<Chapter1>

소프트웨어공학: 소프트웨어를 개발하기 위해 체계적으로 접근하고 원리를 적용하는 것. 소프트웨어 개발과 운영, 유지보수, 소멸에 대한 체계적인 접근법을 사용. 반복사용 가능! 여러가지 원리와 방법을 적용하여 최소의 비용으로 고품질의 소프트웨어를 계획된일정에 맞추어개발.

소프트웨어공학의 목표 : 품질 높임, 생산성 높임, 복잡도 낮춤, 비용 최소화, 개발기간 단축, 대규모 프로젝트 관리

소프트웨어공학의 주제 : 1. 단계적 프로세스2. 품질보증 3. 프로젝트 관리

-단계적 프로세스 : 개발주기 = 계획->분석->설계->구현->테스팅및통합->유지보수.

소프트웨어 위기를 해결하기 위한 소프트웨어 공학의 핵심 접근 방법.

코딩에 치중하지 않고 요구분석, 설계, 코딩, 테스팅등 정해진 절차를 따라 작업.

-요구분석 : 소프트웨어 시스템이 해결해야 할 문제를 이해하기 위한 작업. 문제를 분석하고 사용자의 요구를 추출하고 잘 정리한다. 시스템이 제공해야할 기능,성능,제약사항등을 파악 [결과물: 요구분석명세서]

-설계 : 요구 명세에 기술된 문제의 솔루션을 기술한다.[결과물: 설계명세서]

-코딩 : 설계를 기초로 시공하는 작업.주로 프로그래밍 언어로 원시코드를 작성[결과물: 새 시스템, 유지보수 계획]

-테스팅 : 개발된 sw에 결함이 있는지 검토하고 확인한다. [결과물: 테스팅결과보고서]

-품질 보증: 요구분석 -> 설계 -> 코딩-> 테스팅 -> 유지보수까지의 과정들을 모두 검토한다. 즉 품질보증은 개발하고있는 소프트웨어가 요구와 품질 수준을 만족시키는지 검토하고 확인, 테스팅하는 작업이다. (그림 p.31). output으로 요구분석명세서(SRS),원시코드,소프트웨어가 나온다.

-프로젝트 관리: 프로젝트 계획, 자원관리, risk관리, 프로젝트 수행과 모니터링으로 프로젝트를 관리함. (프로젝트의 세가지 제약(=프로젝트 삼각형): 시간, 비용, 범위.)

-프로젝트 계획 : 개발 시작전에 수행되는 작업. 범위를 결정하고 타당성을 검토하며 소프트웨어 개발이 어떤 방향으로 어떤 방법과 일정, 조직으로 이루어질지 계획

-자원 관리 : 소프트웨어 제품을 개발하는 데 사용된 인력, 도구, 하드웨어는 해당 프로젝트의 자원으로 간주됨.

-리스크 관리 : 프로젝트를 실패로 끝나게 할 수 있는 위험 요소를 예측하고 이를 식별, 분석하여 대비책을 세우는 등의 관리 활동

-프로젝트 수행과 모니터링 : 프로젝트가 시작되면 모든 작업이 계획에 따라 진행되고 있는지 확인하기 위해 모니터링을 한다. 모니터링은 작업의 상태를 파악하여 위험을 확인하고 위험을 해결하거나 조치를 취하는 것. 진행 상태를 정기적으로 파악하여 보고서를 작성하고 중요 시점에 체크리스트를 만들어 점검.

소프트웨어 공학의 연구결과 :방법, 도구, 프로세스, 패러다임으로 나눌 수 있다. p.33

-방법: 소프트웨어 제작에 사용되는 기법이나 절차

-도구 : 자동화된 시스템

-프로세스 : 도구나 기법을 이용하여 작업하는 순서

-패러다임 : 접근 방법, 스타일

좋은 소프트웨어가 가지는 특징 : 평가하는 입장에 따라 다를 수 있지만 대개 품질이 높고 사용자 요구를 잘 만족시키며 결함이 포함되지 않는 소프트웨어를 좋은 소프트웨어라고 한다.

-

컴퓨터 과학 vs 소프트웨어 공학 : 컴퓨터과학은 계산적 효율, 자원 공유, 정확성, 최적화, 성능을 강조하고 이를 개선하기 위한 방법, 알고리즘, 원리, 기술을 다룬다.(기술적인 측면에만 집중). 소프트웨어공학은 생산성, 품질, 비용, 시간을 강조한다. 소프트웨어 공학의 모든 방법, 프로세스, 도구는 이러한 네 가지 목표를 개선하기 위한 것임.(기술 외적인 측면도 고려해야함) (그림 p.36)

소프트웨어 : 프로그램, 프로그램의 개발, 운용, 보수에 필요한 정보 일체. 개념적이고 무형적(실체가 없다).

소프트웨어 특징 네 가지 : 1. 복잡성 2. 순응성 3. 변경성 4. 비가시성

복잡성 : 내면이 복잡, 순응성 : 요구조건에 따라간다, 변경성 : 처음에 만들어진것과 달라질 수 있다. 비가시성 : 실체가 없어서 보여지지않는다.

소프트웨어의 종류(3종류) : 1. 주문형 소프트웨어 2. 패키지형 소프트웨어 3. 임베디드 소프트웨어

-주문형 소프트웨어 : 특정 고객/기업의 요구를 만족시키기 위해 제작 사용되는 카피의 수 적음, 요구되는 하드웨어 성능 낮음, 개발 인력 많음.

-패키지 소프트웨어 : 패키지화하여 상업적으로 판매하는 소프트웨어 ex.워드프로세서, pos시스템,회계관리시스템…

사용되는 카피의 수는 중간, 하드웨어 성능 높음, 개발 인력 중간

-임베디드 소프트웨어 : 다른 시스템에 내장된 소프트웨어. 사용되는 카이의 수 많은 요구되는 하드웨어 성능은 중간, 개발 인력은 적음.

이거 말고도 범용 소프트웨어를 구입해서 커스텀하는 방법도 있긴한데 비용 적게 드는 대신 범용 소프트웨어의 새 버전이 출시되면 또 커스텀화 해야하는 위험이 있다.

시스템 : 필요한 기능을 실현시키기 위한 관련 요소를 어떤 법칙에 따라 조합한 집합체 ex.은행의 업무 전산화에 사용되는 sw는 현금 자동 인출기

시스템의 네 가지 중요한 성질 : 1. 서브시스템 2. 기능적 분할 3. 시스템 경계 4. 자동화 경계 (시스템의 요소 그림 p.21)

소프트웨어 개발작업의 기본활동(=lifecycle)(=생애주기) :

명세화 -> 설계 -> 코딩 -> 검증 -> 유지보수 -> 명세화 …. 반복!

(명세화,구현,검증,유지보수가 기본활동임) (그림 p.22)

소프트웨어 개발작업의 특징: 1. 명세화의 어려움 2.재사용의 어려움 3. 예측의 어려움 4. 유지보수의 어려움 5. 고품질의 어려움

-명세화의 어려움: 고객의 의도를 처음부터 정확히 설명하고 이해하기 어렵다

-재사용의 어려움 : 미리 재사용할 부품을 만들기 어렵다

-예측의 어려움 : 물리 구조는 물리법칙을 따르지만 소프트웨어는 여러가지 외부 요소의 영향을 받아 견적을 내거나 규모를 예측하기 어렵다

-유지보수의 어려움 : 다른 사람이 개발한 sw를 쉽게 이해하기 어렵고 이해해서 고치더라도 다른 부분에서 오작동을 일으킬 수 있다.

-고품질의 어려움 : 소프트웨어는 한 줄의 코드라도 스스로 동작하는 것은 없다. 한 줄의 코드를 수정하더라도 테스트할 내용이 많아지고 모든 실행 조합을 다 테스트할 수는 없다.

소프트웨어의 위기 : sw수요가 급증하고 복잡성이 증가함에 따라 기존 방법이 충분하지 않아서 발생 (같은 인력, 같은 방법, 같은 도구를 쓰기 때문)

소프트웨어 공학의 접근법 : 즉흥적인 개발의 문제점을 극복한다. (즉흥적인 개발은 예산 초과, 개발지연, 낮은품질, 재작업이나 유지보수가 곤란하다는 단점이 있다. 프로그래밍 -> 만족할때까지 수정 -> 개선을 위한 아이디어짜내기 -> 개발 .. 반복)