

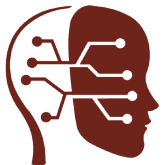
Software Engineering

Lecture 02: 소프트웨어 공학과 개발 프로세스 (Part 1)

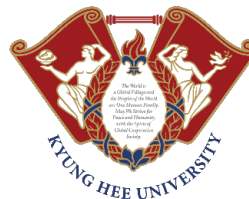
Professor: Jung Uk Kim

ju.kim@khu.ac.kr

Computer Science and Engineering, Kyung Hee University

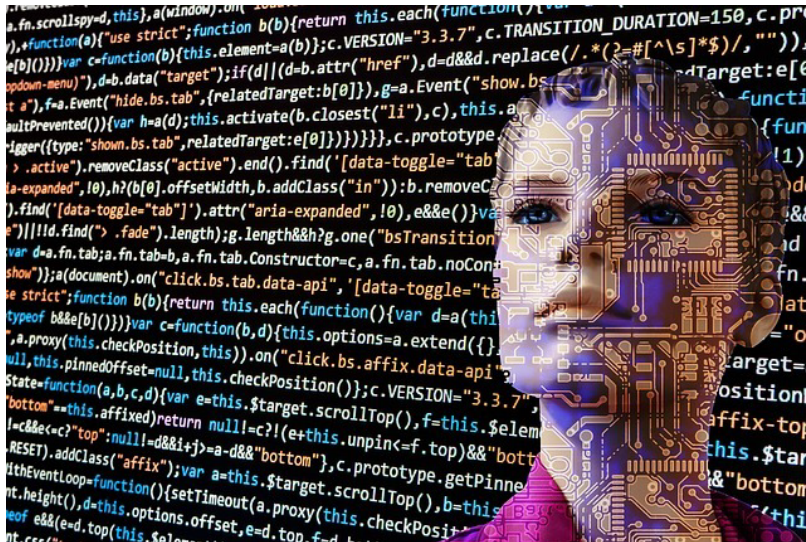


Visual AI Lab.



소프트웨어의 정의

- 소프트웨어 (Software)
 - **Soft (부드러운) + ware (제품)**
 - 컴퓨터를 비롯한 시스템에서 특정 작업을 수행하게 하는 **프로그램의 집합**으로 그 수행에 필요한 **절차, 규칙, 문서 등의 총칭**
 - 개발 과정에서 생성되는 모든 산출물(자료 구조, 데이터베이스 구조, 테스트 결과 등)과 각 단계에서 만들어지는 문서와 사용자 매뉴얼 등을 포함



※ 프로그램: 프로그래밍한 원시 코드 (Source Code)

소프트웨어의 특징

- 소프트웨어의 특징
 - 비제조성
 - 소모되지 않는 소프트웨어
 - 사용자의 요구에 따라 새롭게 생산됨
 - 변경성 (Changeability)
 - 복제 가능 (Duplicability)

소프트웨어의 특징 (하드웨어와의 비교)

- 제조가 아닌 개발

- 제조: 정해진 틀에 맞춰 일정하게 제품을 생산하는 것 (결과물의 차이는 크지 않음)
- 개발: 개인 능력에 따라 차이가 큼 (e.g., 경력 1달의 프로그래머 vs 경력 10년의 프로그래머)



제조

VS

초보

```
#include <stdio.h>

int main()
{
    printf("\n");
    printf(" * \n");
    printf(" * * \n");
    printf(" * * * \n");
    printf(" * * * * \n");

    return 0;
}
```

고수

```
#include <stdio.h>

int main()
{
    int i,j;
    for (i=0;i<5;i++)
    {
        for (j=0;j<=i;j++)
        {
            printf(" * ");
        }
        printf("\n");
    }
    return 0;
}
```

개발

소프트웨어의 특징 (하드웨어와의 비교)

- 소모가 아닌 품질 저하

- 하드웨어 부품: 먼지, 진동, 열, 마모 등으로 고장이 날 수 있음

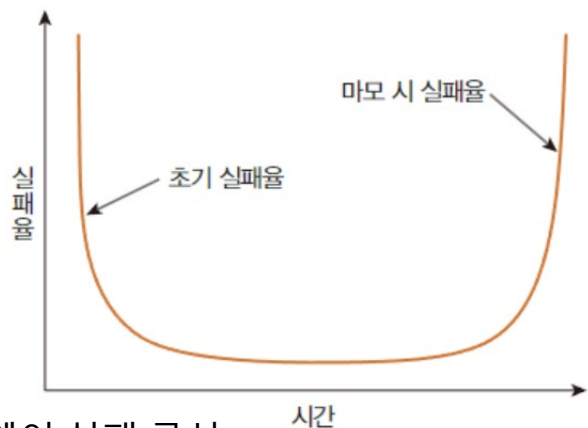
- 실패 곡선: 욕조 곡선(Bathtub Curve)

- 초기 실패율이 높지만, 오류를 해결하면 문제없이 오래 지속
- 오래 사용하면 주변 환경의 문제(먼지, 마모 등)로 실패율이 치솟음

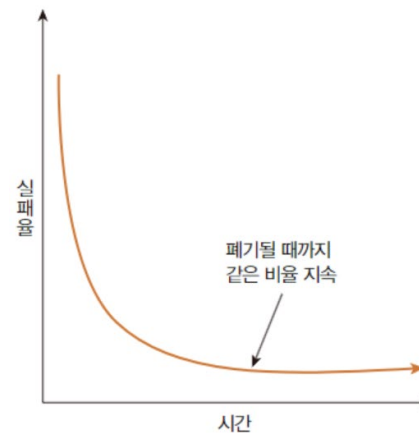
- 소프트웨어: 하드웨어와 달리 닳지 않음

- 실패 곡선

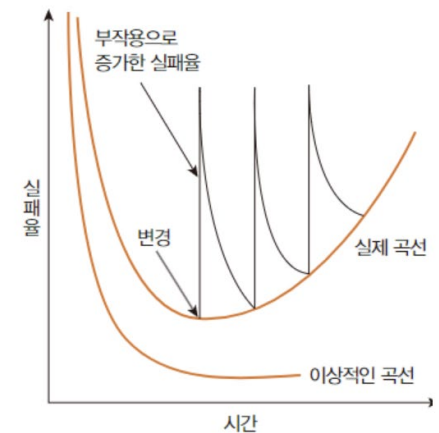
- 이상) 오류를 해결한 후 문제없이 사용 (변경 사항도 없고, 환경 변화도 없을 때)
- 실제) 사용자의 요구가 계속 발생함 → 변경으로 인한 부작용으로 **실패율이 급격히 증가할 수 있음**



하드웨어 실패 곡선



(a) 이상적인 소프트웨어의 실패 곡선



(b) 실제 소프트웨어의 실패 곡선

소프트웨어 개발의 어려움

- 수준에 따른 구분
 - 반려동물 집 짓기
 - 연장과 나무만 있으면 빠른 시간에 지을 수 있음
 - 도면을 따로 그리지 않고 머릿속으로 구상가능
 - 단독주택 짓기
 - 여러 사람이 참여해야 하며, 설계와 공정이 필요
 - 전문적인 도구들이 필요
 - 대형 빌딩 짓기
 - 지반이 안전한지 확인 후 토목공사 (훨씬 복잡)
 - 다양한 전문가 참여, 의견 충돌 가능성 (참여인력 많음)
 - 긴 개발 기간



소프트웨어 개발의 어려움

- 대규모 소프트웨어 개발의 어려움
 - 대형 빌딩 짓기와 유사

개발 과정이 복잡하다

무엇이든지 복잡하면 문제가 많이 발생할 수 있는데 소프트웨어 개발도 예외가 아니다.



개발의 복잡함을 줄이기 위한 방법과 기술 필요

참여 인력이 많다

인력이 많으면 의사소통 경로가 많아져 의사결정 과정도 복잡하고 협력하기도 쉽지 않다. 그리고 중간에 이직하는 사람, 새로 투입되는 사람 등 변화도 많이 발생한다.



개발에 참여하는 팀을 구성하고 관리하는 효율적인 방법 필요

개발 기간이 길다

개발 기간이 길어질수록 프로젝트 진행 상황을 파악하기 쉽지 않고 개발 비용 산정도 어렵다.



프로젝트를 효율적으로 관리하기 위한 체계 필요

소프트웨어 개발의 어려움

- 대규모 소프트웨어 개발의 어려움

실패한 제품을 통해 본 제품 개발



고객이 설명한
제품



프로젝트 리더가
이해한 제품



애널리스트가
디자인 한 제품



실제 도입된 제품



문서화된 제품



컨설턴트가
설명한 제품



프로그래머가
작성한 코드



베타 테스터가
받은 제품



마케팅 부서가
광고하는 제품



고객이 실제로
필요로 하는 제품

소프트웨어 공학의 등장배경

• 소프트웨어의 위기 (Software Crisis)

- **소프트웨어 개발 속도가 하드웨어 개발 속도를 따라가지 못해** 사용자 요구 사항을 감당할 수 없는 문제가 발생

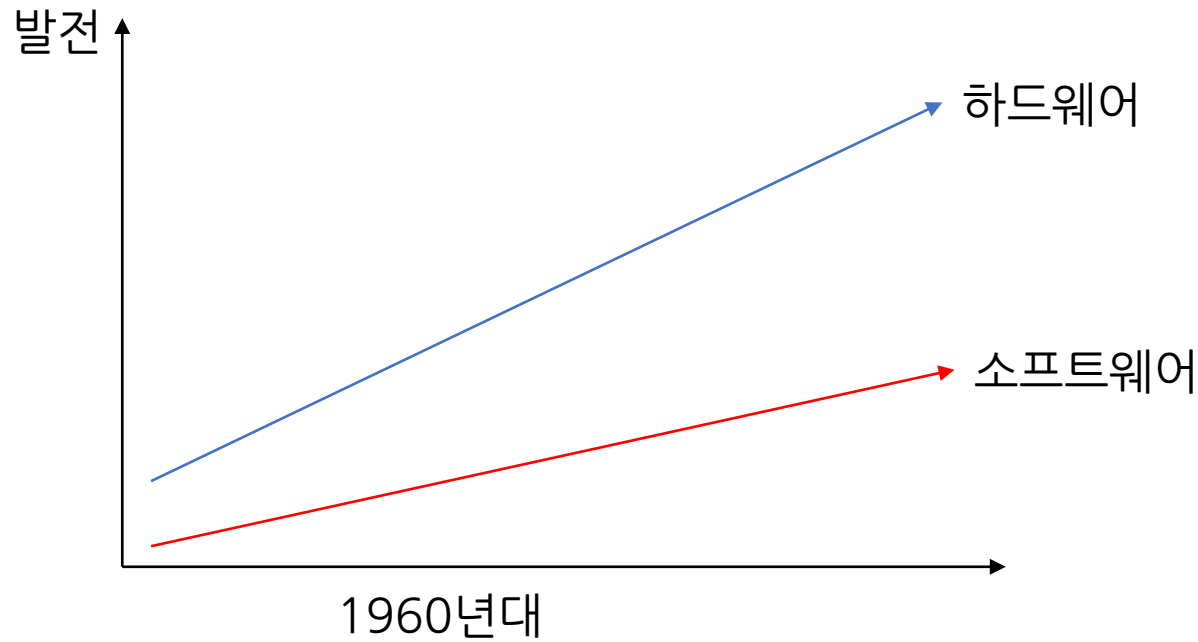
- 예산 초과, 일정 지연
- 비효율적이고 품질이 낮은 소프트웨어
- 고객의 요구사항을 만족 못함
- 관리 불가능



체계적이지 못한 상황
→ 쉽지 않은 대규모 소프트웨어 개발



공학의 개념을 소프트웨어에 주입!



소프트웨어 공학 (Software Engineering)

• 공학

- **공업적인 생산에 응용하여 생산력과 생산품의 성능을 향상·발전시키기 위한 과학 기술의 체계적인 학문**
- **인류의 이익을 위해 과학적 원리, 지식, 도구 등을 활용하여 새로운 제품, 도구 등을 만드는 것**

• 소프트웨어 공학

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



소프트웨어 개발



소프트웨어 운용



소프트웨어 유지보수

소프트웨어 공학의 필요성

- 소프트웨어공학은 **소프트웨어** 있는 심각한 직접적인 손해 또는 간접적인 손해가 따를 수 있는 문제를 해결
- 소프트웨어 제품은 **고객의 문제를 해결하기 위해 구축**, 비즈니스를 운영하기 위하여 사용
- 소프트웨어가 제대로 작동하지 않으면 **재정적 손실이 크고 사용자가 불편을 겪음**

소프트웨어 엔지니어의 역할

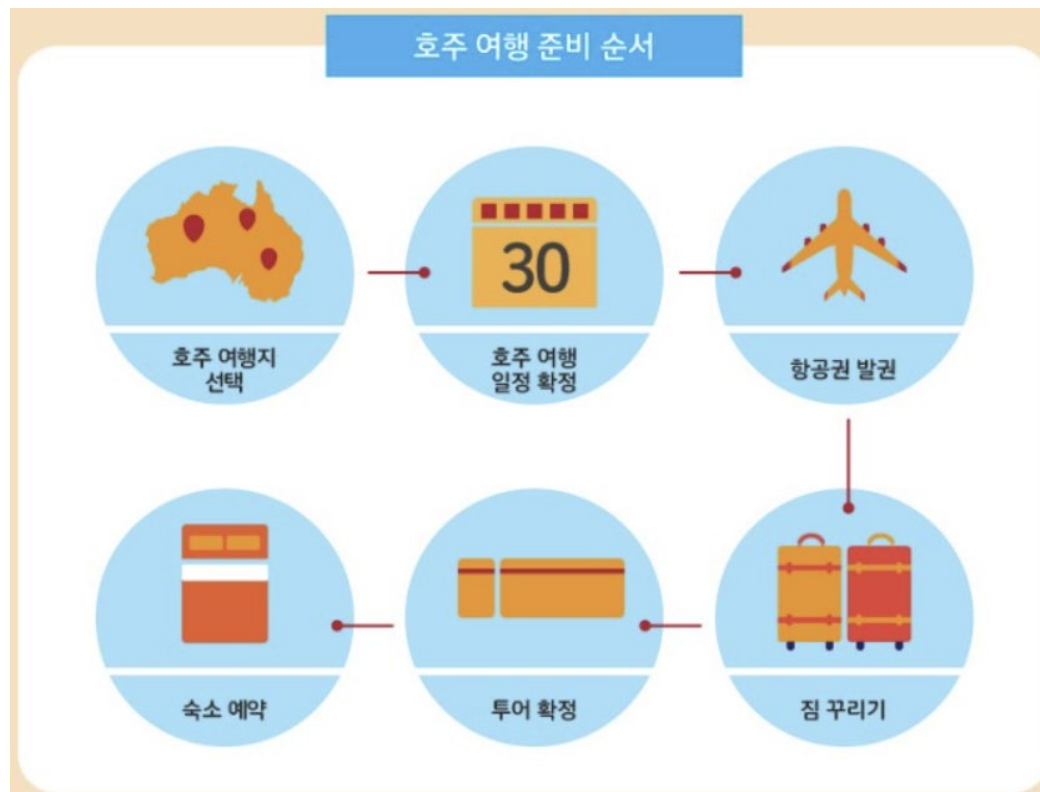
- 역할

- 문제의 분석, 정확한 요구 사항 도출
- 소프트웨어 구조, 입출력, 인터페이스 설계
- 알고리즘, 데이터 구조, 데이터베이스 설계
- 양질의 프로그램을 생산
- 테스트 및 통합 기술
- 사용자 지침서 및 개발 문서 작성
- 성능 측정과 모델 설계
- 유지보수
- 프로젝트 계획 및 관리

소프트웨어 개발 프로세스

- 프로세스 (Process)

- 일을 해결하기 위한 목적으로 그 **순서가 정해져 수행되는 일련의 절차**
 - (예: 공장에서 자동차가 조립되어 완제품이 되는 과정)
- 어떤 일을 해결할 때, 해당 프로세스만 잘 따르면 목적을 달성할 수 있음



소프트웨어 개발 프로세스

- 소프트웨어 개발 프로세스 모델



반려동물 집 짓기

- 혼자 지을 수 있음
- 설계도면이 필요하지 않음
- 균형이 맞지 않아도 큰 문제는 없음



규모가 작은 소프트웨어
(소규모 프로젝트)



고층빌딩 짓기

- 많은 사람이 참여해야 하고 기간도 오래 걸림
- 설계도면이 꼭 필요함
- 균형이 맞지 않으면 안전상 큰 문제 발생



규모가 큰 소프트웨어
(대규모 프로젝트)

소프트웨어 개발 생명주기

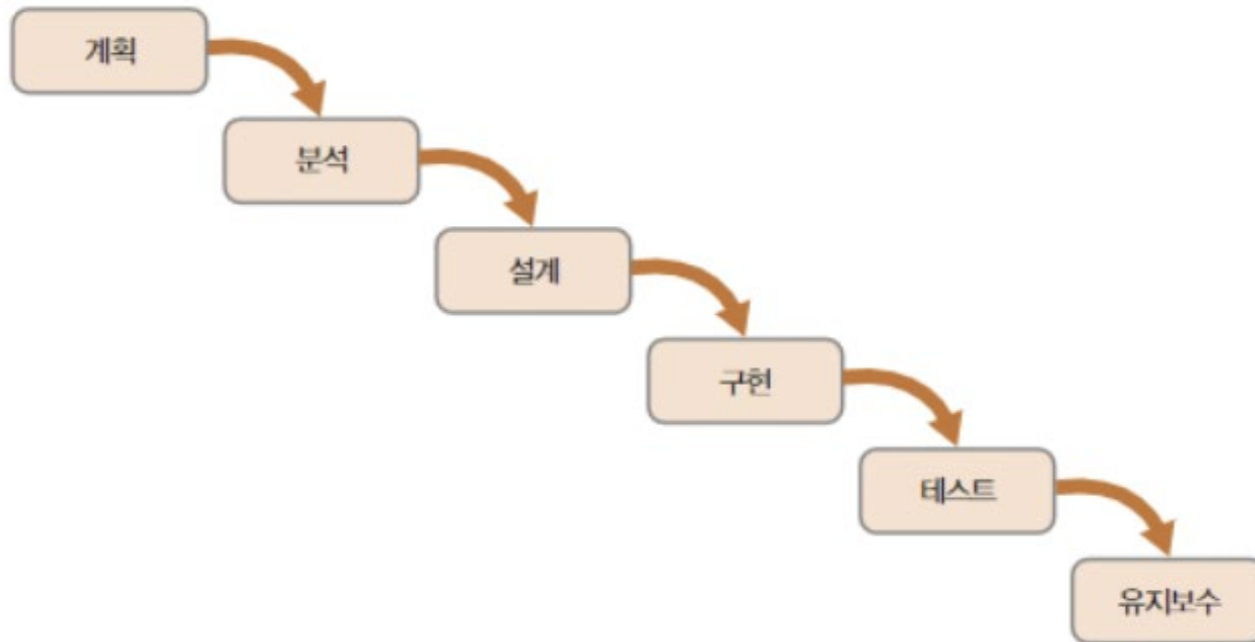
- **소프트웨어 공학에서의 소프트웨어 개발 프로세스의 의미**
 - 소프트웨어 공학

품질 좋은 소프트웨어를 경제적으로 개발하기 위해 **계획을 세우고, 개발하며, 유지 및 관리**하는 전 과정에서 **공학을 적용**해 필요한 이론과 기술 및 도구들에 관해 연구하는 학문

- **소프트웨어 공학 소프트웨어 개발 과정에서 생산성을 높이고, 고품질의 소프트웨어를 생산해 사용자를 만족시키는 것**
- 소프트웨어공학 관점에서의 **소프트웨어 개발 프로세스**
→ **소프트웨어 개발 생명주기 (Software Development Life Cycle)**

소프트웨어 개발 생명주기 (SDLC)

- 소프트웨어 개발 생명주기 (**Software Development Life Cycle**)
 - 하나의 제품인 소프트웨어를 만들기 위해 **계획 단계에서 유지보수 단계에 이르기까지 일어나는 일련의 과정**(계획,분석,설계,구현,테스트,유지보수)



소프트웨어 개발 생명주기

- 소프트웨어 개발 생명주기 6단계 (예시: PC 게임)
 - 계획 (1단계)
 - 비용, 기간 등 프로젝트를 수행하는데 필요한 것에 대해 계획



비용 - 예산
(예: 50억)



기간 - 출시일
(예: 2025. 01.01)



위험분석
(예: 기존 게임과의 차별성)



문제정의 (목표 및 제약조건)
(예: 동시접속자 수 10만명)

소프트웨어 개발 생명주기

- 소프트웨어 개발 생명주기 6단계 (예시: PC 게임)

- 분석(요구분석) (2단계)**

- 개발할 소프트웨어의 **기능, 제약조건, 목표** 등을 소프트웨어 사용자와 함께 **명확히 정의하는 단계**

맵 종류

사막

산

눈

도시

제약 조건 - 맵 크기, 참여인원, 자기장(활동범위), ...

목표 - 한판당 게임시간 30분 이내,
- 소지할 수 있는 아이템 용량 제한

자기장 대기 시간 (단위: 초)

페이지	1	2	3	4	5	6	7	8	9
변경 전	120*+300	200	150	120	120	90	90	60	180*+15
변경 후	90*+300	90	60	60	60	60	60	60	80*+15

· 안전 구역 비활성 시간

자기장 이동 속도 (단위: 미터/초)

페이지	1	2	3	4	5	6	7	8	9
변경 전	19	11.4	16.4	12.3	9.2	6.2	3.1	1.5	3.1
변경 후	21	13.7	9.4	5.2	2.8	2.1	1.7	0.9	1.2

자기장 대미지 (단위: 대미지/픽)

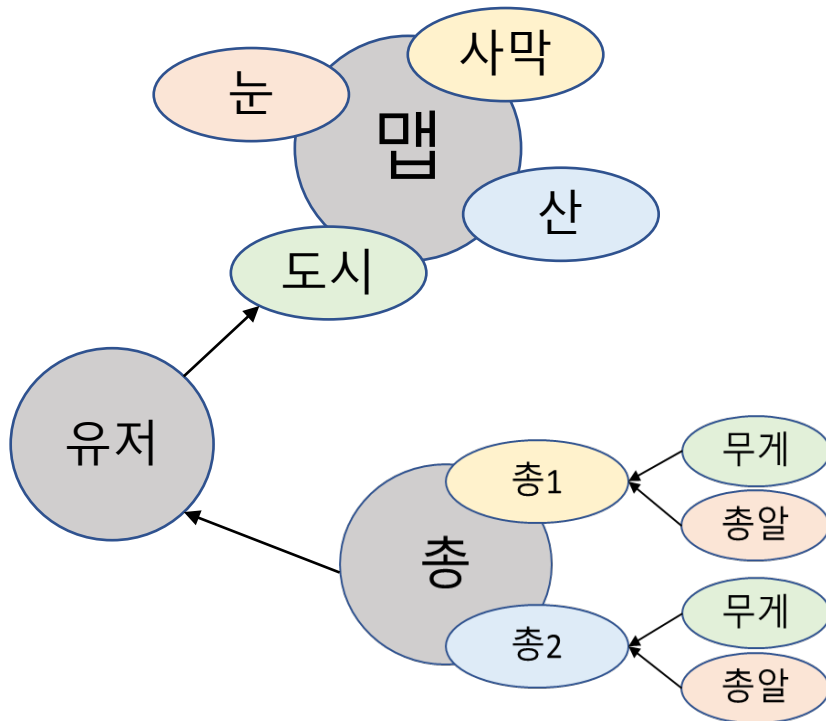
페이지	1	2	3	4	5	6	7	8	9
변경 전	0.4	0.6	0.8	1	3	5	7	9	11
변경 후	0.4	0.6	0.8	1	3	5	8	11	14






소프트웨어 개발 생명주기

- 소프트웨어 개발 생명주기 6단계 (예시: PC 게임)

- 설계 (3단계)**

- 분석 단계에서 표현한 다이어그램을 가지고 코딩할 수 있는 수준으로 환경에 밀접하게 구체화
- 소프트웨어의 구조, 알고리즘을 작성하는 단계



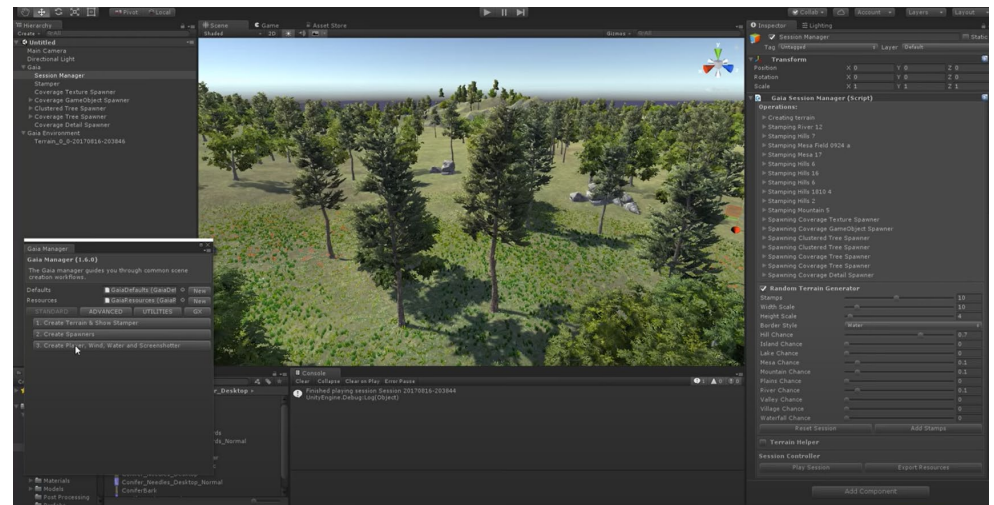
				
AKM	M416	M16A4	SCAR-L	그로자
48 피해당	41 피해당	41 피해당	41 피해당	48 피해당
715 탄속	880 탄속	900 ↑ 탄속	870 탄속	715 탄속
10,000 지지력	3,500 ↓ 지지력	8,000 지지력	9,000 지지력	10,000 지지력
0.100 초 ↑ 연사 속도	0.086 초 연사 속도	0.075 초 ↓ 연사 속도	0.096 초 연사 속도	0.080 초 연사 속도

소프트웨어 개발 생명주기

- 소프트웨어 개발 생명주기 6단계 (예시: PC 게임)

- 구현 (4단계)**

- 프로그래밍 언어를 사용하여 실제로 프로그램을 작성하는 단계
- 프로그래밍 언어를 선택하여야 하고, 프로그래밍 기법과 스타일, 프로그래밍 순서를 결정함



Unity 예시

소프트웨어 개발 생명주기

- 소프트웨어 개발 생명주기 6단계 (예시: PC 게임)

- 테스트 (5단계)**

- 개발한 시스템이 요구사항을 만족하는지, 실행 결과가 예상 결과와 맞는지, 숨어있는 오류는 없는지 찾아내 **완성도를 높이는 단계**



요구사항 체크

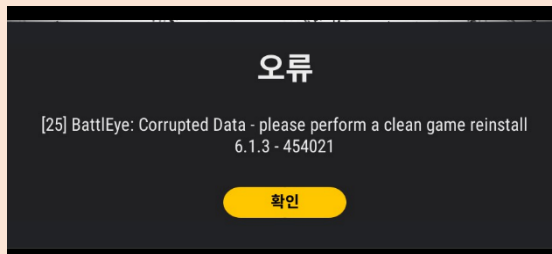
- (1) 개발 비용 체크
- (2) 출시일 체크
- (3) 위험분석 체크
- (4) 목표 체크



실행 결과 체크

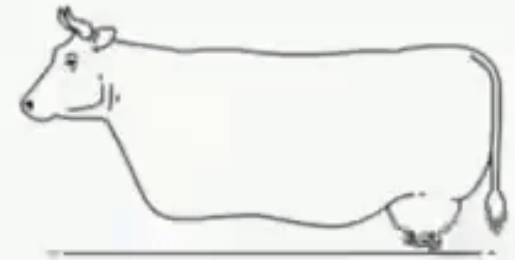
안전한 착지 (예상결과)

→ 비정상적인 착지 (실제)



오류 체크

1st rule of programming: If it works don't touch it. 🐮



일단 돌아간다면 건드리지 마라



좋은 소프트웨어 개발을 위해서는 건드려야 함!

소프트웨어 개발 생명주기

- 소프트웨어 개발 생명주기 6단계 (예시: PC 게임)
 - **유지보수 (6단계)**
 - 시스템이 인수되고, 배포가 이루어진 후 일어나는 모든 활동
 - 모든 활동: 결함을 수정하거나 성능 개선, 소프트웨어 제품 수정(v1.0 → v2.0)
 - 소프트웨어 개발 생명주기 중 **가장 긴 기간**



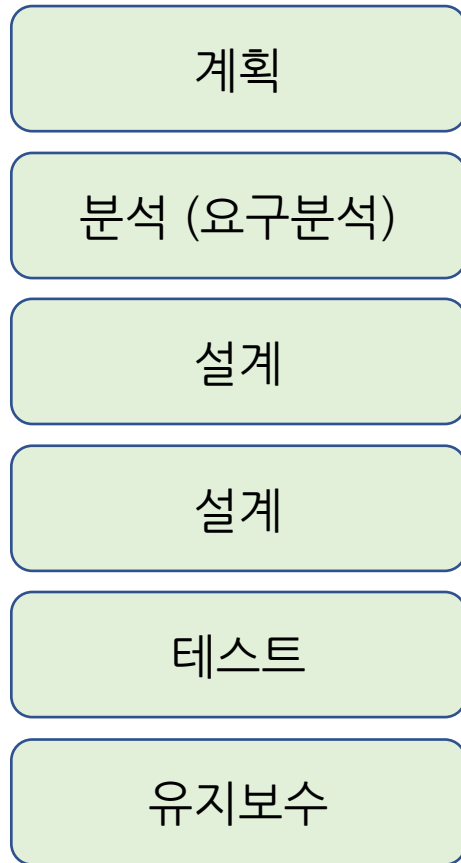
소프트웨어 개발 생명주기 모델

- 소프트웨어 개발 생명주기 모델
 - 소프트웨어 개발 생명주기 모델의 정의
 - 소프트웨어를 **어떻게 개발할 것인가에 대한 전체 흐름을 체계화** 한 개념
 - 각 단계에서 무엇을 **어떤 순서로 작업**하며, **어떤 자원을 사용할 것인지에 대한 지침**
 - 소프트웨어 개발 생명주기 모델의 목적
 - **고품질의 소프트웨어 제품**을 만드는 것
 - 소프트웨어 개발 생명주기 모델의 역할
 - 프로젝트에 대한 **전체적인 기본 골격**을 세워 줌
 - 일정 계획을 수립할 수 있고, **개발 비용 산정 등을 분배**할 수 있음
 - 각 단계별로 생성되는 **문서를 포함한 산출물을 활용**해 검토할 수 있게 해 줌

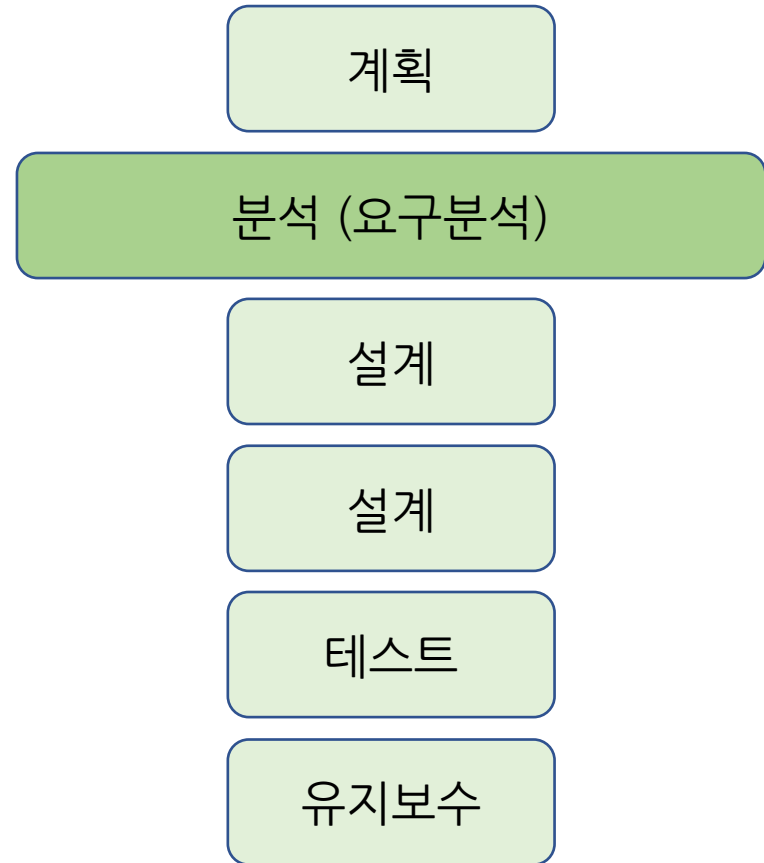
소프트웨어 개발 생명주기 모델

- 특징

- 세부적인 생명주기 모델(SDLC)은 소프트웨어 개발 팀마다 다름



모든 단계를 동일한 중요도로 간주
(e.g., 폭포수 모델)



특정 단계를 높은 중요도로 간주
(e.g., 프로토타입 모델)

소프트웨어 개발 생명주기 모델

- **주먹 구구식 모델**
- **선형 순차적 모델**
 - 폭포수 모델
 - V 모델
- **진화적 프로세스 모델**
 - 프로토타입 모델
 - 나선형 모델
- **단계적 개발 모델**
 - 점증적 개발 모델
 - 반복적 개발 모델: 통합 프로세스 모델
- **애자일 프로세스 모델**

Questions?