IT CookBook, 쉽게 배우는 소프트웨어 공학

[강의교안 이용 안내]

한빛아카데미 Hanbit Academy, Inc.

쉽게 배우는

소프트웨어 공학

Chatpter 02
소프트웨어 개발 프로세스

- 소프트웨어 개발 프로세스의 개념을 이해한다.
- 소프트웨어 프로세스 모델의 종류를 알아본다.
- 주요 프로세스 모델에 대해 자세히 살펴본다.

Section 01 소프트웨어 개발 프로세스의 이해

1. 일상에서의 프로세스 의미

■ 프로세스

- 일을 처리하는 과정 또는 순서

(예 1) 공장에서 자동차, 세탁기 등이 조립되어 완제품이 되는 과정

(예 2) TV요리 프로에서 요리사가 맛있는 요리를 만드는 과정

그림 2-1 요리 프로세스

2. 프로세스의 정의

- **프로세스**
  - 일이 처리되는 과정이나 공정

    즉, 주어진 일을 해결하기 위한 목적으로 그 순서가 정해져 수행되는 일련의 절차

- **프로세스를 따랐을 때의 효과의 예**
  - 요리 레시피 활용하면?
  - 세탁기 사용설명서 활용하면?
  - 화면 지시에 따라 OS 설치하면?

  → 목적 달성

3. 소프트웨어 개발 프로세스(1)

소프트웨어 개발에서의 프로세스

- 작업(task)순서의 집합 + 제약 조건(일정, 예산, 자원)을 포함하는 일련의 활동(activity)
  - 작업(task): SW를 개발할 때 일을 수행하는 작은 단위

(좁은 의미)소프트웨어 개발 프로세스

- SW제품을 개발할 때 필요한 절차, 과정, 구조
- 사용자의 요구사항을 SW시스템으로 구현하기 위한 일련의 활동

(넓은 의미)소프트웨어 개발 프로세스

- 절차, 구조, 방법, 도구, 참여자까지 모두 포함
- SW개발 목적을 이루는데 필요한 통합적 수단

3. 소프트웨어 개발 프로세스(2)

■ 프로세스의 목적

- 이전에 얻은 노하우를 전달 -> 시행착오 감소 -> 빠르게 적응
- guide 역할

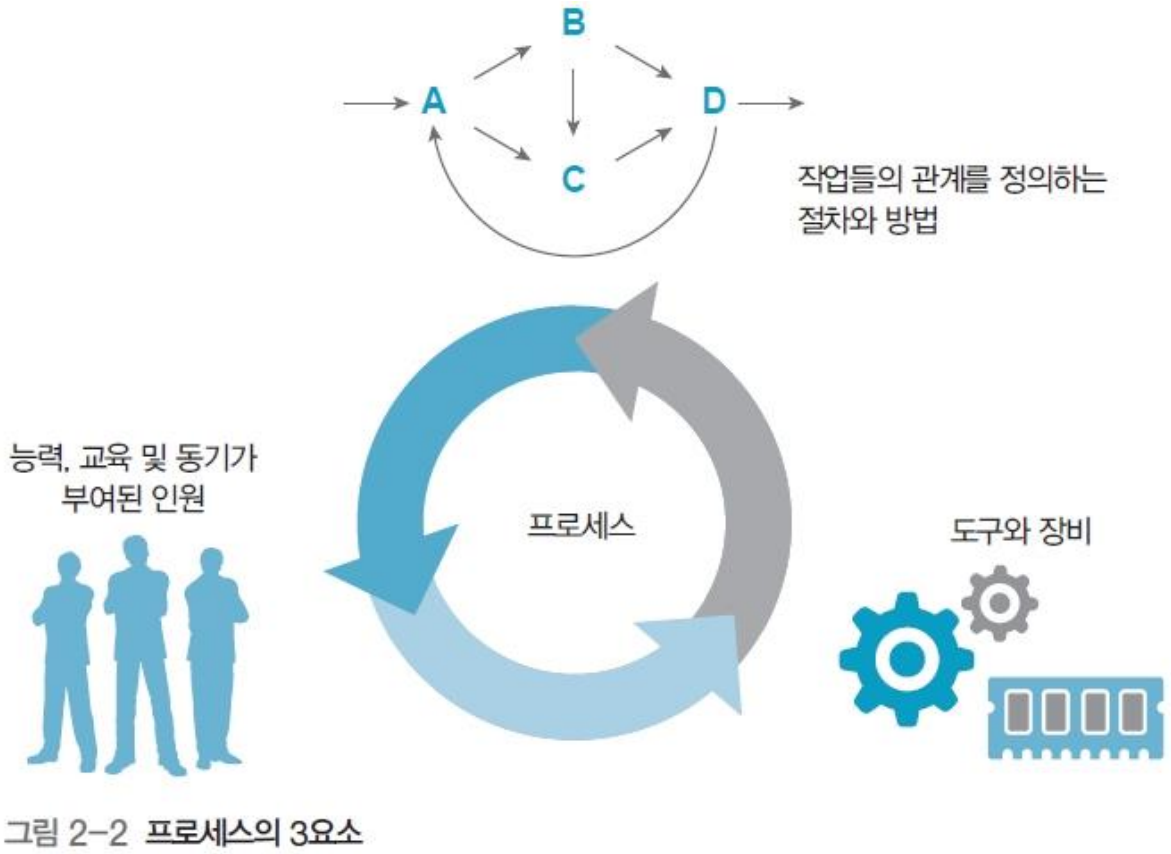


그림 2-2 프로세스의 3요소

Section 02 소프트웨어 프로세스 모델의 이해

1. 소프트웨어 개발 과정

- **작은 규모의 소프트웨어 개발 과정**
  - 개집 짓는 일에 비유 => 바로 코딩에 들어가도 됨
- **대규모의 소프트웨어 개발 과정**
  - 빌딩 짓는 일에 비유
  - 적합한 프로세스 모델을 정하는 것이 좋음

그림 2-3 개집 설계와 빌딩 설계

2. 소프트웨어 프로세스 모델(1)

■ 소프트웨어 프로세스 모델의 정의

- 소프트웨어 개발 생명주기(SDLC Software Development Life Cycle)
- SW를 어떻게 개발할 것인가에 대한 전체적인 흐름을 체계화한 개념
- 개발 계획 수립부터 최종 폐기 때까지의 전 과정을 다룸
- 순차적인 단계로 이루어 짐

■ 소프트웨어 프로세스 모델의 목적

- 공장에서 제품을 생산하듯이 소프트웨어 개발의 전 과정을 하나의 프로세스로 정의
- 주어진 예산과 자원으로 개발하고 관리하는 방법을 구체적으로 정의
- 고품질의 소프트웨어 제품 생산을 목적으로 함

2. 소프트웨어 프로세스 모델(2)

소프트웨어 프로세스 모델의 역할

- 프로젝트에 대한 전체적인 기본 골격을 세워줌
- 일정 계획을 수립할 수 있음
- 개발 비용 산정뿐 아니라 여러 자원을 산정하고 분배할 수 있음
- 참여자 간에 의사소통의 기준을 정할 수 있음
- 용어의 표준화를 가능케 할 수 있음
- 개발 진행 상황을 명확히 파악할 수 있음
- 각 단계별로 생성되는 문서를 포함한 산출물을 활용하여 검토할 수 있게 해줌

Section 03 주먹구구식 모델

1. 주먹구구식 모델

- **Build and fix 모델, code and fix 모델, 즉흥적 소프트웨어 개발 모델**

- **주먹구구식**
  - 주먹으로 구구셈을 따지던 방법에서 유래한 말
  - 정확한 앞뒤 계산 없이 일을 대충 처리할 때 쓰는 말

- **소프트웨어 개발에서의 주먹구구식 모델**
  - 공식적인 가이드라인이나 프로세스가 없는 개발 방식
  - 요구 분석 명세서나 설계 단계 없이 간단한 기능만을 정리하여 개발하는 형태
  - 일단 코드를 작성하여 제품을 만들어본 후에 요구 분석, 설계, 유지보수에 대해 생각

2. 주먹구구식 모델의 개발 단계

① 첫 번째 버전의 코드를 작성하여 제품을 완성한다.

② 작성된 코드에 문제점이 있으면 수정하여 해결한다.

③ 문제가 없으면 사용한다.

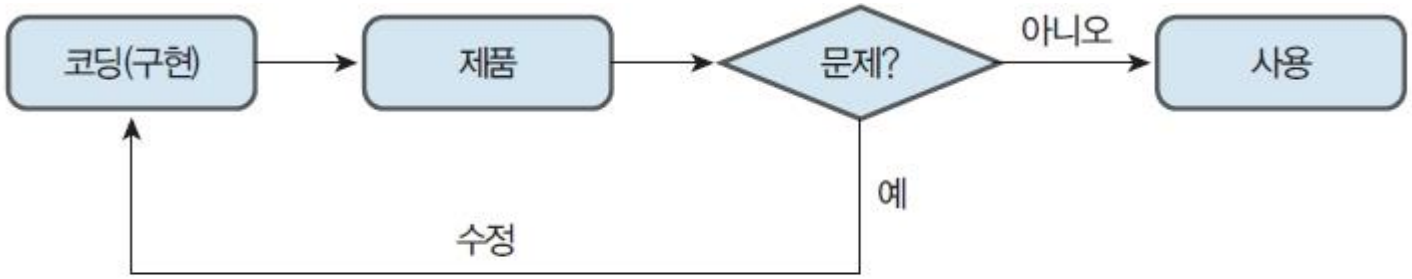


그림 2-4 주먹구구식 모델

3. 주먹구구식 모델의 사용 및 단점

주먹구구식 모델의 사용

- 개발자 한 명이 단시간에 마칠 수 있는 경우에 적합
- 대학 수업의 한 학기용 프로젝트 정도

주먹구구식 모델의 단점

- 정해진 개발 순서나 각 단계별로 문서화된 산출물이 없어 관리 및 유지보수가 어렵다.
- 프로젝트 전체 범위를 알 수 없을 뿐더러 좋은 아키텍처를 만들 수도 없다.
- 일을 효과적으로 나눠 개발할 수도 없으며, 프로젝트 진척 상황을 파악할 수 없다.
- 계속적 수정으로 인해 프로그램의 구조가 나빠져 수정이 매우 어려워진다.

Section 04 선형 순차적 모델

1. 선형 순차적 모델

- **Linear sequential 모델, waterfall 모델, Classic life cycle**
  - 각 단계가 하향식으로 진행됨
  - 각 단계가 병행되거나 거슬러 반복되지 않음
  - 각 단계가 끝날 때마다 확실히 매듭을 짓고 다음 단계로 나감

그림 2-5 폭포수 모델

2. 폭포수 모델의 개발 절차

- 계획단계(3장에서 자세히 다룸)
- 요구분석 단계(4장에서 자세히 다룸)
- 설계 단계(5-6장에서 자세히 다룸)
- 구현 단계(7장에서 자세히 다룸)
- 테스트 단계(8장에서 자세히 다룸)
- 유지보수 단계(10장에서 자세히 다룸)

3. 폭포수 모델의 장점

- **관리의 용이**
  - 절차가 간결하고 이해하기 쉬움
  - 단계별 진척 사항에 대한 관리가 용이함
- **체계적인 문서화**
  - 단계별 산출물을 체계적으로 문서화할 수 있음
- **요구사항의 변화가 적은 프로젝트에 적합**

4. 폭포수 모델의 단점

- 각 단계는 앞 단계가 완료되어야 수행할 수 있다.
- 각 단계의 결과물이 완벽한 수준으로 작성되어야 다음 단계에 오류를 넘겨주지 않는다.
- 사용자가 중간에 가시적인 결과를 볼 수 없어 답답해할 수 있다.
- => 요구 사항의 변화가 많은 상황에는 맞지 않음
- => 요구 사항이 어느 정도 고정되어 있는 경우에 유용함

그림 2-6 폭포수 모델의 단점

Section 05 V 모델

1. V 모델

- 폭포수 모델 + 테스트 단계 추가 확장
- 산출물 중심(폭포수 모델) vs 각 개발 단계를 검증하는 데 초점(V 모델)

※ 자세한 내용은 8장(테스트)에서 다룸

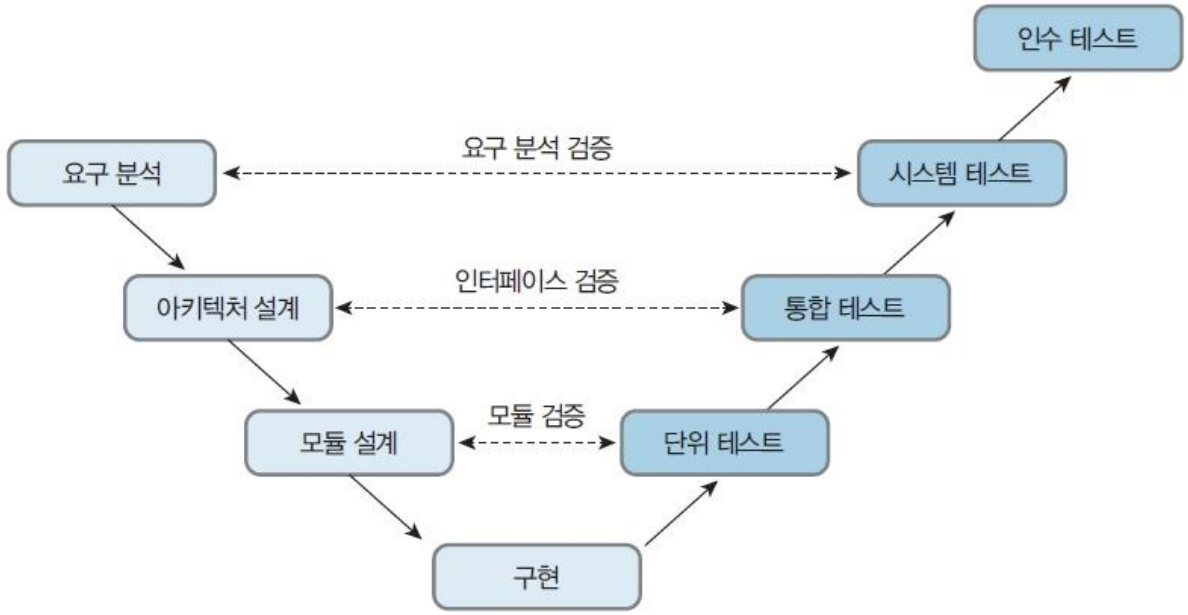


그림 2-7 V 모델

1. V 모델

- **단위 테스트**
  - 개별 모듈 검증
- **통합 테스트**
  - 모듈 간의 인터페이스 확인
- **시스템 테스트**
  - 모듈이 모두 통합된 후 사용자의 요구 사항들을 만족하는지 확인
- **인수 테스트**
  - 시스템이 예상대로 동작하고 요구 사항에 부합하는지 사용자가 확인

Section 06 진화적 프로세스 모델

1. 진화적 프로세스 모델의 등장 배경

- 선형순차적 모델의 대표: 폭포수 모델
- 진화적 프로세스 모델의 대표: 프로토타입 모델
  - 목적: 요구 사항의 변화에 신속히 대응

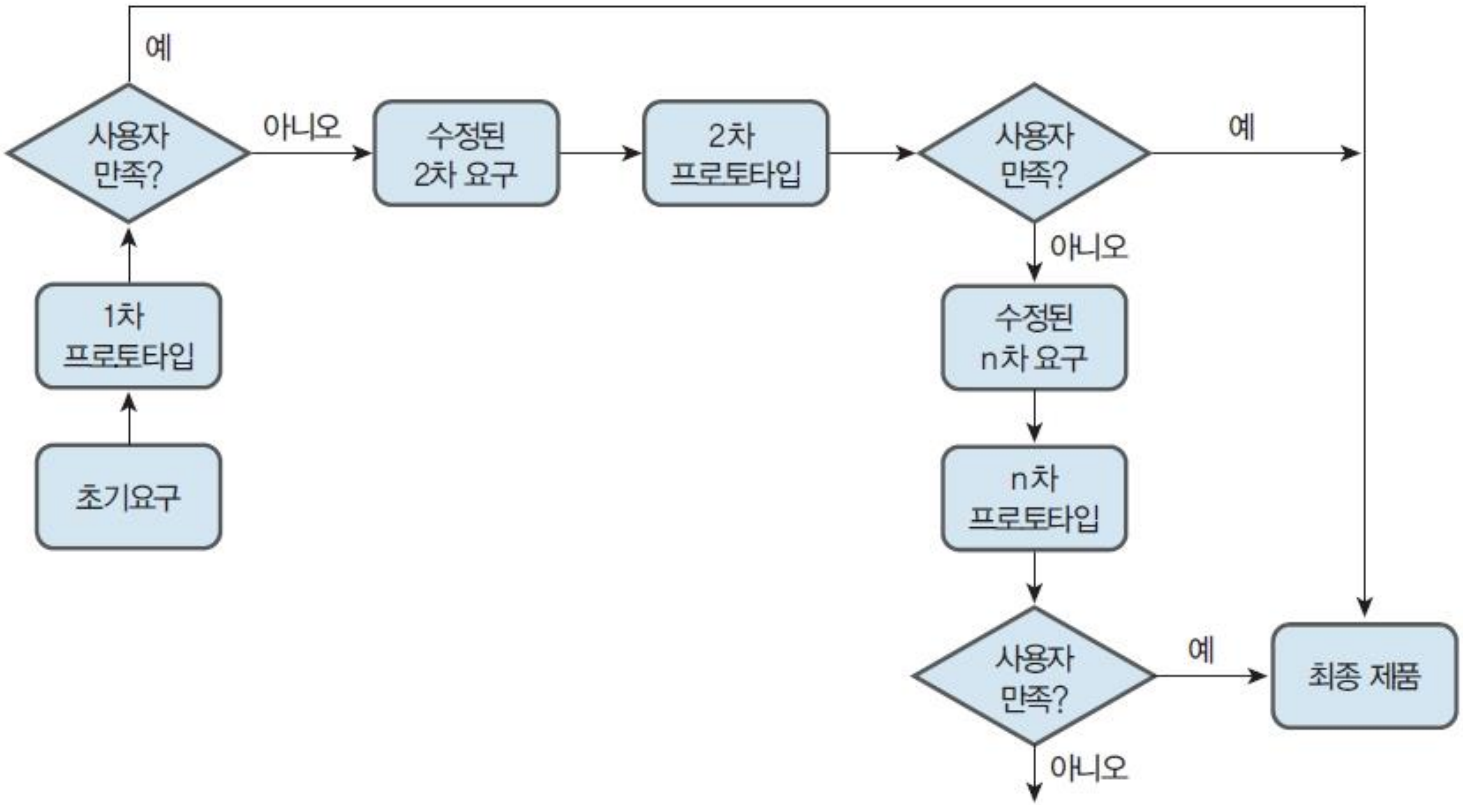


그림 2-8 **진화적 프로세스 모델**

1-1 프로토타입

- **프로토타입**
  - 대량 생산에 앞서 미리 제작해보는 원형 또는 시제품으로, 제작물의 모형
- **소프트웨어 개발에서의 프로토타입**
  - 정식 절차에 따라 완전한 소프트웨어를 만들기 전에 사용자의 요구를 받아 일단 모형을 만들고 이 모형을 사용자와 의사소통 하는 도구로 활용
  - 사용자 인터페이스 중심으로 간략한 기능만 구현

그림 2-9 프로토타입의 예 : 아파트 모델하우스

1-2 프로토타입 모델의 개발 생명주기

■ 프로토타입 모델

- 폭포수 모델 + 프로토타이핑

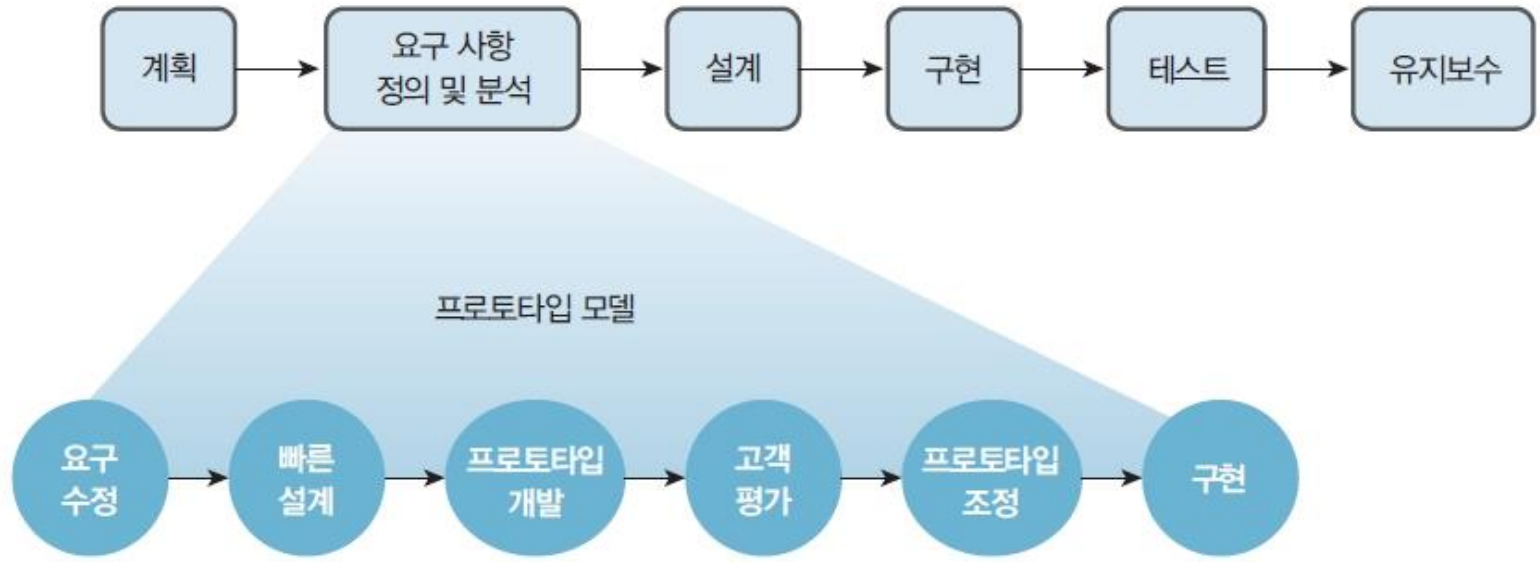


그림 2-10 프로토타입 모델의 개발 생명주기

2. 실험적 프로토타입 모델

■ 실험적 프로토타입 모델

- 최종 프로토타입을 버리고 처음부터 새로 소프트웨어를 개발
- 프로토타입은 사용자의 요구를 알아내기 위해 사용자와 대화하는 도구로 사용함

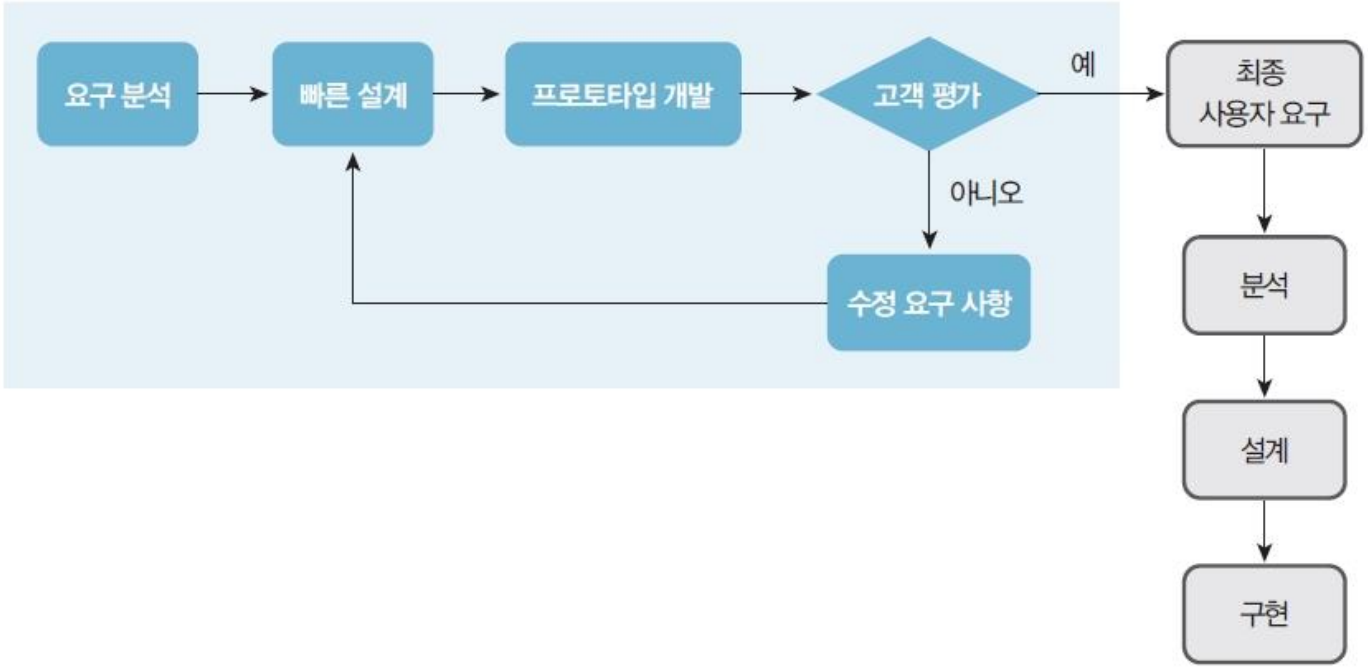


그림 2-11 실험적 프로토타입 모델 절차

3. 진화적 프로토타입 모델

■ 진화적 프로토타입 모델

- 최종 프로토타입을 버리지 않고 지속적으로 발전시켜 최종 소프트웨어를 완성

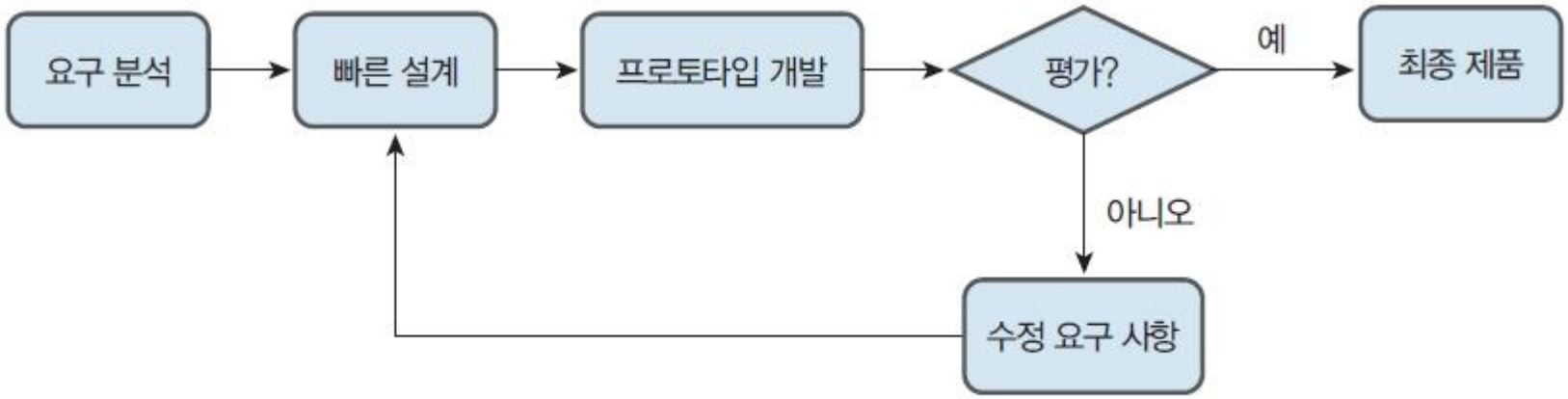


그림 2-12 진화적 프로토타입 모델 절차

4. 프로토타입 모델의 개발절차(1)

① 요구사항 정의 및 분석

- 1차 개략적인 요구 사항 정의 후 2차, 3차, … n차를 반복하면서 최종 프로토타입 개발

② 프로토타입 설계

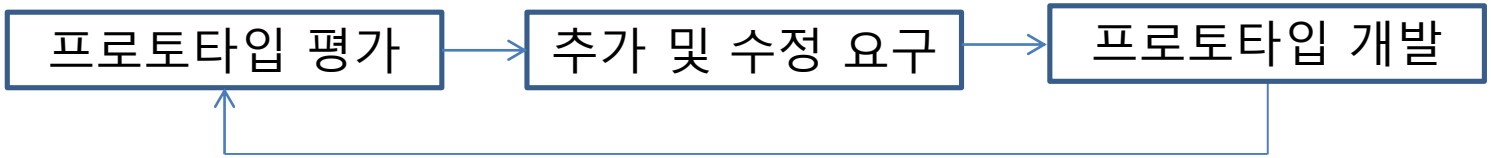
- 완전한 설계 대신, 사용자와 대화할 수 있는 수준의 설계
- 입출력 화면을 통한 사용자 인터페이스 중심 설계

③ 프로토타입 개발

- 완전히 동작하는 완제품을 개발하는 것이 아님
- 입력 화면을 통한 사용자의 요구 항목 확인
- 출력 결과를 통해 사용자가 원하는 것인지 확인

4. 프로토타입 모델의 개발절차(2)

④ 사용자에 의한 프로토타입 평가



⑤ 구현

- 최종 프로토타입 개발

5. 프로토타입 장/단점

장점

- 프로토타입이 의사소통 도구로 활용
- 반복된 요구사항 정의를 통해 사용자 요구가 충분히 반영된 요구 분석 명세서 작성
- 초기 프로토타입 사용을 통한 새로운 요구사항 발견
- 프로토타입 사용을 통한 완성품의 예측 가능

단점

- 반복적 개발을 통한 투입 인력 및 비용 산정의 어려움
- 프로토타이핑 과정에 대한 통제 및 관리의 어려움
  - 폭포수 모델같이 중간 점검을 위한 이정표나 중간 산출물 생성의 어려움
- 불명확한 개발 범위로 인한 개발 종료 및 목표의 불확실성

Section 07 나선형 모델

1. 나선형 모델의 특성(1)

- 진화적 프로토타입 모델 + 위험 분석
- 뱅글뱅글 돌어가면서 점점 완성도가 높은 제품이 만들어짐

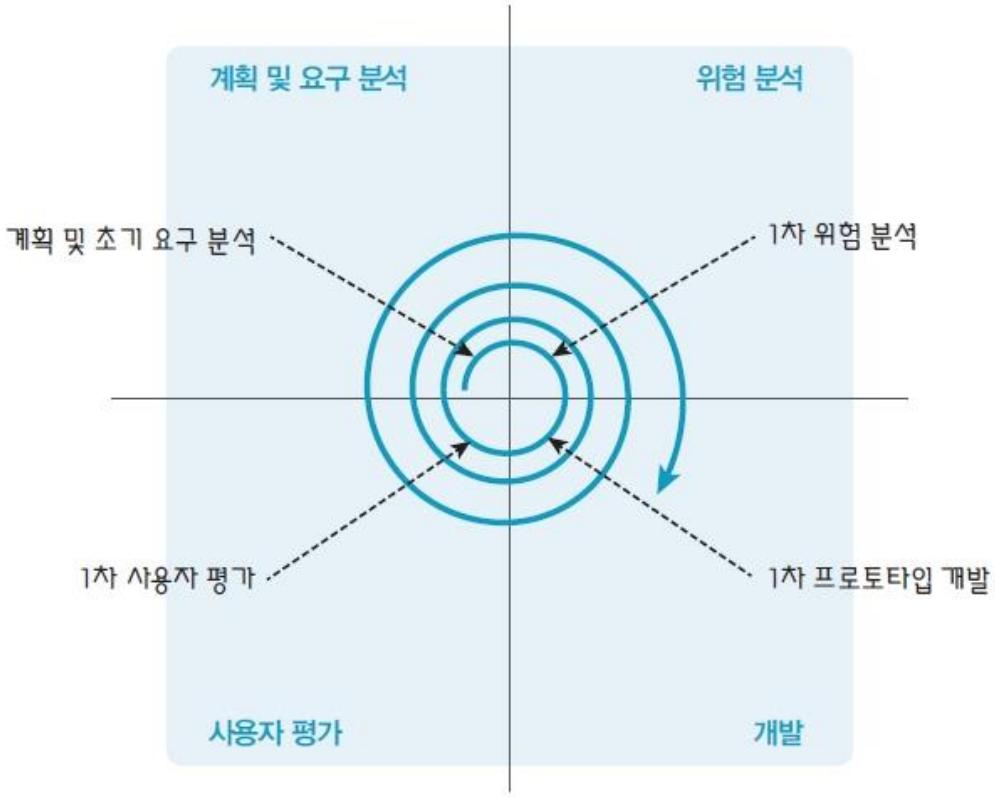


그림 2-13 나선형 모델

1. 나선형 모델의 특성(2)

- **위험 분석 단계의 위험 요소**
  - 소프트웨어 개발 과정이 순조롭게 진행되는데 방해되는 모든 것

- **위험 요소의 예**
  - 빈번히 변경되는 요구사항
  - 팀원들의 경험 부족
  - 결속력이 떨어지는 팀워크
  - 프로젝트 관리 부족
  - … …

2. 나선형 모델의 개발 절차(1)

① 계획 및 요구 분석 단계

- 사용자의 개발 의도 파악
- 프로젝트의 명확한 목표
- 제약 조건의 대안을 고려한 계획 수립
- 기능/비기능 요구사항 정의 및 분석

② 위험 분석 단계

표 2-1 소프트웨어 개발 시 위험 요소

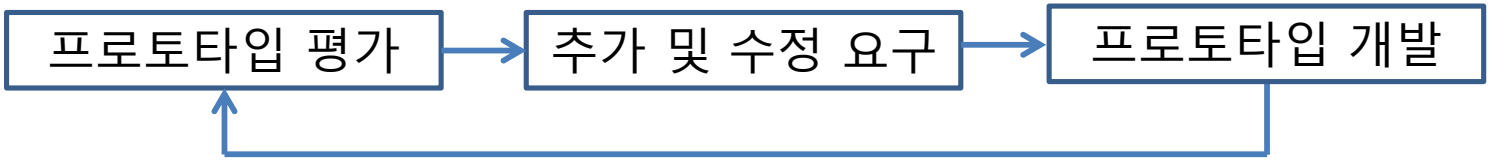
위험 요소	위험 내용
개발자의 이직	프로젝트 수행 중 개발자의 이직
요구 사항 변경	요구 사항 확정 이후에 계속되는 변경 요구
발주사의 재정적 어려움	프로젝트 수행 중 발주사의 경제적 어려움
예상을 빗나간 투입 인력	처음에 예측한 인력보다 더 많은 인력을 필요로 하는 경우
개발 기간의 부족	처음에 예측한 개발 기간을 초과한 경우
개발비의 초과	처음에 예측한 개발비로 완료할 수 없는 경우

2. 나선형 모델의 개발 절차(2)

③ 개발 단계

- 프로토타입 개발 (설계, 구현 등)

④ 사용자 평가 단계



3. 나선형 모형의 장/단점

장점

- 사전 위험 분석을 통한 돌출 위험 요소 감소 → 프로젝트 중단 확률 감소
- 사용자 평가에 의한 개발 방식 → 요구가 충분히 반영된 제품 → 사용자의 불만 감소

단점

- 반복적 개발에 의한 프로젝트 기간 연장의 가능성
- 반복 회수의 증가에 따른 프로젝트 관리의 어려움
- 위험 관리의 중요 → 위험 전문가 필요에 따른 부담

Section 08 단계적 개발 모델

1. 단계적 개발 모델

■ 단계적 개발

- 개발과 사용을 병행하는 과정을 반복하여 진행하면서 완료
  - 먼저 릴리즈1 개발하여 사용자가 사용, 그 동안, 릴리즈2 개발

■ 릴리즈 구성 방법에 따른 분류

- 점증적 개발 방법
- 반복적 개발 방법

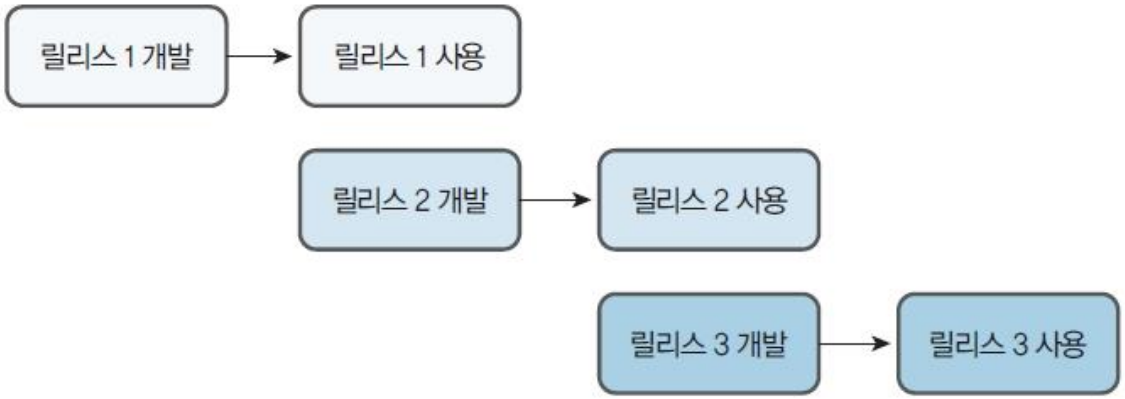


그림 2-14 단계적 개발 모델

2. 점증적 개발 방법

■ 개발 범위 증가

- '하나가 끝나면 그 다음, 또 하나가 끝나면 그 다음...'과 같이 하나씩 늘려 감
- 중요한 것부터 먼저 개발 => 점점 개발 범위를 늘려감

그림 2-15 점증적 개발 방법의 예 : 양식 코스 요리

2-1 점증적 개발 방법의 예

(예 1) 도서 집필

- 1장을 완벽히 쓰고, 2장, 3장, …, 10장까지 완성해나가는 방식으로 책을 집필

(예 2) 3층 건물 건축

(예 3) 대학 종합정보시스템 개발

2. 점증적 개발 방법

- **소프트웨어 개발 시**
  - 예: 5개의 서브 시스템으로 구성
    - 먼저 첫 번째 서브시스템 개발 => 두 번째 서브시스템 개발 및 통합 => n 번째 서브시스템...

그림 2-15 점증적 개발 방법의 예 : 양식 코스 요리

3. 반복적 개발 방법

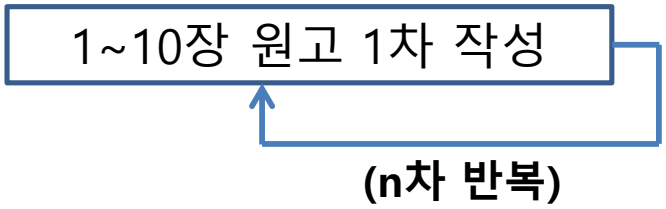
■ 품질 증가

- 초기 시스템 개발 => 보완하여 2차 버전 시스템 개발 => 보완하여 n차 버전 시스템 ...

그림 2-16 반복적 개발 방법의 예 : 한 상 가득 차려진 한정식

3-1 반복적 개발 방법의 예

(예 1) 도서 집필



(예 2) 소프트웨어 개발

- **실제 개발에서는,**
  - 점증적 개발과 반복적 개발을 동시에 사용하는경우가 많음

Section 09 통합 프로세스 모델

1. 통합 프로세스 모델(UP, Unified Process)

폭포수 모델 문제점 → 해결 방안 → 반복적 개발 방법

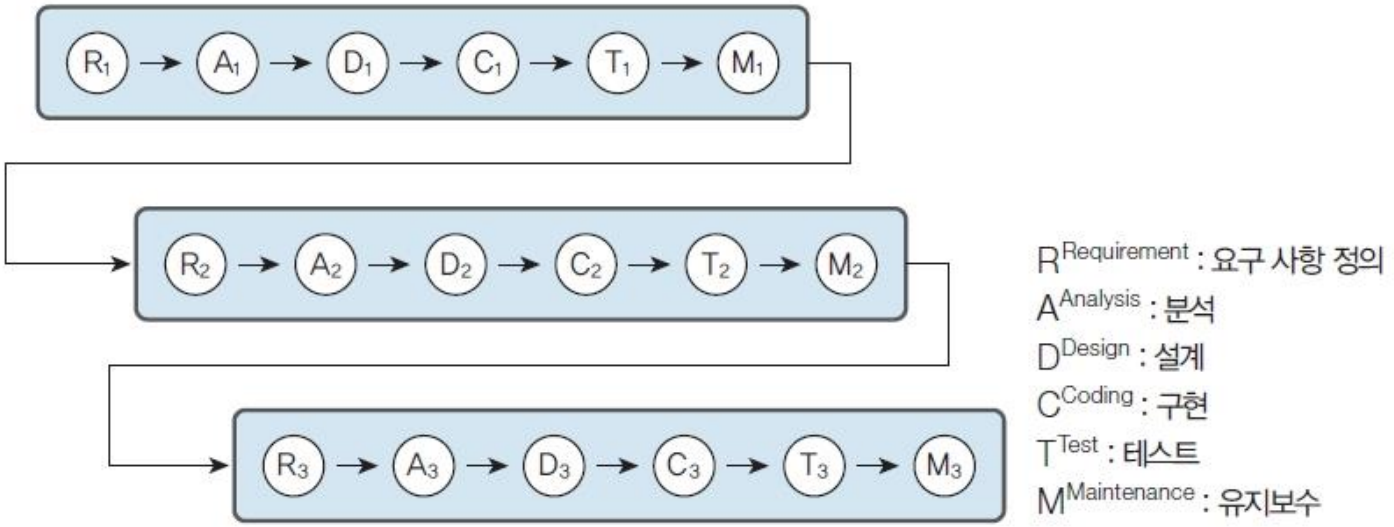


그림 2-17 반복적 개발 방법

1. 통합 프로세스 모델(UP, Unified Process)

통합 프로세스 모델

- 반복적/점증적 프로세스로서, 가장 많이 사용됨
- OMG(Object Management Group)에서 UML과 함께 제안함

통합 프로세스 모델의 절차

- 전체 개발 과정을 4 단계(도입, 구체화, 구축, 전이)로 나눔
- 각 단계는 여러 개의 반복(iteration)으로 구성됨
- 각 반복 내에는 9개의 개발 영역(disciplines)이 행해짐
  - 비즈니스 모델링, 요구사항 정의, 분석 및 설계, 구현, 테스트, 배치
  - 형상(변화) 관리, 프로젝트 관리, 환경 점검

2. 통합 프로세스(UP) 모델

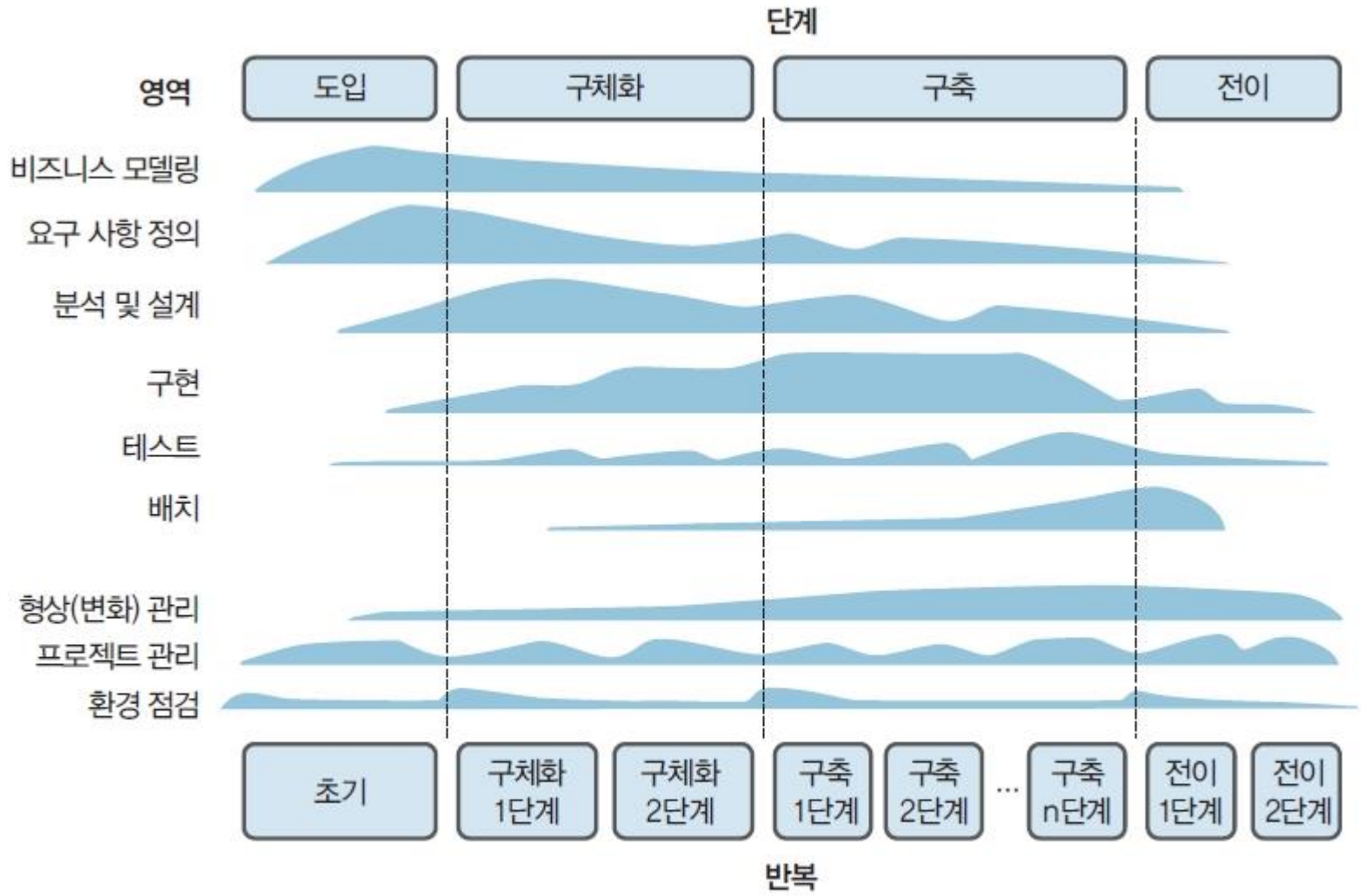


그림 2-18 **통합 프로세스UP 모델**

3. 통합 프로세스(UP) 방법

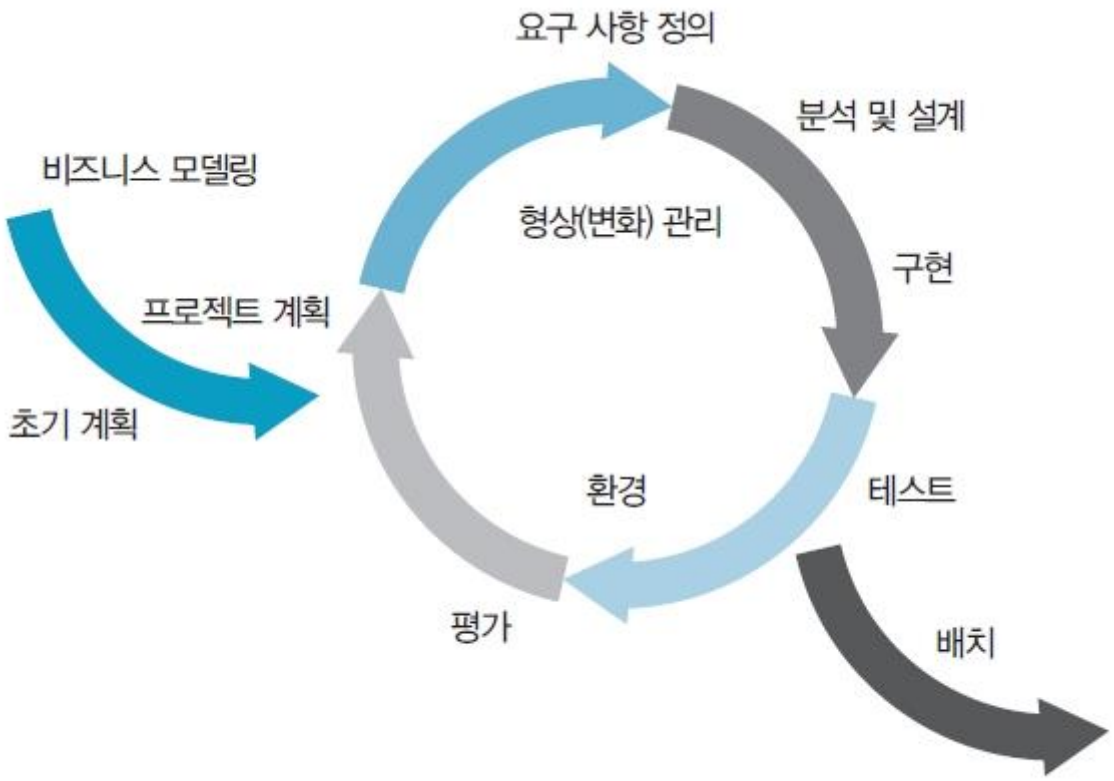


그림 2-19 통합 프로세스UP 방법

4. 통합 프로세스UP 모델의 절차

① 도입 단계inception phase

② 구체화 단계elaboration phase

③ 구축 단계construction phase

④ 전이 단계transition phase

⑤ 도입/구체화/구축전이 단계의 공통 작업

① 도입 단계inception phase

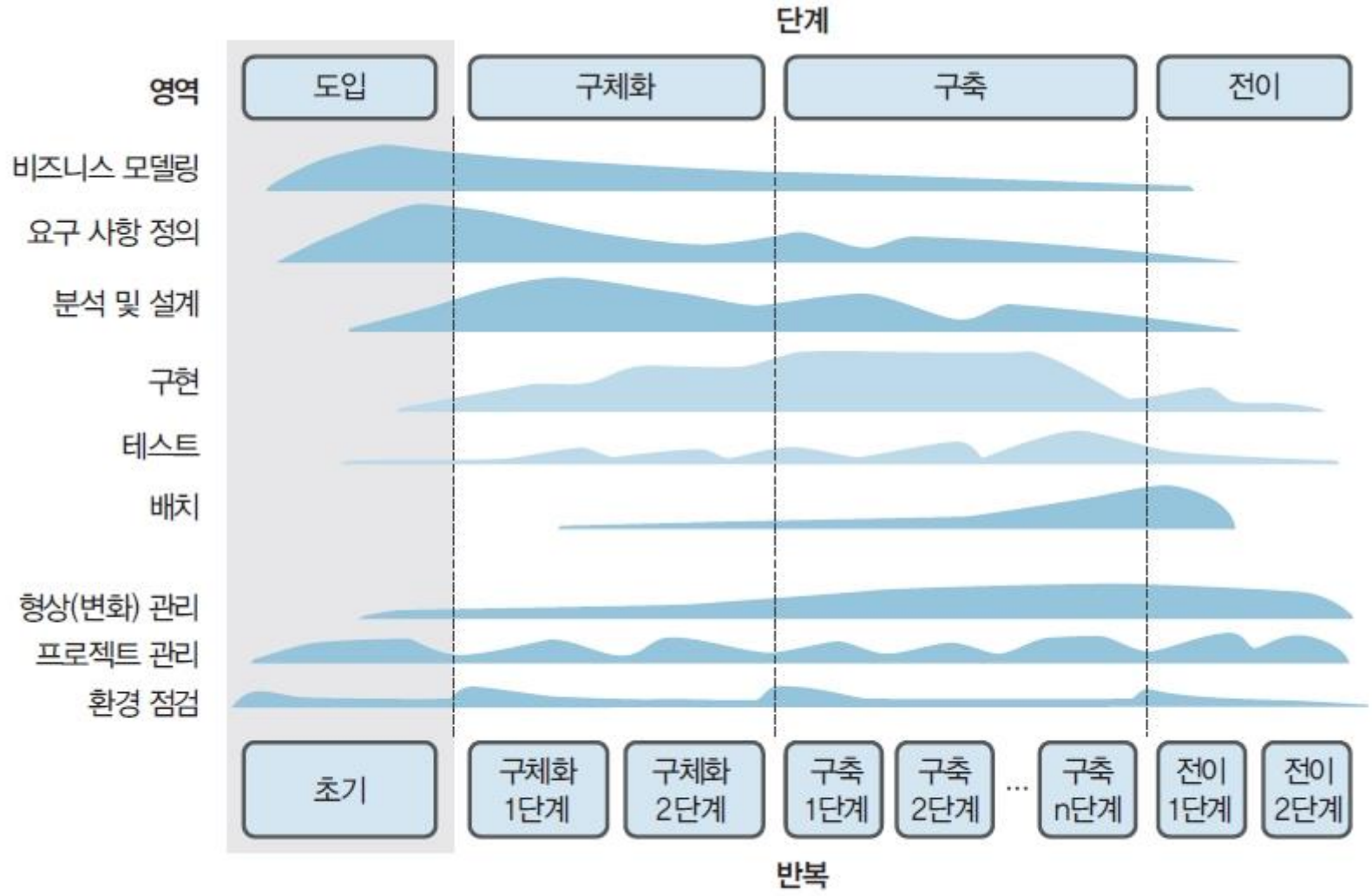


그림 2-20 통합 프로세스 모델의 도입 단계

4. 통합 프로세스UP 모델의 절차

① 도입 단계inception phase

- 프로젝트 시작 단계로서, 구현, 테스트, 배치 작업은 거의 없음
- 주요 활동
  - 개발의 기초가 되는 아이디어 도출
  - 투자 비용 대비 효과 분석
  - 사업 타당성 및 실현 가능성 확인
  - 소프트웨어 개발 목표 수립
  - 개발 범위 파악
  - 비용과 기간 산정
  - 프로젝트 위험 요소 발굴
  - 사용자의 전체 요구 사항 이해 및 정의
  - … …

② 구체화 단계elaboration phase

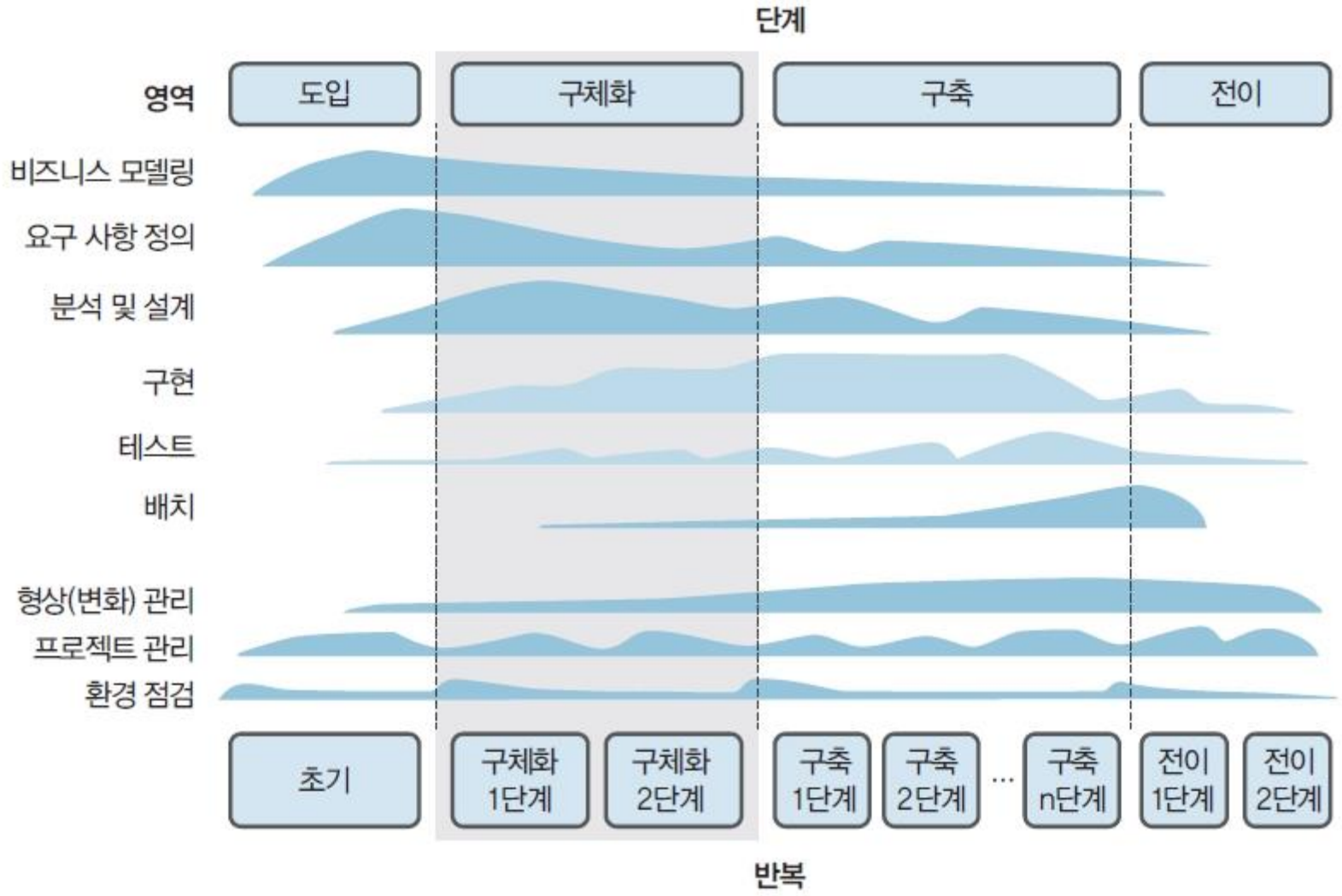


그림 2-21 통합 프로세스 모델의 구체화 단계

4. 통합 프로세스UP 모델의 절차

② 구체화 단계elaboration phase

- 상세 단계, 정련 단계라고도 함
- 보통 2~4개의 반복 단위로 구성됨
- 비즈니스 모델링과 요구 사항 정의 작업은 점차 줄고, 분석 및 설계 작업이 왕성함
- 설계 결과에 따른 구현 작업 및 테스트도 시작됨
- 주요 활동
  - 아키텍처 수립
  - 도입 단계에서 파악한 요구 사항을 상세하게 분석
  - 중대한 위험 요소를 찾아 축소 및 제

③ 구축 단계construction phase

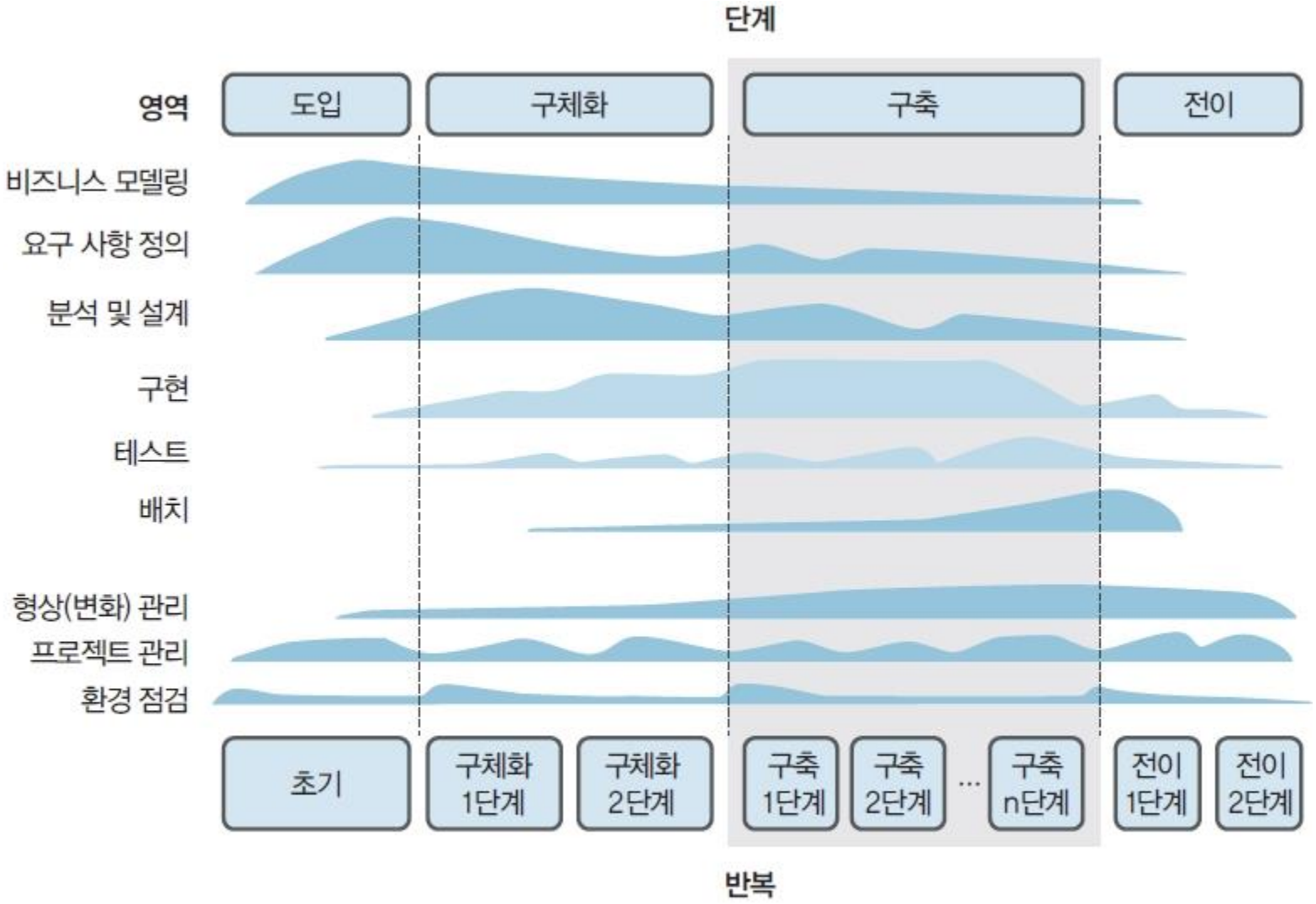


그림 2-22 **통합 프로세스 모델의 구축 단계**

4. 통합 프로세스UP 모델의 절차

③ 구축 단계construction phase

- 구현 작업이 가장 많이 이루어짐
- 비즈니스 모델링과 요구 사항 정의 작업이 많이 줄고, 분석 및 설계 작업도 구체화 단계보다 줄어듬
- 테스트 작업도 점차 늘어나고, 배치 양도 증가
- 주요 활동
  - 인도 가능한 최초 실행 버전의 소프트웨어 개발
  - 모든 개발 요소 구현
  - 단위 테스트 및 통합 테스트 수행
  - 사용자 설명서 작성

④ 전이 단계transition phase

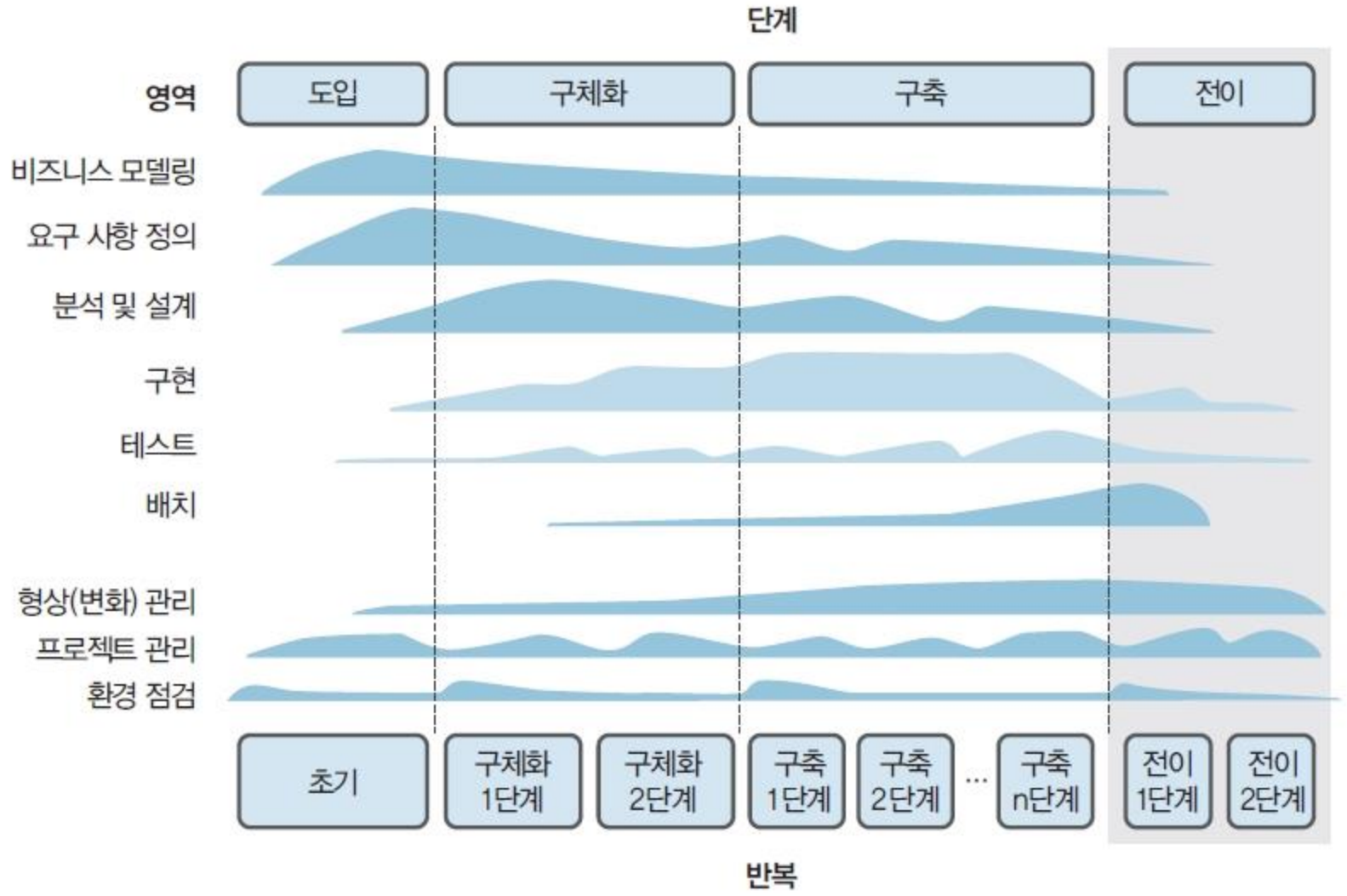


그림 2-23 **통합 프로세스 모델의 전이 단계**

4. 통합 프로세스UP 모델의 절차

④ 전이 단계transition phase

- 이행 단계라고도 함
- 사용자를 위한 제품을 완성하는 단계
- 완성된 제품을 사용자에게 넘겨주는 과정에서 수행해야 할 일을 함
- 배치 작업이 주로 일어남
- 주요 활동
  - 개발된 모듈(컴포넌트)에 대해 베타 테스트 실시
  - 사용자에게 배포 가능한 단위로 묶는 작업 수행
  - 사용자가 사용자 환경에서 직접 테스트(인수 테스트)
  - 소프트웨어 제품, 사용자 설명서 등을 사용자에게 인도
  - 제품 사용자 및 유지보수 담당자에게 교육

⑤ 도입/구체화/구축전이 단계의 공통 작업

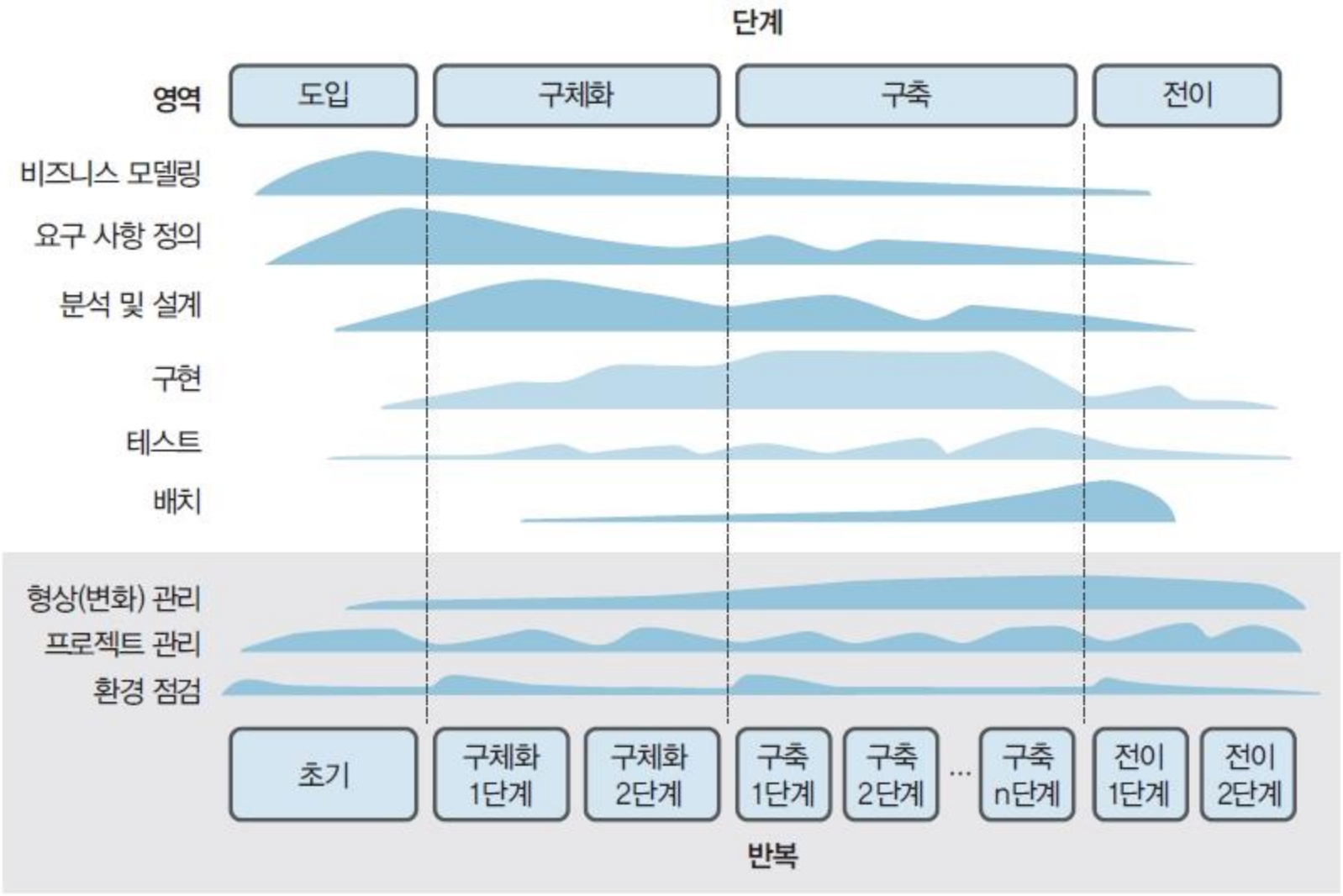


그림 2-24 **통합 프로세스 모델의 공통 작업**

4. 통합 프로세스UP 모델의 절차

⑤ 도입/구체화/구축전이 단계의 공통 작업

- 분석, 설계, 구현, 테스트 작업이 공통으로 일어남
  - 단, 각 단계별로 수행하는 정도에 차이가 있음
- 각 단계 내에서 여러 번의 반복 작업이 일어남(특히, 구체화와 전이 단계)
- 형상 및 변화 관리, 프로젝트 관리, 환경 점검 등은 지속적으로 일어남

Section 10 애자일 프로세스 모델

1. 애자일 프로세스 모델

애자일(agile)

- '날렵한', '민첩한'

애자일 프로세스 모델

- 고객의 요구에 민첩하게 대응하고 그때그때 주어지는 문제를 풀어나가는 방법론
  - 폭포수 모델같은 전통적인 모델은, 산출물 위주의 거대하고 무거움
- 예: 익스트림 프로그래밍(XP, eXtreme Programming), 스크럼(scrum), 크리스털(crystal) 등

애자일의 기본 가치(애자일 선언문)

- 프로세스와 도구 중심이 아닌, 개개인과의 상호 소통 중시
- 문서 중심이 아닌, 실행 가능한 소프트웨어 중시
- 계약과 협상 중심이 아닌, 고객과의 협력 중시
- 계획 중심이 아닌, 변화에 대한 민첩한 대응 중시

1. 애자일 프로세스 모델

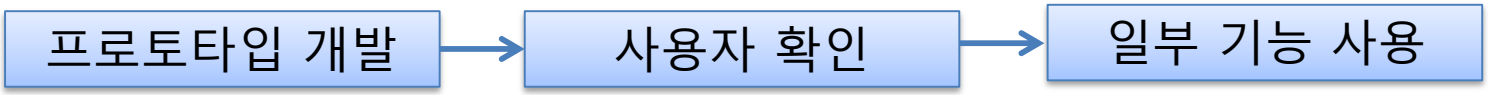
■ 애자일(agile)의 원칙

- 고객을 만족시키기 위해 가치 있는 소프트웨어를 빨리, 지속적으로 제공
- 개발 후반에 새로 추가되는 요구 사항도 기꺼이 받아들임
- 동작 가능한 소프트웨어를 2주~2달 간격으로 자주 고객에게 전달
- 개발자는 업무 담당자와 매일 의견 주고 받음
- 전화, 팩스 보다 가능한 직접 만나서 대화
- 실행 가능한 소프트웨어를 보여줌으로써 진척 사항을 알려줌
- 자율적 사고와 자유로운 분위기로 프로젝트 수행
- 개발 팀은 스스로 정기적인 미팅 진행

1-1 애자일 프로세스 모델의 이해

■ 애자일의 개발 방법

- 반복적인 개발을 통한 잦은 출시를 목표로 함



2. 애자일 방법과 폭포수 모델의 비교

구분	애자일 방법론	폭포수 모델
추가 요구 사항의 수용	추가 요구 사항을 수용할 수 있는 방법의 설계	추가 요구 사항을 반영하기 어려운 구조
릴리스 시점	수시로 릴리즈	최종 완성된 제품을 릴리스
시작 상태	시작 단계는 미흡, 점차 완성도가 높아짐	시작 단계에서의 완성도가 매우 높음
고객과의 의사소통	처음부터 사용자의 참여 유도, 대화를 통한 개발 진행	사용자와 산출물의 근거 중심, 대화 부족
진행 상황 점검	개발자와 사용자는 개발 초기부터 진행 상황 공유	단계별 산출물에 대한 결과로 개발의 진척 상황을 점검
분석/설계/구현 진행 과정	하나의 단계 또는 반복 안에 분석/설계/구현 과정이 모두 포함되어 동시에 진행	분석/설계/구현 과정이 명확
모듈(컴포넌트)통합	개발 초기부터 빈번한 통합.문제점을 빨리 발견하고 수정하는 방식	구현이 완료된 후에 모듈 간의 통합 작업을 수행

3. 애자일 개발 방법론(스크럼)

■ 스크럼 개발 프로세스

- 소프트웨어 개발보다는 팀의 개선과 프로젝트 관리를 위한 애자일 방법론
- 경험적 관리 기법 중 하나
- 구체적인 프로세스를 명확하게 제시하지 않음
- 개발 팀(조직)을 운영하는 효율적인 운영 방식(지침)

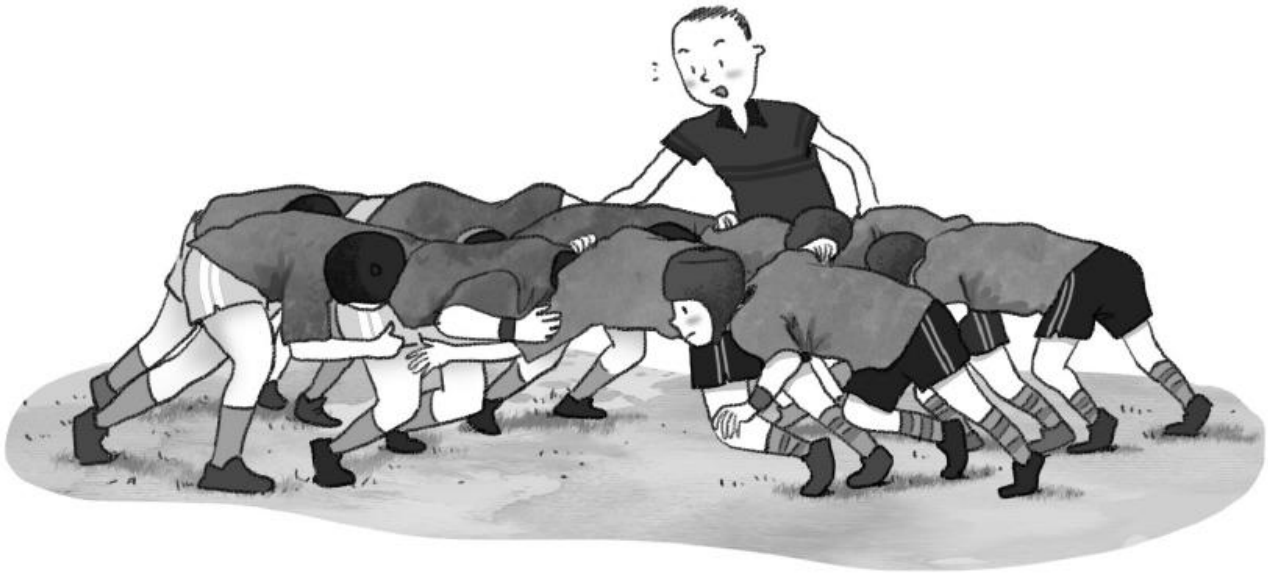


그림 2-25 럭비 경기의 스크럼 대형

4. 스크럼 방식의 진행 과정

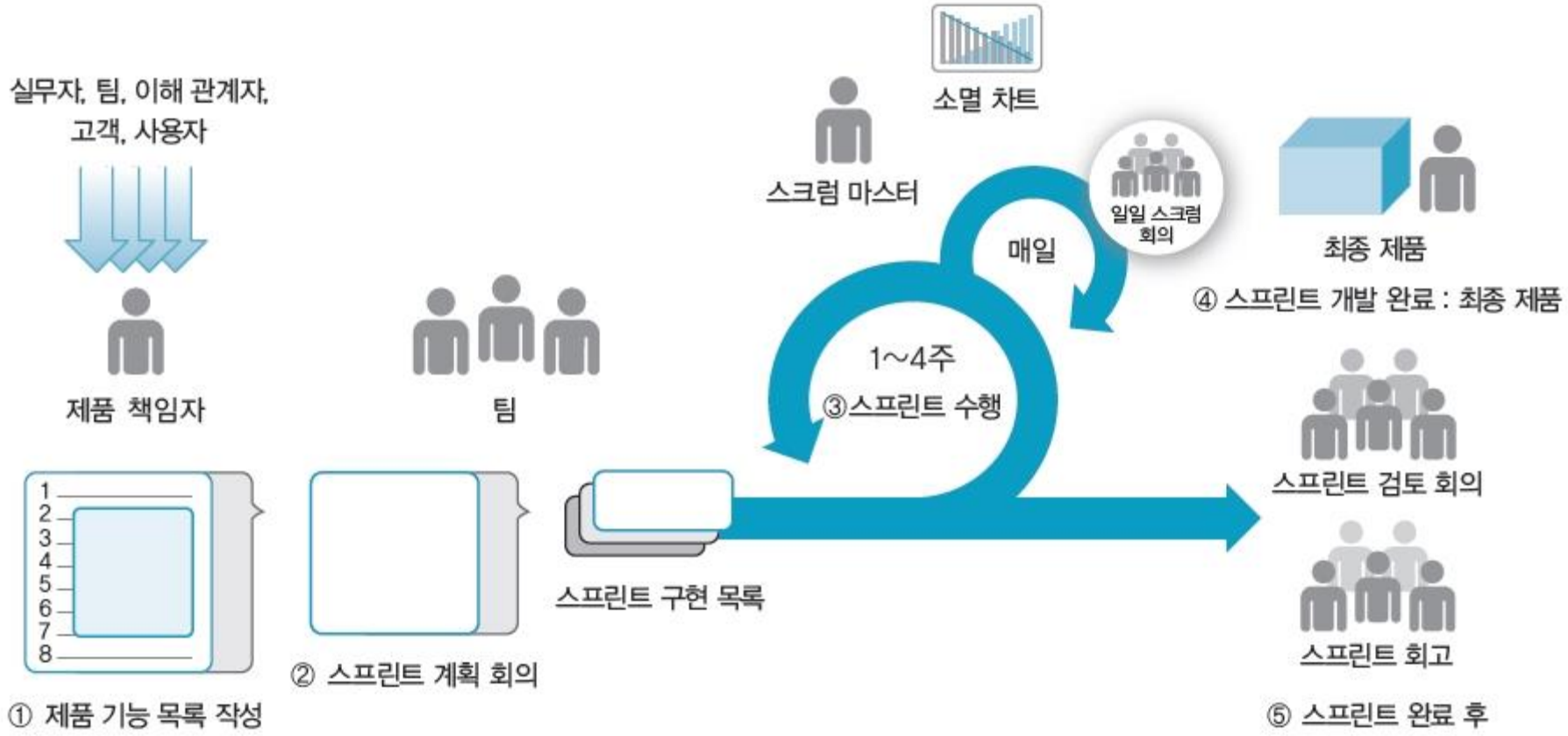


그림 2-26 스크럼 방식의 진행 과정

5. 스크럼 방식에서 사용되는 용어(1)

■ 제품 기능 목록product backlog 작성

- 우선순위가 매겨진 사용자의 요구 사항 목록

표 2-3 제품 기능 목록의 예 : 소프트웨어 공학 원고 목록

순위	요구 사항 목록	요구 사항 내역	작업 소요 기간	작업 월
1	품질(9장)	프로세스 품질과 제품 품질에 대해 기술한다.	30일	2015. 12.
2	테스트(8장)	테스트의 종류를 분류하고 단계별로 설명한다.	40일	2016. 1.
3	요구 분석(4장)	사용자 요구 사항을 정의하는 방법에 대해 기술한다.	20일	2016. 2.
...	.....	.....	30일	.....
9	소프트웨어 개발 프로세스(2장)	개발 프로세스의 종류에 대해 기술한다.	20일	2016. 4.
10	소프트웨어 공학 소개(1장)	소프트웨어 공학의 일반적인 내용을 기술한다.	10일	2016. 5.

5. 스크럼 방식에서 사용되는 용어(2)

- **사용자 스토리user story작성 및 스토리 포인트story point 산정**
  - 사용자 스토리: 메모지 한 장에 구현할 기능을 사용자 관점에서 사용자 언어로 작성한 사용자 요구 사항
  - 스토리 포인트: 사용자 스토리를 수행하는데 걸리는 상대적인 개발 기간(시간)

그림 2-27 사용자 스토리

5. 스크럼 방식에서 사용되는 용어(3)

■ 사용자 스토리user story

- 제품 기능 목록에 정의된 사용자 관점에서의 기능
- 사용자에게 가치를 평가 받을 수 있도록 기능을 표현한 것
- 보통 작은 인덱스 카드를 사용해 필요한 것만 짧게 표현
- 고객의 요구 사항을 문서화한 것이라기보다는 표현했다고 보는 것이 적합
- 유스케이스보다 작은 규모
- 사용자 스토리는 반복을 마치면 사라지지만 유스케이스는 개발 기간 동안 지속
- 사용자와 충분히 대화하여 세부 사항을 구체적으로 서술
- 테스트를 통해 스토리가 완료된 것을 확인
- 다른 스토리에 종속되지 않고 독립적이며, 협상 가능해야 함
- 추정 및 측정 가능해야 함
- 사용자 스토리는 스토리가 큰 것보다는 많은 것이 좋음
- 테스트가 가능해야 좋은 사용자 스토리

5. 스크럼 방식에서 사용되는 용어(4)

- **스프린트sprint**
  - '전력 질주'
  - 작업량으로 볼 때 그렇게 많지 않고, 개발 기간도 짧다.
  - 작은 단위의 개발 업무를 단기간 내에 전력 질주하여 개발한다는 뜻
  - 스프린트 주기: 보통 2~4주
  - 스프린트에 배정된 작업은 중간에 멈추지 않고 완성(팀원 교체도 없음)

그림 2-28 단거리 달리기 같은 스프린트

5. 스크럼 방식에서 사용되는 용어(5)

■ 스프린트의 예

계획: 소프트웨어 공학 원고 작성, 총기간(1년), 1장/(10일~40일)

- 스프린트 = 반복 주기 = 10일~40일

표 2-4 스프린트 구현 목록의 진척 관리

제품 기능 목록	세부 작업 항목(일)	1일	2일	3일	4일	5일	6일	7일	8일	9일	10일	11일	12일	13일	14일	15일	16일	17일
품질(9장)	ISO/IEC 9126(5일)	1	1	1				1	1									
	ISO/IEC 14598(4일)				1	1	1						1					
	ISO/IEC 12119(3일)									1	1	1						
	CMMI(5일)													1	1	1	1	1
총 남은 시간		16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

5. 스크럼 방식에서 사용되는 용어(6)

■ 스프린트 구현 목록sprint backlog

- 각각의 스프린트 주기에서 개발할 작업 목록
- 세부 작업 항목과 작업자, 예상 작업 시간 등에 관한 정보를 작성

표 2-5 스프린트 구현 목록의 예

작업 목록	세부 작업 항목	예상 작업 시간	작업자
품질	제품 품질의 ISO/IEC 9126에 대해 기술한다.	5일	
	제품 품질의 ISO/IEC 14598에 대해 기술한다.	4일	
	제품 품질의 ISO/IEC 12119에 대해 기술한다.	3일	
	프로세스 품질의 CMMI에 대해 기술한다.	5일	
테스트	블랙박스 테스트에 대해 기술한다.	4일	
	화이트박스 테스트에 대해 기술한다.	4일	

5. 스크럼 방식에서 사용되는 용어(7)

- **소멸 차트**burndown chart
  - 시간이 지남에 따라 소멸되고 남은 것을 표현
  - 계획 대비 작업이 어떻게 진행되고 있는지를 날짜별로 남은 작업량으로 표현

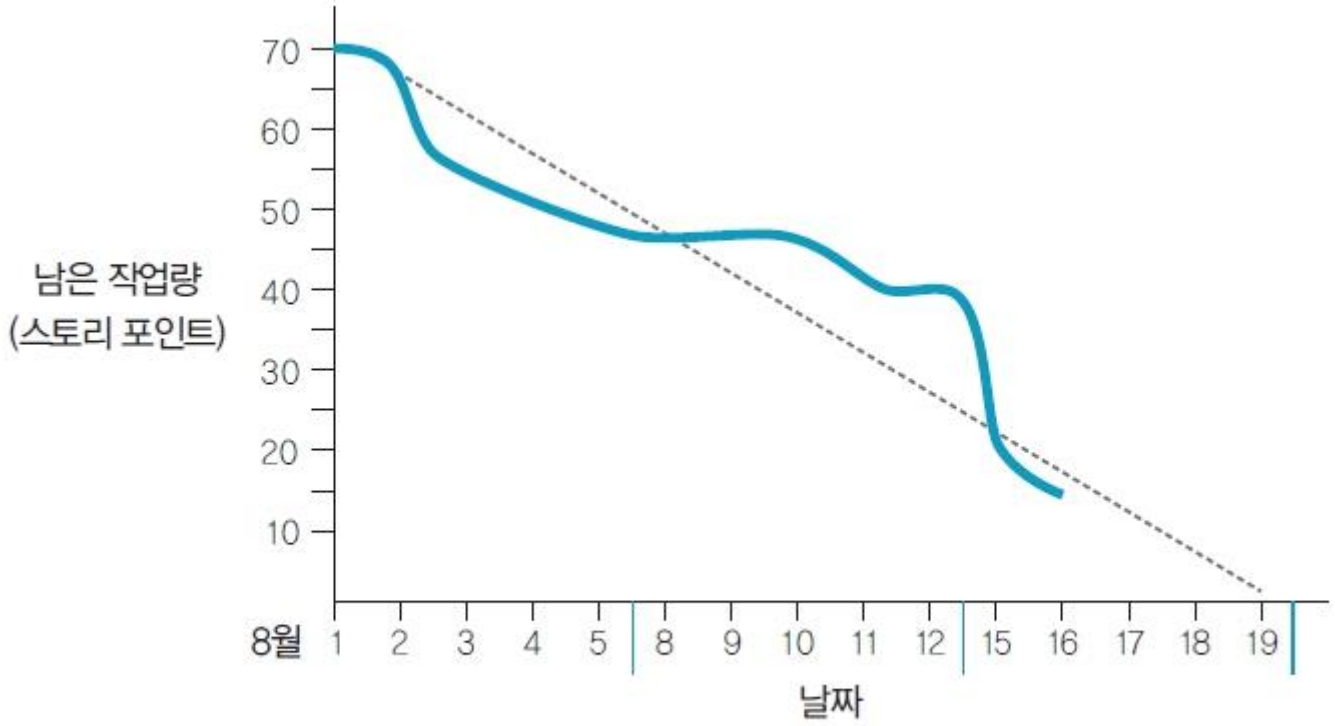


그림 2-29 소멸 차트

6. 스크럼 방식에서의 회의(1)

- **스프린트 계획 회의sprint planning meeting**
  - 전체적인 스프린트 계획 회의
    - 가장 높은 순위의 항목에 관심
    - 그 배경과 목표에 대해 팀원들과 토의
    - 제품 책임자의 의도 파악
  - 세부적인 스프린트 계획 회의
    - 우선순위가 높은 항목의 구현 방법에 대한 구체적인 작업 계획을 세움
    - 결정된 개발 항목에 대한 스프린트 구현 목록 작성
    - 정해진 작업 수행 소요 시간 추정

6. 스크럼 방식에서의 회의(2)

- **일일 스크럼 회의**daily scrum meeting
  - 스프린트 기간에 하는 회의이며, 매일, 서서, 짧게(15분 정도) 함
  - 진행 상황만 점검하고, 스프린트 작업 목록을 잘 개발하고 있는지 확인
  - 모든 팀원이 참석하고, 한 사람씩 어제 한 일을 얘기
  - 한 사람씩 오늘 할 일과 문제점 및 어려운 점 정도만 얘기
  - 매일 완료된 세부 작업 항목을 완료 상태로 옮겨 스프린트 현황판 업데이트
  - 개별 팀원에 대한 진척 상태를 확인
  - 그날의 남은 작업량을 소멸 차트에 표시

6. 스크럼 방식에서의 회의(3)

■ 스프린트 현황판task board

- 개발 팀의 개발 현황(진척도, 남은 작업, 진행 속도)을 나타냄

그림 2-30 스프린트 현황판의 예

■ 최종 제품finished work

- 모든 스프린트 주기가 끝나면 제품 기능 목록에서 개발하려고 했던 제품이 완성

6. 스크럼 방식에서의 회의(4)

- **스프린트 검토회의**sprint review
  - 하나의 스프린트 반복 주기(2~4주)가 끝났을 때 생성되는 실행 가능한 제품에 대해 검토
  - 스프린트 목표를 달성했는지 작업 진행과 결과물을 확인
  - 전체 흐름을 확인하여 비즈니스 가치를 점검

- **스프린트 회고**sprint retrospective
  - 스프린트에서 수행한 활동과 개발한 것을 되돌아 봄
  - 개선점은 없는지, 팀이 정한 규칙이나 표준을 잘 준수했는지 등을 검토
  - 문제점을 확인하고 기록하는 정도로만 진행
  - 추정 속도와 실제 속도를 비교해보고, 차이가 크면 그 이유를 분석
  - 프로세스 품질은 측정하지 않음

6. 스크럼 방식에서의 회의(5)

- **배포 목록**release backlog
  - 이번 배포 본에 포함하기로 결정한 항목

표 2-6 배포 목록

장	내역	작업 기간	스프린트 주기
1장	소프트웨어 공학의 개요	10일	스프린트 1
2장	소프트웨어 개발 프로세스	20일	스프린트 2
3장	계획	20일	스프린트 3
총 남은 시간		100일	
배포 스프린트			총 3스프린트
배포 날짜			2016년 8월 30일

7. 스크럼 방식의 진행 절차

표 2-7 스크럼 방식의 진행 절차

단계	수행 목록	내용
1	제품 기능 목록 작성	• 요구 사항 목록에 우선순위를 매겨 제품 기능 목록 작성
2	스프린트 계획 회의	• 스프린트 구현 목록 작성 • 스프린트 개발 시간 추정
3	스프린트 수행	• 스프린트 개발 • 일일 스크럼 회의 • 스프린트 현황판 변경 • 소멸 차트 표시
4	스프린트 개발 완료	• 실행 가능한 최종 제품 생산
5	스프린트 완료 후	• 스프린트 검토 회의 • 스프린트 회고 • 두 번째 스프린트 계획 회의

8. 제품 책임자, 스크럼 마스터, 스크럼 팀의 역할

표 2-8 제품 책임자, 스크럼 마스터, 스크럼 팀의 역할

담당자	역할
제품 책임자product owner	• 제품 기능 목록을 만듦. • 비즈니스 관점에서 우선순위와 중요도를 매기고 새로운 항목을 추가함. • 스프린트 계획 수립 시까지만 역할을 수행하고, 스프린트가 시작되면 팀 운영에 관여하지 않음.
스크럼 마스터scrum master	• 제품 책임자를 돕는 조력자 • 업무를 배분만 하고, 일은 강요하지는 않음. • 스크럼 팀이 스스로 조직하고 관리하도록 지원함. • 개발 과정에서 스크럼의 원칙과 가치를 지키도록 지원함. • 개발 과정에 방해될 만한 요소를 찾아 제거함.
스크럼 팀scrum team	• 팀원은 보통 5~9명으로 구성되며, 사용자 요구 사항을 사용자 스토리로 도출하고 이를 구현함. • 기능을 작업 단위로 나누고, 일정이나 속도를 추정해서 제품 책임자에게 알려줌. • 하나의 스프린트에서 생산된 결과물을 제품 책임자에게 시연함. • 매일 스크럼 회의에 참여하여 진척 상황을 점검함.

9. 스크럼 방식의 장점

- 실행 가능한 제품을 통해 사용자와의 충분한 의견 조율 가능
- 일일 회의를 통한 팀원들 간의 신속한 협조와 조율 가능
- 일일 회의 시 직접 자신의 일정 발표를 통한 업무 집중 환경 조성
- 다른 개발 방법론들에 비해 단순하고 실천 지향적
- 팀의 문제를 해결할 수 있는 스크럼 마스터가 존재함
- 프로젝트 진행 현황을 통해 신속하게 목표와 결과 추정 가능, 목표에 맞는 변화 시도 가능

10. 스크럼 방식의 단점

- **추가 작업 시간 필요**
  - 반복 주기가 끝날 때마다 실행 가능하거나 테스트할 수 있는 제품을 만들어야 하기 때문
- **일일 스크럼 회의를 15분 안에 마쳐야 함**
  - 길어지는 회의 시간으로 인한 작업의 방해
- **투입 공수 불측정에 따른 효율성 평가 불가**
  - 투입 공수(인력) 불측정으로 인해 얼마나 효율적으로 수행되었는지 모름
- **프로세스 품질 평가 불가**
  - 프로세스 품질 미평가로 인한 품질 관련 활동이 미약하고 품질의 정도를 알 수 없음

:

Thank You