요구공학

개요 및 자료수집

- SW 개발의 목적
  - 고품질의 SW를
  - 정해진 개발 기간 안에
  - 주어진 예산 범위 안에서 개발하는 것

개요 및 자료수집

- requirement
  - 사전적 의미 : 이용자가 어떤 문제를 풀거나 목표를 달성하기 위해 필요한 조건이나 능력
  - SW개발적 의미 : 사용자와 개발자 간의 합의한 개발 범위에서 시스템이 제공해야 하는 기능

- 요구 분석 명세서
  - 개발 초기에 사용자의 요구 사항을 추출하여 정리한 문서
  - What에만 초점 how는 고려치 않는다.(how는 설계단계에서 해야한다.)
- 요구 분석 과정
  - 사용자 요구 파악 -> SW 목표 수립 -> 모델링 -> 요구 분석 명세서
  - SW 요구사항 정의를 위해 사용자의 요구 사항을 조사하고 확인하는 과정
  - 현 상태 파악 -> 사용자 요구 파악 -> 사용자 요구 결정
- 요구 분석 관련자의 역할과 의미

개요 및 자료수집

- 발주사
  - 만들어 달라고 하는 회사
- 경영자
  - 만들어 달라고 하는 회사의 최고 책임자, 사장
- 발주담당자
  - 발주사의 담당자(외주를 위해 모든 절차를 준비하는 담당자)
- 사용자
  - 개발된 시스템을 업무에 실제 사용하는 사람
- 수주자
  - 발주를 받아가는 회사(만들어 주는 개발회사)
- 분석가
  - 발주자의 요구 사항 파악 및 추출 정리하는 담당(수주자의 사람)
- 설계자
  - 요구 분석 명세서를 바탕으로 아키텍쳐, 모듈, 디비 유아이 설계 등을 담당
- 개발자
  - 프로그래머, 분석가, 설계자 등을 말하며 좁은 의미로는 프로그래머

요구 분석이 어려운 이유

- 사용자가 문제가 발생하면 그 분야의 전문가를 찾아간다.
- 하지만 소프트웨어에서는 전문가와 대화하기가 어렵다.

- 1.1) 문제 영역에 대한 이해력 부족 (분석가)
  - 분석가 : IT 관련 분야 전공자(대부분)
  - 개발 (문제)영역의 전문성 부족
  - 사용자 요구 파악의 어려움
  - 잘못된 분석
  - -> 문제 영역 프로젝트관련 유 경험 분석가 투입
- 1.2) 의사소통 문제 발생
  - 사용자
  - sample이 없어 요구 사항 설명이 어렵고, 설명 방법도 잘 모른다.(그래서 프로토타입 사용하긴 함)
  - 사용자 의사 전달 능력이 요구 사항 전달 내용에 영향을 미친다.
  - 일관성이 없거나 불안전한 분석명세서가 될 가능성이 높다.
- 1.3) 계속 변하는 요구 사항
  - 단순한 초기 요구 사항
  - 개발될 SW에 대해 사용자의 이해의 폭이 넓어짐
  - 지식이 늘어나 새로운 요구 사항 발생 및 증가
  - 관련 지식의 증가로 요구 사항 변경 발생
  - -> 변경 요구에 대한 대처
변경 사항에 대한 날짜 별, 기간 별 기록
변경이 미치는 영향에 대해 사전 분석
  - -> 대처를 못하면 요구사항간의 충동, 요구사항 결여, 불일치 발생한다.
- 1.4) 애매모호한 요구 사항
  - 해석을 달리할 수 있는 애매한 표현의 요구
  - 사용자의 일관성 없는 요구
  - 부서간의 상충된 요구
  - 경영진과 실무자간의 상반된 요구(경영자 : 적은 돈으로 효율성 중심, 실무자 : 돈이 좀 들어도 편한것)
  - -> 원치 않는 결과 발생(분석가가 해결해야함), 사용자와 개발자의 마찰(분석명세서에 대한 각각의 다른 해석 때문)

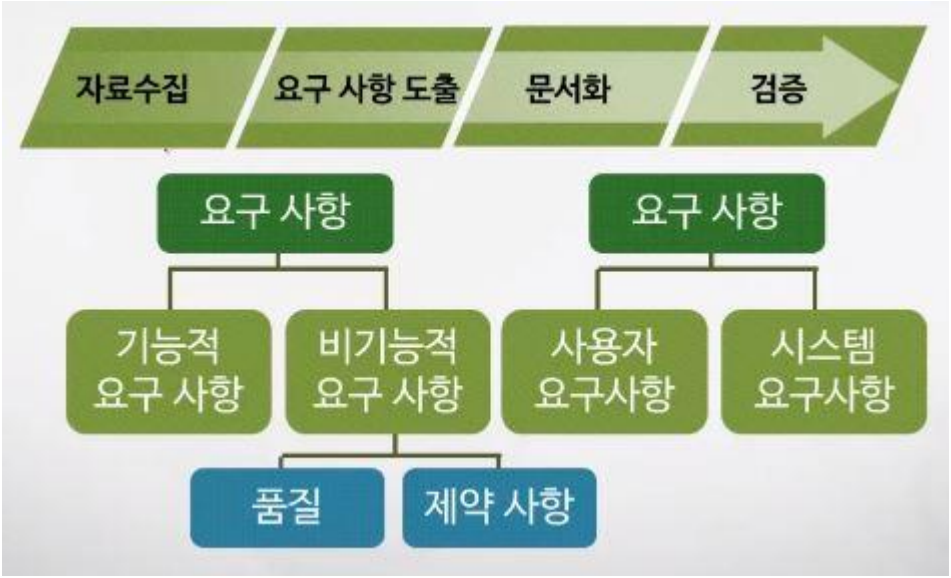
분석가에게 필요한 능력

- 2.1) 의사 소통과 협상 능력
  - 상반된 의견 -> 조율 능력
  - 무리한 요구 -> 우선 순위 결정 및 실현, 가능한 요구 판별 능력력
  - 추상적 요구 -> 논리적 분할을 통한 해결 방안 제시
- 2.2) 개발 업무 영역에 대한 지식
  - 풍부
    투자 대비 효과에 관심
    일을 주도적으로 수행
  - 부족
    고객의 요구에 대한 정확한 이해의 어려움
    문제 발생 시 올바른 해해결책 제시 불가능
- 2.3) 개발 관련 기술에 대한 지식
  - 최신 기술에 대한 지식 필요
  - 사용자 요구에 대한 구현 가능성 판단
  - 논리적으로 정리할 수 있는 문서화 능력
  - 전체 기능의 세분화 능력과 통합된 전체 그림 제시 능력
- 2.4) 상반된 요구에 대한 중재 능력
  - 경영자 : 투자대비 효과에 관심
  - 고객(발주자) : 개발비 적고, 개발 기간 엄수 가능 업체에 관심
  - 사용자 : 업무의 효율적 처리와 편리성 등에 관심
    -> 경영자/고객/사용자의 상반된 요구에 대해 중재 및 조율 능력이 필요

요구 사항 수집 방법

- 3.1) 기존 자료 수집 및 분석
  - 업무 담당자 -> 문서 수집(업무 메뉴어르 업무 흐름도 등) -> 전반적 업무 파악 및 이해
  - 현행 시스템 분석 -> 입력화면, 결과 화면에 대한 출력물 조사
  - -> 현행 요구 사항 파악
- 3.2) 인터뷰(interview)
  - 권장 순서 : 수집된 자료 정리 및 분석 후 인터뷰
  - 분석가 : 인터뷰 중 요구 삭제/변경/추가사항 검토
    -> 수집 요구 사항 확인, 유사 프로젝트 경험 전달 -> 새 기능 권고
  - 분석가 -> 경영자
  - 투자대비 효과 분석 결과 설명(그래프 등 수치화)
  - 경영자의 관심과 전폭적 지원이 프로젝트 성공에 다가간다.
- 3.3) 설문조사
  - 문서 -> 업무파악
  - 인터뷰 -> 추가적인 요구사항 추출
  - 설문 조사 -> 다시 한번 요구 사항 확인 및 추가

요구 분석 절차



- 1.1) 자료 수집
  - 현행 시스템 파악 -> 문제점 도출
  - 인터뷰 서류 검토
- 1.2) 요구 사항 분류 및 도출
  - 수집 자료 정리 및 분류 -> 개발에 반영할 요구사항 도출
- 1.3) 문서화
  - 도출한 요구사항 -> 요구분석명세서 작성
- 1.4) 검증
  - 요구분석명세서 검토 -> 모순사항, 빠뜨린 사항 찾기

요구 사항 분류 및 도출

- 기능적 요구사항
- 비기능 요구사항
- 사용자 요구사항
- 시스템 요구사항

기능적 요구사항

- 사용자가 원하는 기능
  - 사용자 : 그 기능을 시스템을 통해 제공받기를 원한다.
  - 시스템 : 사용자에게 필요한 기능을 제공해주어야 한다

비기능적 요구사항

- 수행 가능한 환경, 품질, 제약사항
- 제약 사항 :
  개발된 SW가 수행되는 환경과 같은 조건
  ex) Java 언어를 사용해 개발하고, CBD개발 방법론 적용
  WebLogic Server를 middleware로 사용
  Window, Linux 모두 실행 가능하게

비기능적 요구사항(품질)

- 2.2.1) 품질
  - 사용자가 개발될 SW에 요구하는 품질 조건
    ex) 24시간 중단되지 않아야 한다.
    보안 등급을 정해 허가 받은 사람만 접근하게 한다.
    초보자, 숙련자 모두 편리하게 사용할 수 있어야 한다.(단축키 등등)
    도서검색시 50만권당 3초안에 걸려야 한다

비기능적 요구사항(품질)

- 2.2.1.1) 신뢰성(reliability)
  - 높은 신뢰를 가진 SW
    주어진 시간과 환경에서 고장 없이 사용할 수 있어야 한다.
    요구한 기능을 정확하고, 일관되게 원하는 정밀도로 수행해야 한다.
  - 신뢰도
    고장없이 동작하는 시간의 비율
    ex) 신뢰도 99% : 100번 수행 시, 오류 없이 99번 동작
    신뢰도 측정 방법 고장 간 평균 시간(MTBF)와 이용 가능성(가용성)을 척도로 사용한다.
  - MTBF 고장 간 평균 시간(Mean Time Between Failure), 고장에서 다음 고장까지 평균 시간
    MTBF = MTTF + MTTR
  - MTTF 평균 실패 시간(Mean Time To Failure), 수리한 후 다음 고장까지의 평균 시간
  - MTTR 평균 수리 시간(Mean Time To Repair), 고장 발생 시점에서 수리시 까지 평균 시간
  - 가용성(availability) = MTTF/(MTTF+MTTR)*100%
    이용 가능성 : 주어진 시점에서 프로그램이 요구에 따라 작동되고 있을 가능성
  - 신뢰도가 높은 SW가 되려면
    고장을 일으키지 않고 회피할 수 있는 능력이 높아야 한다.
    고장이 발생해도 이전 수준으로 성능이 회복되어야 한다.
    고장으로 인해 영향을 받은 데이터들이 정상적으로 잘 복구가 되어야 한다.

비기능적 요구사항(품질)

- 2.2.1.2) 성능(performance)
  - 해당 기능을 정상적으로 수행해야 한다.
  - 사용자가 원하는 조건(응답 시간, 데이터의 처리량 등)을 만족시켜야 한다.
    ex) 동시 접속자 수 10,000명은 가능해야 한다.
    도서 관리 시스템에서 책을 검색한 결과를 2초 이내로 보여줘야 한다.
- 2.2.1.3) 보안성(security)
  - 허가 받은 담당자만 해당 시스템에 접근할 수 있도록 한다.
  - 인증을 받지 않은 사람의 접근을 처음부터 막아 시스템과 데이터를 보호해야 한다.
- 2.2.1.4) 안전성(safety)
  - 작동하는 모든 시스템이 SW 오류로 인해 인명 피해가 발생하지 않아야 한다.
    ex) 임베디드 SW에서 문에 무언가 끼면 문이 열려야 한다

비기능적 요구사항(품질)

- 2.2.1.5) 사용성(usability)
  - SW를 사용시 사용자가 혼란스러워하거나 사용하는 순간에 고민하지 않게 하는 소프트웨어
    ex) 사용성이 떨어지는 예로 핸드폰 번호 작성이 - 을 넣어야할지 말아야할지 명확하게 제시 못하는 경우
    의미가 같은 단어를 혼용하여 사용자를 혼란스럽게 하는 경우
    카드 번호 쓸때 빈칸이 차면 자동으로 다음칸으로 넘어가는 센스
  - 사용성을 높이려면?
    초보자 : 메뉴 사용
    숙련자 : 단축키 사용
    도움말 사용
    일관성 있는 인터페이스

사용자 요구사항

- 사용자 요구 분석 명세서(사용자를 위한)
- 요구 사항 정의서
  사용자의 요구사항을 정리하여 작성한 문서
  사용자와 대화 하기 위해서 거부감을 줄이고 충분히 이해할 수 있도록 쉽게 작성
  표준 양식, 다이어그램 사용
  사용자와 분석가가 서로 충분한 대화를 나누며 함께 작성한다

시스템 요구사항

- 시스템 요구 분석 명세서(개발자를 위한)
  사용자 요구 사항을 정리하여 작성한 문서
  (개발자에게 보여주기 위한 문서)
  기술적 용어나 전문적 표현 사용
  완전하고 일관성 있게 작성
  구조적 발법론의 구조적 언어
  usecase diagram
  검증에 강한 Z명세와 같은 정형화된 수학적 명세 언어

요구 명세 기법

- 4.1) 비정형 명세 기법
  - 자연어, diagram 사용
    - 장점
    작성하기 쉽다.
    쉬운 이해 -> 용이한 전달 -> 사용자의 적극적 참여 유도 가능
    - 단점
    애매모호한 표현 -> 다른 해석 가능 -> 일관성이 떨어짐
- 4.2) 정형 명세 기법
  - Z 정형 명세 언어(수학적 원리와 기법 사용)
    - 장점
    정확하고 간결한 표현 -> 증명 기술을 이용한 일관성/완전성 검증
    정형화된 형태의 명세 -> test case 생성 용이
    - 단점
    수학적 표기법 공부 -> 표기법을 이용한 정확한 표현
- 5. 요구 사항 검증
  - 요구 분석 명세서가 정확하고 완전하게 서술 되었는지 검토하는 활동
  - 사용자의 요구 사항이 완전하게 서술되었는지
  - 작성 시 문서 표준을 따랐는지
  - 다음 단계인 설계에서 사용하기에 적합한지
  - -> 내부적 일치성과 완전성을 검증 한다

요구 명세 기법

속성	설명
완전성(completeness)	모든 요구 사항이 누락되지 않고 완전하게 반영되고 있는가?
일관성(consistency)	요구 사항 간에 모순되거나 충돌되지 않고 일관성을 유지하는가?
명확성(unambiguity)	표현이 애매모호하지 않고 참여자가 명확히 이해할 수 있는가?
기능성(functionality)	'어떻게'보다 '무엇을'에 관점을 두고 서술되었는가?
검증 가능성(verifiability)	사용자가 요구하는 내용과 일치하는지를 검증할 수 있는가?
추적 가능성(traceability)	사용자 요구 분석 명세서와 설계 사양서를 추적할 수 있는가?
변경 용이성 (easily changeable)	변경 시 쉽게 찾아 변경할 수 있도록 작성되었는가?

- 완전성(completeness)
  - 빠진 부분 없이 모두 있음
  - 기능적 요구사항뿐만 아니라 성능, 제약사항 등 누락되지 않고 모두 서술되어야 한다.
    ex) ATM에서 출금 버튼을 누른 후 나가버렸는데도, 계속 출금액 입력을 기다리고 있음
    -> 10초가 지나면 음성 메시지를 통해 알려줌, -> 또 10초가 지나면 초기 상태로 되돌아감.
- 명확성(unambiguity)
  - 계약서와 같은 효력을 발생 -> 문제 발생 시 근거 자료로 활용
    -> 애매모호 하지 않은 명확한 표현으로 작성 -> 관점에 따라 다른 해석이 불가능하게 작성
- 일관성(consistency)
  - 서로 상반된 요구, 불일치한 요구, 중복된 요구가 존재가 없다.
- 변경 용이성(modifiability)
  - 변경하기 쉽게 요구 분석 명세서를 작성하는 것
  - -> 요구 사항이 서로 의존적이지 않고 독립적으로 서술되어야 한다.
- 검증 가능성(verifiability)
  - 검증하기 쉽게 요구분석명세서가 만들어져야 한다.
  - 시스템이 요구 사항을 만족하는지 대해 체계적으로 검사할 수 있게 작성
    ex) 동접자 10만명 해달라.
    잘못된 예) 많은 사람이 동접 가능하게 해달라.
- 추적 가능성(traceability)
  - 추적이 가능하도록 작성
    프로그램에서 오류가 발생 했을 시 이 오류로 인해 영향받는 곳이 어딘지 추적할 수 있어야 한다.
    그것을 요구 분석 명세서에서 확인 할 수 있게 한다

IEEE Std. 830-1998에서 권고하는 요구 분석 명세서의 항목

1. 소개
   1.1 목적
   1.2 범위
   1.3 정의, 약어
   1.4 참조
   1.5 개요

2. 전반적 서술
   2.1 제품 관점
      · 시스템 인터페이스 · 사용자 인터페이스 · 하드웨어 인터페이스
      · 소프트웨어 인터페이스 · 통신 인터페이스 · 메모리 · 운영
   2.2 제품 기능
   2.3 사용자 특성
   2.4 제약 사항
      · 규제 정책, 하드웨어 제약 사항, 다른 응용 프로그램과의 인터페이스, 병렬 수행, 감사기능, 제어 기능, 신뢰성 요구 사항, 안전 및 보안 요구 사항
   2.5 가정 및 의존성
   2.6 요구 사항 할당

3. 구체적 요구 사항
   3.1 외부 인터페이스
      · 사용자 인터페이스, 하드웨어 인터페이스, 소프트웨어 인터페이스, 통신 인터페이스 가능
   3.2 성능 요구 사항
   3.3 로컬 DB 요구 사항
   3.4 설계 제약 사항
   3.5 소프트웨어 시스템 속성
      · 신뢰성, 가용성, 보안성, 유지보수 용이성, 이직성