# 2. 소프트웨어 프로세스와 생명주기

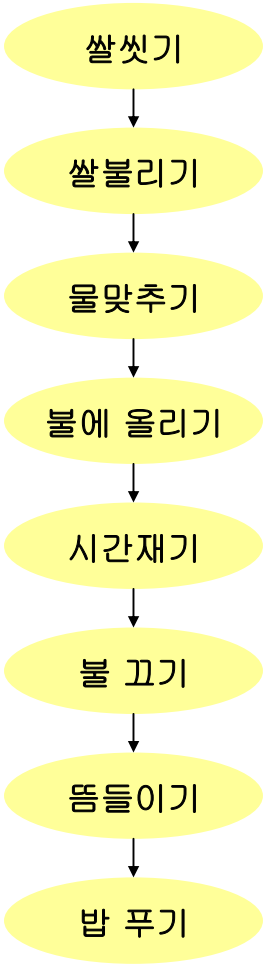
# 주요 내용

- 프로세스란?
- 소프트웨어 프로세스란?
- 소프트웨어 생명주기란?
- 소프트웨어 생명주기 모델의 종류

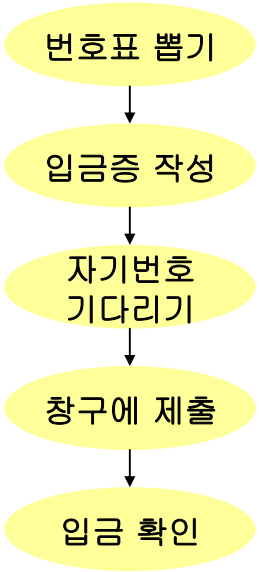
# 목차

- 강의 내용
  - 프로세스
  - 소프트웨어 프로세스
  - 소프트웨어 생명주기
  - 소프트웨어 생명주기 모델의 종류
- 팀 프로젝트(3주차)
  - 팀 구성 확정

# 일상 생활에서의 프로세스




재래식 밥 짓기 프로세스




은행 입금 프로세스

# 프로세스

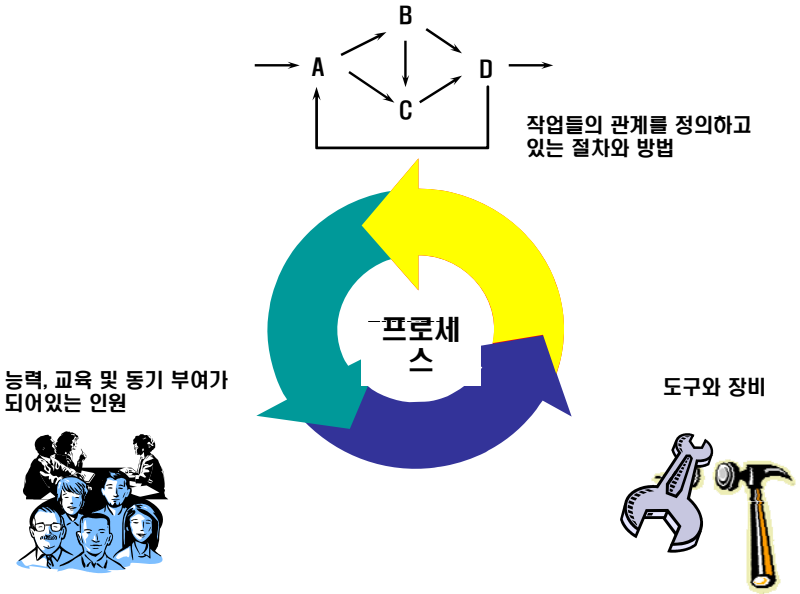
# 프로세스란?

## ❖ 의미

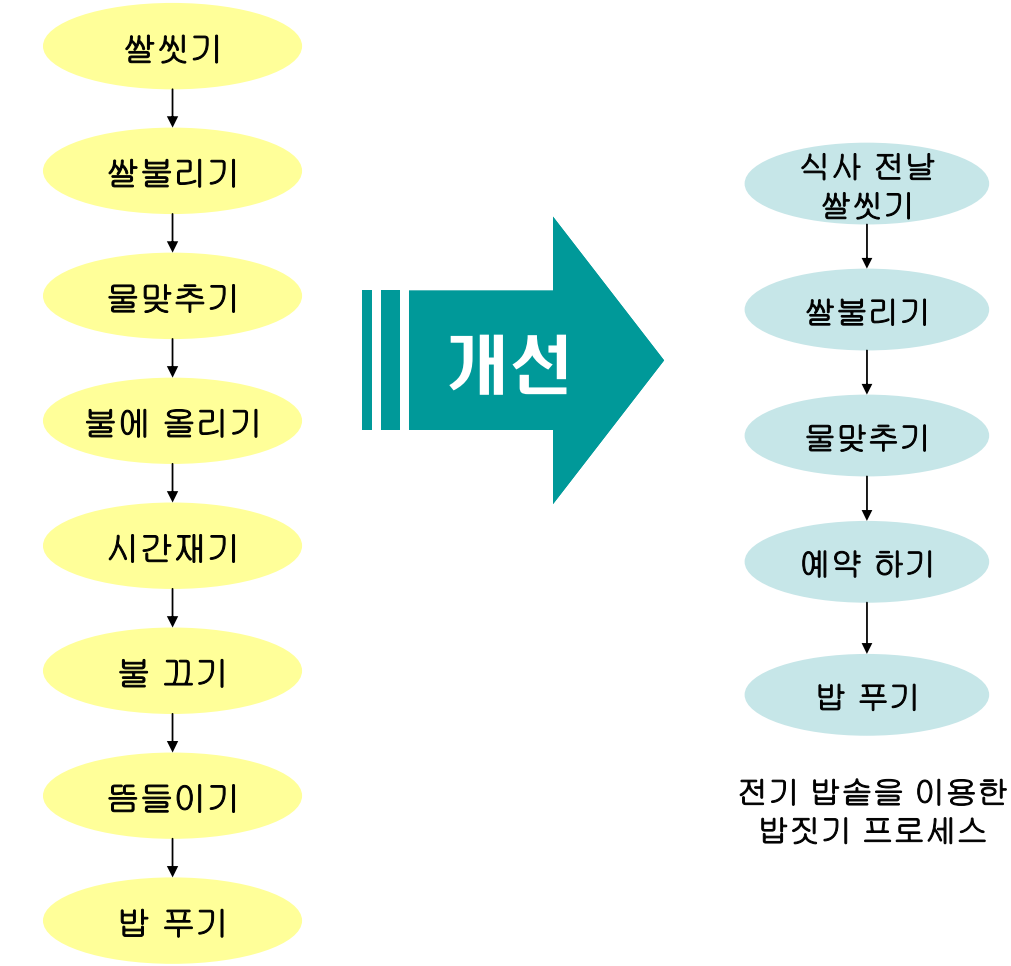
- 주어진 목적을 위해 수행되는 일련의 절차

## ❖ 역할

- 절차, 인력, 기술을 통합
- 각 순서와 활동이 명확하게 정의됨
  - 프로세스를 사용하는 조직원들의 공통된 행동 양식을 지정해주는 역할



# IT를 활용한 프로세스 개선 사례



# 소프트웨어 개발 프로세스

# 소프트웨어 개발 프로세스의 중요성

## ❖ 소프트웨어 개발의 목표

- 정해진 기한 내에, 주어진 예산을 이용해 사용자가 원하는 좋은 품질로 개발하는 것

## ❖ 계속되는 프로젝트 실패

- 소프트웨어의 요구사항이 복잡해지고 규모가 점점 커짐
- 정해진 기간 내에 고품질의 소프트웨어를 개발하는 것이 점점 더 어려워짐

## ❖ 소프트웨어 개발 프로세스의 중요성

- 소프트웨어 제품의 품질은 그 제품을 만들기 위해 사용된 프로세스의 품질에 의해 결정된다 [Watts S. Humphrey]

# 소프트웨어 프로세스

## ❖ 정의

- 소프트웨어 개발에 필요한 절차만이 아니라, 그와 관련된 인력, 방법, 도구 들이 통합되는 수단
- 소프트웨어와 이에 관련된 산출물을 개발, 유지하기 위해 사용하는 활동, 방법, 절차의 집합

| 자료원 | 소프트웨어 프로세스 정의 |
|---|---|
| IEEE-STD-610 | 주어진 목적을 달성하기 위한 순서적인 절차 틀 |
| Olson et al.(1989) | 특정한 목표나 목적을 달성하기 위한 활동, 작업 및 절차들의 집합. |
| SEI CMM<br>(Humphrey, 1989:<br>Paulk et al., 1993) | 소프트웨어의 생산 및 진화에 사용되는 활동, 방법 및<br>실무 활동 들의 집합<br>인력, 절차, 방법, 장치 및 도구들이 원하는 산출물을 생산할 수 있도록 통합하는 수단 |

# 소프트웨어 개발 생명주기

# 소프트웨어 개발 생명주기 (Software Development Life Cycle)

## ❖ 의미

- 소프트웨어를 어떻게 개발할 것인가에 대한 추상적 표현
- 순차적 또는 병렬적 단계로 구성됨
- 개발 모델 또는 소프트웨어 공학 패러다임이라고도 함

## ❖ 특징

- 개발 생명주기의 각 단계에 관련된 활동들이 정의되어 있음
- 단계별 활동들을 통해 다음 단계에 활용될 수 있는 산출물이 작성됨
- 전체 프로젝트의 비용 산정과 개발 계획을 수립할 수 있는 기본 골격 제시
- 참여자들 간에 의사소통의 기준과 용어의 표준화를 가능하게 함
- 문서화가 충실한 프로젝트 관리를 가능하게 함

# 소프트웨어 개발 생명주기 모델의 종류

- 주먹구구식 개발 모델(Build-Fix Model)
- 폭포수 모델(Waterfall Model)
- 원형 모델(Prototyping Model)
- 나선형 모델(Spiral Model)
- UP(Unified Process)
- XP(eXtreme Programming)

# 주먹구구식 개발 모델(Build-Fix Model) (1/2)

## ❖ 개요

- 요구사항 분석, 설계 단계 없이 일단 개발에 들어간 후 만족할 때까지 수정작업 수행

## ❖ 적용 가능한 경우

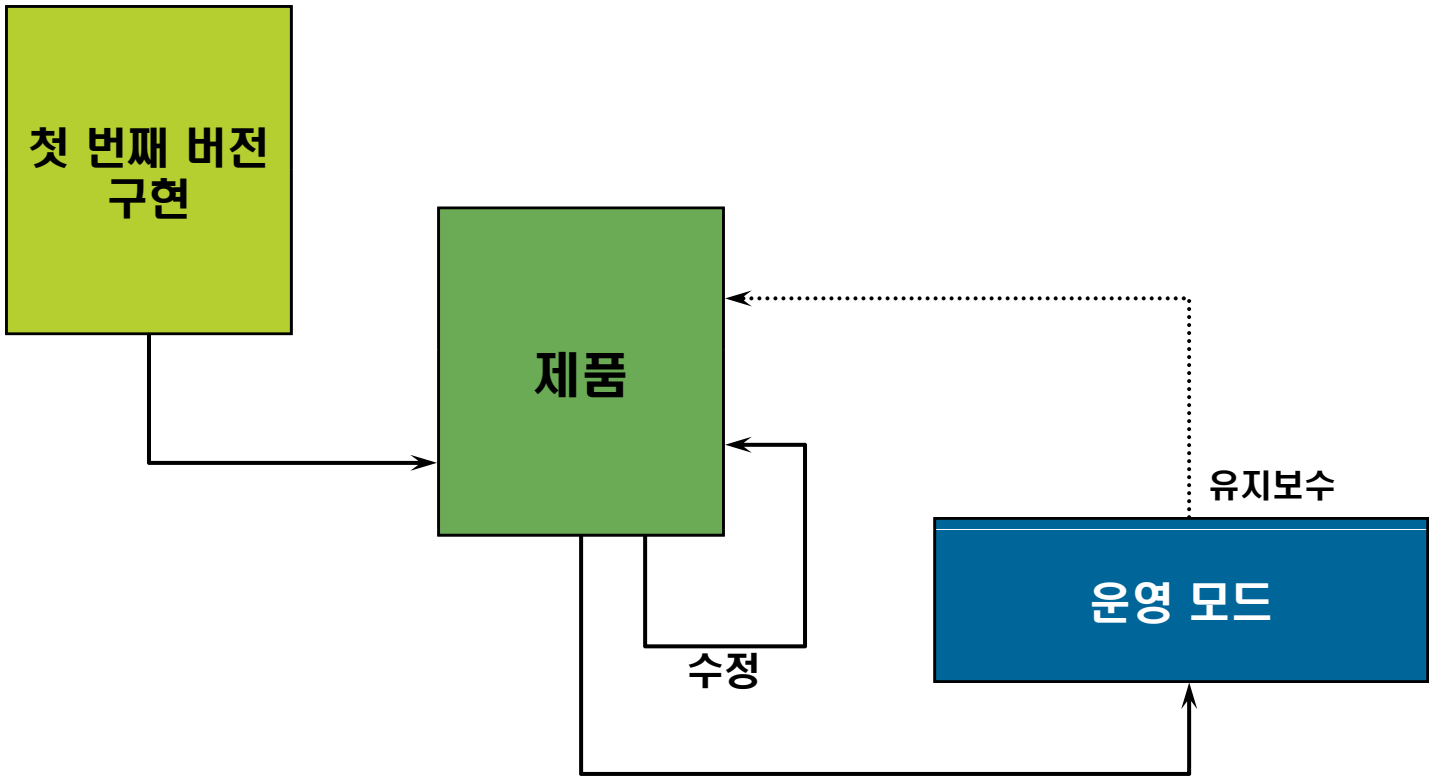
- 크기가 매우 작은 규모의 소프트웨어 개발

## ❖ 단점

- 정해진 개발 순서가 없기 때문에
  - 계획이 정확하지 않음
  - 관리자는 프로젝트 진행 상황 파악에 어려움
  - 개발 문서가 없기 때문에 개발 및 유지보수에 어려움

➡ 이후 체계적인 소프트웨어 개발 생명주기 모델의 연구를 가져옴

# 주먹구구식 개발 모델(Build-Fix Model) (2/2)

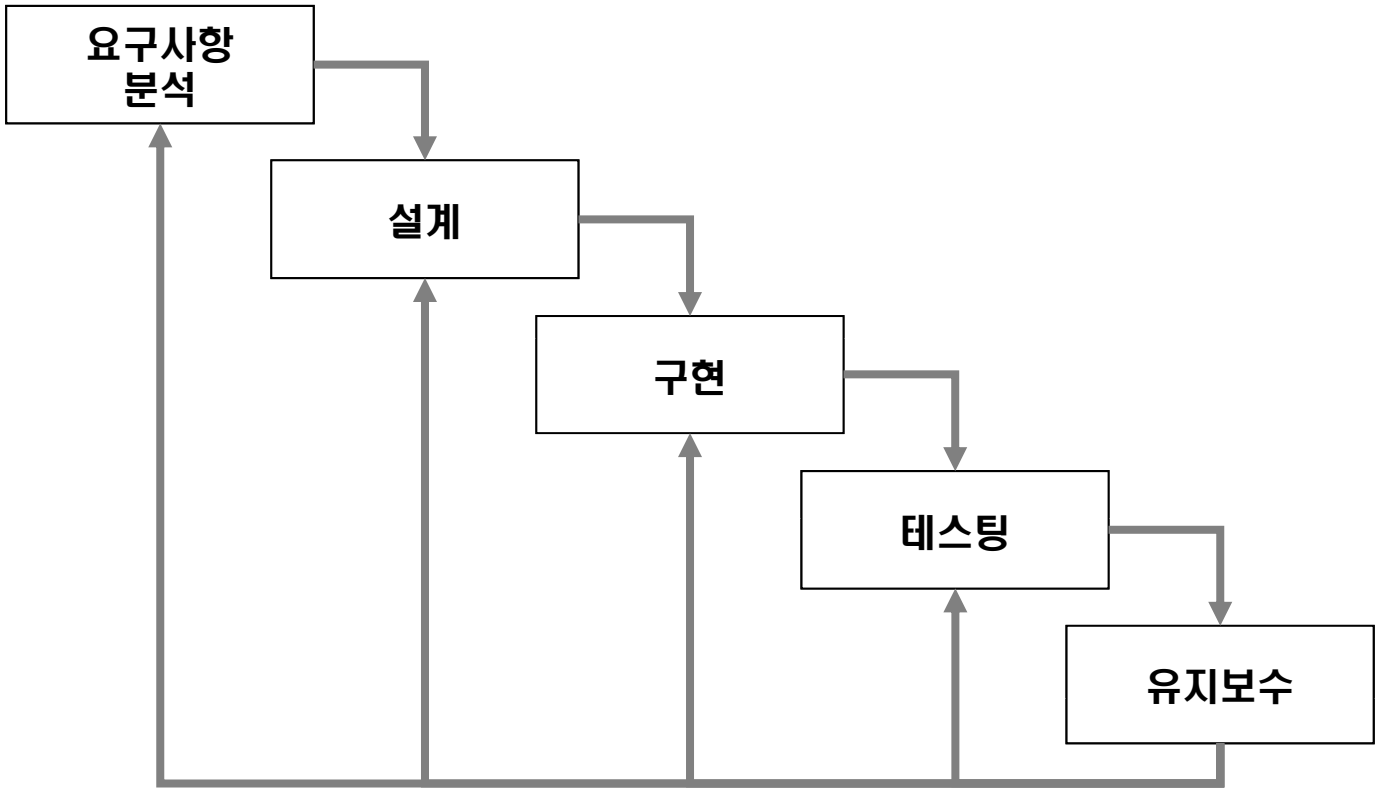


# 폭포수 모델(Waterfall Model) (1/4)

## ❖ 개요

- **순차적으로 소프트웨어를 개발하는 전형적인 개발 모델**
- **대부분의 소프트웨어 개발 프로젝트의 기본적 모델이며 가장 많이 사용되는 모델**
- **소프트웨어 개발의 전 과정을 나누어 체계적이고 순차적으로 접근하는 방법**
  - 개발 과정: 요구사항 분석, 설계, 구현, 테스팅, 유지보수

# 폭포수 모델(Waterfall Model) (2/4)



# 폭포수 모델(Waterfall Model) (3/4)

## ❖ 단계별 활동

- **요구사항 분석**
  - 개발하고자 하는 소프트웨어에 대한 요구사항 수집, 문제 이해 및 분석 단계
  - 소프트웨어 엔지니어 또는 분석가: 고객의 요구사항을 기능, 성능, 인터페이스 등으로 파악하고 문서화
  - 산출물: 요구사항 명세서(Requirement Specification)
- **설계**
  - 프로그램의 데이터 구조, 소프트웨어 구조, 인터페이스 구조, 알고리즘 등 모든 시스템의 구조 결정
  - 산출물: 설계 명세서
- **구현**
  - 설계 명세서를 시스템의 실제 모습으로 변환 시키는 것
  - 산출물: 프로그램
- **테스트**
  - 프로그램이 입력에 따라 요구되는 결과대로 작동하는지, 내부적 이상 여부 및 오류 발견을 위해 수행
  - 테스트 계획을 세운 후 문서화
- **유지보수**
  - 개발된 소프트웨어의 변경사항을 수정하는 것
  - 수정 유지보수, 적응 유지보수, 기능 추가 유지보수 등이 있음

# 폭포수 모델(Waterfall Model) (4/4)

## ❖ 장점

- 각 단계별로 정형화된 접근 방법 가능
- 체계적인 문서화가 가능하여 프로젝트 진행을 명확하게 할 수 있음

## ❖ 단점

- 앞 단계가 완료될 때까지 다음 단계들은 대기 상태여야 함
- 실제 작동되는 시스템을 개발 후반부에 확인 가능하기 때문에 고객이 요구사항 확인하는데 많은 시간이 걸림

# 원형 모델(Prototyping Model) (1/3)

## ❖ 개요

- 폭포수 모델의 단점을 보완한 모델
- 점진적으로 시스템을 개발해 나가는 접근 방법
- 원형(Prototype)을 만들어 고객과 사용자가 함께 평가한 후 개발될 소프트웨어의 요구사항을 정제하여 보다 완전한 요구사항 명세서를 완성함

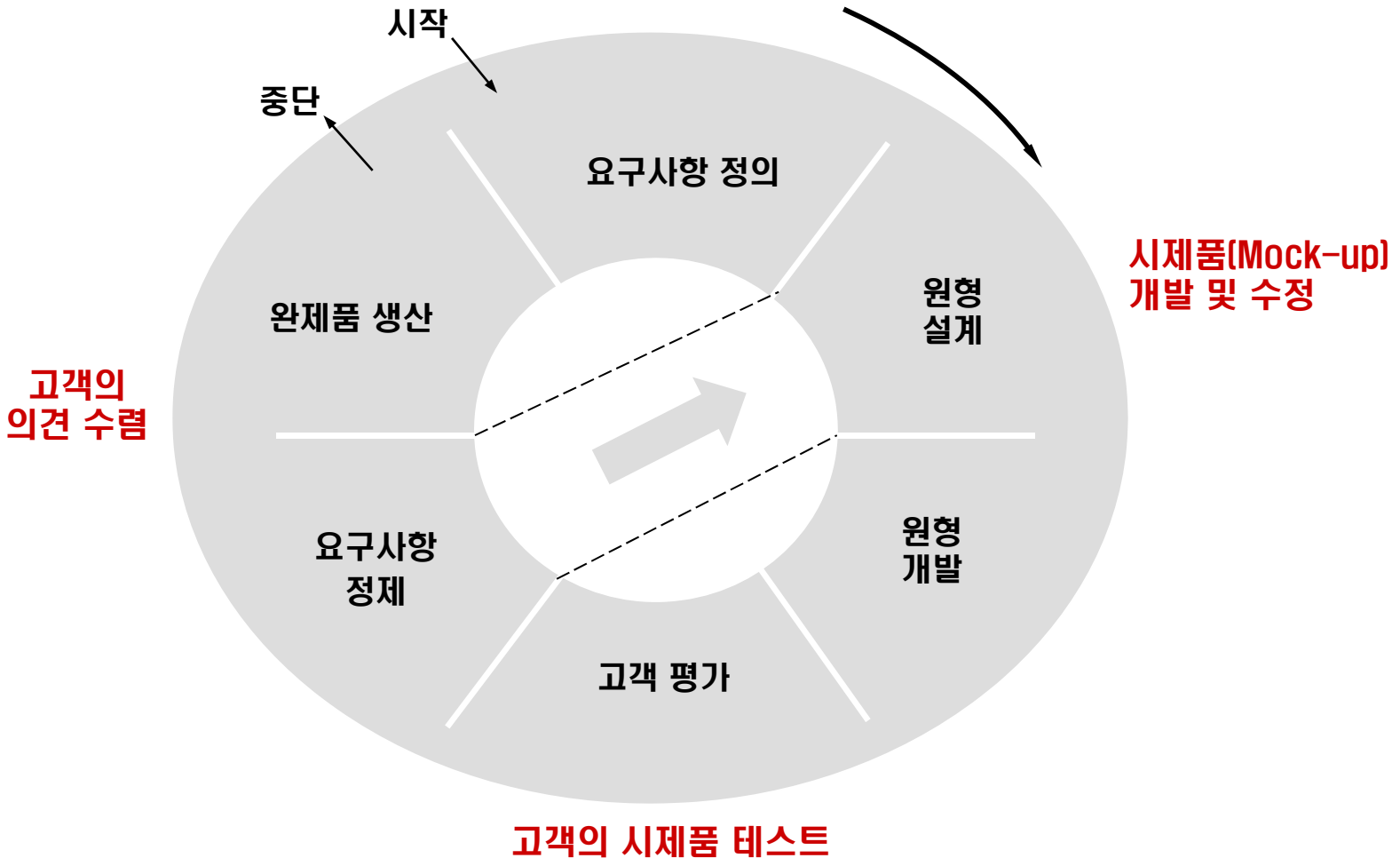
## ❖ 목적

- 원형을 가능한 빨리 개발하여 고객과 검증하는 것
- 방법
  - 고객으로부터 피드백을 받은 후 원형을 폐기
  - 시스템 기능 중 중요한 부분만 구현하여 피드백을 얻은 후 지속적으로 발전시켜 완제품을 제작

## ❖ 적용 가능한 경우

- 소프트웨어 개발 초기에 고객 요구사항을 완전히 파악하기 어려울 때

# 원형 모델 (Prototyping Model) (2/3)



# 원형 모델 (Prototyping Model) (3/3)

## ❖ 단계별 활동

- **요구사항 정의**
  - 고객의 일부 요구사항 또는 불완전한 요구 사항으로부터 제품을 윤곽을 잡음
- **원형 설계**
  - 주어진 요구사항을 기반으로 빠른 설계를 함
  - 주로 제품의 사용자 인터페이스에 초점을 맞춤
- **원형 개발**
  - 설계된 원형을 RAD(Rapid Application Development) 도구 등을 사용하여 빠르게 구현함
  - 고객이 요구하는 기능을 구현하고 필요한 요소를 파악하는데 중점을 둠
  - 프로그램의 신뢰도나 품질이 아니라 가능한 빨리 원형을 구현하는 것이 목적
- **고객 평가**
  - 고객과 개발자가 함께하는 가장 중요한 단계
  - 고객 요구사항을 정확하게 규명하기 위해 원형에 대한 사용 및 평가 시간을 충분히 제공
  - 개발될 소프트웨어의 요구사항 정제에 중요한 정보로 활용
- **원형 정제**
  - 원형이 어떻게 수정되어야 할지를 결정함
  - 원형 개발과 검증, 요구사항 정제의 순환을 반복하여 추가적인 정보를 통해 요구사항을 완성해 나감

# 나선형 모델(Spiral Model) (1/4)

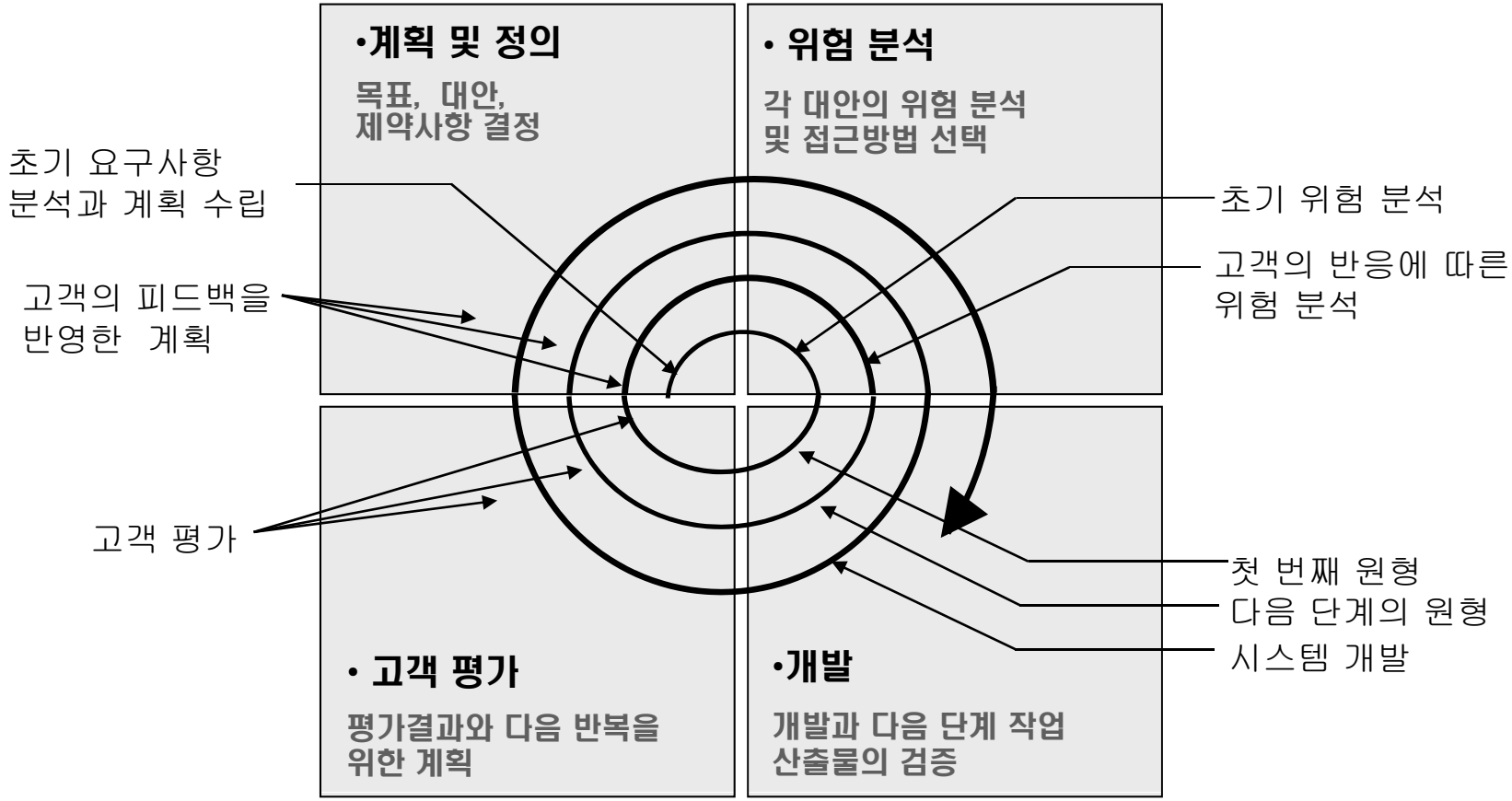
## ❖ 개요

- 폭포수 모형과 원형 모형의 장점을 수용하고 위험 분석(Risk analysis)을 추가한 점증적 개발 모델
- 프로젝트 수행 시 발생하는 위험을 관리하고 최소화 하려는 것이 목적

## ❖ 특징

- 여러 개의 작업 영역으로 구분
- 나선상의 각 원은 소프트웨어 개발의 점증적 주기 표현
  - 가장 안쪽 타원부터개념적 개발 프로젝트, 실제 제품 개발 프로젝트, 제품 향상 프로젝트, 유지보수 프로젝트
- 단계가 명확히 구분되지 않고, 엔지니어가 프로젝트 성격이나 진행 상황에 따라 단계 구분

# 나선형 모델(Spiral Model) (2/4)



# 나선형 모델(Spiral Model) (3/4)

## ❖ 단계별 활동

- **계획 및 정의 단계**
  - 개발자는 고객에게 요구사항을 수집
  - 개발자는 시스템의 성능, 기능을 비롯한 시스템의 목표를 규명하고 제약 조건을 파악
  - 목표와 제약 조건에 대한 여러 대안들을 고려하고 평가함으로써 프로젝트 위험의 원인을 규명 가능
- **위험 분석 단계**
  - 초기의 요구 사항을 토대로 위험 규명
  - 위험에 대한 평가가 이루어지면 프로젝트를 계속 진행할 것인지 아니면 중단할 것인지를 결정
- **개발 단계**
  - 시스템에 대한 생명주기 모델을 선택하거나 원형 또는 최종적인 제품을 만드는 단계
- **고객 평가 단계**
  - 구현된 소프트웨어(시뮬레이션 모형, 원형 또는 실제 시스템)를 고객이나 사용자가 평가함
  - 고객의 피드백을 얻는데 필요한 작업이 포함
  - 다음 단계에서 고객의 평가를 반영할 수 있는 자료 획득 가능

# 나선형 모델(Spiral Model) (4/4)

## ❖ 적용 가능한 경우

- **개발에 따른 위험을 잘 파악하여 대처할 수 있기 때문에**
  - 고비용의 시스템 개발
  - 시간이 많이 소요되는 큰 시스템 구축 시

## ❖ 장점

- **프로젝트의 모든 단계에서 기술적인 위험을 직접 고려할 수 있어 사전에 위험 감소 가능**
- **테스트 비용이나 제품 개발 지연 등의 문제 해결 가능**

## ❖ 단점

- **개발자가 정확하지 않은 위험 분석을 했을 경우 심각한 문제 발생 가능**
- **비교적 새로운 접근 방식**
- **폭포수, 원형 모델에 비해 상대적으로 복잡하여 프로젝트 관리 자체에 어려울 수 있음**

# UP(Unified Process) (1/4)

## ❖ 개요

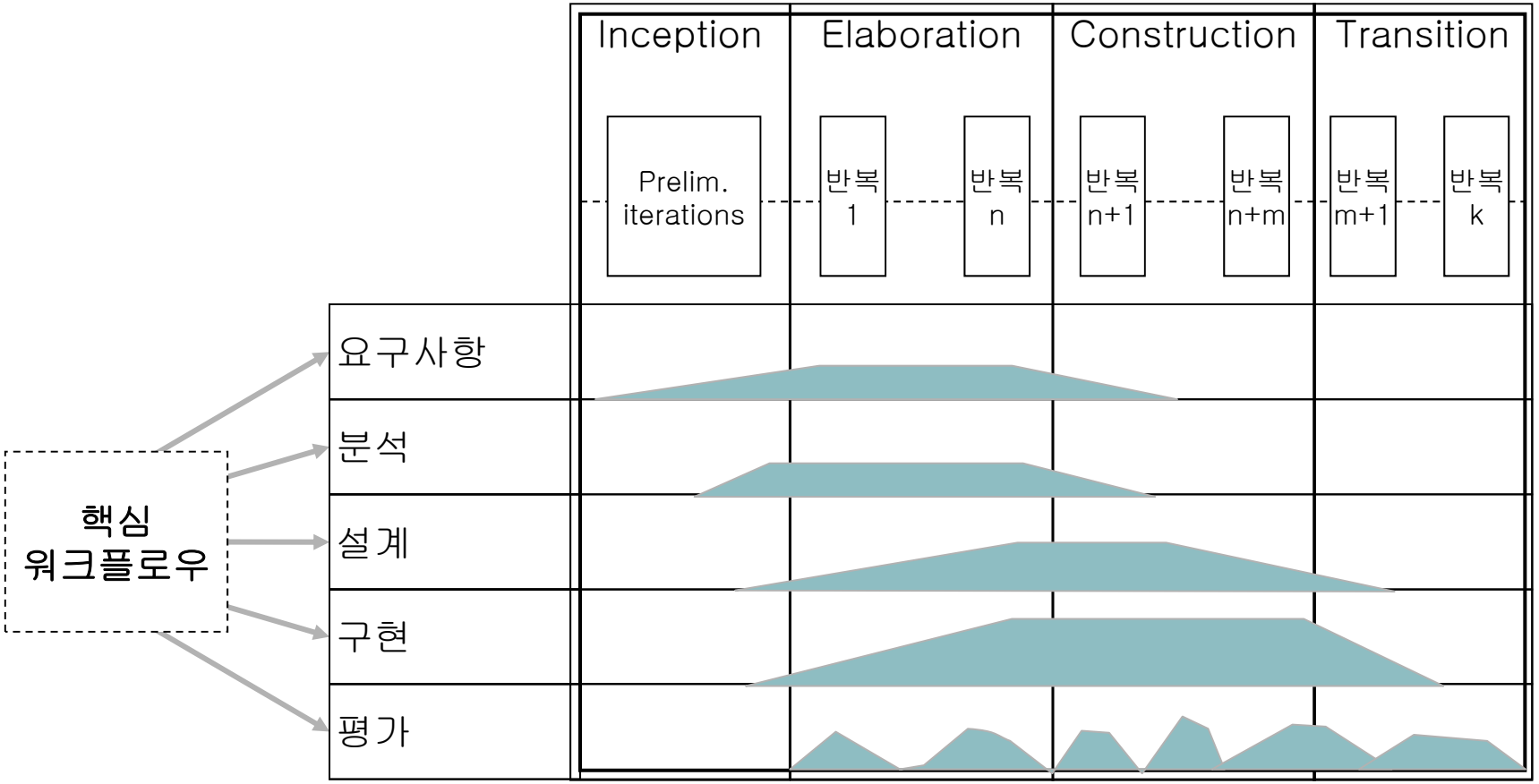
- 1999년 Jacobson, Booch, Rumbaugh에 의해 개발됨
- 소프트웨어 개발이 각각 생명주기를 가지는 여러 번의 반복(iteration)을 거쳐 수행되는 모델

## ❖ 특징

- 반복마다 실행 가능한 릴리즈가 산출
- 반복이 거듭될수록 향상되어 결국 최종 시스템으로 발전됨

# UP(Unified Process) (2/4)

❖ **반복 프로세스의 4가지 범주**



# UP(Unified Process) (3/4)

## ❖ 단계별 활동

- **도입(Inception)**
  - 소프트웨어 개발 주문에 관련된 사람들(고객, 사용자, 재무적 지원자 등)과의 준비적 상호 작용 단계
  - 요구사항 분석, 원형에 대한 설계, 구현 단계
- **상세(Elaboration)**
  - 어떠한 시스템이 필요한지를 확정하는 단계
  - 요구사항 분석 및 아키텍쳐(Architecture) 확정 단계
- **구축(Construction)**
  - 기본 기능만을 가진 제품 산출 단계로 주요 설계 및 구현을 수행
  - 여전히 릴리즈를 위해서는 기능을 보강해야 함
- **이행(Transition)**
  - 제품 릴리즈 완성 단계로서 구현 및 테스트 수행

# UP(Unified Process) (4/4)

## ❖ 장점

- 위험 요소 초기 발견 가능
- 아키텍쳐에 대한 정의를 중요한 요소로 삼고 개발하기 때문에 이해가 쉽고 견고하며 변경 관리가 용이함

# XP(eXtreme Programming) (1/4)

## ❖ 개요

- 1990년대 초, Kent Beck에 의해 고안된 개발 방법론
- 요구사항 변경으로 인한 비용이 개발 기간에 상관없이 일정하게 유지되도록 것을 주목적으로 함
- 고객, 관리자, 프로그래머에 대한 역할 및 권한과 4가지 가치를 중시함
  - 4가지 가치: 의사소통(Communication), 단순성(Simplicity), 피드백(Feedback), 용기(Courage)

## ❖ 특징

- 프로그래머와 고객, 동료 프로그래머와의 의사소통 중요시 함
- 단순하고 명확한 설계 유지
- 가장 우선순위가 높은 것을 먼저 개발함
- 되도록 초기에 고객에게 시스템을 전달하여 피드백을 받음
- 프로그래머는 요구사항과 기술의 변경에 용감하게 대응할 수 있음

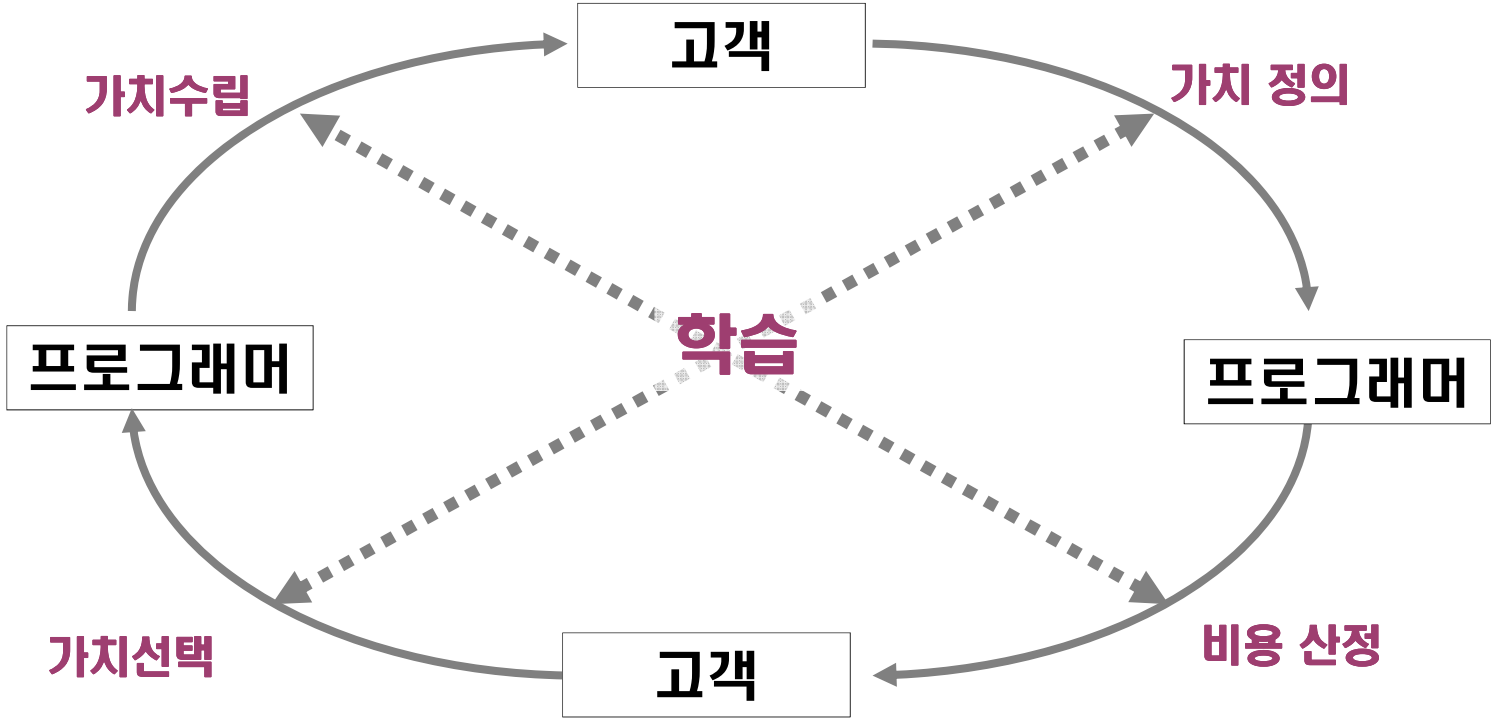
# XP(eXtreme Programming) (2/4)

## ❖ 역할 설명

- **프로그래머**
  - 분석, 설계, 테스트, 코딩, 시스템 통합 수행
  - 각 업무에 대한 난이도를 추정
  - 고객에게 시스템 전달을 위한 속도를 조절/관리
- **관리자**
  - 고객과 프로그래머가 함께 잘 일할 수 있도록 지원
- **고객**
  - 시스템에 대한 요구사항을 정리
  - 우선순위를 설정

# XP(eXtreme Programming) (3/4)

❖ 각 과정에서의 경험을 다음 생명주기에 반영된다.



# XP(eXtreme Programming) (4/4)

## ❖ XP의 12가지 실천 사항

- 계획 세우기(Planning Process)
- 소규모 릴리즈(Small Release)
- 상징(Metaphor)
- 단순한 디자인(Simple Design)
- 지속적인 테스팅(Continuous Testing)
- 리팩토링(Refactoring)
- 짝 프로그래밍(Pair Programming)
- 공동 코드 소유(Collective Code Ownership)
- 지속적인 통합(Continuous Integration)
- 주당 40시간 업무(40 hour Week)
- 현장 고객 지원(On-site Customer)
- 코딩 표준(Coding Standard)

# 연습문제

1. 소프트웨어 생명 주기의 역할은 무엇인가?
2. 소프트웨어 개발 생명주기 단계 중 가장 오랜 시간이 걸리며, 대부분의 비용을 차지하는 단계는 무엇인가?
3. 소프트웨어 개발 생명주기 중 폭포수 모델(Waterfall Model)의 개발 단계를 표현하라.
4. 폭포수 모델에 대하여 설명하라.
5. 실제 상황이 나오기 전에 가상으로 시뮬레이션을 통해 최종 결과물에 대한 예측을 할 수 있는 소프트웨어 개발 생명주기는 무엇인가?
6. 원형 모델의 가장 큰 장점은 무엇인가?
7. 원형 모델의 개발에 필요한 작업을 순서대로 표현하라.
8. 나선형 모델에 대하여 설명하라.
9. 나선형 모델의 태스크(task)를 나열하라.

# 팀 프로젝트

## 3주차

# 이번 주 할일

- 프로젝트를 진행하기 위한 팀 구성을 확정합니다.

- 제출 내용
  - 팀명
  - 팀원, 팀장(각 이름, 학번)

# 다음 주 제출 문서

- 템플릿 참조하여 프로젝트 제안서를 작성하시오.