Software Engineering

저자 Roger S. Pressman, ph.D.

프리젠테이션 Jang Yong-un

목표

- 현재 Software engineering의 현황을 살펴본다.
- 프로세스 진행 과정을 알아본다.
- 기본적인 Software engineering 활동들과 각 활동들의 진행과정을 살펴본다.
- 앞으로의 Software engineering 활동의 미래
  에 대해 생각해 본다.

1. Software Engineering
-계층화된 기술

- Software Engineering의 정의
  - 실제 장치에서 효율적으로 작동하고 신뢰할 수 있는 경제적인 소프트웨어를 얻기 위한 적절한 공학 원리를 사용하거나 제정하는 것.(-Fritz Buer)

  - 한편 층을 이룬 계층화된 기술이기도 하다.

1 Software Engineering
-계층화된 기술

Software engineering layer
Tools
Methods
Process
A Quality focus

1. A Quality focus : 가장 기본이 되는 부분.

2. Process : 각 계층을 결합해 주고, 합리적인 소프트웨어를 신속하게 개발
   할 수 있게 해준다.(프로젝트 통제, 어떤 방법이 적용되는지, 이정표역할)

3. Method : 소프트웨어를 어떻게 만들지를 정한다.

4. Tools : Process나 Method를 자동, 반자동적으로 지원

2. A Process Framework

- A Process Framework
  - 완성된 소프트웨어 process를 위한 기초가 된다.
    - Communication : 고객과의 의사소통, 협력, 요구 수용
    - Planning : 기술적인 업무, 위험요소, 요구되는 자원, 일정 등을 계획
    - Modeling : 고객의 요구를 더 잘 이해하고 요구에 대한 달성 계획을 설명 위한 모델 만들기
    - Construction : 코딩과 테스트
    - Deployment : 배포, 사후관리

2. A Process Framework

- Umbrella activities

  : process framework을 보충하기 위한 활동
  - Software project tracking and control
  - Risk management
  - Software quality assurance
  - Formal technical review
  - Measurement
  - Software configuration management
  - Reusability management
  - Work product preparation and production

2. A process Framework

- 프로세스 모델
  - 예기치 못한 상황에 대한 유동성이 중요하다.
  - 여러 모델들의 몇 가지 기초적인 부분은 미묘하게 차이가 난다.
    - 전반적인 활용 업무의 흐름과 상호독립성
    - 각 프레임워크 활용에서 업무들이 정의된 정도
    - 각 작업 결과물이 정의되고 요구된 정도
    - 어떤 품질 보증 활동이 적용되었는가
    - 진행의 세밀함과 엄밀한 전반적인 정도
    - 사용자와 의뢰인의 프로세스 연과 정도
    - 팀간의 역한 분담 정도

3. Software process model

- Software process model은 개발 전략을 일컫는다.
- 규정된 모델
  - The Waterfall Model
  - Incremental Process Models
  - The Spiral Model
  - Specialized Process Models
  - The Format Methods Model
  - The Unified Process

3. Software process model

- The Waterfall Model
  - 프로그램에 대한 고객의 요구와 문제점이 잘 이해될 때 사용한다.
    - 가장 오래된 소프트웨어 공학 패러다임
    - 그러나,
      - 실질적인 프로젝트와 맞지 않다.
      - 고객과의 정확한 요구들을 알기 힘들다.
      - 프로젝트 진행 기간이 길다.

3. Software process model

- Incremental Process Model
  - Waterfall model을 반복하는 모델, 필요에 따라 소프트웨어 정교화하는 모델.
    - 소프트웨어 개발 시간을 고객의 요구에 맞출 때 용이하다.
    - 기술적인 발전에 보조를 맞출 수 있다.

3. Software process model

- Evolutionary Process Model
  - 반복되는 일련의 활동을 통해서 점차적으로 완성된 버전으로 개발하게 하는 모델
    - Prototype
      - 고객과의 대화
      - 계획과 모델링 디자인(사용자가 보게 될 것들)
      - 견본 생성
      - 견본을 평가 받음
    - 그러나,
      - 소비자로부터의 지속적인 간섭을 받게 된다.
      - 빠른 개발을 위해 적절하지 못한 방법을 사용한다.
    - 하지만 서로 합의 상태에서 일을 하므로 소프트웨어 품질을 높일 수 있다.

3. Software process model

- The Spiral Model
  - 앞선 모델의 반복되는 Prototype과, waterfall모델의 체계적인 특징을 결합했다.
    - 동그란 형태의 모델로 각 구역마다 활동이 진행된다. 각 구역이 지날 때마다 위기를 관리 한다.(결과물 조합, 개발상황관리)
    - 프로젝트 관리자는 반복횟수를 설정한다.
    - 하지만,
      - 전문적인 지식이 필요하다.

3. Software process model

- Specialized Process Model
  - 좀더 세부적이고 전문적인 접근 모델들
    - Component-Based Development
      - Spiral model의 특징을 가지고 있으나, 소프트웨어 구성 요소(목적에 맞게 잘 설계된 인터페이스를 갖춘 요소)를 조합 한다는 점에서 다르다.
    - The Formal Methods Model
      - 형식적 모델.
      - 엄격하고, 엄밀한 인용을 통해 애매모호함, 불안전성, 불일치 등을 찾아내서 고칠 수 있다. 그러나, 오랜 시간과 높은 수준의 전문 지식이 필요하다.

3. Software process model

- Specialized Process Models
  - Aspect-Oriented Software Development
    - 과거 사용된 프로그램의 특징들을 이용한다.

      이러한 특징들에 대한 프로세스, 방법론적 접근을 정의하고, 디자인하고, 생성

      ->한마디로 지름길을 만든다.

3. Software process model

- The Unified Process
  - 고객 대화를 통한 요구를 파악하는 능률적 방법에 중점을 둔 프로세스
  - 소프트웨어 구조와, 올바른 목표설정
  - 반복하면서, 발전하며, 진화한다.
    - 과정
      - Inception phase : 고객과의 대화, 계획 활동
      - Elaboration phase : 고객과의 대화, 모델링 활동
      - Construction phase : 표준 소프트웨어 프로세스를 정의한다.
      - Transition phase : 제작 활동, 소프트웨어 테스트
      - Production phase : 소프트웨어의 배포, 피드백 활동이 동시에 일어난다.

3. Software process model

- Agile Software Development
  - 프로세스와 툴 각각의 *독립과 상호작용*
  - 이해도 높은 문서화 *작업 소프트웨어*
  - 협의에 대한 *고객 합동*
  - 앞선 계획과 다른 *변화 대응*
    - 아래와 같은 상황에서 특징을 갖는다.
    - 고객의 요구가 일관 되지 않는다.
    - 각 과정 사이에 어떤 일들이 필요한지 명확하지 않다.
    - 분석, 디자인, 개발, 시험이 우리가 원하는 방향으로 이루어지지 않는다.

3. Software process model

- Agile Software development
  - How do we create a process that can manage unpredictability?

  - 대표적인 Agile Software model XP
    - 4가지 frame work
      - Planning : 고객이 story를 작성하고, 우선순위를 둔다. XP팀은 견적을 낸다.
      - Design : SPIKE SOLUTION, REFACTORING
      - Coding : TEST CREATION-IMPLEMENT, PAIR PROGRAMMING
      - Testing : ACCEPTANCE TEST(BY CUSTOMER)

4. THE MANAGEMENT SPECTRUM

- 3P를 관리한다.
- 구성
  - People : Management Maturity model
  - The problem : joint application design
  - The Process : CMMI Guidelines을 준수
    - (1) continuous model
      - 특별한 목표나 실행 그리고 각 진행 분야 판단
        - 진행분야(프로젝트 계획 요구 관리, 측정, 분석, 환경설정 관리) 능력을 측정
    - (2) staged model
      - (1)과 같으나 성숙도를 측정한다. 성숙도를 충족하기 위해선 목표설정이나 실행에 있어서 각 진행분야가 만족되어 있어야 한다.
    - (3) CMMI model
      - Continuous model과 유사하나 능력이아닌 각 프로세스 분야의 성취 정도와 연관된다.

5. Software Project Mangement

- 소프트웨어 프로젝트 관리
  - 견적, 위험 관리, 스케쥴링, 통제 활동
  - 측정과 측정기준들
    - 소프트웨어 품질에 대한 지표제공
    - 분석, 품질확인이나 디자인활동을 할 때 사용된다.
  - 프로젝트 견적
    - 제출기한과 필요한 예산을 측정해야 한다.
      - 방법 Effort estimation techniques : 각 기능을 성취하기 위한 표를 통한 수치를 통해 측정
      - Size-Oriented Estimation : 코드라인 수이나 기능을 통해 측정
      - Empirical Models : 관행을 이용한다.

5. Software Project Management

- 위험 분석
  - 앞으로 있을 지 모르는 위험을 평과, 확인, 우선 순위, 관리해야 한다.
  - 이 분석목표는 실제로 일어날 위험을 확인하고 이에 대한 영향과 이 영향을 최소화하는 것이다.
- 스케쥴링
- Tracking and Control
  - 잘 정의된 진행 틀은 Tracking을 용이하게 한다.
  - 품질과 변화를 통제한다.

6. Software Quality Assurance

- 품질보증에는 기능 요구, 문서화된 개발 기준 그리고 함축적인 특징과 같은 요소가 포함된다.
- 품질 확인 사항들
  - 요구사항을 준수할 것.
  - 개발 기준이 정의되며 이를 준수할 것
  - 명시되지 않은 요구들을 충족할 것(유지 보수와 같은 것을)

6. Software Quality Assurance

- 3가지 기준과 요소 –McCall
  - 3가지 기준
    - (1) 제품 기능, (2) 제품 수정 및 교정, (3) 제품의 환경에 따른 유동성.
  - 몇 가지 요소
    - 정확성, 신뢰성, 효율성, 실용성, 유지가능성, 유연성, testability, 이동성, 재 사용성, 정보상호 응용성
- 그외
  - 품질을 확인하기 위한 기술진, 소프트웨어로 직접 시험.
  - 변화가 소프트웨어 품질유지를 어렵게 한다.

7. 소프트웨어 구성 관리

- 변화 인식, 변화에 대한 대응, 변화통제 예술

  변화에 대한 실수를 줄임으로써 생산성을 극대화한다.(By Babich)
- 구성 관리 순서
  - (1) 변화 감지
  - (2) 변화 통제
  - (3) 적절히 수행되는지 확인
  - (4) 변화 기록 및 배포

8. The Technical Spectrum

- 산발적인 기술을 탈피하여 정의된 기술들이 등장했다.
- Software Engineering Methods-기조방향
  - Conventional Software engineering methods : 일반적인 정보 입력 출력
  - Object-oriented approaches :
    객체간 상호작용
  - Formal methods

8. The Technical Spectrum

- Problem Definition
  - 고객과 협의
    - (1) 개발자와 사용자간의 시스템에 대한 견본 만들기
    - (2) 데이터, 기능, 형태를 표현하는 자세한 분석 모델 만들기

8. The Technical Spectrum

- Analysis Principles
  - 여러 방법이 있지만 일정한 원칙들이 지켜지고 있다.
    - 데이터 범위 모형화 : 사용자가 접할 데이터, 데이터 간의 관계 분석
    - 기능적 형식 모형화 : 객체인가? 그저 엄밀한 절차로 기능을 표현할 것인가?
    - 시스템의 행동 : 외부요인들, 결과에 대한 반응
    - 데이터, 기능, 행동 모델 분할 : 복잡한 것들을 간단히 할 수 있다.
    - 복잡한 진행에서 기초적인 표현으로.

8. The Technical Spectrum

- Analysis Methods
  - 유사점들
    - 분석 방법은 데이터간 관계, 데이터 자체에 대한 제공.
    - 프로그램이 데이터를 다루는 방법, 데이터와 기능 조합
    - 프로그램의 목적을 알 수 있게 해줌
    - 프로그램의 세부적 접근을 가능하게 해줌
    - 프로그램 설계 주요 내용을 알게 해줌

8. The Technical Spectrum

- Analysis Method
  - Analysis Method를 적용한 analysis Model의 4가지 요소
    - 시나리오 중심 요소 : 유저 중심
    - 클래스 중심 요소 : 객체 중심
    - 행동 요소 : 사건을 어떻게 대처하는지
    - 흐름지향 요소 : 데이터의 흐름 중심

8. The Technical Spectrum

- 분석 방법은 각자 자신만의 디자인 수행한다. 이 방법은 아래와 같은 원리들을 갖는다.
  - 데이터를 관리하는 알고리즘 생성
  - 각각의 알고리즘이나 자료구조는 서로 감추어져 있다.
  - 독자적 기능을 가진 요소들은 서로 독립적이다.
  - 소규모의 알고리즘 요소를 통해서 디자인 되어야 한다.

8. The Technical Spectrum

- The Design Pyramid
  - 특별한 디자인을 알아보기보단 기본적인 틀을 알아본다.
    - 구조적 디자인은 계층적 디자인과 객체간의 소통 디자인이 있다.
    - 인터페이스 디자인은 사람, 컴퓨터간의 인터페이스와, 컴퓨터 시스템간 인터페이스, 소프트웨어 구성물간의 인터페이스가 있다.
    - 구성요소 수준의 디자인은 알고리즘 절차 설계가 있다.

8. The Technical Spectrum

- Program Construction
  - 앞으로 프로그램 개발에서 언어에 대한 문제보단, 분석과 설계에 있어서의 개발의 문제가 중요한 영향을 끼칠 것이다.

8. The Technical Spectrum

- Software Testing
  - Testing의 목적(by- Glen Myers)
    - 에러를 찾는다.
    - 아직 발견되지 않은 높은 가능성의 에러를 찾는다.
    - 에러를 수정한다.
    - 신뢰도를 높인다.

      - 적은 노력과 시간을 가지고 클래스간의 에러를 체계적으로 찾는다.

8. The Technical Spectrum

- Software Testing Strategy
  - 기능단위로 test 할 것 그 후 전반적인 시스템단위로 테스트
  - 다른 Testing 기술을 동시에 사용할 것
  - 개발자가 테스팅을 지휘할 수도 있으나 대형 프로젝트는 독립적인 테스팅 그룹이 시험한다.
  - 디버깅은 기본.

8. The Technical Spectrum

- Software Testing Strategy
  - (1) 기능이 잘 작동하는지 확인

  ->back box testing
  - (2) 기능이 명세한 대로 돌아가는지, 내부적인 활동들이 잘 진행되는지 확인

  ->white box testing

9. Software Engineering Patterns

- 소프트웨어 진행에서의 일정한 틀을 제공해 준다. 완성된 소프트웨어 프로세스를 만드는데 도움을 준다.
- Process Patterns
  - 추상적인 진행에서 세부적인 진행까지의 일련의 반복되는 과정.
- Analysis Patterns
  - 프로젝트마다 반복되는 요구사항

9. Software Engineering Pattern

- Design Patterns
  - 설계문제 해결 노하우
    - 이 작업에는 어떤 디자인 패턴을 사용 할까?
    - 어떤 패턴이 다시 쓰일까?
    - 형식적으로는 사용할 수 있지만, 기능적, 구조적으로 다른 패턴은 무엇이 있을까?
  - 디자인 패턴의 구성요소
    - 이름, 의도, 비슷한 패턴, 문제 예시, 사용 예시, 패턴 사용을 위한 구조, 패턴을 진행하기 위한 클래스, 각 참석자들의 역할, 효과, 관련된 참조패턴

10. The Road Ahead and The Three Rs

- 앞으로 직원들은 감소하고, 국제적인 아웃 소싱, 소프트웨어 공학의 혁신이 있을 것이다.
- 미래에 펼쳐질 3가지 이슈를 알아보자.
  - Reuse
  - Reengineering
  - Retooling : Reuse와 Reengineering지원

11. Conclusion

- 어떤 산업에서 소프트웨어 공학을 이용하며 소프트웨어 개발 양식을 창조하는 활동 수준이 미래 소프트웨어 공학 분야의 중요한 이슈가 될 것이다.