

컴퓨터의 이해, 소프트웨어 공학

K-Digital Training

Intro to Computer Science

개발자가 하드웨어를 알아야 하는 이유

- 컴퓨터의 기본 구조를 이해하고 컴퓨터에서 동작하는 소프트웨어에 대한 개발이 이뤄져야 Clean Code를 작성할 수 있음!!

Computer

What is Computer?

What is Computer?

Compute + er

Computation vs Calculation

"calculation" implies a strictly arithmetic process,
whereas "computation" might involve applying rules in a systematic way

Computer vs Calculator

- **Stored Program** computer -> Computer
 - Stores and Executes instructions
- **Fixed Program** computer -> Calculator
 - just calculate

엇? 그럼 공학용 계산기는???

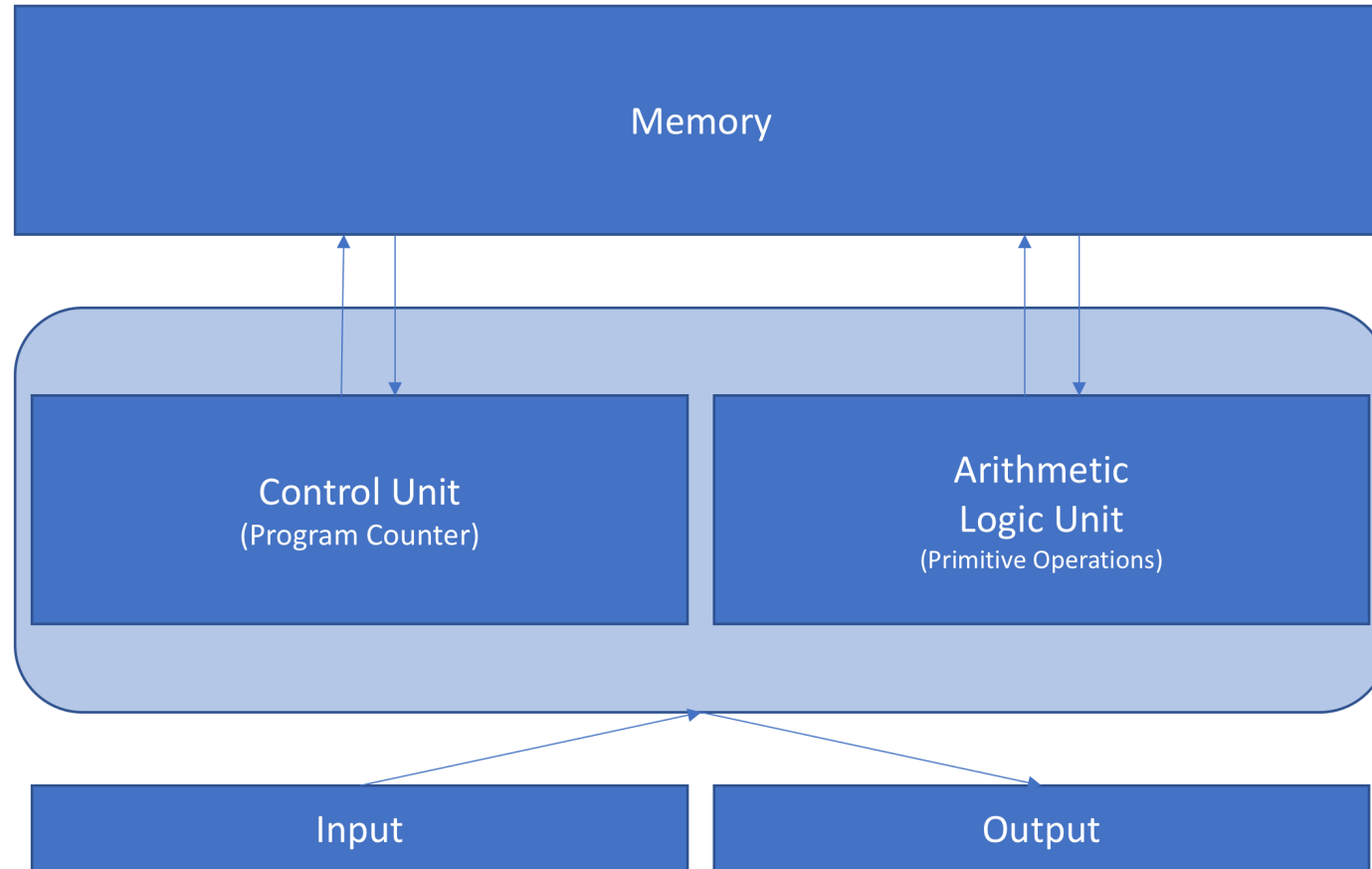
Computer Science and Engineering

- 컴퓨터의 소프트웨어를 다루는 학문
- 컴퓨터라는 물리적 기기를 연구하는 것이 아닌 Computer의 개념과 구조를 이해하고 구현하는 학문

Computer



Basic Computer Architecture



Basic Computer Architecture

- Program Counter - contains the address (location) of the instruction being executed at the current time
- ALU(Arithmetic Logic) - `+, -, *, /, AND, OR, NOT,`

CPU and MicroProcessor



Architecture Naming

- x86

8080 - 8bit

8086 - 16bit

8088 - 8bit

80286 - 16bit

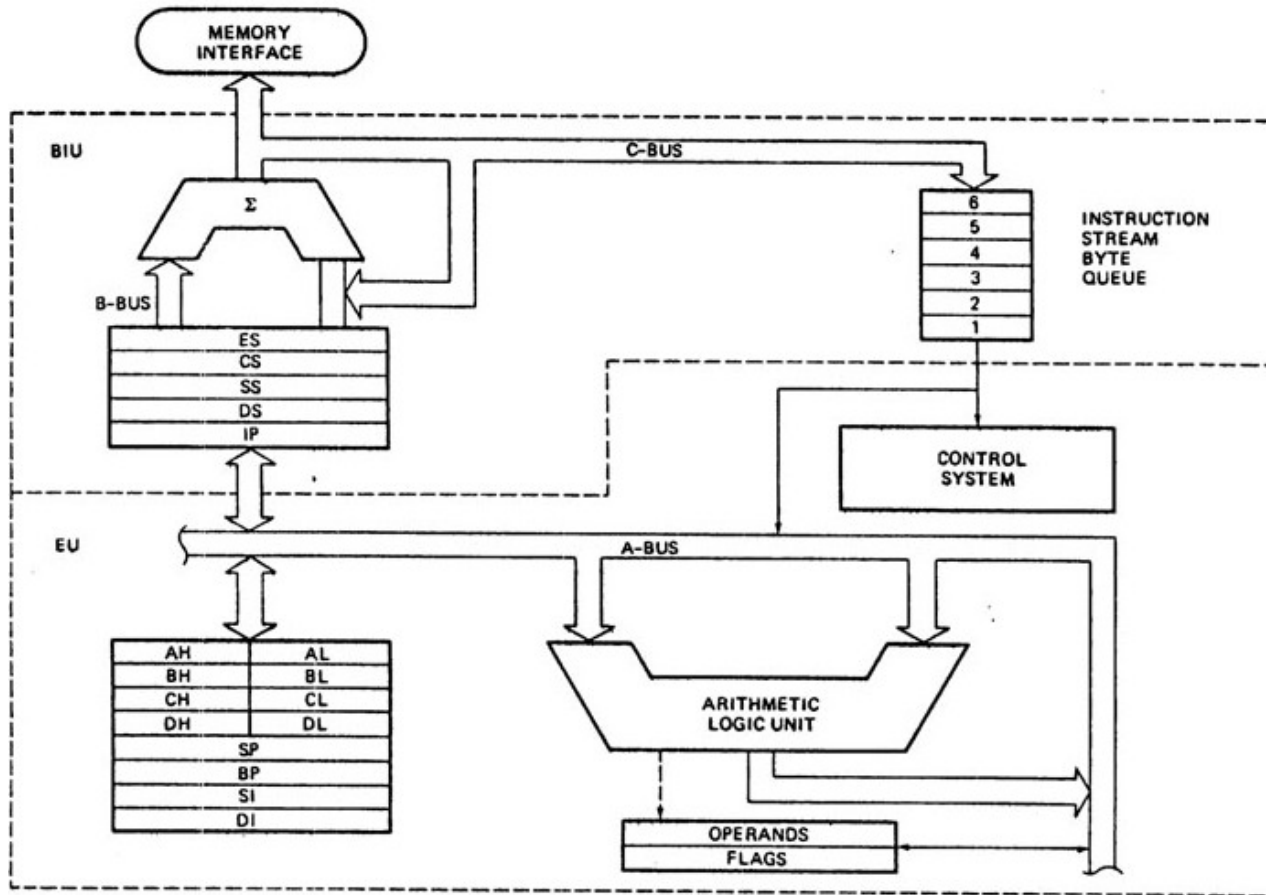
80386 - 32bit

..

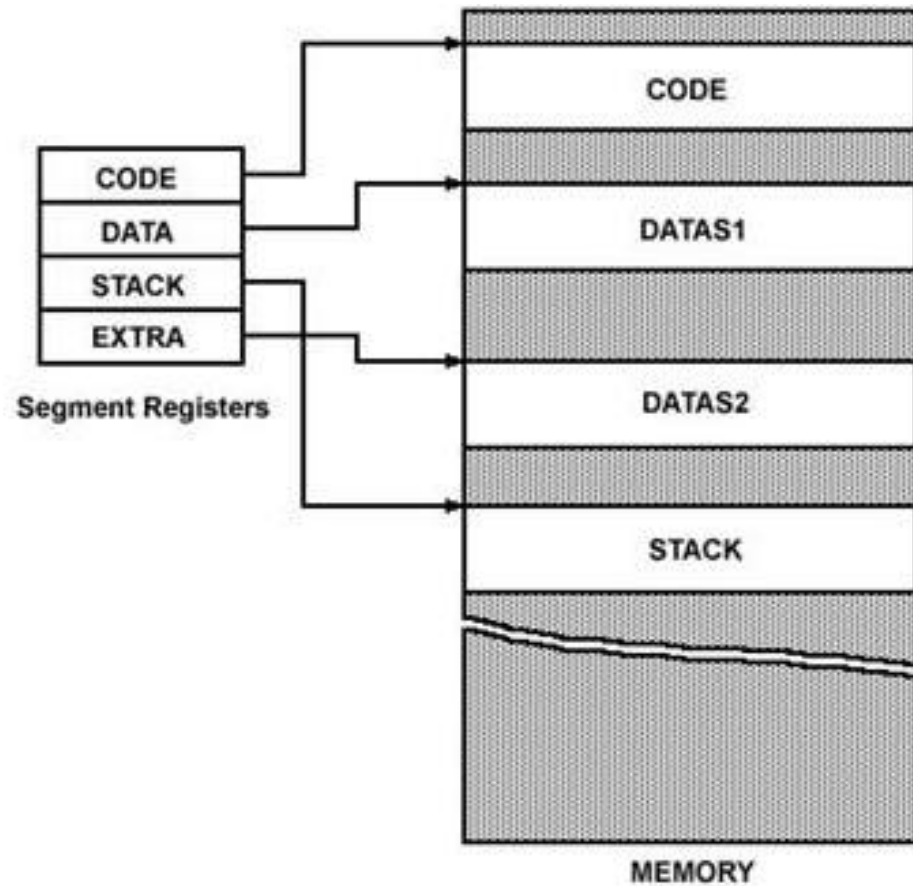
Architecture Naming

- IA64
Itanium - IA64 based 64bit, 1999
...
- AMD64
Opteron - x86-64based 64bit, 2003
Athlon, AMD Phenom, AMD FX
Ryzen
..
- ~~Intel64~~ == AMD64
Xeon - x86-64 based 64bit, 2004
Core 2
Core i Series
..

Block Diagram of 8086



Memory Segments of 8086



CISC&RISC Architecture

- Complex Instruction Set Computers
 - 복잡한 명령구조
 - 어드레싱에 강점
 - 전력 신경쓸 필요없이 고성능 컴퓨팅에 사용
 - Intel x86, AMD64, ..
- Reduced Instruction Set Computers
 - 명령어의 단순화
 - 메모리 접근 횟수가 적음
 - 저전력 프로세싱에 사용
 - SPARC, ARM, ..

Memory

- 컴퓨터에서 사용할 수 있도록 정보를 저장하는 공간

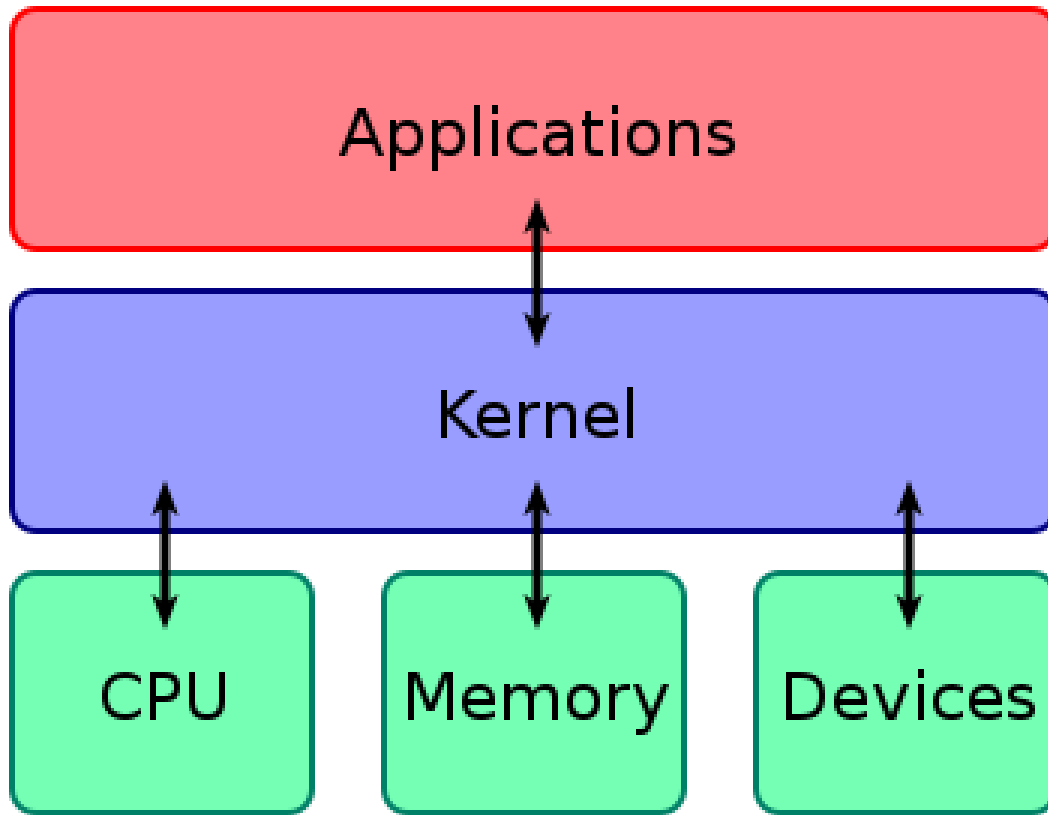
- Random Access Memory
 - 자유롭게 읽고 쓸 수 있는 주기억장치
 - 메모리의 주소로 그 위치에 접근
 - RAM의 어떤 위치로든 같은 시간에 접근(Random Access)
 - 컴퓨터가 느려지면 재부팅을 하세요!

- Read Only Memory
 - 전원이 공급되지 않아도 그 정보를 유지하는 주기억장치
 - 비싸거나 느려서 안정적인 정보를 저장해야 할 때 사용
 - BIOS, OS, Firmware 정보 저장에 쓰임

OS

- Operating System: 운영체제
- 시스템 하드웨어를 관리하고, 응용 프로그램 실행을 위한 하드웨어 추상화 플랫폼과 공통 시스템 서비스를 제공하는 시스템 소프트웨어

Kernel



- 하드웨어와 응용프로그램을 이어주는 운영체제의 핵심 시스템소프트웨어

Operating System == Kernel??

운영체제의 핵심 역할 수행

- 하드웨어, 프로세스 보안
- 시스템 자원 관리(스케줄링)
- 하드웨어 추상화 - 일관성 있는 인터페이스 제공

- Windows 10 - Windows NT 10.0
- MacOS - NXU/Darwin
- Linux - Linux

Type

- Single-tasking / Multi-tasking
 - 한번에 1개 / n개 의 프로그램을 동시 수행(achieved by time-sharing)
- Single-user / Multi-user
- Distributed

Hardware <--> Operating System <--> Application Software <--> User

Chronicles of OS

Unix

- Starting in the 1970s by AT&T
- Ken Thompson, Denis Ritchie, ..

Unix-like

- Solaris
- BSD
- MacOS

Linux

- Unix-clone OS
- GNU/Linux
- Sep 17 1991 by Linus Torvalds

Distribution of Linux

- Android
- Tizen
- Chrome OS
- ..

Windows

- CP/M-DOS -> MS-DOS
- Windows 1
- ..
- Windows 10
- Windows 95
- Windows 98
- Windows 2000

Windows 9x vs Windows NT

MS-DOS based -> 16bit

WindowsNT Kernel(3.1) based -> 32bit

WindowsNT Kernel(6.1) based -> x86-64(AMD64)

Database

data

- 컴퓨터가 처리할 수 있는 문자, 숫자, 소리, 그림 따위의 형태로 된 정보.
- Latin "Datum"의 복수형 "Data"에서 유래

Database

- 체계화된 데이터의 모임
- 여러 응용 시스템들의 통합된 정보들을 저장하여 운영할 수 있는 공용 데이터들의 묶음

DB?? DBMS??

DBMS(DataBase Management System)

- 데이터의 모임인 Database를 만들고, 저장, 관리 할 수 있는 기능을 제공하는 응용프로그램
- Oracle, Mysql, MariaDB, DB2, MS SQL Server, ..

DBMS의 조상님

CURSOR <-- --> Char: ← → Word: Home End	UP DOWN Field: ↑ ↓ Page: PgUp PgDn Help: F1	DELETE Char: Del Field: ^Y Record: ^U	Insert Mode: Ins Exit/Save: ^End Abort: Esc Memo: ^Home
--	--	---	--

FIRSTNAME	Claire
LASTNAME	Buckman
ADDRESS	8307 Santa Anita Blvd
CITY	Oxnard
STATE	CA
ZIPCODE	93034
PHONE	(555)456-9059

EDIT	<C:>	CLIENTS	Rec: 1/49
------	------	---------	-----------

DBMS의 조상님

dBASE

- 마이크로컴퓨터용 최초의 DBMS
- 1979년 Ashton이 개발
- SQL이 아닌 독자 스크립트 언어로 실행 -> dbf 파일 생성

Characteristics

- 데이터의 무결성
- 데이터의 중복 방지
- 보안(추상화, 접근권한)
- 성능 향상
- 프로그램 수정과 유지 보수 용이

Differences between DataBase & File System


자기기술성

File System

- .hwp -> 한글
- .doc -> Microsoft Word
- .xls -> Microsoft Excel

DB

- Only SQL(RDBMS)

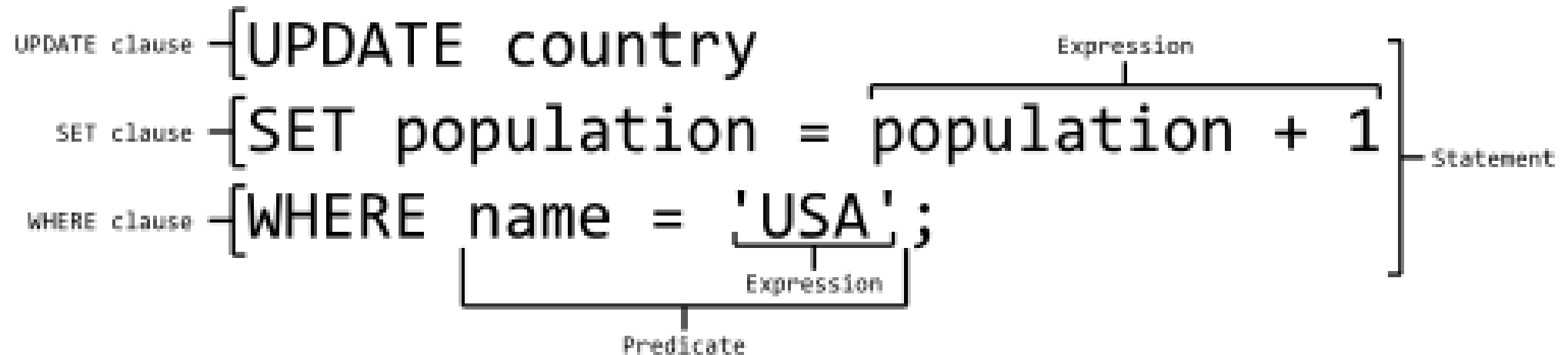


A screenshot of a file explorer window displaying a list of files. The files are listed in a single column, each preceded by a small blue icon representing a document. The file names are in Korean and include file extensions. The list includes various versions and drafts of documents, such as '졸업논문.hwp', '졸업논문수정.hwp', and '졸업논문최종완성본final최종.hwp'. The last file in the list is 'ㅠ유서.hwp'.

- 졸업논문.hwp
- 졸업논문수정.hwp
- 졸업논문수정1.hwp
- 졸업논문수정2.hwp
- 졸업논문완성본.hwp
- 졸업논문완성본1.hwp
- 졸업논문완성본2.hwp
- 졸업논문최종완성본.hwp
- 졸업논문최종완성본1.hwp
- 졸업논문최종완성본2.hwp
- 졸업논문최종완성본final.hwp
- 졸업논문최종완성본final1.hwp
- 졸업논문최종완성본final2.hwp
- 졸업논문최종완성본final최종.hwp
- 졸업논문최종완성본final최종1.hwp
- 졸업논문최종완성본final최종2.hwp
- ㅠ유서.hwp

SQL(Structured Query Language)

데이터 관리를 위해 설계된 특수 목적의 프로그래밍 언어



SQL - 데이터 정의언어

데이터를 정의

CREATE - DB 개체 정의

DROP - DB 개체 삭제

ALTER - DB 개체 정의 변경

SQL - 데이터 조작언어

데이터 검색, 등록, 삭제, 갱신

INSERT - 행, 테이블 데이터 삽입

UPDATE - 테이블 업데이트

DELETE - 특정 행 삭제

SELECT - 테이블 검색 결과 집합

SQL - 데이터 제어언어

데이터 액세스 제어

GRANT - 작업 수행권한 부여

REVOKE - 권한 박탈

RDBMS vs NoSQL

구분	RDBMS	NoSQL
형태	Table	Key-value, Document, Column
데이터	정형 데이터	비정형 데이터
성능	대용량 처리시 저하	작은 수정시 저하
스키마	고정	Schemeless
장점	안정적	확장성, 높은 성능
유명	Mysql, MariaDB, PostgreSQL	MongoDB, CouchDB, Redis, Cassandra

RDBMS

[PostgreSQL Docs](#)

[MariaDB Docs](#)

name	age
John	17
Mary	21

```
rdb =  
{  
    name:[John, Mary],  
    age:[17, 21]  
}
```

Table == Relation

Primary Key	Attribute1	Attr2	Attr3	Attr4
Tuple1				
Tuple2				
Tuple3				
Tuple4				

NoSQL

MongoDB Docs

```
nosql =  
[  
    {  
        name: John,  
        age: 17  
    },  
    {  
        name: Mary,  
        age: 21  
    },  
    ...  
]
```


Document vs Key-value

```
document
{
    key: value,
    key: {
        key: value,
        key: value
    }
}
```

```
key-value
{
    key: value,
    key: value,
    key: value
}
```

noSQL

- 확장가능성, 스키마 없는 데이터 모델에 유리
- Row, Document, key-value 등 다양

RDBMS와 다른점

- Schemaless
- Join 불가능(reference 등으로 구현)
- No Transaction
- 수평확장 용이

종류

- {Key:Value} = Redis
- [Column] = Cassandra, HBase
- Document {Key:{Key:Value}} = CouchDB, MongoDB

MongoDB

- BSON(Binary JSON) 기반 Key-Value Store
- JSON 형태 문서
- Collection -> Document -> Key:Value Data

Requirements

- DB instance(mLab)
- pymongo(`$ pip install pymongo`)
- pandas(`$ pip install pandas`)
- requests(`$ pip install requests`)
- jupyter notebook(`$ pip install jupyter`)

MongoDB with jupyter

connect

```
from pymongo import MongoClient  
client = MongoClient("mongodb://..")  
client.{DBName}.collection_names()
```


Assign DB, Collection

```
db = client.{DBName}  
new_collection = db.{CollectionName}
```

or

```
db = client[{DBName}]  
new_collection = db[{CollectionName}]
```

INSERT data

```
some_user = {  
    "name": "Fastcampus",  
    "email": "help@fastcampus.co.kr",  
}  
new_collection.insert_one(some_user)
```

SELECT

```
new_collection.find_one( )
```

or

```
query = {}  
new_collection.find_one(query)
```

SELECT all data

```
query = {}  
cursor = new_collection.find(query)  
[item for item in cursor]
```

SELECT * WHERE name = "kdt"

```
query = {"name":"kdt"}  
new_collection.find_one(query)
```

INSERT lots of data in one time

```
data_list = [  
    {  
        "name": "jyp",  
        "email": "jyp@kdt.co.kr",  
    },  
    {  
        "name": "gd",  
        "email": "gd@kdt.co.kr",  
    },  
]  
  
new_collection.insert_many(data_list)
```

WHERE in ("jyp", "gd")

```
query = {  
    "name": {  
        "$in": ["jyp", "gd"]  
    }  
}  
new_collection.find(query)
```

AND, OR

```
query = {  
    "name": "jyp",  
    "email": "jyp@kdt.co.kr",  
}  
new_collection.find(query)
```

```
query = {  
    $or: [  
        {"name": "gdragon"},  
        {"email": "gd@kdt.co.kr"}  
    ]  
}  
new_collection.find(query)
```


Operator

```
{field:{<operator>:<value>}}
```

Operator	NoSQL
=	\$eq
!=	\$ne
>	\$gt
>=	\$gte
<	\$lt
<=	\$lte
IN	\$in

COUNT(*)

```
new_collection.count( )
```

GROUP BY

```
# like temporary table
cursor = collection.aggregate([
    {
        "$group": {condition}
    }
])
```

pymongo with requests

import requests

```
import requests  
  
url = ""  
headers = {}  
  
response = requests.get(url, headers=headers)
```

json decode

```
item_list = response.json()[ "items" ]
```

insert lots of data

```
item_list.insert_many(item_list)
```

Store data into MongoDB

How to Design Database?

Schema

- Database의 구조와 제약조건에 대한 전반적인 명세 기술
- Database의 Blueprint
- 외부(서브)스키마, 개념스키마, 내부스키마로 구성

외부(서브) 스키마

- 프로그램 사용자가 필요로 하는 데이터베이스의 논리적인 구조를 정의

개념 스키마

- 조직 전체의 관점에서의 구조와 관계를 정의
- 외부 스키마의 합과 그 사이의 데이터의 관계 등등
- 일반적인 스키마의 정의

내부 스키마

- 저장장치의 입장에서 데이터베이스가 저장되는 방법을 기술

Install MongoDB, Robomongo

MongoDB

Software Engineering

Software Engineering

Definition

Software engineering (SWE) is the application of engineering to the design, development, implementation, testing and maintenance of software in a systematic method.

--> 소프트웨어의 개발, 운용, 유지보수 등의 생명 주기 전반을 체계적이고 서술적이며 정량적으로 다루는 학문

DevOps

used to refer to a set of practices that emphasizes the **collaboration and communication** of both software developers and other information-technology (IT) professionals while automating the process of software delivery and infrastructure changes.

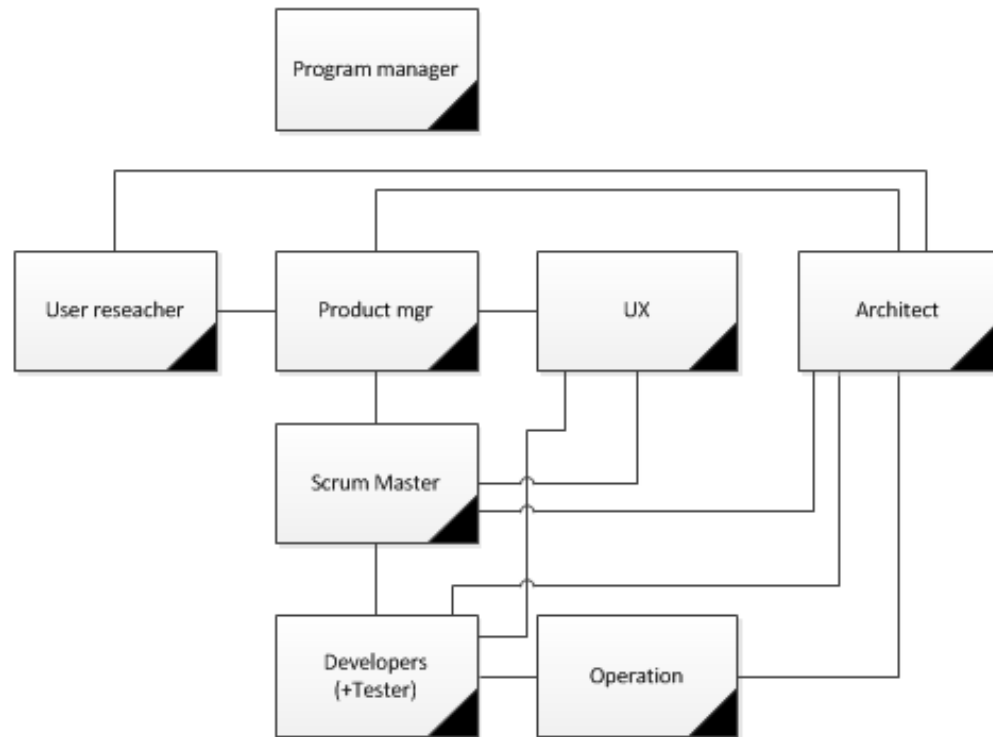
It aims at establishing a **culture** and **environment** where building, testing, and releasing software can happen **rapidly, frequently**, and more **reliably**.

DevOps

- 기존의 개발과 운영 분리로 인해 발생하는 문제들
문제 발생 -> 비방 -> 욕 -> 상처 -> 원인분석 -> 문제해결
- 좋은 소프트웨어를 위한 필수조건
 - 기획팀과의 원활한 소통으로 요구사항을 충실히 반영
 - 운영팀과의 원활한 소통으로 소비자 불만과 의견을 반영

DevOps

운영과 개발을 통합하여 커뮤니케이션 리소스를 줄이고, 개발 실패 확률을 줄임과 동시에 보다 안정적인 서비스를 운영할 수 있음!!



Software Development Life Cycle

SDLC란?

- 소프트웨어를 계획, 개발, 시험, 배포하는 과정
- 요구사항 분석 -> 설계 -> 구현 -> 테스트 -> 유지 및 보수

Requirements Analysis

Requirements

무엇이 구현되어야 하는가에 대한 명세

시스템이 어떻게 동작해야 하는지 혹은 시스템의 특징이나 속성에 대한 설명

Requirements Analysis

시스템 공학과 소프트웨어 공학 분야에서 수혜자 또는 사용자와 같은 다양한 이해관계자의 상충할 수도 있는 요구사항을 고려하여 새로운 제품이나 변경된 제품에 부합하는 요구와 조건을 결정하는 것과 같은 업무

Requirements Analysis

나(개발자)와 클라이언트(사장) 모두를 만족시키기 위한 연결고리

Requirements Analysis

- 요구사항 유도(수집): 대화를 통해 요구사항을 결정하는 작업
- 요구사항 분석: 수집한 요구사항을 분석하여 모순되거나 불완전한 사항을 해결하는 것
- 요구사항 기록: 요구사항의 문서화 작업

Requirements Layer



Business Requirements

| "Why"

Business Requirements

"왜" 프로젝트를 수행하는지

- 고객이 제품을 개발함으로써 얻을 수 있는 이득
- Vision and Scope(비전과 범위)

User Requirements

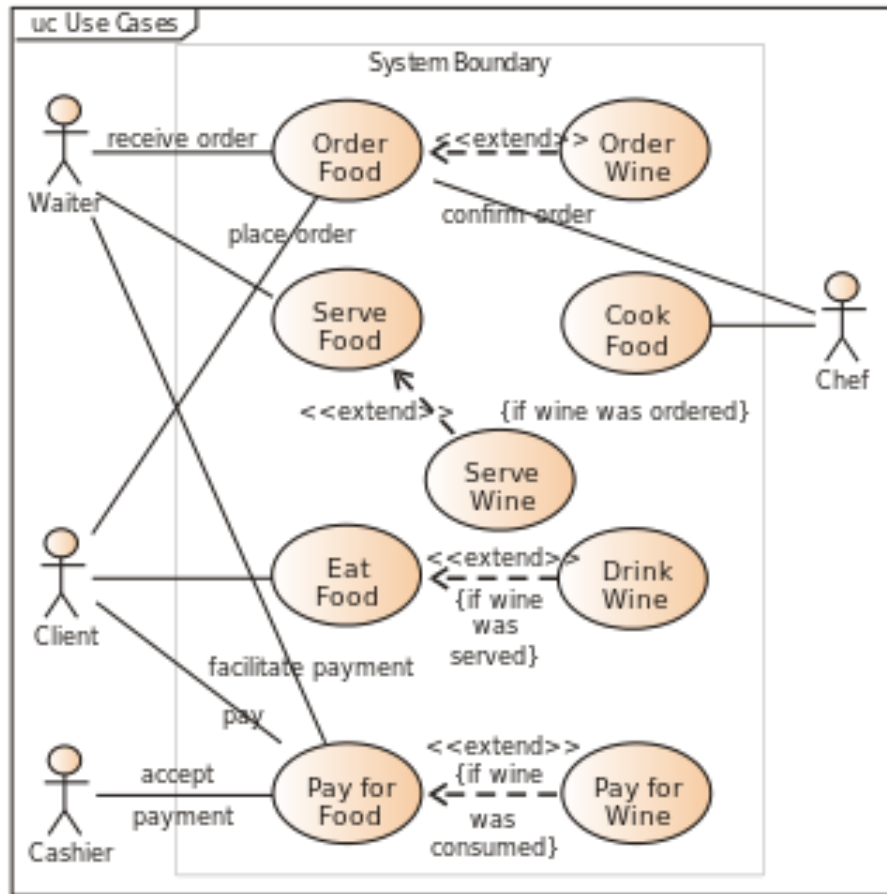
| "What"

User Requirements

사용자가 이 제품을 통해 할 수 있는 "무엇"

- Use cases, Scenarios, User stories, Event-response tables, ..

use case diagram



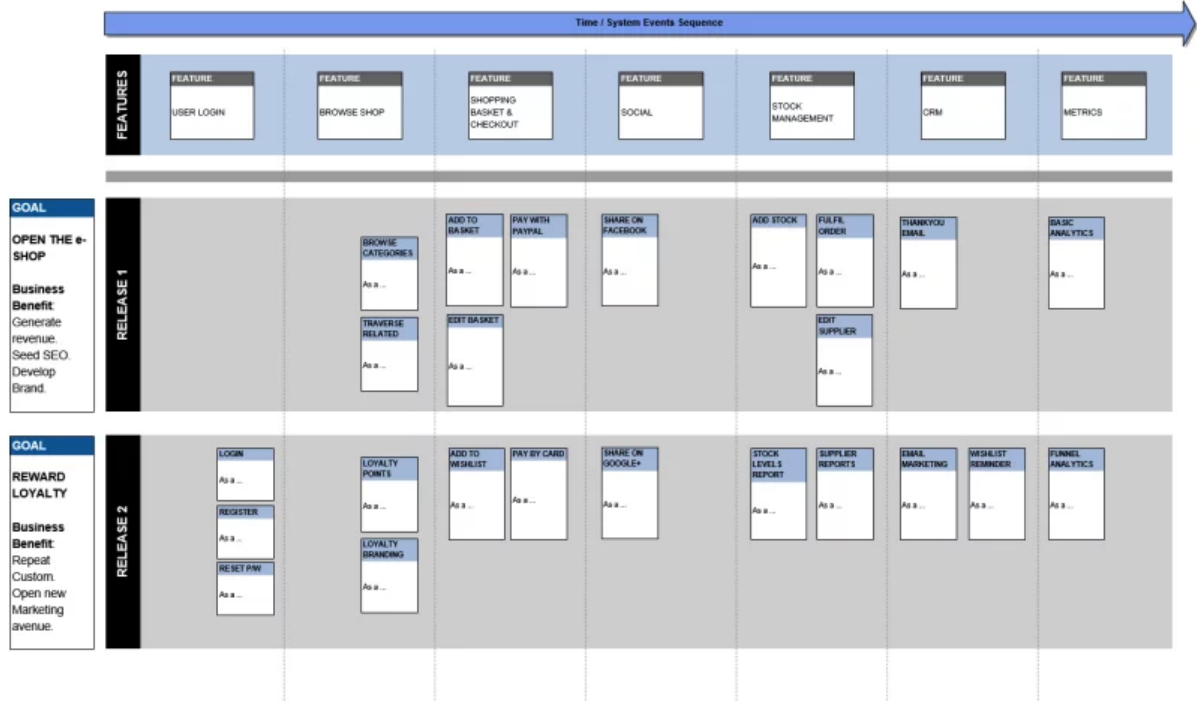
user stories

02 VERSION 5

User Story Map

Small Map with 2 Releases

Wednesday, 16 March 2016



Functional Requirements

| "What"

Functional Requirements

개발자가 이 제품의 "무엇"을 개발할 것인지

- '~ 해야 한다' 로 끝나 반드시 수행해야 하거나 사용자가 할 수 있어야 하는 것들에 대해 작성

System Requirements

- 여러개의 서브 시스템으로 구성되는 제품에 대한 최상위 요구사항을 설명
- 컴퓨터: 모니터 + 키보드 + 마우스 + 본체 + 스피커

Business Rules

- 비즈니스 스트럭처의 요구나 제약사항을 명세
- "유저 로그인을 위해서는 페이스북 계정이 있어야 한다."
- "유저 프로필 페이지에 접근하기 위해서는 로그인되어 있어야 한다"

Quality Attribute

- 소프트웨어의 품질에 대해 명세
- "결제과정에서 100명의 사용자가 평균 1.5초의 지연시간 안에 요청을 처리해야 한다"

External Interface

- 시스템과 외부를 연결하는 인터페이스
- 다른 소프트웨어, 하드웨어, 통신 인터페이스, 프로토콜, ..

Constraint

- 기술, 표준, 업무, 법, 제도 등의 제약조건 명세
- 개발자들의 선택사항에 제한을 두는 것

When the well is full, it will run over.

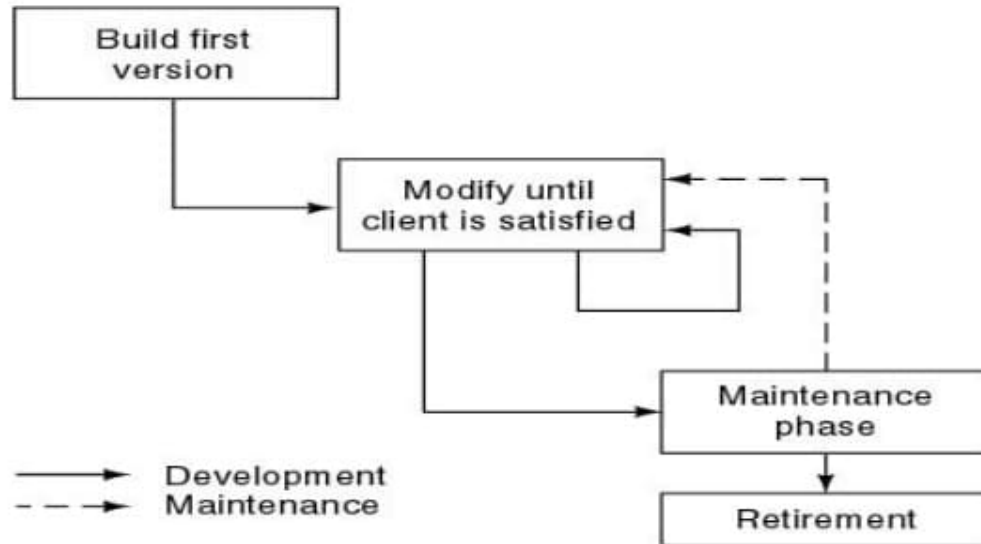
지나치게 자세한 명세작성

- 명세서는 말 그대로 명세일 뿐, 실제 개발 단계에서 마주칠 모든 것을 담을 수 없음
- 개발을 언어로 모두 표현할 수 없음
- 명세서가 완벽하다고 해서 상품도 완벽하리란 보장은 없음
- 때로는 명세를 작성하기 보단 프로토타이핑이 더 간단할 수 있음.

Software Development Lifecycle Process Model

Build-fix Model

[Build and Fix Model]



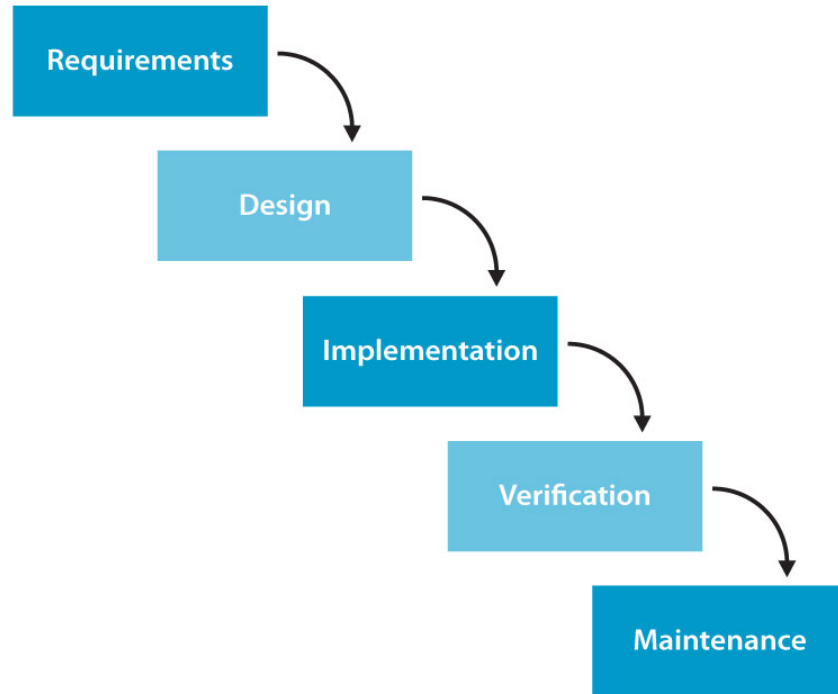
Build-fix Model

설계없이 일단 개발, 만족할 때까지 수정

시작이 빠름

계획이 정확하지 않음, 개발 문서가 없고 진행상황 파악이 힘들

Waterfall Model



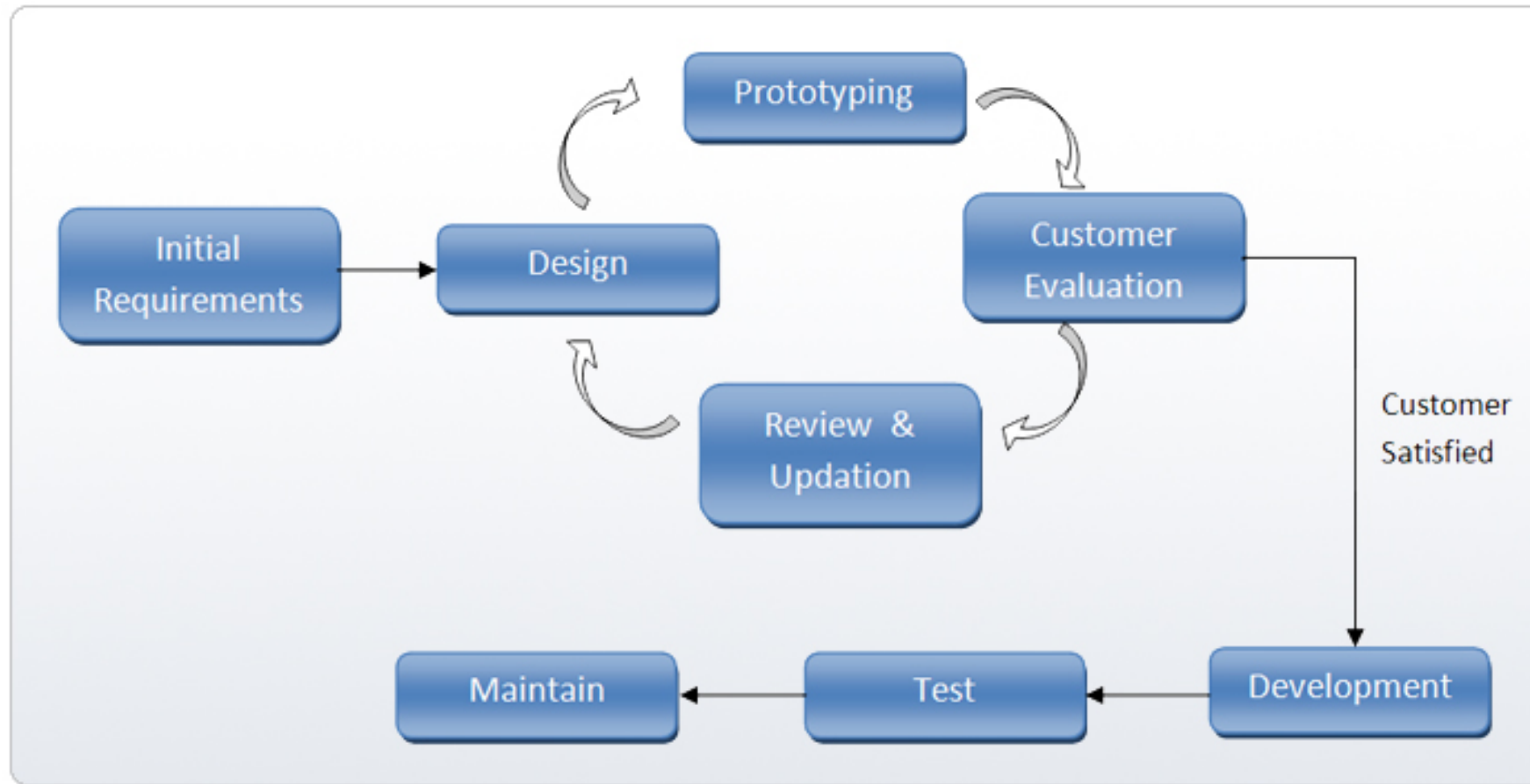
Waterfall Model

순차적인 개발 모델, 가장 많이 사용됨

정형화된 접근 가능, 체계적인 문서화 가능

직전 단계가 완료되어야 진행 가능

Prototype Model



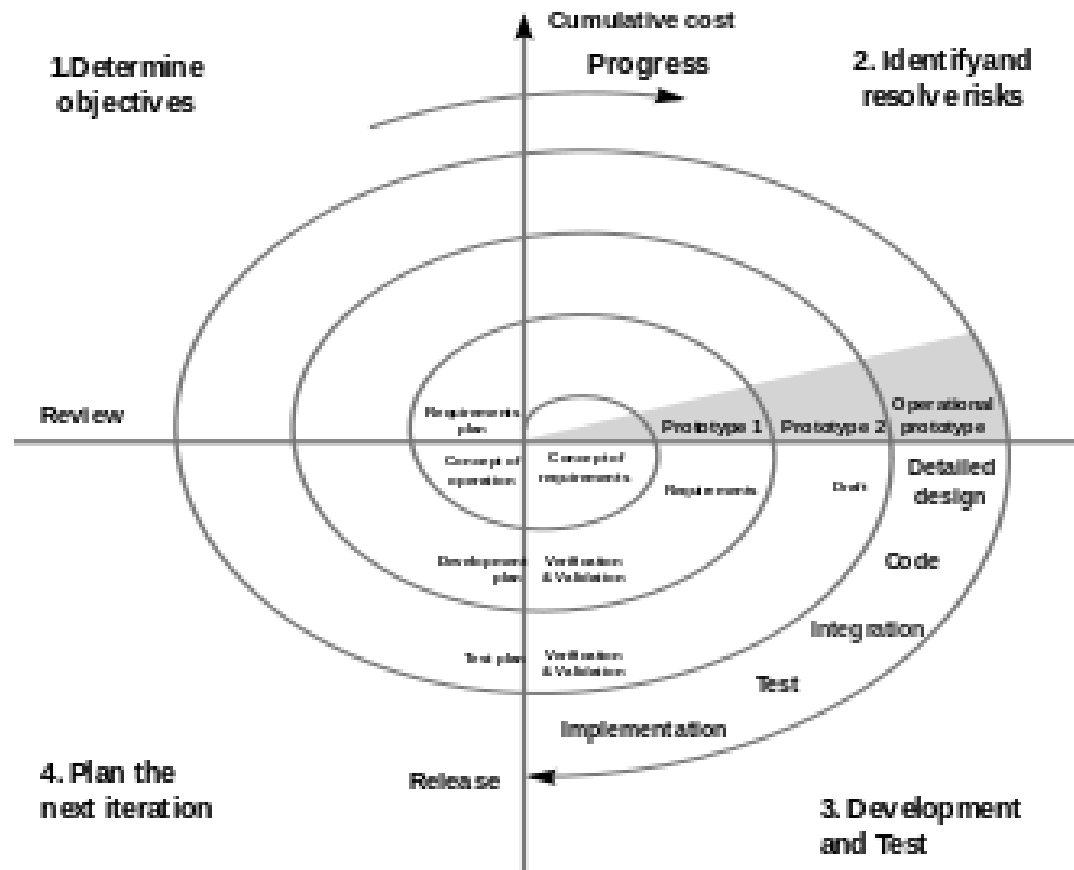
Prototype Model

고객 요구사항을 적극적으로 반영하는 모델

빠른 개발과 고객 피드백을 빠르게 반영할 수 있음

대규모 프로젝트에 적용하기 힘들

Spiral Model



Spiral Model

대규모 or 고비용 프로젝트

프로젝트의 위험요인을 제거해 나갈 수 있음

각 단계가 명확하지 않음

이외에도..

- RAD(Rapid Application Development) Model
- Iterative Development Model
- V Model
- Component Based Development

Software Development Process

in Agile

UP(Unified Process)

- 도입(분석위주), 상세(설계위주), 구축(구현위주), 이행(최종 릴리즈)의 반복

XP(eXtreme Process)

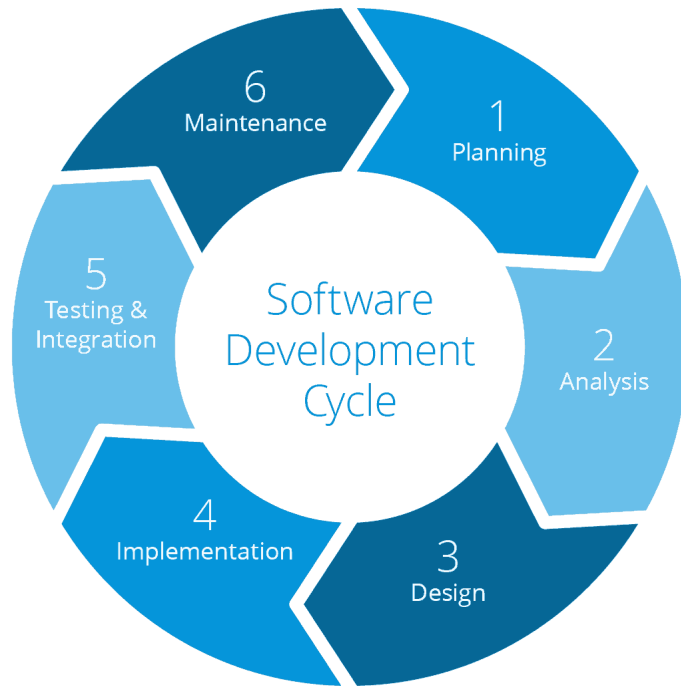
- 스크럼 마스터가 주도적으로 프로세스를 주도하며, 고객과 개발자 사이의 소통을 중시함
- Product Owner와 Development Team, Customer로 roles을 구분하고 각자의 역할에 충실
- TDD 중시

TDD

Test Driven Development

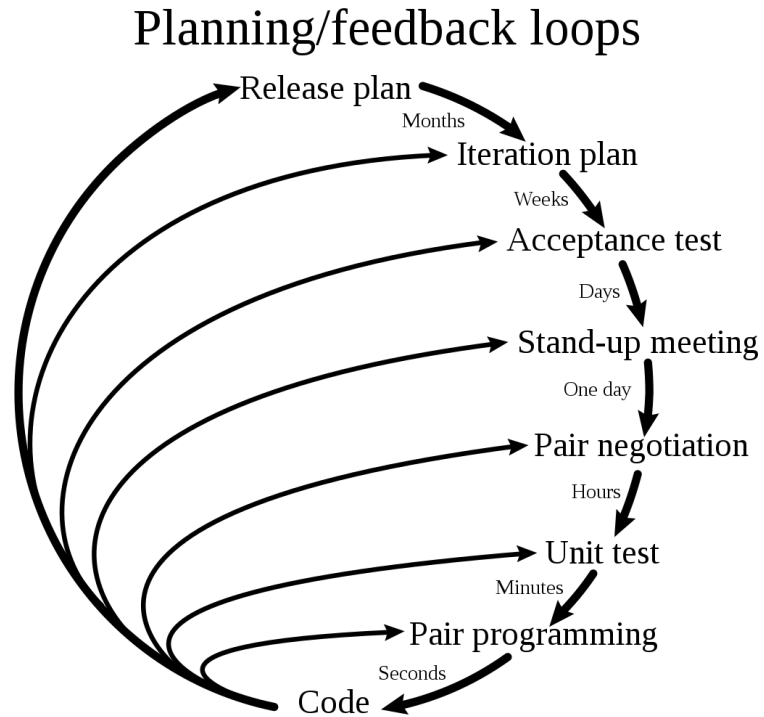
- 객체지향적
- 재설계 시간 단축
- 디버깅 시간 단축
- 애자일과의 시너지(사용자 중심적)
- 테스트 문서 대체
- 추가 구현 용이

Agile Software Development



- 프로젝트의 생명주기동안 반복적인 개발을 촉진하는 개발모델
- TMP(Too Much Plan)과 TLP(Too Less Plan)의 타협
- Code-oriented Methodology
- XP(eXtreme Programming), Scrum 등의 상세 방법론 존재

eXtreme Programming

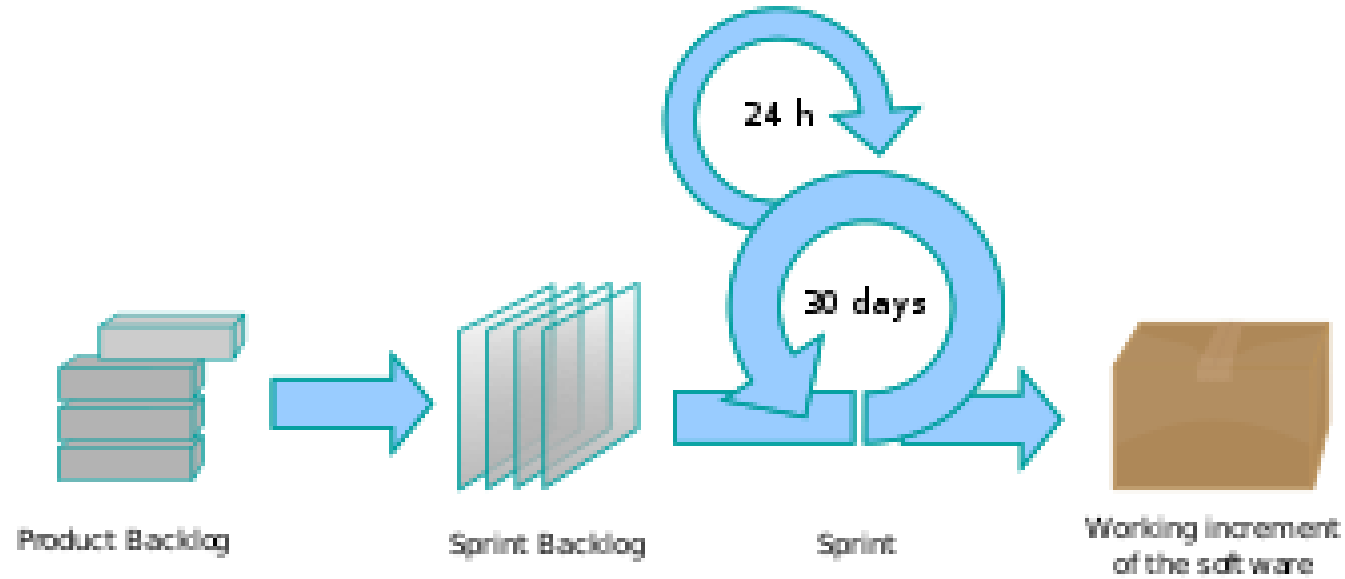


- 고객 중심의 양질의 소프트웨어를 빠른 시간안에 전달한다!
- Business Requirements의 변동이 심한 경우 적합한 개발 방법.
- Test! Test! Test! - Test Driven Development

Key Process - XP

- Role: Project Manager, Technical writer, Interaction Designer, Architect, Tester, Programmer, User(Stakeholder)
- Planning: 2주 주기로 계획을 세우고, 프로토타입을 통해 개발 방향 점검
- Test-Driven Development: Test Code를 먼저 작성하고 기능을 개발한 뒤, 테스트를 통해 검증
- Pair Programming: 2인 이상의 팀을 이뤄 한 명이 Drive 하고, 한명은 QA 또는 Navigator로 참여.

Scrum



- 상호,점진적 개발방법론
- 개발할 기능, 수정사항에 대해 우선순위를 부여한 뒤, 이 순서대로 Task 진행
- 매일 15분의 회의 진행
- 1~4주의 Sprint(기획~리뷰)

Key Process - Scrum

- Role: Product Owner, Scrum Master, Developer
- Product Backlog: 제품 전체의 요구사항
- Planning meeting: Sprint 목표와 Sprint Backlog 계획
- Sprint Backlog
- Daily Scrum: 어제 한 일, 오늘 할 일, Issue 등 공유

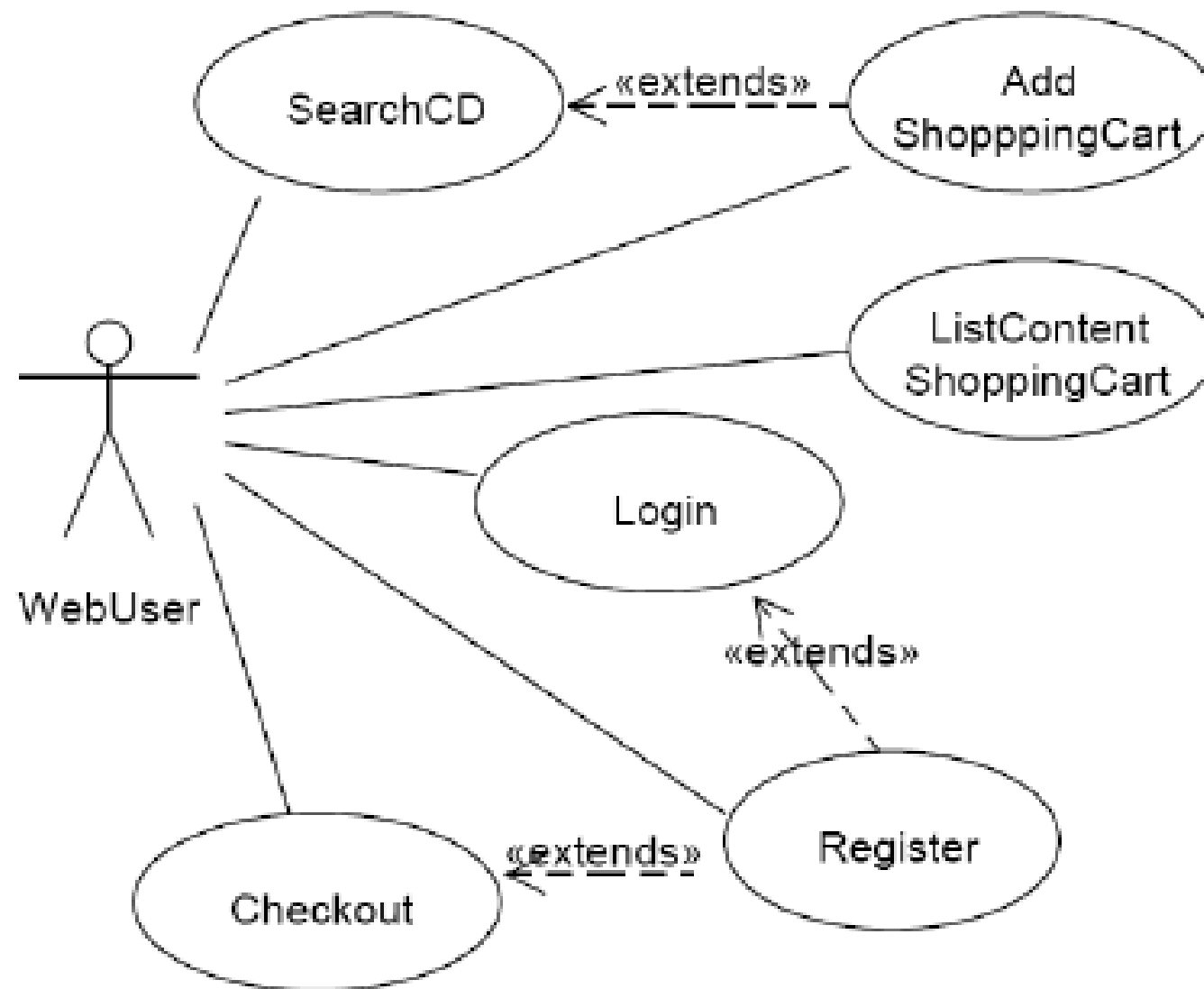
Scrum with XP

1. Sprint 주기: 2주(Deadline: 중간발표, 최종발표일)
2. Requirement Analysis -> Design -> Implementation(Scrum with XP)
 - Planning Meeting: Sprint 기간 중 구현할 내용계획
 - Sprint Backlog 작성(w/ github projects): M/H 추산 및 분배를 통해 Task 분배
 - Daily Scrum(어제 한 일, 오늘 할 일, Issue): 매일 일과 시작 전 15분 간
 - Test-Driven Development(Optional): 테스트코드 먼저 작성 후 본 코드 작성

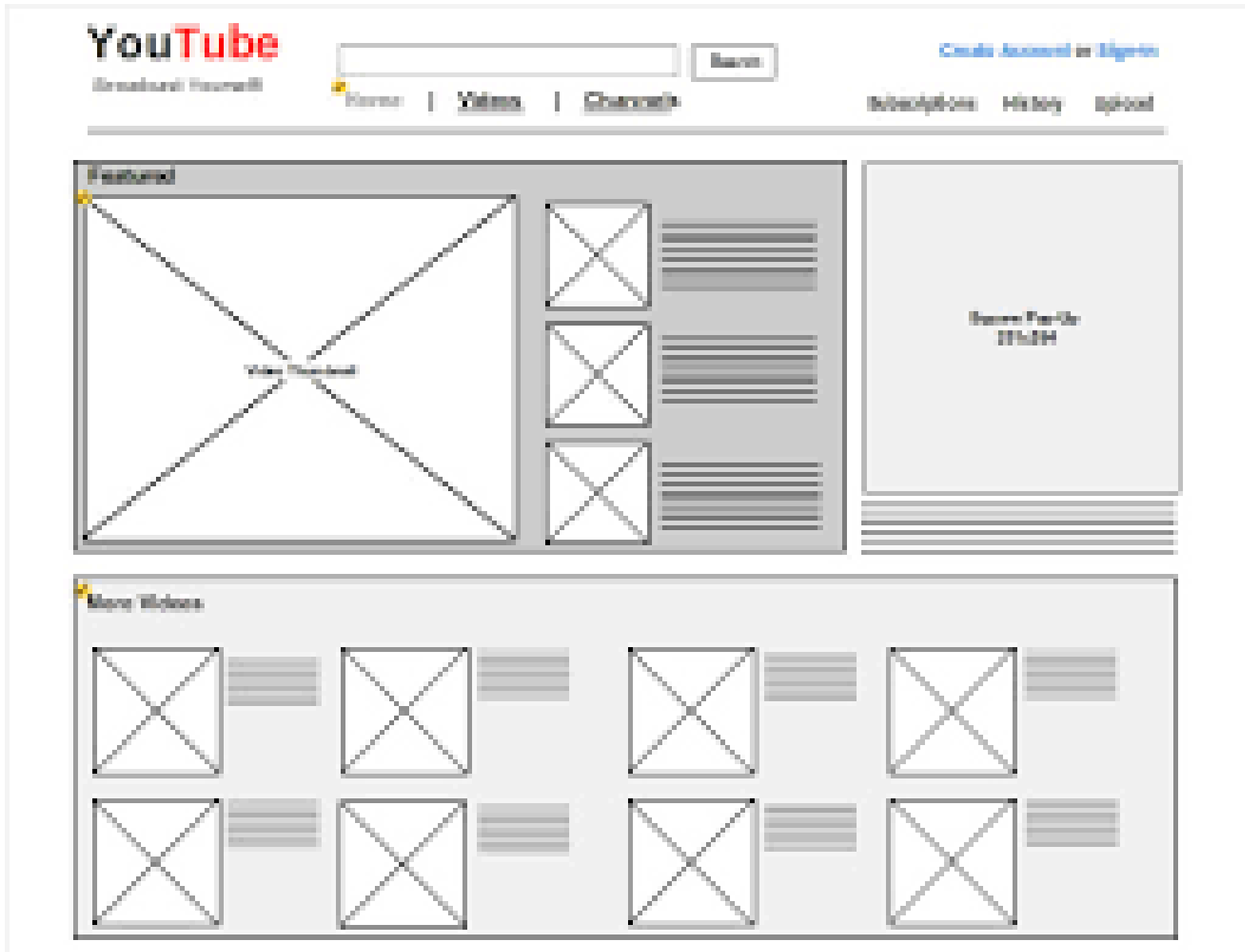
Before Implementation(1)

- Requirement Analysis
 - Client
 - Functional
 - External interface
 - Performance
- Wireframe, Usecase, Storyboard
- Design Prototype
- ERD(Entity Relationship Diagram)
- API Design

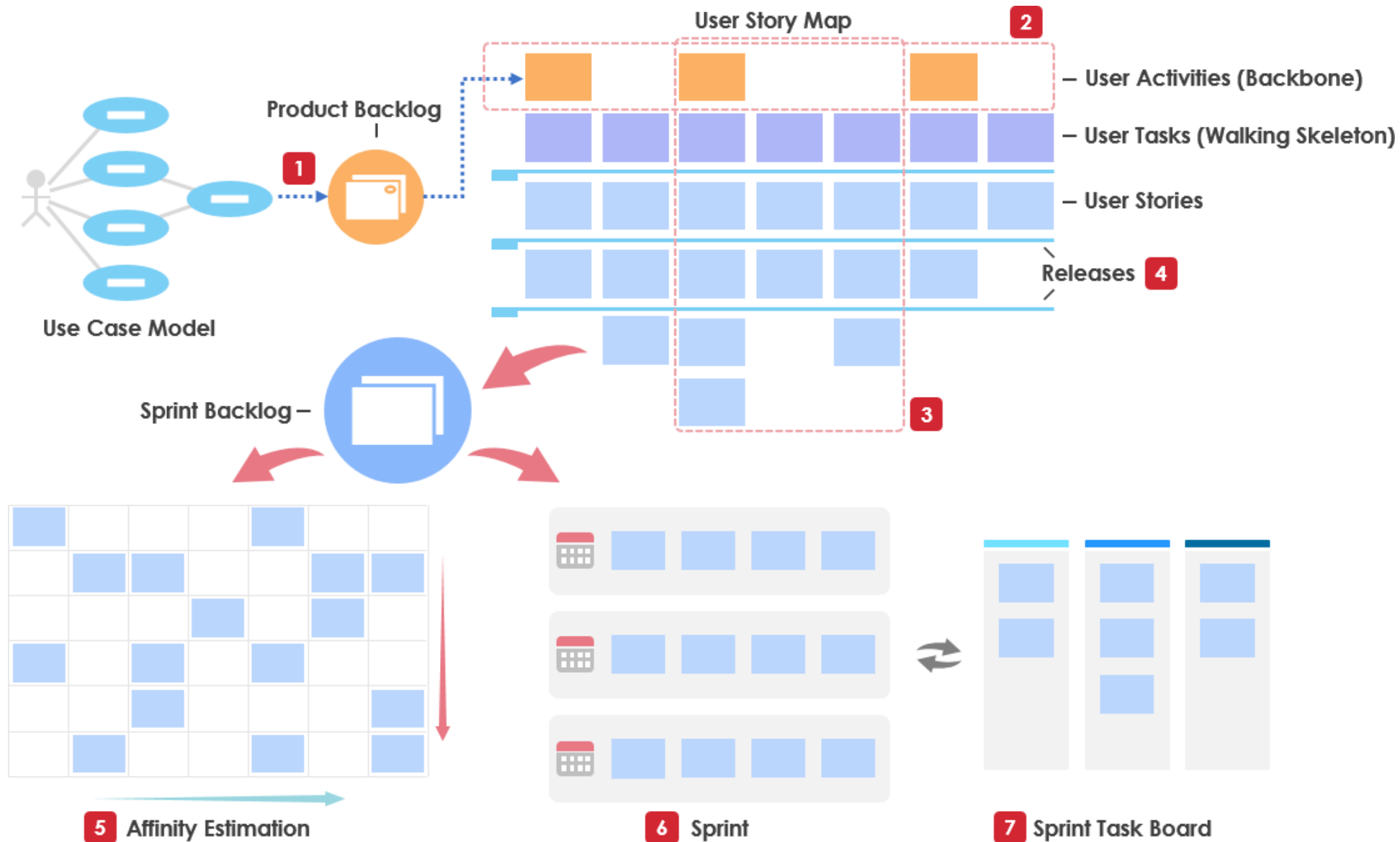
Usecase



Wireframe



end-to-end process



Before Implementation(2)

- 요구사항을 분석한다.
- AdobeXD, Sketch, Framer 등의 도구를 이용하여 Design Prototype, Usecase, Storyboard를 구성
- Data Flow, ERD를 작성
- API 기획회의
- API Design
- 기획 발표 PPT 작성

Sprint

- 앞서 분석한 기획안을 바탕으로 Sprint Backlog 작성(w/ github projects)
- Issue 관리와 Communication 동시 수행 가능

Daily

- 오전 15분 간 Daily Scrum 진행(어제 한 일, 오늘 한 일, Issue)
- 각자의 Task 진행