플랫폼의 기능

1. 소프트웨어 운영 환경 제공

2. 소프트웨어 개발 및 운영 비용 감소

3. 소프트웨어 개발 생산성 향상

4. 동일 플랫폼간 네트워크 효과 유발

플랫폼 성능 측정 주요 지표

1. 응답시간 2. 업무량/ 처리량 3. 가용성 4. 사용률

데이터베이스의 기능

1. 데이터 저장과 개발 및 유지보수 측면에서 중복성 통제

2. 다중 사용자 간의 데이터 공유

3. 권한 없는 사용자의 데이터 접근 통제

4. 다양한 사용자에게 다양한 형태의 인터페이스 제공

5. 데이터 사이에 존재하는 복잡한 관련성 표현

6. 데이터베이스의 무결성 보장

7. 백업과 복구 기능 제공

트랜잭션 특성

1. Atomicity(원자성) : 한가지라도 실패할 경우 전체가 취소되어 무결성 보장

2. Consistency(일관성) : 실행 성공 후 항상 모순 없이 일관성 있는 DB 상태 보존

3. Isolation(고립성) : 현재 수행 중인 트랜잭션이 완료될 때까지 트랜잭션이

생성한 중간 연산 결과에 다른 트랜잭션들이 접근할 수 없음을 의미

4. Durability(영속성) : 성공이 완료된 트랜잭션의 결과는 영구적으로 데이터베이스에 저장됨.

가상화 기능 분석

가상화 : 프로세서, 메모리, 스토리지, 네트워크를 포함하여, 이들로 구성된 서버나 장치들을 가상화 함으로써 높은 수준의 자원 사용율과 분산 처리능력을 제공한다.

클라우드 컴퓨팅 서비스 유형

IaaS : 가상머신환경 제공

PaaS : 자원 및 개발도구 제공

SaaS : 클라우드 상에서 소프트웨어와 데이터베이스 제공

요구공학 :

시스템 요구사항 문서를 생성, 검증, 관리하기 위하여 수행되는 구조화된 활동의 집합을 의미한다.

개발절차

타당성조사 – 도출 – 분석 – 명세 – 확인/검증

- 요구사항 분석

: 도출된 요구사항들 간의 상충을 해결, 소프트웨어가 환경과 어떻게 상호작용하는지 분석

1. 구조적 분석 방법론

가. 자료 흐름도(DFD)의 구성요소

Process(프로세스) : 자료의 처리/변환 과정, Data Flow(자료흐름) : 자료의 흐름.

Data Store(자료저장소) : 파일, 데이터 베이스 등 저장소의 위치

Terminator(단말) : 자료의 출처와 도착지

나. 자료 사전(DD)

= : 자료의 정의, + : 자료의 연결, () : 자료의 생략,

[ | ] : 자료의 선택, {} : 자료의 반복, \*\* : 자료의 설명

다. 상태전이도의 구성요소

시스템의 상태 : 시스템의 수행중인 상태

상태 변화 : 시스템이 어떤 상태에서 다른 상태로 변환하는 과정

조건과 행동 : 조건과 상태를 변화시킬 때 시스템이 취하는 행동을 명시

2. 유즈케이스(Use Case) 기반 방법론

가. 유즈케이스 다이어그램의 구성요소

Actor(행위자) : 시스템과 상호작용하는 사람 또는 사물

Use Case : Actor가 시스템을 통한 일련의 행위

Relation(관계) : Actor와 Use Case간의 관계

- 연관 : 상호작용 관계

- 확장 : 특정조건을 만족하면 수행 - - - ->

- 포함 : 유즈케이스를 실행하기 위해 반드시 포함해서 실행되어야 할 유즈케이스 - - - ->

- 일반화 : 유사한 유즈케이스들이나 액터들을 모아 추상화

- 그룹화 : 여러 개의 유즈 케이스를 단순화 (package)

System : 전체 시스템의 영역

- 요구사항 명세 : 도출된 요구사항을 분석하여 정의하는 단계

1. 요구사항 명세서(SRS)

가. 기능적 요구사항 : 시스템에서 제공되어야 할 특정기능

데이터 모델, 데이터 흐름 처리 모델, 프로세스 모델

나. 비 기능적 요구사항 : 시스템의 전체적 품질이나 기능적 요구사항의 구현 시 고려해야 하는 제약사항

S/W뿐만아니라 시스템 전체에 대한 요구사항, 성능, 보안, 아키텍처, 안정성

다. 요구사항 검토 방법

동료 검토 : 작성자가 요구사항 명세서를 설명하고 이해관계자들이 설명을 들으면서 결함을 발견하는 형태

워크 스루 : 소프트웨어 시스템 개발 단계마다 실시하는 비정형 검토 회의

인스펙션 : 소프트웨어 요구, 설계, 원시코드 등의 저작자 외의 다른전문가/ 팀이 검사하여 오류를 찾는 공식적 검토 방법

- 요구사항 확인/검증 : 사용자의 요구가 정확하게 요구사항 명세서에 기술되었는지 검토, 베이스라인으로 설정하는 활동

가. 요구사항 관리 절차

요구사항 협상 : 가능한 자원과 수용 가능한 위험 수준에서 구현 가능한 기능을 협상하기 위한 기법

요구사항 기준선 : 공식적으로 검토되고 합의된 요구사항 명세서

요구사항 변경 관리 : 요구사항 기준선을 기반으로 모든 변경을 공식적으로 통제하기 위한 기법

요구사항 확인 : 구축된 시스템이 이해 관계자가 기대한 요구사항에 부합되는지 확인하기 위한 방법

요구사항 관리도구의 주요기능

1. 프로젝트 생성, 2. 요구사항 작성, 3. 요구사항 등록/추출 4. 요구사항 이력관리 5. 요구사항 베이스라인

6. 요구사항 추적성 7. 협업 환경 8. 외부 연동 환경 9. 확장성

요구사항 추적 매트릭스 작성 방법 : 요구사항 및 업무내용을 도출할 해당 업무명을 기재

1. 요구사항ID 2. 요구사항 3. 해결방안 4. 출처 5. 분석 6. 설계 7. 개발 8. 테스트

요구사항 분석 도구/모델

1. CASE 도구 : 구조화된 요구 사항 명세서에 대해서는 자동화된 일관성 분석을 제공

-Upper CASE : 계획수립, 요구분석, 기본설계 단계를 지원하고 이를 다이어그램으로 표현

-Middle CASE : 새 설계 작업을 지원하며 화면, 출력등의 작성을 지원

-Lower CASE : 시험,. 유지보수 작업 지원하며 소스코드와 시스템 명세서 획득

-I-CASE : 위의 세가지를 통합, Rational ROSE, COOL 등이 국내에서 사용 중임.

HIPO 모델 : 하향식 소프트웨어 개발을 위한 문서화 도구이며 기능과 자료의의존관계를 동시에 표현해 보고 이해가 쉽다.

1. 가시적 도표: 전체적인 기능과 흐름 2. 총체적 도표: 전반적 정보 3. 세부적 도표: 기능을 구성하는 기본 요소를 상세히

소프트웨어 개발 방법론

1. 애자일 : 변화에 대한 신속한 대응으로 요구사항을 지속적으로 분석, 반영하여 배포 시간차를 최소화하는 개발 방법론

- 소프트웨어 개발 방법에 있어서 아무런 계획이 없는 개발 방법과 계획이 지나치게 많은 개발 방법들 사이에 타협점을 찾고자 하는 방법론이며 계획이 없는 방법론의 경우, 앞으로의 일을 예측하기 힘들고 효율적이지 못하다는 것이고, 계획에 너무 의존하는 경우는 그 형식적인 절차를 따르는데 필요한 시간과 비용을 무시할 수 없으며, 전체적인 개발의 흐름 자체를 느리게 하는 단점을 가지고 있다.

- 에자일 선언문

1. 개인과 상호작용, 2. 작동하는 소프트웨어 3. 고객과의 협상, 4. 변화에 대응하기를 가치 있게 여긴다.

- 에자일의 종류

XP(eXtreme Programming), SCRUM(스크럼), Lean(린), Kanban(칸반)

XP의 5가지 가치 : 의사소통, 단순함, 피드백, 용기, 종류

Scrum의 특징 : 협업중심, Sprint 수행, Daily Meeting, 프로젝트 관리 강조, 포괄적 정의 및 포용

Lean 개발방법 7가지 원칙(낭배늦권빠통전)

낭비의 제거, 배움증폭, 늦은결정, 팀에 권한위임, 빠른 납품, 통합성 구축, 전체를 볼 것

Kanban의 3가지 규칙(가윕웤)

위크플로우 가시화, WIP 제한, Workflow 측정 및 최적화

- 에자일과 / 전통 방법론의 비교

요구사항의 베이스라인 : 프로젝트 전과정에 걸쳐 진화 / 초기 요구사항에 베이스라인을 설정

아키텍처 정의 방법 : 실제 기능 구현을 통해 빠른 시간 내에 아키텍쳐의 실현 가능성을 증명

/ 모델과 사양을 상세화하는 과정을 통해 어플리케이션과 아키텍쳐를 초기에 정의

테스트 방법 : 시간과 비용이 투입되기 전에 기능을 검증 / 구현된 이후 점차 확장해 나가는 방식

표준 프로세스 적용 : 유연하게 개발(주기적 조정) / 표준화된 프로세스 재정이 중요(정형화, 상세화)

릴리즈 : 빠른 릴리즈 / BigBang릴리즈

중심사상 : 학습중심 / 계획중심

분석모델 확인

모델링의 기법

1. 모델링 : 실세계 물리현상을 특정한 목적에 맞추어 이용하기 쉬운 형식으로 표현하는 일

- 개념 모델링의 역할

문제 도메인의 엔티티들과 그들의 관계 및 종속성을 반영한다.

종류 : 유즈케이스 다이어그램, 데이터 흐름 모델, 상태 모델, 목표기반 모델, 사용자 인터액션, 객체모델, 데이터모델

표기법 : UML

2. UML : 산출물을 명세화, 시각화, 문서화할 때 사용되는 표준화된 범용 모델링 언어

- 특징

가시화언어 : 오류가 적고 의사소통을 용이하게 한다.

구축언어 : 다양한 객체지향 프로그램 언어로 변환가능

문서화언어 : 시스템에 대한 평가, 통제, 의사소통 문서

명세화언어 : 단순 표기법이 아닌 구현에 필요한 개발적 요소 및 기능에 대한 명세를 제공

-구성요소

View : 모델화된 시스템의 서로 다른 모형 제공

Diagram : View의 내용을 나타내기 위한 9가지 다이어그램

모델요소 : 객체지향개념을 표현하기 위해 사용되는 요소(클래스, 속성, 오퍼레이션)

General Mechanism : 모델 요소에 대하여 주석 정보와 의미 제공

- UML다이어그램의 종류

구조적 다이어그램

시스템의 정적구조와 다양한 추상화 및 구현수준에서 시스템의 구성요소들 간의 관계를 보여준다.

Class : 시스템 내 클래스들의 정적 구조, 객체들의 집합으로 속성과 동작으로 구성

Object : 클래스의 여러 Object인스턴스를 나타냄.(실제 클래스를 사용)

Component : 코드의 물리적 구조를 표현, 클래스 구현에 대한 정보를 포함, 실질적프로그래밍에 사용

Deployment : 하드웨어와 소프트웨어 간의 물리적 구조, 실질적 컴퓨터와 Device간의 관계를 표현

Package(UML 2.0) : 시스템 계층적인 구조를 표현

Composite Structure(UML 2.0) : 전체 클래스 안에 각 컴포넌트 클래스 표현, 클래스 내부 구조 파악에 용이

행위 다이어그램

시스템 내의 객체들의 동적인 행위를 보여주며, 시간의 변화에 따른 시스템의 연속된 변경을 설명해준다.

Use Case : 사용자의 입장에서 본 시스템의 행동을 표현, 시스템의 기능적인 요구

State : 클래스의 객체가 가질 수 있는 모든 가능한 상태와 상태간의 전이를 표현

Activity : 행위(Activity)의 순서적 흐름을 표현

상호작용 다이어그램

Sequence : 객체와 객체가의 상호작용을 메시지 흐름으로 표현

Communication : 상호작용에 참여하는 객체/컴포넌트 간의 관계를 명시적으로 표현

Interation Overview(UML 2.0) : 상호작용에 대한 제어흐름을 표현

Timing(UML 2.0) : 시간적 제약과 객체상태 변화를 표현

가. 클래스 다이어그램 :

객체의 멤버, 연산(메서드)의 구성과 객체들과의 연관, 의존, 상속 등의 정적인 관계를 나타내는 UML 다이어그램

- 클래스의 구성요소

클래스명(Class Name) : 공통의 속성, 연산, 관계, 의미를 객체들의 집합으로 정의

속성(Attribute) : 클래스의 속성, 접근지정자, 데이터타입 등 구조적 특징을 표현

연산(Operation) : 메서드라고도 하며 클래스에서 정의한 동작을 호출하는 연산자, 접근지정자, 리터타입, 파라메터를 표현

- 클래스 다이어그램 표기법

구성요소/ Name : Class Name, Interface, Attribute : 짧은 명사나 명사구로 이름을 붙인다.

Operation(Method) : 객체 행동에 영향을 주기 위해 특정 Class의 객체로부터 요청할 수 있는 서비스를 표현

표기법/ 접근제한자 :

‘\_’ : Private : 해당 클래스 내에서 ‘#’ : 동일 패키지 내, ‘+’ : Public : 어디서든

Attribute : 의미있는 명사형으로 표시, Opreation : 행위를 나타내는 동사형으로 표시

- 클래스간 관계

일반화관계(Generalizaition)(IS-A) : 객체지향개념에서 상속관계, UML에서는 일반화 관계로 모델링

연관관계(Association)(HAS-A) : 클래스들이 상호 메시지를 주고 받는 관계를 표현

의존관계(Dependency)(use) : 한클래스가 다른 클래스에 의존적인 연관관계를 나타날 때 사용

집합관계(Aggregation) : 클래스들 사이의 전체와 부분의 관계를 나타냄, 하나의 클래스가 다른 클래스를 포함하는 관계

합성관계(Composition) : ‘’ , 전체 객체의 라이프타임과 부분 객체의 라이프타임이 동일하며,

전체 객체가 없어지면 부분 객체도 없어짐.

실체화관계(Realization)(implement) : 인터페이스 클래스를 다른 클래스가 구현해주는 관계, 구체적으로 실체화

나. 시퀀스 다이어그램 :

문제해결에 필요한 객체를 정의하고 객체 간 주고받는 메서드의 순서를 시간의 흐름에 따라 보여주는 다이어그램.