Mc Graw Hill Education

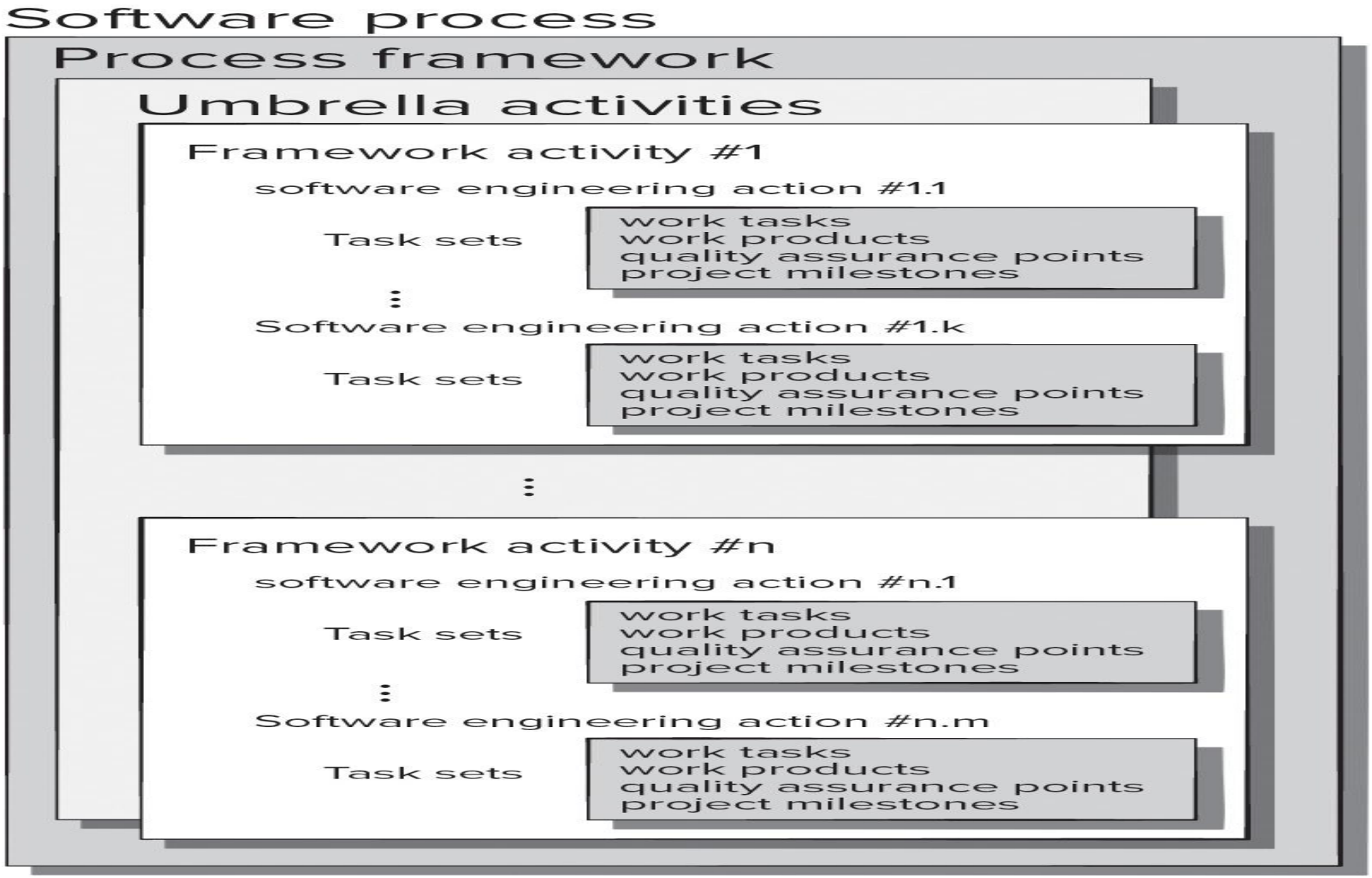
Because learning changes everything.®

Chapter 2

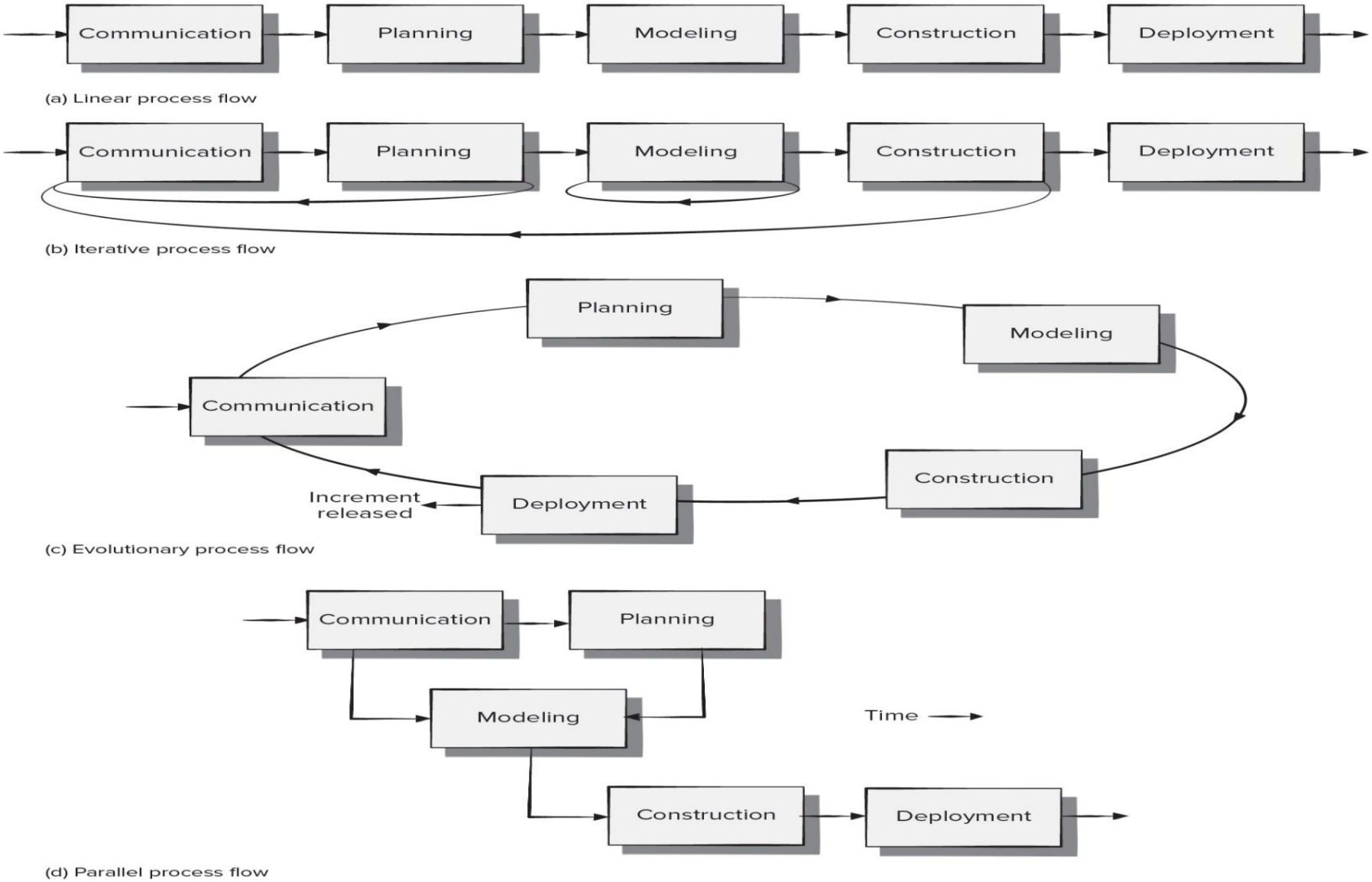
Process Models

Part 1 - The Software Process

Generic Process Model



Process Flow



Identifying a Task Set

A task set defines the actual work to be done to accomplish the objectives of a software engineering action.

A task set is defined by creating several lists:

- A list of the tasks to be accomplished.
- A list of the work products to be produced.
- A list of the quality assurance filters to be applied.

Process Assessment and Improvement

- The existence of a software process is no guarantee that software will be delivered on time, or meet the customer's needs, or that it will exhibit long-term quality characteristics.
- Any software process can be assessed to ensure that it meets a set of basic process criteria that have been shown to be essential for successful software engineering.
- Software processes and activities should be assessed using numeric measures or software analytics (metrics).

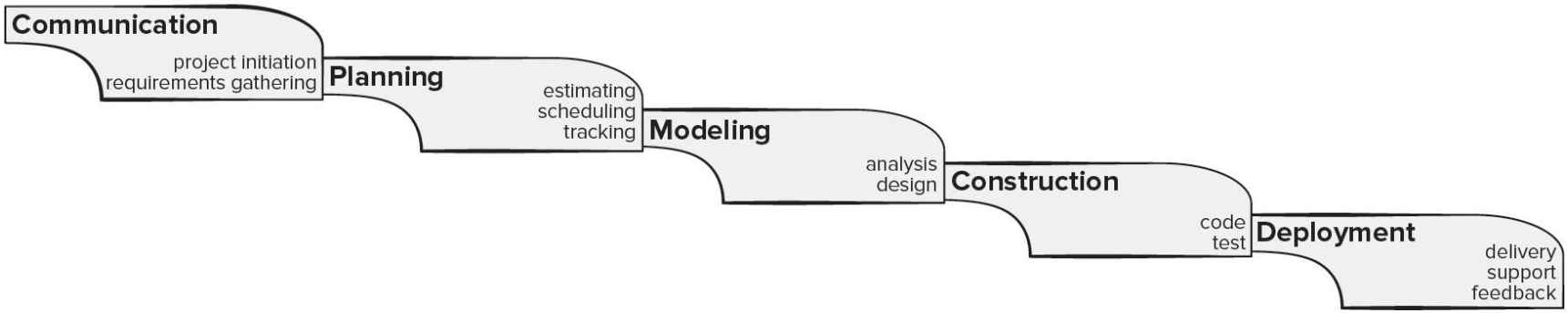
Prescriptive Process Models [1]

Prescriptive process models advocate an orderly approach to software engineering.

That leads to a two questions:

- If prescriptive process models strive for structure and order, are they appropriate for a software world that thrives on change?
- If we reject traditional process models and replace them with something less structured, do we make it impossible to achieve coordination and coherence in software work?

Waterfall Process Model



Pros

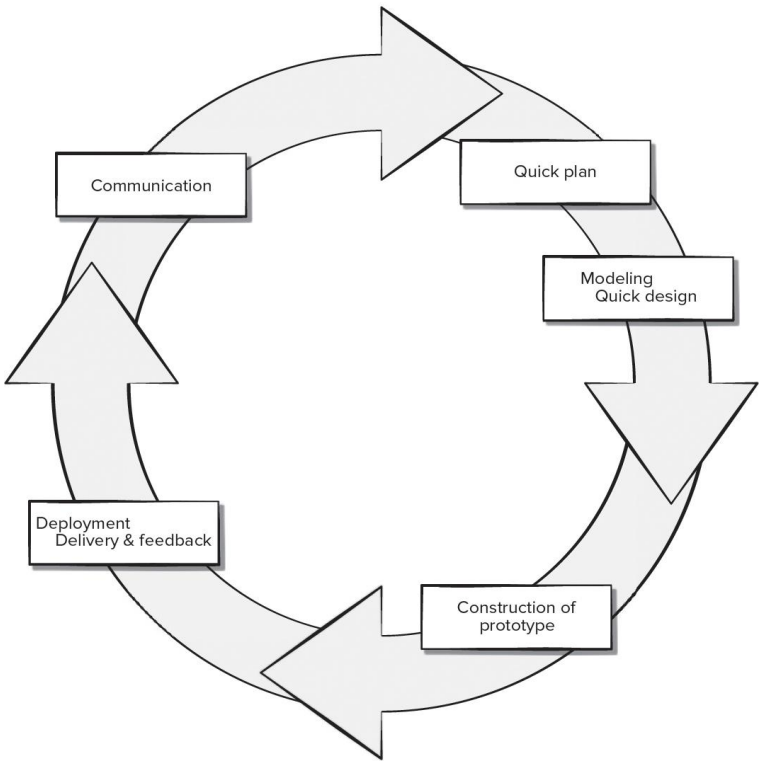
- It is easy to understand and plan.
- It works for well-understood small projects.
- Analysis and testing are straightforward.

Cons

- It does not accommodate change well.
- Testing occurs late in the process.
- Customer approval is at the end.

Access the text alternative for slide images.

Prototyping Process Model



Pros

- Reduced impact of requirement changes.
- Customer is involved early and often.
- Works well for small projects.
- Reduced likelihood of product rejection.

Cons

- Customer involvement may cause delays.
- Temptation to “ship” a prototype.
- Work lost in a throwaway prototype.
- Hard to plan and manage.

Access the text alternative for slide images.

프로토타입 프로세스 모델

- 프로토타입 모델(Prototype Model)
  - 점진적 모델의 한 유형으로, 개발 및 배포, 사용자 만족도 측정, 조정 및 수정을 통한 시스템 개발 전략을 기반으로 소프트웨어 시스템 개발
  - 기능 메뉴(혹은 버튼)만 포함하는 사용자 인터페이스의 원형을 보여주거나 사용자에게 중요하다고 판단되는 핵심 모듈만 우선적으로 개발하여 사용자에게 제공하고 이를 통해 사용자의 요구사항을 만족했는지 여부를 판단하고, 이를 바탕으로 최종 시스템을 구현

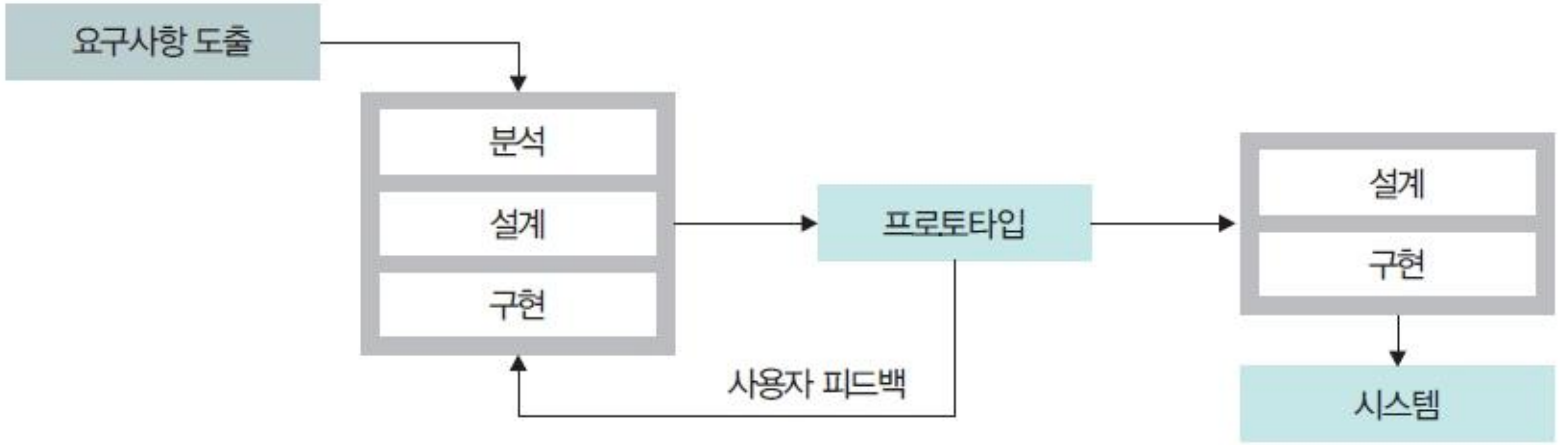


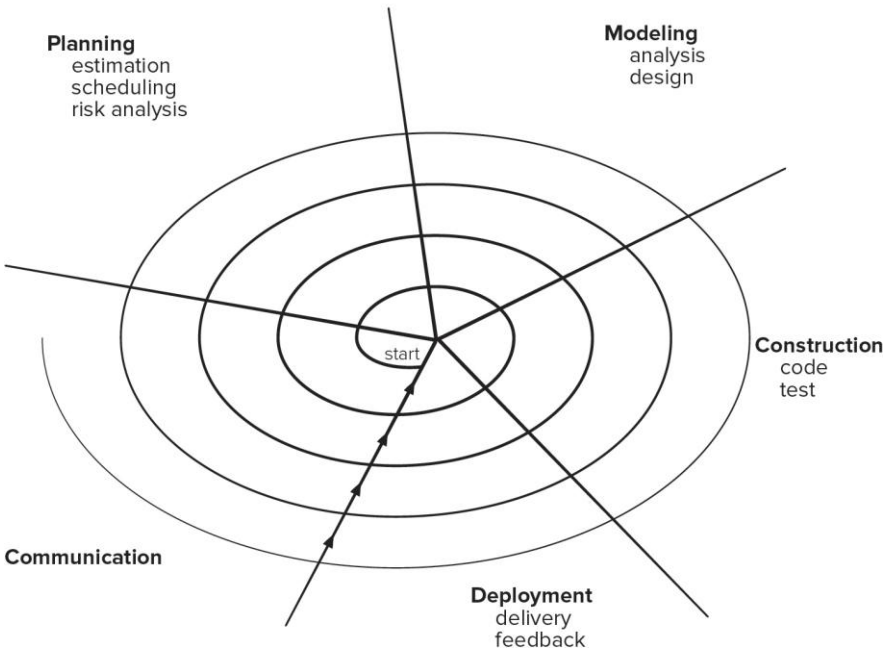
그림 3-5 프로토타입 모델에 의한 소프트웨어 개발 프로세스

프로토타입 프로세스 모델

- 프로토타입 모델의 이점
  - 사용자 요구사항을 검증하여 개발 비용과 시간을 단축할 수 있다.
  - 프로토타입을 통해 사용자와 개발자 혹은 개발자 간 의사소통이 이루어져 상호 동일한 개념을 확보할 수 있다.
  - 소프트웨어를 개발하는 동안 보다 빠른 시점에서 오류 탐지가 가능하다.

- 프로토타입 모델의 문제점
  - 사용자 검증 과정 이후에 변경이 발생하는 부분을 고려하지 못한다.

Spiral Process Model



Pros

- Continuous customer involvement.
- Development risks are managed.
- Suitable for large, complex projects.
- It works well for extensible products.

Cons

- Risk analysis failures can doom the project.
- Project may be hard to manage.
- Requires an expert development team.

Access the text alternative for slide images.

나선형 프로세스 모델

- 나선형 모델(Spiral Model)
  - 소프트웨어 시스템 개발 시 위험을 최소화하기 위해 점진적으로 전체 시스템으로 개발해나가는 모델로 소프트웨어 개발 과정이 반복적이고 점진적으로 진행되는 나선 모양
  - 목표 설정→위험 분석→구현 및 테스트→평가 및 다음 단계 수립의 활동 반복

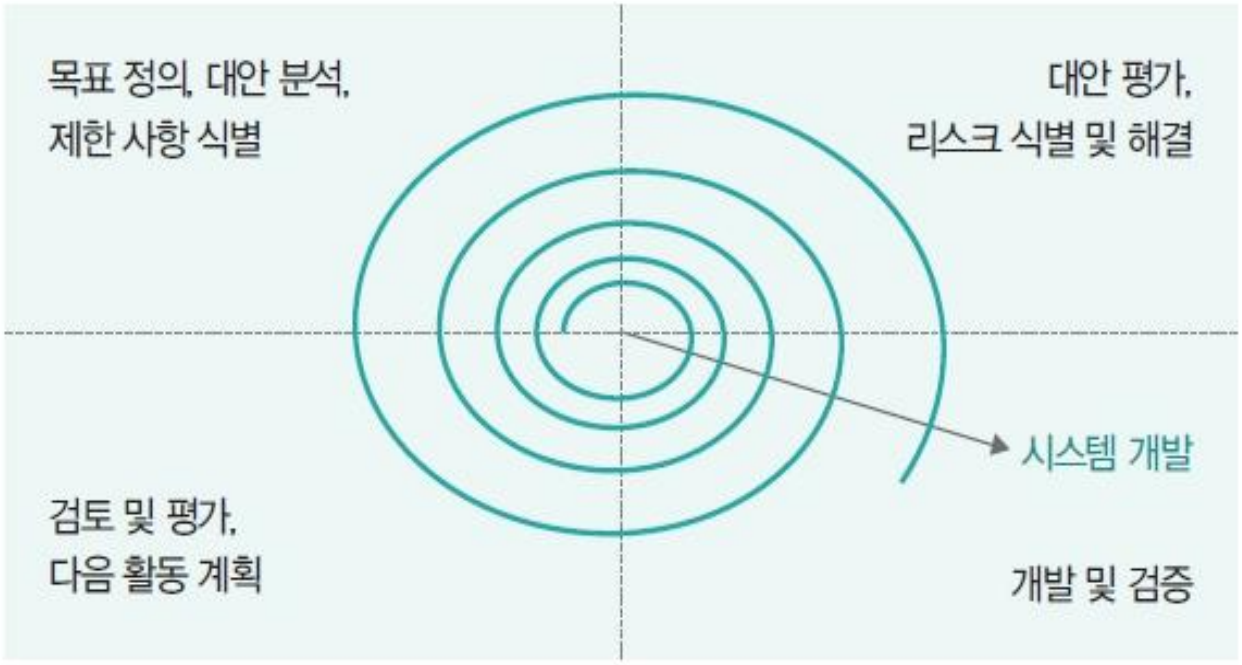


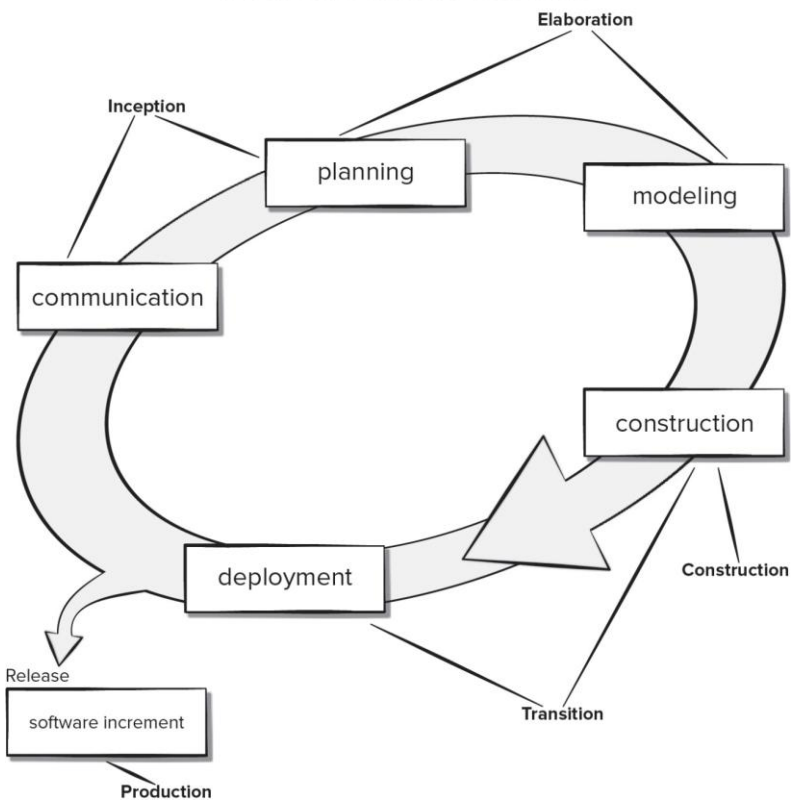
그림 3-6 나선형 모델에 의한 소프트웨어 개발 프로세스

나선형 프로세스 모델

- 나선형 모델의 장점
  - 체계적인 위험 관리로 위험성이 큰 프로젝트를 수행하기에 적합하다.
  - 사용자 요구사항을 더욱 상세히 적용할 수 있다.
  - 변경되는 요구사항을 반영하기 쉽다.
  - 최종 개발된 소프트웨어 시스템에 대한 사용자 만족도와 품질이 높아진다.

- 나선형 모델의 문제점
  - 프로젝트 기간이 길다.
  - 반복되는 사이클이 많아질수록 프로젝트 관리가 어렵다.
  - 위험 분석에 따른 비용이 발생하고 위험 관리를 위한 전문적 지식이 요구된다.

Unified Process Model



Pros

- Quality documentation emphasized.
- Continuous customer involvement.
- Accommodates requirements changes.
- Works well for maintenance projects.

Cons

- Use cases are not always precise.
- Tricky software increment integration.
- Overlapping phases can cause problems.
- Requires expert development team.

Access the text alternative for slide images.