SOFTWARE MAINTENANCE: A TUTORIAL

BY KEITH H. BENNETT

200711431

박성훈

(2008. 10. 6)

SOFTWARE ENGINEERING FIELD

- Software Engineering 중요 문제점
  - Software의 규모와 복잡성

- Software Maintenance의 목적
  - 제 시간의 완성
  - 요구조건의 충족
  - critical systems에 안전한 적용

SOFTWARE MAINTENANCE

- 정의 :

  Software System 또는 정당한 결점이 발생한 구성에 대한 수정, 퍼포먼스 또는 다른 능력의 향상, 환경 변화에 적응하는 과정
- 조사에 의하면 life cycle의 40 ~ 90%의 비용이 유지보수비용으로 사용
- 빠르게 관리를 하지 않을 시 "Applications backlog" 가 일어남,
  안전하고 싸게 비즈니스를 하기 위해 마케팅에 필요함

HISTORY OF MAINTENANCE

- 초창기 10년
  - 새로운 프로그램 사용
- 1960 – 1970년대
  - Software Maintenance 의 중요성을 인식하고 시작함
- 1980년대
  - 오래된 Architecture들이 완전히 새로운 디자인을 요구 받으며 어려워짐
- 1990년대
  - Software와 개발의 증진과 발전적인 변화

TYPES OF SOFTWARE MAINTENANCE

1. *Perfective maintenance(완벽한 유지보수)*
   - 변화는 사용자 요청의 결과로서 필요하다.
2. *Adaptive maintenance(채택적인 유지보수)*
   - 변화는 OS, 하드웨어, DBMS 등의 변화의 결과로서 필요하다.
3. *Corrective maintenance(고장 수리)*
   - 소프트웨어에서 결함의 확인과 제거
4. *Preventative maintenance(예방 정비)*
   - 변화는 소프트웨어가 계속적으로 유지될 수 있도록 한다.

REQUIREMENTS OF MAINTENANCE

- 빠른 변화
- 효과적인 비용
- 최악의 변화에도 신뢰도는 하락하면 안됨
- 시스템의 지속성은 하락해서는 안됨
  - 그렇지 않을 시, 미래에 더욱 큰 비용을 초래함
  - "Laws of evolution" (by Lehman)에 의해 증명됨

LAW OF EVOLUTION - LEHMAN

- 1st law – 계속되는 변화
  - 실제로 사용되어지는 프로그램은 반드시 적어도 그 환경에서 사용 가능하도록 변화하거나 점진적으로 진보한다.
- 2nd law – 복잡성의 증가
  - 발전적으로 프로그램을 변화시키는 것은 구조가 점점 복잡하게 되어가는 것을 의미한다. 여분의 자원은 보존하는 것과 구조를 단순화 하는 것에 전념하여야 한다.

PROBLEMS OF SOFTWARE MAINTENANCE

- Domino or ripple effect
  - 소스 코드를 바꾸었을 때, 실질적으로 많은 것들이 바뀜 (문서, 디자인, test suites)
- 유지보수가 영향을 끼치는 3가지
  - Alignment with organizational objectives (조직적 목적을 가진 단체)
  - Process issues (처리 문제)
  - Technical issues (기술적 문제)

ALIGNMENT WITH ORGANIZATIONAL OBJECTIVES

- Maintenance는 조직을 위한 명확히 가늠할 수 없는 이득 없이 큰 자원을 소모하는 주요 활동으로 조직의 고위 관리자에 의해 전망 됨

PROCESS ISSUES

1. 변화를 위한 요구
2. 소프트웨어와 조직 이 둘에 관한 영향 분석, 가격 분석, 시스템 포괄에 관해 연합된 요건
3. 소프트웨어의 변화가 이루어졌을 때 Regression test를 통하여 에러 발생을 예방

TECHNICAL ISSUES

- 쉽게 이해할 수 있는 소프트웨어 설계
  - 유지관리 활동의 대부분의 시간을 이를 위해 보냄
- 비용이 효율적인 테스트 방법
  - 변경된 부분만을 테스트 하는 것이 효과적이나 현재는 불가능
  - regression test 포함

OTHER ISSUES

- 초기의 개발 단계에서 설계비용 절감은 나중에 유지보수 과정에서 더 많은 비용 유발
- 보다 쉽게 진화하는 소프트웨어를 개발하는 관리 과정이 어렵다

ORGANIZATIONAL ASPECTS

- Software Maintenance의 관리가 가장 큰 문제
- Maintenance는 비영리 발전기, 자원이 세어나가는 곳으로 여겨짐
- Maintenance는 적은 투자금과 가난한 환경을 경험
- Maintenance가 외부에서 조달되는 경향

SOFTWARE AS AN ASSET

- 1993년 Foster가 투자비용 모델을 제안했으며 일본에서 사용 되였음
- Maintenance가 높은 위치를 지키고 정당한 재무지원을 받을 수 있도록 확실히했다.
- 유지보수 매니저가 비용과 이익을 계산하기 위해 변화의 재무 암시를 평가하도록 허락
- COCOMO 기술, 응용 관리를 도울 AMES 프로젝트

PROCESS MODELS

- Process Management
  - 상품을 개발하기 위해, 또는 서비스를 실행하기 위해 이행되는 일의 "방향, 제어, 조정" 으로서 정의
- Software process model은 이미 입증 된 process들이 필요함
- 조직의 성숙도 평가
- 프로세스 개선을 위해 metric 제공

OVERVIEW OF IEEE STANDARD FOR SOFTWARE MAINTENANCE

- 7 개의 activity model
  - Problem Identification
  - Analysis
  - Design
  - Implementation
  - System Test
  - Acceptance Test
  - Delivery

OVERVIEW CONTINUED…

- 각각 7개의 activity model은 5개의 행동을 함
  - Input life cycle products
  - Output life cycle products
  - Activity definition
  - Control
  - Metrics

OVERVIEW CONTINUED 2..

- 분석은 가장 어렵지만, 가장 중요하다
  - 실행 가능성 분석
  - 변경의 요구조건, 요구 툴, 계획 테스트 방법을 결정
  - 모든 영향을 받은 성분 확인
  - 3번째에 대한 테스트 전략이 수평으로 되고 복귀 테스트 요건이 개발
    - Unit testing, integration testing, functional acceptance testing

MORE ON STANDARD

- 각 단계를 위한 품질관리가 단계적으로 움직이는 것을 확실하게 할 최소 프로세스를 제공
- 소프트웨어 개발의 고전적인 개념에 기초를 둠
  - 빠른 응용프로그램 개발과 최종사용자 컴퓨팅과 경영 간부 급 이슈를 숨기지 않음

TECHNICAL ASPECTS OF SOFTWARE MAINTENANCE

- 필요 한 기술은 작은 변화와 더불어 초기 발전과 비슷
- Impact 분석이 유지보수에 필요함
  - 사용자의 번역은 소프트웨어 용어로 문제의 생존 능력을 결정할 문제를 나타냄
  - 바뀌는 것의 주요성분을 확인
  - 변한 솔루션 검사
- Impact 분석의 결과에 기초를 둔 가장 좋은 솔루션을 결정

TECHNICAL PROBLEMS

- Ripple effect
  - 소프트웨어 성분의 변화는 다른 성분에 영향을 끼칠 경향을 가지고 있다.
- 정적, 동적 분석 사용
- Impact 분석으로 인하여 더 이상의 변화를 확인할 수 없을 때 까지 Impact를 주어 확인한다.

TRACEABILITY

- Impact 분석을 실행하는데 이용될 수 있는 semantic link를 제공
- 몇몇의 link는 결정하기 어려움
- 대부분의 Impact 분석이 code level 끝남
- 분석을 위해 ripple propagation graph를 이용하면서 문서화의 모델로 사용
  - 코드와 관계없이 비용의 분석, 평가를 허용
- 그 반대의 경우도 마찬가지로 수행할 수 있는 코드에 대한 설계명세의 변화가 진행하는 것을 허용

LEGACY SYSTEMS

- 비 정식적 정의
  - 심하게 부분만 수정된 오래된 시스템
  - 대게 큰 시스템
  - 옛 언어로 만들어진 시스템
  - 문서화가 없는 시스템
  - 많은 데이터들이 오늘날에도 이 시스템을 통하여 처리 되는 시스템
  - 오래되었지만 여전히 기능을 하는 시스템
  - 성장시키기가 어려운 시스템
  - 다른 시스템으로 대체하기엔 비싼 시스템

ANALYSIS OF LEGACY SYSTEMS

- 역 공학 접근
  - 가장 효과적인 접근
- 캡슐화 접근 (대중적)
  - 새로운 시스템이 점차 오래된 시스템의 기능을 인수함

REVERSE ENGINEERING

- 정의
  - 시스템들의 성분과 그들의 상호성을 확인하기 위해 주제 시스템을 분석하는 프로세스, 다른 형태의 더 높은 레벨의 추상적 시스템의 개념을 만드는 것
- 기능
  - 시스템을 새로이 만들지 않고 변경함
  - 프로그램 이해에 도움 제공
- 방식
  - 재 문서화
  - 디자인 복구

METHODS OF REVERSE ENGINEERING

- Slicing
  - 정적인 분석 기술
  - 지정된 변수에 영향을 미치는 소스 코드만을 조사
- Tools to attach notes the source code
  - 모든 시스템의 재 문서화를 회피
  - 안정된 소스 부분은 생각지 않음
  - 비용 절약

MORE ON LEGACY SYSTEMS

- 프로그램 이해
- 재설계 : legacy system의 활발한 변화
  - 상대적인 같은 레벨의 추상에서 다른 하나에 대한 어떤 표시로부터 변화
  - 시스템들의 외부 행동 보존
  - 증가 보수성
- Re-engineering
  - 시험과 그것을 다시 구성할 주제 시스템의 새로운 형태로 교체
  - 가장 급진적이고 가장 비쌈

CRITERIA FOR REVERSE ENGINEERING

- *List of criteria*
  - Management creteria
    - Enforcing product and process standards (such as the IE EE draft standard introduced above)
    - Permit better maintenance management
    - Legal contesting of reverse engineering legislation
    - Better audit trails

CRITERIA FOR REVERSE ENGINEERING

- List of criteria
  - Quality criteria
    - Simplification of complex software
    - Facilitating detection of errors
    - Removing side effects
    - Improve code quality
    - Undertaking major design repair correction
    - Production of up-to-date documentation
    - Preparing full test suites
    - Improving performance
    - Aligning with practices elsewhere in the company
    - Financial auditing
    - Facilitate quality audits(e.g., ISO9000)

CRITERIA FOR REVERSE ENGINEERING

- *List of criteria*
  - Technical creteria
    - To allow major changes to be made
    - To discover the underlying business model
    - To discover the design and requirements specification
    - To port the system
    - To establish a reuse library
    - To introduce technical innovation such as fault toleranc e, graphic interfaces, and so on
    - To record many different types of high-level representat ions
    - To update tool support
    - Disaster recovery

CRITERIA FOR REVERSE ENGINEERING

- 분석은 legacy system에서 억제된 비즈니스 규칙부터 시작해야 함
  - legacy system은 오랜 경험을 나타냄

REVERSE ENGINEERING TECHNIQUES

- 제어 흐름과 데이터 흐름의 사용은 그래프로 나타낸다
  - 제어 흐름의 재설계
  - 유지보수를 지지하는 데에 상업 도구를 이용
- 프로그램 계획이나 진부한 접근
  - 프로그램의 다수는 일반적인 디자인 아이디어를 이용
  - 라이브러리의 소스코드를 사용

THE END..