**요구사항 분석 실무**

**1교시. 요구사항정의 개요**

**2교시. 현행 시스템 분석**

**3교시. 요구사항 확인**

**4교시. 기능 요구사항 작성 실습**

**5교시. 테스트케이스 설계**

**6교시. 테스트케이스 작성 실습**

-------------------------------------------------------------

전통적인 프로그램 개발 공정

분석->설계->개발->테스트

(ex 폭포수 방법론, 애자일 방법론…)

요구사항 분석은 분석과 설계 사이

Software Requirements 요구 공학

객체지향 개발 방법론, 도메인 주도 개발 방법론…등

CBP는 CBD/DDD 베이스

-------------------------------------------------------------

**요구사항정의 개요**

**요구사항이란?**

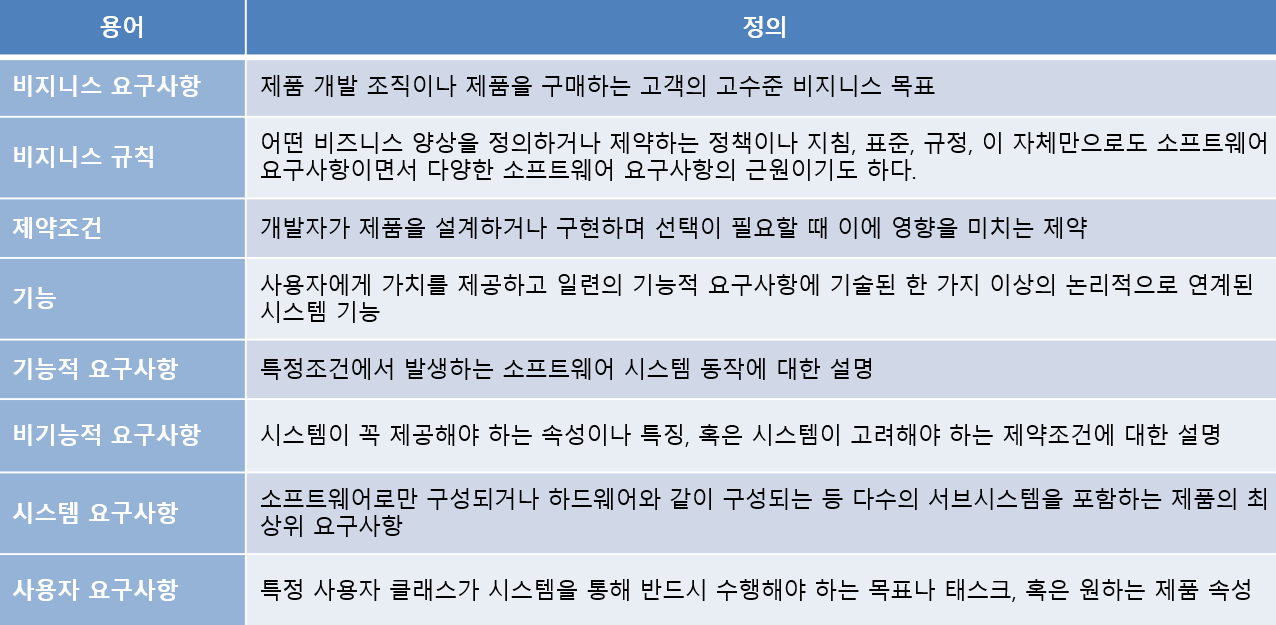
**‘요구사항’의 여러가지 해석**

컴퓨터 프로그래밍이 탄생한 지 수십 년이 지난 지금도 소프트웨어 전문가들은 "요구사항"이 무엇인지에 대해 여전히 격렬한 논쟁을 벌이고 있다…. 고 한다.

요구사항이란 "설계안을 도출하기 위한 모든 것" - Lawrence 1997

요구사항이란 무엇을 구현해야 하는가에 대한 명세다. 요구사항은 시스템이 동작하는 방법이나 시스템 속성 혹은 특성을 설명한 것이다. 이는 시스템 개발 프로세스의 제약조건이라고 볼 수 있다.

- Ian Sommerville and Pete Sawyer 1997



기능 : 액션, 이벤트

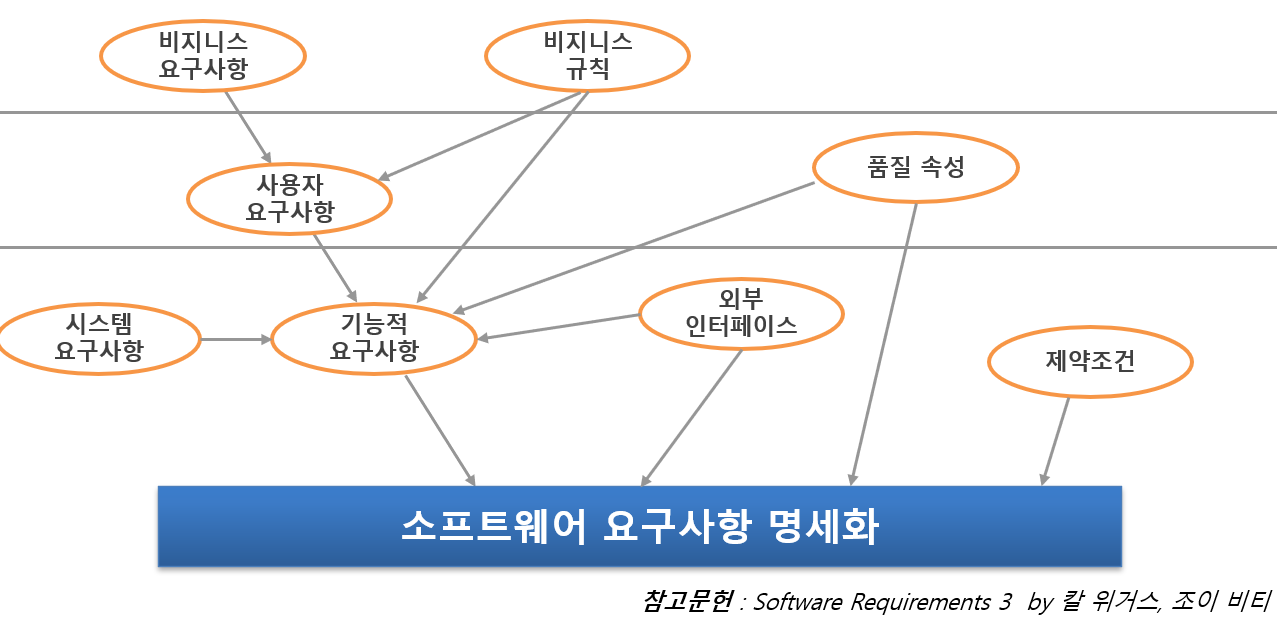
기능적 요구사항 : 눈에 보이는 또는 보이지 않는

시스템 -> 서버, 네트워크

정의 Definition or 개발 Development

vs 명세 Specification

Business Analyst/Consultant(중계) vs 개발팀 설계자



인터페이스 : 시스템 사이의 통신

-------------------------------------------------------------

**요구사항의 중요성**

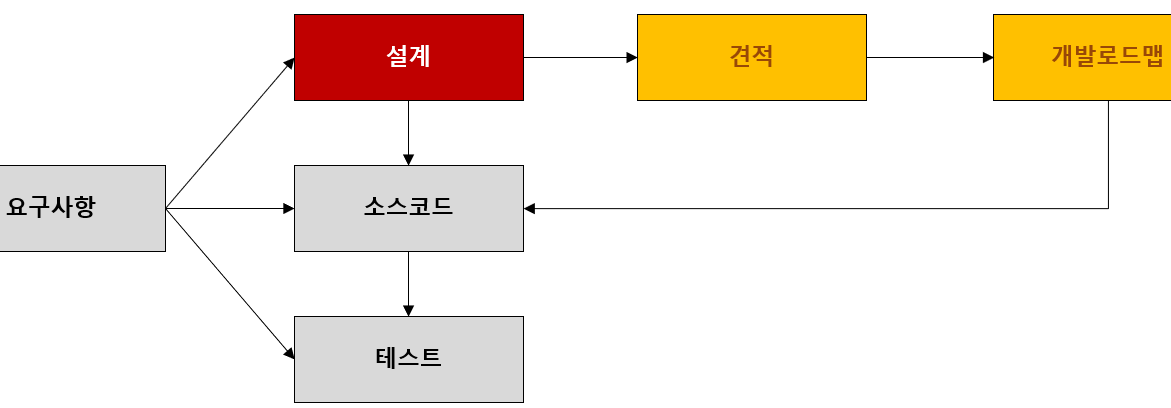
무엇을 개발해야 하는가? 를 결정

소프트웨어 개발에 있어서, 출발점이자, 달성해야 하는 최종 목표

소프트웨어/시스템의 범위 결정

요구사항 중 기능목록은 개발리소스 목록의 원천이 됨

견적과 개발로드맵



Cf. 뱅크웨어는 B2B

식별(뭐가 있는지 없는지 찾는 것)->분석->선정->제안(제일 어려운 작업)->수주->계약->수행(개발)->사후

정보요청서(RFI: Request for Information)

* 제안요청서(RFP: Request for Proposal)
* 제안서

-------------------------------------------------------------

**요구사항 정의가 어려운 이유**

사용자 참여 부족

불충분한 사용자와의 대화

빈약하고 불충분한 요구사항

점점 늘어나는 사용자 요구사항

모호한 요구사항

간과된 이해관계자

조율되거나 검토되지 않은 요구사항  
(a.k.a 그냥 좋을 것 같아서 )

**커뮤니케이션의 중요성**

말했다 vs. 안 했다

하기로 했다 vs. 하기로 한 적 없다.

너무 늦었다 vs. 꼭 해야 한다.

지금 결정해야 한다 vs. 그럼, 일단 이렇게 하자.

지금 결정해야 한다 vs. 둘 다 되게 해주세요.

들은 적 없다 vs. 그런 것 까지 말해줘야 하나?

**프로젝트 현장의 현실**

프로젝트 구성원이 최종 시스템 이미지에 대한 상호 공감이 없는 상태에서 시작

IT는 업무요건을 잘 모르고, 현업은 IT를 잘 모르고 시작함.

요구사항 정의는 미래의 시스템을 상상하며 합의하는 과정

이해당사자(현업)간의 이해관계 상충

멀티벤더 간의 이해관계 상충

프로젝트 초반에는 R&R이 불분명한 상태에서 프로젝트가 시작됨.

특히 멀티벤더인 경우에는 이 문제가 더욱 부각된다.

Regulation & Compliance 규정

Legacy System (기존 시스템)

Legacy를 잘 아는 사람이 있는가?

Legacy를 잘 아는 사람이 프로젝트에 투입되는가?

대외기관 I/F (인터페이스)

COTS(Commercial, off-the-shelf) (완제품)

요구사항정의는 기본적으로 문장.   
따라서, 요구사항의 정의 참여자는 명확하게 문서화하는 기술을 갖추어야 함.

모든 요구사항의 명세는 동일 수준일까?

-------------------------------------------------------------

**요구사항의 검증**



\*요구사항 정의

외부설계 : 데이터모델링, 서버, 네트워크 설계

내부설계 : 화면 설계, 기능 설계

거래테스트(간단)

결합테스트(복잡) : 앞사람이 만든 프로그램이 뒤에서 잘 이어지는지

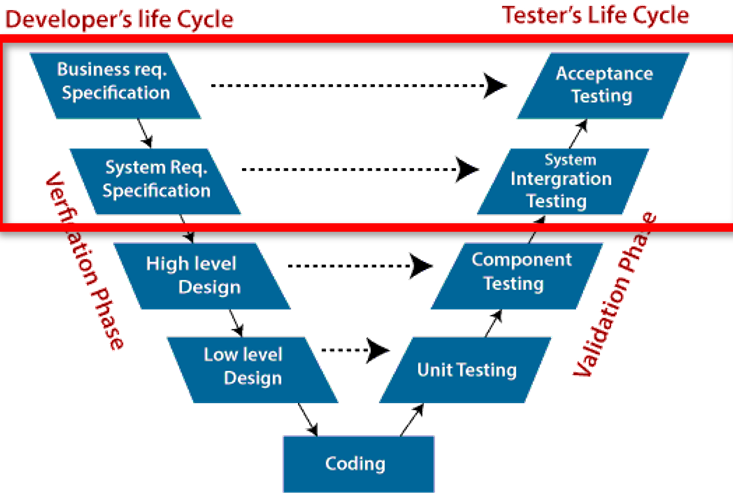
통합테스트 : 전체적으로 잘 돌아가는지, 요구사항 정의가 잘 됐는지, 잘 만들어졌는지 테스트(개발팀 야근 多)

인수테스트/교육 : 프로젝트 철수하며 인수인계

컷오버 : 이행, open, go live

**V모델**

개발 프로세스에 따라, 각 개발단계에 대응하는 테스트단계를 도식화한 것



요구사항 정의할 때, 통합테스트에서 어떤 것을 테스트할 지 테스트 정의를 같이 설계해야 한다.

개발할 때 그 테스트 케이스를 통과할 수 있을지를 염두에 두고 개발해야 누락을 줄일 수 있다.

**개발프로젝트의 해결 방식**

문서화 및 합의

프로젝트 단계별 구체화

(마일스톤 : 표지석, 이정표)

프로토타이핑(실험모델, 임시)

ex. jsp화면을 파워포인트로 먼저 구상

요구사항 추적관리

(요구사항은…변할 수 밖에 없다… 유연한 설계 필요)

솔루션 찾기(오픈소스 등 대안 찾기)

패턴의 적용(과거에 어떻게 했었는지)

그리고… 테스팅(센스와 내공 필요)

**현행 시스템 분석**

**현행 시스템 파악**

현행 시스템이 어떤 하위 시스템으로 구성되어 있는지

제공하는 기능이 무엇인지

다른 시스템들과 어떤 정보를 주고받는지

어떤 기술요소를 사용하고 있는지

사용하고 있는 소프트웨어 및 하드웨어는 무엇인지

네트워크는 어떻게 구성되어 있는지

-------------------------------------------------------------

**분석 절차**

**1단계. 구성/기능/인터페이스 파악**

현행 시스템의 구성, 기능, 인터페이스 현황을 파악하는 단계

각 업무에 속하는 단위 업무 정보시스템들의 명칭, 주요 기능들을 파악

단위 업무 시스템에서 제공하는 기능들을 주요 기능과 하부 기능으로 구분하여 계층형으로 파악

단위 업무 시스템이 다른 단위 업무 시스템과 주고받는 데이터의 종류, 데이터 형식, 프로토콜, 연계유형, 주기 등을 파악

**2단계. 아키텍처 및 S/W구성 파악**

현행 시스템의 아키텍처 및 소프트웨어 구성 현황을 파악하는 단계

기간 업무를 수행하기 위하여 계층별로 어떠한 기술 요소들을 사용하고 있는지를 구성도로 작성

단위 업무 시스템의 업무 처리를 위해 설치되어 있는 소프트웨어들의 제품명, 용 도, 라이선스 적용 방식, 라이선스 수를 파악

**3단계. H/W 및 N/W 파악**

현행 시스템의 하드웨어 및 네트워크 구성 현황을 파악하는 단계

단위 업무 시스템들이 어디에 위치하고 있는 서버에서 운용되고 있는지, 서버의 주요 사양(CPU 처리 속도, 메모리 크기, 하드디스크의 용량 등)과 수량, 이중화 가 적용되어 있는지, 어떠한 네트워크 구성을 가지고 있는지를 파악

**4단계. 개발 기술 환경 정의**

현행 시스템 파악을 토대로, 개발 기술 환경을 정의

운영체제, DBMS, 미들웨어, 오픈 소스 순

신뢰도, 가용성, 성능, 호환성, 구축비용, 기술지원 등을 고려하여 선정

**요구사항 확인**

**요구공학**

요구사항 개발 프로세스: 도출 ->분석 ->명세 ->확인



**요구사항 도출**

요구사항이 어디에 있고, 어떻게 수집할 것인가?

이해관계자(Stakeholder) 식별

개발 팀과 고객 사이의 관계 형성

**요구사항 분석**

요구사항들 간 상충되는 것을 해결

소프트웨어의 범위 파악

소프트웨어가 환경과 어떻게 상호 작용하는지 이해

시스템 요구사항을 정제하여 소프트웨어 요구사항을 도출

개념 모델링(Conceptual Modeling)

: 문제 도메인의 엔티티(entity)들과 그들의 관계 및 종속성 식별

데이터 모델(Data Model), 데이터 흐름 모델(Data Flow Model), 유스케이스 다이어그램(Use Case Diagram), 사용자 인터랙션(User Interactions) 등을 작성

UML(Unified Modeling Language)의 활용

: 유스케이스 다이어그램, 구조 다이어그램(Structure Diagram), 행위 다이어그램(Behavior Diagram)

**요구사항 명세**

체계적으로 검토, 평가, 승인될 수 있는 문서를 작성

시스템 정의, 시스템 요구사항, 소프트웨어 요구사항 등을 작성

**요구사항 확인**

분석가가 요구사항을 이해했는지 확인(Validation)이 필요

요구사항 문서가 회사의 표준에 적합하고 이해 가능하며, 일관성이 있고, 완전한지 검증(Verification) 하는 것이 중요

이해관계자들이 문서를 검토해야 하며, 이 요구사항 정의 문서들에 대해 형상관리

리소스가 요구사항에 할당되기 전에 문제를 파악하기 위하여 검증을 수행

-------------------------------------------------------------

**요구사항의 분류**

기능적 요구사항/비기능적 요구사항

사용자 요구사항/시스템 요구사항

**기능적 요구사항**

사용자가 원하는 기능

사용자의 요구를 빠진 것 없이 정확하게 도출하고, 도출된 기능을 요구 분석 명세서에 완전하고 일관성 있게 표현해야 하며, 시스템에도 전부 반영하여 사용자에게 100% 제공해야 한다.

완전성(completeness)

사용자가 원하는 모든 기능이 포함되어 있어야 한다

일관성(consistency)

요구사항들이 서로 간에 모순이 있어서는 안 된다

**기능 요구사항 정의서**

기능 요구사항 정의서는 기능의 패턴으로 구분하여 작성

채널서비스(모바일 화면)

화면

시스템 인터페이스

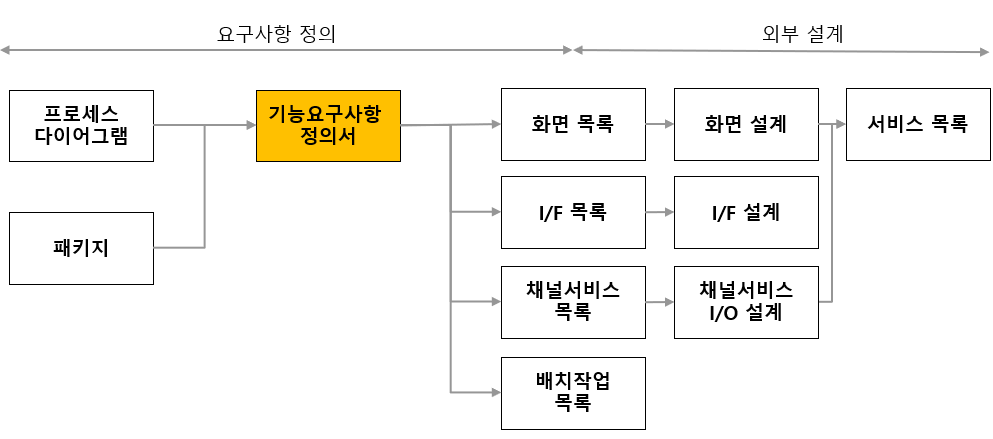
배치작업

(Batch 일괄처리작업 <-> 실시간처리작업 Online)

기타 중요 업무규칙

각 기능 요구사항은 개발방법론 상의 후 공정인 외부설계의 단위와 연결된다.

프로젝트 종료 시점까지 요구사항 추적관리의 형태로 추적되고 검증된다.



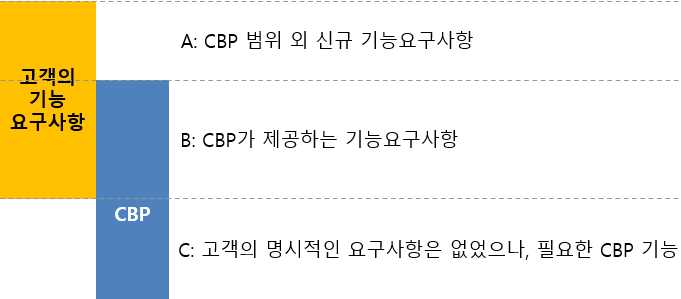
**Fit/Gap**

차이점을 분석하는 행위

고객의 요구사항은 관심 분야에만 집중된다.

고객의 요구사항만으로 기능 요구사항을 정의하면, 계단이 없는 2층 건물을 짓는 실수를 범하게 된다.

이것은 명백히 건축설계자의 오류이다.



C영역이라고 해서, 현재의 CBP기능을 그대로 사용한다는 것은 아닐 수 있으므로, 데모를 통해 기능을 이해시키고, 변경 요구사항을 받아야 한다.

**기능 요구사항 기술(Specification)의 기술(Technique)**

**전제사항 작성하기**

기능 요구사항은 시스템의 제공하는 서비스를 기술하는 것이다.

해당 서비스를 제공하기 위한 타 솔루션의 의존관계 또는 제약사항을 기술한다.

전제사항은 견적에 중대한 영향을 미칠 수 있으므로, 작성하는 개발팀이나 리뷰하는 현업 모두, 주의깊게 검토해야 한다.

**설계와 요구사항을 구분하기**

요구사항은 고객이 원하는 것을 기술한 것이고, 설계는 IT리소스의 책임할당에 대한 것이다.

요구사항은 What, 설계는 How.

그런데, What과 How는 문맥에 따라 달라지는 면이 있기 때문에 구분하기 어려운 면이 있다.

**개발팀을 짜증나게 하는 고객**

설계에 간섭하거나 의사결정을 내리려고 할 때(잘못된 요구사항)

**고객을 짜증나게 하는 개발팀**

설계 결정을 고객에게 미룰 때

**필요 이상으로 세부적으로 작성하지 않는다.**

요구사항은 외부설계와 상세설계를 진행하면서 더욱 세부적인 의사결정을 하게 된다.(적절한 Depth)

이후 단계에서 상세화할 것을 미리 적는 경우, 요구사항을 현행화하는데, 너무 많은 공수가 들게 된다.

기능요구사항의 업무 분류 : 대 – 중 – 소

**냄새나는 요구사항 판별하기**

요구사항 = 테스트케이스

테스트케이스를 쓸 수 없다면, 요구사항이 너무 추상적이거나 모호하다는 뜻이다.

유비쿼터스 랭귀지(UML 등)를 준수하지 않았다.

요구사항이 설계에 지나치게 간섭하거나 제약하고 있다.

상호 모순된 요구사항

비효율적인 방식으로 기술되고 있다.

동일한 요구사항이 지나치게 중복되어 기술되어 있다.

문장보다는 테이블(표)이 훨씬 명확하다.

-------------------------------------------------------------

**업무규칙**

**업무규칙 정의하기**

다양한 방법으로 정의할 수 있지만, 가장 적절한 형태로 효율적으로 작성해야 한다.

-------------------------------------------------------------

**비기능적 요구사항**

예쁜거… 잘 되게 하는거…

기능성이 아닌 범주의 요구사항을 의미

수행 가능한 환경, 품질, 제약 사항 등

품질(얼마나 우수하냐)

신뢰성(reliability) : 장애 없이 동작하는 시간의 비율

성능(performance) : 원하는 조건(응답 시간, 데이터의 처리량 등)을 만족시키는 것

보안성(security) : 접근 통제, 권한 관리 등

안전성(safety) : 소프트웨어 오류로 인해 인명 등의 피해가 발생하지 않도록

사용성(usability) : 혼란스러워 하거나 사용하는 순간에 고민하지 않게

제약 사항

자바 언어를 사용해 개발하고, CBD 개발 방법론을 적용해야 한다.

레드햇 리눅스 엔터프라이즈 버전에서 실행해야 한다.

윈도우 운영체제와 리눅스 운영체제에서 모두 실행할 수 있어야 한다.

가용성, 변경용이성, 성능, 보안, 시험용이성, 사용편의성…

**테스트케이스 설계**

**테스팅이란?**

어플리케이션 또는 시스템(구성요소를 포함해서)의 동작과 성능, 안정성이 요구사항을 만족하는지 확인하는 과정이다.

요구사항을 만족시키지 못함 = 결함(Defect)

결함(Defect)와 에러는 다른 의미임

\* 버그(실수) : 프로그래머의 의도와 다르게 작동

\* 에러(Error) : Exception(예외상황)과 비슷, 프로그램이 정상적으로 실행되지 않음

\* 결함(Defect) : 요구사항을 만족시키지 못함

**테스팅의 원리**

-완벽한 테스팅(Exhaustive Testing)은 불가능하다.

모든 가능성(입력과 사전조건의 모든 조합)을 테스팅하는 것은 지극히 간단한 소프트웨어를 제외하고는 가능하지가 않다.

항공, 의료, 원자력설비 등 안전이 최우선인 소프트웨어인 경우, 더 강도높은 테스팅을 할 뿐이지, 완벽한 테스트가 수행되는 것은 아니다.

-테스팅을 개발 초기에 시작한다.

코딩을 한 후에 테스트케이스를 도출하는 것이 아니다.

요구사항 정의서와 설계서 등의 개발산출물을 분석하여 테스트케이스를 도출하는 과정을 통해 결함을 발견해야 한다.

-결함 집중(Defect clustering)

대다수의 결함들은 소수의 특정 모듈에 집중되어 발생한다. 프로그램의 복잡도가 높은(중요도가 높은, 사용빈도가 높은) 프로그램에서 집중발생.

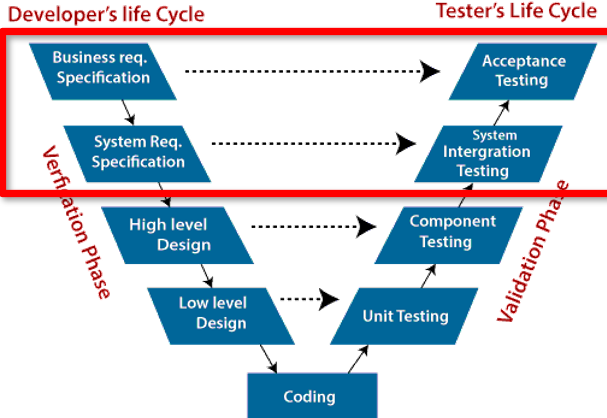
-살충제 패러독스(Pesticide paradox)

동일한 테스트케이스로 동일한 테스트를 반복적으로 수행한다면, 나중에는 더 이상 새로운 결함을 찾아내지 못한다.(죽을 때까지 계속 뿌리고 뿌리고 뿌리고… 모기가 내성이 생김, 살충제 업그레이드, 다시 죽을 때까지 뿌리고 뿌리고 뿌리고… 살충제=테스트)

테스트케이스는 정기적으로 리뷰하고 추가해야 한다.

**V모델**

개발 프로세스에 따라, 각 개발단계에 대응하는 테스트단계를 도식화한 것



저레벨은 바로 테스트하니까 금방 잡지만, 요구사항에 대한 테스트는 늦게 실행되므로 테스트 케이스를 미리 정의하지 않으면 곤란.

**테스트 레벨(규모)**

-Unit Test

특정 클래스의 메소드 단위의 테스트

의존관계를 끊고 테스트

-거래단위테스트

하나의 거래(=서비스)를 완성하기 위해 Sequence Diagram에서 표현되는 클래스 간의 의존관계를 포함한 테스트

-통합테스트

프로세스 단위의 테스트

프로세스와 프로세스의 결합을 테스트

-인수테스트

종료시점에서 하는 테스트

다양한 형태로 나타날 수 있음

보통 코어뱅킹에서는 영업점테스트를 의미하기도 함

-기능 테스트

기능 요구사항을 만족하는지 확인하는 테스트

-비기능 테스트

성능, 가용성, 보안성, 운영절차 등 품질속성을 만족하는지 확인하는 테스트

-회귀(Regression) 테스트(중요)

반복 테스트

과거에 정상적으로 작동하는 기능이 소프트웨어의 다른 기능을 수정한 이후에도 여전히 제대로 작동하는지 확인하는 테스트

ex. 게시글 입력기능 수정했는데 조회에 문제가 생김

-신구비교테스트

Legacy(As-Is)시스템의 결과와 To-Be시스템의 결과가 동일한지 검증하는 테스트

-싸이클테스트

일정기간 시간의 흐름을 갖는 업무 프로세스를 테스트

-데이터이행테스트 (Data Transfer)

Legacy(As-Is)시스템으로부터 이행한 데이터의 정합성을 확인하는 테스트

-테마집중테스트

상품출시, 정보보안, 권한관리, 회계분개 처리 등등 주제별 테스트

-대외연계테스트

대외기관과의 I/F의 정합성을 확인하는 테스트

(대내 : 시스템 안, 대외 : 우리 시스템 외)

**테스트 설계 기법에 따른 테스트의 종류(방식)**

-White Box Test vs Black Box Test

White Box Test = 구조 기반 테스트케이스 설계

코드와 개발 설계 등의 소프트웨어 구현 정보를 기반으로 테스트케이스를 도출한다. 잘 보임, 빠짐없이 설계 가능

수행된 테스트케이스를 바탕으로 테스트커버리지(Test Coverage, 얼마나 꼼꼼하게 했나)를 측정할 수 있다

\*테스트 커버리지 7~80% -> 잘했다

Black Box Test = 명세 기반 테스트케이스 설계

명세(Specification)를 기반으로 테스트케이스를 도출한다.

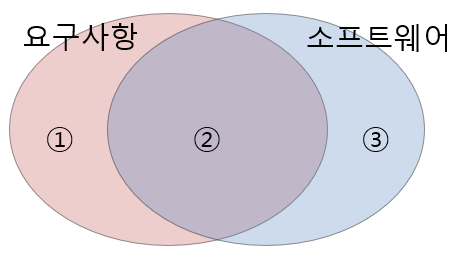
커버리지를 측정할 수 있으나, 그 의미가 구조 기반 기법의 커버리지에 비해 제한적이다.

-그리고, 경험 기반 테스트케이스 설계

과거의 결함 사례, 테스트 참여자의 경험과 지식을 기반으로 테스트케이스를 도출한다.

테스트케이스를 미리 설계하지 않고, 결함을 발견하고, 테스트케이스를 도출한다.

소프트웨어 개발 후 테스트케이스 도출이 가능하다.



① 요구사항에 명시되어 있으나 구현되지 않은 경우

명세 기반의 테스트케이스를 작성해야 한다.

② 요구사항대로 구현되었으나 때에 따라 정상작동 하지 않는 경우

요구사항을 기반으로 테스트케이스를 작성해 실행하고 추가적인 테스트케이스를 수행한다.

③ 요구사항에 명시되어 있지 않지만 구현된 경우

구조기반으로 테스트케이스를 작성해 실행한다.

**명세(Specification) = 예제 = 테스트케이스**

명세 : 요구사항 목록

요건이 기술되는 언어는 모호하여, 복잡한 요건의 경우는 예제로 구체화해야 코딩이 될 수 있다는 것을 이미 경험하였고, 그렇게 실천하고 있다.

공식예제로서 요건의 모호함을 방지한다.

요건을 기술하면서, 예제로서 구체화해내지 못한다면, 이 요건은 바뀔 가능성이 농후하다.

공식예제를 인수테스트케이스로 삼는다.

뭔가 잘못되었을 때, 이 공식예제를 가지고 커뮤니케이션할 수 있고, 공식예제를 정제하여 더욱 분명하게 합의할 수 있다.

**TDD(Test Driven Development)**

테스트 주도 개발

TDD의 가장 중요한 특징

외부설계가 끝나면, 개발스프린트가 시작하기 전에 테스트의 성공조건(=테스트케이스)에 대해서 합의한다.

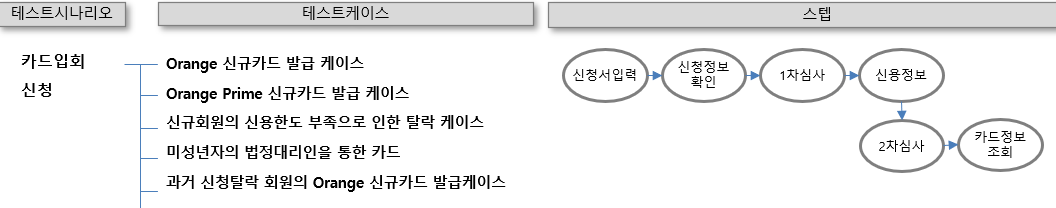
즉, 개발 이전에 테스트케이스를 작성해야 한다.

테스트자동화

반복적인 Regression테스트

\*ATDD(Automated) 자동화된 테스트주도개발

**테스트시나리오, 테스트케이스, 스텝**



테스트시나리오

애플리케이션의 테스트 되어야 할 기능/특징(a feature)을 기술한 일반적으로 한 줄 짜리 문장

보통 유스케이스나 업무프로세스로부터 도출됨

테스트 시나리오가 테스트 케이스와 일-대-다 관계를 가짐

테스트케이스

특정한 기능/특징(a feature)이 비즈니스 요구사항을 준수하는지 여부를 평가하기 위한 조건들의 집합

해당 테스트케이스를 확인하기 위한 상세스텝을 갖는다.

스텝

테스트케이스를 수행하기 위한 상세한 스텝

입력값(input values), 기대값(expected value)를 가짐

**테스트케이스 설계하기**

1단계 – 테스트시나리오 정하기

프로세스 다이어그램에서 설정한 프로세스와 동일

2단계 – 테스트케이스 도출하기

동등 분할(Equivalence Partitioning) - 가장 기본적인 원리. 배터리잔량표시 사례. 누락이 없도록.

테스트 시나리오에서 다루는 데이터 엔티티의 서브 타입

계좌개설 시나리오

그 외 업무규칙

3단계 – 스탭설계하기

외부설계 리소스단위를 업무프로세스 흐름에 맞추어 배치

(화면,

배치작업,

센터컷,

I/F,

채널서비스)

정규 프로세스를 활용한다. 자기만의 데이터셋을 만들지 말 것

특정 테스트케이스의 스탭흐름은 Self-Decisive

프로세스 다이어그램의 분기조건은 테스트케이스의 스탭 흐름에는 존재하지 않는다.

3단계 – 스탭설계하기

배치작업의 고려사항

처리건수의 정당성, Commit-interval의 확인

입력값과 기대값의 기재

추상적인 동치분류를 지양한다.

10,000원보다 작은 금액 → 15,000원과 9,500으로 작성

기능을 확인할 수 있는 다른 방법(이미 있는 기능을 활용)을 찾아본다.

예를 들면, 카드입회신청등록의 정상처리여부는 카드정보조회 기능을 이용하여 확인한다.

테스트케이스 목적에 부합하는 기대값만 압축하여 작성한다.

Regression Test를 위한 고려사항

테스트데이터를 재활용하지 않는다.

(ex. 등록테스트 -X> 삭제테스트)