Architectual design(구조설계)

2020년 6월 13일 토요일 오전 2:02

Architectual design(구조설계) 비즈니스 요구사항->기술

Architecture=**시스템구성 컴포넌트, 컴포넌트간의 관계, 컴포넌트 다루는 정보를 정의**

아키텍쳐= 비즈니스 요구사항만족 하는 전체 시스템에 대한 구조 정의 문서

5.정의

아키텍처는 주요 시스템 구성 요소와 이들의 통신을 식별하는 것을 포함한다.

시스템 설계 초기단계, 구체화와 같이 진행될수 있음.

6.명시적 아키텍처의 장점

1)(**Stakeholder communication**)이해관계 당사자들간의 대화 원활

2)**System analysis** • 시스템이 비기능적 요건을 충족할 수 있는지 여부를 분석할 수 있음을 의미한다.

3)**Large-scale reuse** 재사용

7.아키텍쳐 특징

1.**perfomance**(성능) (분산할것이냐 vs 로컬라이즈할것이냐?)

스트레스 테스트, 피크 분석, 사용된 함수 빈도 분석, 필요한 용량 및 응답 시간이 포함됩니다.

2.**security**(데이터의 안전성): 내부 레이어에 중요 정보가 있는 레이어 아키텍처를 사용하십시오.

3.**safety** 소수의 하위 시스템에서 안전에 중요한 기능을 로컬라이즈해야함.

4.**Availability : toleance해야한다.**

5.maintainability :컴퍼넌트들이 대체 가능해야함

8.**충돌**

1)대형 그레이트 구성 요소를 사용하면 성능은 향상되지만 유지 보수성은 감소한다.

2)중복 데이터를 도입하면 가용성은 향상되지만 보안은 더욱 어려워진다.

3)안전 관련 기능의 국소화는 일반적으로 더 많은 통신을 의미하기 때문에 성능이 저하된다.

Safety<->perfomance

12 아키텍쳐 디자인 decision

1.아키텍처 설계는 독창적인 프로세스이므로 개발중인 시스템 유형에 따라 프로세스가 다릅니다.

2.일반적으로는 통용됨

13.결정하기위해 고려해야할점

1.사용할 수있는 일반적인 애플리케이션 아키텍처가 있습니까?

2.시스템은 어떻게 배포됩니까?

3.어떤 아키텍쳐 스타일이 적합합니까?

4.시스템을 구성하는 데 어떤 접근법이 사용됩니까?

5.시스템은 어떻게 모듈로 분해됩니까? 등등

14.재사용하면 유용

15.스타일: 아키텍쳐 모델의 스타일을 이용하면 조금더 쉽게해결가능함

16.모델

1)Static

2)Dynamic

3)interface

4)Relationship

5)Distribution

17.많이 사용되는 시스템 구성 유형

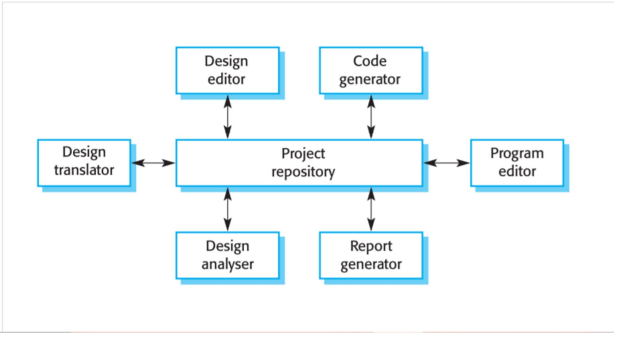
- 서비스를 요청(서버로부터) 서버클라이언트
- 공유 데이터의 repositorystyle
- 추상적인 유형

18 repository model

1) 각각의 서브시스템들은 각각의 데이터를 가지고 있고 공유함.
2) 데이터가 중앙에있고 서브시스템들에게 전달해줌

19. CASE toolset architecture!!!

그림 그릴수 있어야함 각각의 서브시스템들이 repository와 데이터를 주고받음



20.**repository model특징**

장점:

-만약에 없다면 다른 서브시스템과 직접 데이터를 주고받아야하는데 데이터를 효율적으로 주고받을수있게된다.

-백업과 보안에대해서 생각하지않아도됨

-공유모델들도 이미 만들어져있다.

단점:

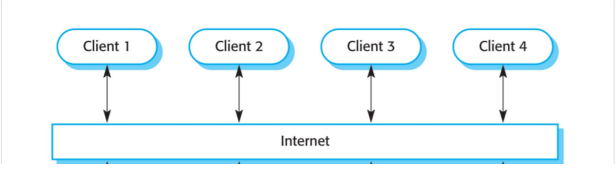
-정해진 형식대로 데이터를 주고 받아야한다. -(번거롭고 복잡한 형식 거칠수 있다)

-데이터의 evolution이 어렵다. 전체적인 시스템을 고려해야해서

---

21 Client-Server 모델

22 Client들이 인터넷망을통해 다양한서비스를 요청하고 서버가 서비스들을 제공해줌 네트워크를 통해 제공됨.



23 Client-server

장점

-데이터의 전송이 적접적이다.

-저렴하게 하드웨어 이용

-업데이트가 용이함

장점
-데이터의 전송이 적접적이다.
-저렴하게 하드웨어 이용
-업데이트가 용이함
단점
-공유 데이터 모델이 없으므로 하위 시스템은 다른 데이터구조를 사용합니다. 데이터 교환이 비효율적 일 수 있습니다.
-각 서버의 중복관리
-이름과 서버스의 중앙등록장치가 없습니다. 이용가능한 서버와 서비스를 찾기어려울수있음

24.Absract machine model(층모델)
Sub- interfacing하는 모델사용
Layers안에 system을 조직하여 서비스를 제공함
층에따라 sub-system의 점층적인 개발지원하여 인터페이스 변경되면 인접layer만 영향받음

25 **Version management system**

_____Modular decomposition
서브시스템에서 모듈로 분해하는 유형

27 Sub-systems and modules 차이
Sub system은 다른 subsystem들과 독립적이고 그만의 권리가있고 서비스를 주고받는다.
Module은 시스템 컴포넌트인데 다른컴포넌트간의 서비스를 주고받지만
독립된 시스템이라고 보기는힘들다.

28 서브 시스템이 모듈로 분해되는 또 다른 구조 레벨.
두가지 분해 모델들
Object models 객체 단위로 상호작용함
객체 지향적 분해는 객체의 클래스, 특징과 연산자들을 식별하는것 관련되있음
구현될 때 객체는 이러한 클래스에서 생성되며 일부 제어 모델은 객체 연산을 조정하는데 사용된다.
장점: 재사용가능, 다른 객체에 영향 안끼치고 변경가능
단점:그러나 객체 인터페이스 변경은 문제를 일으킬 수 있으며 복잡한 실체는 객체로 표현하기 어려울 수 있다.

파이프라인-dataflow 기능적모듈로 분해 input to output
입력에따른 산출문 만들어내기 위한형태(함수)
데이터 처리 시스템에 광범위하게 사용되는 일괄 처리 순차 모델
대화형 시스템에는 적합하지 않음.

장점: 변형 재사용가능,
이해 관계 당사자들에게 설명하기 쉬움
새로운 변형추가하기 쉽다
Sequential system에 적용하기 단순하다.
단점: event based interaction은 힘듬
데이터 전송에 common한 format만 사용가능

35 Control styles(시스템분해모델과 분리된다.)
하위 시스템 간의 제어 흐름과 관련됨

Centralised control
한 서브시스템이 다른 서브시스템들의 제어와 관련된 전체책임이있다.
Centralised control종류
Call-return model : subroutine의 Top down 형식으로 구조적으로 이동
순차시스템으로 적용가능함.
Manager model

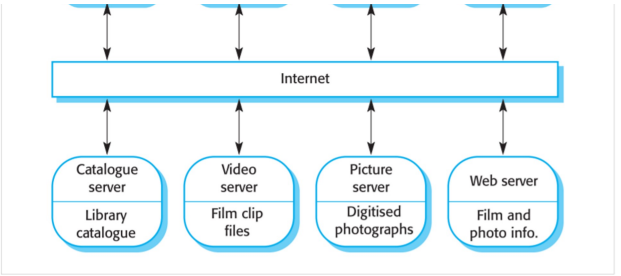
(병렬시스템)concurrent systems 적용가능	Realtime system-control

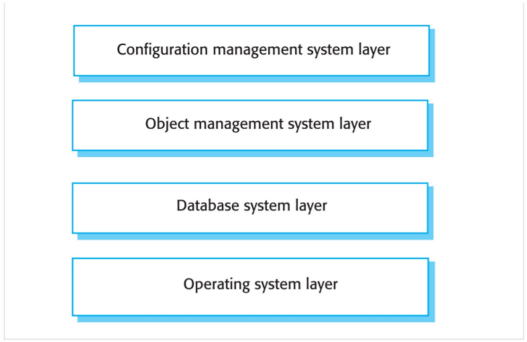
한 시스템component가 멈추거나 시작함을 조정
Case에따른 순차시스템으로 적용가능하다.

Event-based control(Event-driven systems)
각각의 서브시스템들이 다른 서브시스템 또는 시스템의 환경에서 발생한
외부에서 발생한 이벤트들에 반응한다.

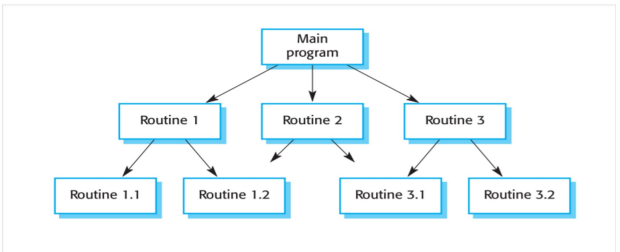
Broadcast models :모든 서브시스템에 브로드캐스트함
Integrating(통합에 효과적이다)
특징:모든 서브시스템은 이벤트처리가능
하위 시스템은 특정 이벤트에 대한 관심을 등록한다.
상황이 발생하면 제어장치가 이벤트를 처리할 수 있는 서브시스템으로 전달된다.
Control policy가 없다. 서브시스템에서 결정한다.
그러나 서브시스템은 언제 이벤트를 처리해야하는지 수 없다.

Interrupt-driven models : interurupts 처리기 로부터 interrupts가 탐지되면 Real timesystem에 사용된다.
빠른반응을 해야하는 real-time system에 필요하다.
Interrupt의 유형에따라 handler의 역할이 나누어져있다.
유형은 메모리위치와 연관되어있다.
빠른반응을 허가하지만 프로그램짜기 복잡하고 유효성 검증이힘듬.





Call-return model



Real-time system control