소프트웨어공학 워드정리

2014152019 심지섭

3장. 소프트웨어 프로세스 구조

3.1 소프트웨어공학의 프로세스 흐름

액티비티 중의 액션과 태스크들이 순서와 시간적 측면에서 구성

Flow중심 : 선형 ->반복 ->진화 ->병행

3.2 소프트웨어공학 프로세스 패턴

소프트웨어공학 작업을 수행하는 동안 1.발생되는 프로세스 관련 문제 묘사

2.문제가 발생하는 환경 식별

3.문제에 대한 하나 또는 그 이상의 증명된 해결방안 제시

템플릿: 소프트웨어 프로세스관점에서 문제 해결방안을 묘사 하는 일관된 방법

1. 패턴이름 : 프로세스 관점에서 패턴을 묘사는 이름

2. 효과 : 패턴이 필요한 환경 , 문제를 확실하고 해결방안에 미치는 결과

3. 유형 : 패턴유형 [스테이지(액티비티), 태스크(액션/작업 태스크), 페이즈(액티비티 순서) 패턴 ]

4. 초기상황 : 패턴이 적용될 조건

5. 문제 : 패턴에 의해 해결되어야 하는 특수한 문제

6. 해결방안 : 패턴을 어떻게 성공적으로 구현할 것인지 기술

7. 결과상황 : 패턴이 성공적으로 구현되면 생길 조건들을 기술

8. 관련 패턴 : 이 패턴에 직접적으로 관련 있는 모든 프로세스 패턴 리스트

9. 용도와 예 : 패턴이 적용 가능한 특수한 사례 .

4장. 프로세스모델

4.1 프로세스정의

프로세스(활동순서의 집합): 의도한형태의 결과를 얻기 위한 활동 (activities), 제약조건(constraints), 그리고 자원 (resources)을 포함하는 일련의 과정.

4.2 프로세스가 중요한 이유

1.일련의 활동들에 대해서 일관성과 구조를 강요하기 때문

2.활동들의 이해, 통제, 실험, 개선 등을 도움

3.우리의 경험을 획득하여 다른 사람에게 전달가능.

4.3 프로세스특징

1.주요한 프로세스 활동 (major process activity) 을 규정

2.자원을 이용하고, 일정과 같은 제약조건을 지킴

3.중간 프로덕트와 최종 프로덕트를 생산함

4.서로 연결되어 있는 서브프로세스로 구성

1)각각의 서브프로세스가 고유의 프로세스모델을 갖도록 프로세스계층구조로 정의되거나 조직될 수 있음

6.진입과 출구기준 (entry and exit criteria) 을 가짐

1)각 프로세스의 활동은 활동의 시작과 끝을 알 수 있도록 구성되어 있음

7.활동은 순차적으로 구성되어 있음

1}하나의 활동이 다른 활동들과 관계를 가지고 수행됨을 명확히 하기위해

8.모든 프로세스는 각 활동의 목표를 설명하는 일련의 지침을 포함함

9.제약조건 또는 제어는 활동 , 자원 또는 프로덕트에 적용되기도 함.

4.4 프로세스모델링을 하는 이유

1.개발 프로세스 기술 시 공통된 이해 형식

1)소프트웨어 개발에 포함된 활동 , 자원 , 제약조건 등

2.불필요한 내용 및 생략된 부분을 찾는데 도움을 줌

1)개발팀 간에 존재하는 일관적이지 못하거나 불필요한 내용 ,

2)프로세스와 프로세스구성요소에서 생략된 것들을 찾는 경우

3.고품질의 소프트웨어 개발 , 오류의 조기발견 그리고 예산과 일정제약조건과 같은 개발목표를 반영함

1)모델이 완성되면 개발팀은 목표를 수행하기에 적절한 후보 활동들을 평가함

4.모든 프로세스는 특정한 상황에 따라 적절히 대응할 수 있음

1)프로세스모델의 구축은 유동적인 상황이 일어날 수 있음을 이해하는데 도움이 됨.

4.5 각 프로세스모델의 장단점 및 활용

1.폭포수모델

1.장점

1)다양한 소프트웨어 개발 활동을 기술 하는데 이용됨

2)개발자들에게 필요한 행동이 무엇인지를 계획 하는데 있어 매우 유용함

3)단순하므로 소프트웨어 개발에 능숙하지 못한 고객들에게 쉽게 설명

할 수 있도록 해주며 , 다음 개발단계로 진행하기 위해 필요한 중간

프로덕트를 표시해 줌

2.단점

1)개발 동안 발생할 수 있는 프로덕트와 활동의 변화에 대처할 방법이 없음

2)소프트웨어 개발을 창조적 과정이 아닌 제조 과정으로 봄

3)최종 프로덕트를 만들어낼 반복적 활동이 없음

4)최종 프로덕트가 나올 때 까지 오래 기다림.

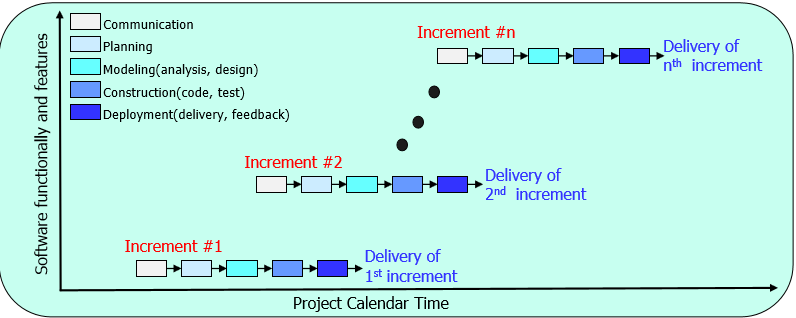
2.V-모델

폭포수 모델의 변형

1. 커뮤니케이션 , 모델링 , 초기 구축 액티비티와 관련된 액션 에 대한 품질보증 액션 의 관계
2. 왼쪽 : 문제의 기본적인 요구사항 을 점차적으로 정교화
3. 오른쪽 : 왼쪽을 진행하면서 생성한 각 모델을 입증 하는 테스트 수행.

장’단점: 대부분 폭포수모델과 비슷하나 각 모델을 입증하는 테스트를 수행함으로써 단점을 보완함

3.Increment-모델



A.사용자 교육이 조기 에 이루어짐

1. 몇 가지 기능이 누락되더라도 훈련 (training) 은 초기 릴리스 에서 시작됨
2. 훈련 프로세스는 개발자로 하여금 특정 기능이 어떻게 수행되는지 관찰 하도록 하고 다음 릴리스에서 보강하도록 함
3. 개발자들은 사용자에게 책임 을 다할 수 있음

B.지금까지 제공된 적이 없는 기능 만이 초기 시장을 개척 가능

C.빈번한 릴리스는 운영중인 시스템에서 보고되는 문제점을 개발자가 신속하고 전체적으로 고칠 수 있도록 함

D.개발 팀은 릴리스별로 다른 영역의 전문성에 중점 을 둘 수 있음

4.진화적 모델

1)프로토타입 ( 반복적인 특성 ) + 폭포수모델 ( 통제적 , 체계적인 특성 )

2)특성

A.리스크를 감소 하면서 시스템의 정의와 구현 정도를 증가시키는 순환

B.적합하고 상호 만족스러운 해결방안을 이해관계자에게 보장하기

앵커 포인트 이정표

3)진화적 릴리스의 형태

A.초반 반복 과정에서 얻는 릴리스모델이나 프로토 타입

B.후반 반복 과정에서 얻는 릴리스

- 점증적으로 완전한 버전

공학적인 프로덕트.

5.동시모델

1)프로세스 모델의 반복과 동시적인 요소

2)모든 액티비티는 동시에 존재하지만 , 서로 다른 상태

- 상태 변환 : 이벤트

3)액티비티 , 액션 , 태스크를 연속적으로 한정하기보다

- 프로세스와 네트워크로 정의.

7장. 모델링-실무가이드원칙

7.1 프로세스를 가이드 하는 원칙정의

1.애자일 하라

1)업무 (경제적인 행동), 기술적인 접근(단순,작업산출물 : 간결, 부분적인 결정)

2.모든 단계에서 품질에 초점을 두어라

1)프로세스 활동, 행위, 작업 출구 조건 : 작업 산출물의 품질에 기준

3.적응할 준비를 하라

1)프로세스 : 문제, 인력, 프로젝트의 제약조건에 조정

4.효율적인 팀을 꾸려라

1)인력(소프트웨어 개발 핵심), 자체 조직화 팀 구축(상호 신뢰와 존경)

5.의사소통과 조정을 위하여 매커니즘을 설립하라

1)중요 정보소실, 이해관계자들 최종산출물 생성 노력 실패 :프로젝트 실패 이유

6.변경을 관리하라

1)메커니즘 수립 : 변경, 요청, 평가, 승인, 수립 되는 방법 관리

7.리스크를 평가하라

1)개발 과정에서 많은 잘못 : 긴급 대책 수립이 필수적

8.다른 사람을 위하여 가치를 제공하는 산출물을 생성하라.

1)산출물:프로세스 활동,행위,작업,가치제공->모호함/누락 없이,필요한정보 포함.

7.2 실무를 가이드 하는 원칙

1.분할하여 공략 하라

1)분석과 설계: 관심사 분리 (SoCs), 큰문제를 작은 요소로 분해->해결이 쉬움

2.추상화의 사용을 이해하라

1)추상화(의사소통 단계 사용) -> 복잡한 문제를 단순화, 상세한 의사소통 제거 3.일관성을 유지하라

1)소프트웨어 개발 단계에서 일관성 유지 -> 익숙한 문맥으로 SW 사용 용이

4.정보의 전송에 초점을 맞춰라

`1)정보 : 인터페이스를 통해 전달,그 결과로 에러,누락에 초점,모호성 가능성->테스트

5.효율적인 모듈성을 갖는 소프트웨어를 만들어라

1)모듈성 : 효율적, 배타적-> 높은 응집력, 낮은 결합도

6.패턴을 찾아라

1)패턴 : 문제점/솔루션->통찰력, 경험 소통 공동언어, 요구충족/ 좋은 구조 이해 7.가능하면 문제와 그 솔루션을 다양한 관점에서 나타내라

1)문제/해결책 다른 관점 평가:훌륭한 통찰력,에러/누락 발견->시나리오/클래스/동작 관점

8.소프트웨어를 유지할 누군가를 기억하라.

1)SW : 향상(결함 수정,환경변화에 적응,이해관계자 요구사항)->유지보수에 적용.

7.3 각 프레임워크를 가이드 하는 원칙

-의사소통/계획원칙

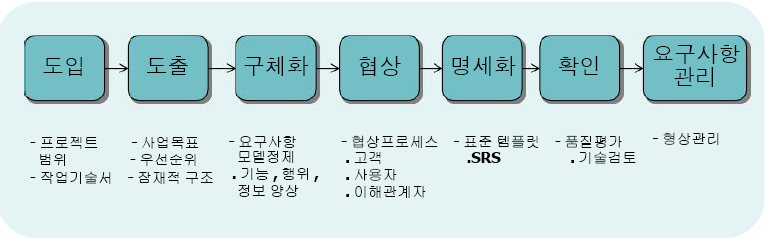
-요구사항/디자인모델링 원칙

-구축/배치원칙

7.4 프로젝트계획보고서내용

8장. 요구사항이해

8.1 요구공학프로세스

1.도입

1)문제의 기본이해 확립

이해관계자와 소프트웨어팀 간의 사전 의사소통과 협력의 효율성

2)솔루션을 원하는 사람

해결책을 원하는 솔루션의 본질

2.도출

1)사업목표수립

2)우선순위 결정 메커니즘 수립

3)의사소통과 조정을 위하여 메커니즘을 수립

4)잠재적인 구조에 대한 근거적인 설계

5)이해관계자 목표충족

6)도출을 위한 질문내용

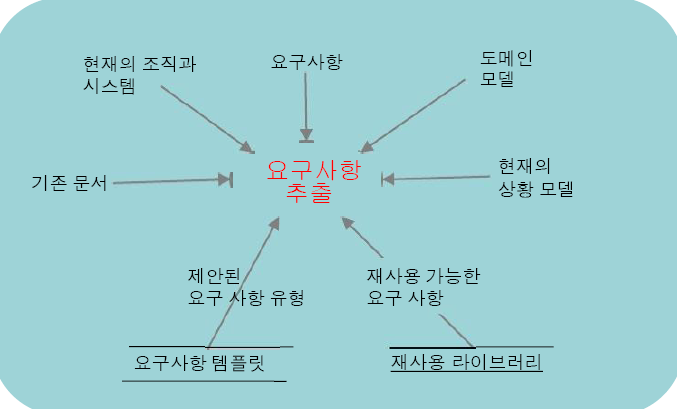
1.시스템/제품의 목표 무엇인지?

2.무엇이 달성되려 하는지?

3.시스템/제품의 비즈니스요구에 어떻게 적합한지?

4.시스템/제품이 어떻게 매일 사용될 수 있을지?

8.2 요구사항을 추출하기위한 가능한자원



3.구체화

1)요구사항모델 정제

소프트웨어기능,행위,정보의 다양한 양상 정의

2)구체화

최종사용자(액터)가 시스템 상호작용 기술

사용자 시나리오 생성과정제

사용자시나리오:분석클래스(도메인개체들)로 분해->속성정의

클래스간 협력정의->보조 다이어그램 생성

4.협상

1)충돌일반적

고객과 사용자들: 주어진 자원보다 달성하는 것 이상요구

다른 고객이나 사용자: 상충하는 요구사항 제안

버전이“우리의 특정요구에 필수적”이라고 주장

2)협상프로세스: 충돌절충

우선순위정의: 우선순위에 따라 충돌논의

요구사항의 위험도와 비용을 평가

요구사항제거/통합/수정->당사자간의 만족유도.

8.3 유스케이스 내용을 더 완벽한 관점을 제공하는 방법

1.주요 액터와 부수적 액터는 누구인가?

2.액터의 목적은 무엇인가?

3.이야기를 시작하기전 있어야하는 사전조건은 무엇인가?

4.액터에 의해 수행되어야 할 주요작업이나 기능은 무엇인가?

5.이야기가 기술되는 것에 의해 고려해야하는 예외사항은 무엇인가?

6.액터의 상호작용이 가능함에 따라 어떠한 변화가 있는가?

7.액터가 획득하고 생산하고, 변경하는 시스템정보는 무엇인가?

8.액터가 외부환경의 변경에 대해서 시스템에게 알려줄 것인가?

9.액터가 시스템으로부터 바라는 정보는 무엇인가?

10.액터는 예상치 못한 변경에 대해 알려줄 것 인가?

8.4 요구사항검증방법

1. 각 요구사항이 시스템 또는 제품을 위하여 전반적인 목표에 일치하는가?

2. 모든 요구사항이 적절한 추상화 레벨에서 명세 되었는가?(부적절한 상세화 레벨)

3. 요구사항이 진정으로 필요하거나 또는 시스템의 목적에 필수적이지 않을 수도 있는

추가적인 특징을 표현하는가?

1. 각 요구사항이 경계가 정해져 있고 명확한가?
2. 각 요구사항은 속성을 갖고있는가? 즉 소스(특정개인)는 각 요구사항에 대해서 잘 알려져 있는가?
3. 요구사항 중 어떤 요구사항이 다른 요구사항과 충돌되는가?
4. 각 요구사항이 시스템 또는 제품을 저장할 기술적 환경에서 달성 가능한가?
5. 각 요구사항은 일단 구현되면 ,시험가능한가?
6. 요구사항모델은 구축될 시스템의 정보, 기능, 그리고 행위를 적절히 반영하는가?
7. 요구사항모델이 시스템에 대한 보다 상세한 정보를 점진적으로 노출하는 방식으로“분할”되었는가?
8. 요구사항패턴이 요구사항모델을 단순화 하는데에 사용되었는가?
9. 모든 패턴이 적절히 검증되었는가?
10. 모든 패턴이 고객요구사항과 일치하는가?

8.5 좋은 요구사항 문서작성 전략

1.사용자요구사항과 시스템요구사항을 구분:비기술자위축

2.단순히 주요기능만을 다룸: 자율적인 개발보장

3.근거를 제시하려는 노력:변경 시 유용성 확인.

9장. 요구사항 모델링

1.요구공학 모델링활동의 모델의 종류 및 특징

2) 분석모델링 중 초점을 두는 point

1.사용자의 상호작용이 특정한 상황에서 발생 하는지

2.객체들이 시스템을 조작 하는지

3.기능이 시스템에서 반드시 수행 해야 하는지

4.동작을 시스템이 보여줘야 하는지

5.인터페이스가 정의 해야 하는지

6.제약조건이 적용 되는지.

3) 시나리오기반 모델링 프로세스

1.초기 유스케이스 제작

유스케이스: 정보의 제작자,고객, 시스템 자체에서 발생하는 상호작용 추출

2.초기 유스케이스 정제

초기 시나리오 평가 질문

1)액터가 이 시점에서 어떤 다른 활동을 받아들일 수 있는가 ?

2)액터가 이 시점에서 특정 오류 조건을 접할 가능성 이 있는가 ?

만약 그렇다면 , 그것을 무엇이라고 할 수 있겠는가 ?

3)액터가 이 시점에서 다른 동작을 접할 가능성 이 있는가 ?

만약 그렇다면 , 그것을 무엇이라고 할 수 있겠는가 ?

12장. 설계개념

1. 설계의 목표

견고함, 편리함 및 즐거움을 줄 수 있는 모델/표현을 만들어 내는 것.

1. 설계완성에 필요한 설계모델의 종류 및 특징

1) 클래스 모델

 SW 구축에 필요한 클래스 실현과 필수 데이터구조인 데이터/클래스 설계로 변형

 CRC 다이어그램 : 객체, 관계, 클래스 속성, 상세정보 -> 데이터 설계 활동 기반

 상세한 클래스 설계 : 각 소프트웨어 컴포넌트 설계 시 작성

2) 아키텍처 설계

 SW 주요요소사이 관계 , 시스템에 정의된 요구사항을 달성하는데 사용

아키텍처 스타일 과 아키텍처가 구현될 수 있는 방법에 영향을 주는 제약조건을 정의

 요구사항 모델로부터 도출 -> 아키텍처 설계 표현 ( 컴퓨터 기반 프레임 워크 )

3) 인터페이스 설계

 소프트웨어를 포함하는 시스템 및 이를 사용하는 사람과 소프트웨어가

상호작용 하는 방법을 기술

인터페이스 : 정보의 흐름 (데이터 / 제어)과 특정한 행동 유형

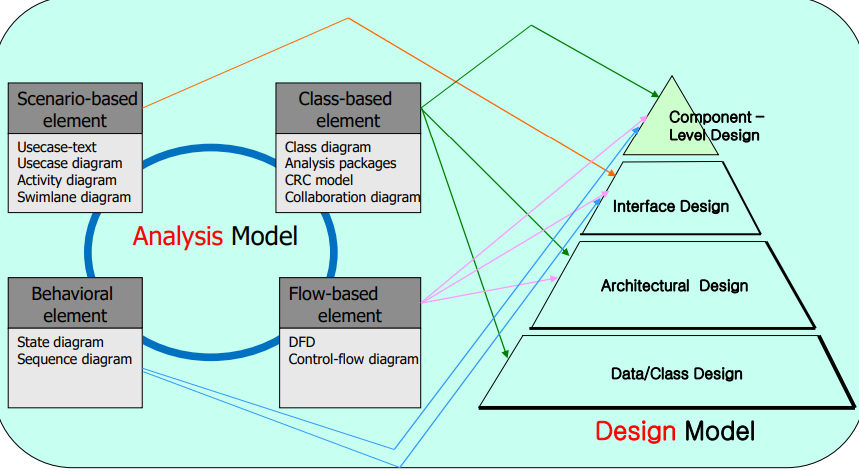
 사용 시나리오 모델 -> 인터페이스 설계에 필요한 많은 정보 제공

4) 컴포넌트 수준 설계

 소프트웨어의 구조적 요소 -> 소프트웨어 컴포넌트의 절차적 기술 로 변환

 클래스 기반 모델과 행동 모델로부터 얻은 정보 -> 컴포넌트 설계의 기반 .

1. 분석모델에서 설계모델의 변화관계



4) 좋은 설계평가와 품질가이드라인

좋은 설계 평가를 위한 가이드라인 -> 설계 프로세스 목표

1. 설계는 모든 요구사항을 수용

 요구사항 모델 : 포함된 명시적인 모든 요구사항을 구현

 이해당사자 : 원하는 암시적인 모든 요구사항을 수용

1. 설계는 읽기 쉽고 이해하기 쉬워야 함

 코드를 생성하는 사람

 소프트웨어를 시험하는 사람

 향후 소프트웨어 지원을 담당할 사람

1. 설계는 소프트웨어 전반적인 그림 제공
2. 설계는 구현 관점에서 데이터, 기능 및 행동 영역을 다뤄야 함.

설계 품질 가이드 라인 : 설계 프로세스의 실질적 목표

1. 설계는 3 가지 특성을 만족하는 아키텍처 생성

 잘 알려진 아키텍처 스타일 또는 패턴을 사용하여 생성

 좋은 설계 특성을 보이는 컴포넌트들로 구성

 구현과 시험이 가능하도록 혁신적인 방법으로 구현 : 작은 시스템 순차적 설계

1. 설계는 모듈화

 소프트웨어는 요소/하부시스템으로 논리적 분할

1. 설계는 데이터, 아키텍처, 인터페이스 및 컴포넌트 구분하여 표현
2. 설계는 자료구조를 생성

 자료구조는 구현할 클래스에 적절하고 잘 알려진 데이터 패턴으로 부터 도출

1. 설계는 독립적인 기능적 특성을 갖는 컴포넌트를 도출
2. 설계는 연결 복잡성을 줄이는 인터페이스 도출

 컴포넌트와 외부 환경간

1. 결정한 설계 기법을 반복적으로 적용하여 설계

 소프트웨어 요구사항 분석 중 얻은 정보

8. 설계는 의미를 효과적으로 전달하는 기호(notation) 사용하여 표현.

5) 설계작업프로세스

1. 데이터 구조 설계

 정보 도메인 모델을 조사하고 데이터 객체 및 그 속성

2. 아키텍처 스타일 ( 패턴 ) 선정

 분석 모델 을 사용하여 소프트웨어에 적합한 것

3. 분석 모델 / 분석 모델 요소를 하부시스템으로 할당

 하부시스템 ( 기능적으로 응집성 ), 하부시스템 ( 인터페이스 설계 )

 분석 클래스 또는 기능 을 각 하부시스템에 할당

4. 설계 클래스 또는 컴포넌트 생성

 분석 클래스 설명 ( 설계 클래스로 변환 ) : 클래스와 연관된 메소드와 메시지 정의

5. 외부 시스템 또는 장치와의 필요한 인터페이스 설계

6. 사용자 인터페이스 설계

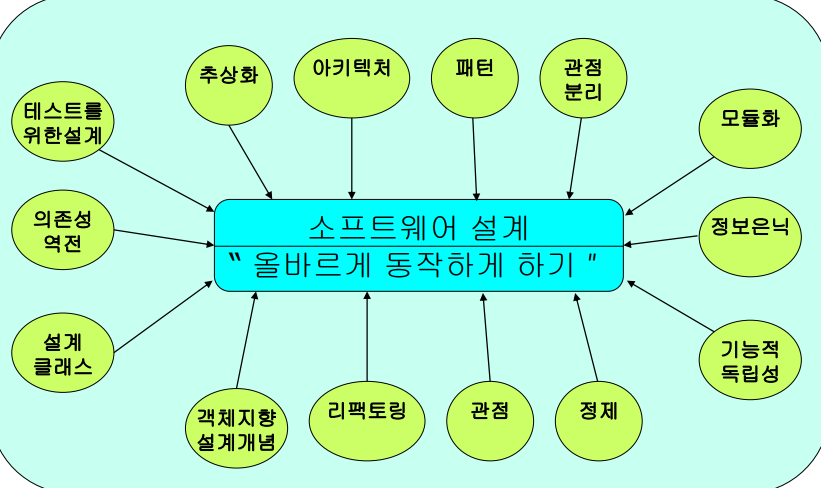
 인터페이스 행동 모델 생성 , 인터페이스 객체 / 제어 메커니즘 정의

7. 컴포넌트 수준 설계를 수행

 컴포넌트 수준 데이터 구조 정의

8. 배치 모델 개발.

6) 설계개념과 특징



13장. 아키텍쳐 개념

1. 소프트웨어 아키텍처가 중요한 이유

1.소프트웨어 아키텍처는 모든 사이의 의사소통을 용이하게 하는 표현을 제공

2.아키텍처는 이어지는 모든 소프트웨어 엔지니어링 작업에 중대한 영향을 미치는 초기 설계를 강조

3.아키텍처는“시스템을 어떻게 구성하고 그 컴포넌트들이 함께 어떻게 작동하는지에 관한 모델을쉽게 이해할 수 있도록 제공”.

1. 소프트웨어 아키텍처 용어를 이해하는 관점 별 비유

1.언어 비유 : 이해당사자 집단에 걸친 소통의 촉진자

높은 고객초점을 가진 이해당사자(관리자/마케팅전문가):경계결정시 협상기준

2.문헌 비유 : 설계자와 소프트웨어 유지보수자

지식전달과 산출물 구축과 지원

과거에 구축된 아키텍처 솔루션을 문서화하는데 사용

컴포넌트와 설계를 재사용하는데 관심

3.청사진 비유 : 프로그램작성자

개발자관점 : 아키텍트,설계자,SW엔지니어(아키텍처기술->정보전달수단)

4.결정비유:프로젝트 관리자

아키텍처:비용,사용성,보수성,성능 같은 속성의 결정의 산물.

1. 아키텍처스타일과 패턴의 차이

1.패턴의 범위는 덜 광범위하므로 ,아키텍처의 특정측면에 초점

전체적으로 아키텍처 스타일

2.패턴은 규칙을 제시, 인프라수준에서 기능성의 일부측면을 다루는 방법을 소프트웨어가 어떻게 다루어야 하는지를 설명

3.패턴은 아키텍처 상황 안에서 특정한 동작측면을 다루는 경향

4.패턴은 문제와 컨텐츠, 해결책에 초점.

1. 아키텍처스타일의 종류와 특징

1]데이터중심 아키텍처

1.데이터저장소(파일,DB)가 아키텍처 중심을 차지

다른 컴포넌트들은 데이터를 갱신,추가,삭제,수정 하기위해 저장소에 접근

2.클라이언트 소프트웨어가 중앙저장소 접근

데이터저장소(수동적),클라이언트 소프트웨어(데이터접근)

3.클라이언트프로세스

독립적으로 수행

4.접근법변형:저장소를“칠판”형태로 변환

데이터변경 발생 시->클라이언트 소프트웨어에 알려주는 방식

5.통합성(Integrability)을 강하게 함

기존 컴포넌트 변경

새로운 컴포넌트 아키텍처에 추가

2]데이터흐름 아키텍처

1.데이터가 일련의 계산/조작 컴포넌트 통해 출력데이터 변경 시 적용

2.파이프엔 필터 패턴

필터(컴포넌트 ): 특정데이터 입력 받아 ,특정형태데이터 산출토록 설계, 독립적

파이프:필터(컴포넌트)간 데이터를 전송

3.배치순서

데이터흐름이 변형의 한 흐름

데이터 묶음을 받아서 일련의 연속 컴포넌트(필터)를 적용하여 변형.

3]호출 및 복귀 아키텍처

1.상대적으로 수정과 축소 및 확대가용이한 프로그램 구조를 가능

2.세부스타일

1)주프로그램/하부프로그램 아키텍처

-구조는 기능을 제어계층으로 분할

. 주프로그램 : 다수프로그램 컴포넌트 호출

. 프로그램 컴포넌트 : 다른 컴포넌트 호출

2)원거리 프로시저 호출 아키텍처

-주/하부 프로그램 아키텍처 구성요소

. 네트워크상의 다수컴퓨터에 분산.

4]객체지향 아키텍처

1.시스템 컴포넌트 : 데이터와 데이터조작연산 모두 포함

2.컴포넌트들 사이의 교류와 협력 : 메시지전달 활용.

5]층위구조(Layered)아키텍처

1.다수층을 정의, 각각은 점진적으로 기계어에 가까워지는 연산수행

2.계층

사용자층:컴포넌트(사용자 인터페이스 처리)

중간층 : 유틸리티 서비스와

응용 소프트웨어 기능제공

내부층 : 운영체제 인터페이스.

1. 아키텍처 결정해가는 과정에서 고려사항

1.경제성(economy)

불필요한특징/비기능적요구사항->아키텍처 불필요한 복잡성

깔끔하고 불필요한 내용을 감소->추상화기술=>가장 좋은 소프트웨어 구축

2.가시성(visibility)

설계모델생성->시험할 소프트웨어 엔지니어들에게 아키텍처 결정/이유 명확

미 가시화 경우->설계 후 모델을 시스템 구현할 사람들에게 중요한 설계 및 도메인 개념을 정확히 전달 불가능

3.공간분할(spacing)

숨겨진 종속성 없이 관점을 분리해내는 것이 바람직한 설계개념-> 공간분할

충분한 공간분할->모듈형 설계, 너무 세분하게 분할->조각만 만들고 가시성 떨어짐

도메인 주도형 설계 : 설계에서 분리할 것 과 응집단위로 취급할 것을 식별 도움

4.대칭성(symmetry)

아키텍처 대칭성 : 시스템이 그 속성 안에서 일관되고 균형을 이룬다는 것

구조적/행위적으로 활용 가능

대칭설계:이해하고 종합하고 소통하기가 용이.

5.시급성(emergence)

시급성,자기조직화 행위와 통제:확장 가능 하고, 효율적, 경제적 아키텍처 생성

실시간 소프트웨어 : 시스템의 행위를 정의하는 이벤트의 순서와 지속기간은 긴급한 품질요소

1. 아키텍처설계프로세스

1.시스템배경도

아키텍처배경도(ACD:ArchitectureContextDiagram)

소프트웨어가 그 경계의 외부에 있는 개체들과 상호작용하는 방법을 모형화

2.원형(Archetype)정의

목표시스템을 위한 아키텍처설계에 필수적인 핵심 추상화 표현하는 클래스/패턴

요구사항 모델의 일부: 원형도출<-분석클래스를 조사하여 가능

3.아키텍처를 컴포넌트로 정제 : 전체 아키텍처 구조

1)소프트웨어 아키텍처를 컴포넌트로 정제->시스템 구조 보임

2)컴포넌트 선정

3)요구사항모델의 클래스 이용 : 분석클래스(응용 프로그램 도메인 안에 있는 개체)

4.시스템실현

1)추가적인 정제 필요-> 아키텍처 실현

2)아키텍처 실현-> 아키텍처 구조 및 컴포넌트가 적절하다는 것을 검증 및 특정한 문제 적용가능

19장. 품질

1. 품질 계수



2) 품질 치수

1. 성능 품질

 최종 사용자에게 가치를 제공하는 방법, 모든 콘텐츠/기능 제공하는가 ?

2.기능의 품질

 처음 최종사용자를 놀라게 하고 기쁘게 할 기능 제공하는가 ?

3. 신뢰성

 실패없이 모든 기능/능력 제공? 필요시 이용가능? 오류 없는 기능 제공?

4. 적합성

 로컬 및 외부 소프트웨어 표준 준수 ? 설계와 코딩 규칙 준수 ?

5. 내구성

 유지 및 의도하지않은 부작용 발생시 부주의 없이 수정?

6. 보수

 허용 가능한 단시간에 변경 / 수정 / 변화 / 수정 시 지원부서 필요 정보획득

7. 미학

 명백한 특정 우아함 , 독특한 흐름 및 분명한 존재감 가지고 있는지 동의

8. 지각

 과거의 인식으로 부정적 / 긍정적 인식.

1. 품질 특성
2. 기능성[Functionality]

 명시된 요구와 내재된 요구를 만족하는 기능을 제공하는 소프트웨어 제품 능력

1. 신뢰성[Reliability]

 규정된 조건에 사용될 때 규정된 성능 수준을 유지할 수 있는 능력

 사용자가 오류를 방지할 수 있도록 하는 SW제품의 능력

1. 사용성[Usability]

 능력사용자에 의해 쉽게 이해하고 학습되며 선호할 수 있게 하는 SW제품의 능력

1. 효율성[Efficiency]

 투입된 자원에 대하여 제공되는 성능의 정도

 요구되는 기능을 수행하기 위해 필요한 자원의 소요 정도

1. 유지보수성[Maintainability]

 운영환경과 요구사항 및 기능적 사양에 따른 SW의 수정, 개선 등 변경될 수 있는 능력

1. 이식성[Portability]

 SW가 다른 HW,SW 등의 환경으로 옮겨갈 수 있는 능력

 다른 환경으로 이전되는 SW능력의 정도

1. 품질 비용

품질의 부족으로 인한 후속 비용을 실행할 때 발생하는 모든 비용

1) 예방 비용

(1) 모든 품질관리 및 품질 보증 활동 을 계획 하고 조정 하는데 필요한 관리

2) 활동 비용

(2) 전체 요구사항과 설계 모델을 개발하기 위한 추가 기술 활동 의 비용

(3) 테스트 계획 비용

(4) 이러한 활동과 관련된 모든 교육 비용.

2) 평가 비용

(1) 소프트웨어 엔지니어링 작업 에 대한 기술 검토

(2) 데이터 수집 및 지표 평가 에 대한 비용

(3) 시험과 디버깅 에 대한 비용.

3) 실패 비용

(1) 내부 실패 비용 : 출하 전 오류 검출 비용

- 오류를 정정하기 위해 재작업 ( 수리 ) 의 실행 시 필요한 지용

- 재작업 부주의로 인한 감소되어야 할 부작용을 생성할 때 발생 하는 비용

- 실패 모드를 평가하기 위해 조직에 허용된 품질 측정 기준 수집 관련 비용

(2) 외부 실패 비용 : 고객 배송 후 발견된 결함

- 불만 해결, 제품의 반품 및 교환, 고객 도움 라인 지원, 보증 업무와 관련된 인건비.

5) 품질 관리와 보증

3) 품질 관리 : 검증 활동

1. 품질목표를 충족하는지 확인 하기 위한 일련의 소프트웨어공학 조치포함

2. 모델들은 완전하고 일정한지를 확인 하기 위해 검토

3. 검사 시작 전에 오류를 발견하고 수정하기 위해 코드를 검사

일련의 검사 단계 : 처리 로직 , 데이터 조작 , 인터페이스 통신 오류

 품질 미 충족 : 측정과 피드백 -> 프로세스 조율.

4) 품질 보증 : 시스템 적인 활동

4. 훌륭한 소프트웨어공학 기술 , 합리적인 프로젝트 관리 , 품질관리 조치를

지원하는 인프라 구축

5. 구성 : 품질관리 조치의 유효성을 평가하는 감사 및 보고기능 세트

6. 목적 : 관리와 기술직원에게 제품 품질에 관한 필요한 정보를 통지 하고 ,

그것에 의해 획득된 통찰력과 제품 품질을 얻을 수 있는 자신감 작용.

21장. 소프트웨어 품질보증

1. SQA 내용
   * + 1. SQA 프로세스
       2. 품질보증의 정의와 품질 제어 방법(기술재검토와 다각도 검사전략 )
       3. 효과적인 소프트웨어공학 실무 ( 방법과 도구 )
       4. 소프트웨어 개발과 수정 작업 제어
       5. 소프트웨어 개발 표준의 준수 프로시저
       6. 방법의 기술과 측정.
2. 소프트웨어 품질 보증 요소

1. 표준

- IEEE, ISO, 다른 표준화 기구 : SQA 작업은 표준이 모든 개발 작업에 적용되도록 제도화 .

2. 검토와 감사 추적

- 검토 (오류를 찾는 품질제어 작업). 감사추적 (품질지시서에 의하여 반드시 수행되는 목적)

3. 테스트

- 오류 발견 목적을 가진 품질 제어 기능. 테스트 기본 계획 수립, 효율적인 수행

4. 오류 / 결함 수집과 분석

- 오류 도입되는 것 이해, 오류 없애는 최상의 활동을 위해 -> 오류와 결함 자료 수집/분석

5. 변경관리

- 원초적으로 변경관리 못하면 혼란/부실한 품질 초래 . 충분히 변화 대응하는 관리 관례 도입

6. 교육

- 소프트웨어공학 관례 향상을 주도, 이해관계자 교육. SQA 조직:교육 프로그램의 주제 안자이고 지지자

7. 소프트웨어 판매자 관리

- 외부공급업체와 계약으로 품질 권한 통합 , 공급업체에 품질관례 제시 -> 높은 품질 보장

8. 보안 관리

- 소프트웨어 보안 달성을 위하여 적절한 과정과 기술 사용

9. 안전

- 소프트웨어 오류의 영향을 평가하고 위험을 최소화하기 위해 요구되는 단계 실시 책임 갖음

10. 위험관리

- 위험 관리 행위와 위험관련 비상계획이 제대로 설정되고 시행하여야 함.

1. 신뢰성과 가용성과 안전성 차이

1.소프트웨어 신뢰성

소프트웨어 오류가 발생할 가능성을 결정 하기 위한 통계 분석을 사용

위험이나 경미한 실수 로 반드시 고장 발생이 유발 되는 것은 아니다

소프트웨어 안전성

실패 사고로 이어질 수 있는 상태가 발생할 수 있는 조건 들을 살펴보는 길

고장은 백지 상태부터 고려하고 있지는 않지만, 전체 기본 시스템과 제반 환경을 포함한 컴퓨터 시스템 전반에서 평가.

1. 소프트웨어 신뢰성과 가용성 측정
2. 소프트웨어 신뢰성 단순 측정 [Mean Time Between Failure]

MTBF = MTTF + MTTR [Mean Time To Failure, Mean To Time Repair]

신뢰성:지정된 시간동안 고장없이 운영될 수 있는 확률

2. 소프트웨어 가용성[서비스 요청 시 서비스를 제공할 수 있는 확률]

가용성 = {MTTF/(MTTF + MTTR)} \* 100 %

가용성 측정 : MTTR 에 더욱 가까우며 소프트웨어의 유지보수의 간접적인 측정.

1. ISO 9001:2000 표준 요소
2. 품질관리시스템의 체계를 설정 하라 : 시스템개발, 시행, 개선, 중요성 강조 방법 정의
3. 품질 시스템을 기록 하라 : 과정 기술 , 업무매뉴얼 제작 , 문서 관리 방법 개발
4. 품질 관리와 보증을 지원 하라 : 품질 중요성 촉진 , 소비자 만족 중점 , 품질 계획서정의 , 의사소통 기법 정의
5. 품질관리시스템위한 검토기법을 설정 하라 : 검토방법과 환류기법 명확, 후속조치절차 정의
6. 인력, 교육 그리고 제반시설 요소들을 포함하는 품질요소들을 명확 하게 하라
7. 관리기법을 설정 하라 : 계획, 고객 요구, 기술 활동, 프로젝트 감시와 관리
8. 개선을 위한 방법을 정의 하라 : 품질데이터와 지표를 평가 , 일관적인 작업과 품질개선을 위한 방법들을 정의하라.
9. SQA 계획에 대한 표준 구조
10. 목적 및 계획의 범위
11. SQA 의 범위 내에 속하는 모든 소프트웨어 엔지니어링 산출물 설명

(모델, 문서, 코드)

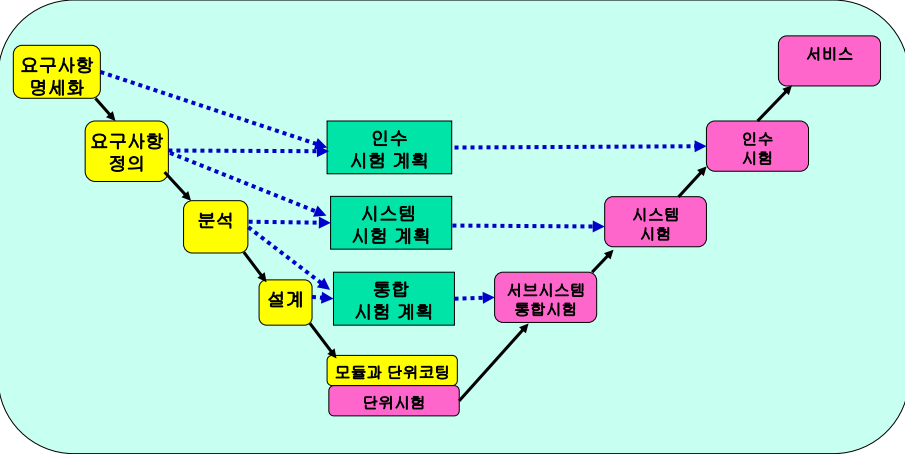
1. 소프트웨어 프로세스 중에 적용되는 모든 적용 가능한 표준과 관행
2. SQA 활동과 작업 (평가 및 감사 포함) 그리고 소프트웨어 과정을

통해 얻어진 배치

1. SQA 활동 및 작업을 지원하는 도구와 방법
2. 소프트웨어 구성 관리 절차
3. SQA 와 관련 되어진 모든 조합과 안전 관리와 방법
4. 조직의 역할과 제품의 품질에 대한 책임.

22장. 소프트웨어 테스팅 전략

1. 소프트웨어 테스팅 전략의 일반적인 특성
2. 효과적인 테스팅을 수행하려면
3. 효과적인 기술검토 수행 -> 많은 오류들 테스팅 시작전에 제거
4. 테스팅은 컴포넌트 수준에서 시작 하고, 전체 컴퓨터 - 기반 시스템 을
5. 통합하는 방향으로 “바깥쪽으로” 진행
6. 다양한 소프트웨어공학 접근법과 시점에 따라 : 서로 다른 테스팅 기법 적합
7. 테스팅 : 소프트웨어 개발자와 독립적인 테스트 그룹에 수행
8. 테스팅과 디버깅 : 서로 다른 활동
9. 디버깅은 모든 테스팅 전략에 수용.
10. 검증 및 확인의 정의 및 차이점
11. 검증
    1. 소프트웨어 특정 기능을 올바르게 구현 하였는지를 보장하는 일련의 작업
    2. “우리가 제품을 올바르게 만들고 있는가”
12. 확인
    1. 개발된 소프트웨어가 고객의 요구사항에 맞는지를 보장하는 일련의 작업
    2. “우리가 올바른 제품을 만들고 있는가”
13. 테스팅 전략과 나선형 전략, 개발과 시험을 연결하는 시험 계획



1. 테스팅 전략의 절차적 관점에서 프로세스 단계

1)단위 테스팅

-완벽한 커버리지 와 최대한 오류 검출 보증

.컴포넌트 제어구조에서의 특정경로 검사 테스팅 기법

2)통합 테스팅

-검증과 프로그램구축이라는 이중문제

-통합시입 /출력 초점을 맞추는

테스트케이스 설계 기법

3)확인 테스팅

-소프트웨어가 모든 기능, 동작,

성능요구사항을 충족하는 최종보증 제공

4)상위 레벨 테스팅

-컴퓨터시스템공학(SE벗어남)의 넓은 개념

-다른 시스템 요소와 결합

.하드웨어, 사용자, 데이터베이스

-달성유무 검증

.시스템 요소가 적절히 조화

1. . 모든 시스템 기능/성능달성.
2. 단위 테스팅, 통합 테스팅, 시스템 테스팅 종류 및 차이점

1. 단위 테스팅

1)소프트웨어설계의 가장 작은 단위: 소프트웨어 컴포넌트 또는 모듈을 검증초점

2)컴포넌트수준 설계기술서를 가이드

3)모듈의 경계내부에 있는 오류를 찾아내기 위해 주요제어경로가 테스트

4)테스트의 상대적 복잡성과 테스트가 찾아내는 오류들은 단위 테스팅을 하기위해 설정된 제한된 범위로 한정

5)컴포넌트의 경계내에서 내부처리 로직과 데이터구조에 초점

2. 통합 테스팅

소프트웨어 아키텍처를 구축하는 동시에 인터페이스와 관련된 오류를

찾아내기 위한 테스트를 수행하는 체계적인 기법

목적: 단위테스트가 끝난 컴포넌트들을 가지고 설계에서 지시된

프로그램 구조를 만드는 것

“서로 결합하는”(인터페이스) -> 점증적인 접근방식

-점증적인 통합

1.하향식 통합 테스팅(Top-down integration testing)

1)모듈들은 메인 제어 모듈(메인 프로그램) 부터 시작하여 계층구조를 따라 아래방향으로 이동 하면서 통합

2)메인 제어 모듈에 종속되는 모듈들(최하위 모듈):깊이-우선,넓이-우선중

한가지 방식으로 구조에 통합

3)깊이-우선(depth-first integration)통합

-프로그램 구조의 주요 제어경로에 있는 모든 컴포넌트를 통합

-주요경로선택:임의적(주로:왼쪽에서 오른쪽)

4)넓이-우선(Breadth-first integration)통합

-구조를 수평적으로 따라 이동하면서 각레벨에서 바로 하위 컴포넌트 통합

2. 상향식 통합 테스팅(Bottom-up integration testing)

1)원자 모듈(프로그램구조의 최하위레벨에 있는 컴포넌트)로부터 구축과 테스팅 시작

2)컴포넌트들이 아래에서 위로 통합되기 때문: 스텁 필요없음(하위모델사용)

통합이 위쪽으로 이동하기 때문

3)별도의 테스트드라이브에 대한 필요성 줄어듬.

4)상위 두레벨이 하향식으로 통합되었다면

드라이버의 수는 실질적으로 줄일 수 있고, 클러스터의 통합도 간소화

-회귀테스트

3. 회귀 테스팅(regression testing)

1)변화로 인해 의도하지 않은 부작용이 전파되지않았다는 것을 보증하기위해

이미 수행된 테스트의 일부분 다시-실행하는 것

-통합 테스트 중 모듈추가: 소프트웨어변경(데이터흐름변경/IO/제어로직)

2)변경이 의도하지 않은 동작이나 추가적인 오류가 나타나지 않도록 도움

3)테스트케이스의 일부를 다시-실행함으로써 수동/자동화된 캡처/플레이백도구

-Capture/playback:테스트케이스와 연속적인 플레이백 및 비교결과제공

-스모크 테스팅

4. 스모크 테스팅(Smoke testing)

1)시간이중요한 프로젝트에서 시간을 조절하기위한 메커니즘으로 설계

2)복잡하고 시간이중요한 소프트웨어프로젝트에 적용: 제품소프트웨어

3)스모크 테스팅 방법의 활동

1.코드로 변환된 소프트웨어 컴포넌트는 빌드로 통합

-빌드: 하나 또는 그 이상의 제품기능의 구현이 요구되는

모든 데이터파일, 라이브러리, 재사용가능한 모듈, 컴포넌트포함

2.일련의 테스트는 빌드의 기능을 제대로 수행하지 못하게 하는 에러들을 찾아내기 위해 설계

-목적: 소프트웨어프로젝트가 계획된 일정을 맞추지 못하도록 가능성이 가장

큰 오류(show-stopper:HW/SW 못쓰게 하는Bug)를 밝혀내는데 있음

3.빌드는 다른 빌드와 통합되고, 전체제품(현재상태에서)은 매일 스모크 테스팅

-통합방식은 하향식 또는 상향식이 될 수 있음.

6) 소프트웨어 결함 유형

1)알고리즘결함(algorithm micfaults)

컴포넌트의 알고리즘 또는 논리가 처리되는 단계 중에 무엇인가

잘못되었기 때문에 주어진 입력에 대해서 적절한 출력을 보여주지

않을 때 발생함

* + 1. 너무 이른(늦은)분기
    2. 잘못된 조건에 대한 테스팅
    3. 변수 초기화 혹은 루프 불변량(loop invariants)설정에 대한 망각
    4. 특정한조건(0으로나누는연산)에 대한 테스트 망각
    5. 부적절한 자료형을 가진 변수비교.

2)계산 및 정밀도결함

공식 구현의 오류나 요구된 정도의 정확성에 대한 결과를 계산하지

못할 때 발생함

1. 표현식에서 정수 및 고정 혹은 부동소수점 변수들을 같이 사용시
2. 부동소수점 자료의 부적절한 사용, 예기치 않은 절사(truncation)또는

연산들의 순서가 적합한 정밀도보다 낮은 결과를 가지게 할 수 있음.

3)문서화결함

문서화가 프로그램이 실제 수행하는것과 일치하지 않을 경우

1. 대부분의 우리들은 수정하기위해 코드를 검토할 때 문서화를 믿는 경향이 있기 때문에 결함.
2. 발생시 프로그램의 수명 후반부에 다른 결함들의 급증을 초래.

4)스트레스 또는 과부하 결함(stress or overload faults)

자료구조들이 지정한 용량을 초과할 때 발생(큐의 길이,버퍼의 크기).

5)용량 또는 경계결함(capacity or boundary faults)

시스템활동이 명세 된 한계까지 도달하지 못해 시스템의성능이 받아들여질 수 없을 때 발생함.

6)타이밍 또는 조정결함(timing or coordination faults)

이벤트를 조정하는 코드가 정확하지 않을 때 발생함.

7)처리능력 또는 성능결함(throughput or performance faults)

시스템이 요구사항에 미리 정해진 속도로 수행되지 않을 때 발생함

8)복구결함(recovery faults)

실패가 우연히 발생했을 때와 시스템이 설계자가 원하고 고객이

요구한대로 작동되지 않을 때 발생함

–(예)시스템처리 중에 전원고장 발생-복구의 문제

9)하드웨어 및 시스템 소프트웨어결함(hardware and system software faults)

공급된 하드웨어 및 시스템 소프트웨어가 문서화된 작동조건과

프로시저에 따라서 실제로 작동되지않을 때 발생함

10)표준 및 프로시저 결함(standards and procedures faults)

시스템이 테스트되고 수정되기 때문에 결함이 생성되는 환경에서는

발생할 수도 있음.

31장. 프로젝트 관리

1) 프로젝트관리 정의

1. 계획, 작업모니터링, 사람들에 대한 제어, 프로세스

2. 소프트웨어초기의 개념으로부터 완전히 작동하여 배포되는 기간 중에

일어날 수 있는 사건 까지를 포함

2) 프로젝트관리의 범위와 세부내용

1. 인간

인적요인(People factor) 중요

-모든 기관: 비즈니스전략목표 달성하기위해 작업인력에게 매력을 느끼게

하고, 그들을 개발, 동기부여, 조직화, 작업능력을 지속적으로 향상

1)인적역량 성숙모델(People-CMM)

-소프트웨어 인력을 위한 중요한 실무분야정의

. 사람의 배정, 의사소통과 협동, 작업환경, 훈련, 보상, 수행능력분석 및 개발, 경력개발, 작업진단개발, 팀/문화개발

-높은 인적역량 성숙도 성취한 조직

. 소프트웨어프로젝트관리업무를 효과적으로 수행가능성이 높음.

-조직이 성숙한 소프트웨어 프로세스 생성지침

. 소프트웨어 역량 성숙모델: 통합(SCMM I).

2. 제품

1)프로젝트가 계획되기 전 업무

-제품의 목적과 범위가 수립

-해결책, 대안 고려

-기술적, 관리적 제한조건 파악

2)프로젝트 계획 전 업무에 대한 상세한정보가 없을 경우: 정의불가능

-합리적인(정확한) 비용추정

-위험요소의 효과적인 평가

-관리가능한 프로젝트의 일정 (현실적인 프로젝트작업의 분화 또는 진행상황 징후)

3)제품의범위

-기본적인 데이터, 기능 그리고 제품을 특징짓는 행위

-특성들의 정량적인 방법으로 범위 정의

제품의 목적과 범위가 확실: 해결책에 대한 대안 고려

-대안

. 납품시한, 예산상의제약, 개인적능력, 기술적상호작용

. 많은 요인에 의한 제한조건하에서 “최선의” 해결방안 선택가능.

3. 프로세스

1)소프트웨어 프로세스

-소프트웨어개발을 위한 종합적인 계획을 수립할 수 있는 프레임워크 제공

-소프트웨어프로젝트 특성들과 프로젝트팀의 요구사항들을 프레임워크 작업에 적응

. Task set : 작업, 이정표, 산출물, 품질보증요건

2)프로세스모델에 추가사항: 보호행위

-소프트웨어품질보증, 소프트웨어 형상관리 및 측정.

4. 프로젝트

계획되고 제어된 소프트웨어 프로젝트: 복잡도를 관리할 수 있는 유일한 방법.

1)소프트웨어프로젝트성공/실패율: 250개대형프로젝트(1998~2004)

-25개프로젝트: 일정,비용, 품질목표달성[10 %]

-50개정도프로젝트: 35% 미만일정지연, 비용초과[20 %]

-175개프로젝트: 크게 지연/초과, 미완성된 채 종결[70 %]

2)프로젝트 실패 피하는 방법

-프로젝트관리자/엔지니어 경고신호 피해야 함

-프로젝트를 계획, 관찰, 제어할 수 있는 상식적인 방법 개발.

3) 프로젝트계획서에 포함내용

1. 프로젝트범위 및 프로젝트 스케줄

2.프로젝트 팀 구성

3.제안된 시스템의 기술적 기술(technical description)

4.프로젝트표준안, 절차, 제안된 기술 및 도구

5.품질보증계획, 형상관리계획

6.문서화계획(documentation plan)

7.데이터관리계획(data management plan)

8.자원관리계획(resource management plan)

9.테스팅 계획

10.교육 계획

11.보안 계획

12.위험관리 계획

13.유지보수 계획.

4) 위험관리 프로세스

Risk identification(위험식별) 점검목록

1.기술위험

2.소프트웨어 혹은 하드웨어 기술로부터 유도된 위험

3.인적위험

4.개발팀에 있는 사람과 관련된 위험

5.조직의 위험

6.소프트웨어가 개발되는 조직의 환경과 관련된 위험

7.도구위험

8.지원소프트웨어와CASE 도구로부터 생기는 위험

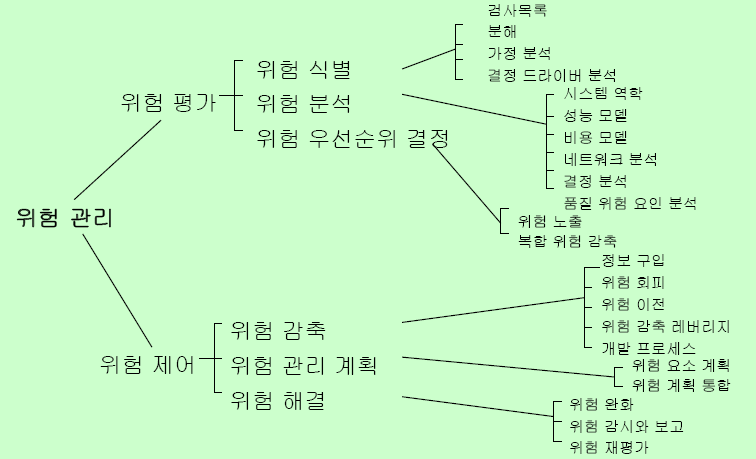
9.요구사항위험

10.요구사항 변경과 관리를 위한 프로세스로부터 생긴 위험

11.추정위험

12.시스템을 만들기 위해서 필요한자원에 대한 추정으로부터 생긴 위험.

5) 위험관리 단계



36장. 시스템 유지보수

1) 소프트웨어 개발하는 동안 변경사례



2) 유지보수 종류와 특징

1. 수정유지보수(Corrective Maintenance)

1) 시스템의 일상적인 기능을 제어하기 위해 유지보수팀이 결함에 따른

문제에 역점을 두고 유지보수 하는 것을 의미함

2) 실패가 발생하면 발견된 실패는 유지보수팀의 관심을 끌게 됨

3) 유지보수팀은 사용자와의 상호작용 없이 적절하게 작동하도록 하기위해

설계, 코드, 테스트를 다시 수행함.

2.적응유지보수(Adaptive Maintenance)

1) 적응적 변경은 결함을 수정하지 않지만 시스템의 진화에 따라

적응하도록 하는 여분의 파라미터를 추가함

2) 적응유지보수는 하드웨어나 환경에 대해서도 변경을 수행할 수 있음.

3.기능향상유지보수(Perfective Maintenance)

1) 결함으로 인해 제안된 변경 뿐만 아니라 시스템의 일부측면을

향상시키기 위한 변경을 포함하고 있음

2) 완벽한 유지보수의 예

3) 문서화가 항목을 분류하기 위해 변경됨

4) 테스트 슈트가 테스트 적용범위를 향상시키기 위해 변경되며

가독성을 강화하기 위한 코드 및 설계의 수정.

4.예방유지보수(Preventive Maintenance)

1) 결함을 예방하기 위한 시스템의 일부측면들의 변경을 포함하고 있음

2) 보통 프로그래머나 코드분석가가 아직 실패를 유발하지 않으며

손상이 발생하기전 결함을 정정하는 활동을 통해

실제적이거나 잠재적인 결함을 찾았을 때 가능함.

3) 유지보수에서 발생하는 문제

1. 제한된 이해

1)소프트웨어 및 하드웨어 요구와 사용자 요구간의 균형 외에도

유지보수팀은 인간이해의 한계를 다루어야 함

2)소프트웨어유지보수노력의47%가수정될소프트웨어를이해하는데소요

3)유지보수 프로그래머가 직면하는 문제의 절반이상이 사용자의

기술이나 이해부족에 따른 것이라는 것을 발견함

4)명확하고 완벽한 문서화와 교육의 중요성을 시사

5)유지보수팀은 또한 사람을 잘 다루는 기술을 필요로 함.

2. 관리우선순위

1)유지보수팀은 시스템의 필요에 대한 고객의 관리요구에 중점

2)관리우선순위는 종종 기술적인 우선순위를 무시함.

3. 사기(Morale:의욕)

1)유지보수 중에 발생하는 문제의 11.9%가 낮은 사기와 생산성에

기인한다고 지적함

2)사기가 낮은 주요요인

3)유지보수팀을 이류(second-class)로 분류하는 측면

4)프로그래머: 때로 시스템이 계속 동작하도록 유지하는 것 보다

시스템 설계 및 개발에 더 많은 기술이 필요하다고 생각함

5)유지보수 프로그래머: 개발자들이 결코 예상하지못하였던 문제를

다루어야 함

6)유지보수기간동안 발생하는 문제의 80%는 프로그래머가 동시에

여러 프로젝트에서 작업하기 때문에 문제해결에 충분한 시간을

투자할 수 없다는데 있음.

4. 기술적인문제

1)구현시에 채택하였던 특정 패러다임이나 프로세스에 따라 기술적인

문제가 발생하기도 함.

2)가공물과 패러다임

3)객체지향 프로그램을 유지보수 하는 것 역시 복잡한상속으로 인해 고도로 상호 연결된 컴포넌트를 포함하기 때문에 많은 문제가 발생가능

4)부적절한 설계 명세서와 저질프로그램 및 문서화는 유지보수 노력의 약

10%정도의비율을차지함

5)문제는 하드웨어 및 소프트웨어 또는 데이터를 신뢰할 수 없을 때 발생.

5. 테스팅의 어려움

1)언제 테스트를 해야 할지를 결정할 때 문제가 될 수 있음

2)시간 가용성 문제 외에도 테스팅을 위한 테스트 데이터가 만족스럽거나

적절하지 못할 수도 있음

3)테스터들이 설계나 코드변경의 효과를 예측하고 준비하는 것이 쉬운 일이 아님

4) 시스템을 유지보수하기위해 필요한노력을 결정요인

1.애플리케이션유형

2.시스템 참신성

3.교체와 유지보수 스태프의 가용성

4.시스템생존기간

5.변화하는 환경에 대한 의존도

6.하드웨어특성

7.설계품질

8.코드품질

9.문서화의품질

10.테스팅 품질.

5)소프트웨어 재성의 종류와 특징 그리고 프로세스

1.재구성

1)이해와 변경이 용이하도록 소프트웨어를 재구성함

재구성과 관련된 세가지 주요한 활동

정적분석은 코드를 시맨틱 네트워크나 방향성 그래프로서 표현하는데

이용할 수 있는 정보를 제공함

변형기술을 기초로 하는 연속된 단순화를 통해 정제됨

정제된 표현은 구조화되고 동등한코드를 생성하기위해 번역, 이용됨.

2.재문서화

시스템문서화를 생산하는 소스코드에 대한 정적인 분석을 포함

출력에 포함되는 내용

컴포넌트호출관계

데이터 인터페이스 테이블

데이터사전정보

데이터흐름 테이블 또는 다이어그램

의사코드

테스트경로

컴포넌트와 다양한 전후 참조.

3.역공학

소스코드로부터 소프트웨어 시스템에 관하여 명세서와 설계정보를

제공함(정보를 추출함)

역공학의 핵심은 상세한 소스코드 구현으로부터 명세서를 추상화

4.재공학

재공학은 역공학의 확장

전체시스템의 기능을 변경하지 않고 새로운 소프트웨어 소스코드를 생산함

완전히 자동화된 재공학은 가까운 장래에는 가능할 것 같지 않으므로

프로세스는 변환과 인간과의 상호작용을 결합해야만 함