



Chapter 4. 정리

1. 프로세스 정의, 중요한 이유, 특징
2. 프로세스 모델링 하는 이유
3. 프로세스 모델의 장단점 및 활용

When I dream

*I could have all the gifts
I want and never ask please.*





제7장.모델링-실무가이드 원칙

March. 2018

Young-gon, Kim

ykkim@kpu.ac.kr

Department of Computer Engineering

*K*orea *P*olytechnic *U*niversity



Topics covered

- ◆ 소프트웨어공학 지식
- ◆ 핵심 원칙
- ◆ 각 프레임워크 활동을 가이드하는 원칙들
- ◆ Project Planning

1. 소프트웨어공학 지식

◆ 실무(practice)

- 일상

- Sets { 개념, 원칙, 기법, 도구}

- 소프트웨어

- 관리자 : 소프트웨어 프로젝트 관리

- 소프트웨어 엔지니어 : 컴퓨터 프로그램 개발 허용

- 소프트웨어 프로세스 모델 : 업무 처리를 위한 필요기술 및 관리기법 연관

- 무계획적인 목적, 불분명한 접근 -> 좀더 효율적, 성공적으로 변환

◆ 소프트웨어공학 지식

- 3년 반감기 (3-year half-life) : 소프트웨어 개발 지식 주기

- 지식이 3년내 : 50% 감소(쓸모 없어짐) -> 특히 기술관련 지식 영역

- 소프트웨어공학 원칙 : 3년 반감기 적용 않됨

- 주요 원칙 : 소프트웨어 엔지니어의 업무를 가이드하는 기본 아이디어

- 소프트웨어공학 모델, 방법론, 도구.

2. 핵심 원칙

◆ 프로세스를 가이드하는 원칙

- 애자일 하라
 - 업무 (경제적인 행동), 기술적인 접근(단순, 작업산출물 : 간결, 부분적인 결정)
- 모든 단계에서 품질에 초점을 두어라
 - 프로세스 활동, 행위, 작업 출구 조건 : 작업 산출물의 품질에 기준
- 적응할 준비를 하라
 - 프로세스 : 문제, 인력, 프로젝트의 제약조건에 조정
- 효율적인 팀을 꾸려라
 - 인력(소프트웨어 개발 핵심), 자체 조직화 팀 구축(상호 신뢰와 존경)
- 의사소통과 조정을 위하여 매커니즘을 설립하라
 - 중요 정보소실, 이해관계자들 최종산출물 생성 노력 실패 : 프로젝트 실패 이유
- 변경을 관리하라
 - 메커니즘 수립 : 변경, 요청, 평가, 승인, 수립 되는 방법 관리
- 리스크를 평가하라
 - 개발 과정에서 많은 잘못 : 긴급 대책 수립이 필수적
- 다른 사람을 위하여 가치를 제공하는 산출물을 생성하라.
 - 산출물 : 프로세스 활동, 행위, 작업 가치 제공, -> 모호함/ 누락 없이, 필요한 정보 포함.

2. 핵심 원칙

◆ 실무를 가이드하는 원칙 : 기술적인 작업

- 분할하여 공략 하라
 - 분석과 설계: 관심사 분리 (SoCs), 큰문제를 작은 요소로 분해->해결이 쉬움
- 추상화의 사용을 이해하라
 - 추상화(의사소통 단계 사용) -> 복잡한 문제를 단순화, 상세한 의사소통 제거
- 일관성을 유지하라
 - 소프트웨어 개발 단계에서 일관성 유지 -> 익숙한 문맥으로 SW 사용 용이
- 정보의 전송에 초점을 맞춰라
 - 정보 : 인터페이스를 통해 전달, 그 결과로 에러, 누락에 초점, 모호성 가능성->테스트
- 효율적인 모듈성을 갖는 소프트웨어를 만들어라
 - 모듈성 : 효율적, 배타적-> 높은 응집력, 낮은 결합도
- 패턴을 찾아라
 - 패턴 : 문제점/솔루션->통찰력, 경험 소통 공동언어, 요구충족/ 좋은 구조 이해
- 가능하면 문제와 그 솔루션을 다양한 관점에서 나타내라
 - 문제/해결책 다른 관점 평가 : 훌륭한 통찰력, 에러/누락 발견-> 시나리오/클래스/동작 관점
- 소프트웨어를 유지할 누군가를 기억하라.
 - SW : 향상(결함 수정, 환경변화에 적응, 이해관계자 요구사항)-> 유지보수에 적용.

3.각 프레임워크 활동을 가이드하는 원칙들

◆ 의사소통 원칙 : 고객과의 의사소통

- **경청** 하라
 - 화자의 **단어/말에 집중**, 명확치 않으면 **방해않는 범위에서 명확하게 요구**, 이야기중에 말/행동에 **논쟁 회피**
- **의사소통 이전에 준비**하라
 - 만나기전 : 문제 이해 시간 필요, **사전 의제 준비** (미팅 주관 책임자)
- **누군가를 활동을 활성화**하라
 - 생산적인 대화 : 발생하는 **갈등 중재**, **리더 필요** (다른 원칙 준수)
- **얼굴을 대면**하는 의사소통이 최고이다
 - 의사소통 좋은 활동 : 관련 정보 다른 표현 (**그림의 간략한 문서**)
- 의사 결정을 **메모하거나 문서화**하라
 - 기록자 역할 : **중요한 점**, **결정 사항**, **미결정사항** -> 작성
- **협력을 유지**하라
 - 협력과 합의 -> **구성원의 정보 수집**, **작은 협력**(팀 구성원사이 믿음 구축 역할)-> 팀을 위한 **공동 목적** 제공
- **초점을 유지**하라 : **토의를 모듈화**하라
 - 참여자 많을 수록 : 회의 주제 벗어날 확률 높음 . **중재자 필요** (결정이 된 주제 만을 두고 **토의 모듈** 유지)
- 뭔가 불분명한 점이 있으면 **도식화**하라 .
 - 구두 로써 의사소통 실패한 경우 : **스케치**, **도식화** -> 명확성 제공
- 무엇인가에 **동의 / 부 동의 / 특징또는 기능이 불분명**, **명확화 못하더라도 계속**하라
 - **계속 진행**하라 -> 의사소통을 신속하게 달성하는 가장 좋은 방법
- 협상은 경쟁 / 게임이 아니고 , **양쪽 당사자 모두 이기는 것이 최선**이다 .
 - 협상 필요 : 기능 , 특징 , 우선순위 , 제품 인수일 . **협상** : 타협 필요 -> 팀 협력 -> 공동 목표 달성.

3.각 프레임워크 활동을 가이드하는 원칙들

◆ 계획 수립 원칙 : 전략 / 기술적인 목표의 로드맵 정의

- 프로젝트 범위를 이해하라
 - 범위 : 소프트웨어 팀이 가야할 목적지 제공 -> 로드맵
- 이해관계자가 계획 활동에 관여하게 하라
 - 이해 관계자(우선순위 정의, 프로젝트 제한조건 결정), 엔지니어(인수 순서, 시간 계획, 프로젝트 이슈): 협상
- 계획은 언제나 반복적이라는 것을 인식하라
 - 프로젝트 수행중 많은 것이 수정 : 반복적 점증적 모델 소프트웨어 점증 인계후 재 계획:사용자 피드백 근거)
- 자신이 알고 있는 무엇인가에 근거하여 추정하라
 - 추정목적(팀 현재 수행한 작업 근거- 노력, 비용, 작업기간, 작업 지표제공), 추정 불신뢰(정보 모호/미신뢰)
- 계획을 정의할 때 위험도 고려하라
 - 비상계획 필요 (많은 영향 , 높은 위험 확률 파악), 위험 발생 -> 조정 (예산 손실 및 조정 필요)
- 현실을 직시하라
 - 실제 상황 (의사소통에 잡음 , 누락 / 애매성 발생 , 근로시간 부정확)-> 프로젝트 계획 고려
- 계획을 정의하는 것처럼 세분화를 조정하라
 - 세분화 : 프로젝트 진행됨에 따라 상세레벨, 하위 세분화 계획 -> 장시간 동안 계획
- 어떻게 품질을 보장하려 하는지 정의하라 .
 - 계획 (팀 품질 보장 파악) 정의 필요 : 기술검토 , 페어 프로그램 구축
- 어떻게 변경을 수용하려 하는지 기술하라
 - 변화가 소프트웨어 작업이 진행되는 것처럼 어떻게 변경되는 지 감지 (고객 변경 , 팀 수행 , 영향 / 비용 ?)
- 자주 계획을 추적하고 요구에 맞게 조정하라 .
 - 일일 스케줄 지연 가능 : 계획대비 실제 실행 작업 불일치 -> 문제 영역을 매일 경과 추적.

3. 각 프레임워크 활동을 가이드하는 원칙들

◆ 애자일 모델링 원칙 : 추상화의 다른 수준에서 목적달성

- 소프트웨어 팀의 기본 목표는 모델을 만드는 것이 아닌 , **소프트웨어를 만든 것이다** .
 - **신속성**(빠른 시간에 고객에게 SW 제공) : 신속성 모델이 가치 , 모델 회피 (프로세스 지연 , 적은관점 제공)
- **가볍게 이동**하고 , **필요이상으로 모델**을 만들지 마라
 - 모델 : 변경시마다 최신 상태 유지. **SW 쉽게/빠르게 구축할 수 있는 모델** 구축(신모델 :구축 시간 많이 소요)
- 문제나 소프트웨어를 묘사할 **가장 단순한 모델**을 만드는데 매진하라
 - 모델 단순하면 생성될 **소프트웨어**도 단순->**통합/테스트/유지보수 용이**, **구성원 이해/비판 용이/산출물 최적**
- **변화를 잘 수용하는 방법**으로 모델을 구축하라
 - **합리적인 모델 없이** -> 중요한 특징을 빠트리고 설계
- 만들어진 각각의 모델에 대해 **명백한 목적을 명시**할 수 있게 하라
 - 모델 생성시마다 자신에게 **왜 하는지에 질문** 필요 -> 모델의 확고한 정당성 미증명되면 시간 허비하지마라.
- 개발한 모델을 **즉시 시스템에 적용**하라
 - 응용시스템마다 **고유한 모델 표기법과 규칙 적용** 필요
- **유용한 모델**을 만들도록 노력해라 . 그러나 완벽한 모델을 만드는 것은 잊어라
 - 완벽한 모델 : 프로젝트 진행됨에 따라 **상세레벨** . -> 장시간 동안 계획
- 모델의 **구문에 독단적**이지 마라 .
 - 모델링동안 **표기법 일치** . 모델의 중요한 특징 : **다음 단계의 소프트웨어공학 작업** 수행토록 정보 전달 .
- 서류상 문제가 없이 보이지만 , **직감으로 모델이 맞지 않는다면** 아마 걱정될 것이다 .
 - **숙련된 엔지니어(직감을 믿음** , SW 작업은 많은 교훈을 가르침 모델이 실패 확정적 -> **모델 검토 재 모델**)
- 당신이 할 수 있는 한 **피드백**을 받아라 .
 - 모든 모델 : 소프트웨어 팀의 구성에 의해 검토 필요 -> **모델링 실수/오해 수정**, **생략된 기능/특징 피드백**.

3.각 프레임워크 활동을 가이드하는 원칙들

◆ 요구사항 모델링 원칙 : 요구사항분석문제와 원인확인

- 문제의 정보 도메인은 반드시 표현되고 이해되어야 한다 .
 - 정보 도메인 : 시스템에 입력되는 데이터 , 시스템에 출력되는 데이터 , 지속적으로 데이터 객체를 수집 및 조직하는 저장소
- 소프트웨어가 수행하는 기능은 반드시 정의되어야 한다
 - 소프트웨어 기능 : 사용자에게 직접 제공 / 특징에 대한 내부지원 , 입력된 데이터 변환 , 시스템 요소 제어
- 소프트웨어의 동작은 (외부 사건의 결과로) 반드시 표현되어야 한다
 - 소프트웨어 동작 : 외부환경과 상호작용에 의해 발생
- 정보 , 기능 및 동작을 묘사하는 모델은 단계화된 (또는 계층적인) 방식에서 상세하게 드러난 방법으로 반드시 분할해야 한다 .
 - 분할정복 전략 : 복잡한 문제를 전체를 해결하기 어려운 것을 해결 .
분할 (관심) 분리 -> 핵심전략
- 분석 작업은 필수적인 정보로부터 구현 세부 사항을 향해 전달되어야 한다 .
 - 분석 모델링 : 최종 사용자의 관점으로부터 문제 기술 시작 .
 - 문제의 본질 : 해결책을 고려하지 않고 기술.

3.각 프레임워크 활동을 가이드하는 원칙들

◆ 디자인 모델링 원칙 : 시스템의 다양한 뷰 제공

- 설계는 **요구사항 모델에 대해 추적**할 수 있어야 한다 .
 - 요구사항 모델 : 문제 정보 도메인 , 사용자의 **가시적인 기능** , **시스템 동작** , **서비스 제공 방법론**
- 언제나 **시스템 아키텍처를 구축**하는 것을 고려하라 .
 - SW 아키텍처 : **인터페이스** , **데이터 구조** , **프로그램 제어 흐름 동작** , **테스트 실행 방법** , **시스템 유지보수**
- **데이터의 설계**는 기능 처리의 설계와 같이 중요하다 .
 - 잘 구조화된 데이터 설계 : **프로그램 흐름을 단순** , **컴포넌트 설계 / 구현 용이** , **처리전반을 효율적**
- **인터페이스 (내부 및 외부)** 는 주의 깊게 설계되어야 한다
 - 잘 설계된 인터페이스 : **통합을 용이** , **컴포넌트 기능 검사**하는 시험자 지원
- 사용자 인터페이스 설계는 **최종사용자의 요구에 전환되고** , **사용용이성 강조**해야 한다
 - 모든것 (내부기능 , 데이터 구조 , 아키텍처)이 잘 설계 되고 , **허접한 인터페이스** : 소프트웨어가 형편없다 .
- 컴포넌트는 수준 설계는 **기능적으로 독립**되어야 한다 .
 - 기능적 독립 : 소프트웨어 **일관성 척도** . 컴포넌트에 제공되는 **기능성** : **응집**되어야 함 .
- 컴포넌트는 외부환경에 대하여 **상호간 느슨한 결합**을 가져야 한다 .
 - 결합도 증가 : **에러 전파가 증가** 가능성 , **유지보수성 감소** -> **결합도는 낮게 유지**
- 설계 표현 (모델)은 **쉽게 이해**할 수 있어야 한다 .
 - 설계가 이해하기 어려움 : 설계는 효율적인 **의사 소통 매체로 제공**하지 못함 .
- 설계는 **반복적으로 개발**되어야 한다 .
 - 반복 : **단순성**에 노력. 첫활동 (설계 정교화 / 에러 수정), 그 다음 반복 (**설계를 가능한 단순하게 작성** 매진)
- 설계 모델의 제작은 **애자일 접근법을 배제**하지 마라 .
 - 지극히 높은 수준 목적과 최신 멀티스레드 실시간 환경에 있는 **다른 모듈과 상호작용을 이해**하기 어렵다.

3. 각 프레임워크 활동을 가이드하는 원칙들

◆ 구축 원칙

- 구축 활동 : 코딩 + 테스트
- 테스트
 - 통합 테스트 : 시스템에 구축됨에 따라 실행
 - 유효성 테스트 : 완벽한 시스템을 위하여 요구사항들이 충족 여부 평가
 - 검수 테스트 : 고객에 의해 요구되는 특징과 기능을 테스트

1) 코딩 원칙

1.1 사전 원칙들 : 한줄의 코드를 작성하기 전에 명심

- 해결하려고 하는 문제로 이해하라 .
- 기본적인 설계 원칙과 개념을 이해하라 .
- 구축하려는 SW 요구를 충족 , 운영될 환경에 부합하는 프로그램 언어 선택하라.
- 작업하기 쉽게 만들어 주는 도구를 제공하는 프로그래밍 환경을 선택하라.
- 일단 코드가 완성된 컴포넌트를 적용할 많은 단위 테스트를 만들어라.

3.각 프레임워크 활동을 가이드하는 원칙들

◆ 구축 원칙

1) 코딩 원칙

1.2 코딩 원칙들 : 코딩을 시작하게 되면 명심

- 구조적인 프로그래밍이 실무를 따르도록 알고리즘을 제한하라 .
- 페어 프로그래밍 방식의 사용을 고려하라 .
- 설계 요구사항을 충족할 데이터 구조를 선택하라 .
- 소프트웨어 구조를 이해하고 그것과 일관되게 인터페이스를 만들어라 .
- 조건부 로직을 가능한 단순하게 유지하라 .
- 테스트하기 쉽게 중첩 루프를 만들어라
- 의미있는 변수 이름을 선택하고 , 다른 로컬 코딩 기준을 따르라
- 자기 문서화 되는 코드를 작성하라
- 이해를 돕는 시각적 레이아웃을 만들어라 (들여쓰기와 빈 줄들)

1.3 유효성 원칙들 : 첫번째 코딩 완료한 후 명심

- 적절한 시점에 코드 워크스루를 실행하라 .
- 단위 테스트들을 수행하고 발견한 오류들을 수정하라 .

3. 각 프레임워크 활동을 가이드하는 원칙들

◆ 구축 원칙

2) 테스트 원칙

- 모든 테스트는 **고객 요구사항에 추적**해야 한다 .
 - 테스트 목적 (**오류 발견**), 심각한 결함 : 요구사항 충족시키는 프로그램 실패 .
- 테스트는 테스트가 **시작되기 전에 계획**되어야 한다
 - 테스트 계획 (요구사항모델 완료후 시작 ~ **코드 생성전**), **테스트 사례** (설계모델 확정후)
- **파레토 법칙**을 소프트웨어 테스트에 적용시켜라
 - 모든 프로그램 **컴포넌트 20%** : 테스트 동안 발견되지 않는 모든 **오류의 80%**
- 테스트는 “**작은것**”에서 **시작하여 “큰것**”으로 진행해야 한다
 - 처음 계획 (**개별 컴포넌트**에 초점), 테스트 처리 과정 (통합된 클러스터 -> **전체시스템**)
- **완전한 테스트는 불가능**하다
 - 경로 조합은 많음 : 모든 경로 조합 수행 불가능 -> **프로그램 논리**를 고려
- 예상되는 **결함 정도와 상응하는 테스트 노력**을 시스템의 각 모듈에 적용시켜라
 - 최신의 모듈이거나 개발자에 의해 **최소한으로 이해**
- **정적인 테스트 기술은 높은 결과**를 얻을 수 있다
 - 소프트웨어 **문서에 기인** : 요구사항 명세서 , 코드 검토서 , 사용자 설명서
- 결함을 추적하고 테스트를 통해서 발견된 **결함에서 패턴**을 찾아라
 - 발견된 결함 형태 (소프트웨어 안정서의 척도), 시간 경과후 발견된 **결함 패턴** (결함예측)
- 소프트웨어가 정상적으로 동작하는 것을 증명하는 **테스트사례**를 포함하라 .
 - 유지보수 적응 : 상호작용된 컴포넌트 시험 , **회귀테스트 사례**

3.각 프레임워크 활동을 가이드하는 원칙들

◆ 배치 원칙 : 인수 , 지원 , 피드백

- 소프트웨어에 대한 **고객의 기대는 반드시 관리**되어야 한다 .
 - 약속에 다른 기대수준 관리
 - 고객에게 대립되는 메시지 (너무 자주 많은 인계 약속 -> 더 많은 기대 -> 실망)
- **완성된 배포 패키지는 조합되고 테스트**되어야 한다 .
 - 사용자에게의해 철저한 **베타 테스트** 실행:하드웨어, OS, 주변장치, 네트워크 배치)
- **지원체제는 반드시 소프트웨어가 배포되기 전에 확립**되어야 한다 .
 - 지원 : 계획되고 , 지원요소 준비 , 적절한 기록 유지 메커니즘 구축
 - > 소프트웨어 팀은 지원 종류 평가 실행
- **적합한 교육적 자료는 반드시 최종 사용자에게 제공**되어야 한다
 - 훈련 자료 : 개발 -> 문제 해결 가이드 라인, 소프트웨어 증분 차이점 설명서 지원
- **결함이 있는 소프트웨어는 먼저 수정된 뒤 배포**되어야 한다
 - 낮은 고품질 제공 (망각) Vs. 적절한 저수준 품질 제공 (상기).



4. Project Planning

◆ 프로젝트 관리

- SW 를 개발하고 관리는 조직의 요구사항에 일치
- SW 를 적정한 시간 (스케줄) 에 의해서 잘 인도되는 것을 보증 하는 활동

◆ 관리자는 관리 항목 에 대하여 책임

- Proposal writing (제안서 작성)
 - 프로젝트 목표 / 수행방법 기술 , 비용 / 일정 추정치 , 특정 팀 계약 합리화
- Project planning and scheduling (프로젝트 계획 수립 및 일정 관리)
 - 제품생성 활동 , 기준 , 이정표 , 산출물 식별
- Project costing (프로젝트 비용 산정)
 - 요구되는 자원 추정
- Project monitoring and reviews (프로젝트 감시 및 검토)
 - 진행상황 추적 , 계획 대비 진도와 비용 비교 , 자원이 조직 목표와 일치 검사 , 변경
- Personnel selection and evaluation (인력 선발 및 평가)
 - 숙련된 인력 참여 , 제약조건 (Expert, 인력 양성) 에서 인력 선발
- Report writing and presentation (보고서 작성 및 발표)
 - 상세 프로젝트 보고서 -> 추상적이고 간단한 요약 보고서 작성 , 효율적 표현.

4. Project Planning

◆ 프로젝트 진행을 철저하게 계획

- 소프트웨어 프로젝트의 효율적인 관리를 가능하게 함

◆ 프로젝트 계획

- 프로젝트 시작부터 끝까지 계획
- 문제가 발생할 것을 예측해서 해법까지 준비
- 프로젝트 시작할 때의 계획은 그것을 추진하는 원동력
- 프로젝트 진행 중에 계획 수정 가능
- 프로젝트 진행 중에 더 좋은 정보의 추가 가능
- Type of project plan

품질 계획	품질 절차와 표준을 기술 (작성)
확인 계획	검증에 사용되는 방법 , 자원 , 절차를 기술
형상관리 계획	형상관리 절차 , 구조를 기술
유지보수 계획	유지보수 요구사항 , 유지보수 비용 , 노력 예측
직원발전 계획	팀 구성원 기술과 경험 향상 방안을 기술

4. Project Planning

◆ 프로젝트 계획 프로세스

프로젝트 제약조건 설정

프로젝트 인자의 초기 추정

프로젝트 이정표와 산출물 정의

While 프로젝트가 완료되지 않거나 취소 **loop**

프로젝트 일정 결정

일정에 따라 활동 시작

(잠시) 대기

프로젝트 진척사항 검토

프로젝트 인자 추정치 조정

프로젝트 일정 갱신

프로젝트 제약조건과 산출물 재협상

if (문제가 발생) **then**

기술적 검토 및 가능한 수정 시작

endif

end loop

제약조건 (기간 , 인력 , 예산)

인자 (프로젝트 구조 , 규모 , 기능분산)

milestone, 보고서

반복

일정추정 , 계획에 따라 프로세스 활동 정의

2~3 주

진척사항 확인하여 차이점 기록

인자들의 초기 예측치 불안전으로 수정필요

프로젝트와 일정을 원래 가정 수정

지연이 되면 재협상

재협상 실패 , 일정 불만족

제약조건 내에서 일정 만족키 위한 대안

4. Project Planning

◆ 프로젝트 계획 보고서

1. Introduction
(개요)
2. Project organization
(프로젝트 조직)
3. Risk analysis
(위험 분석)
4. Hardware/software resource requirements
(하드웨어 소프트웨어 자원 요구사항)
5. Work breakdown
(업무 분할)
6. Project schedule
(프로젝트 일정)
7. Monitoring and reporting mechanism
(감시와 보고 체계)

1. **목표**를 간단히 기술, 프로젝트에 영향을 미치는 **제약 조건 (예산, 시간 등) 설정**
2. 개발팀의 **구성** 방법, 참여할 **사람** 및 그들의 **역할**을 설명
3. 가능한 프로젝트 **위험 (risk)**, 위험이 생길 **가능성**, 위험 **감소 전략**을 제안
4. 개발에 필요한 **하드웨어 및 소프트웨어**를 설명, **구입 가격과 납기 예측**
5. 프로젝트의 **업무**들을 나누고, 각 업무별로 **이정표 (milestone) 와 산출물**을 정의
6. 업무 사이의 **관계**를 설명, 각 업무의 **이정표를 도달하기 위해 예상 되는 시간, 사람** 등을 배정
7. 생성해야 할 **관리 보고서를 정의**, 보고서가 만들어졌을 때 **프로젝트 감시 메커니즘**이 사용.



Homework

◆ Chapter7. 모델링-실무가이드 원칙

7.1 프로세스를 가이드하는 원칙 정의

7.2 실무를 가이드 하는 원칙

7.3 각 프레임워크를 가이드하는 원칙

- 의사소통/계획 원칙

- 요구사항/디자인 모델링 원칙

- 구축/배치 원칙

7.4 프로젝트 계획 보고서 내용



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 모델중 1개를 선정 및 이유

3.2 특수한 프로세스 모델중 1개를 선정 및 이유

4장 실무 가이드 원칙

4.1 각 프레임워크를 가이드하는 원칙에서 원칙별로 중요한것 각 3개씩 정의[HW 7.3 항목 중심으로 작성]

4.2 프로젝트 계획 보고서 작성