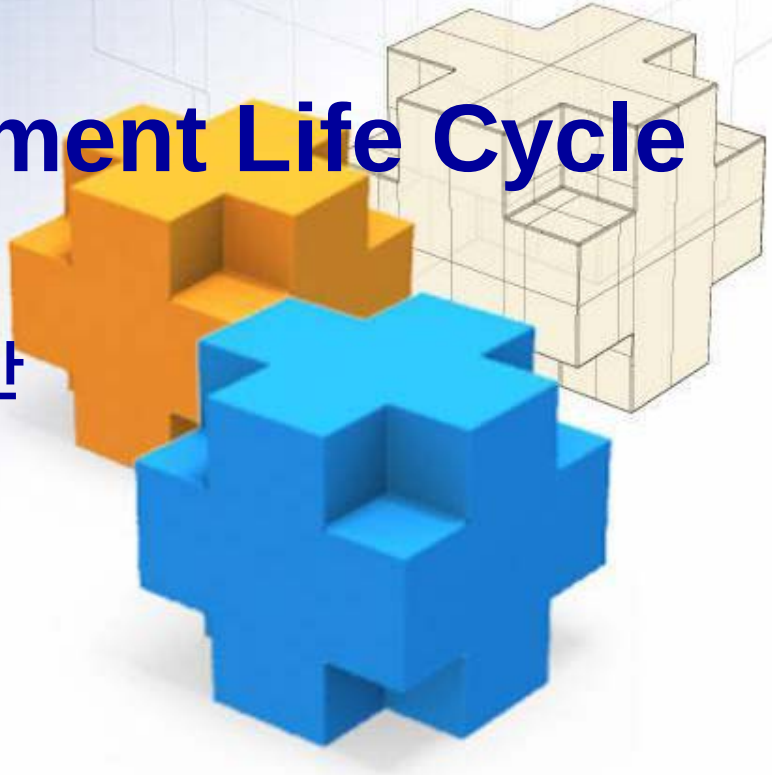
LECTURE

1

소프트웨어 개발 프로세스

Software Development Life Cycle



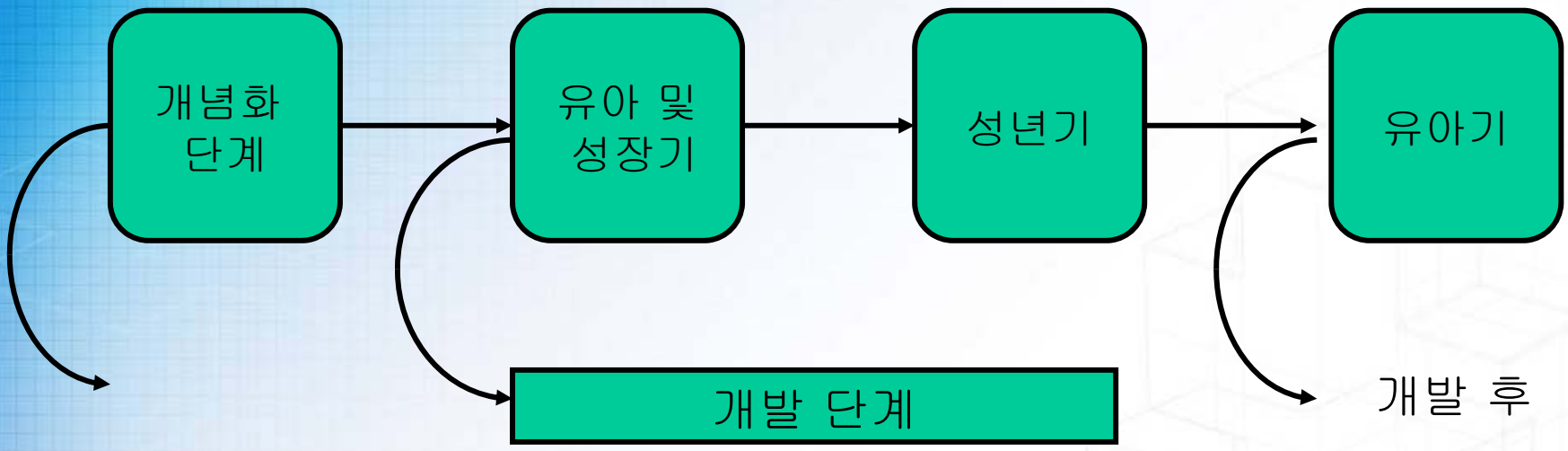
최은만

개 요

- 소프트웨어 개발 생명주기란 무엇인가?
- 왜 생명주기 프로세스가 필요한가?
- 생명주기 모델과 적용
  - "Code-and-fix"
  - 폭포수 모형
  - 프로토타이핑 모형
  - 점증적 모형
  - 나선형 모형
  - V 모형
  - 일정 중심 설계 모형, 진화적 출시 모형
  - 애자일 모형

소프트웨어 생명주기 모형

❑ 소프트웨어 생명주기



생명주기란?

- **소프트웨어를 개발해 나가는 단계나 과정**
  - 컨셉트를 정하는 것부터 소멸될 때까지
  - 몇 달 또는 몇 년이 걸릴 수 있음

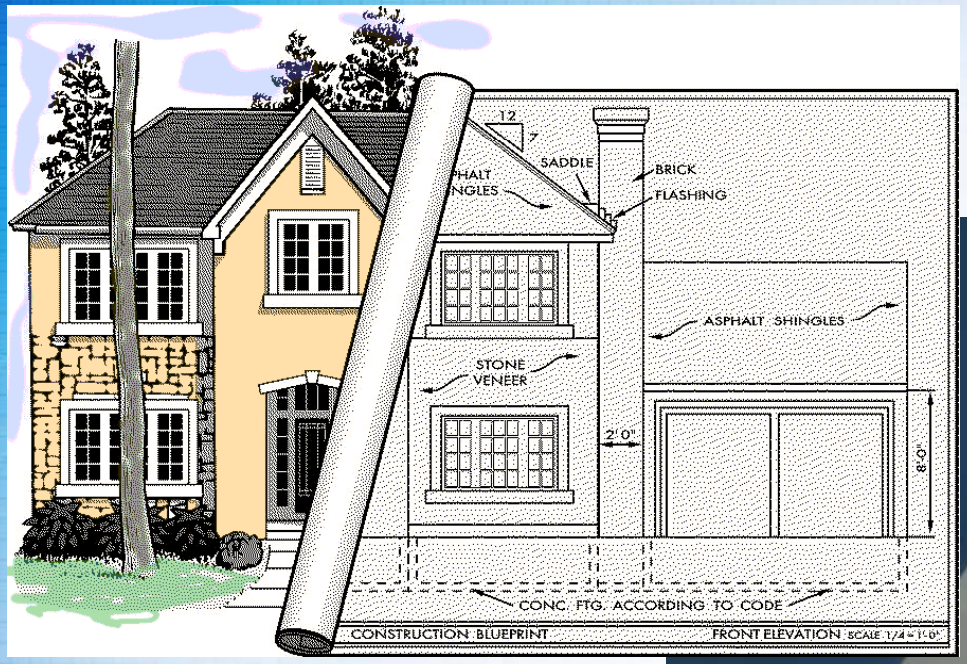
- **각 단계의 목표**
  - 명확한 작업 단계
  - 손에 잡히는 결과
  - 작업의 검토
  - 다음 단계의 명시

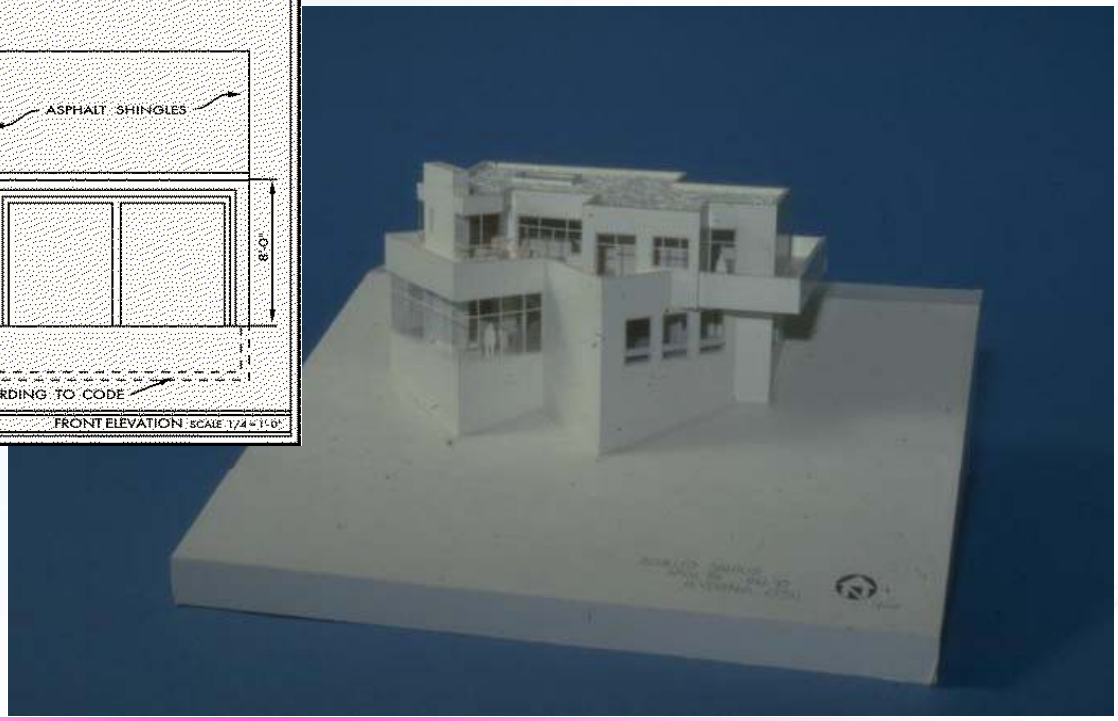
생명주기를 사용하는 이점

- 어떤 작업을 해야 할지를 구조적으로 제공
- "Big picture"를 볼 수 있게 함. 따라가면서 작업하면 실수에 괘념하지 않고 목표에 도달
- 생명주기가 없다면 결정이 중구난방이 될 수 있음
- 관리 도구

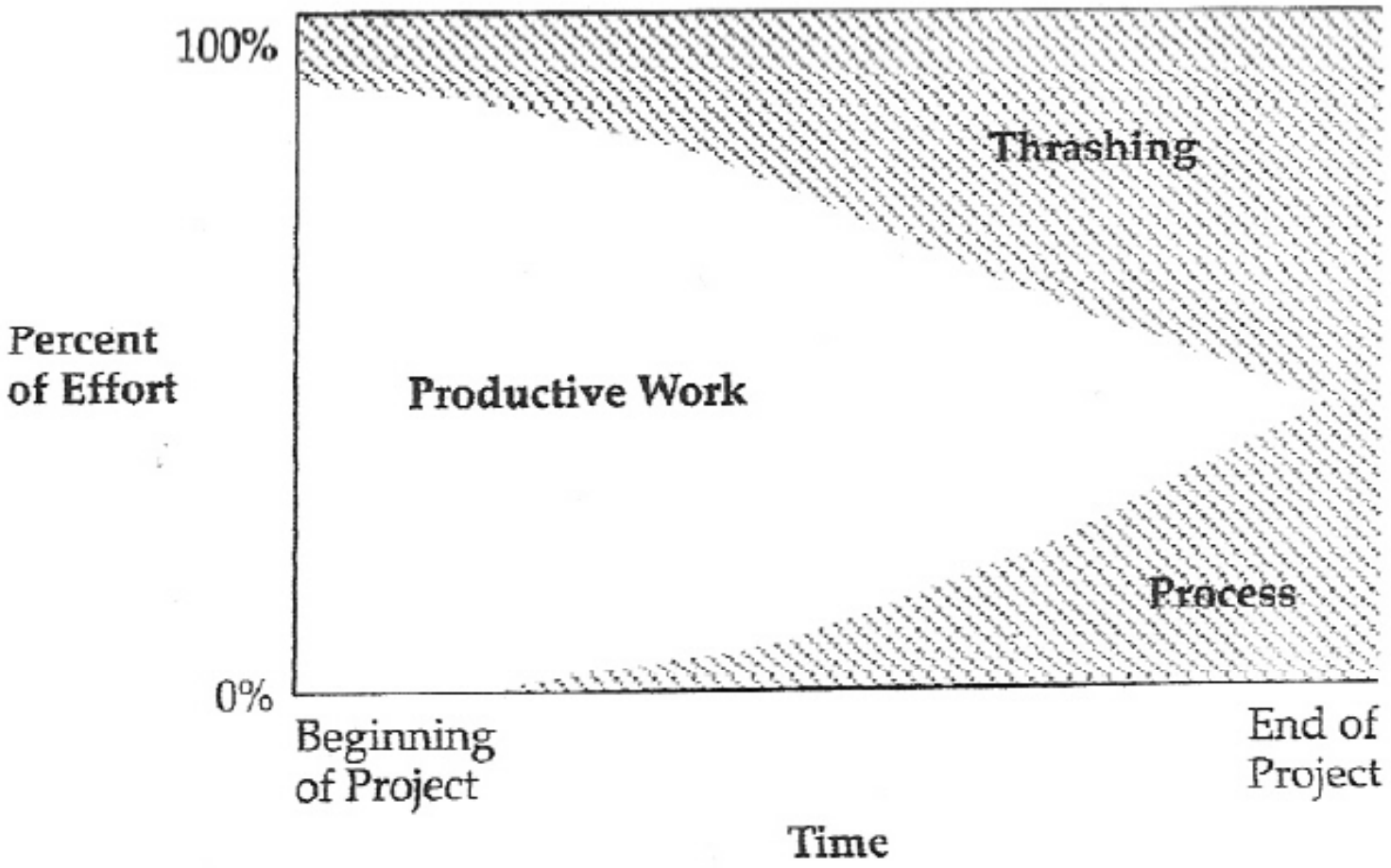
SE와 유사한 작업들

❑ 건물의 건축

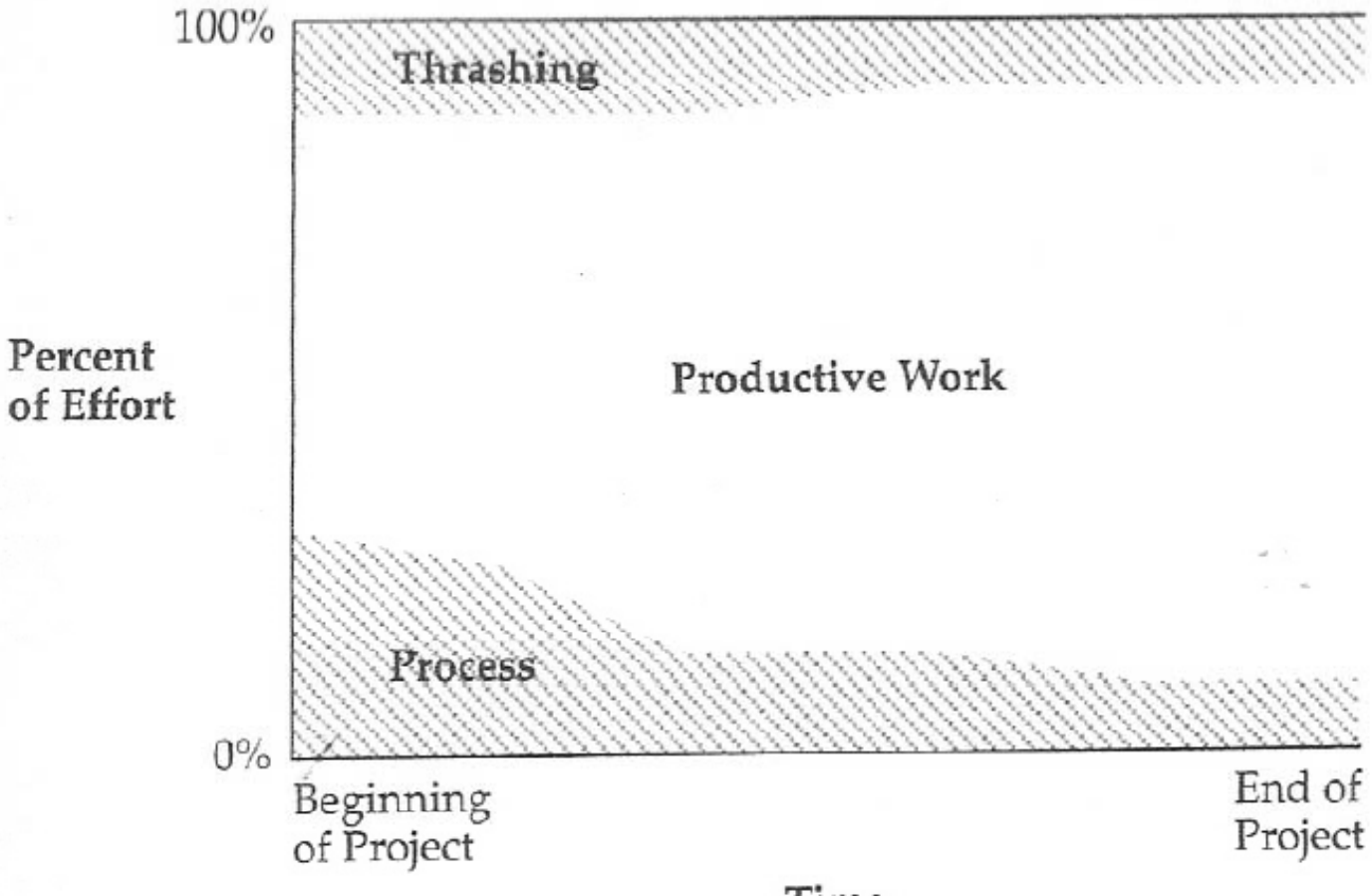




프로세스에 관심을 두지 않은 프로젝트



조기에 프로세스에 관심을 둔 프로젝트



Survival Guide:
McConnell p25

Code-and-Fix 형

- 공식적인 가이드라인이나 프로세스 없이 소프트웨어를 개발해 나가는 형태

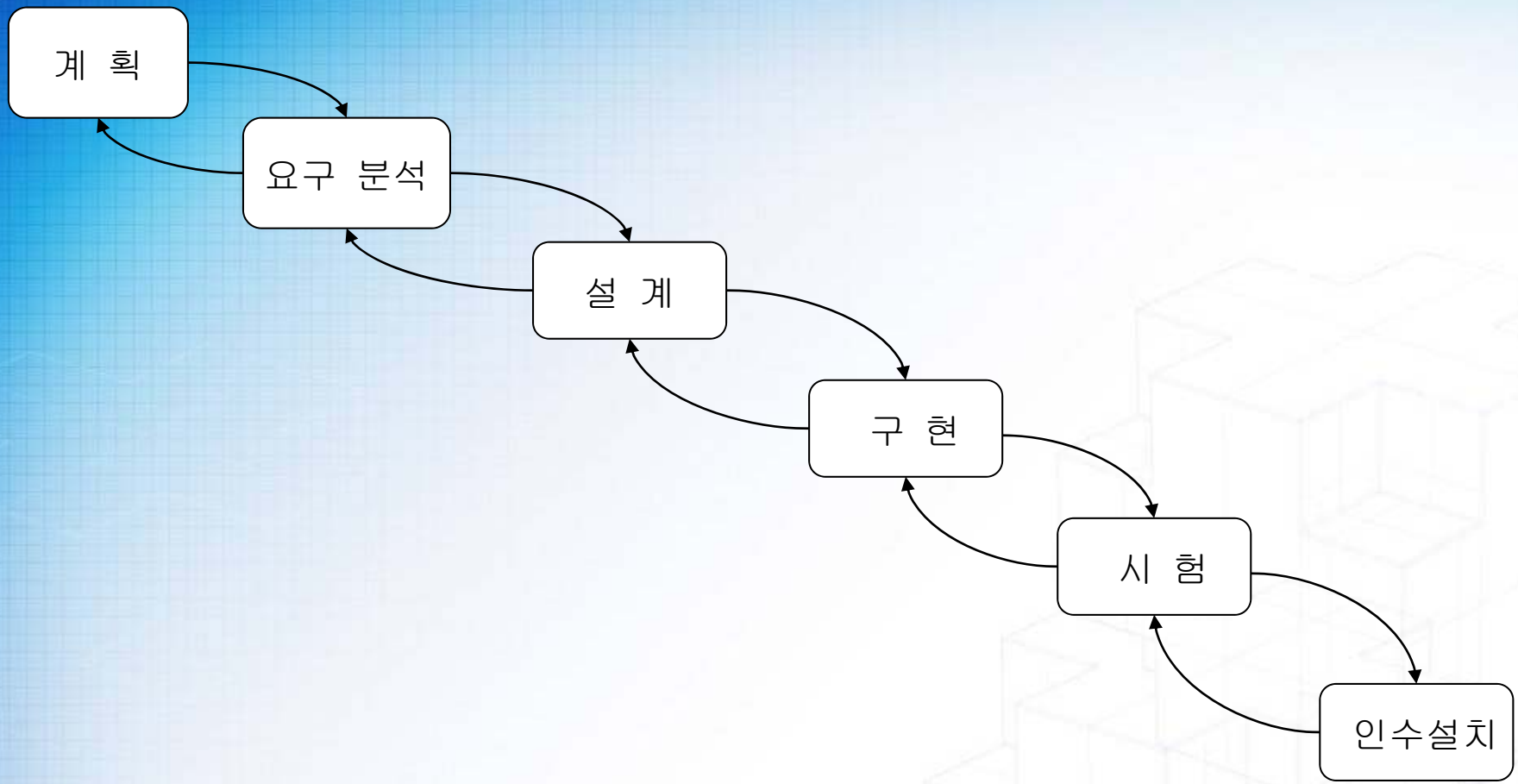
- Code-and-Fix의 단점
  - 중요한 작업(설계, 테스트)들이 무시됨
  - 각 작업이 언제 시작되어야 할지 언제 끝날지 불명확
  - 대규모 작업에 적용하기 어려움
  - 개인의 작업을 리뷰 하거나 평가하기 어려움

- 문제가 늦게 발견될수록 고치는 비용은 커짐

대표적인 생명주기 모형

- 중요한 모형
  - 폭포수 모형
  - 프로토타이핑 모형
  - 점증적 모형
  - 나선형 모형
  - V 모형
  - 일정 중심 설계 모형
  - 진화적 출시 모형

폭포수(waterfall) 모형



계획

- 다음 질문의 대답을 찾는 단계
  - How much will it cost?
  - How long will it take?
  - How many people will it take?
  - What might go wrong?

Why 단계

- 범위 정하기

ROI

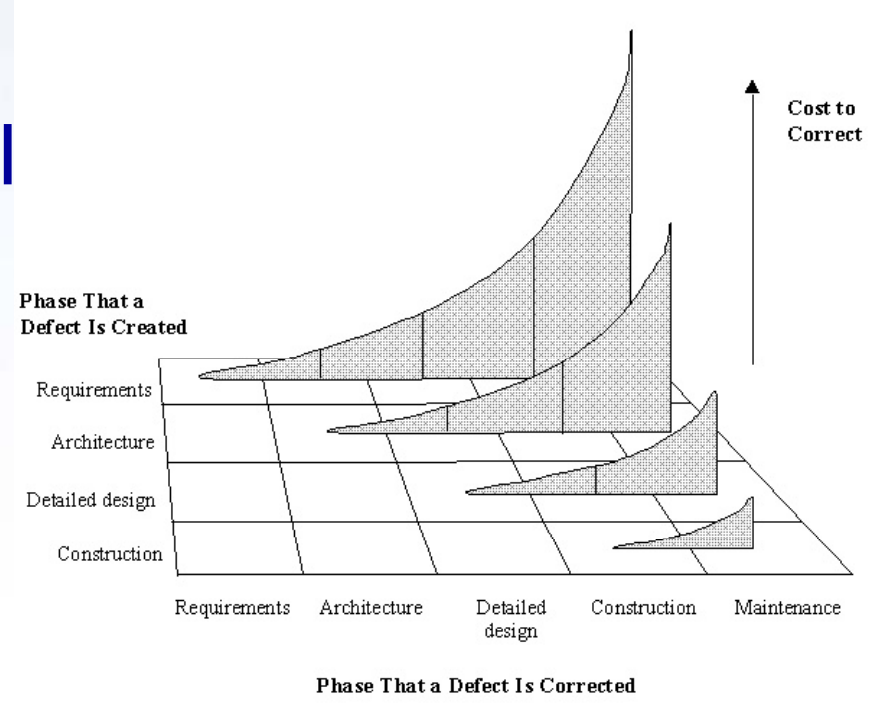
- 산정(Estimation)
- 리스크 분석

Concept 정립

- 일정 계획
- 관리 전략 수립

요구 분석

- 요구 – 시스템이 가져야 할 능력(capability)과 조건(condition)
- What 의 단계
- 응용 분야(도메인)에 집중
- 가장 중요하고도 어려운 단계
  - 조그만 차이가 큰 오류로 변함
- 결과물: 요구분석서(SRS)



설계

- How의 단계
- 솔루션에 집중
- 아키텍처 설계
- 데이터베이스 설계
- UI 설계
- 상세 설계
- 결과물: 설계서(SD)

구현

- ‘Do it’ 단계
- 코딩과 단위 테스트
- 설계 또는 통합 단계와 겹치기도 함
  - 전체 일정을 줄이기 위하여
  - 협력 작업이 필요한 경우
- 특징
  - 압력 증가
  - 최고의 인력 투입
- 이슈
  - Last minute change
  - Communication overhead
  - 하청 관리

통합과 테스트

- 병행
  - 통합해 나가면서 테스트 시작
- 모듈의 통합으로 시작
- 점차 완성된 모듈을 추가
- 통합은 개발자가 주로 담당
- 테스트는 QA 팀이 주로 담당
- 단계적인 테스트
  - 단위, 통합, 시스템
- 목적 중심 테스트
  - 스트레스 테스트, 성능 테스트, 베타 테스트, Acceptance 테스트, Usability 테스트

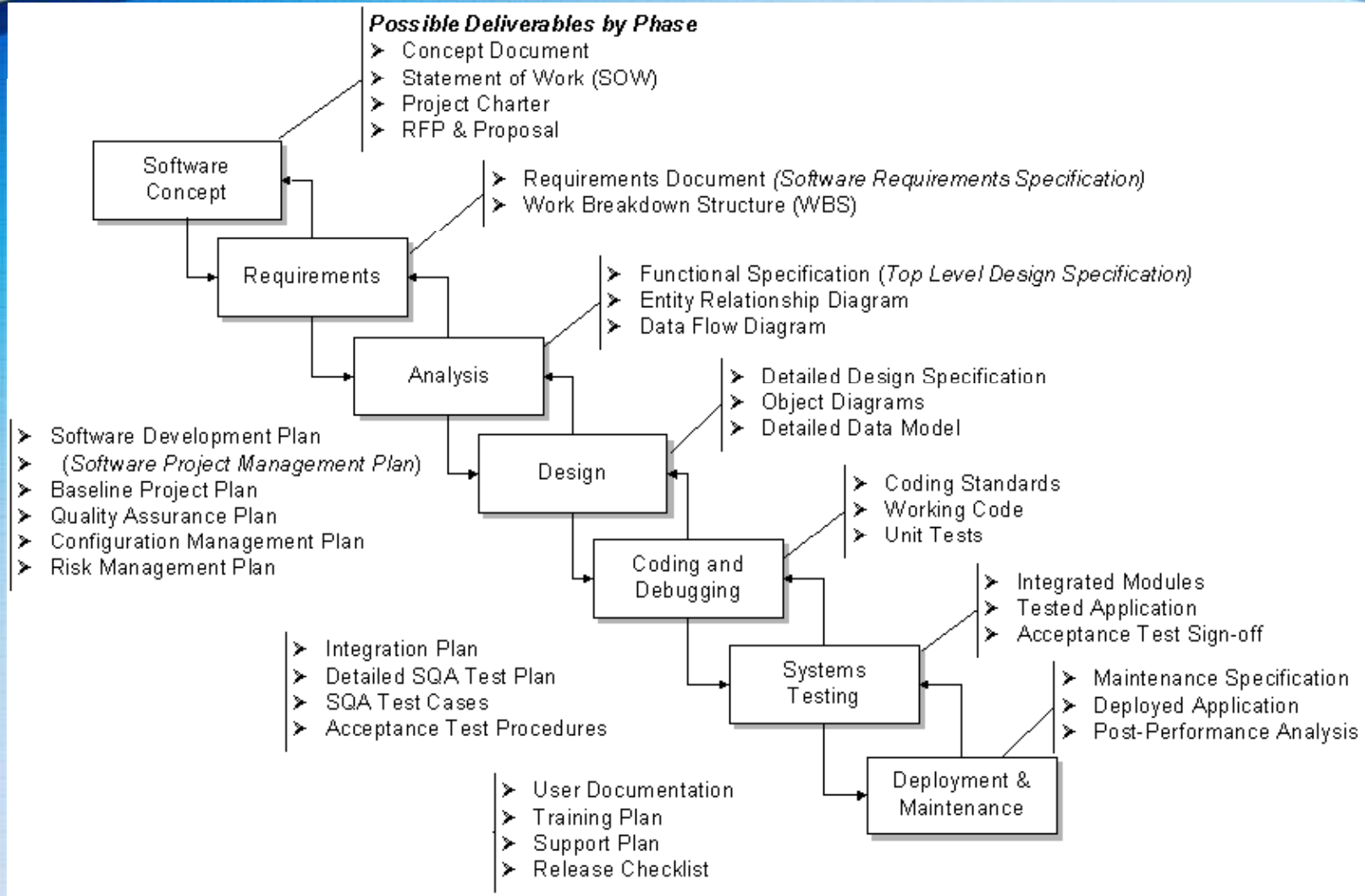
설치와 유지보수

- 시스템의 타입에 따라 다른 설치 방법
  - Web-based, CD-ROM, in-house, etc.
- 이전(Migration) 정책
- 시스템을 사용을 시작하게 하는 방법
  - 병행 운용
- 설치는 개발 프로젝트의 일부, 유지보수는 별개
- 유지보수
  - 결함을 고침
  - 새 기능 추가
  - 성능 추가

폭포수(waterfall) 모형

- 1970년대 소개
  - 항공 방위 소프트웨어 개발 경험으로 습득
- 각 단계가 다음 단계 시작 전에 끝나야 함
  - 순서적 - 각 단계 사이에 중복이나 상호작용이 없음
  - 각 단계의 결과는 다음 단계가 시작 되기 전에 점검
  - 바로 전단계로 피드백
- 단순하거나 응용 분야를 잘 알고 있는 경우 적합
  - 한 번의 과정, 비전문가가 사용할 시스템 개발에 적합
- 결과물 정의가 중요
- Method vs. Methodology

폭포수 모델의 단계별 결과물



폭포수 모형의 장단점

❑ 장점

- ❍ 프로세스가 단순하여 초보자가 쉽게 적용 가능
- ❍ 중간 산출물이 명확, 관리하기 좋음
- ❍ 코드 생성 전 충분한 연구와 분석 단계

❑ 단점

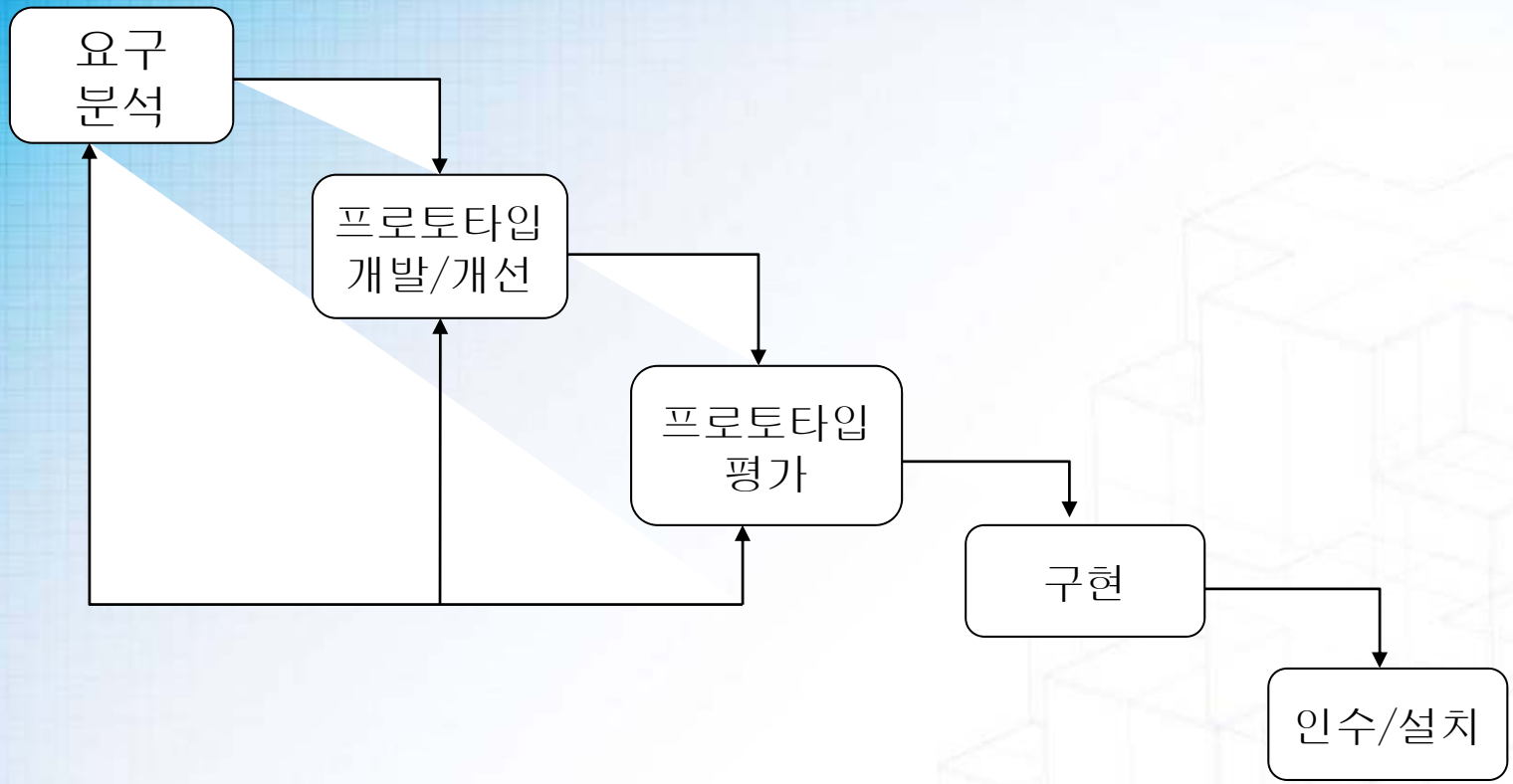
- ❍ 처음 단계의 지나치게 강조하면 코딩, 테스트가 지연
- ❍ 각 단계의 전환에 많은 노력
- ❍ 프로토타입과 재사용의 기회가 줄어듦
- ❍ 소용 없는 다종의 문서를 생산할 가능성 있음

❑ 적용

- ❍ 이미 잘 알고 있는 문제나 연구 중심 문제에 적합
- ❍ 변화가 적은 프로젝트에 적합

프로토타이핑 모형

❑ Rapid Prototyping Model(RAD)



프로토타이핑 모형

- **프로토타입(시범 시스템)의 적용**
  - 사용자의 요구를 더 정확히 추출
  - 알고리즘의 타당성, 운영체제와의 조화, 인터페이스의 시험 제작
- **프로토타이핑 도구**
  - 화면 생성기
  - 비주얼 프로그래밍, 4세대 언어 등
- **공동의 참조 모델**
  - 사용자와 개발자의 의사소통을 도와주는 좋은 매개체
- **프로토타입의 목적**
  - 단순한 요구 추출 – 만들고 버림
  - 제작 가능성 타진 - 개발 단계에서 유지보수가 이루어짐

프로토타이핑 모형의 장단점

❑ 장점

- ❍ 사용자의 의견 반영이 잘 됨
- ❍ 사용자가 더 관심을 가지고 참여할 수 있고 개발자는 요구를 더 정확히 도출할 수 있음

❑ 단점

- ❍ 오해, 기대심리 유발
- ❍ 관리가 어려움(중간 산출물 정의가 난해)

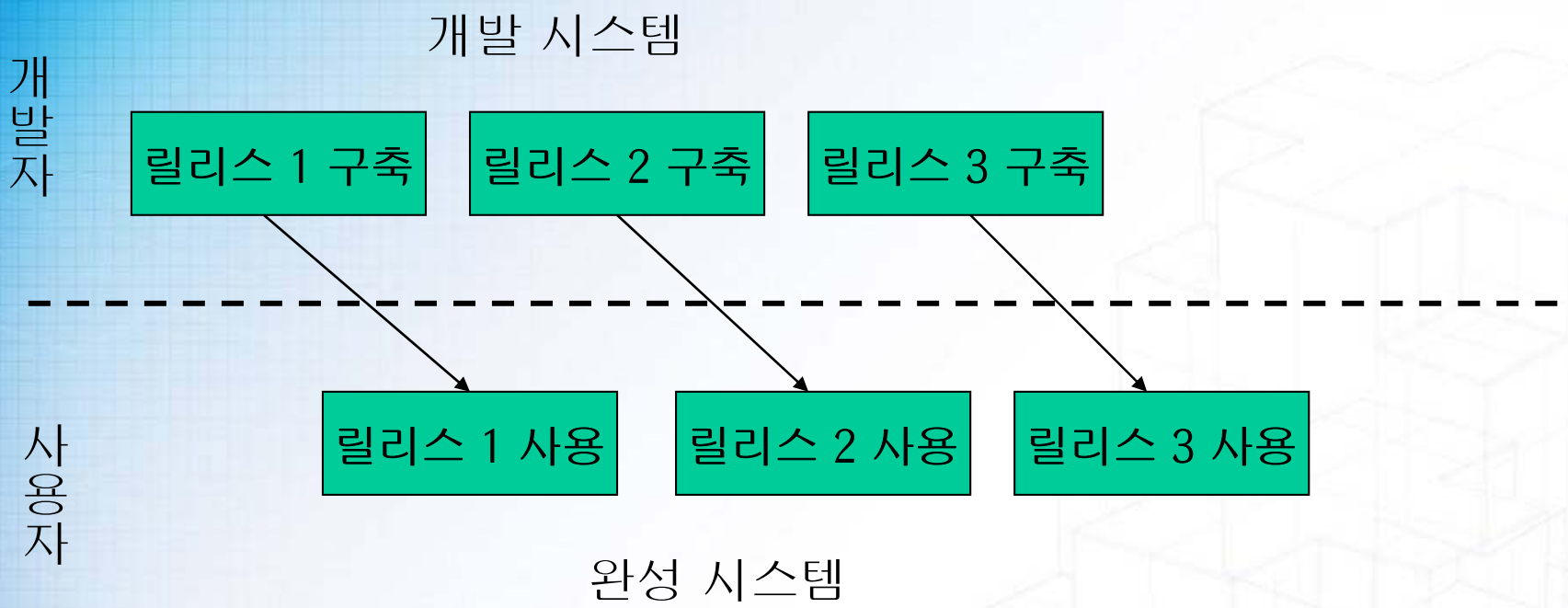
❑ 적용

- ❍ 개발 착수 시점에 요구가 불투명할 때
- ❍ 실험적으로 실현 가능성을 타진해 보고 싶을 때
- ❍ 혁신적인 기술을 사용해 보고 싶을 때

점증적 모형

❑ 개발 사이클이 짧은 환경

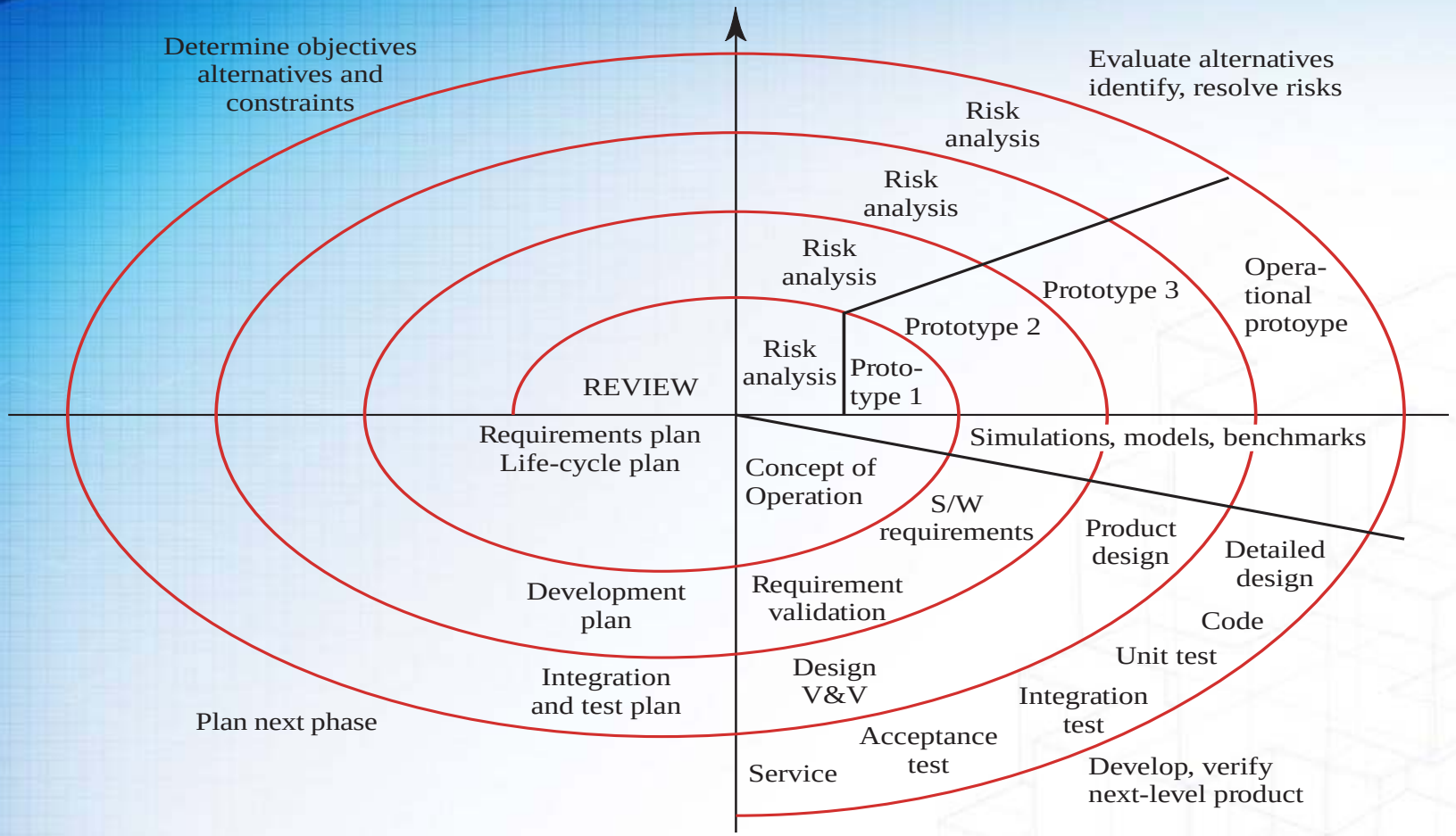
- ❍ 빠른 시간 안에 시장에 출시하여야 이윤에 직결
- ❍ 개발 시간을 줄이는 법 – 시스템을 나누어 릴리스



점증적 모형

- **릴리스 구성 방법**
  - 점증적 방법 – 기능별로 릴리스
  - 반복적 방법 – 릴리스 할 때마다 기능의 완성도를 높임
- **단계적 개발**
  - 기능이 부족하더라도 초기에 사용 교육 가능
  - 처음 시장에 내놓는 소프트웨어는 시장을 빨리 형성시킬 수 있음
  - 자주 릴리스 하면 가동 중인 시스템에서 일어나는 예상하지 못했던 문제를 신속 꾸준히 고쳐나갈 수 있음.
  - 개발 팀이 릴리스마다 다른 전문 영역에 초점 둘 수 있음.

나선형(spiral) 모형



나선형(spiral) 모형

- 소프트웨어의 기능을 나누어 점증적으로 개발
  - 실패의 위험을 줄임
  - 테스트 용이
  - 피드백
- 여러 번의 점증적인 릴리스(incremental releases)
- Boehm이 제안
- 진화 단계
  - 계획 수립(planning): 목표, 기능 선택, 제약 조건의 결정
  - 위험 분석(risk analysis): 기능 선택의 우선순위, 위험요소의 분석
  - 개발(engineering): 선택된 기능의 개발
  - 평가(evaluation): 개발 결과의 평가

나선형(spiral) 모형의 장단점

❑ 장점

- ❍ 대규모 시스템 개발에 적합 - risk reduction mechanism
- ❍ 반복적인 개발 및 테스트 - 강인성 향상
- ❍ 한 사이클에 추가 못한 기능은 다음 단계에 추가 가능

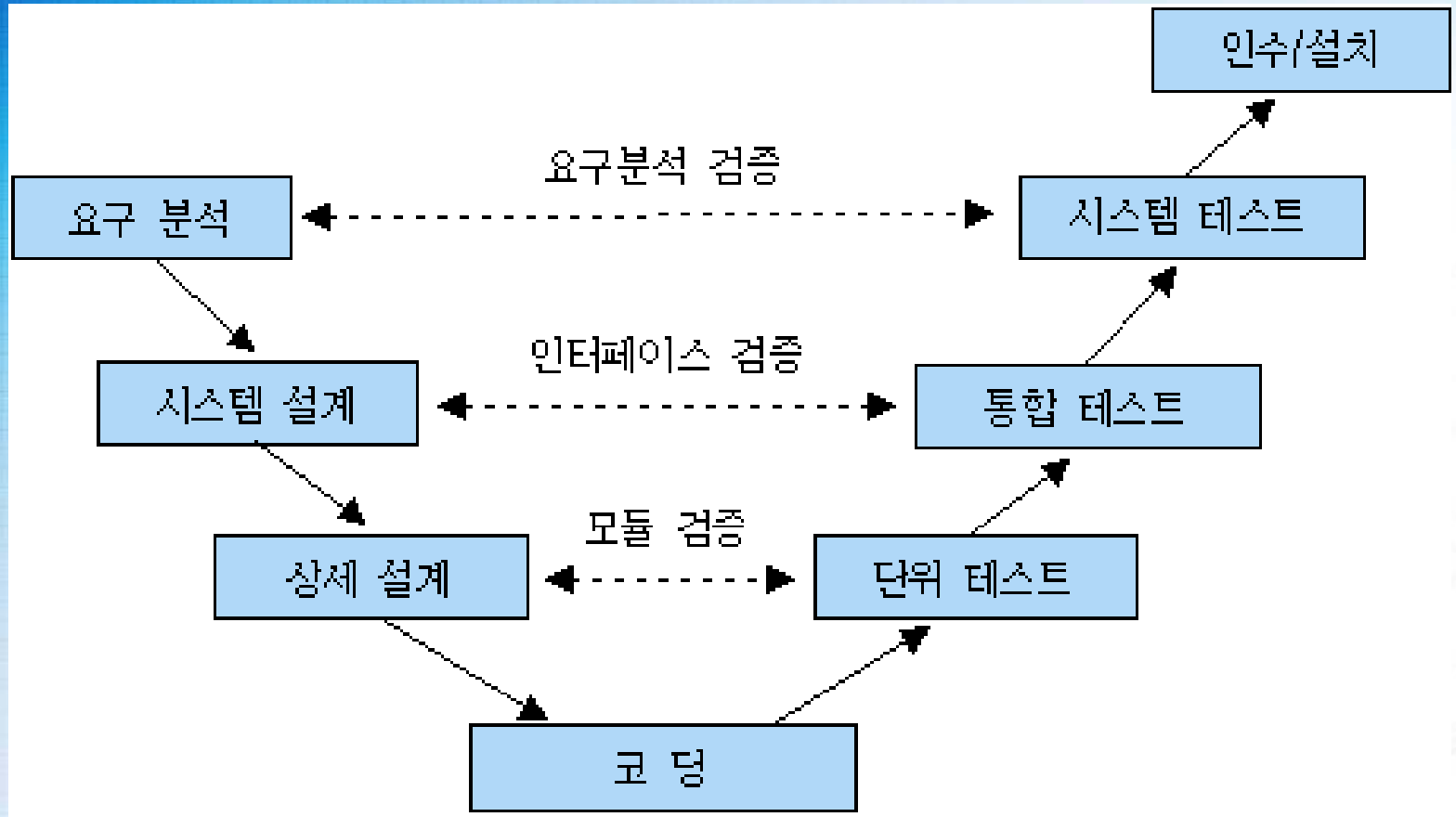
❑ 단점

- ❍ 관리가 중요
- ❍ 위험 분석이 중요
- ❍ 새로운 모형

❑ 적용

- ❍ 재정적 또는 기술적으로 위험 부담이 큰 경우
- ❍ 요구 사항이나 아키텍처 이해에 어려운 경우

V 모형


인수/설치
요구분석 검증
요구 분석
시스템 테스트
인터페이스 검증
시스템 설계
통합 테스트
모듈 검증
상세 설계
단위 테스트
코 딩

V 모형

❑ 폭포수 모형의 변형

- ❍ 감추어진 반복과 재 작업을 드러냄
- ❍ 작업과 결과의 검증에 초점

❑ 장점

- ❍ 오류를 줄일 수 있음

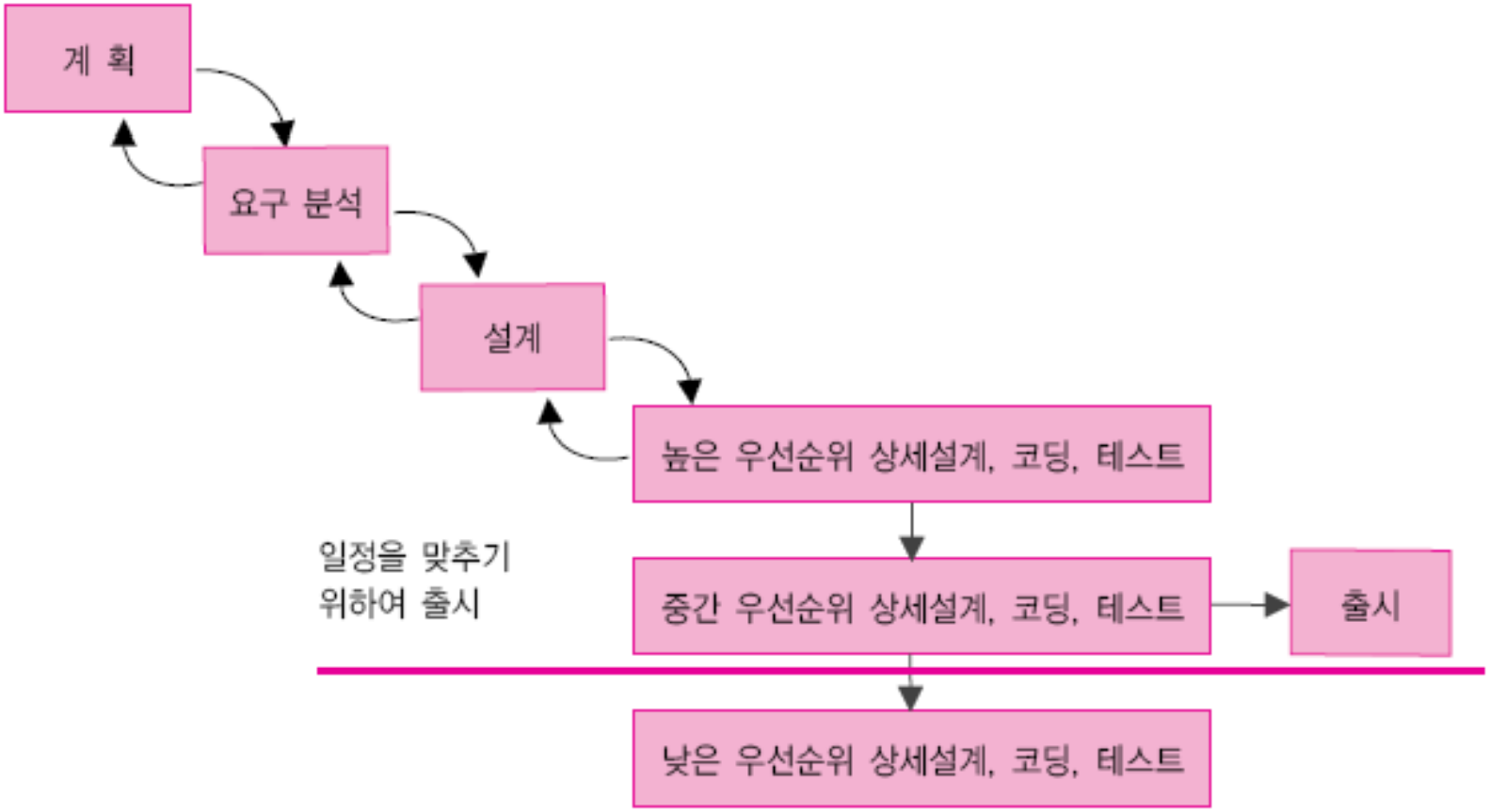
❑ 단점

- ❍ 반복이 없어 변경을 다루기가 쉽지 않음

❑ 적용

- ❍ 신뢰성이 높이 요구되는 분야

일정 중심 설계 모형


계 획
요구 분석
설계
높은 우선순위 상세설계, 코딩, 테스트
일정을 맞추기
위하여 출시
중간 우선순위 상세설계, 코딩, 테스트
출시
낮은 우선순위 상세설계, 코딩, 테스트

일정 중심 설계 모형

❑ 특징

- ❍ 사용자의 요구에 대하여 우선순위를 정하고 이를 기초로 각 사이클을 계획
- ❍ 초기 단계에 중요한 기능들을 설계, 구현하여 시스템의 골격을 만듦
- ❍ 상대적으로 덜 중요한 기능을 나중에 함으로 일정 조정 가능

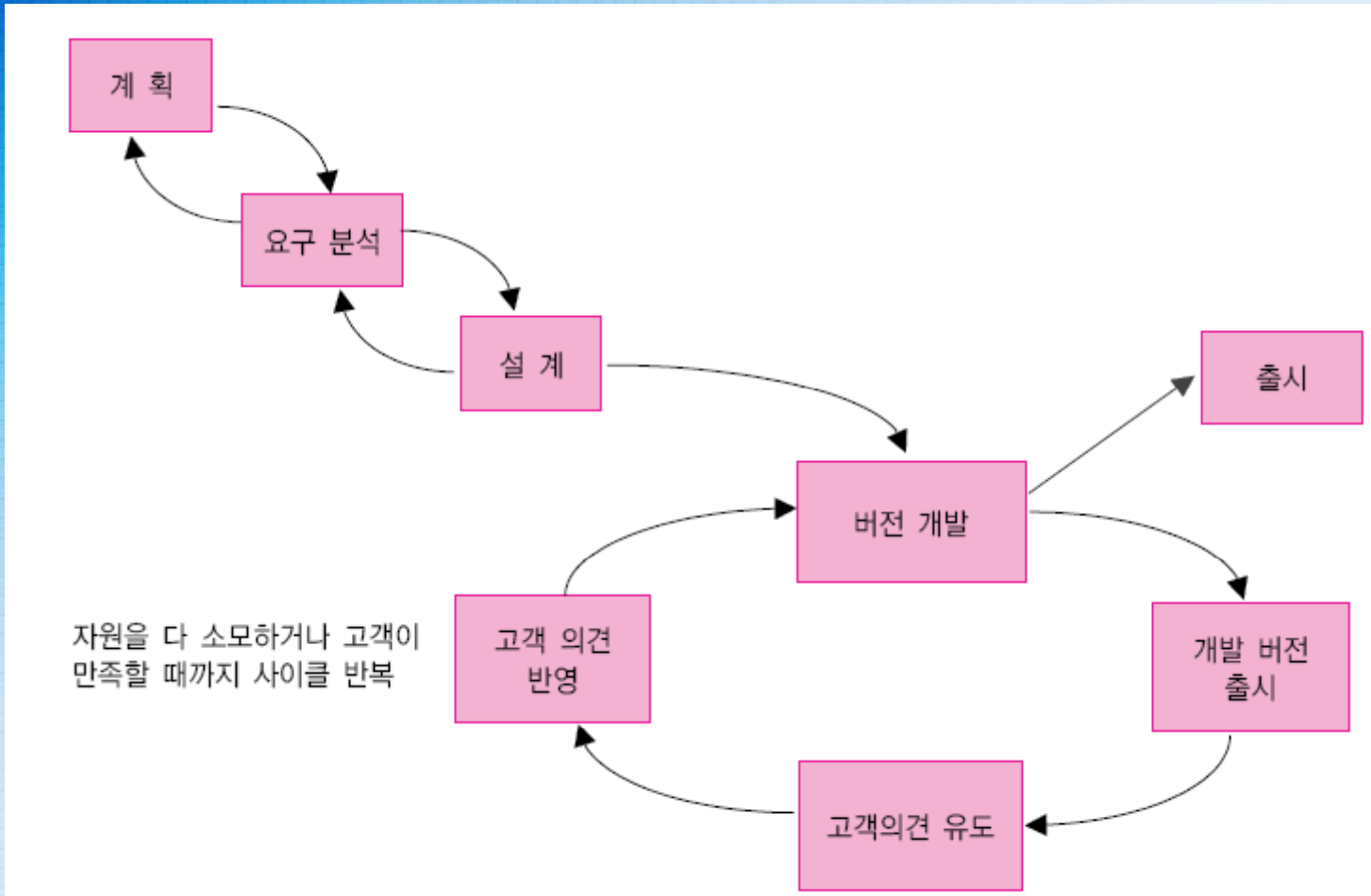
❑ 단점

- ❍ 우선순위가 낮아 출시에 포함되지 않을 기능을 분석 하고 설계하는 데 시간을 낭비

❑ 적용

- ❍ 소프트웨어 제품의 출시 날짜가 매우 중요한 경우
- ❍ 목표 일정을 달성할 수 있을지 불확실할 때

진화적 출시 모형


계 획
요구 분석
설 계
버전 개발
출시
고객 의견
반영
개발 버전
출시
고객의견 유도
자원을 다 소모하거나 고객이
만족할 때까지 사이클 반복

진화적 출시 모형

❑ 진화적 출시(evolutionary delivery)

- ❍ 고객의 요구를 여러 사이클에 걸쳐 개발하여 보여주면서 제품을 개선해 나가는 모형

❑ 프로토타이핑 모형과 다른 점

- ❍ 고객의 요구를 프로토타이핑 모형처럼 전적으로 수용하지는 않음
- ❍ 고객의 반응으로 바뀔 가능성이 적은 부분이 시스템의 핵심
- ❍ 프로토타이핑 모형은 시스템에서 눈에 띄는 부분을 먼저 강조하고 나중에 시스템 기반에 있는 구멍을 메워나가는 식

모델이 많은 이유

- 모델의 선택은 프로젝트의 상황과 요구에 좌우됨
- 모델을 잘 선택하면 생산성이 높아짐
- 경우에 따라서는 최적의 효과를 내기 위하여 모델의 융합이 이루어지기도 함

모델 비교와 선택

[표 1.1] 소프트웨어 생명주기 모형의 장단점

모형의 비교항목	폭포수	프로토타이핑	점증적 모형	나선형 모형	V-모형	일정중심 설계	진화적 출시
• 요구사항 애매한 경우의 적합성	중	상	하	상	중	중	상
• 낯선 아키텍처 구현 적합성	중	상	하	상	중	하	하
• 높은 안정성의 개발 능력	상	중	상	상	상	중	상
• 시스템 확장용이	상	상	상	상	상	중	상
• 위험 관리 능력	중	중	중	상	중	중	중
• 일정을 맞출 능력	중	하	중	중	중	상	중
• 업무 부하 낮음	상	중	중	중	상	중	중
• 중도 방향 수정성	하	상	하	중	하	중	상
• 고객에 보이는 진행 상황 가시성	중	상	중	상	중	중	상
• 관리자에게 보이는 진행 상황 가시성	상	중	상	상	상	상	상
• 사용을 위하여 특수 지식 필요 없음	상	하	중	하	상	하	중

프로세스 모델에 관한 시험 문제

다음과 같은 시스템 개발에 가장 적합한 소프트웨어 개발 모형을 제안하고 그 이유를 쓰시오.

- 자동차의 Anti-lock Braking을 제어하기 위한 소프트웨어
- 병원에서 현재 사용하는 회계 시스템을 대체하기 위한 새 시스템
- 항공기 승객이 비행기를 놓쳤을 경우 빨리 대체 항공기 편을 찾을 수 있게 하는 대화형 시스템