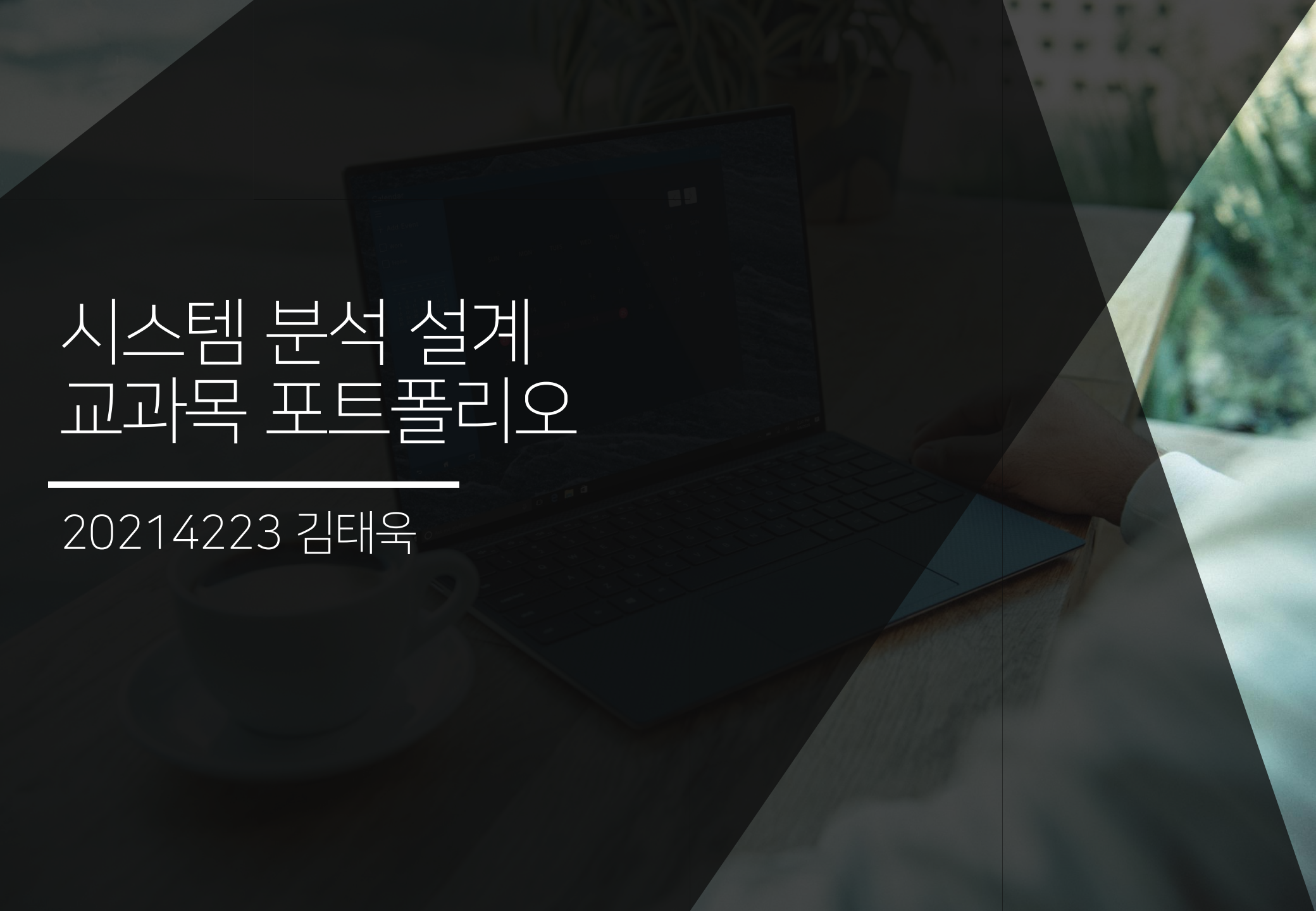
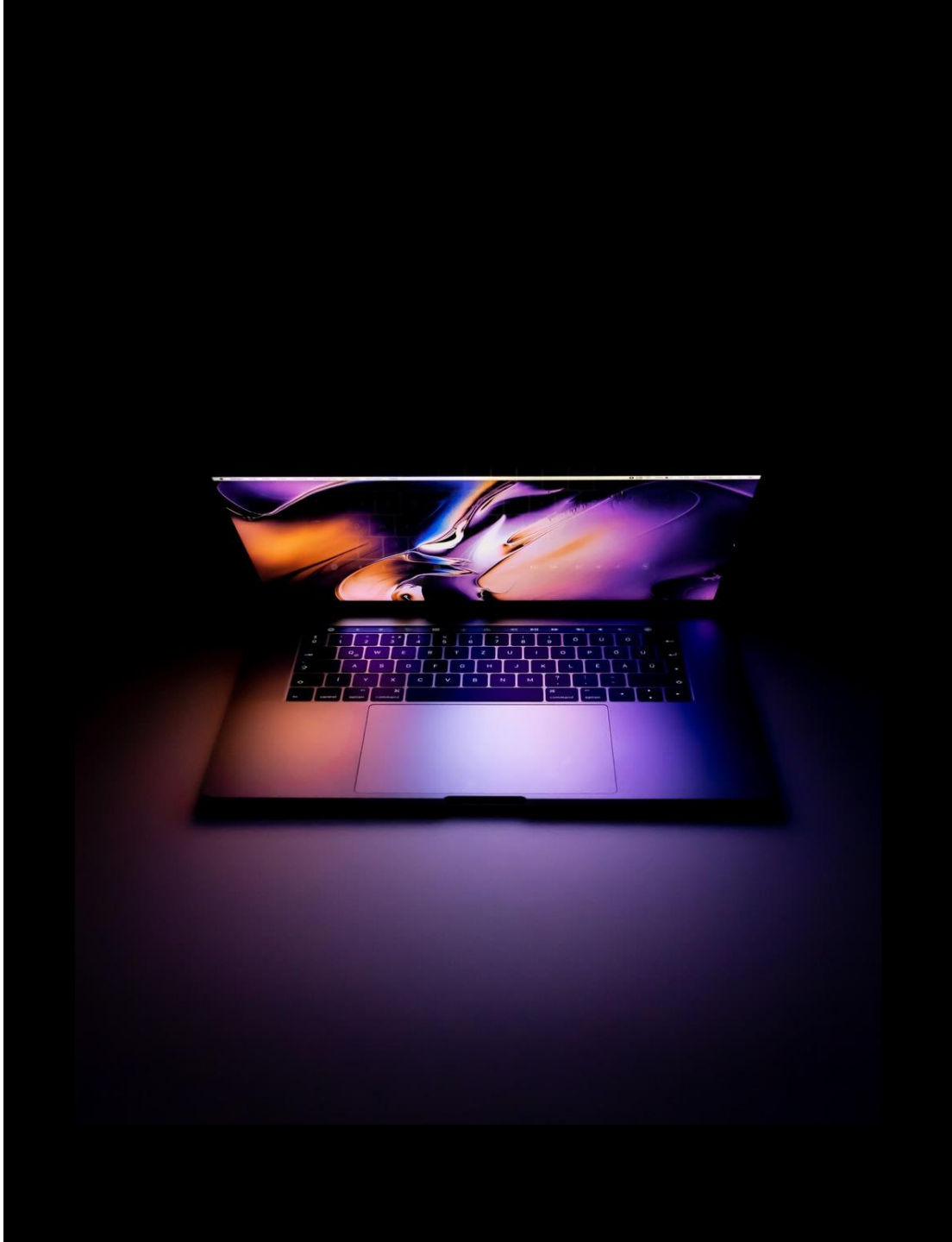


시스템 분석 설계
교과목 포트폴리오

20214223 김태욱

Contents

- 소프트웨어 공학
- 깃과 깃허브
- 안드로이드 스튜디오



1.

소프트웨어 공학

소프트웨어가 사용되는 곳

- 금융
- 노트북
- 스마트폰
  - 애플: IOS
    - Objective-C -> Swift로 바뀌는 추세
  - 안드로이드
    - JAVA
- 자동차
- 건물
- 항공기
- 의료

프로그램과 소프트웨어의 차이

프로그램

- 원시코드(C, JAVA, Python, …) 간의 상호작용

소프트웨어

- 원시코드
- 모든 산출물(자료구조, DB구조, 테스트 결과 등등)
- 각 단계마다 생산되는 문서
- 사용자 매뉴얼

→프로그램 뿐만 아니라 그 이상의 것도 포함하는 매우 포괄적인 개념

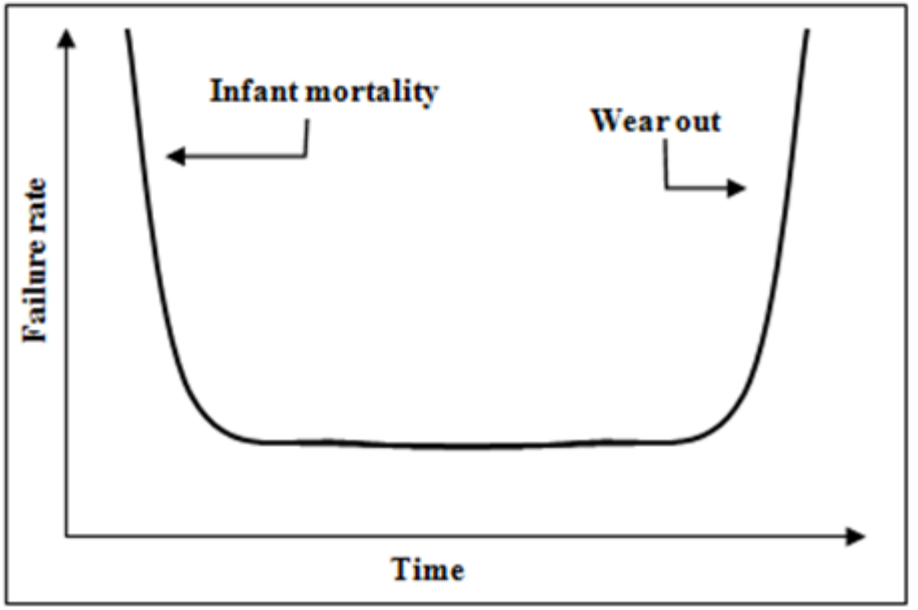
소프트웨어의 특징

- 제조가 아닌 개발

- 소모가 아닌 품질 저하
  - H/W: 오래 사용하면 부품이 닳고, 고장 발생 빈도↑, 기능↓
  - S/W: 오래 사용해도 닳지 않음, 고장 발생 빈도↓, 기능 동일

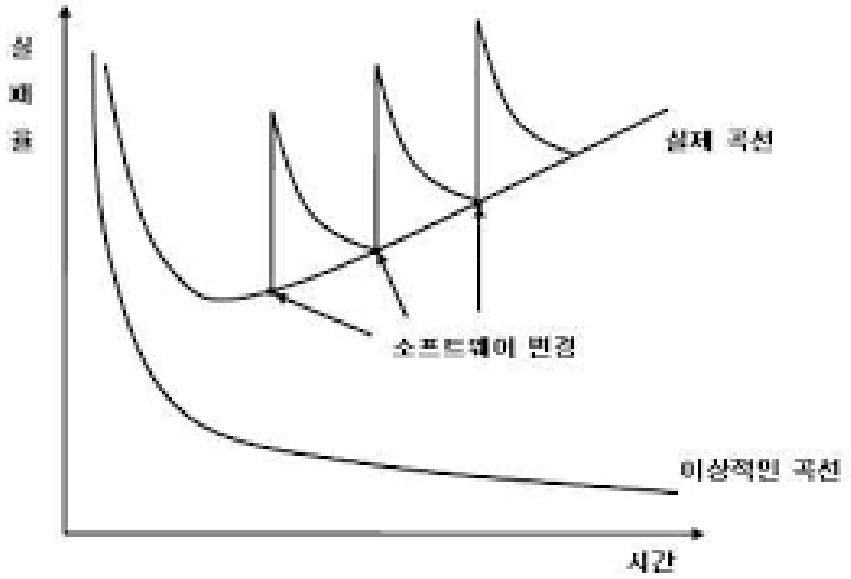
H/W의 실패 곡선

- 초기 실패율 높음
- 오류 해결(실패율 낮음)
- 오랜 기간 사용하며 문제 발생
- 다시 실패율 증가
- **실패율이 U자형 곡선을 그림**



S/W의 실패 곡선

- 발견되지 않은 오류로 실패율 높음
- 오류 해결
- 오랜 기간 사용(실패율 낮음)
- **실패율이 L자형 곡선을 그림**



실제 소프트웨어 실패 곡선

- 이상적인 곡선과 다르게 업데이트마다 실패율이 급격히 증가
- 오류를 해결
- 업데이트마다 순차적으로 반복됨

소프트웨어의 개발 속도

- 하드웨어보다 소프트웨어의 개발 속도가 느림
  - 하드웨어와 소프트웨어의 근본적인 개발 방법 차이 때문
  - H/W: 검증받은 부품을 조립하는 형태의 생산
  - S/W: 처음부터 만들어가는 개발 형태

  →해결 방안: **CBD 개발 방법론 (Component-Based Development)**

소프트웨어 공학

- 공학의 특성
  - 제약 사항: 정해진 기간, 주어진 비용

  →과학적 지식을 활용하여 문제를 해결하는데 한정된 기간과 비용의 제약을 받음

- 소프트웨어 공학

  →소프트웨어 개발 과정에 공학적인 원리를 적용하여 소프트웨어를 개발

소프트웨어 개발 생명주기 (SDLC)

1. 계획
2. 분석
3. 설계
4. 구현
5. 테스트
6. 유지보수

소프트웨어 개발 생명주기 (SDLC)

1. 계획
   - 개발 비용 산정
   - 일정 계획
   - 위험 관리
2. 분석
   - 기존 시스템의 문제점 파악 → 새로운 요구사항 도출 → 다이어그램 작성
   - 개발 방법론에 따른 표현 도구
     - 구조적 방법론: DFD(Data Flow Diagram), DD(Data Dictionary)
     - 정보공학 방법론: E-R 다이어그램(Entity-Relationship)
     - 객체지향 방법론(OOP)
   - 최종 산출물: **요구분석 명세서**

소프트웨어 개발 생명주기 (SDLC)

3. 설계
   - 설계 원리: 분할 정복, 추상화, 단계적 분해, 모듈화, 정보 은닉
   - 소프트웨어 아키텍처, 객체지향 설계
   - 아키텍처 스타일
   - GoF의 디자인 패턴
   - 모듈 평가 기준: 응집도와 결합도
4. 구현
   - 실질적인 코딩
   - 간략한 프로그래밍 언어의 역사
   - 표준 코딩 규칙
5. 테스트

소프트웨어 개발 생명주기 (SDLC)

6. 유지보수
   - 수정 유지보수
   - 적응 유지보수
   - 기능보강 유지보수
   - 예방 유지보수

프로젝트 관리와 지식 체계의 9가지 관점

프로젝트 관리

- 형상 관리: 문서 관리를 중심으로(중간 산출물을 통한 관리)

PMBOK(Project Management Body of Knowledge)

1. 프로젝트 **통합** 관리
2. 프로젝트 **범위** 관리
3. 프로젝트 **일정** 관리
4. 프로젝트 비용 관리
5. 프로젝트 품질 관리
6. 프로젝트 인적자원 관리
7. 프로젝트 **의사소통** 관리
8. 프로젝트 위험 관리
9. 프로젝트 조달 관리

프로세스

의미

- 일이 처리되는 과정 or 공정
- 주어진 일을 해결하기 위한 목적으로 수행되는 일련의 절차

목적

- 노하우 전달
- **가이드 역할**

프로세스 모델

정의

- S/W를 어떻게 개발할 것인가에 대한 전체적인 흐름을 체계화한 개념
- 순차적 단계로

목적

- 소프트웨어 개발의 전 과정을 하나의 프로세스로 정의
- 주어진 예산과 자원으로 개발, 관리하는 방법을 구체적으로 정의

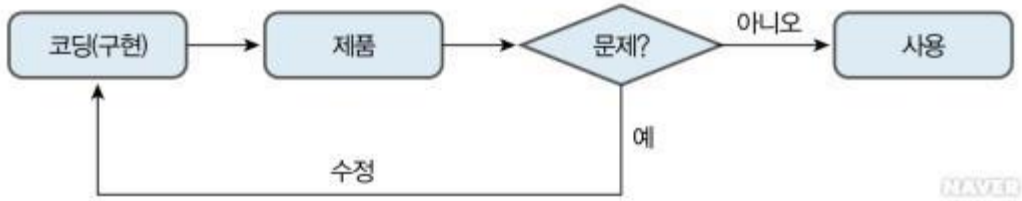
프로세스 모델의 종류

주먹구구식 모델

- 정확한 앞뒤 계산 없이 일을 대충 처리할 때
- 요구 분석 명세서나 설계 단계 없이 간단한 기능만 정리하여 개발하는 형태

단점

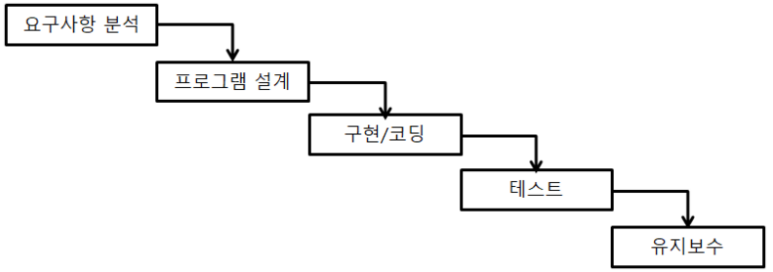
- 개발 순서나 단계별로 문서화된 산출물이 없어 유지 보수가 어려움
- 프로젝트 전체 범위를 알 수 없음, 좋은 아키텍처를 만들 수 없음



프로세스 모델의 종류

선형 순차적 모델 (폭포수 모델)

- 관리의 용이, 체계적인 문서화
- 요구사항의 변화가 적은 프로젝트에 적합



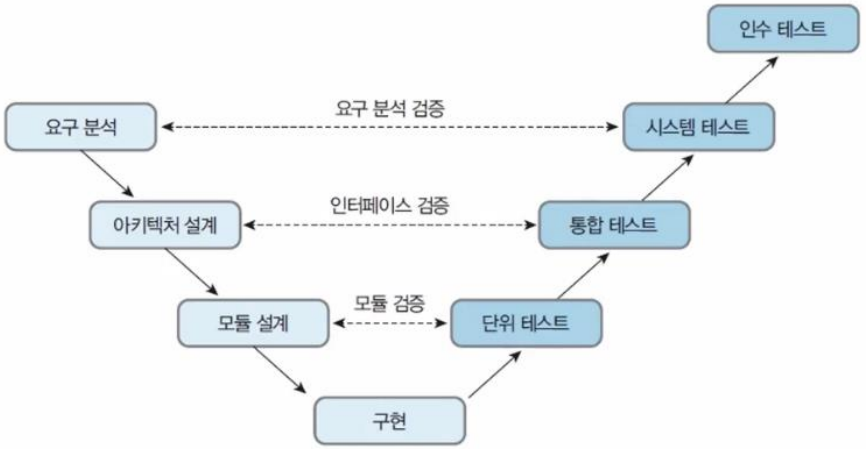
단점

- 각 단계는 앞 단계가 완료되어야 수행 가능
- 각 단계의 결과물이 완벽한 수준으로 작성되어야 다음 단계의 오류를 넘기지 않음
- 사용자가 중간에 가시적인 변화를 볼 수 없음

프로세스 모델의 종류

V모델

- 폭포수 + 테스트 단계 추가 확장
- 산출물 중심(폭포수) vs 각 개발 단계를 검증하는데 초점 (V모델)



프로토타입

- 대량 생산에 앞서 미리 제작해보는 원형 또는 시제품, 제작물의 모형

- 사용자의 요구를 받아 모형을 만들고 이를 통해 사용자와 의사소통 하는 도구로 활용

프로토타입 개발 생명주기

- 요구 사항 정의 및 분석 단계에서 개발 진행

1. 요구 수정
2. 빠른 설계
3. 프로토타입 개발
4. 고객 평가
5. 프로토타입 조정
6. 구현

- 이후 다음 프로세스 모델 진행 (설계…)

2.

깃과
깃허브

깃? 깃허브?

깃(Git)

- 분산 버전 제어 SW(Version Control System)
- 파일이 변경되는 것을 기록하여 버전 이력을 추적, 관리하는 시스템

깃허브(GitHub)

- 깃을 위한 웹 저장소
- 개발자들의 커뮤니티 협의(협업) 공간

→버전 관리를 위한 서버 저장소 및 프로젝트 개발을 위한 협업 관리 서비스

깃허브 용어

- Fork: 내 계정의 저장소에 다른 사람의 저장소를 복사하는 것
- Upstream: 깃허브의 모든 오픈 프로젝트 저장소
- Origin: 나의 저장소(깃허브의 내 계정에 있는 fork한 저장소도 포함)
- 로컬 저장소: PC에 복사한 지역 저장소
- Push: 로컬 저장소의 프로젝트를 오리진에 업데이트할 때

깃허브 용어

- Fetch(Pull이라고도 함): 오리진에서 로컬 저장소로 가지고 올 때
- Fetch & Merge: Upstream에서 origin으로 끌어올 때
- Clone: 주인이 내 것인 저장소를 로컬 저장소에 끌어올 때 (Fork와 차이점은 소유자가 다른 사람의 것)

오픈 소스 프로젝트 기여 과정

1. 프로젝트의 깃허브 저장소를 포크

2. 나의 PC에 복사(Clone)
   - 로컬 복사본에 변경 사항을 적용 또는 편집 뒤에 커밋하고 저장소애 push

3. 병합 요청
   - 오픈 소스에 병합할 때 Pull Request를 통해 요청을 보내고 오픈 소스 원작자가 검토를 끝낸 후 그에 따라 병합 여부를 결정

3.

안드로이드 스튜디오

DBMS

- 앱 구동 중 메모리에 저장된 데이터는 모두 사라지기에 다음 번 실행 시 필요한 데이터를 저장해야 함

- 안드로이드에서 지원하는 데이터 저장 방식
  1. DBMS
  2. SharedPreference
  3. File system
- SQLite란 안드로이드 앱에서 데이터를 저장할 때 사용하는 관계형 데이터베이스 시스템으로 간편하게 사용 가능함

데이터베이스 객체 생성

- 객체 생성 이벤트 때 DB 생성

```
@Override
public void onCreate(SQLiteDatabase db) { db.execSQL(DBContract.SQL_CREATE_TABLE); }
```

- 해당 질의어

```
static final String SQL_CREATE_TABLE = "CREATE TABLE IF NOT EXISTS " +
        TABLE_NAME + "(" +
        ROU_ID + " INTEGER NOT NULL PRIMARY KEY," +
        ROU_NAME + " TEXT NOT NULL, " +
        ROU_SET_COUNT + " TEXT NOT NULL, " +
        ROU_EXER_TIME + " TEXT NOT NULL, " +
        ROU_BREAK + " TEXT NOT NULL)";
```

- 데이터베이스 객체 생성

```
dbHelper = new DBHelper( context: this);
db = dbHelper.getReadableDatabase();
```

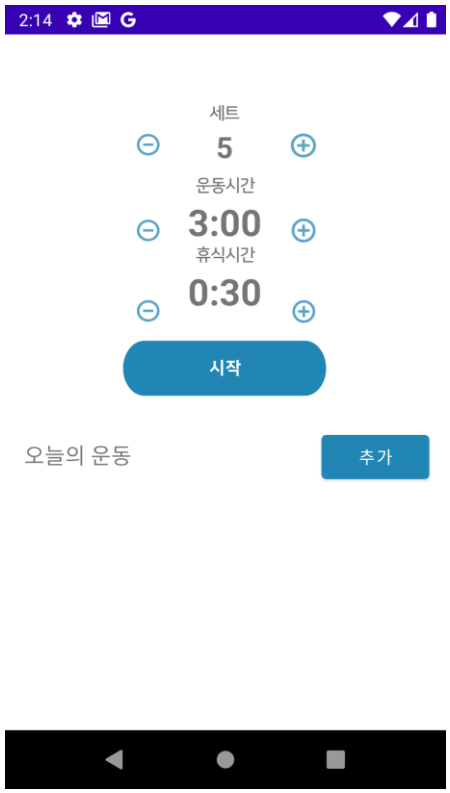
데이터베이스에 데이터 삽입

- 인텐트로 받아온 데이터를 ContentValues 객체에 담아 저장

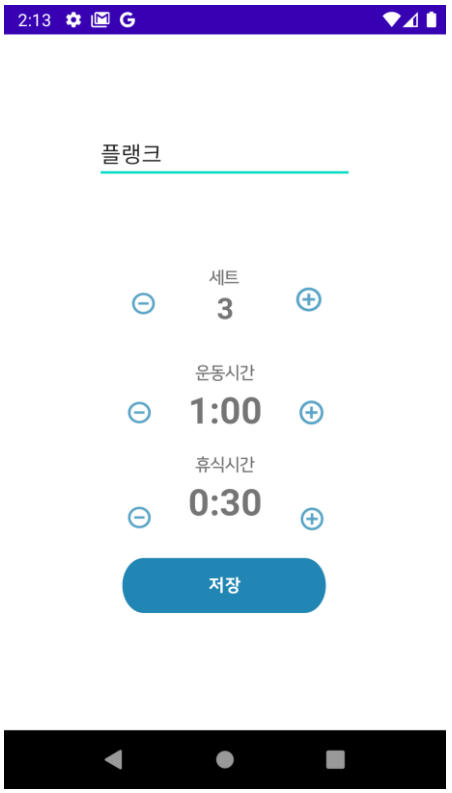
```
values.put("ID", String.valueOf(maxRoutineListCount));
values.put("NAME", intent.getStringExtra( name: "exerName"));
values.put("COUNT", intent.getStringExtra( name: "setCount"));
values.put("EXER", intent.getStringExtra( name: "exerTime"));
values.put("BREAK", intent.getStringExtra( name: "breakTime"));
routineListData.add(temp);

db = dbHelper.getWritableDatabase();
db.insert(DBContract.TABLE_NAME,  nullColumnHack: null, values);
simpleAdapter.notifyDataSetChanged();
```

실행 화면



첫 화면



아이템 추가 액티비티

결과 화면