1강 소프트웨어공학 개요 (1)

컴퓨터과학과 김희천 교수

1장. 소프트웨어 공학 개요(1)

소프트웨어 공학 개요

소프트웨어의 중요성

- 컴퓨터 시스템에 대한 의존도 증가

소프트웨어 개발에 필요한 지식

- 프로그래밍 외에 설계/시험 방법, 응용 분야, 인간공학, 경영 등에 관한 지식이 필요

소프트웨어 개발과 관리

- 설계와 구현 과정에서 도구와 방법론을 적용
- 프로젝트 계획과 관리 기술이 필요

1.1 소프트웨어와 시스템

소프트웨어

소프트웨어의 포괄적 의미

- 프로그램 + 프로그램의 개발, 운용, 보수에 필요한 정보 일체

소프트웨어의 성질

- 비가시성(Invisibility)
  - 무형, 관리가 힘듦
- 테스트 가능(Testability)
- 복잡성(Complexity)
  - 규칙적이고 정형적인 구조가 없음

소프트웨어의 특성

소프트웨어의 성질(계속)

- 변경성(Changeability)
  - 하드웨어에 비해 수정이 용이
  - 순응성(Conformity)
- 장수(Longevity)
  - 마모되지 않음
- 복제 가능(Duplicability)

1.1 소프트웨어와 시스템

소프트웨어의 분류

- 기능에 따른 분류
  - 응용 소프트웨어
  - 시스템 소프트웨어
- 성격에 따른 분류
  - 프로토타입
  - 연구 생산물
  - 패키지
    - 다수 사용자용, generic S/W
  - 주문형 소프트웨어
    - 특정 사용자용, custom S/W
- 응용 분야에 따른 분류

1.1 소프트웨어와 시스템

시스템

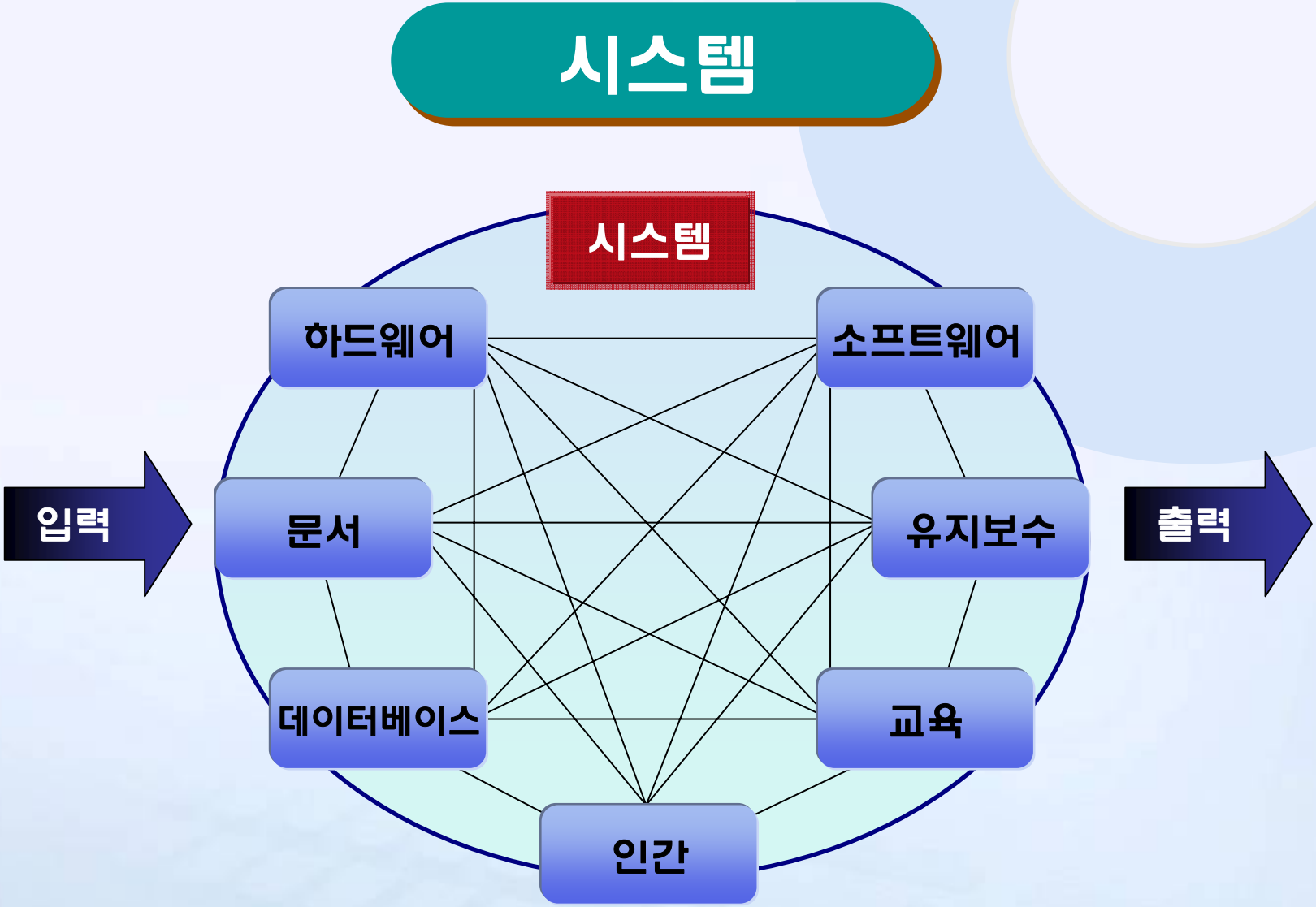
시스템의 의미

- 유기적으로 상호 작용하는 구성요소들의 집합
- 컴퓨터 시스템은 하드웨어, 소프트웨어, 사용자, 운영자 및 주변 환경으로 구성됨
- 소프트웨어 자체도 하나의 시스템

시스템의 특성

- 시너지 효과
- 변화에 적응해야 함
- 상충되는 요구와 이해 관계의 절충안

1.1 소프트웨어와 시스템

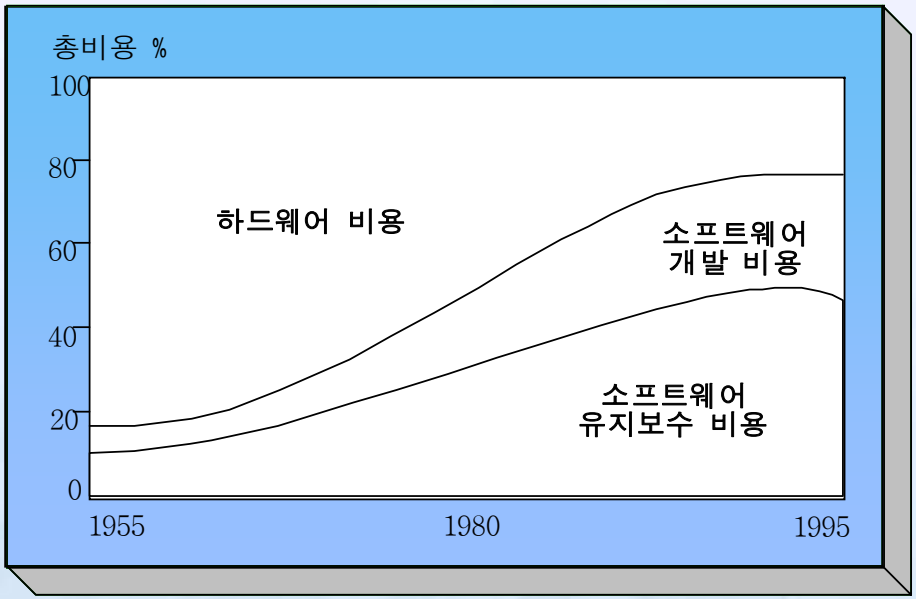


1.2 소프트웨어의 위기(Crisis)

소프트웨어 생산성의 문제

소프트웨어 생산성의 문제

- 소프트웨어 수요의 증가에 비해 공급 능력이 부족
- 소프트웨어 비용의 상대적 증가
- 하드웨어 비용 하락, 인건비 상승



소프트웨어의 위기(Crisis)

소프트웨어 공정의 문제점

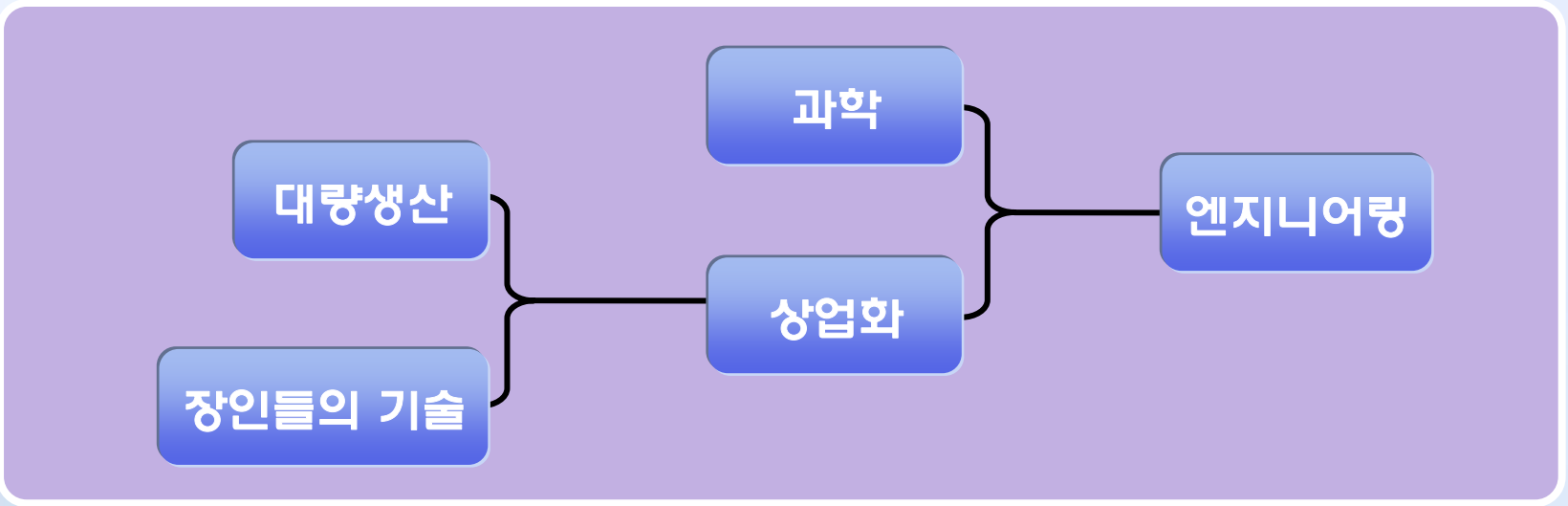
- 생산성 문제
- 부정확한 개발 계획
  - 비용 초과, 기간 지연
- 생산 기술과 관리 기술의 부족
  - 개인의 역량에 의존해서는 안됨
- 품질의 문제
  - 신뢰성과 성능이 부족
- 유지보수 문제
  - 엄청난 유지보수 비용

1.3 소프트웨어 공학

공학이란?

- 현실의 문제 해결을 위해 합리적이고 일반적인 해결 방법을 탐구하는 학문
- 제조물의 설계와 구현을 중요시

엔지니어링의 발전 원리



1.3 소프트웨어 공학

소프트웨어 공학

정의

- 질 좋은 소프트웨어를 경제적으로 생산하기 위하여 공학, 과학 및 수학적 원리와 방법을 적용하는 것
  - Watts Humphrey, SEI
- 소프트웨어의 개발, 운용, 유지보수 및 소멸에 대한 체계적인 접근 방법
  - IEEE Computer Society
- 품질, 효율, 비용, 인정에 관한 공학적인 접근 원리
  - F. Brooks

목표

- 품질(Quality)
- 생산성(Productivity)
  - 최소의 비용으로 주어진 기간 안에 개발

1.3 소프트웨어 공학

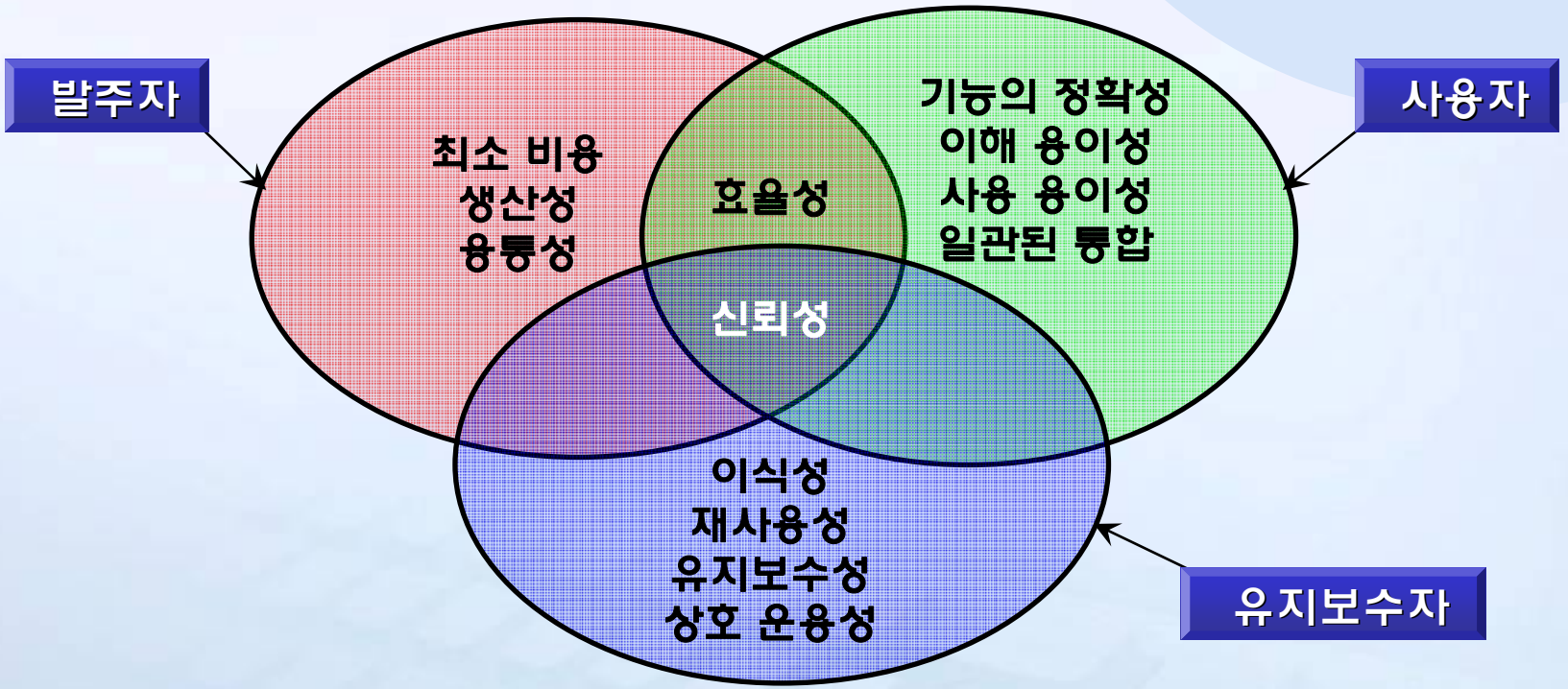
소프트웨어 공학이 다루는 주제

분 야	의 미	사 례	요리 비유
방법(method)	소프트웨어 제작에 사용하는 기법이나 절차	구조적 분석, 설계 방법 객체지향 분석, 설계 방법	익히는 방법 (구이, 찜, 훈제 등)
도구(tool)	자동화된 시스템	설계 도구 프로그래밍 도구 테스트 도구	요리 도구 (프라이팬, 압력 솥, 오븐 등)
프로세스 (process)	도구와 기법을 사용하여 작업하는 순서	Unified Process eXtreme Programming	조리 순서(recipe)
패러다임 (paradigm)	사물이나 현상을 바라보는 시각이나 철학	S/W 개발 패러다임 –절차적 패러다임 –객체지향 패러다임	음식 스타일 (한식, 일식, 중식, 퓨전 등)

1.4 좋은 소프트웨어의 조건

고품질

- 소프트웨어 공학의 목표 중 하나
- 소프트웨어를 대하는 입장에 따라 품질에 대한 관점이 달라짐



1.4 좋은 소프트웨어의 조건

소프트웨어 품질(Quality)

정확성(Correctness)

- 기능적으로 맞게 동작, 표준에 적합
- 요구 분석서의 기능과 일치하는지 점검

신뢰성(Reliability)

- 소프트웨어가 주어진 기간 동안 제대로 작동할 확률
- 오작동의 빈도와 관련 있음
- 정확성을 위한 필요조건

강인성(Robustness)

- 요구 명세에 표시하지 않은 상황(오류 입력)에서도 제대로 작동하는 성질

1.4 좋은 소프트웨어의 조건

소프트웨어 품질(Quality)

성능(Performance)

- 수행 속도
- 알고리즘의 시간 복잡도
- 시뮬레이션, 스트레스 테스트

사용 용이성(Usability)

- 사용자가 시스템을 사용하기 쉬워야 함
- 사용자 인터페이스

유지보수성(Maintainability)

- 변화하는 요구를 쉽게 수용하도록 설계되어야 함
- 소프트웨어 결함을 쉽게 수정할 수 있어야 함
- 진화성

1.4 좋은 소프트웨어의 조건

1.4 좋은 소프트웨어의 조건

프로세스 품질

품질의 종류

- 프로덕트 품질과 프로세스 품질

프로세스 품질

- 개발 과정이 품질에 많은 영향을 준다는 주장
- 프로세스 개선 모델
  - CMM, SPICE

향상 방안

- 특정 오류가 언제 어디서 발견되는가?
- 어떻게 하면 개발 과정에 오류를 조기에 발견할 수 있는가?
- 어떻게 하면 오류에 대한 내성이 있는 시스템, 즉 오류가 소프트웨어를 정지시키는 확률이 적은 시스템으로 만들 수 있는가?
- 프로세스가 좋은 품질을 보장하는 더 효율적이며 효과적인 다른 방법이 있는가?

1강 소프트웨어공학 개요 (1)

컴퓨터과학과 김희천 교수

학습2로 넘어가세요

1강 소프트웨어공학 개요 (2)

컴퓨터과학과 김희천 교수

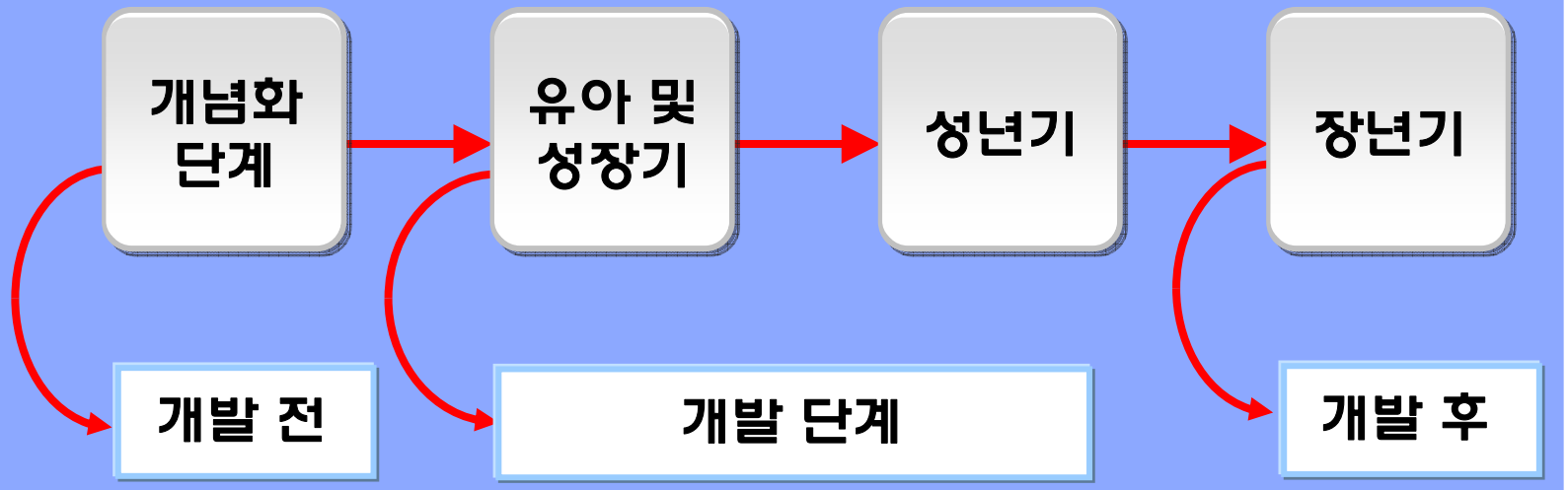
1장. 소프트웨어 공학 개요(2)

1.5 소프트웨어 생명주기 모형

소프트웨어 생명주기

소프트웨어 생명주기

- 소프트웨어 개발의 시작에서 폐기까지의 일련의 개발 단계
  - 설계 단계나 그 이전 단계를 중요시 함



1.5 소프트웨어 생명주기 모형

프로세스 모형

개발 프로세스

개발 모형

- 실제 소프트웨어 프로세스를 단순화, 추상화시킨 것
- 개발 단계별로 수행 할 작업들과 중간 결과물 및 다음 단계 진행을 위한 기준을 정의함
- 소프트웨어 공학자가 어떤 업무를 어떤 순서로 할 것인가에 관한 지침 제공, 실정에 맞는 개발 팀의 고유한 모형의 정립 필요

일반적 소프트웨어 개발 단계

- 요구 분석과 정의
- 시스템 설계
- 프로그램 설계
- 프로그램의 작성(구현)
- 테스트(단위 테스트, 통합 테스트, 시스템 테스트)
- 시스템 설치
- 유지보수

1.5 소프트웨어 생명주기 모형

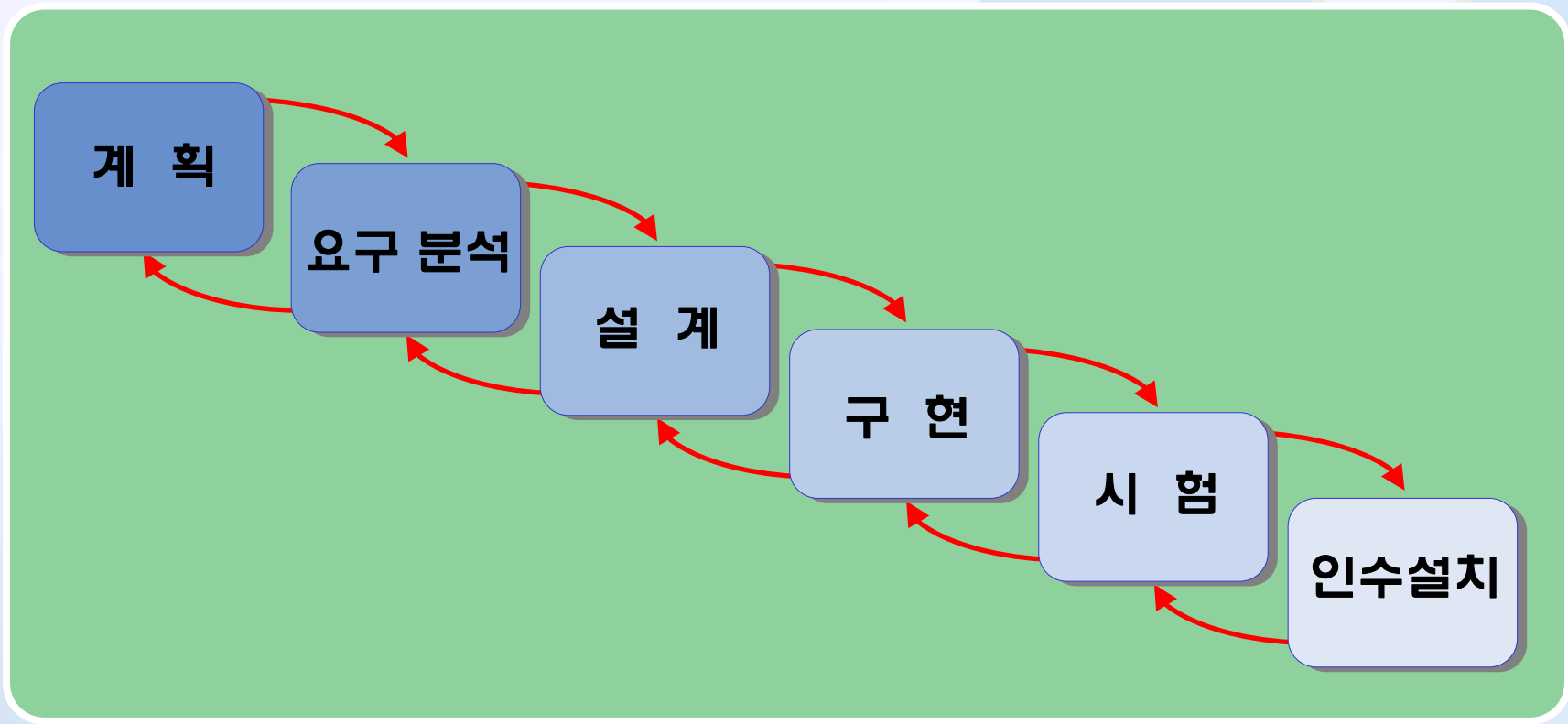
대표적인 생명주기 모형

중요한 모형

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

1.5 소프트웨어 생명주기 모형

폭포수 (waterfall)모형



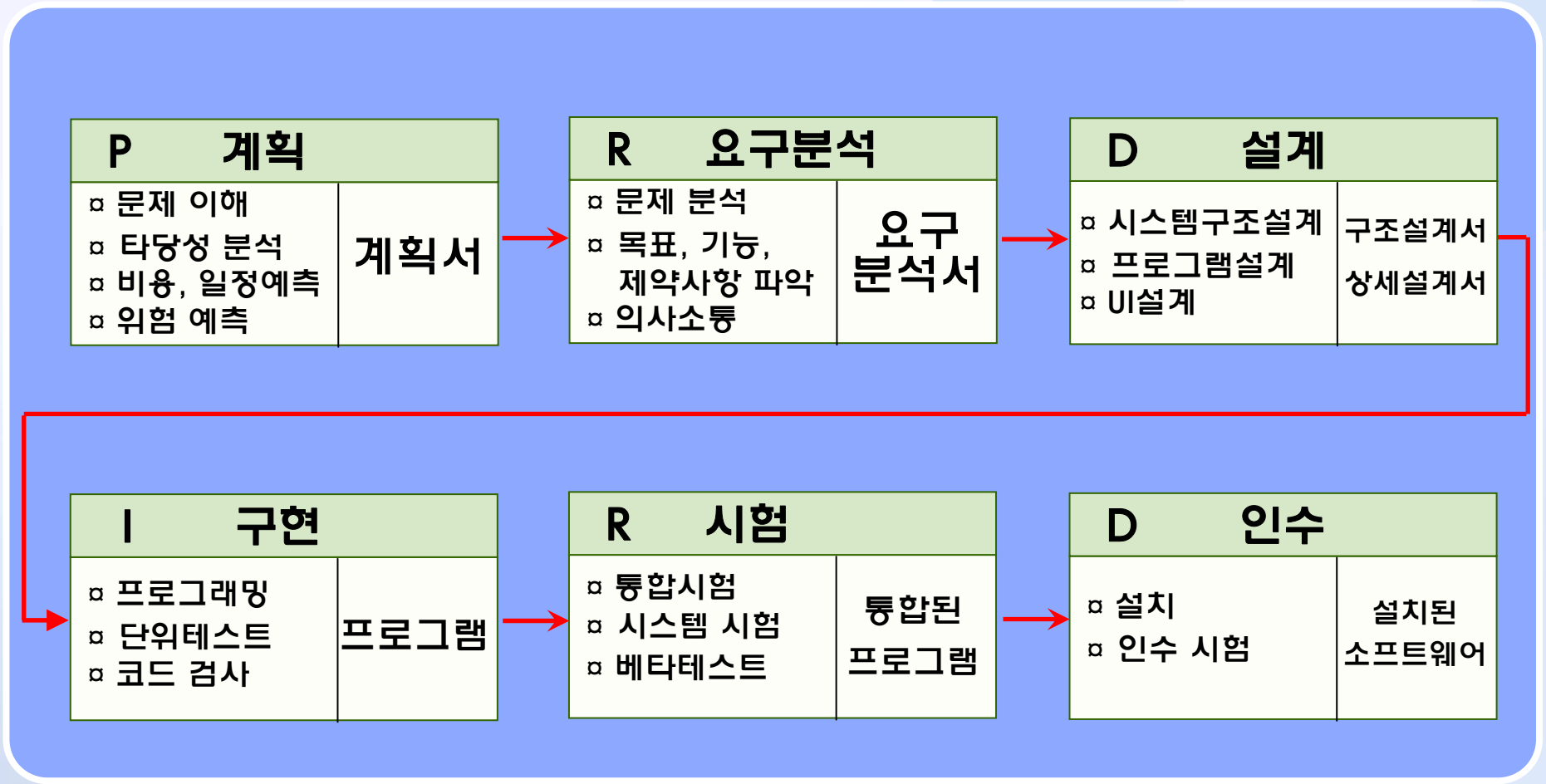
1.5 소프트웨어 생명주기 모형

폭포수 (waterfall)모형

- 최초 프로세스 모형, 1970년대 소개
  - 다른 공학에서 사용하는 프로세스로부터 유도된 것
  - 단계별 작업이 분리되어 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 전 단계로 피드백
- 결과물 정의가 중요
- Method와 Methodology

1.5 소프트웨어 생명주기 모형

폭포수 모형의 프로세스와 결과물



1.5 소프트웨어 생명주기 모형

폭포수 모형의 장단점

장점

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

단점

- 처음 단계를 지나치게 강조하여 코딩, 테스트가 지연
  - 불필요한 다종의 문서를 생산할 가능성 있음
- 프로토타입과 재사용의 기회가 줄어듦
- 변화하는 고객의 요구에 대응하기 어려움
  - 실제 프로세스는 작업의 병렬 수행과 반복이 존재

적용

- 요구사항이 명확한 경우, 응용 분야를 잘 알고 있는 경우, 연구 중심 문제 등에 적합
- 변화가 적고 위험이 적은 프로젝트에 적합

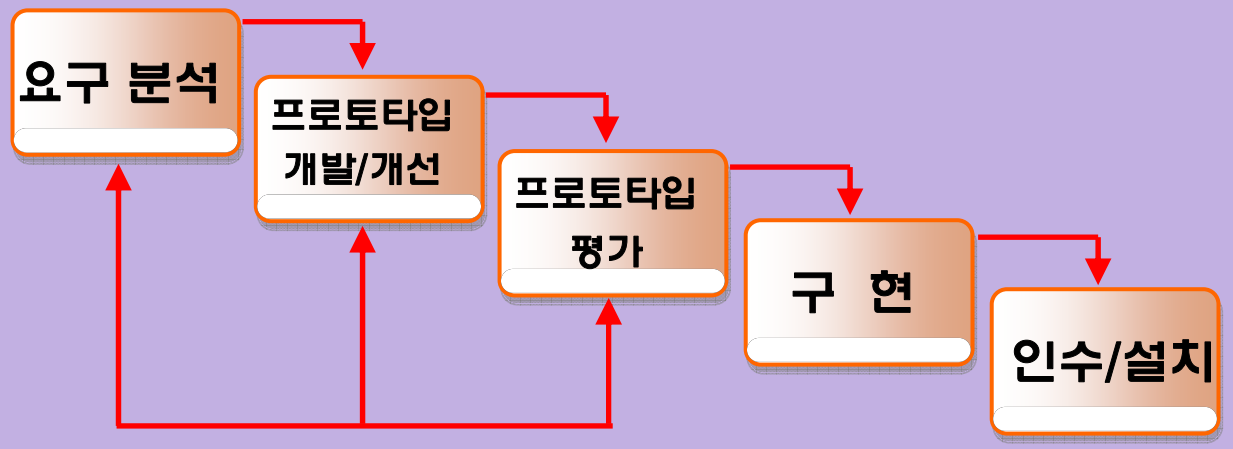
1.5 소프트웨어 생명주기 모형

프로토 타이핑 모형

기본 아이디어

- 초기 구현을 사용자에게 보여주고, 피드백을 통해 시스템을 점진적으로 완성

Rapid Prototyping Model (RAD)



1.5 소프트웨어 생명주기 모형

프로토 타이핑 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

두 가지 프로토타입

- 단순한 요구 추출 – 만들고 버림
- 진화적 프로토타입 – 최종 결과물로 발전

프로토 타이핑 모형의 장단점

장점

- 일찍 시스템의 모습을 볼 수 있어 기능성과 유용성을 확인
- 발주자와 개발자 간 공통의 참조 모델을 제공
- 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

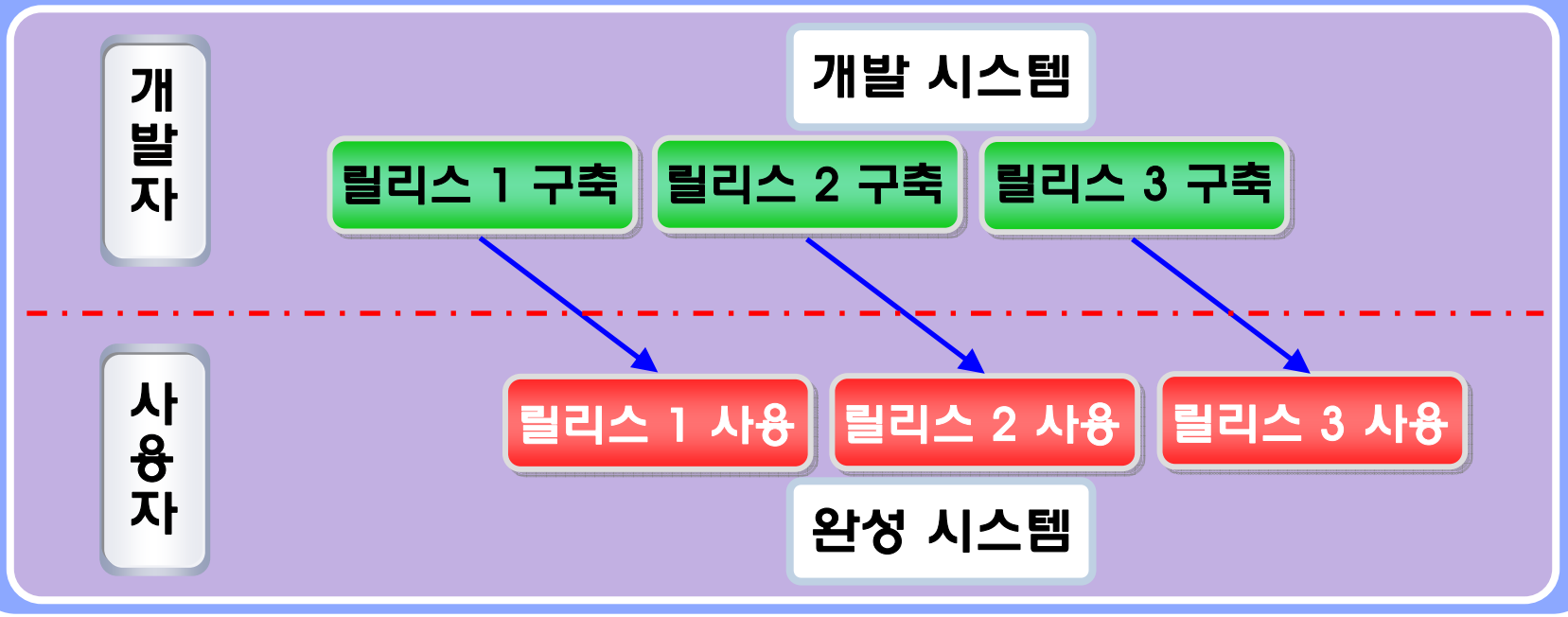
- 개발 착수 시점에 요구가 불투명할 때
- 실험적으로 실현 가능성을 타진해 보고 싶을 때
- 혁신적인 기술을 사용해 보고 싶을 때

1.5 소프트웨어 생명주기 모형

점증적 모형

장점

- 최근의 비즈니스 환경은 빠른 개발을 원함
- 개발 시간을 줄이는 법 – 시스템을 나누어 릴리스
  – 개발과 운용이 혼재함



1.5 소프트웨어 생명주기 모형

점증적 모형

점증적 개발 방법

- 시스템을 기능별로 여러 서브 시스템으로 나누고 하나씩 개발 (릴리스A, 릴리스B, 릴리스C)

반복적 개발 방법

- 시스템 전체 기능을 대상으로 하며 릴리스를 개발할 때 완성도를 높임 (릴리스1.0, 릴리스1.1, 릴리스1.2)

단계적 개발 (점증적 방법과 반복적 방법의 병행)

- 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

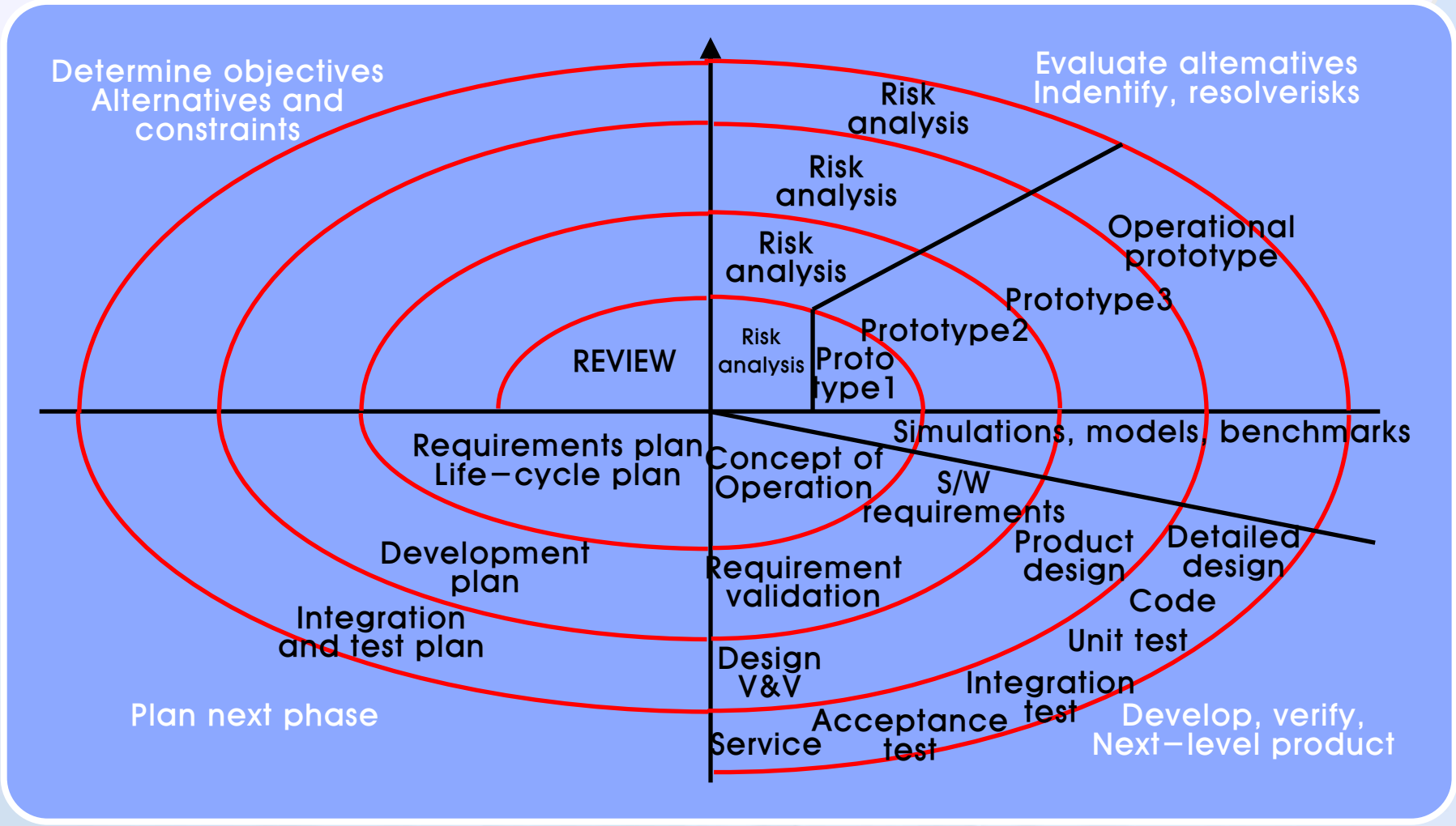
나선형(spiral) 모형

Boehm의 모델

- 위험 관리 측면을 중요시 함
- 반복적 개발 방법과 유사
  - 나선형
  - 가장 안쪽 나선은 타당성 조사, 그 다음은 요구 정의, 그 다음은 설계 …
  - 매 나선마다 계획과 검토 후 위험 요소 분석이 뒤따름
- 관점에 따라
  - 중요 기능을 먼저 개발하고 다른 기능을 추가한다면 여러 번의 점증적인 릴리스로 볼 수 있음

1.5 소프트웨어 생명주기 모형

나선형(spiral) 모형



1.5 소프트웨어 생명주기 모형

나선형(spiral) 모형

소프트웨어의 기능을 나누어 점증적으로 개발

- 실패의 위험을 줄임
- 테스트 용이
- 피드백

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

1.5 소프트웨어 생명주기 모형

나선형(spiral) 모형의 장단점

장점

- 대규모 시스템 개발에 적합
  - 위험을 관리하고 최소화 함
- 반복적인 개발 및 테스트를 통한 강인성 향상
  - 한 사이클에 추가 못한 기능은 다음 단계에 추가 가능

단점

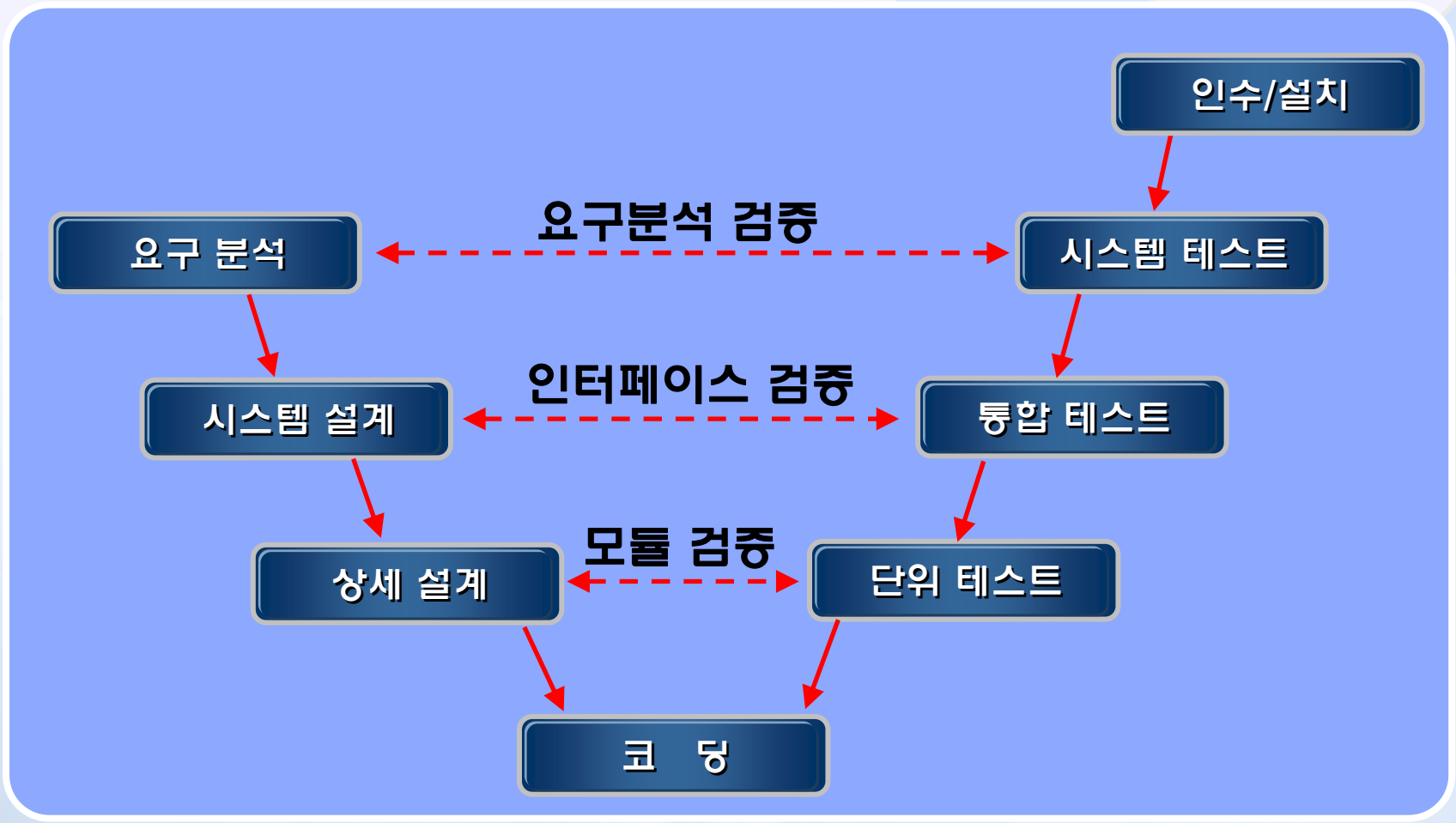
- 폭포수 모델이나 프로토타이핑 모델보다 복잡하여 프로젝트 관리가 어려움
- 다수 고객을 상대하는 상업용 제품인 경우 적용이 어려움
- 새로운 모형으로 많이 사용하지 않아 충분히 검증이 안됨

적용

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

1.5 소프트웨어 생명주기 모형

V모형



1.5 소프트웨어 생명주기 모형

V모형

폭포수 모형의 변형

- 감추어진 반복과 재 작업을 드러냄
  - 점선은 오류 발생시 되돌아 갈 수 있음을 의미
- 작업 결과의 검증과 테스트 작업을 강조함
- 상세 설계는 단위 테스트, 시스템 설계는 통합 테스트, 요구는 시스템 테스트 과정에서 검증함

장점

- 오류를 줄일 수 있음

단점

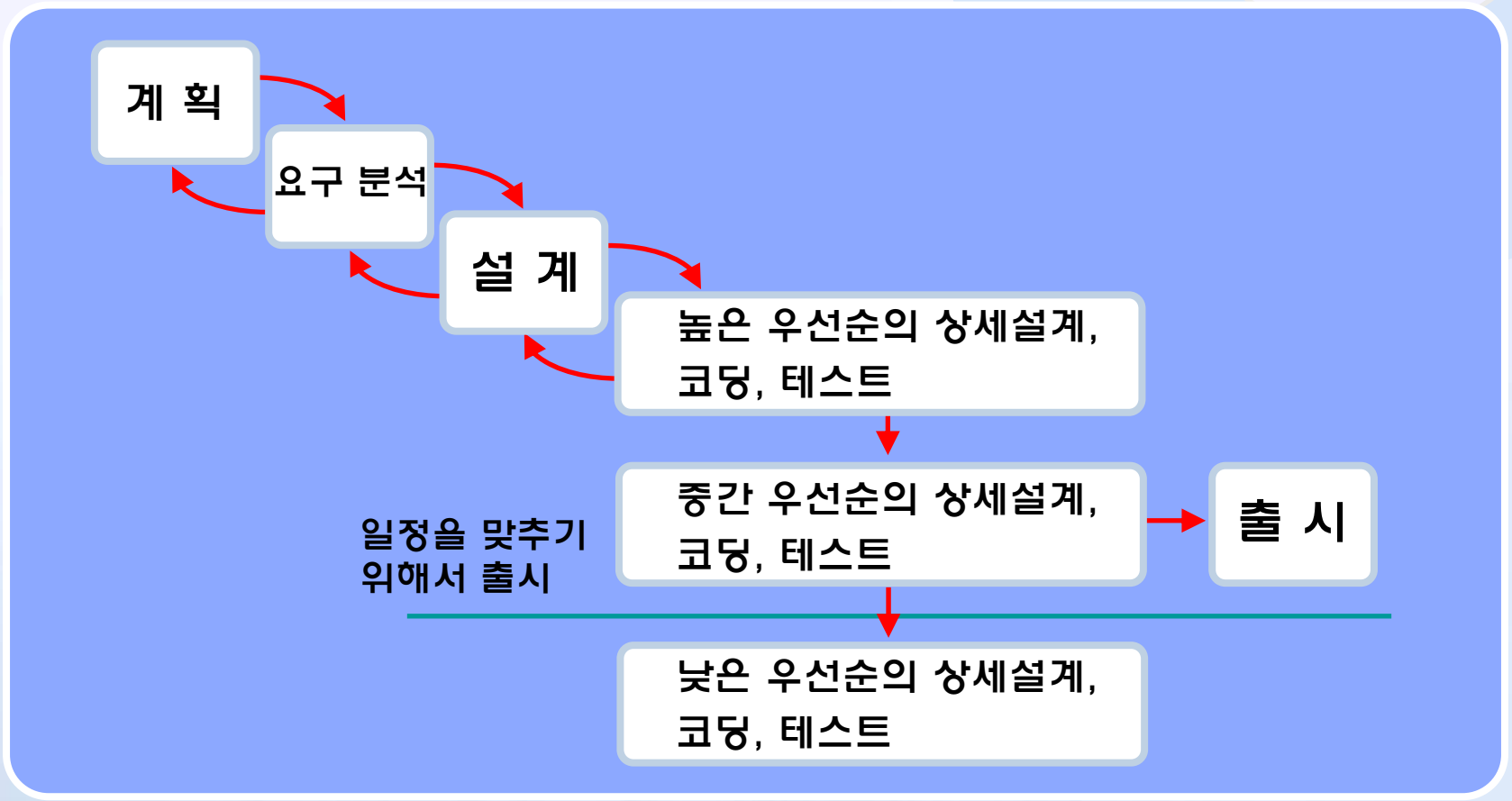
- 반복이 없어 변경을 다루기가 쉽지 않음

적용

- 신뢰성이 높이 요구되는 분야

1.5 소프트웨어 생명주기 모형

일정 중심 설계 모형



1.5 소프트웨어 생명주기 모형

일정 중심 설계 모형

특징

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

단점

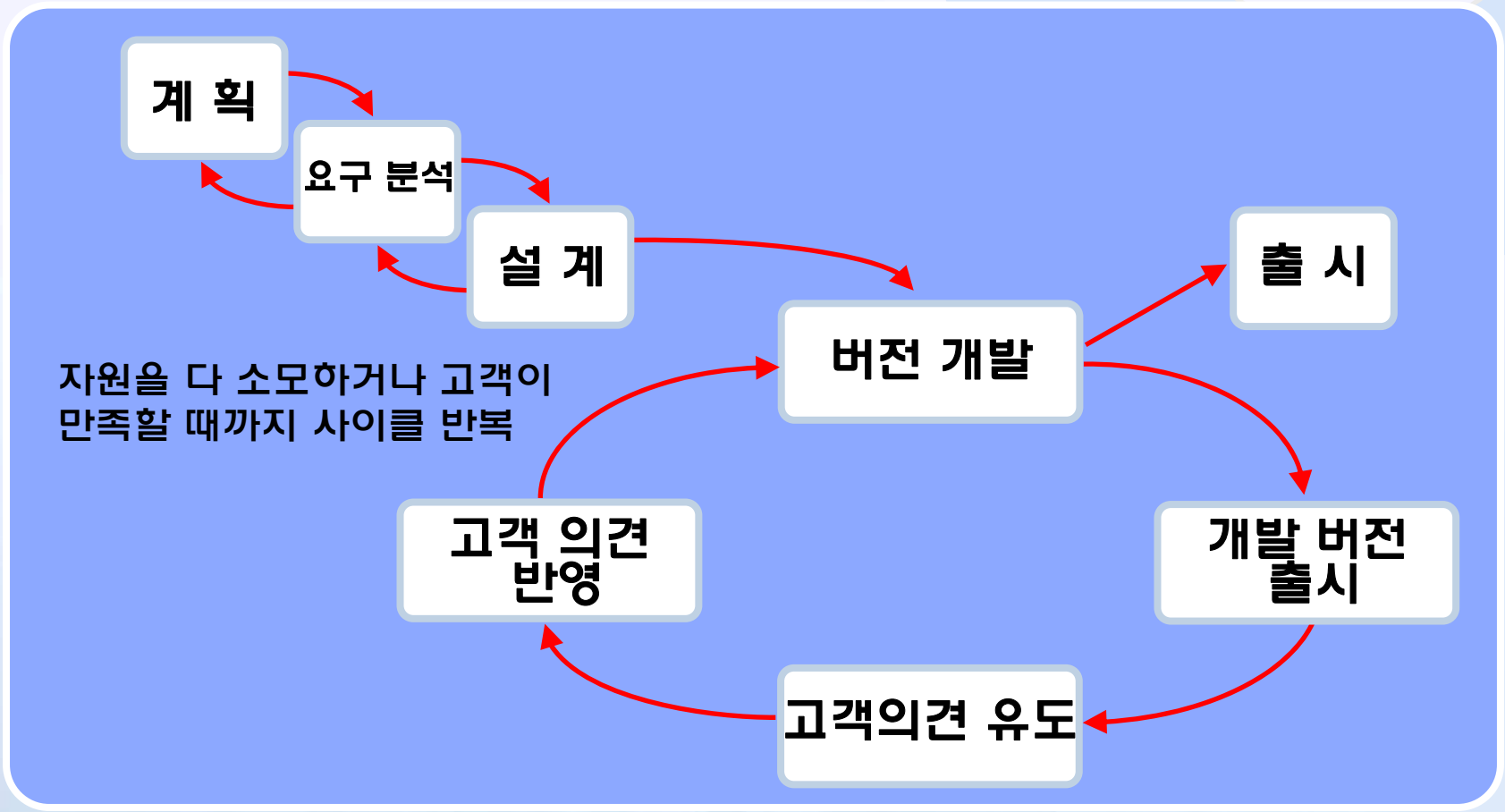
- 우선순위가 낮아 출시에 포함되지 않을 기능을 분석 하고 설계하는 데 시간을 낭비

적용

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

1.5 소프트웨어 생명주기 모형

진화적 출시 모형



1.5 소프트웨어 생명주기 모형

진화적 출시 모형

진화적 출시(evolutionary delivery)

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

프로토타이핑 모형과 다른 점

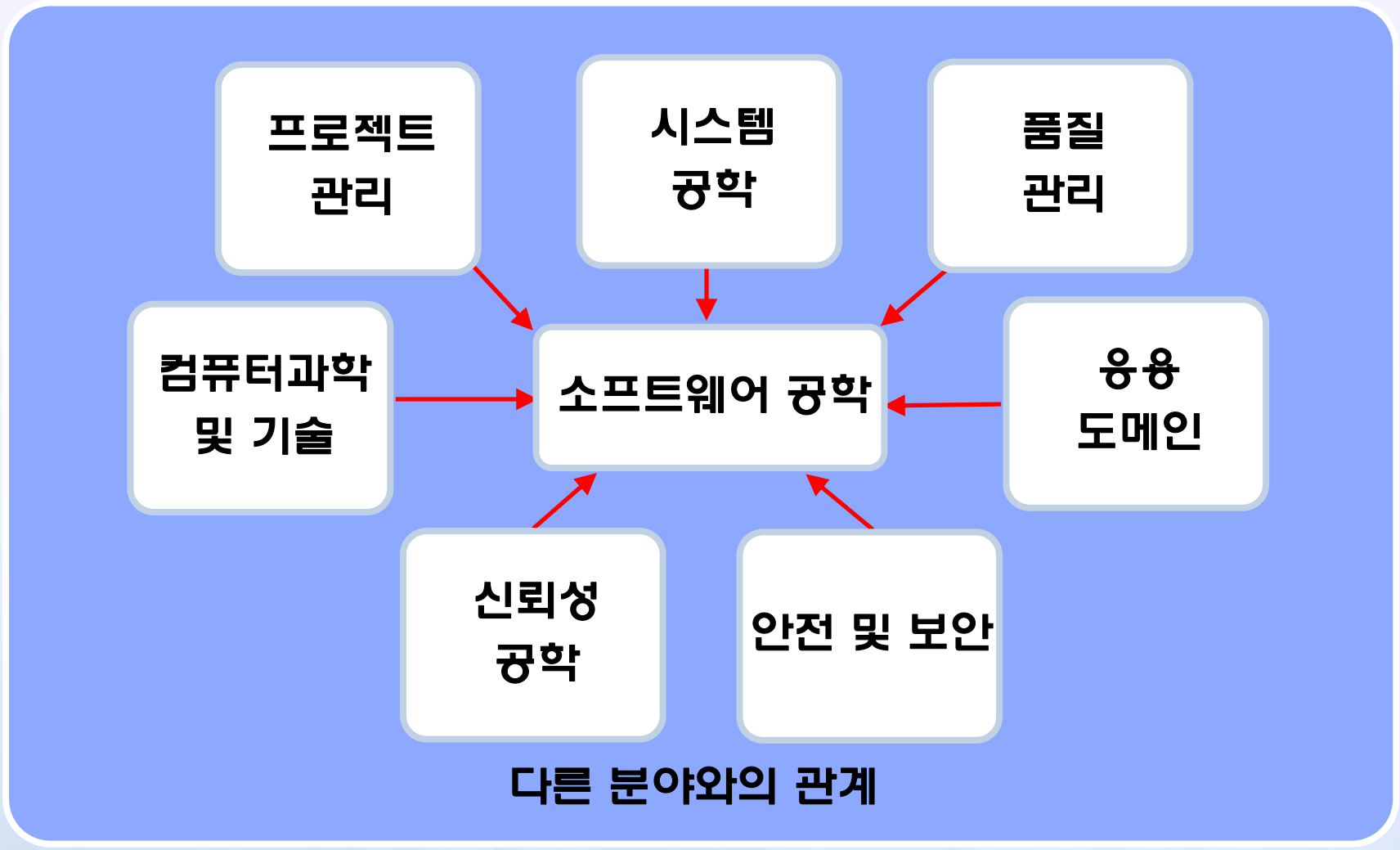
- 고객의 요구를 프로토타이핑 모형처럼 전적으로 수용하지는 않음
- 고객의 반응으로 바뀔 가능성이 적은 부분이 시스템의 핵심
- 프로토타이핑 모형은 시스템에서 눈에 띄는 부분을 먼저 강조하고 나중에 시스템 기반에 있는 구멍을 메워나가는 식

1.5 소프트웨어 생명주기 모형

모델 비교와 선택

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

1.6 소프트웨어 공학 근본 지식



다른 분야와의 관계

1.6 소프트웨어 공학 근본 지식

SEWBOK

- 소프트웨어 요구분석
  - 사용자의 요구의 추출,분석,검증,관리에 관한 지식(3장, 5.4절, 6장)
- 소프트웨어 설계
  - 사용자가 원하는 요구를 만족시킬 수 있는 솔루션에 해당하는 설계를 작성하는 데 필요한 지식(4장, 6장)
- 소프트웨어 구축
- 소프트웨어 테스트
  - 테스트에 관한 기본 개념,수준,측정,과정,도구에 관한 지식(8장)
- 소프트웨어 유지보수
  - 유지보수 개념과 작업 및 관리,비용,측정에 관한 지식(9장)

1.6 소프트웨어 공학 근본 지식

SEWBOK

소프트웨어 형상 관리

- 시스템을 이루고 있는 구성요소들을 잘 파악하고 이들의 변경, 릴리스를 잘 관리하는데 필요한 지식(9.2절)

소프트웨어 엔지니어링 관리

- 소프트웨어 프로젝트의 계획,실행,평가,조정 등의 관리에 대한 지식과 객관적인 측정에 관한 지식(2장)

소프트웨어 엔지니어링 프로세스

- 소프트웨어 프로세스의 정의,구현, 측정,관리,변경,개선에 관한 지식 (1장,2장)

소프트웨어 엔지니어링 도구와 방법

- 소프트웨어 개발 방법 및 도구와 컴포넌트 통합에 관한 지식(5장, 11장)

1.6 소프트웨어 공학 근본 지식

SEWBOK

소프트웨어 품질

- 프로덕트 품질의 개념과 품질 특성, 품질 보증을 위한 계획, 활동에 관한 지식(10장)

기타 관련 지식

1강 소프트웨어공학 개요 (2)

컴퓨터과학과 김희천 교수

다음강의 2강 계획