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
- 통합된 코드의 테스트는 QA 팀이 주로 담당
- 단계적인 테스트
  - 단위, 통합, 시스템(end-to-end 테스팅이 목적)

통합과 테스트

- 목적 중심 테스트
  - 스트레스(과부하) 테스트,
  - 부하 테스팅 (성능 테스트와 신뢰성 테스트 포함)
  - Usability 테스트
  - Regression 테스트
  - 베타 테스트 vs 알파 테스트.
  - 인수(Acceptance) 테스트 <➔"functional test", "acceptance test", "customer test, "story test"

설치와 유지보수

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

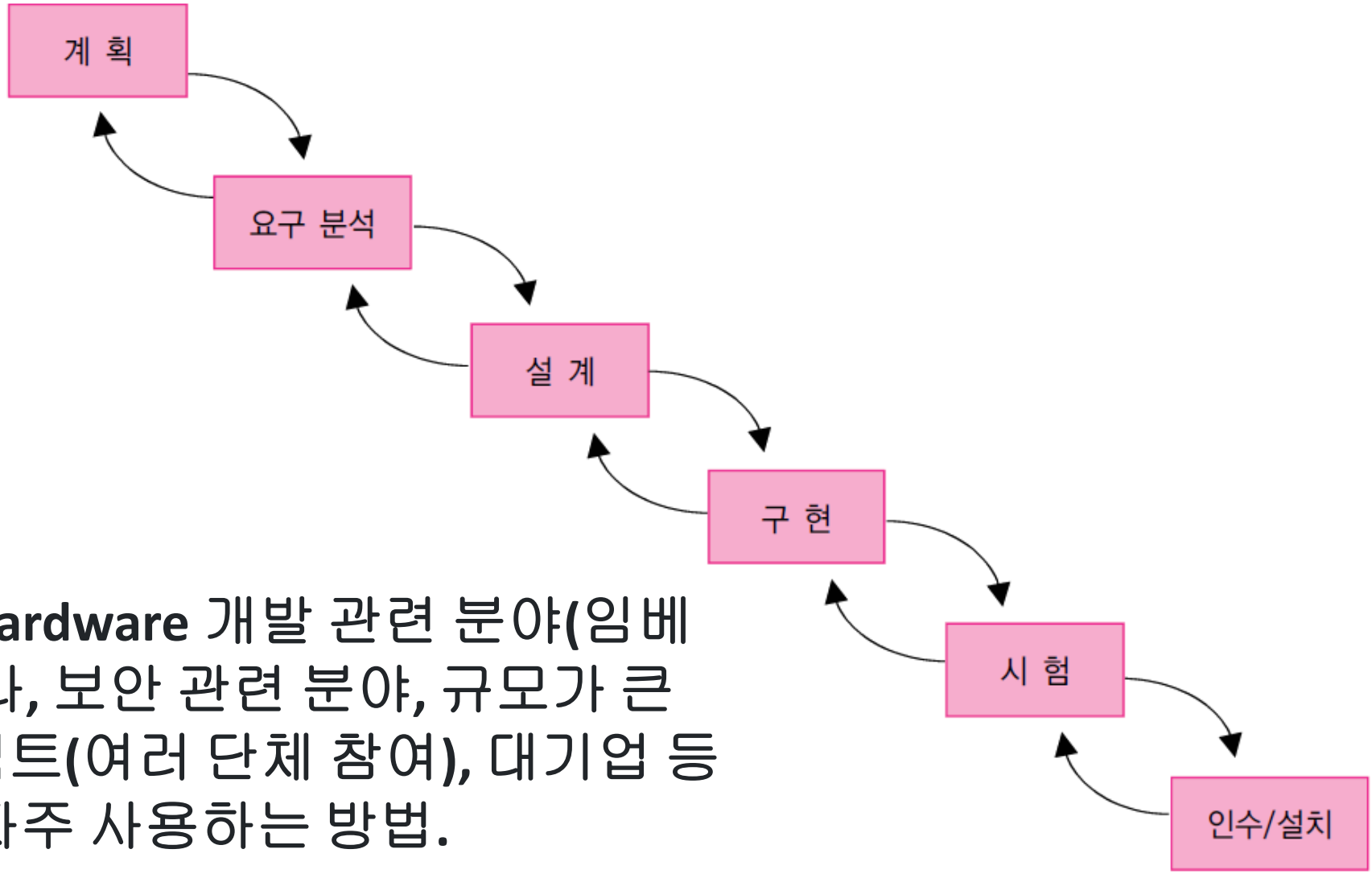
Software Process Models

- **Specification(명세)** : 소프트웨어가 어떤 기능을 해야 하는지 정리
- **Design and Implementation(설계 및 개발)**
- **Validation(검증)** : 고객이 원하는 대로 만들어졌는지 검증
- **Evolution(진화)** : 고객의 요구 변화에 맞추어 수정

Software process model classification

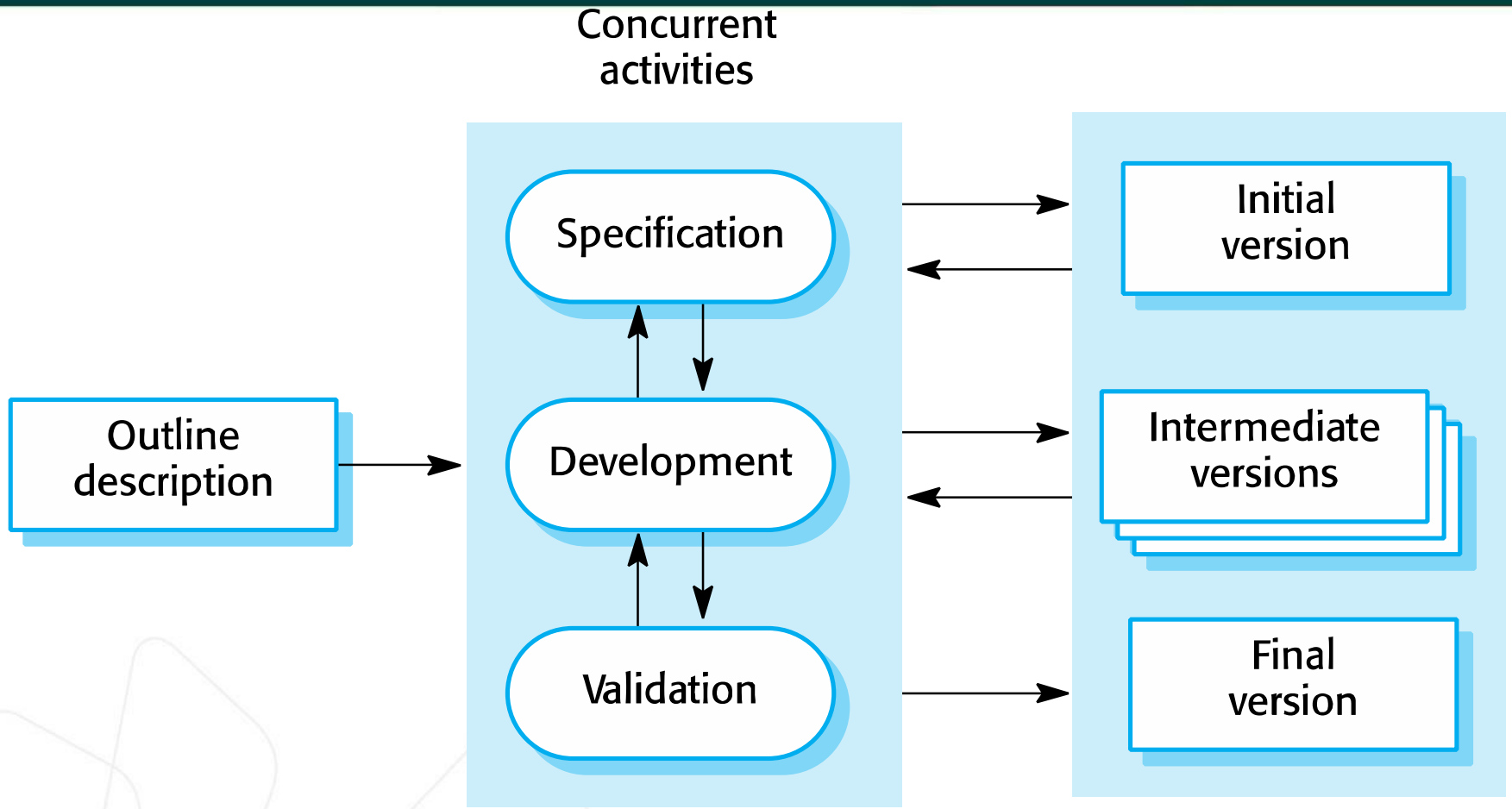
- **The waterfall model (or cascade model)**
  - Plan-driven model. Separate and distinct phases of specifi cation and development.
- **Incremental development**
  - Specification, development and validation are interleaved. May be plan-driven or agile.
- **Integration and configuration**
  - The system is assembled from existing configurable comp onents. May be plan-driven or agile.
- In practice, most large systems are developed using a process that incorporates elements from all of the process models.

The waterfall model



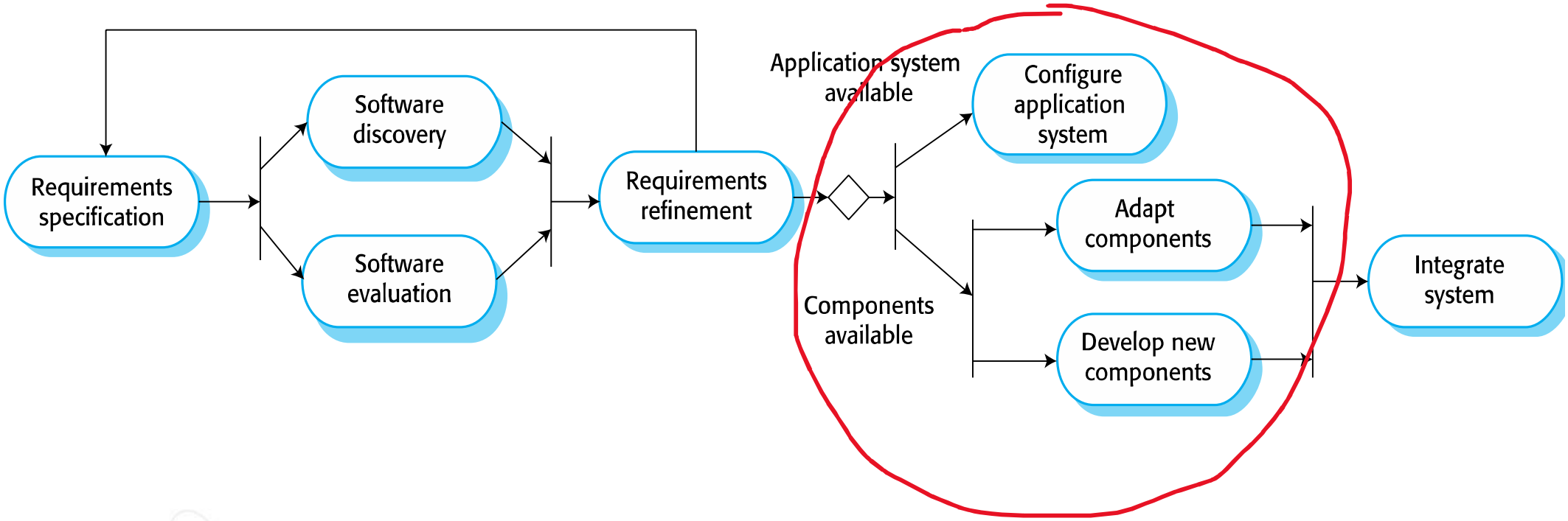
보통 **hardware** 개발 관련 분야(임베디드)나, 보안 관련 분야, 규모가 큰 프로젝트(여러 단체 참여), 대기업 등에서 자주 사용하는 방법.

Incremental development



- **requirement** 변화, **customer feedback**에 대응하기 쉬움.
- 새로운 기능이 추가될 때마다 시스템 구조는 약화.
- **Agile** 방식 에 사용 **(Plan-based** 방식에도 사용**: version up** 계획 미리 세움)

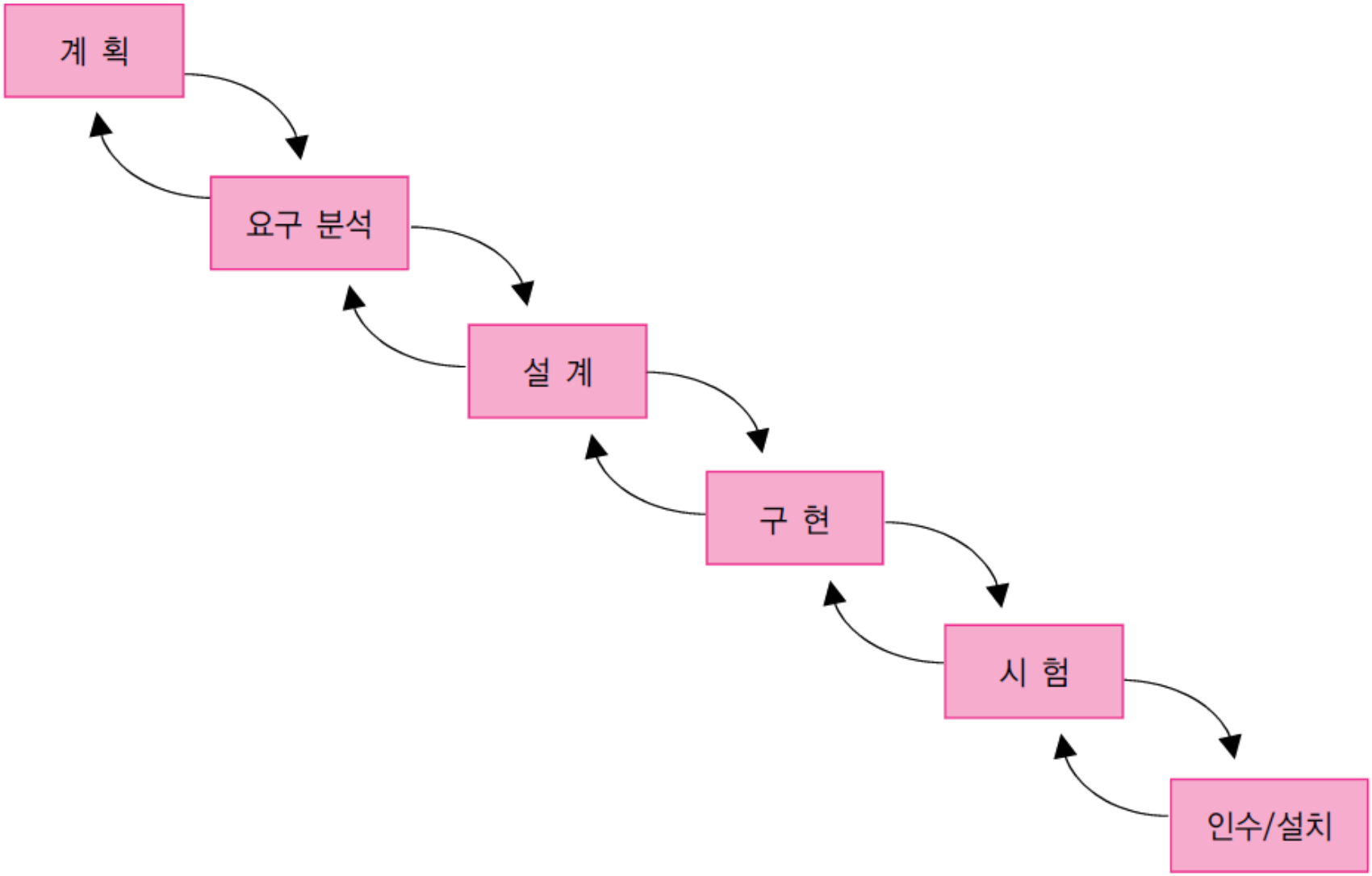
Integration and configuration



- 소프트웨어 재사용에 초점을 맞추어, 시스템을 기존에 존재하던 것들로 구성

- Types of reusable software
  - Stand-alone application systems or components (COTS-Commercial Off The Shelf) that are configured for use in a particul ar environment.
  - 상용 기성품 소프트웨어 (COTS)는 주로 제3자 입장의 업체가 사전에 만들어 놓은 소프트웨어이다. COTS는 일반 대중을 대상으로 판매하거나 임대할 수 있고 라이센스 계약도 가능하다.
  - ex) Windows DLL, game engine,
    browser add-ons, spreadsheet etc.
  - Collections of objects that are developed as a package to be in tegrated with a framework such as .NET or J2EE.
  - Web services that are developed according to service standards and which are available for remote invocation.

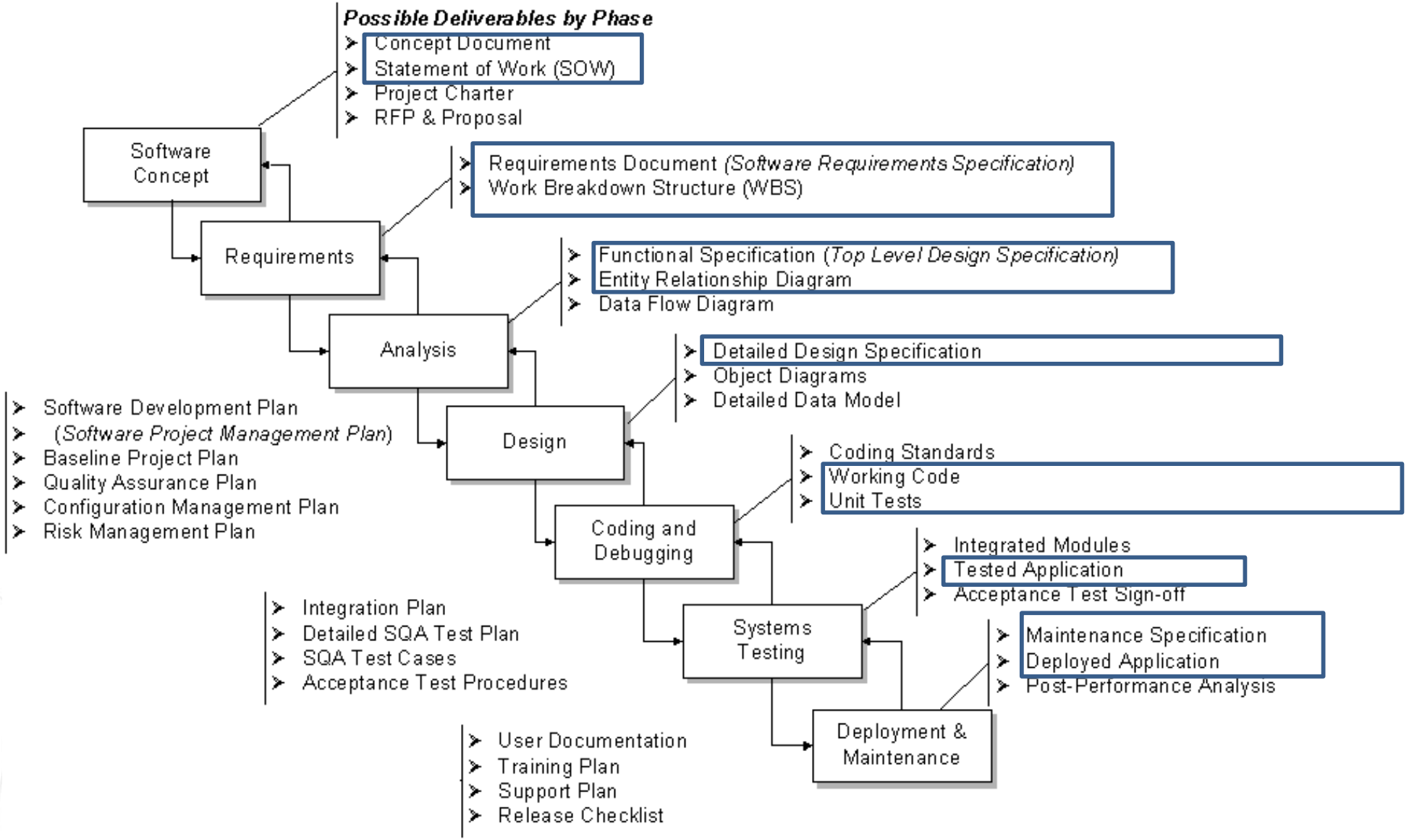
(1) 폭포수(waterfall) 모델



폭포수(waterfall) 모델

- 1970년대 소개
  - 항공 방위 소프트웨어 개발 경험으로 습득
- 각 단계가 다음 단계 시작 전에 끝나야 함
  - 순서적 - 각 단계 사이에 중첩이나 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(deliverable) 정의가 중요
- **Method**(specific tool/procedure) **vs. Methodology**(approach)

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

그외 모델

- 프로토타이핑(prototyping) 모델
- 나선형(spiral) 모델
- Unified Process(UP)
- RAD (Rapid Application Development)
- V 모델
- 애자일(Agile) 프로세스
  - Extreme Programming, Scrum, Kaban 등