<Chapter 2. 프로세스와 방법론>

방법론 : 정의된 작업들을 어떤 순서로 어떤 방법으로 하는가를 다루는 것.

프로세스 없는 소프트웨어의 개발 : CODE AND FIX. 개발 -> 고객 혹은 개발자가 만족할 때까지 수정 -> 개선을 위한 아이디어 짜내기 -> 수정 …반복. 설계나 계획, 목표가 없다. 체계적인 test작업이나 품질보증 차원의 활동에 대한 필요성 인식이 없다. 그래서 개발과 유지보수에 비용이 많이 들고 문제점이 늦게 발견될 수록 비용이 더욱 커진다.

프로세스와 방법론 비교 : p.46

-프로세스 특징: 절차적이고 이론적이다.

단계적인 작업의 틀을 정의한것, 무엇을 하는가에 중점, 결과물의 표현에 대하여 언급 없음, 패러다임에 독립적, 각 단계가 다른 방법론으로도 실현 가능. Ex)폭포수 프로세스, 나선형 프로세스, 프로토타이핑프로세스, unified 프로세스, 애자일 프로세스

-방법론 특징: 실천 방법이다.

프로세스의 구체적인 구현에 이름, 어떻게 하는가에 중점, 결과물을 어떻게 표현하는지 표시, 패러다임에 종속적, 각 단계의 절차, 기술, 가이드라인 제시. Ex) 구조적 분석, 설계 방법론, 컴포넌트, 객체지향 방법론, 애자일 방법론

소프트웨어 생명주기(=life cycle) 5단계: 소프트웨어 개발에 대한 기술적, 관리적 이슈를 다루는 작업.

요구분석-> 설계 -> 구현 -> 테스트 -> 유지보수

각 단계가 서로 협력하여 전체 목적을 만족한다.

프로세스(절차)(단계) : 소프트웨어 시스템을 구축하기 위해 수행되는 작업의 단계 소프트웨어 개발에 대한 기술적, 관리적 이슈(시간,비용,사람…)를 다루는 작업. (p.48그림 참고-프로세스명세와 프로세스 모델)

프로세스의 종류 : 1. 개발프로세스 2. 관리 프로세스 3. 기타 프로세스

(p.49그림 참고 – 소프트웨어 프로세스의 종류)

-개발프로세스 : 소프트웨어 개발에 직접 관여함. (기술적 이슈)

-관리프로세스 : 비용, 시간등을 맞추기 위한 작업(비기술적 이슈)

-기타프로세스 : 형상관리프로세스 + 프로세스관리 프로세스

개발 프로세스의 정의 : 프로젝트는 단계로 이루어지며 각 단계는 프로젝트의 목표를 만족시킬 수 있는 잘 정의된 작업으로 정의되어야한다. 프로세스는 비용을 줄이기 위해 그 단계에 유입된 결함을 찾아내는데 초점을 둔다. (너무 많은 단계로 나누지 않아야함) 작업결과, 검증조건,작업방법,출구조건,진입조건을 명확히 기재해야함.

(p.51 그림 꼭 참고)

프로세스 모델 : 이론적인 프로세스를 실제로 적용시킬 구체적인 방법. Ex) 폭포수, 나선형, 진화적, 프로토타이핑모델, 애자일 , unified 프로세스가 있다.

-폭포수 모델 : 가장 오래되고 가장 널리 사용된 프로세스 모델.

각 단계가 다음 단계 시작전에 끝나야함. 중복없고 순서적이고 당므단계 시작전 점검한다. 역류존재x(피드백 x). 직능 중심의 프로젝트 조직에 적합. 크고 오래 유지되어야한다면 적합하다. 폭포수 모델 확장한게 v 모델 (p.57그림참고)

장점 : 프로세스가 단순하여 쉽게 적용가능, 관리 용이, 코드 생성전 충분한 연구,분석을 거침.

단점 : 소용없는 다종의 문서 생성가능성있음, 변경불가, 테스트를 시스템 완성후에야 시작

프로토타입 : 견본, 시제품

-프로토타이핑 모델 : 피드백 받기 위해 프로토타입을 만들어서 사용자에게 보여주고 평가하게 한다. 개발착수시점에 요구가 불분명할때, 혁신적 기술을 써보고 싶을 때, 실험적으로 실현가능성을 타진해보고 싶을 때 적합. = > risk관리

프로토타이핑 도구 : (UI위주) 화면생성기, 시스템의 작동을 시뮬레이션해서 보여줌.

공동 참조 모델의 역할 수행(사용자와 개발자의 의사소통을 보여주는 좋은 매개체)

목적 : 단순 요구 추출 -> 만들고 버림(일회용), 개발 단계에서 지속적인 유지보수 (진화형), 개발자 입장에서 문제 이해 제대로 했는지 확인가능. (p.59그림 참고)

장점: 사용자 의견 반영 잘 됨, 사용자 참여율 높일 수 있고 개발자는 사용자의 요구를 정확히 도출.

단점 : 기대심리 유발, 관리 어렵고 개발비용 증가

-나선형 모델 : 소프트웨어의 기능을 나눠서 점진적으로 개발해서 실패위험이 낮고 테스트 용이하고 피드백 반영가능. 여러번의 릴리즈가능(업데이트가능), 재정적/기술적으로 위험 부담이 큰경우 혹은 요구사항/아키텍쳐 이해가 어려운 경우 적합함.

단계 : 목표,방법,제약조건 결정 -> 위험요소 분석 및 해결 -> 개발과 평가 -> 다음 단계의 계획 을 반복. (p.61참고)

장점: 리스크관리에 좋아서 대규모 시스템에 적합. 한 사이클에 기능 추가 못했으면 다음 단계에 하면 됨.

단점: 관리 복잡, 성공사례 적음, 위험분석 모르고 지나치면 리스크 높음.

-진화적 모델 : 개발 사이클이 짧아서 빨리 출시해야할때 좋음. 시스템 나눠서 릴리즈 하기 때문에 시간이 단축된다.

구성 방법 : 점증적 방법 + 반복적 방법

점증적 방법 : 기능별로 릴리즈.

반복적 방법 : 릴리즈 할 때마다 기능 완성도를 높인다.

p.62

장점 : 기능 좀 부족하더라도 초기에 사용교육 가능, 유저 요구 빨리 반영해서 빨리 문제 해결 가능. 시장에 빨리 나올 수 있다.

단점: 관리 복잡해서 큰 프로젝트에는 적합하지 않다. 실패가능성이 높음. 위험분석에 크게 의존함.

-Unified 프로세스(=up)(=소프트웨어통합프로세스) : 여러개의 사이클로 구성되며 각 사이클은 시스템을 출시함으로 종결됨. (p.63).

과정은 네 단계. 도입 -> 정련 -> 구축 -> 전환.

유스케이스 : 비지니스 프로세스 모델링한것.

도입 : 간단한 유스케이스 모델/소프트웨어구조/프로젝트계획작성. 1~2회정도 반복

정련 : 대부분의 유스케이스 작성, UML다이어그램 그림. 여러번반복

구축 : 남아있는 유스케이스 구현하고 시스템에 통합.

전환 : 시스템 배치, 테스팅, 사용자 교육.

장점: 방법론과 프로세스가 문서화가 잘 되어있다, 코드 재사용성이 좋다. 사용자 요구 변경, 관련된 리스크 적극해결. 통합을 위한 시간, 노력 줄일 수 있다.

단점: 프로세스가 너무 복잡하다. 사용자의 협동, 의사소통 가이드라인이 없다. 조직화되지 않은 개발.

-애자일 프로세스 : 2~6주간의 짧은 주기로 개발 반복. 단계적으로 시스템 전체를 완성함.

애자일 프로세스에서 중요한 점: 형식적인 문서보다는 커뮤니케이션을 통해 나아감. 사용자는 문서가아닌 실행되는 소프트웨어를 통해 요구를 확인. 사용자의 요구는 프로젝트 중간에 바뀔 수 있음. 각 반복주기의 반성의견을 다음계획에 포함시킴.

Ex)익스트림 프로그래밍(=극단적 프로그래밍) :

요구사항이 왔다갔다하는 소규모team. 사용자 스토리(기능/요구 우선순위), 매일 빌드와 통합(실행파일 만듬), 테스트 주도 개발(테스트함), 페어프로그래밍 (p.67)

Ex)스크럼 :

개발팀원 모두가 소통하고 협력해 짧은 주기를 반복하여 소프트웨어 개발하는 작업/결과물. 짧은주기(스프린트 = 주기이름), 백로그에 우선순위 부여. 단점으로는 문서가 없다는 것.

-형상관리프로세스(지원프로세스중하나임) : 개발 중 발생하는 변경을 체계적으로 컨트롤. 개발작업과 독립적인 방법. (p.74)

직능조직: 분석만 하는팀, 설계만 하는팀 처럼 그 분야의 전문성을 가진 조직이 있음.

지원 프로세스 : ISO(국제 표준화 기구)/ICE 12207에서의 프로세스 그룹.

(관리, 엔지니어링, 계약, 운영 지원에 대한 지원 프로세스 존재.)

-관리 프로세스 : 관리하는데 필요한 모든 작업 관리(계획(목표,비용,일정,중간점검 계획), 모니터링과 제어, 분석(목표와비교)), 모니터링과 제어가 제일 시간 많이 걸림(개발 프로세스 전기간), 계획은 시작전에함. (p.71)

-품질보증 프로세스 : 지원 프로세스중 하나. 프로세스와 프로덕트에 대한 품질을 관리하고 향상시킴.

프로세스 관리 프로세스 : p.73그림 참고

검사 프로세스 : 개발 결과에서 결함 방지/ 결함 찾음, 정의된 프로세스에 따라 동료 그룹이 작업결과 검사.

검사와 테스트의 차이 : 검사 = 소프트웨어 코드뿐만아니라 다양한 것 검사함. 테스트 = 소프트웨어 코드 검사.

방법론 : 가장 디테일한 방법을 제시 1. 구조적 방법론 2. 정보공학 방법론 3. 객체지향 방법론

소프트웨어 프로세스의 각 작업을 어떻게 수행할것인지 정의.

방법론과 프로세스의 차이 : 일반적으로 작업만을 명시. 관계는 나타내지않음.

패러다임: 큰 개념이 바뀐 것

-구조적 방법론 : 객체지향 방법론과 대비됨. 70~80년대에 많이 사용됨. 분리와 정복원리 적용. 자료 흐름도를 구조도(모듈사이의 관계를 나타내는 그래프)로 변경하는 과정. 운영체제, 시스템 프로그래밍에 이용되고 있음.

-정보공학방법론: 기업중심, 전략적 시스템 계획 중심, 데이터중심, 분할과 정복, 공학적 접근, 사용자의 적극적 참여. (p.77)

-객체지향방법론 : 클래스를 모델링 함. 객체(=대상)을 묘사함. 자료와 메서드를 가까운 곳에 정의해 객체로 묶어둠. 객체 사이에 메세지를 호출하여 원하는 기능을 담당하게 함.

-세가지 방법론의 비교 : p.79참고.