터페이스를 갖는 객체의 트리구조
	함유패턴	다른객체를 보유하고 이를 관리하기위한 목적으로 객체를 생성
	파이프및필터	각 프로세스의 출력이 다음의 입력이되는 프로세스체인
행동패턴	논리함유패턴	명령객체를 처리하거나 논리함유처리 객체가 다른 객체에 전달
	명령패턴	명령객체
	반복패턴	반복자는 기본형을 노출않고 순차적으로 집합객체요소를 검색
	조정자패턴	서브시스템에서 일련의 인터페이스에 통합된 인터페이스 제공
	방문자패턴	객체로부터 알고리즘을 분리하는 방법
	계층적방문자패턴	트리와 같은 계층적 데이터구조에서 각 노드를 방문하는 방법

3.아키텍처 스타일 : 패턴

◆ 아키텍처 패턴 **탬플릿**

1. 패턴 이름
2. 문제
3. 동기부여
4. 문맥
5. 힘
6. 해결책
7. 의도
8. 협력
9. 결과
10. 구현
11. 알려진 용도
12. 관련 패턴

1. 패턴의 **본질 설명** (짧지만 의미있게)
2. 패턴이 다루는 **문제 설명**
3. **문제의 예제**를 제공
4. 문제가 애플리케이션영역을 포함하여 **상주환경 설명**
5. 문제해결방식에 영향을 미치는 **힘(제약조건)**의 시스템 나열
6. 문제를 위해 제안된 **해결책의 상세한 설명** 제공
7. **패턴과 역할**이 무엇인지 설명
8. 어떻게 **다른 패턴들이 해결책에 기여**하는지 설명
9. 패턴구현시 고려하는 **잠재적인교환과 패턴이용결과** 설명
10. 패턴구현시 고려하는 **특별한 문제 식별**
11. 실제 애플리케이션에서 **설계 패턴의 실제적 사용 예** 제공
12. **상호 참조 관련 디자인 패턴**

3.아키텍처 스타일: 조직화와 정제

◆ 아키텍처의 좀 더 **상세한 분석**을 위한 **질문**

분류	질문
제어	아키텍처안에서 **제어를 어떻게 관리**하는가?
	명확한 제어계층이 존재하고, 제어계층안에서 **컴포넌트의 역할**은 무엇인가?
	컴포넌트는 시스템안에서 **제어를 어떻게 전달**하는가?
	컴포넌트들 사이에 어떻게 **제어를 공유**하는가?
	제어위상(통제작업이 취하는 기하학적형태)은 무엇인가?
	제어를 동기화하는가? 아니면 컴포넌트 **동작방식이 비동기**적인가?
데이터	컴포넌트 사이에 어떻게 **데이터를 교류**하는가?
	데이터흐름은 연속적인가? 아니면 데이터객체는 **산발적으로 시스템에 전달**되는가?
	데이터전송방식(하나의 컴포넌트에서 전달, 전역적공유)은 무엇인가
	데이터컴포넌트(칠판,저장소)가 존재한다면, 그**역할**은무엇인가?
	기본 컴포넌트들은 데이터컴포넌트들과 어떻게 **상호작용**하는가?
	데이터컴포넌트는 **수동적**인가, 아니면 **능동적**인가?(데이터컴포넌트와 다른컴포넌트)
	데이터와 제어는 시스템안에서 어떻게 **상호작용**하는가?

4.아키텍처 고려사항

◆ 아키텍처 결정해가는 과정에서 아키텍처 고려사항

- **경제성(economy)**
  - 불필요한 특징 / 비기능적 요구사항 -> **아키텍처** 불필요한 **복잡성**
  - 깔끔하고 불필요한 내용을 감소 -> **추상화 기술** => 가장 좋은 소프트웨어 구축
- **가시성(visibility)**
  - 설계모델 생성 -> 시험할 소프트웨어 엔지니어 들에게 **아키텍처 결정/이유 명확**
  - 미 가시화 경우 -> 설계후 모델을 시스템 구현할 사람 들에게 중요한 **설계 및 도메인 개념을 정확히 전달 불가능**
- **공간분할(spacing)**
  - **숨겨진 종속성 없이 관점을 분리**해 내는 것이 바람직한 설계 개념 -> 공간분할(Spacing)
  - 충분한 공간분할->모듈형 설계, 너무 세분하게 분할 -> **조각만 만들고 가시성 떨어짐**
  - **도메인 주도형** 설계 : 설계에서 **분리할 것** 과 **응집단위로 취급할 것**을 식별 도움
- **대칭성(symmetry)**
  - **아키텍처 대칭성** : 시스템이 그 **속성 안에서 일관되고 균형**을 이룬다는 것
    - 구조적/행위적으로 활용 가능
  - 대칭 설계 : **이해하고 종합하고 소통하기가 용이** .
- **시급성(emergence)**
  - **시급성,자기 조직화 행위와 통제** : **확장 가능** 하고, **효율적, 경제적** 아키텍처 생성
  - **실시간 소프트웨어 :시스템의 행위**를 정의하는 **이벤트의 순서와 지속기간**은 긴급한 품질 요소

5.아키텍처 결정

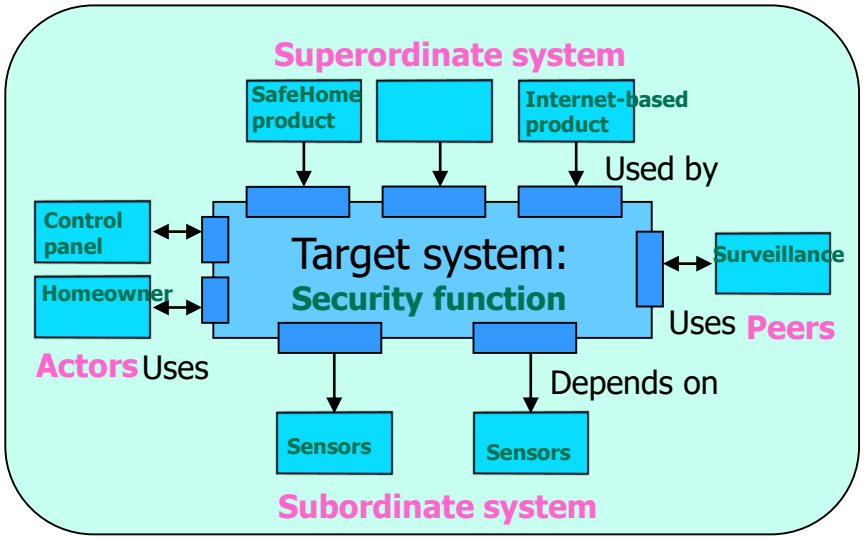
◆ 시스템아키텍처 결정

- 핵심 설계 문제와 아키텍처 해결안을 선택한 **이면의 근거를 파악**
- 결정 요소
  - 소프트웨어 **시스템 구성**, **구조적 요소** 와 이들간 **인터페이스**
  - 더 **큰 서브시스템으로 이들을 조립** 하는 것
  - **아키텍처 패턴, 적용기술, 미들웨어 자산, 프로그래밍 언어**
- 서비스 지향 아키텍처 결정(SOAD) **모델링 -> 향후 개발과정의 지침서**
  - 1) **지침 모델 (Guidance Model)**
    - 특정한 응용프로그램 **장르**에서 아키텍처 **스타일**을 적용할 때 필요한 **아키텍처 결정에 관한 지식** 포함
    - 아키텍처 결정이 내려야 하는 지점에 **잠재적 대안들 중 무엇을 선택** 해야 하는지 고민할 **품질속성을 문서화**
    - 아키텍처가 가능한 최선의 결정을 내릴 수 있도록 이전 소프트웨어 응용 프로그램에서 얻어진 **잠재적 대안 솔루션 포함**
  - 2) **결정모델 (Decision Model)**
    - 필요한 **아키텍처 결정을 문서화** 하고 이전 프로젝트에서 실제로 **이루어진 결정을 그 타당성과 함께 기록**.
    - 하나 이상의 **지침 모델을 활용**할 수 있고, 프로젝트 완료후 **지침모델에 피드백**.

6.아키텍처 설계

1. 시스템 배경도

- 아키텍처 배경도 (ACD : Architecture Context Diagram)
  - 소프트웨어가 그 **경계의 외부에 있는 개체들과 상호작용**하는 방법을 모형화
- 아키텍처 배경도의 구조 요소
  - **상위 시스템** : 목표 시스템을 **상위 처리방식**의 일부로 사용하는 시스템
  - **하위 시스템** : 목표 시스템 **사용/기능 완성**에 필요한 **데이터/처리하는 시스템**
  - **동료수준 시스템** : 동일수준에서 **상호작용**하는 시스템
  - **액터** : 처리에 정보생성/사용하는 **시스템과 상호작용하는 개체**(사람,장치)시스템
- 예) 주택 보안 기능
  - 상위시스템 : SafeHome 제어기, 인터넷기반시스템
  - 하위수준시스템 : 센서
  - 동료수준 : 감시기능
  - 액터 : 제어판 , 주택소유주



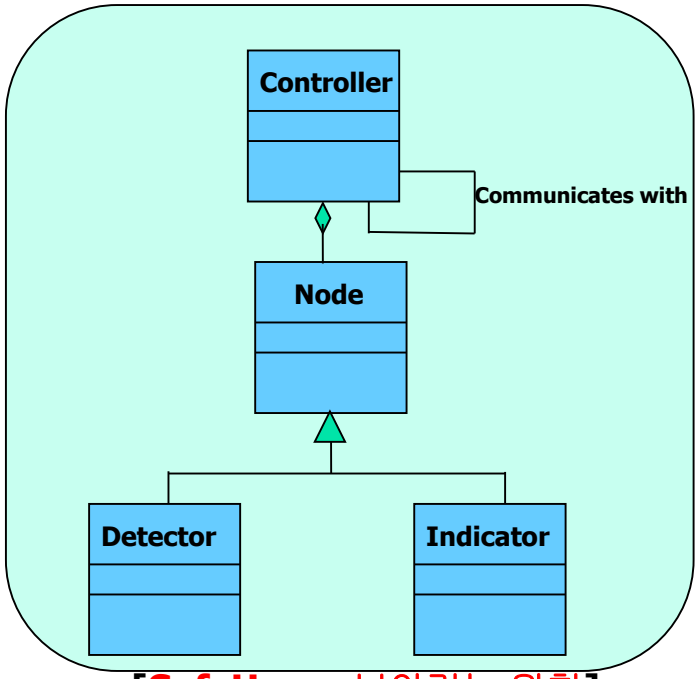
[아키텍처 배경도]

6.아키텍처 설계

2. 원형 (Archetype) 정의

- 목표 시스템을 위한 아키텍처 설계에 **필수적인 핵심추상화 표현하는 클래스/패턴**
- **요구사항 모델의 일부 : 원형 도출** <- **분석클래스를 조사**하여 가능
- 예) 주택 보안 기능으로 원형 정의

노드	보안기능의입력/출력요소의응집력 집합[다양한센서,경보(출력)표시]
탐지기	목표시스템에정보를투입하는모든 탐지장비들을포괄하는추상화
표시기	경보조건발생을표시하기위한모든 메커니즘(경보사이렌,점멸등,벨)을 추상화
제어기	노드의 무장 또는 무장해제를 허용하는 메커니즘을 도해하는 추상화



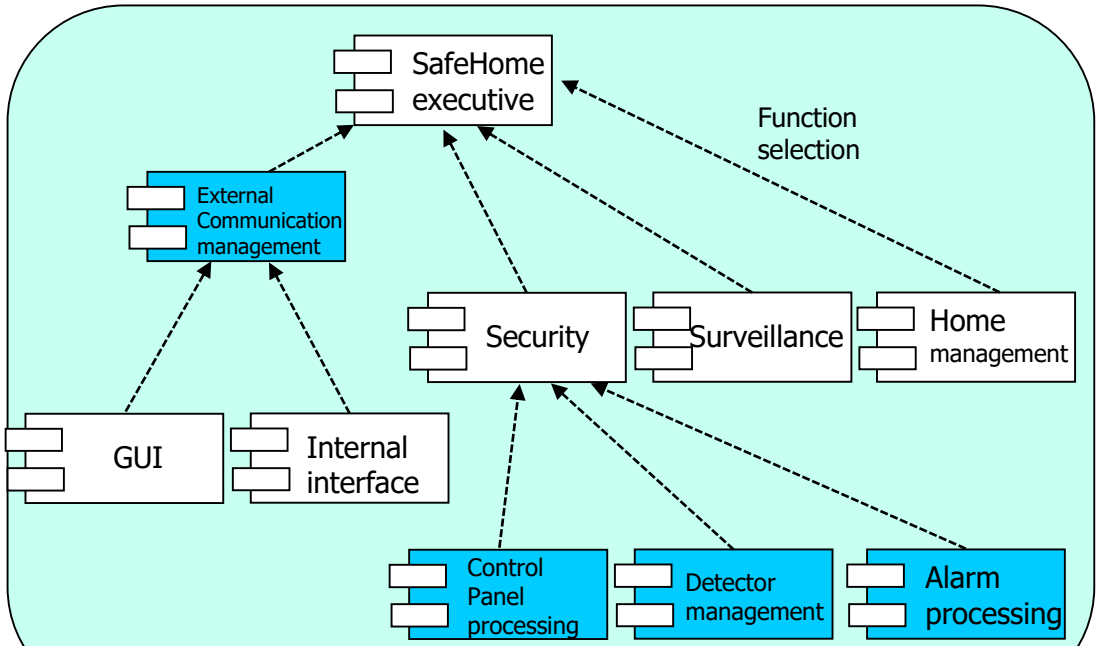
[SafeHome 보안기능 원형]

6.아키텍처 설계

◆3. 아키텍처를 컴포넌트로 정제 : 전체 아키텍처 구조

- 소프트웨어 아키텍처를 **컴포넌트로 정제** -> **시스템 구조** 보임
- **컴포넌트 선정**
  - 요구사항모델의 **클래스 이용**: **분석클래스**(응용 프로그램 도메인 안에 있는 **개체**)
- **예) 주택 보안 기능으로 상위 컴포넌트 세트 정의**

외부 통신 관리	다른 인터넷 기반 시스템 및 외부 경로 통지와 같은 외부 개체와의 보안기능 통신을 조정
제어판 처리	모든 제어판 기능을 관리
탐지기 관리	시스템에 부착된 모든 탐지기에 대한 접근을 조절
경보 처리	모든 경보조건을 확인하고 작용

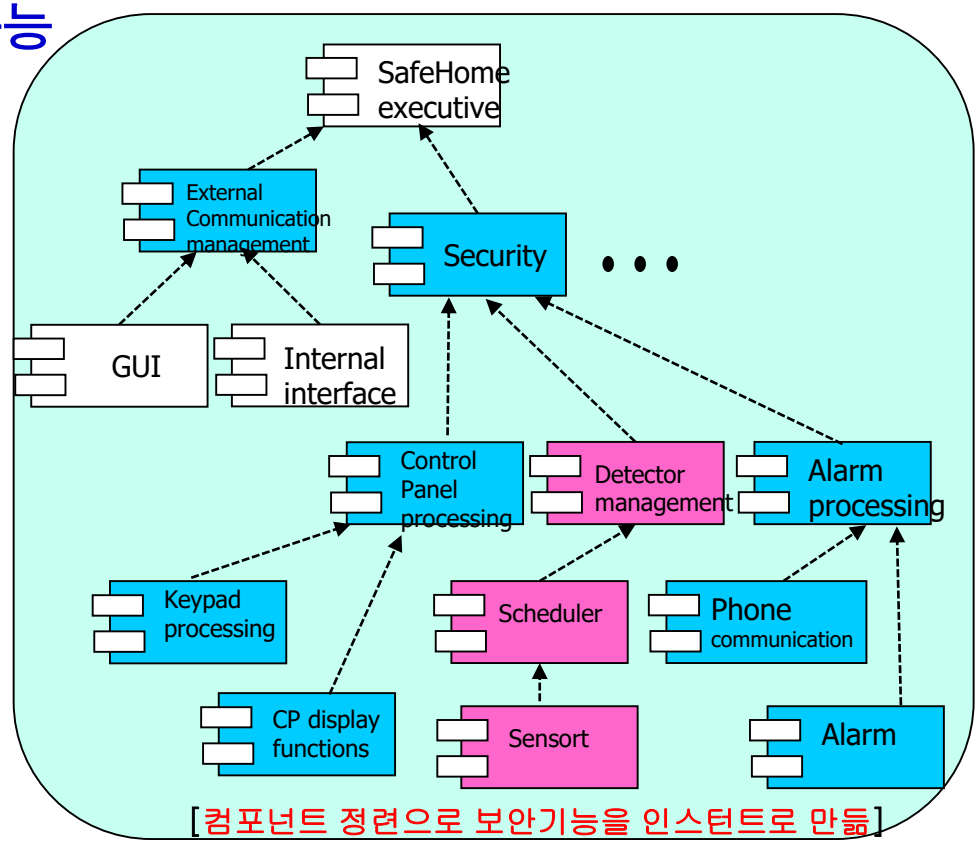


[최상위 수준 컴포넌트로 만든 SafeHome의 전반적 아키텍처 구조]

6.아키텍처 설계

◆ 4. 시스템 실현

- **추가적인 정제** 필요 -> 아키텍처 실현
- 아키텍처 실현 -> 아키텍처 구조 및 컴포넌트가 적절하다는 것을 **검증 및 특정한 문제 적용 가능**
- 예)주택보안 기능으로 아키텍처 실현
  - **탐지기 관리** 컴포넌트
    . 보안시스템이 사용하는 각 센서 객체를 주기적으로 조사한 기능 구현하는 스케줄러 인프라 컴포넌트와 상호작용.



7.아키텍처 설계에 대한 평가

◆ 아키텍처 상충분석 방법(ATAM: Architecture Tradeoff Analysis Method)

1) **시나리오 수립** : 사용자 관점의 유스케이스(시스템 표현)

2) **요구사항 , 제약 및 환경 설명을 도출**

- 요구공학 일부로 필요, 모든 이해당사자 관점에서 다루고 있는 확인 필요

3) 시나리오/요구사항을 다루기위해 선택한 **아키텍처 스타일/패턴 기술**

- **모듈**관점(업무/정보은닉), **프로세스**관점(성능), **테이터흐름**관점(기능요구사항)

4) 각 속성을 별도로 고려하여 **품질 속성을 평가 -> 품질속성 개수 결정**

- **품질속성** : 신뢰성, 성능, 보안성, 보수성, 유연성, 시험성, 이식성, 재사용성, 상호운영성

5) 특정한 아키텍처 스타일에 대한 **아키텍처 속성**에 대해 **품질 속성**의 **민감성 식별**

- **민감성** 지점 : 아키텍처 변동에 의하여 **중대한 영향**을 받는 속성

6) 5) 에서 실시한 민감성 분석을 사용하는 후보 아키텍처 (3) 에서 개발됨) 비판

- 아키텍처 민감성지점 결정-> **상쇄점을 찾는것**(다수의 속성들이 민감한 아키텍처 요소를 식별하는것) * C/S 아키텍처 의 성능-> **서버 대수(극도의 민감)->상쇄점**

정리 및 Homework

1) 소프트웨어 아키텍처가 중요한 이유
2) 소프트웨어 아키텍처 용어를 이해하는 관점별 비유
3) 아키텍처 스타일과 패턴의 차이
4) 아키텍처 스타일의 종류와 특징
5) 아키텍처 결정해가는 과정에서 고려사항
6) 아키텍처 설계 프로세스

Project

1장. 프로젝트 개요
- 1.1 프로젝트 제목
- 1.2 선정 이유
- 1.3 팀 운영 방법

2장 시스템 정의
- 2.1 시스템 간략한 설명
- 2.2 유사 사례 간략한 설명

3장 프로세스 모델
- 3.1 규범적인 프로세스 모델 선정 및 이유
- 3.2 특수한 프로세스 모델 선정 및 이유

4장. 실무 가이드 원칙
- 4.1 각 프레임워크 원칙에서 중요한 3 개 정의
- 4.2 프로젝트 계획 보고서

5장. 요구사항 획득
- 5.1 기능 요구사항과 비기능 요구사항 정의
- 5.2 표준 양식을 사용한 시스템 요구사항 명세 3개 작성
- 5.3 정형적인 형식에 따른 유스케이스 작성

6장. 시스템 설계
- 6.1 설계 개념의 중요한 개념을 적용
- 6.2 설계 모델에 따른 요소별 설계

Chapter 7. 아키텍처 개념
- 7.1 아키텍처 스타일 선정및 이유(개인별로 1개 선정/이유 -> 팀에서 최종 결정)
- 7.2 아키텍처 설계 프로세스 정의 및 설계(팀별로 1개의 프로세스 정의, 기능을 정의하고, 설계)