개발방법론 현장적용 실무

2011. 08

목 차

I. 개발방법론의 이해
- **1.** 개 요
- **2.** 종류 및 선정기준

II. SW개발방법론 적용 및 프로세스
- **1.** IT중소기업의 개발방법론 적용
- **2.** SW개발방법론 프로세스

III. 요구분석 핵심기법 및 고려사항
- **1.** 요구분석 핵심기법
- **2.** 요구사항 단계 고려사항

IV. 설계단계 핵심기법 및 고려사항
- **1.** 시스템 설계 작성절차
- **2.** 설계기준 SW아키텍처 기법
- **3.** UML 기반 설계기법
- **4.** 설계단계 고려사항

V. 시험계획 및 실시를 위한 핵심기법
- **1.** 시험계획 및 실시 프로세스
- **2.** 시험설계 기법 및 작성 방법
- **3.** 시험평가 및 종료활동

VI. SW개발방법론 확대전략
- **1.** 관리방법론과의 연계
- **2.** Agile 프로세스의 방법론 적용
- **3.** 공공프로젝트 요구사항 감리대응기법 반영
- **4.** SW개발방법론 활용 프로세스 품질 인증
- **5.** 개인정보(PIMS) 및 보안설계, 구현 적용

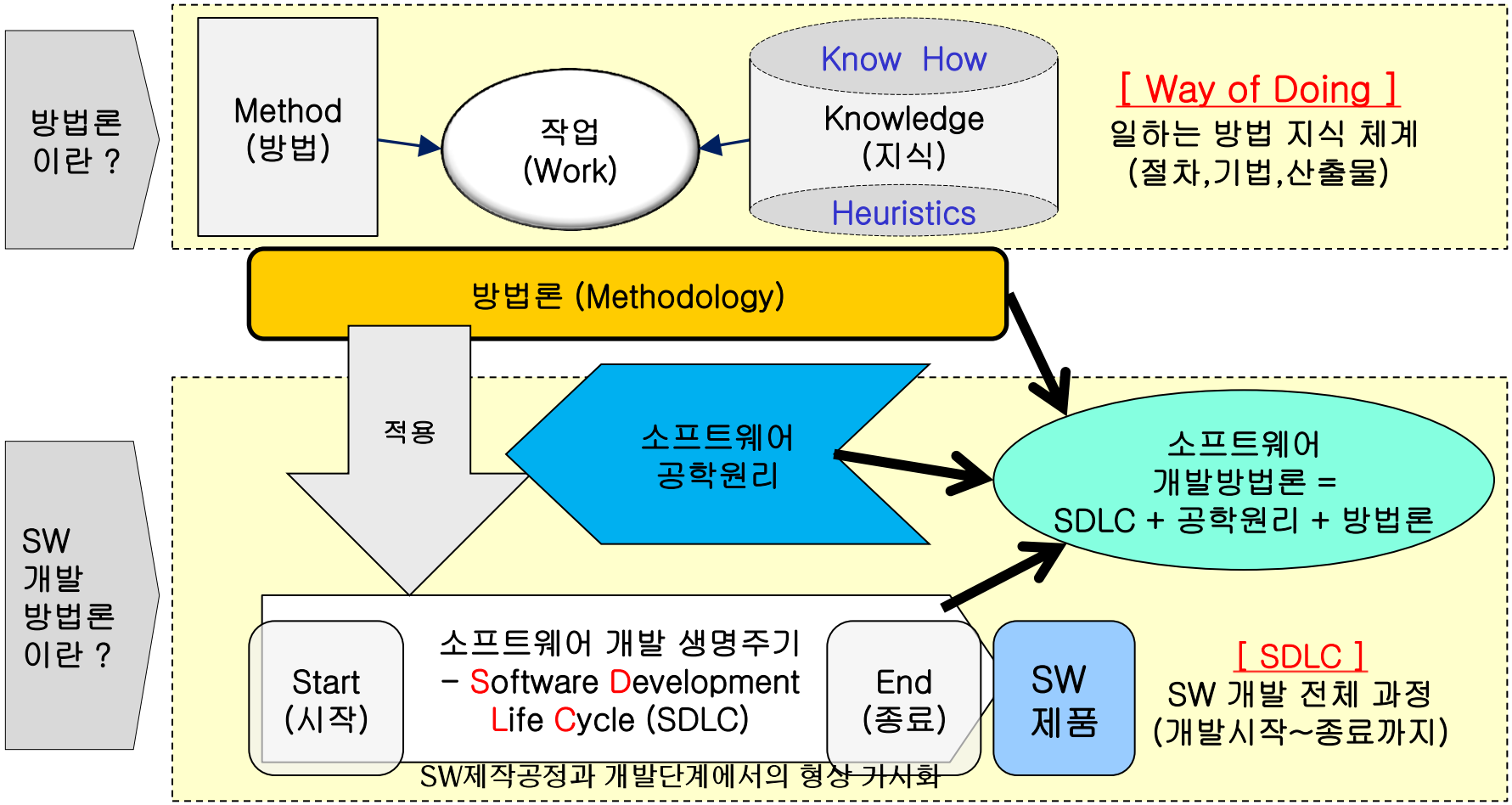
강의 일정

일정 / 내용	Day 1	Day 2	Day 3
Ⅰ. 개발방법론 의이해	▬▬▬		
Ⅱ. **SW** 개발방법론 적용 및 프로세스	▬▬		
Ⅲ. 요구분석 핵심기법 및 고려사항		▬▬	
Ⅳ. **UML**기반 설계단계		▬▬	
Ⅴ. 시험계획 및 실시를 위한 핵심기법		▬▬	▬▬
Ⅵ. **SW**개발방법론 확대전략			▬

Ⅰ. 개발방법론의 이해

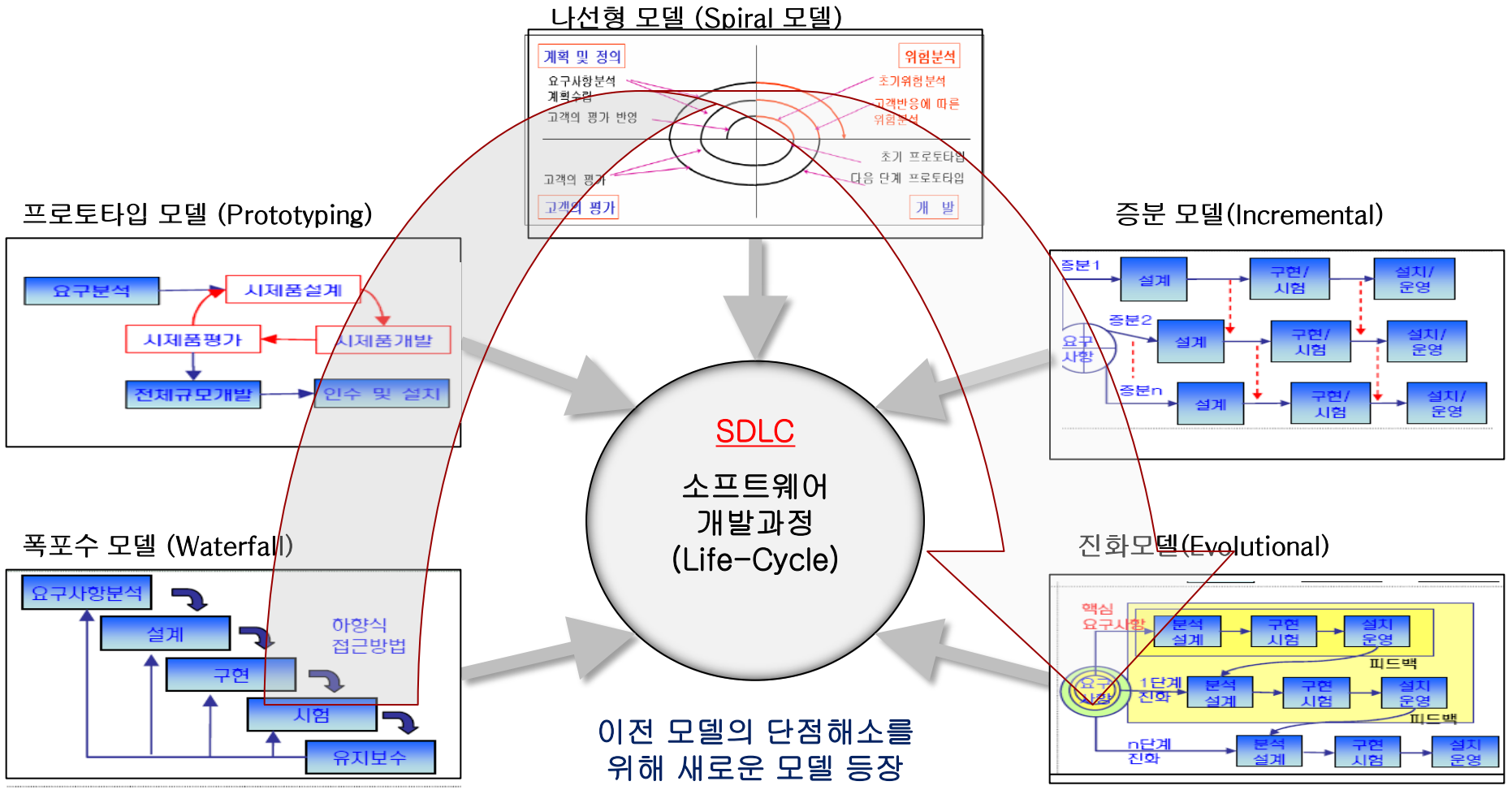
1) 개발방법론과 SDLC의 개념

소프트웨어 개발 생명주기(SDLC)는 개발하는 과정을 말하며, 개발방법론은 SDLC에 소프트웨어 공학원리를 적용하여 개발하는 방법(Method)을 설명하는 지식(Knowledge)을 의미함



2) 소프트웨어 개발 생명주기(SDLC) 모델의 유형

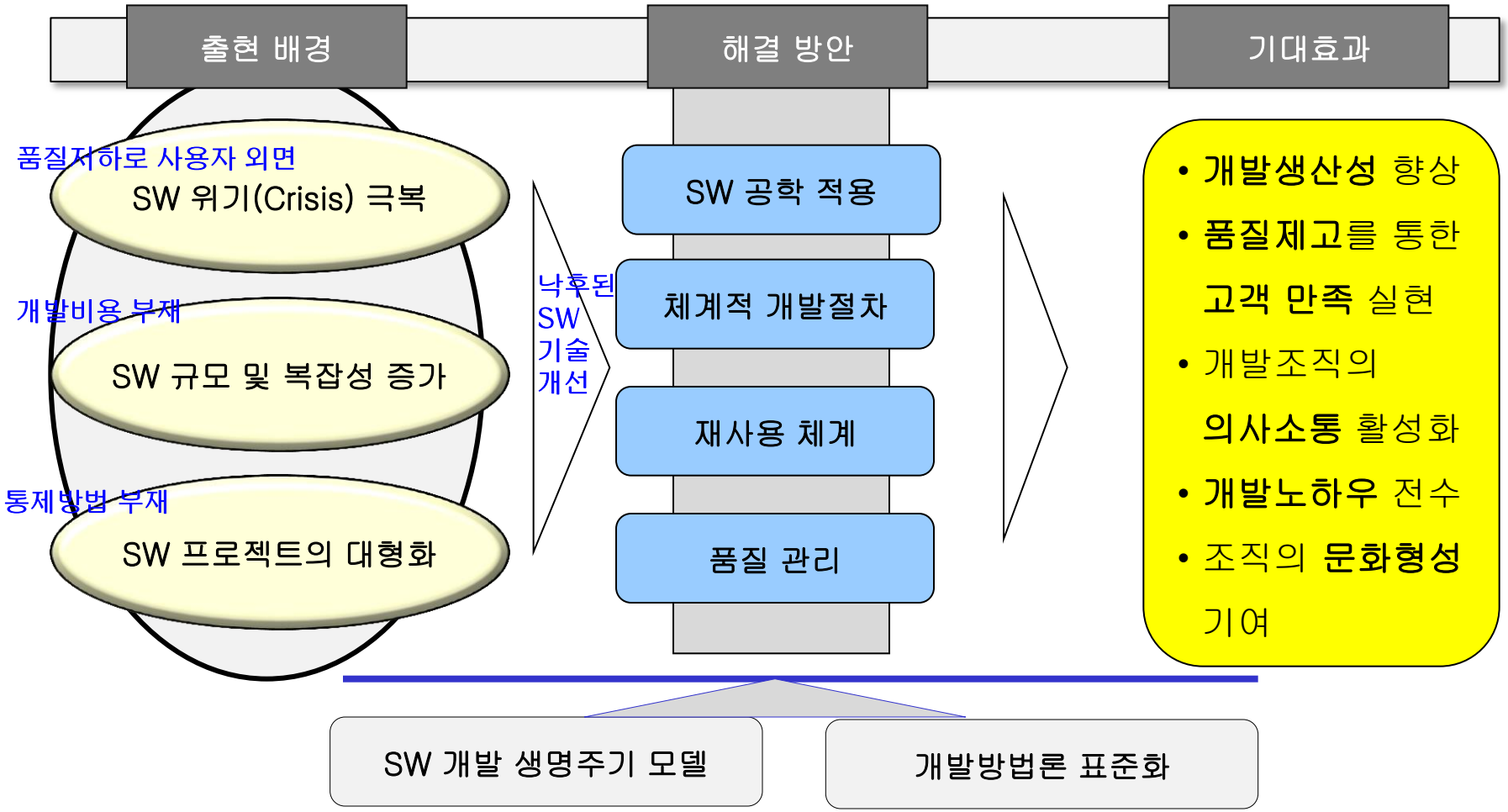
SDLC는 소프트웨어 개발하는 동안 보다 더 효과적(Effectiveness)이고 효율적(Efficiency)인 구현하기 위해 이전 모델의 단점을 보완하면서 여러가지 모델로 발전해 왔음



2) 소프트웨어 개발 생명주기(SDLC) 모델의 유형 – 계속

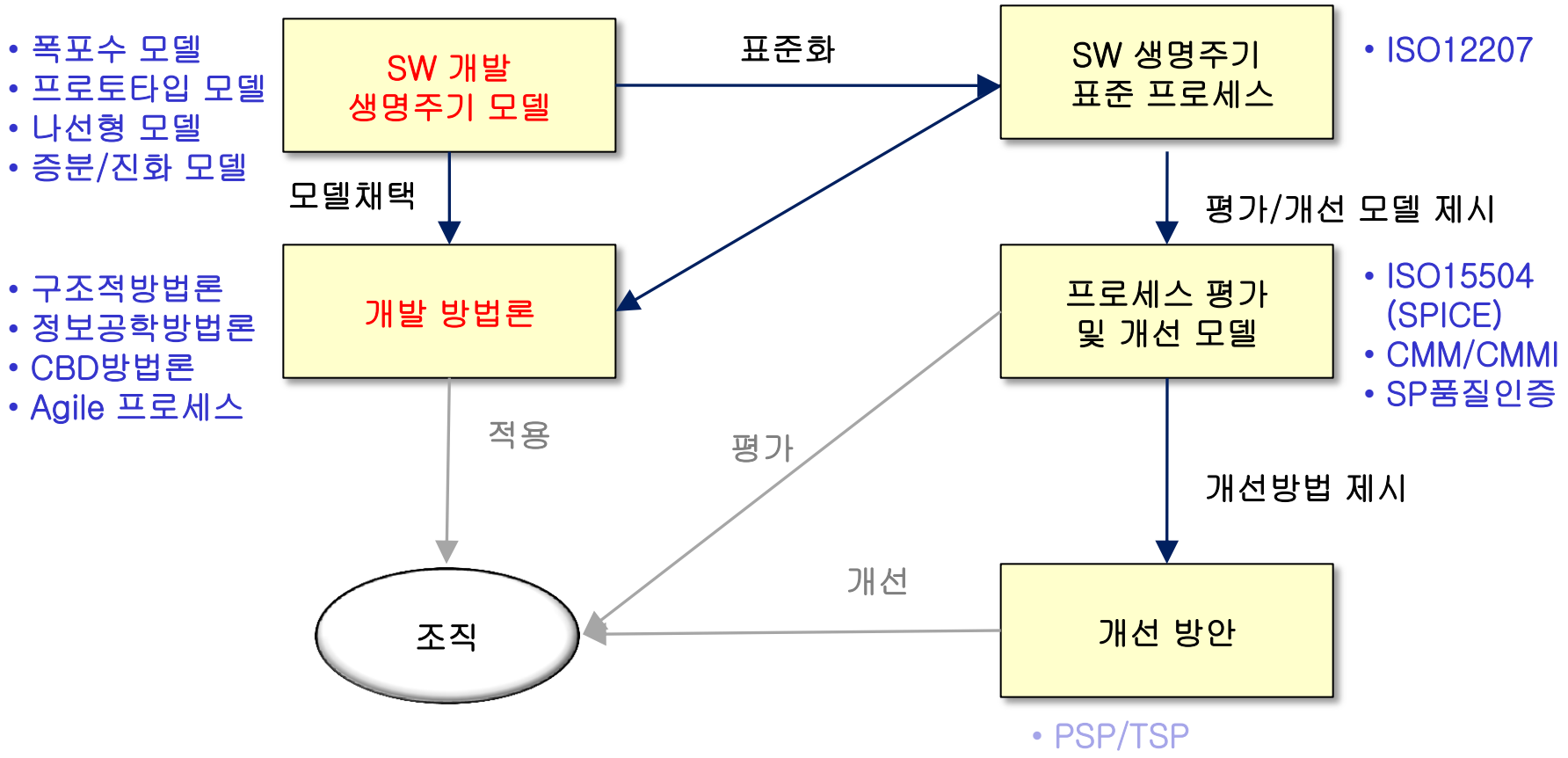
3) 개발방법론과 SDLC의 필요배경

SW산업의 위기극복과 개발활동 신뢰성 회복을 위한 대안의 하나로써 공학적기법인 소프트웨어 개발생명주기(SDLC)와 개발방법론 표준화가 필요하게 됨



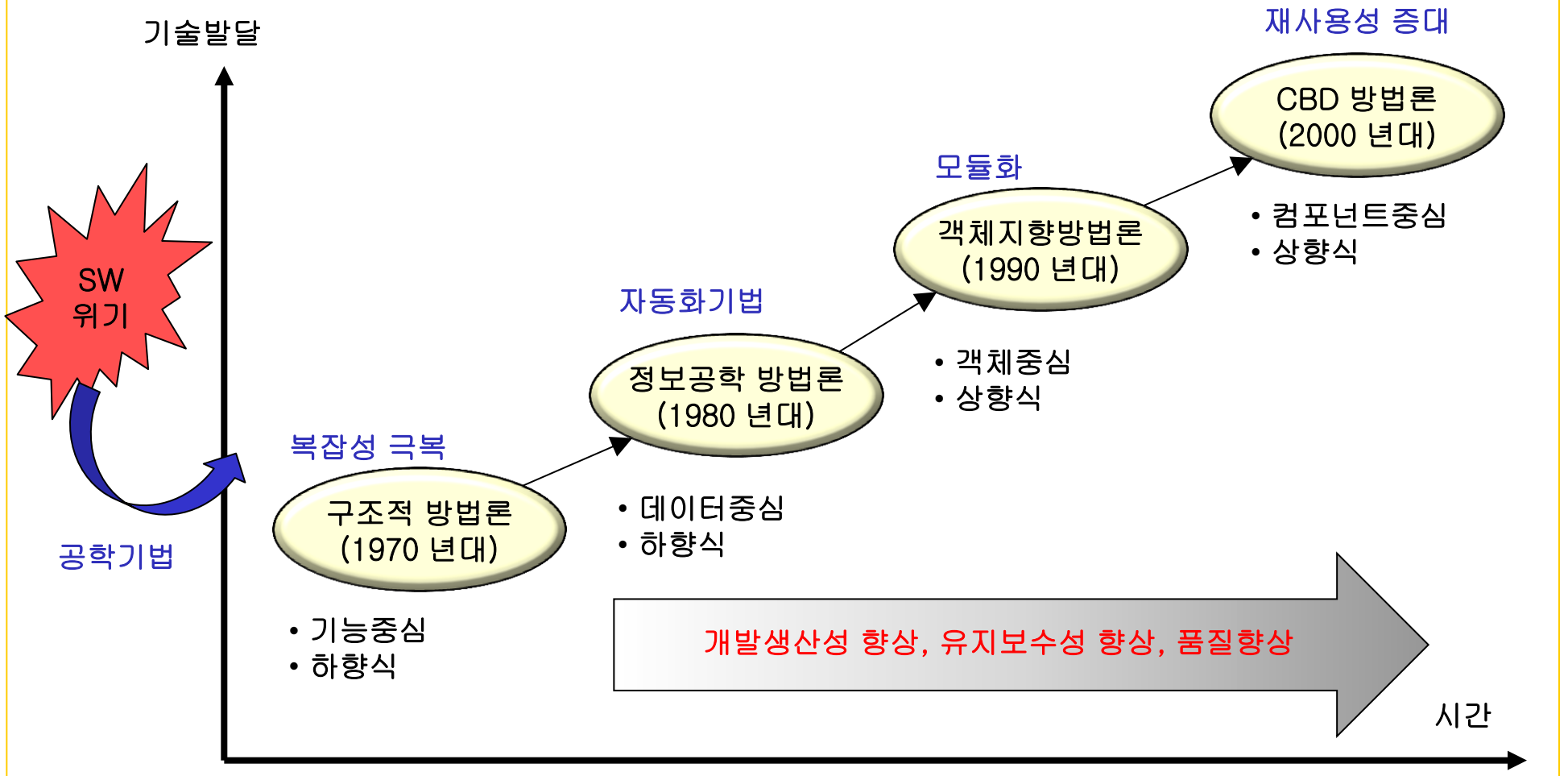
4) SDLC와 개발방법론 관련 개념

SDLC에 공학적관리를 적용한 것이 개발방법론이며, 이를 이용하여 프로세스 평가 및 개선모델을 이용한 평가, IT개발조직의 개선방법 제시 등 으로 활용할 수 있음



1) 개발방법론의 종류

개발방법론은 1960년 이후 S/W 공학과 개발 기법의 발달로 다양한 형태의 개발방법론으로 발전함



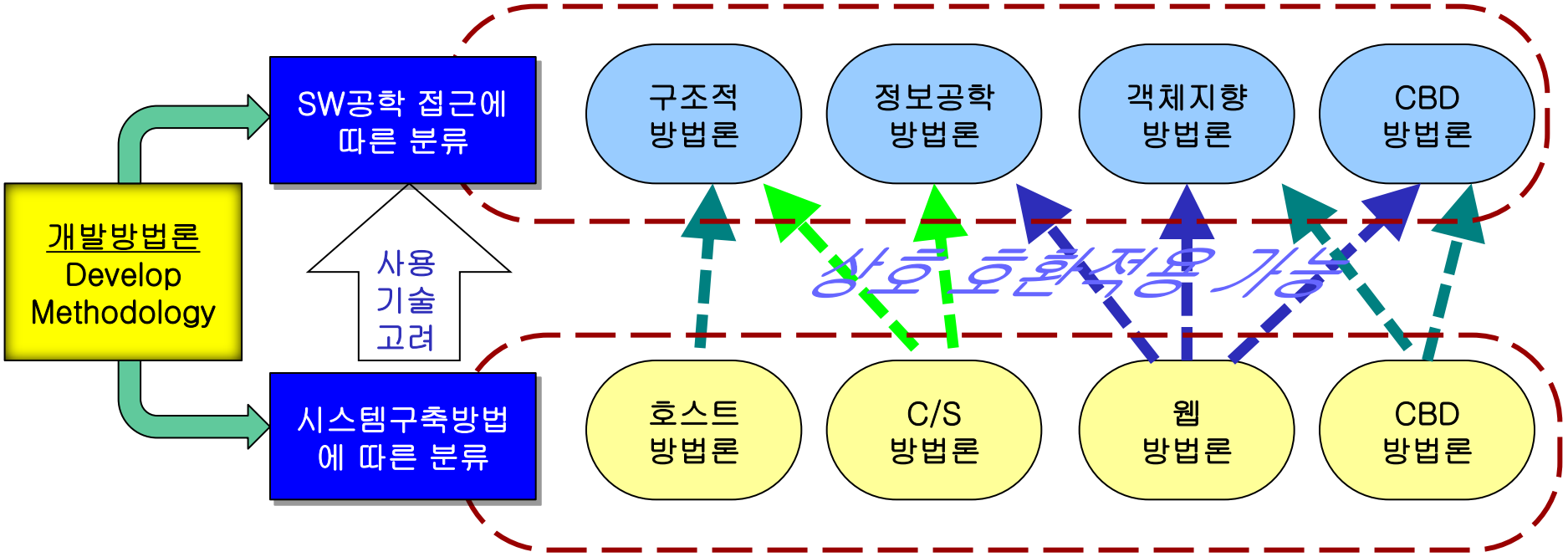
2) 개발방법론의 비교

주요 개발방법론들의 핵심적인 개념을 살펴보면 다음과 같다.

구분	구조적 방법론	정보공학 방법론	객체지향 방법론	CBD 방법론
① 시기	1970 년대	1980 년대	1990 년대	2000 년대
② 목표	비즈니스 프로세스 자동화	경영전략적 정보시스템 구축	환경변화에 유연한 정보시스템 구축	컴포넌트 개발 및 활용
③ SDLC	폭포수 모델	폭포수 모델, 프로토타이핑	반복적, 점증적 모델	반복적, 진화적 모델
④ 초점	기능 중심	자료구조 중심	객체 중심	컴포넌트 중심
⑤ 규모	소규모 적합	대규모 적합	모든 규모	모든 규모
⑥ 주요 기술	Mainframe Computing	Client Server Computing	Web Computing	Web Computing
⑦ 종류		Method/1 , IEM	RUP	마르미3, MSF/CD
⑧ 장점	• 배치방식 개발 유용, • 사례 많음	자료중심으로 비교적 안정적	• 자연스럽고 유연함 • 소스 재사용성 향상	생산성, 품질, 비용, 위험개선
⑨ 단점	• 기능은 불안정 요소 • 유지보수, 재사용성 낮음	• 어플리케이션은 여전히 기능적 설계 • 기능의 유지보수/ 재사용성 낮음	• 전문가 부족 • 기본적 SW기술 필요	• 컴포넌트 유통,평가, 인증환경 개선 필요 • 테스트 환경 부족

Tip. 개발방법론들 간의 관계

개발방법론은 일반적으로 두가지의 분류 방법(공학기법, 시스템 구축방법)을 사용하여 분류하지만, 각 기술은 상호 대체가 가능한 패턴이 있으며 이를 잘 이해하여 조직에 적용할 필요가 있음

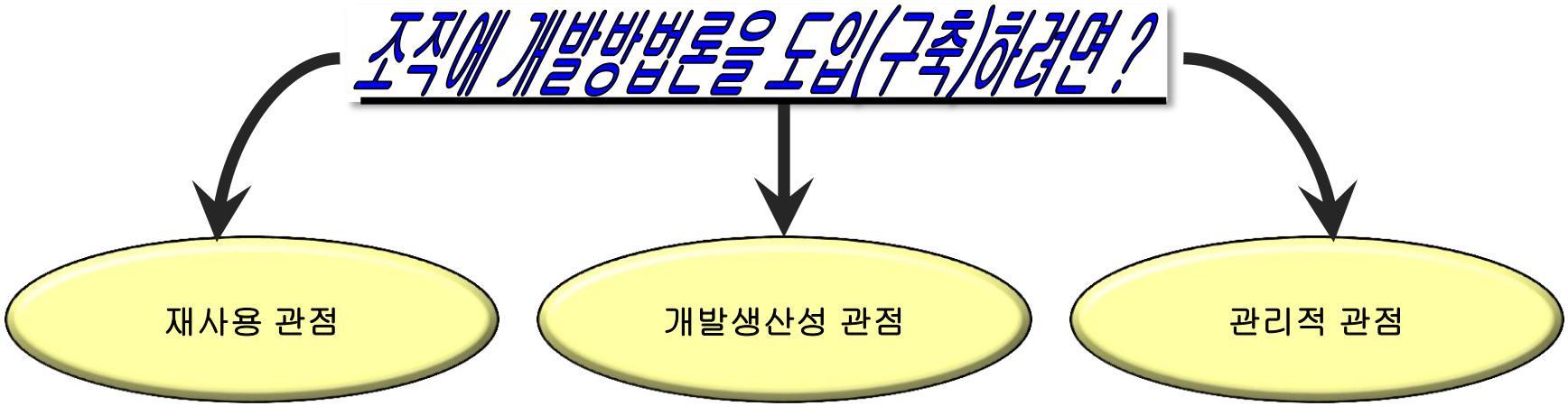


[그림] 개발방법론간의 상호 대체 가능한 일반적인 관계 패턴

시스템 구축시 주로 사용하는 SW공학기술을 고려, 타 방법론 도입하여 채택 가능

3) 개발방법론 도입시 핵심 고려사항

IT중소기업의 개발활동을 개발하기 위해 개발방법론을 도입 시에는 다음의 사항에 대한 고려가 필수적으로 선행되어야 함

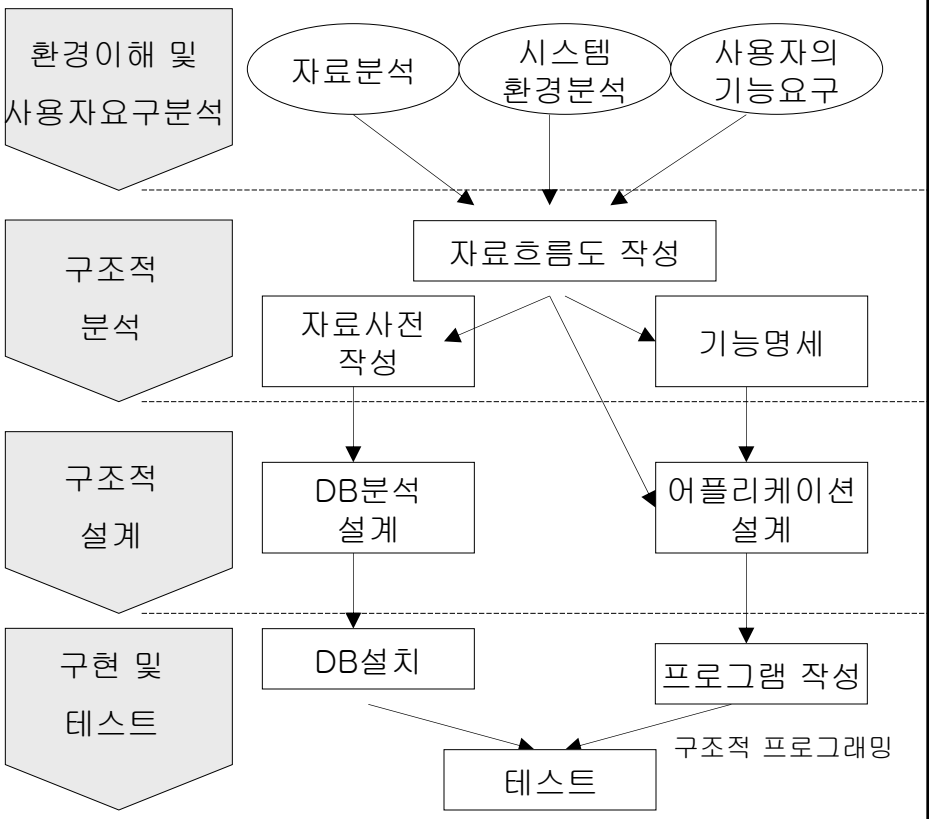


☞ 3가지 관점을 만족하기 위한
: 프로세스 도입, 지원환경 조성, 개발자 기술역량 향상 방안 과 함께 고려가 필요함

4) 개발방법론의 종류 – ① 구조적방법론 (Strutural Methodology)

기능 (업무활동) 중심의 방법론으로 정형화된 절차 및 도형중심 도구를 이용하여 사용자 요구사항 파악 및 문서화를 수행하는 방법론
– 1970년대 소프트웨어 모듈화의 활성화를 시작으로 기능적인 분할 시도, Top Down 수행

■ 개념도



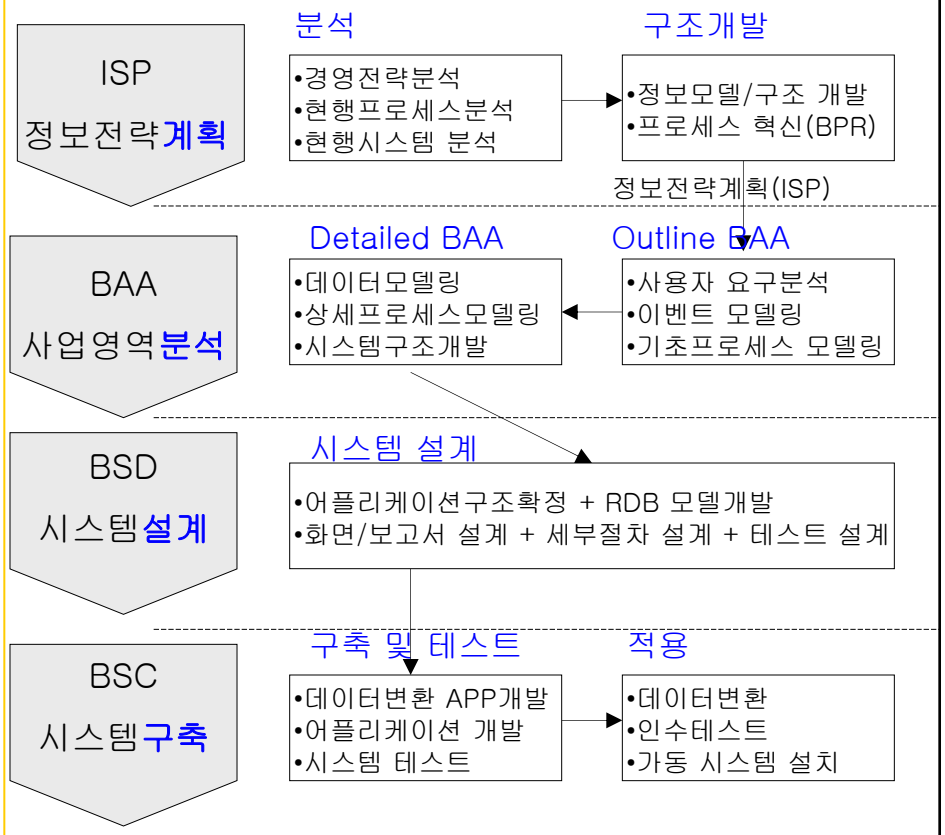
■ 기본원리

- **추상화** • 관심분야만 개념화 시켜 표현
- **정보은닉** • 하나의 모듈변경이 타 모듈 영향 없음
- **구조화** • 계층적 구조 → 수평분리, 수직분리
- **단계적 상세화** • 단계 진행하면서 점차적 요구 구체화
- **모듈화** • 시스템을 서브시스템, 프로그램, 모듈 등으로 구분, 개별 단위별 설계

4) 개발방법론의 종류 – ② 정보공학방법론 (Information Engineering Methodology)

기업전체 또는 기업의 주요부문간 정보시스템의 계획, 분석, 설계 및 구축위한 **데이터중심**의 정형화된 기술의 집합을 연계하여 응용하는 개발방법론 (현장에서 가장 많이 사용되는 방법론임)
– James Martin이 정보시스템 개발을 공학적으로 접근하기 위해 체계화

■ 개념도



■ 기본원리

- **기업중심**
  - 기업의 전략경영 지원 정보전략시스템 초점
- **ISP 중시**
  - 경영층의 요구와 견해를 시스템에 반영
- **데이터중심**
  - 변하지 않는 데이터 이용, 유지보수 줄이고 변화에 적극 대응
  - 프로세스와 데이터 분리, 상관분석 검증
- **분할과 정복**
  - 수직적(ISP→BAA→BSD→BSC) 분할
  - 수평적(데이터,프로세스,상관관계)분할
- **사용자 참여**
  - 작업 초기 사용자 참여유도 ➔ 불명확한 요구감소

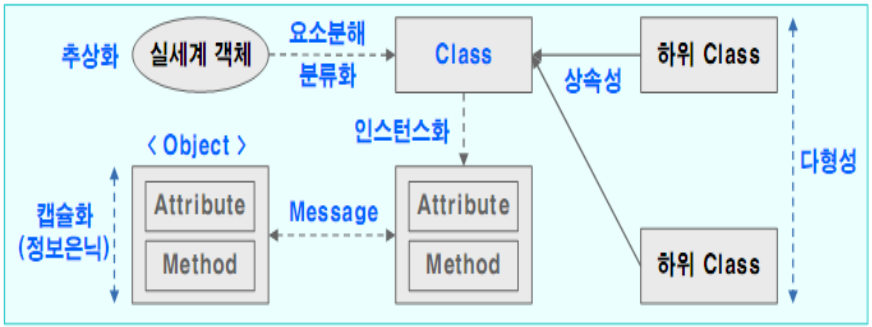
※ 핵심기술: Repository, 통합Case도구, 4GL, 프로토타이핑 (사용자참여) ☞ 공학적 접근지원을 위한 도구사용 시작

4) 개발방법론의 종류 – ③ 객체지향방법론 (Object Oriented Methodology)

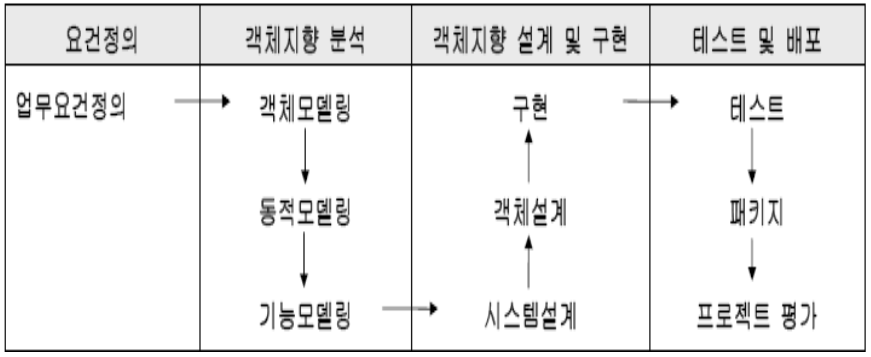
프로그램을 **객체와 객체간의 인터페이스 형태로 구성**하기 위하여 문제영역에서 객체와 클래스, 이들간의 관계를 식별하여 설계모델(객체, 동적, 기능)로 변환하는 방법론
– 복잡한 메커니즘의 현실세계를 **인간이 이해하는 방식으로** 시스템에 적용시키는 개념

■ 개념도

1) 객체지향이란?



2) 객체지향 절차도



■ 기본원리

캡슐화
- 동일한 유형 데이터와 기능을 하나로 묶어 모듈 (클래스, 객체)로 관리

정보은닉
- 모듈 내부의 정보를 외부에 숨기고, 메시지만으로 상호작용

다형성
- 동일 인터페이스, 서로 다른 응답(기능)
- 다중정의(수평적), 재정의(수직적)

상속성
- 수퍼Class 성질 서브Class에 자동부여
- 프로그램 쉽게 확장하는 강력한 수단

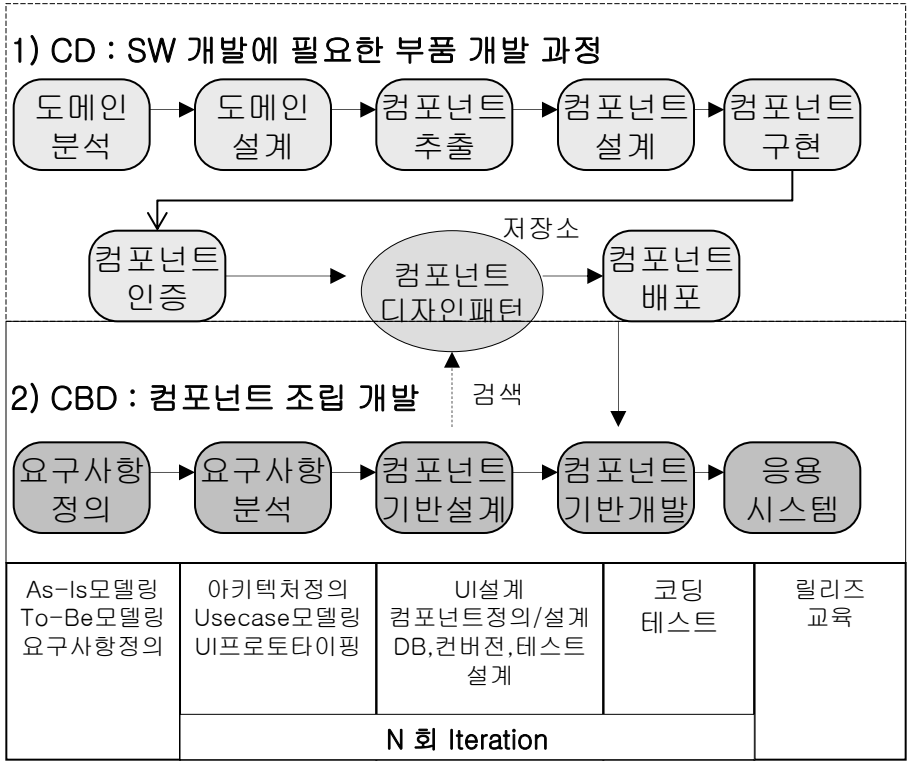
추상화
- 문제의 중요측면 주목, 상세내역 제거 (자료추상화, 기능추상화, 제어추상화)
- 복잡함 간단하게, 분석 초점 명확히

※ 객체지향의 핵심원리를 방법론에 적용함으로써, 현실세계 및 인간의 사고방식과 유사한 매커니즘을 적용

4) 개발방법론의 종류 – ④ CBD방법론

소프트웨어의 재사용 향상 및 개발기간 단축, 신뢰성 높은 개발을 목적으로 소프트웨어 개발에 Software IC개념인 컴포넌트를 생성, 조립하여 소프트웨어를 개발하는 개발방법론
– 컴포넌트 개발(CD)과 컴포넌트기반 개발(CBD)로 분류됨

■ 개념도



■ 기본원리

1) 컴포넌트 개발 관점

- **실행가능 컴포넌트**
  - 실행시간에 바인딩할 수 있도록 컴파일이 완료된 상태(실행코드)
- **패키지화**
  - 컴포넌트의 용도, 유형, 기술표준, 인터페이스 정보 명세화
- **표준화**
  - 재사용 및 교체 가능한 컴포넌트 개발 표준 준수(EJB, COM+, COM)

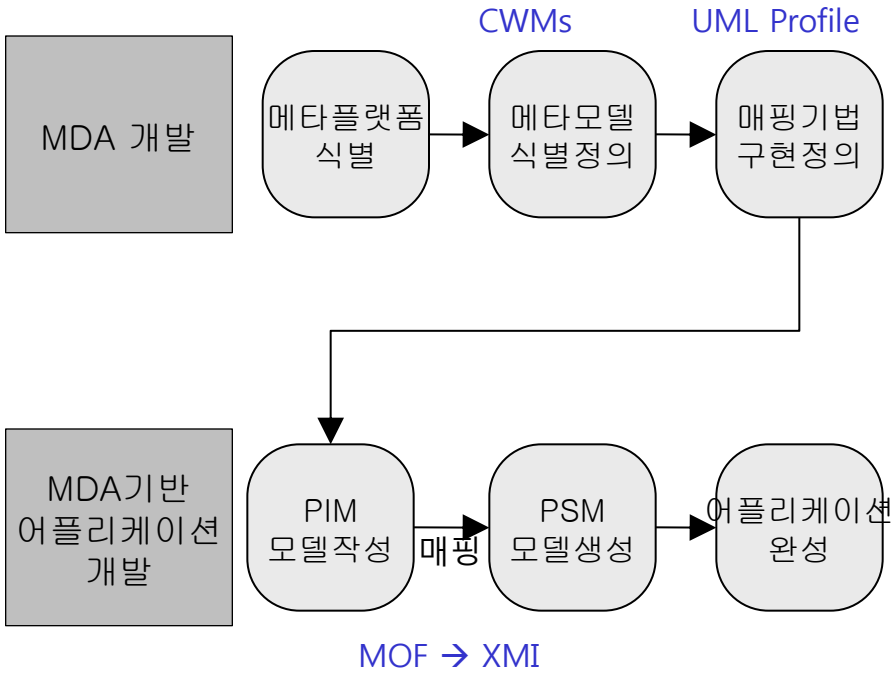
2) 컴포넌트 기반 조립 개발 관점

- **조립 개발**
  - 잘 정의된 인터페이스 단위 조립개발
- **반복적 접근**
  - 개발공정과 관리공정 분리 및 조화
  - 반복을 통해 개발위험 식별 제거
- **아키텍처 기반**
  - 아키텍처 기반의 개발방식
  - 컴포넌트 개념의 분석과 설계

4) 개발방법론의 종류 – ⑤ MDD 방법론(Model Driven Development)

플랫폼 독립적인 SW모델로부터 플랫폼 종속적인 SW모델로 자동 변환하고, 소스코드를 자동 생성함으로써 원하는 플랫폼에 맞는 SW를 쉽고 빠르게 개발할 수 있는 개발방법론
– MDA(Model Driven Architecture)로 만들어진 SW모델(메타모델)을 적용하여 개발

■ 개념도



■ 기본원리

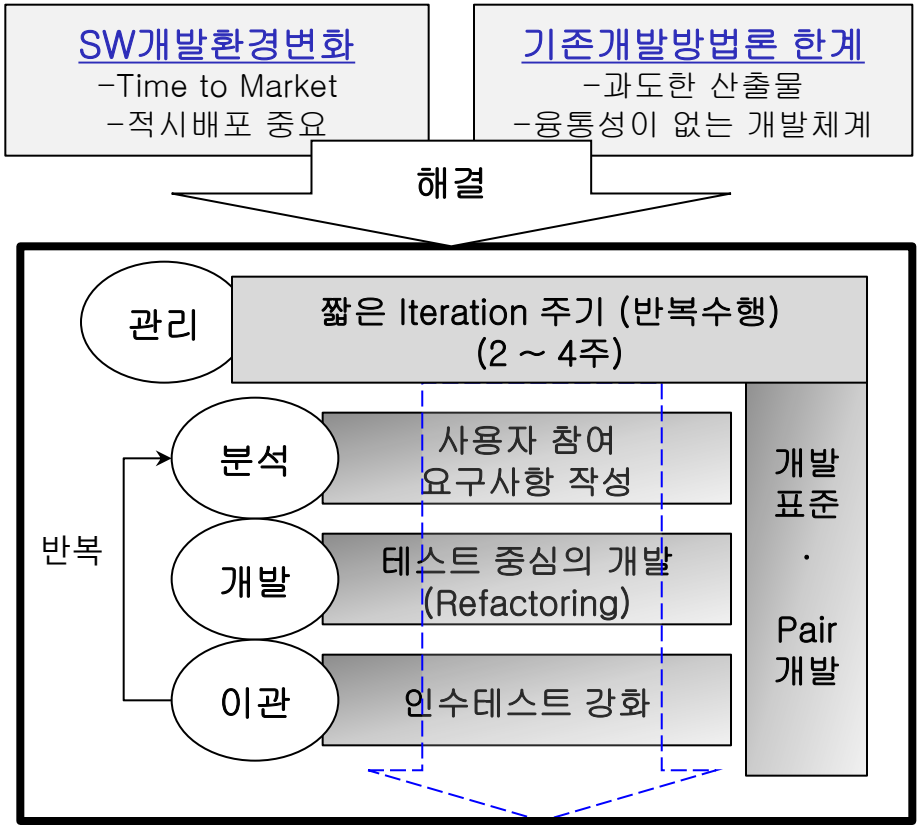
- **메타모델 기반**
  - 구현환경에 독립적인 모델구축 재사용
  - 자동으로 구현종속모델로 변환
- **설계와구현 분리**
  - UML로 작성된 설계모델 아키텍처와 구현환경 종속된 코드개발의 분리
- **다중플랫폼 지향**
  - PIM 모델을 UML Profile을 이용하여 다양한 플랫폼에 구축 가능

1) PIM (Platform Independent Model)
   – 메타모델을 기반으로 한 독립적 모델(UML로 모델링)
2) PSM (Platform Specific Model)
   – 플랫폼(구현환경) 종속 구현모델로 변환하는 설계모델
3) CWM (Common Warehous MetaModel)
   – 데이터 변환을 위한 표준 모델 제시
4) MOF (Meta Object Facility)
   – 메타데이터 정의,조작,통합 프레임웍 & 저장소
5) UML Profile (구현환경별 프로파일 생성)
   – PIM을 PSM으로 변환해주는 메타모델
6) XMI (XML Metadata Interchange)
   – MOF기반모델을 XML로 매핑하기 위한 표준사양

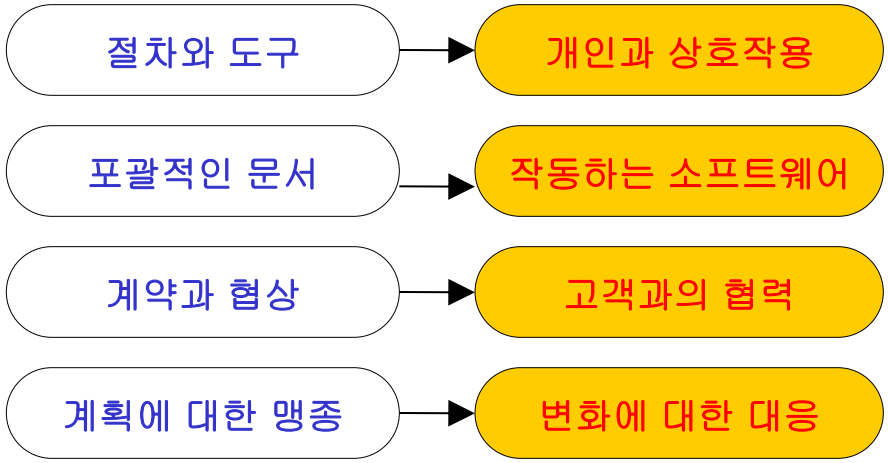
4) 개발방법론의 종류 – ⑥ Agile Process

절차보다는 사람이 중심이 되어 변화에 유연하고 신속하게 적응하면서 효율적으로 시스템을 개발할 수 있는 프로세스 (방법론적인 절차와 기법, 산출물이 아직 체계화가 되지 않은 상태여서 방법론으로 분류하지 않고 프로세스로 분류함)

■ 개념도



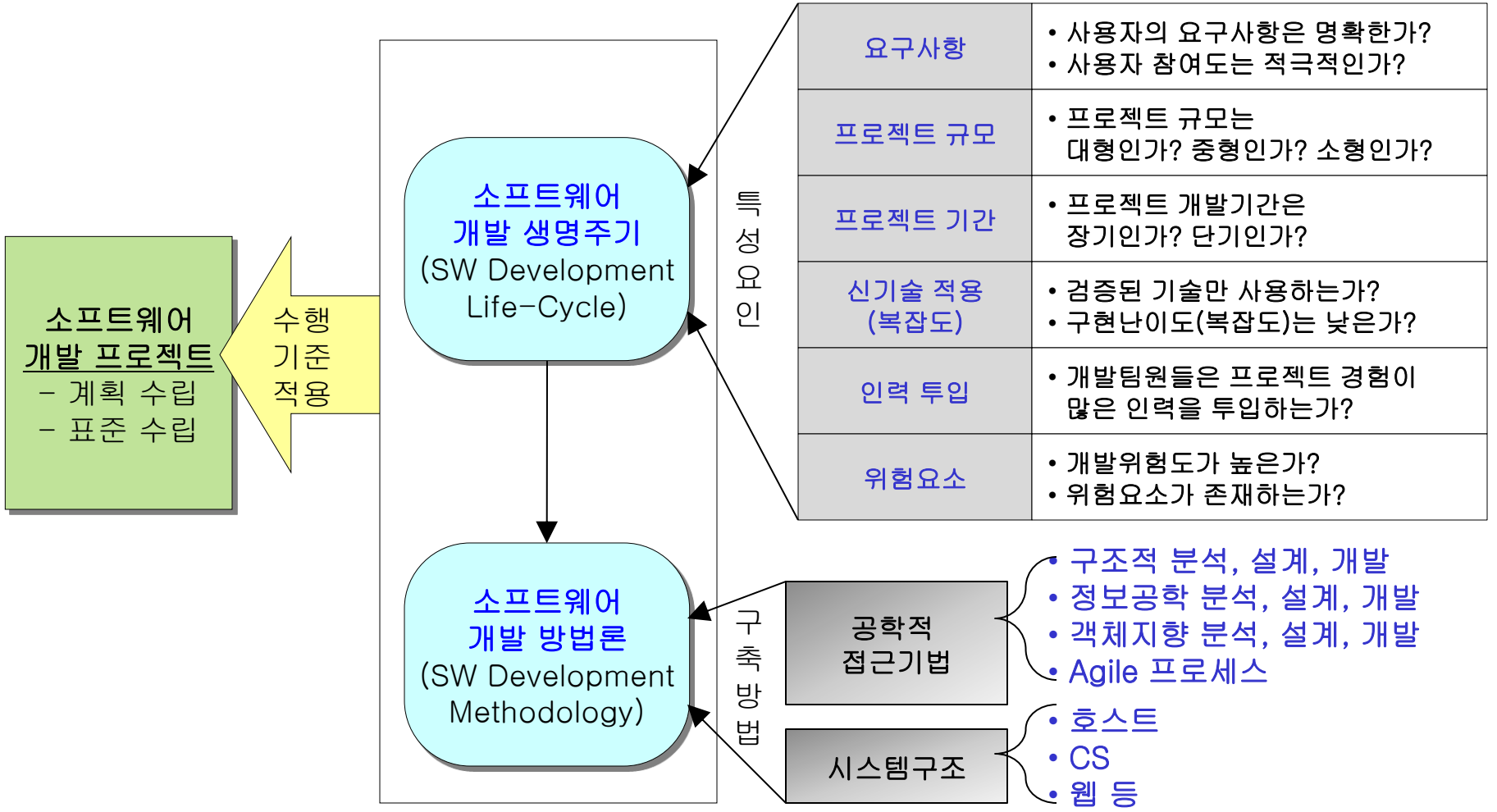
■ 핵심가치(Manifesto)와 주요 특징



- Predictive하기보다는 Adaptive한 개발
  : 가변적인 요구에 대응
- 프로세스 중심이라기 보다는 사람 중심의 개발 지향
  : 책임감 있는 개발자와 전향적인 사용자

5) 개발방법론 선정기준

IT 프로젝트에서 개발방법론을 선정하는 경우, 먼저 프로젝트 특성요인을 고려하여 개발수명주기 모델을 선택하고, 적용되는 SW공학적 기법에 따른 개발방법론을 선정해야 함



5) 개발방법론 선정기준 - 계속

프로젝트의 SDLC(개발수명주기) 선정시 아래와 같은 특성을 고려하여 선정할 수 있다

특성	폭포수 Waterfall	프로토타이핑 Prototyping	증분형 Incremental	진화형 Evoluationary	나선형 Spiral	RAD
① 대규모		●	●	●	●	
② 위험 多		●	●	●	●	
③ 참조모델 多	●					●
④ 요구사항 불명확		●	●	●	●	
⑤ 장기간 수행					●	
⑥ 충분한 예산					●	
⑦ 낮은 복잡도	●					●
⑧ 정확성 필요		●			●	
⑨ 적극적인 고객			●	●		●

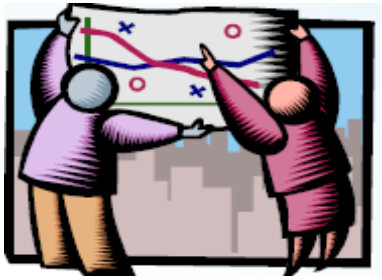
Ⅱ. 개발방법론 적용 및 프로세스

1) IT중소기업에 개발방법론 적용방안

IT중소기업에 개발방법론을 적용하고자 하는 경우에 반드시 사전에 조직에서 사용할 수 있는 **공통 표준개발방법론을 개발하여 활용 가능성을 검토**하여야 함

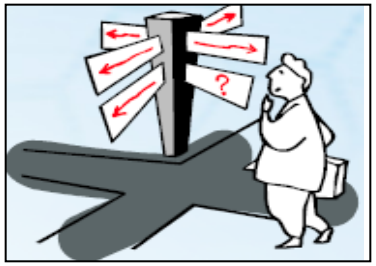
■ 개발방법론 표준화 과정

1. 개발업무 영역 이해



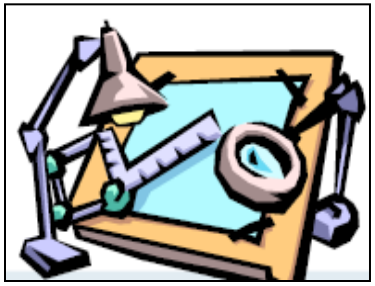
2. Reference 검색 (절차, 활동, BP사례)

3. 개발체계 검토 및 핵심프로세스 도출

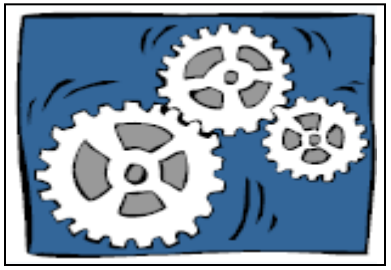


4. 개발방법론에 포함할 핵심 프로세스 결정

5. 개발방법론 구축 또는 갱신



6. PILOT 수행



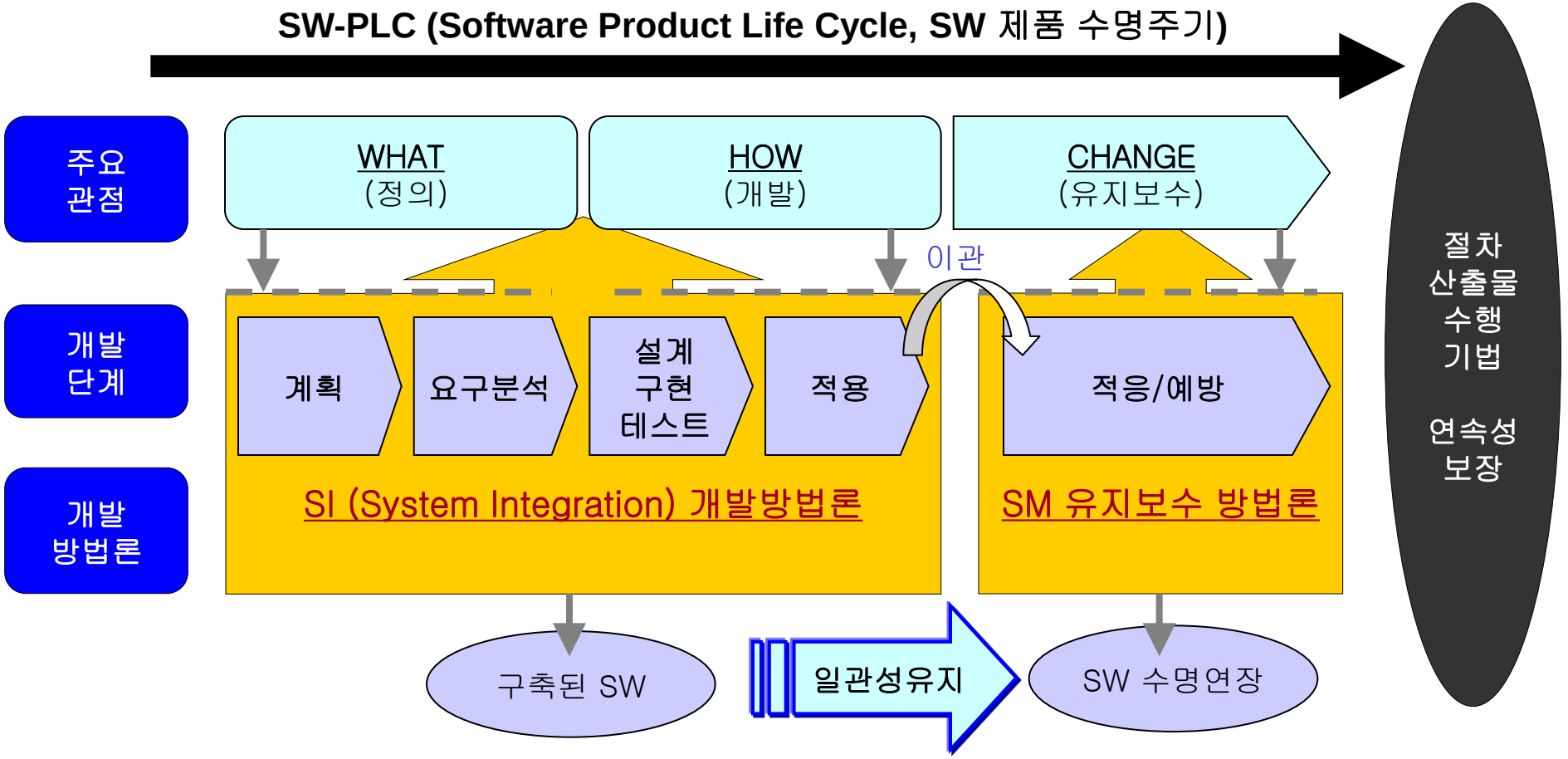
7. 수행결과 검토

8. 개발방법론 표준화 및 전사 확산

1) IT중소기업에 개발방법론 적용방안-계속

특히, 개발방법론 구축시 SW를 초기 개발(Develop)하고 지속적으로 유지보수(Maintenance) 할 수 있는 **전체 과정의 일관성 유지**를 위해서는 SI와 SM방법론을 모두 구축하여야 함

■ 개발방법론 적용범위

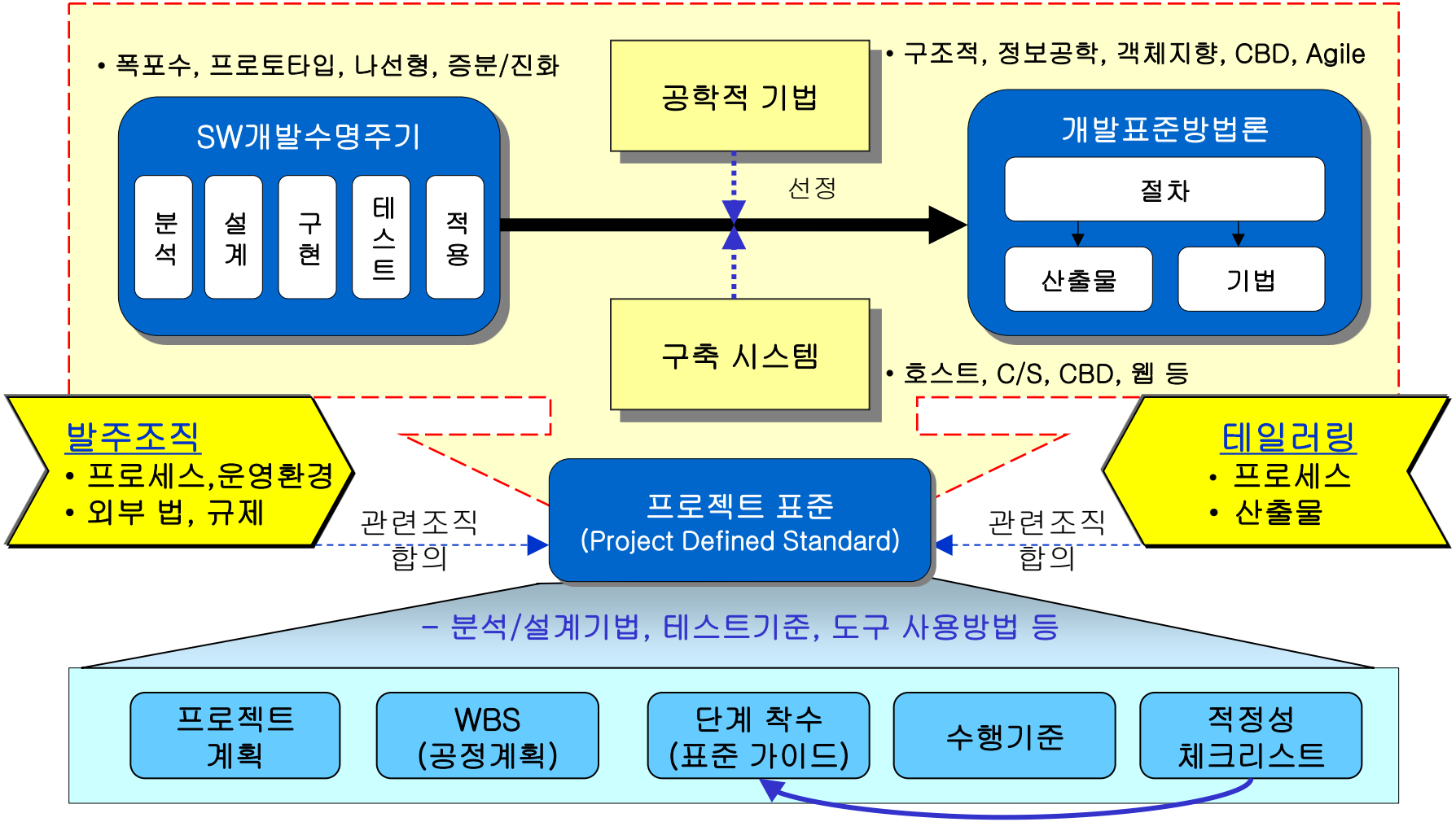


1) IT중소기업에 개발방법론 적용방안-계속

■ 개발방법론 적용범위 - 계속

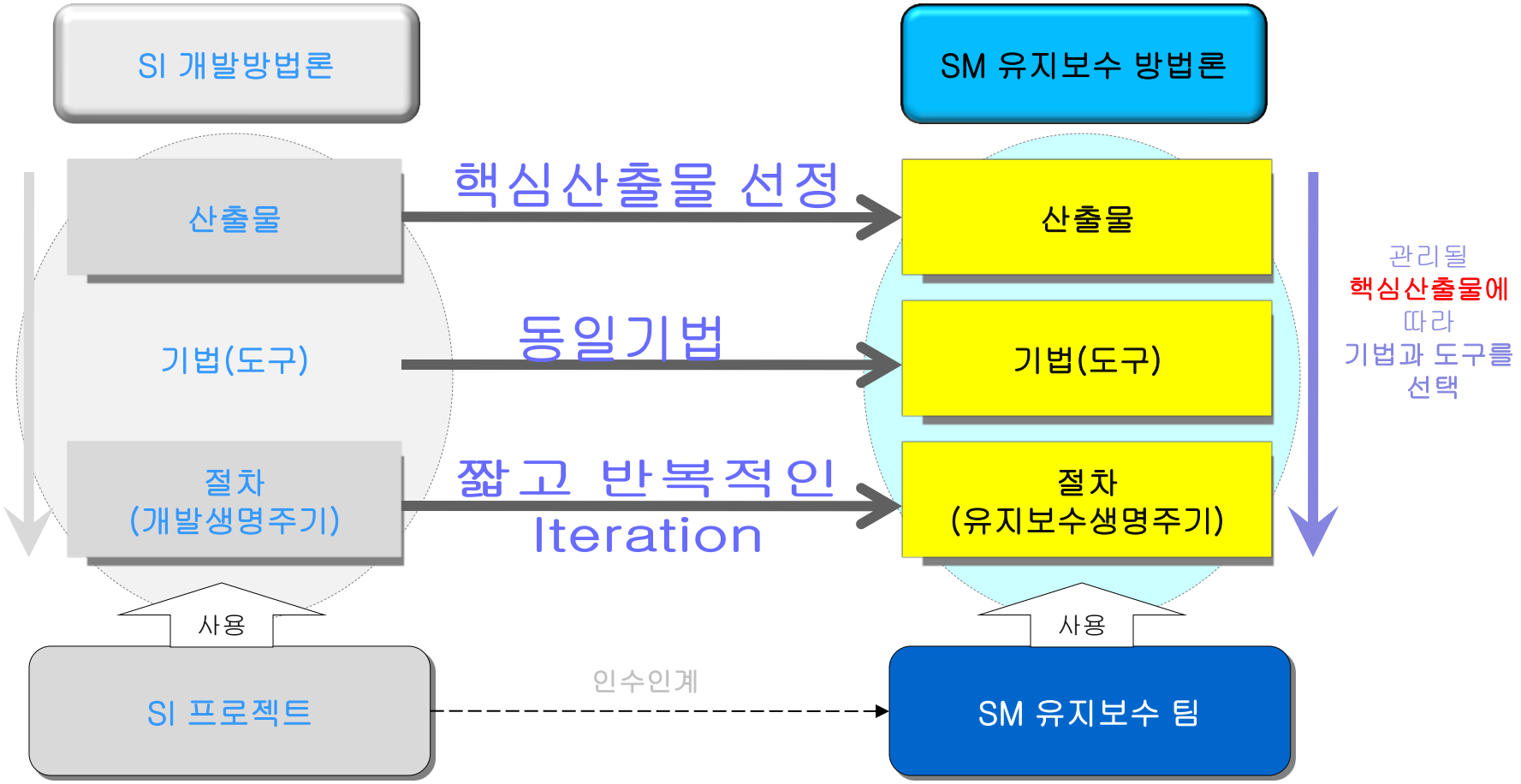
2) SI프로젝트 개발방법론 적용

정보시스템을 새로 구축하는 경우에 해당하며, 표준 개발방법론을 참조하여 프로젝트의 환경을 고려한 개발표준(Project Defined Standard)을 수립함



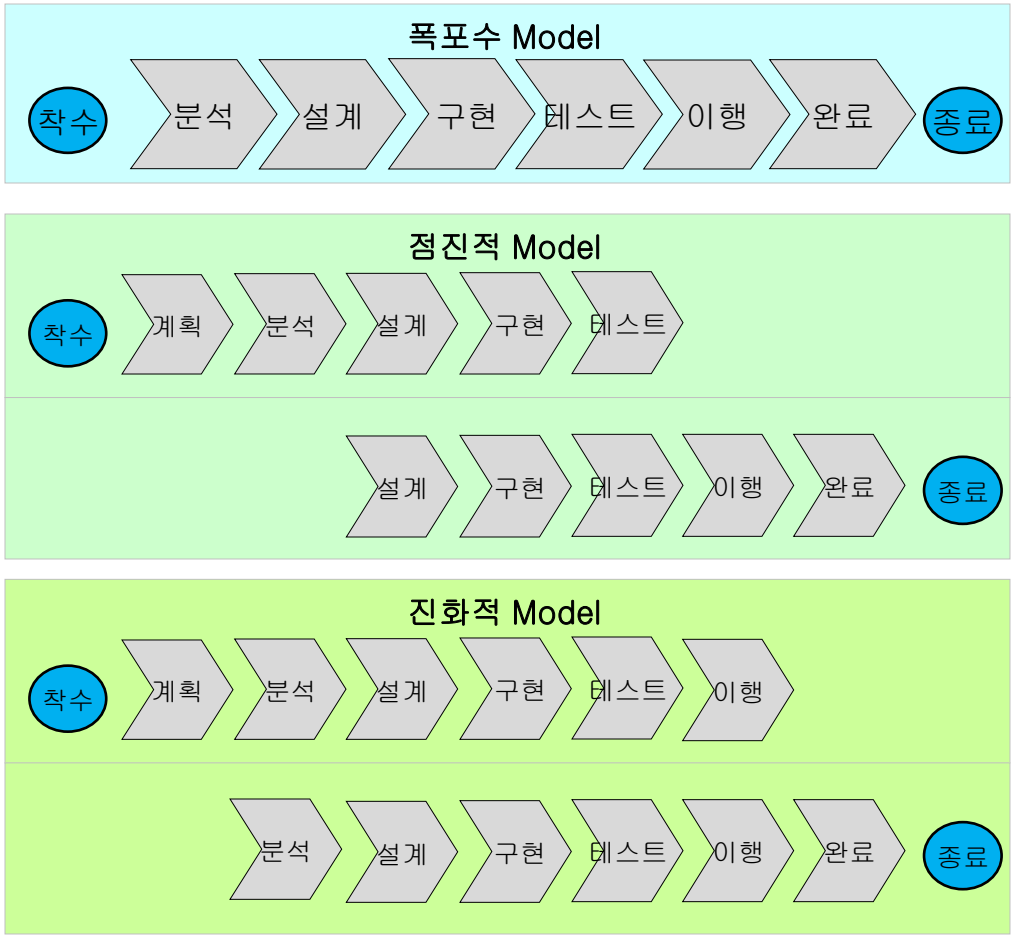
3) SM 유지보수 개발방법론 적용

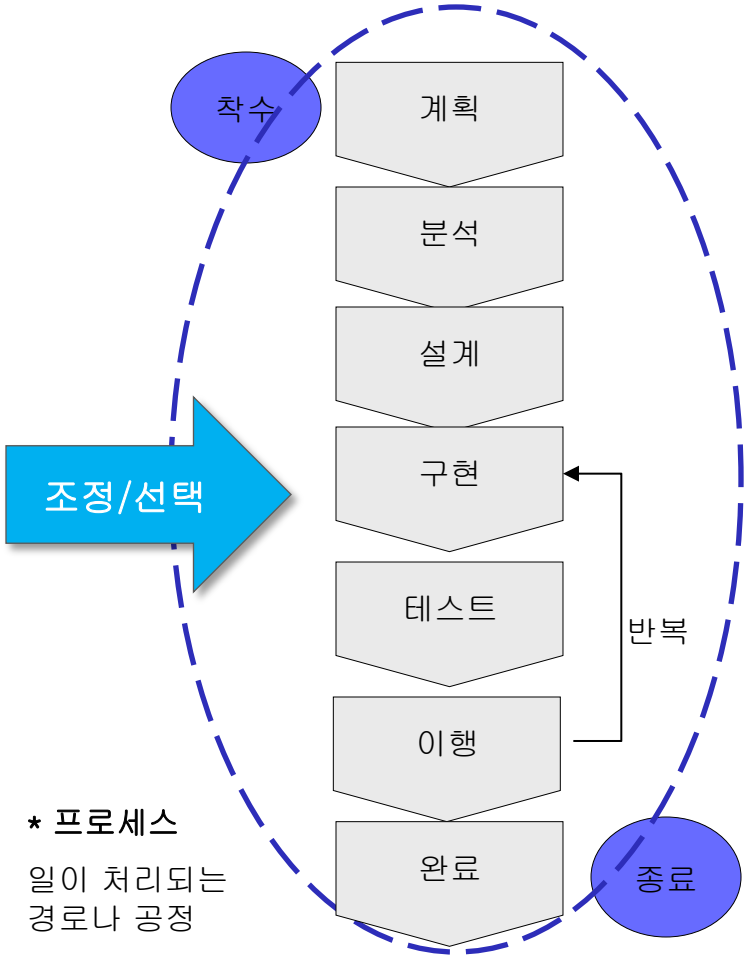
SM 유지보수에 개발방법론 적용하는 경우에도 SI 개발방법론과 동일한 산출물, 기법을 사용하되 핵심 산출물만 재사용하고, 조직적으로 적용할 개발표준과 유지보수를 위한 SW생명주기를 적용



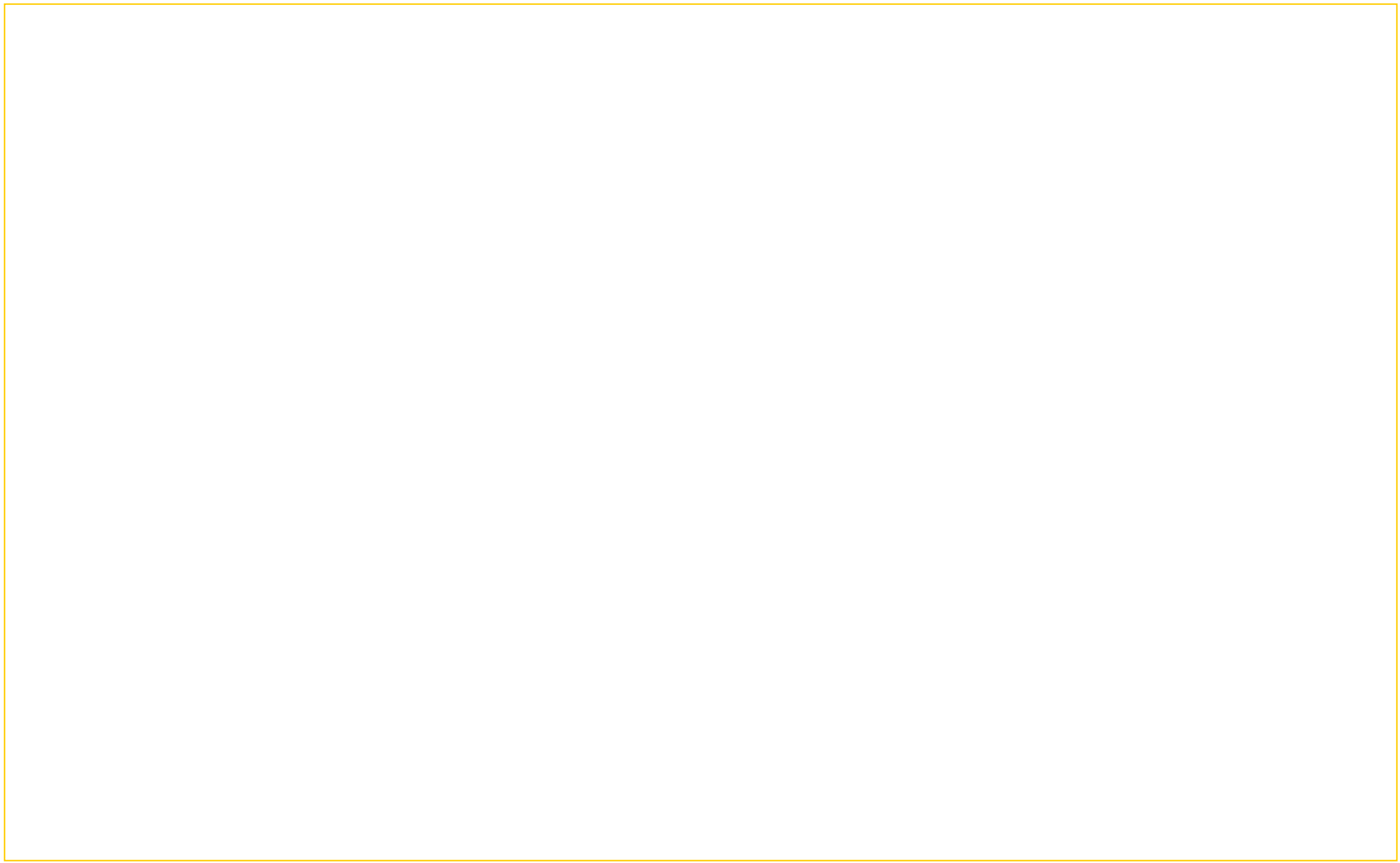
1) SW 개발방법론 프로세스

SW개발방법론의 프로세스는 개발생명주기(SDLC)를 적용하여 구현함. 개발생명주기는 대부분 유사한 Life-Cycle을 가지고 있어서, IT조직에서는 필요에 따라 선택하여 사용하면 됨





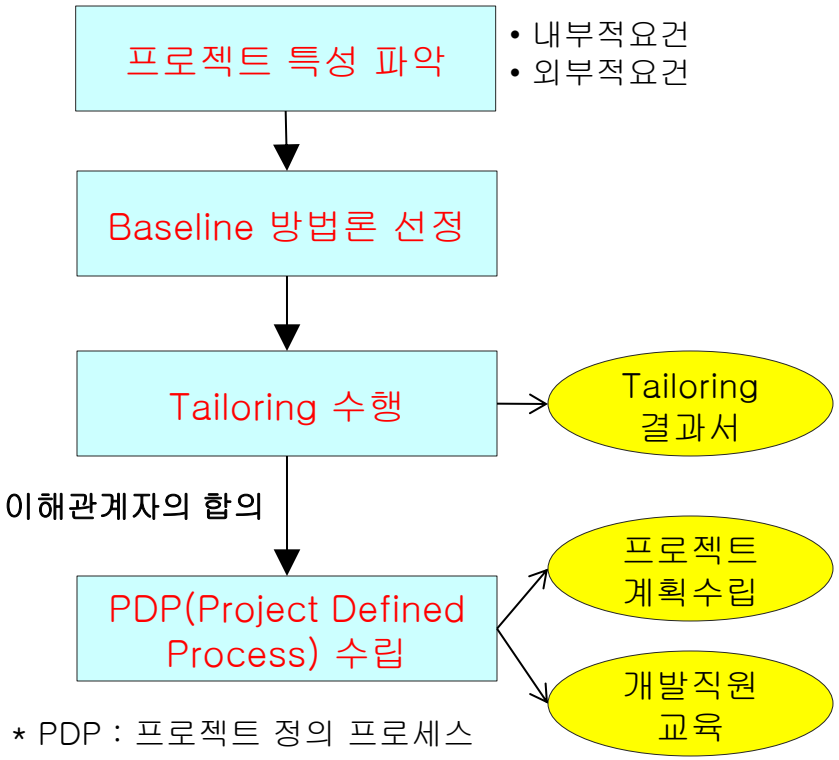
1) SW 개발방법론 프로세스 - 계속

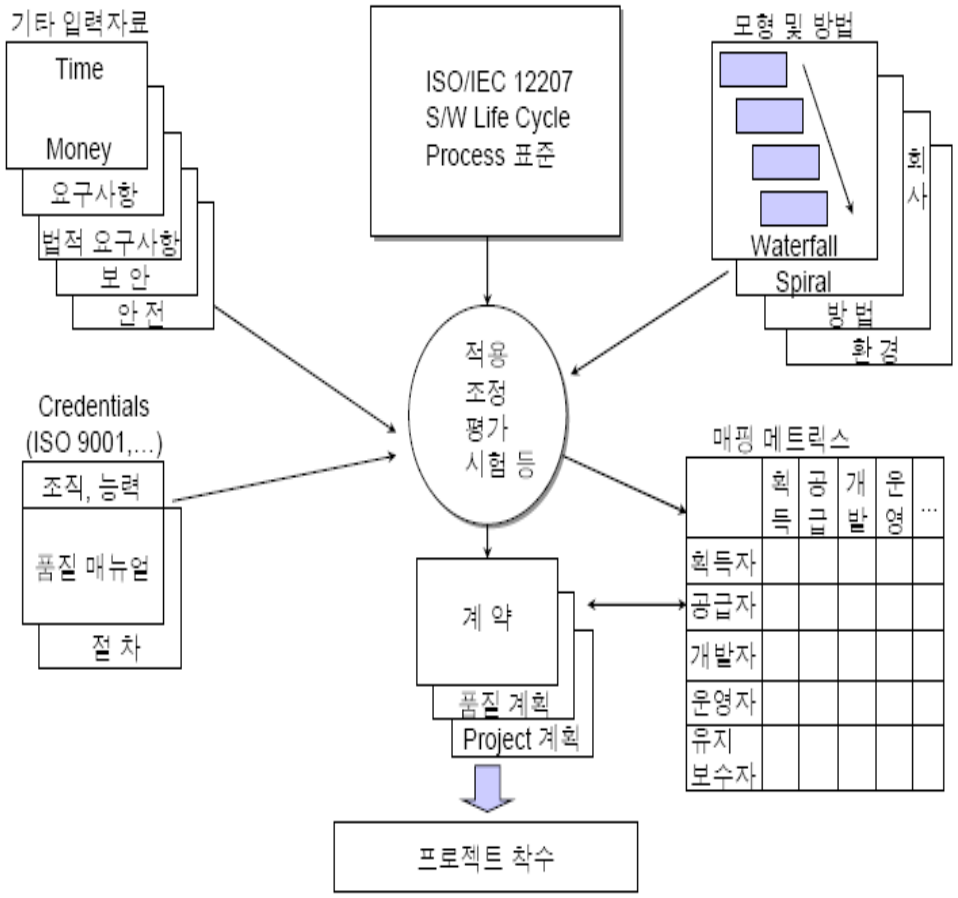


2) 개발방법론 테일러링

SW개발생명주기와 SW개발방법론을 선택한 이후에는 프로젝트 상황에 맞도록 하기 위해 기 정의된 개발방법론의 절차/기법/산출물 등을 수정하여 적용하여야 함

■ 방법론 테일러링(Tailoring) 절차





2) 개발방법론 테일러링 – 테일러링 결과

테일러링(Tailoring) 결과서는 자사의 SW개발 표준 프로세스와 개발방법론을 고려하여 작성할 수 있으며, 테일러링이 많이 발생하는 절차, 기법, 산출물 등은 표준SW개발방법론의 개정이 필요함

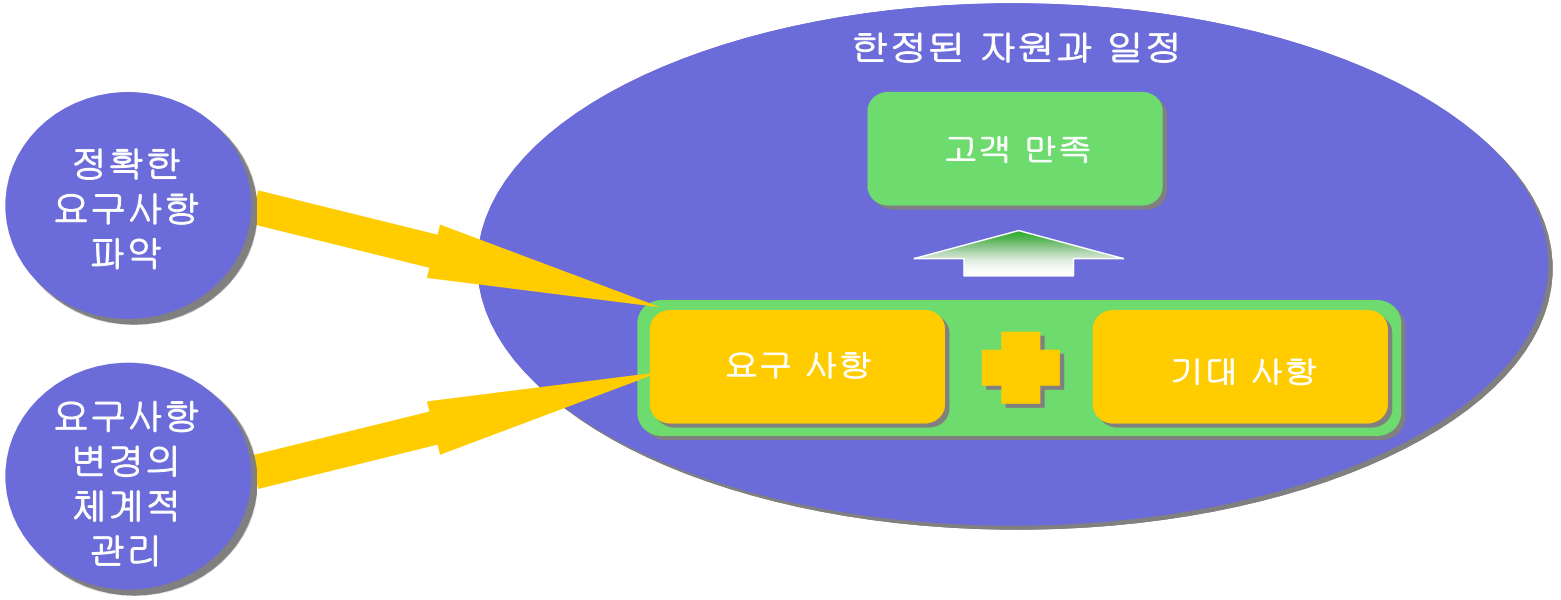
■ 표준 SW개발 방법론 테일러링 (Tailoring) 결과서

표준 개발방법론				반영 정도	조정내용 및 적용방안	조정사유 및 근거
단계	활동	작업	산출물			
분석단계	요구사항정의	시스템기본요건정의	요구사항정의서 (기능,기술,인터페이스,시스템화)	적용		
	신)논리모델 구축	프로세스모델링 (기능차트기법)	기능차트 DFD 프로세스정의서	미적용 적용 적용	기능차트와 프로세스정의서 통합	소형 프로젝트
		데이터 모델링	ERD 엔티티 목록 엔티티 정의서	적용 미 적용 미 적용	ERWin 대체	EA표준 * 메타사용
		이벤트 모델링	이벤트 목록 이벤트 시나리오	적용 적용	산출물통합	소형 프로젝트
		프로세스 대 엔티티 연관분석	프로세스 대 엔티티 상관도	미 적용	설계단계로 통합	소형 프로젝트

Ⅲ. 요구분석 핵심 기법

1) 성공적인 프로젝트가 되기 위한 조건

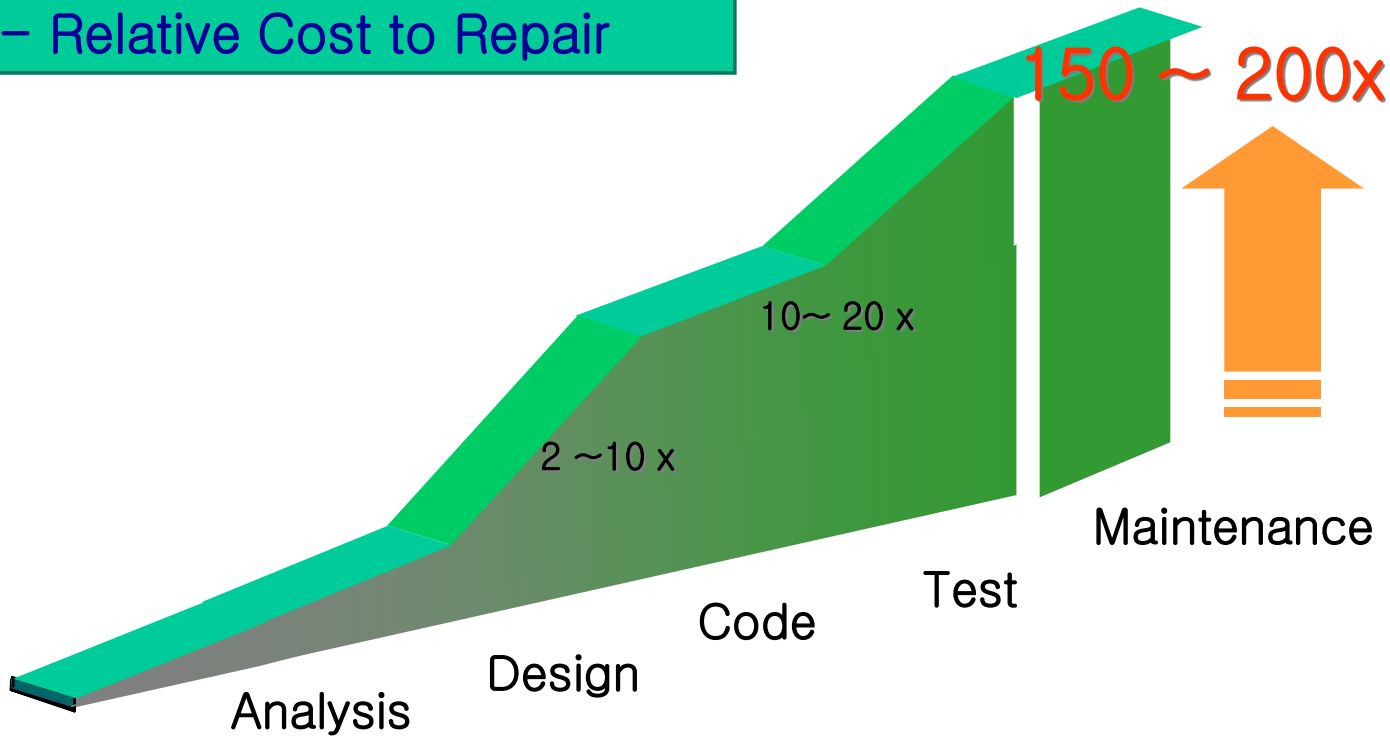
- 한정된 자원과 일정 안에서 고객의 요구사항을 100% 만족 시키며 기대 사항을 만족 시켜야 함
- 정확한 요구사항의 빠른 파악과 변경에 대한 관리가 프로젝트 성공을 좌우함



2) 요구사항 관리의 목적과 효과

잘못된 요구사항에 대한 비용

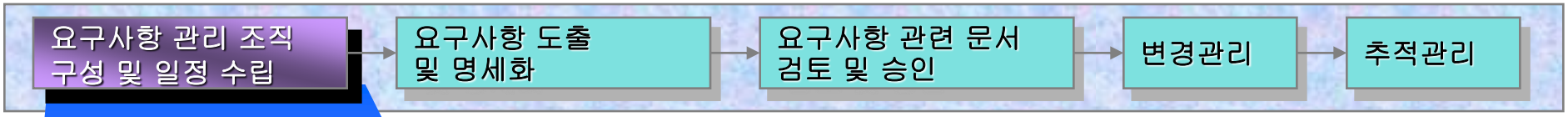
– Relative Cost to Repair



2) 요구사항의 종류

구분	기능적 요구사항	비기능적 요구사항
정의	- 시스템의 업무기능적 요구사항	- 시스템이 만족해야 하는 특성,제약 등을 규정
특성	- 데이터모델,(생성,삭제) 데이터흐름처리모델 - 프로세스 모델	- 신뢰성, 성능, 보안성, 안정성 등 시스템 전체에 대한 요구사항
사례	- 계좌개설,계좌통합	- 시스템의 성능, 메모리 사양, 품질속성, 네트워크 성능 등
비고	- 100%만족 목표	- 현실적으로 100%만족 시킬 수 없음

3) 프로세스 개요 및 세부 프로세스



- 요구사항 관리 조직 구성
- 요구사항 관리 일정 수립 및 교육 실시

선행기준

- 고객과 프로젝트 수행에 대한 합의가 이루어지고, 프로젝트 관리자가 선정된다.

목적

요구사항 관리 조직을 구성하고, 활동 수행을 위한 일정을 수립하고, 관련 조직의 지원을 위한 합의 도출 및 프로젝트 관리자의 승인을 득한다.

Input

- RFP, 제안서, 계약서

Outputs

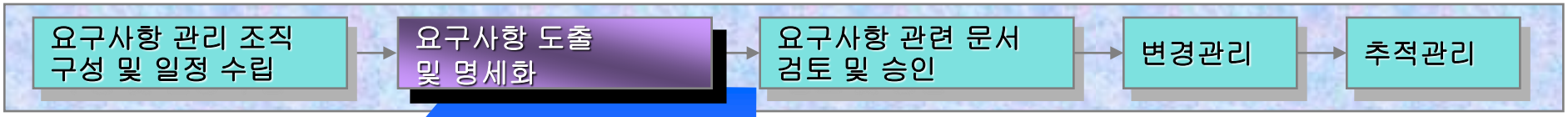
- 프로젝트 기술서(프로젝트 조직도, 프로젝트 작업 계획서)

완료 기준

- 프로젝트 관리자에 의해 요구사항 관리 조직 및 일정이 검토되고, 승인된다.

3) 프로세스 개요 및 세부 프로세스 (계속)

■ 세부프로세스



- 요구사항 도출
- 요구사항 명세화

선행기준

- 요구사항 관리 조직이 구성되고 일정이 수립된다.

목적

프로젝트에서 선정한 방법론에 따라 요구사항을 도출하고 명세화 하여 문서로 작성한다.

Input

- RFP
- 제안서
- 계약서
- 현행 시스템 정보

Output

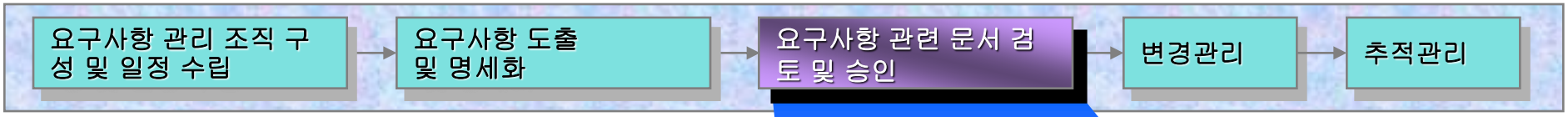
- 요구사항 관련 문서

완료 기준

- 요구사항 관련 문서에 대한 작성/검토가 이루어진다.

3) 프로세스 개요 및 세부 프로세스 (계속)

■ 세부프로세스



목적

요구사항 관련 문서가 고객의 요구사항을 잘 반영하고 있는지 확인하기 위하여 관련 내부 조직의 검토 및 고객과 합의가 이루어진다. 또한 이 활동을 통하여 요구사항의 Baseline이 설정되므로 구성관리가 시작된다.

Input

- 요구사항 관련 문서

Output

- Inspection 계획 및 결과서
- 승인된 요구사항 관련 문서

완료 기준

- 요구사항 관련 문서의 고객검토, 승인이 이루어지고, Baseline이 설정된다.

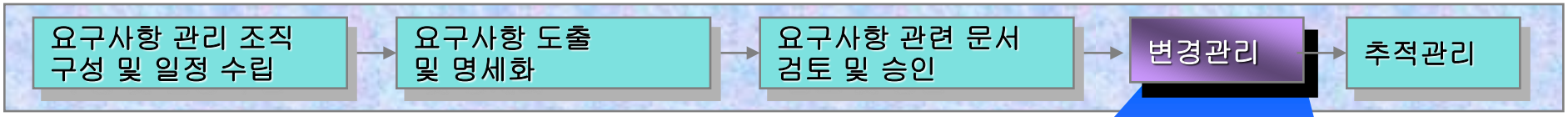
- •요구사항 관련 문서 내부 검토 및 승인
- •요구사항 관련 문서 고객 검토
- •검토 결과 요구 사항 보완
- •고객 승인 획득
- •명세서 Baseline 설정

선행기준

- 요구사항 관련 문서가 작성된다.

3) 프로세스 개요 및 세부 프로세스 (계속)

■ 세부프로세스



목적

요구사항 변경 요청에 따라 변경에 따른 영향이 검토되고, 변경에 대한 승인이 완료되면 변경 작업이 수행된다.

Input

- 요구사항 관련 문서
- 변경 요청서

Output

- 승인된 변경 요청서
- 변경된 요구사항 관련 문서

완료 기준

- 요구사항 변경 요청이 승인/기각/보류된다.
- 요구사항 변경 승인이 반영된다.

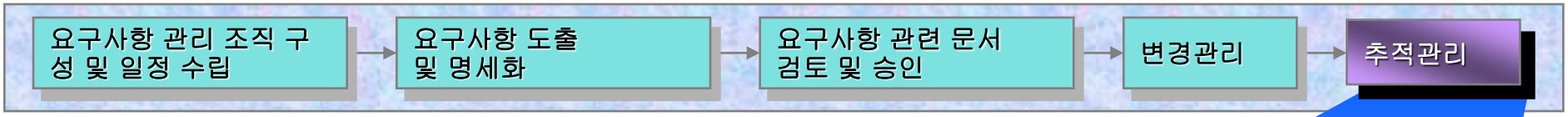
- •요구사항 변경 요청
- •요구사항 변경 검토 및 승인
- •변경 승인된 요구사항 적용
- •변경 완료 및 관련 문서 갱신

선행기준

- Baseline이 설정된 요구사항에 대한 변경 사유가 발생한다.

3) 프로세스 개요 및 세부 프로세스 (계속)

■ 세부프로세스



목적

승인된 요구사항이 프로젝트 라이프 사이클 전 개발 단계에 모두 반영되고 있는지 요구사항의 상태를 추적 관리 한다.

Input

- 요구사항 관련 문서
- 프로젝트 기술서
- 추적 대상 산출물

Output

- 요구사항 추적 매트릭스

완료 기준

- 요구사항 추적이 완료된다.

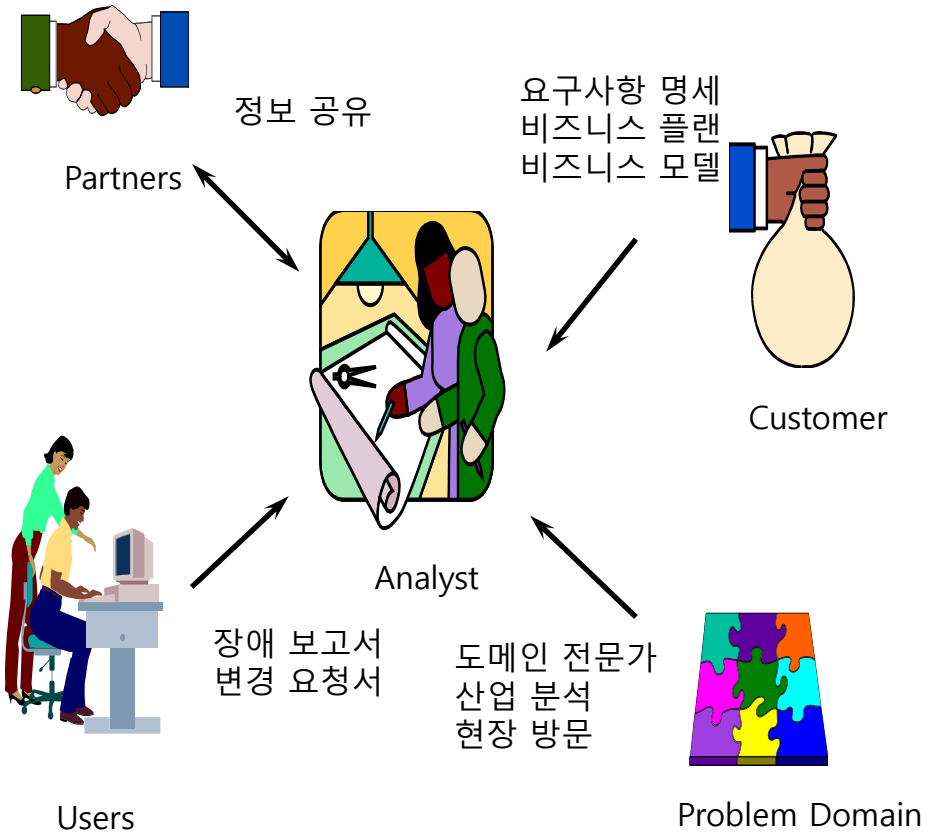
- •요구사항 추적 일정 수립
- •요구사항 추적 대상 산출물 정의
- •요구사항 추적성 기록
- •요구사항 현황 추적

선행기준

- 요구사항 관련 문서의 최초 승인이 이루어지고, Baseline이 설정된다.

1) 요구사항 분석

❑ 요구사항 식별 원천



❑ 요구사항 도출 기법

- ❖ 인터뷰 (Interview)
- ❖ 설문 (Questionnaire)
- ❖ 워크샵 (Workshop)
- ❖ 브레인스토밍 (Brainstorming)
- ❖ 롤플레잉 (Role Playing)
- ❖ 스토리보드 (Story Board)
- ❖ 프로토타이핑 (Prototyping)

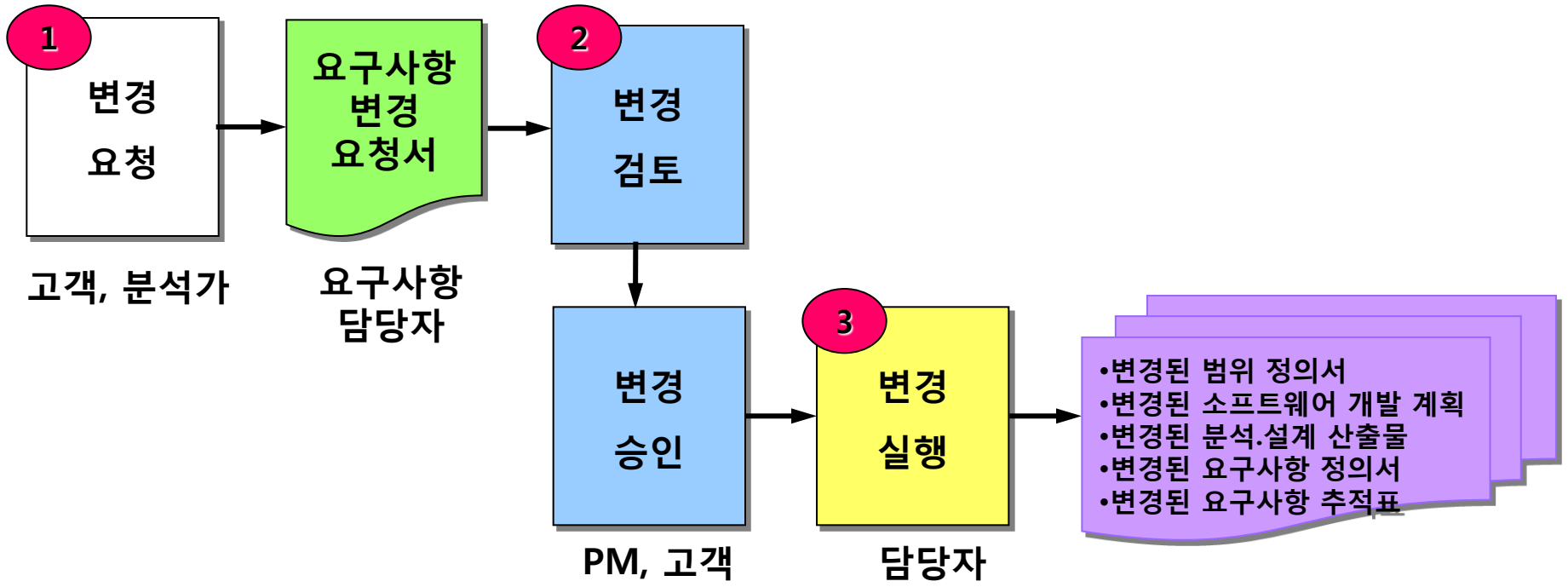
1) 요구사항 분석

1) 요구사항 정의서

- 요구사항 정의서란?
  - 요구사항 정의서 문서의 목적은 시스템의 높은 수준 요구사항과 특성을 정의하고 분석하고 수집한다.
  - 이해관계자의 필요한 능력과 이들이 왜 필요한 지에 초점을 두고 있다.

- 문제기술 부분
  - 초기 요구사항 작업 과정에서 간략하게 문제를 기술하기 위해 협력해야 한다.
  - 이해관계자들이 서로 약간 다른 문제들을 풀고자 시도할 가능성을 줄일 수 있다.
  - 간략한 문제 기술은 보통 빨리 작성된다.
  - 높은 수준에서 목표와 문제 요약
  - 유스케이스에 걸친 중요한 비기능적, 질적 목표

1) 요구사항 변경관리절차

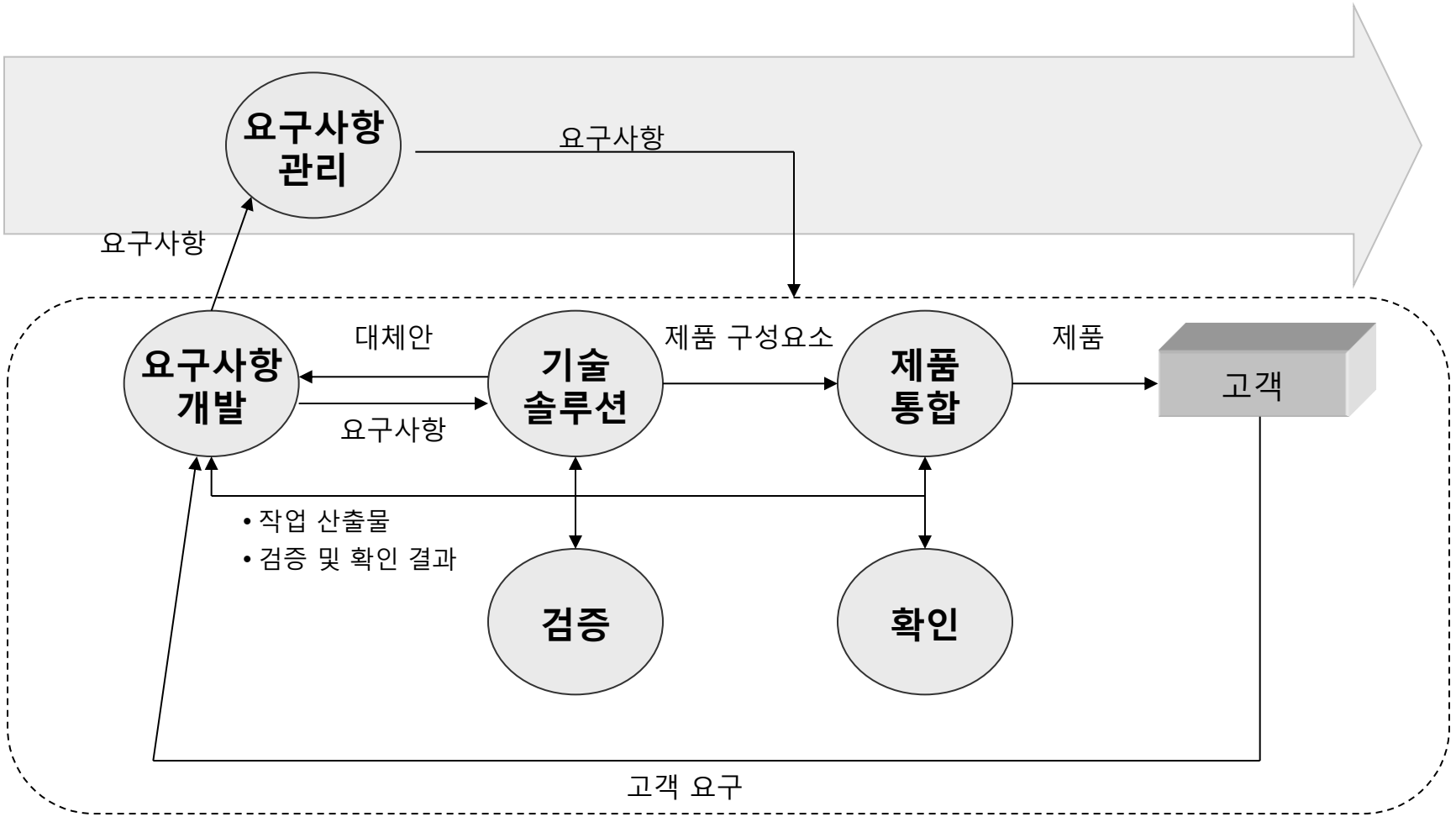


1.요구사항 변경요청
(1) 요구사항 변경요청서 작성 (2) 요구사항 변경요청서 접수 (3) 변경에 따른 영향 분석 (4) 변경에 대한 업무량, 일정 견적

2.요구사항 변경 검토 및 승인
(1) 변경관리 회의 (2) 요구사항 변경 승인

3.요구사항 변경 실행
(1) 소프트웨어 개발 계획 변경 (2) 변경 요구사항 정리 (3) 연계 산출물 및 활동의 변경 수행 (4) 변경 결과 확인

1) 엔지니어링 영역 내 프로세스간 관련성



1) 요구사항정의 및 추적관리 체계 강화 요약

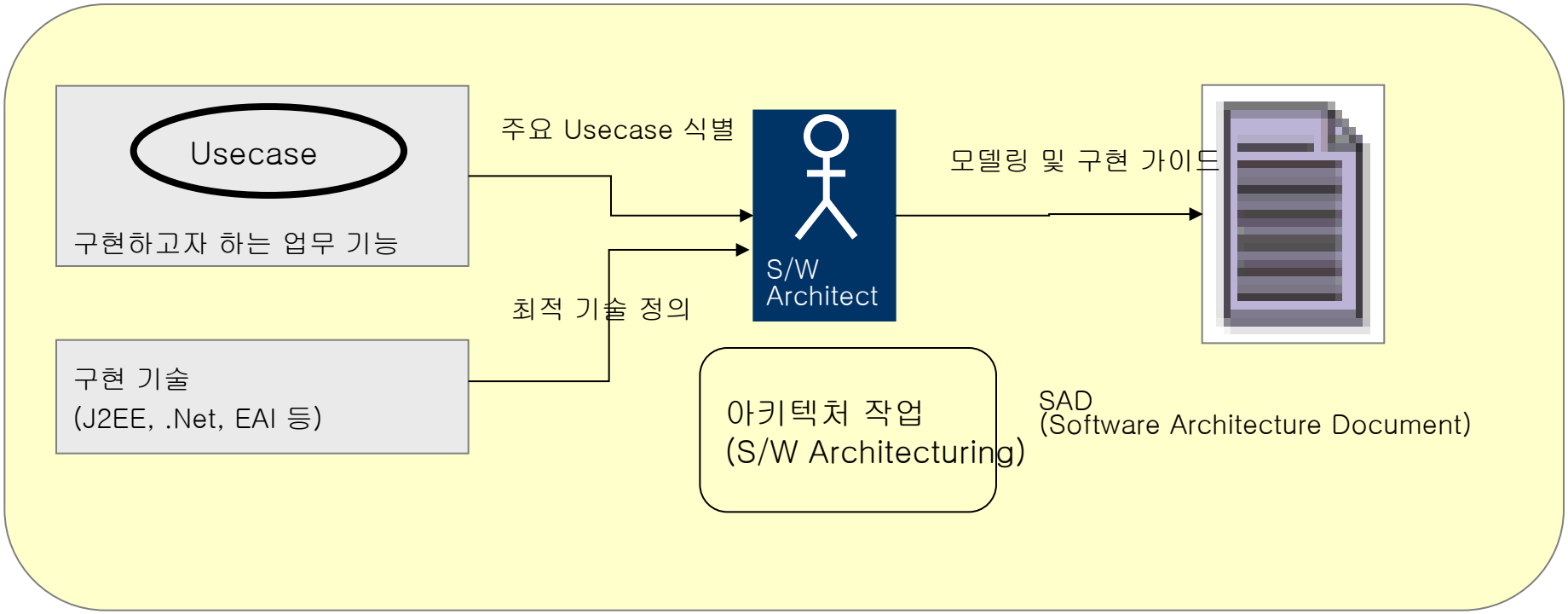
요구사항의 정의부터 시스템으로 구축되어 사용자에게 인도됨을 확인할 수 있는 체계를 구축하여야 함

Process	▪ Biz. 요구사항 작성/ 확인 활동 강화 (과제정의서	▪ Biz. 요구사항 작성/ 검토 및 확인 활동 강화		▪ 요구사항정의 단계에서 발생한 미결사항의 현행화 및 추적 현황 관리
양식/ 가이드/ 체크리스트	▪ Biz. 요구사항목록 작성	▪ 개정된 요구사항정의서		▪ 개정된 요구사항추적표
System	▪ Biz. 요구사항 관리 기능(시스템 구축요건)	▪ IT 요구사항의 상태	▪ 품질시스템 변경 관리 기능 개선	▪ 요구사항추적표의 Upload와 관련 산출물과 연결기능

Ⅵ. 설계 핵심기법

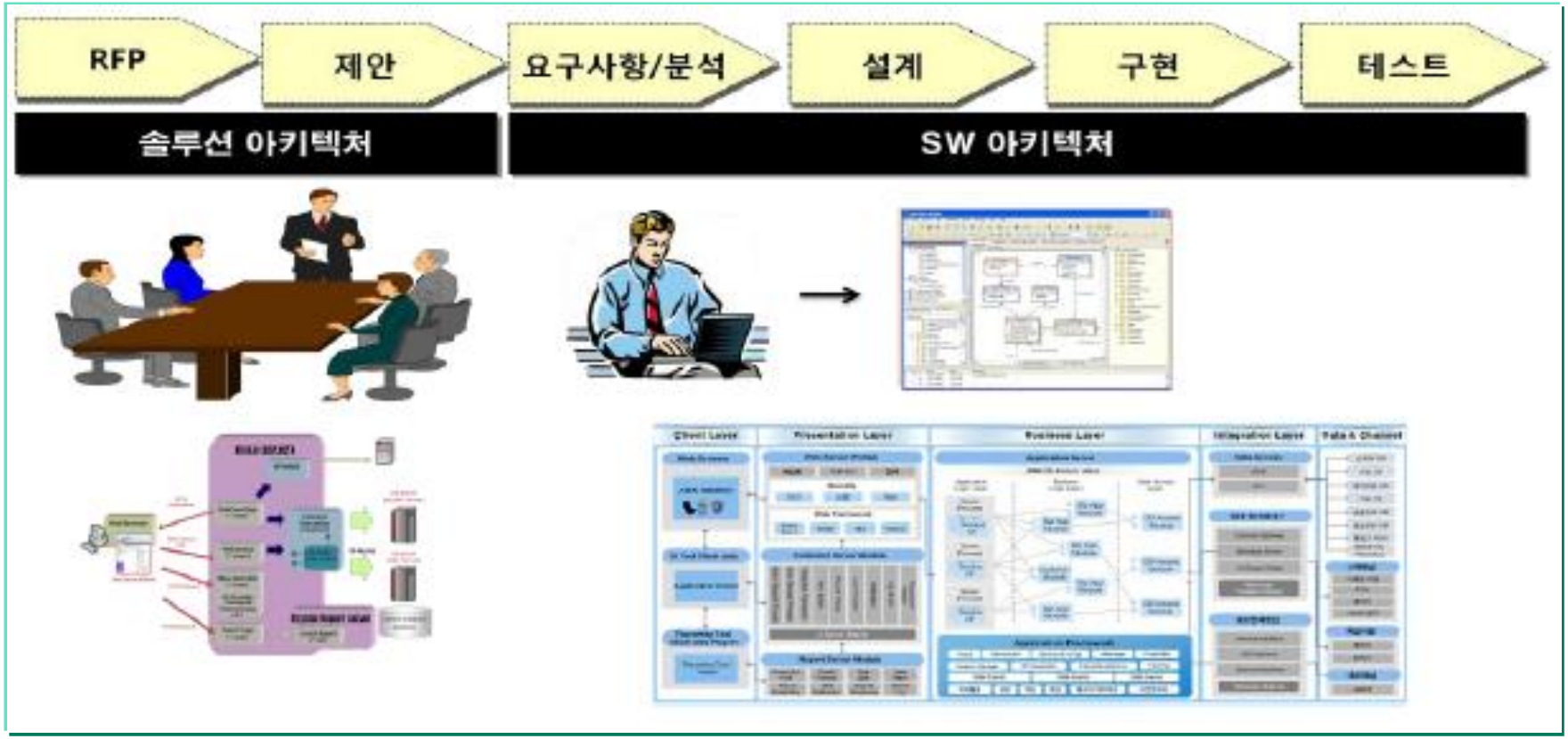
1) S/W Architecture의 정의

프로젝트 수행에 있어서 Software Architecture란 구현하고자 하는 핵심 업무 기능 요건을 잘 파악하고, 프로젝트 내에서 사용할 수 있는 다양한 기반 기술을 이용하여 구현한 최적의 구현 모델 및 가이드로서, 이러한 Software Architecture는 SAD (Software Architecture Document)로 표현된다.



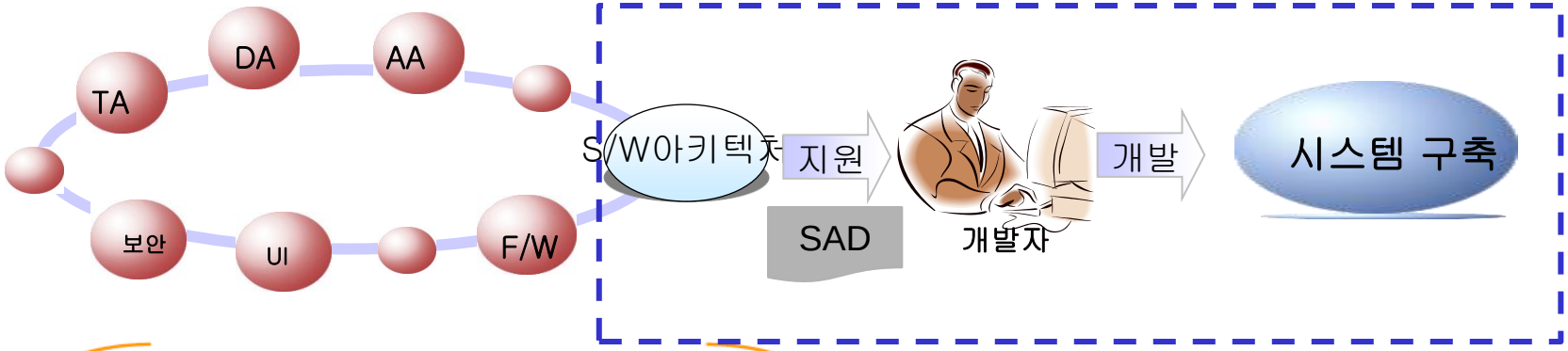
1) S/W Architecture의 정의 (계속)

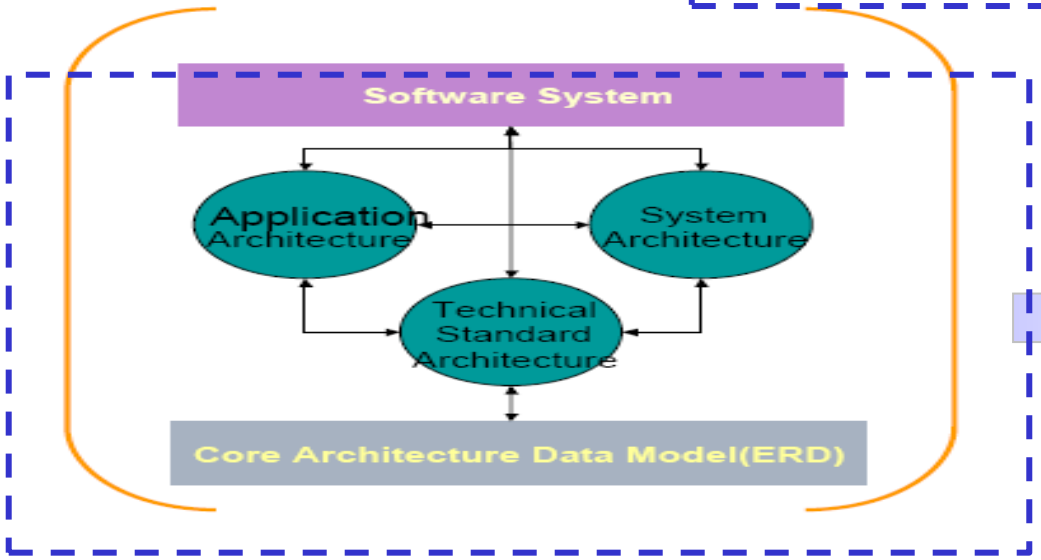
SI 서비스 산업에서 SW 아키텍처는 다양한 사업 진행 범위에서 주요한 역할을 수행하며 프로젝트의 성공을 이끌어 나감



1) S/W Architecture의 정의 (계속)

SA(Software Architecture)는 Enterprise Architecture(AA, TA, DA) 표준과 기술 인프라를 기반으로 최적의 업무 시스템이 구현될 수 있도록 가이드 및 기술 지원활동을 수행함

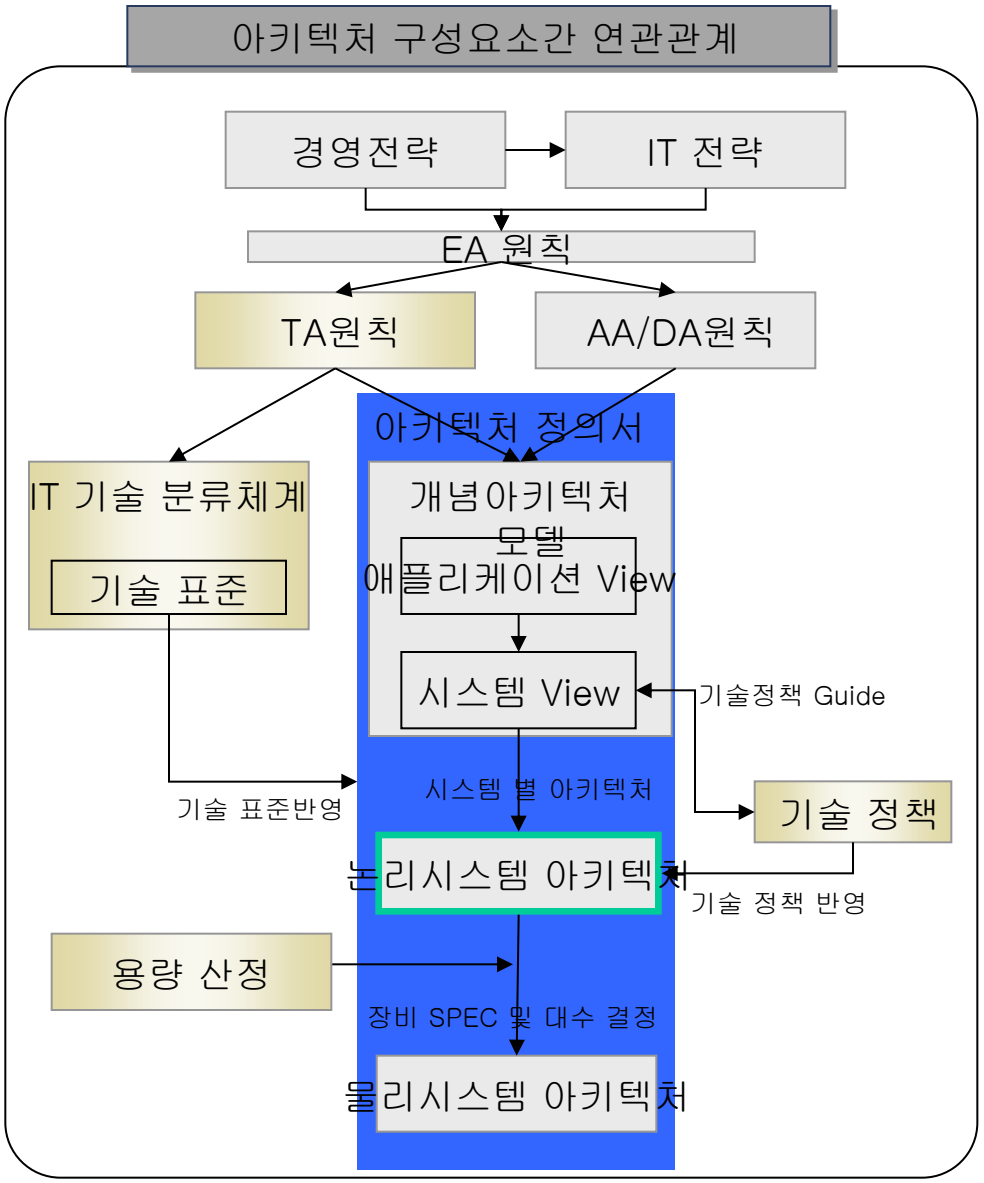


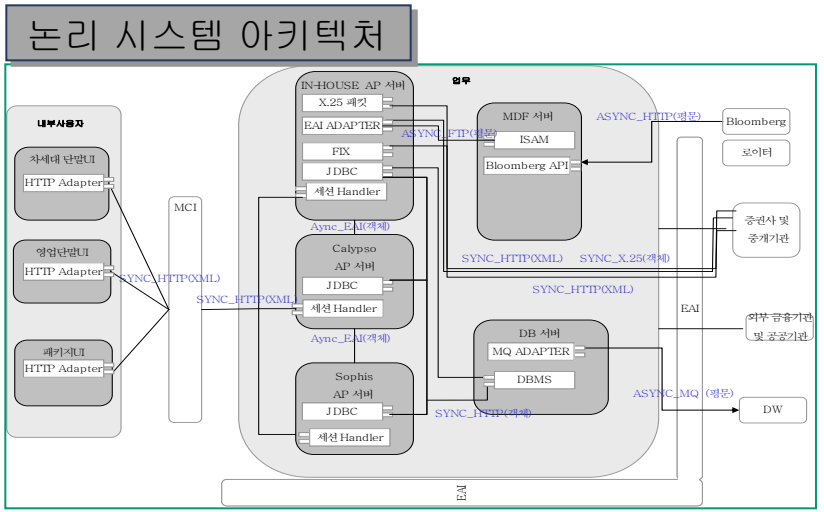


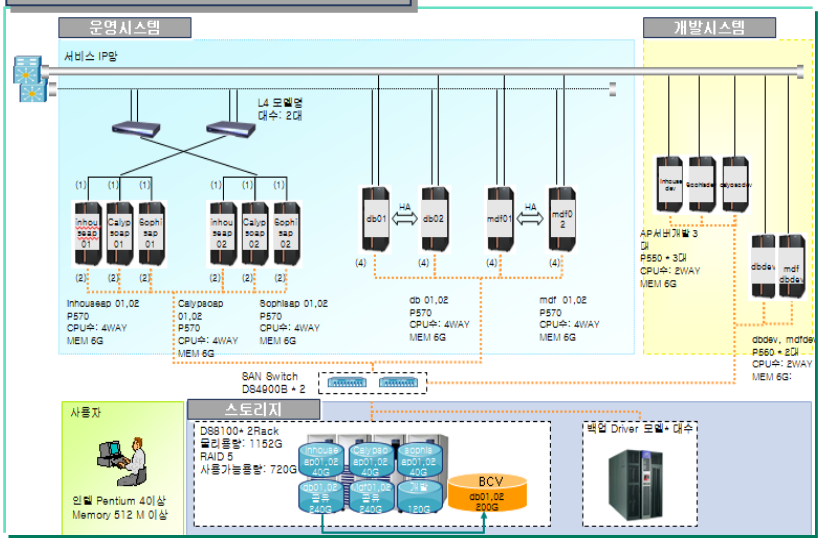
Vision

아키텍처 기반 하에 구현한 시스템은 그 조직의 비전을 실현하는 수단이다

1) 논리/물리 시스템아키텍처의 위치 및 작성 목적

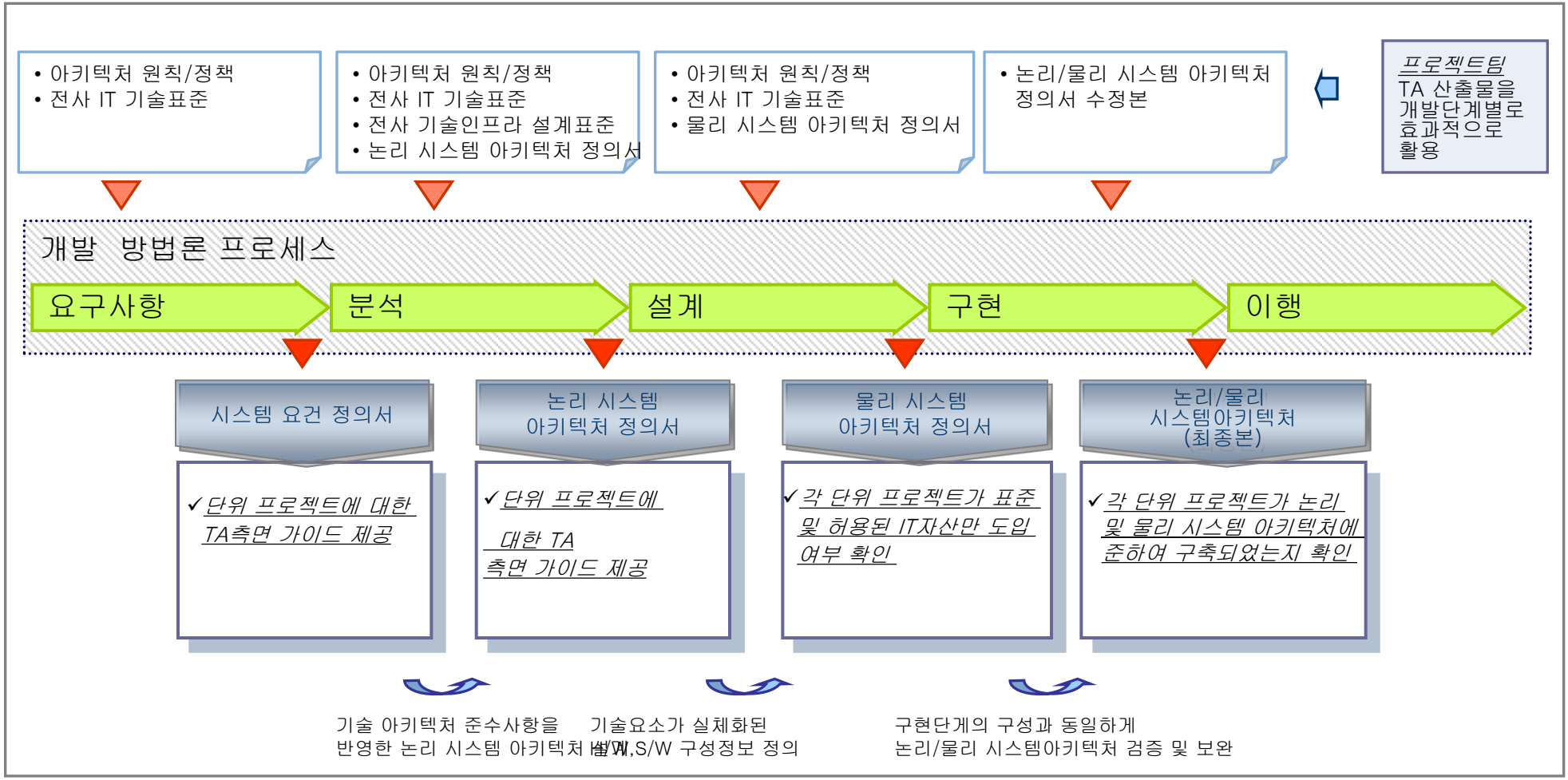




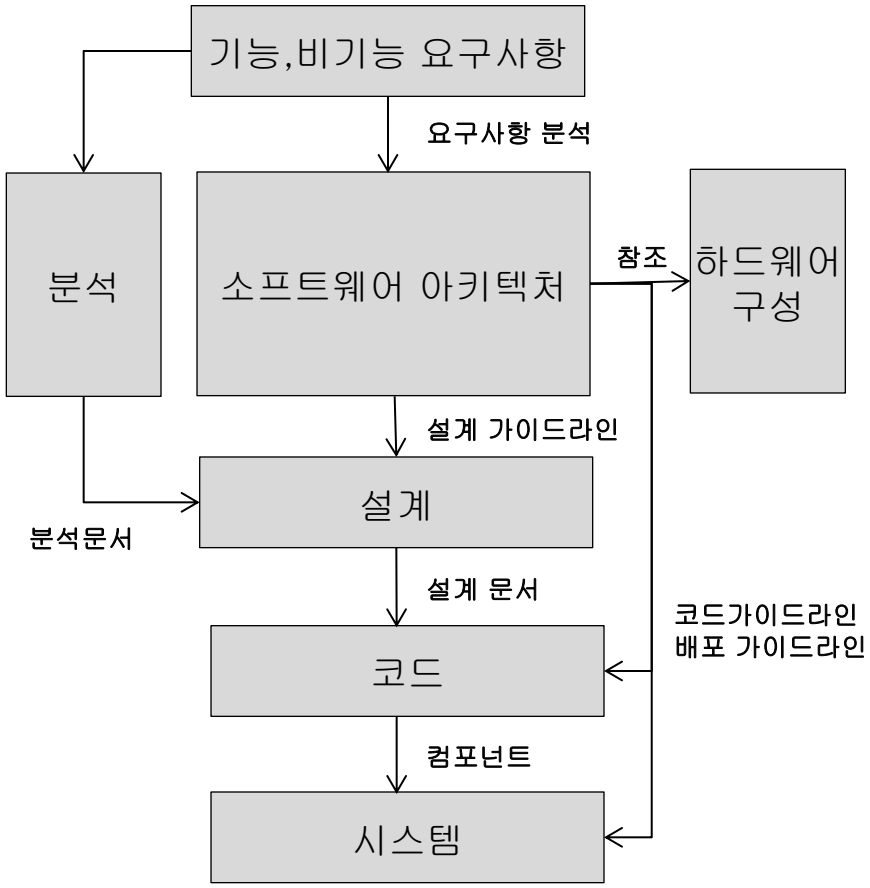


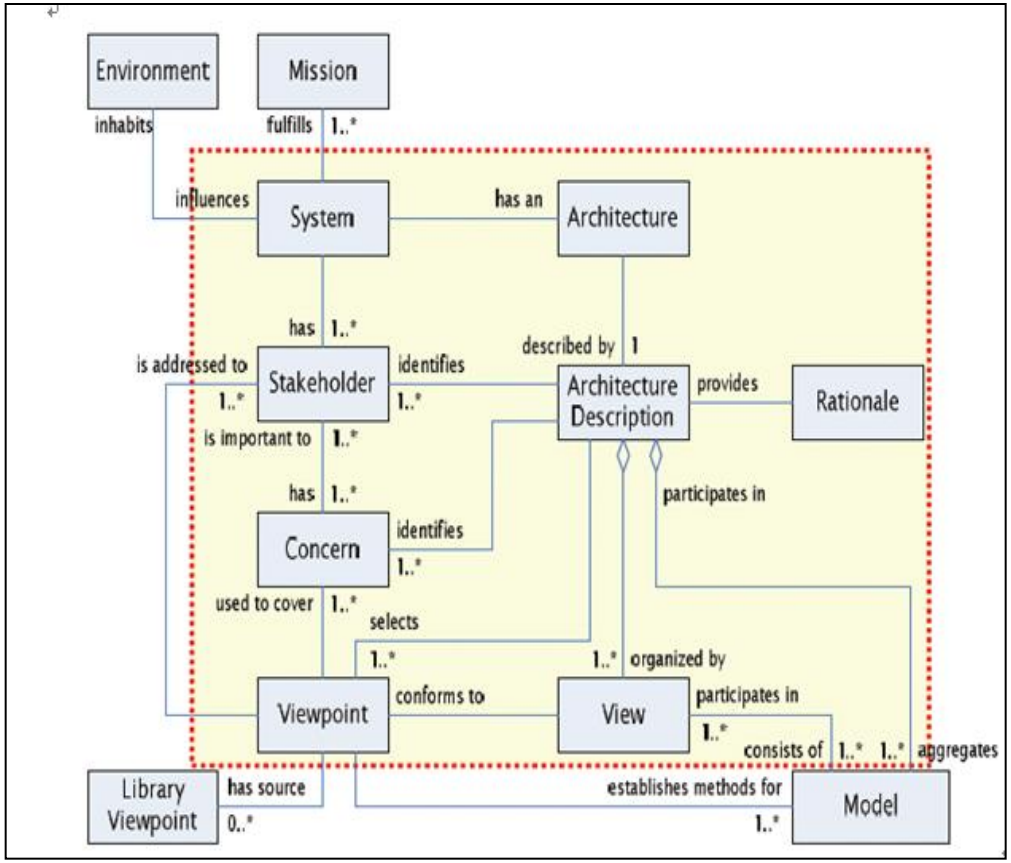
1) 논리/물리 시스템아키텍처의 작성

아키텍처의 각 산출물은 단위 프로젝트별의 각 개발 단계별로 시스템 아키텍처의 준수 사항을 제시하고 이를 준수하여 아키텍처의 설계여부를 파악할 수 있도록 활용됨



1) SW 아키텍처의 구성


기능,비기능 요구사항
요구사항 분석
분석
소프트웨어 아키텍처
참조
하드웨어 구성
설계 가이드라인
설계
분석문서
설계 문서
코드
코드가이드라인
배포 가이드라인
컴포넌트
시스템


Environment
Mission
inhabits
fulfills
1..*
influences
System
has an
Architecture
has
1..*
described by
1
is addressed to
1..*
Stakeholder
identifies
1..*
Architecture Description
provides
Rationale
is important to
1..*
has
1..*
participates in
Concern
identifies
1..*
used to cover
1..*
selects
1..*
1..*
organized by
Viewpoint
conforms to
View
participates in
1..*
consists of
1..*
1..*
aggregates
Library Viewpoint
has source
0..*
establishes methods for
1..*
Model

1) SAD(Software Architecture Document)

1. SAD 개요
- 1.1 SAD 목적 및 필요성 - SAD 구성 목차 정의, SA/SAD의 중요성 및 역할 설명
- 1.2 SAD 적용범위 - 전체 개발공정 및 이해관계자의 원칙, 변경관리 절차 표현
- 1.3 이해관계자 구성 - 고객/사용자/개발팀/운영팀/QA/TA등, 의사소통 전략 제시
- 1.4 view 정의 - 아키텍쳐표준정의, viewpoint/concern정의, view산출물

2. 아키텍쳐 배경
- 2.1 시스템 환경 - 목표시스템구성, Context구성, 솔루션, 개발/서버/운영환경
- 2.2 시스템 환경 제약사항 - 시스템제약사항(개발요소기술, 사용자PC환경, 기타제약)

3. 아키텍쳐 요구사항
- 3.1 품질모델 적용기준 - 적용품질모델선택, 품질속성/시나리오/유틸리티/달성전략
- 3.2 아키텍쳐 요구사항 - 요구사항도출, FRUPS모델기반 품질속성 시나리오 작성
- 3.3 아키텍쳐 영향요소 분석 - 유틸리티 트리에 의한 영향요소 분류 및 설계 전략 수립

4. 참조 아키텍쳐
- 4.1 참조 viewpoint - 참조가능한 후보viewpoint(4+1view, simens - 4view등)
- 4.2 참조 아키텍쳐 스타일 - 표준스타일정의(Layered:Presentation, Biz-Logic, Data)

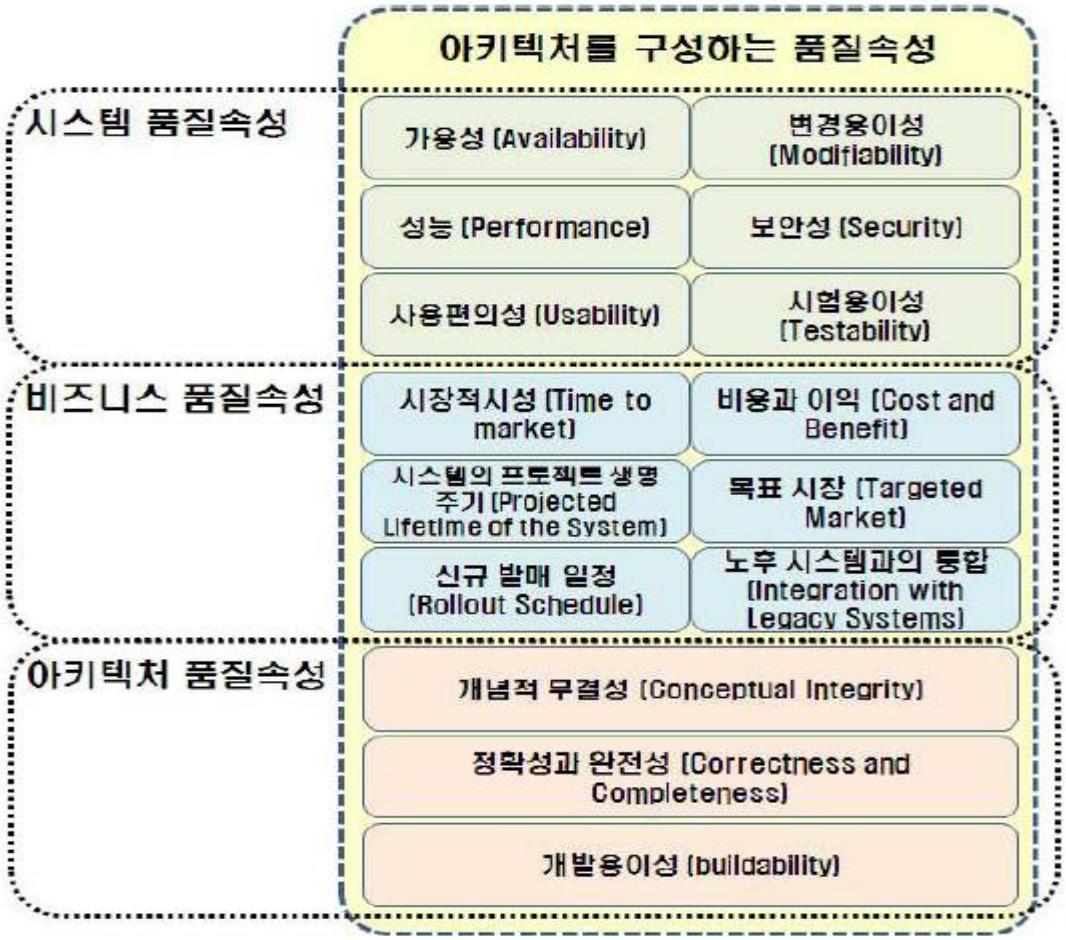
5. 설계 전략
- 5.1 업무 컴포넌트 설계 전략 - 업무 컴포넌트 도출 방안, 구현방안, 정제방안
- 5.2 공통 컴포넌트 설계 전략 - 공통 컴포넌트 도출 방안, 구현방안, 정제방안
- 5.3 시스템 공통 기능 설계 전략 - 공통기능정의, 인증/권한, 코드구현, 인터페이스, DB-Access등

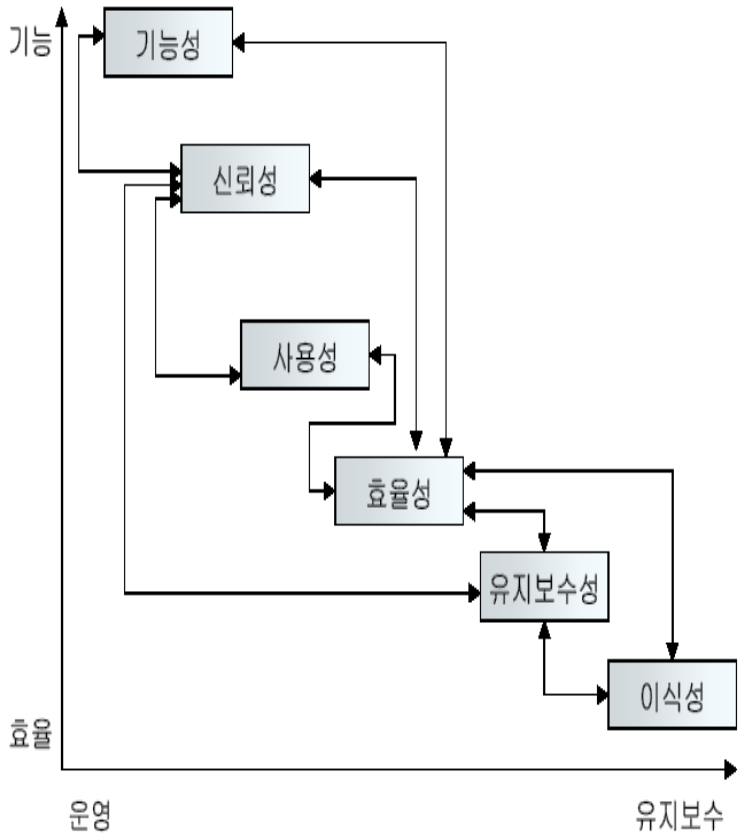
6. 시스템 뷰(view)
- 6.1 시스템 Overview - 개별 view에 대한 전체 요약, 누락부분 반영
- 6.2 view 사이의 관계 - 개별 view간 인터페이스 내용 및 제약 사항
- 6.3 Business Process View - 전제조건/제약사항, 작업흐름 뷰
- 6.4 Use-Case View - 전제조건/제약사항, 유즈케이스 뷰
- 6.5 Logical View - 전제조건/제약사항, Context뷰, 정적/동적뷰(분석), 논리ERD
- 6.6 Development View - 전제조건/제약사항, 정적/동적뷰(설계), 물리ERD
- 6.7 Deployment View - 전제조건/제약사항, 배포뷰, 패키지구조, 물리ERD

7. 기타
- 7.1 용어사전 - Glossary(시스템URL), Data-Dictionary
- 7.2 참고문서 - 참조도서, 참조표준

1) 아키텍처 품질 속성

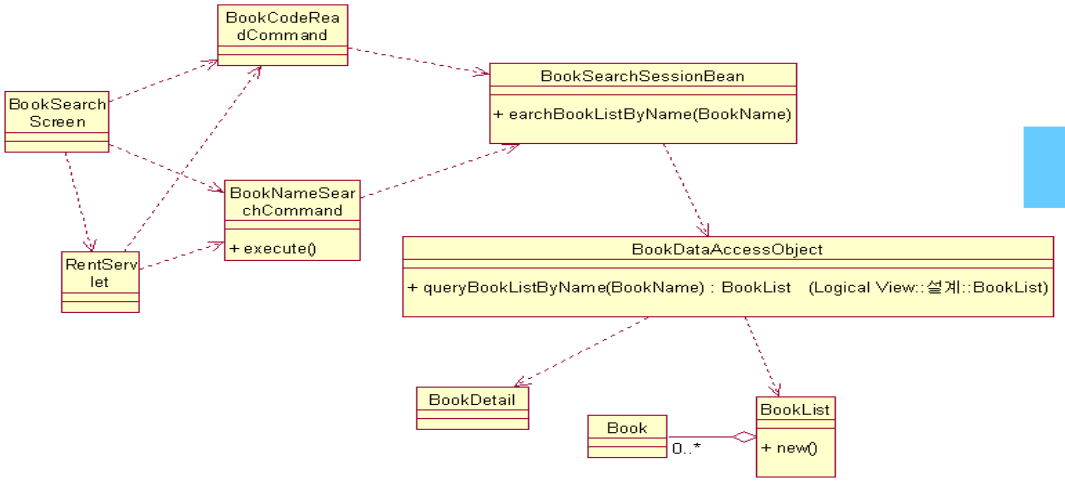
아키텍처의 품질은 최종적으로 개발된 시스템이 만족시켜야 하는 중요한 요구사항이다. 아키텍처 설계를 위한 품질은 각각의 품질요소마다 상호 배타적일 수 있어 복수의 품질 사이의 절충관계를 고려하여야 한다.

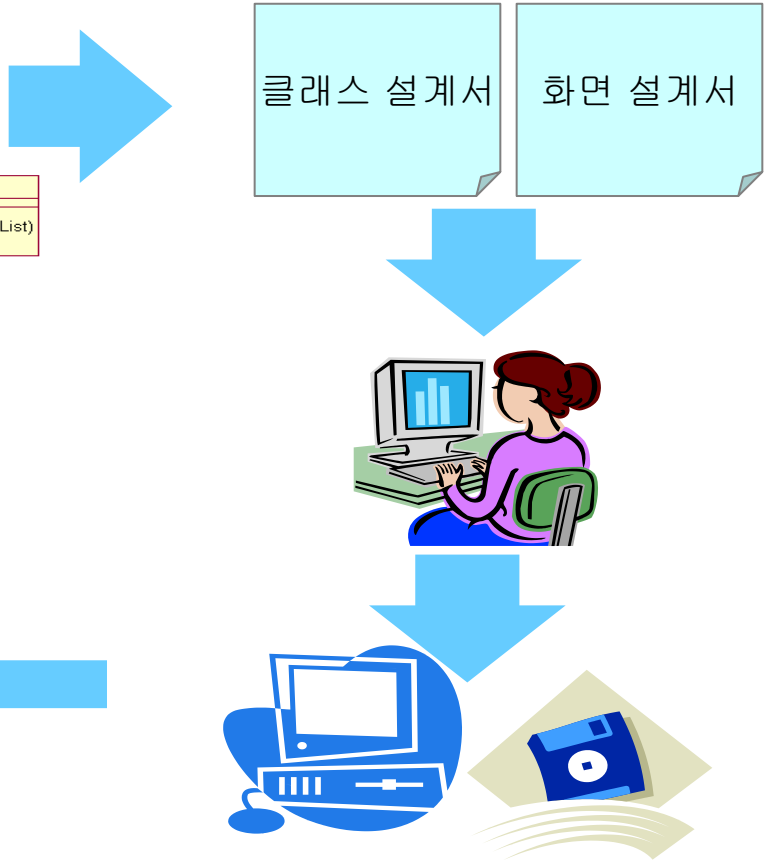




1) 아키텍처 모델

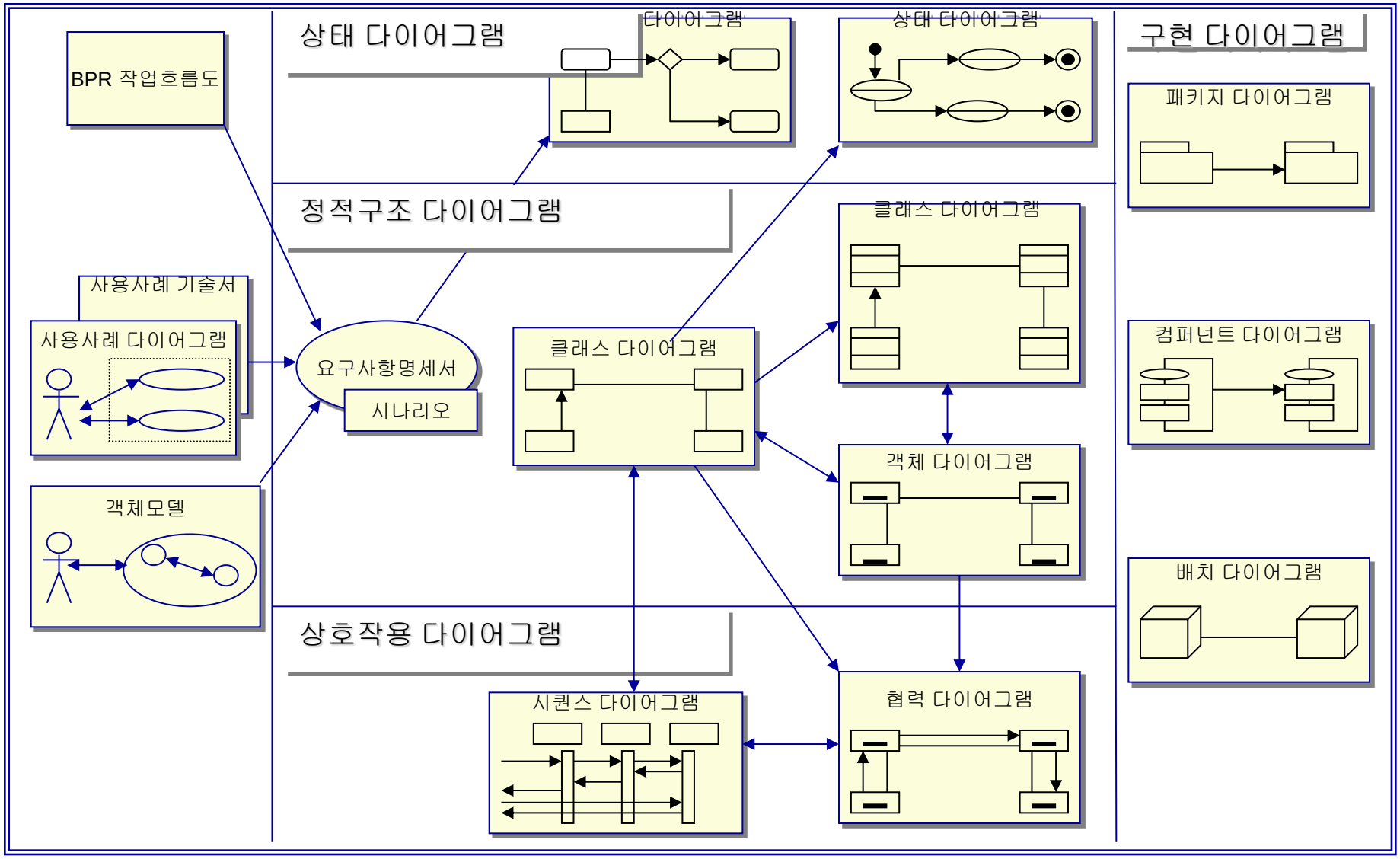
UML을 통하여 의사소통과 문서화가 용이하고, 소프트웨어 개발 과정을 잘 설명할 수 있음





- 설계모델의 문서화를 통한 설계 사양서 작성
- 개발 수행 후 테스트
- 변경 사항 모델 반영

1) UML(Unified Model Language)

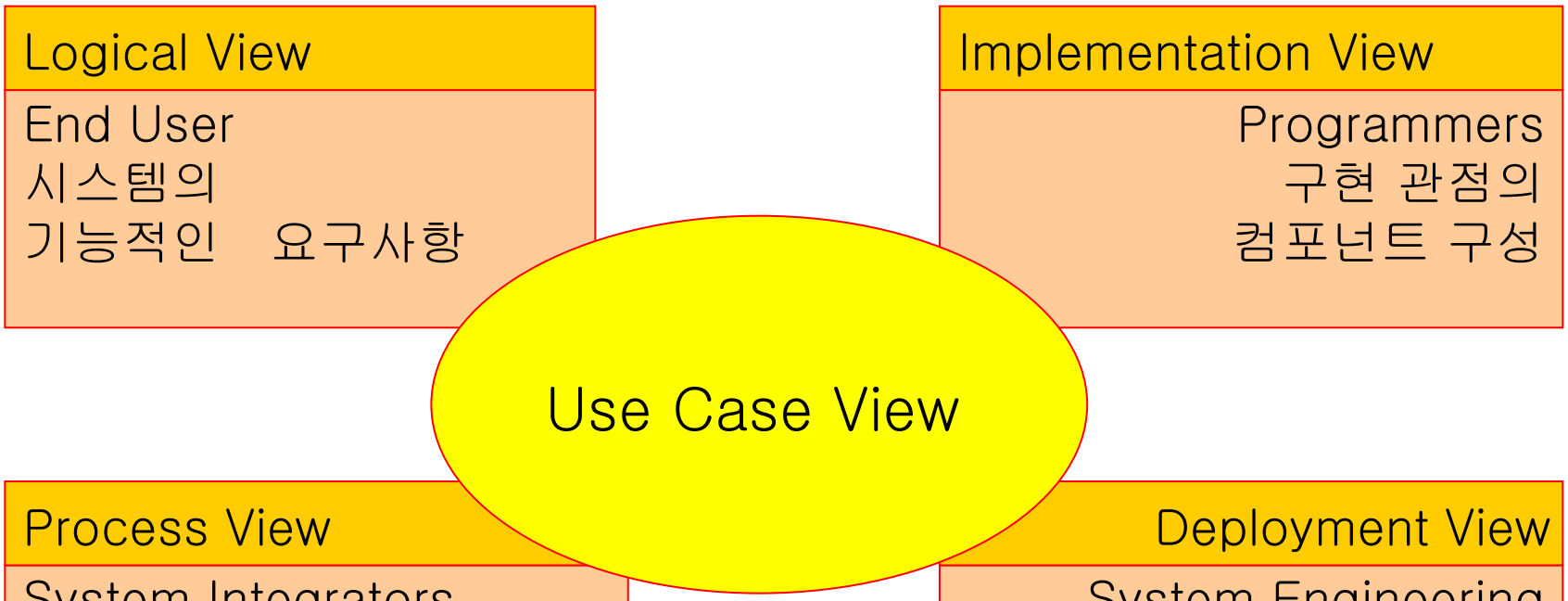


1) UML(Unified Model Language) 계속

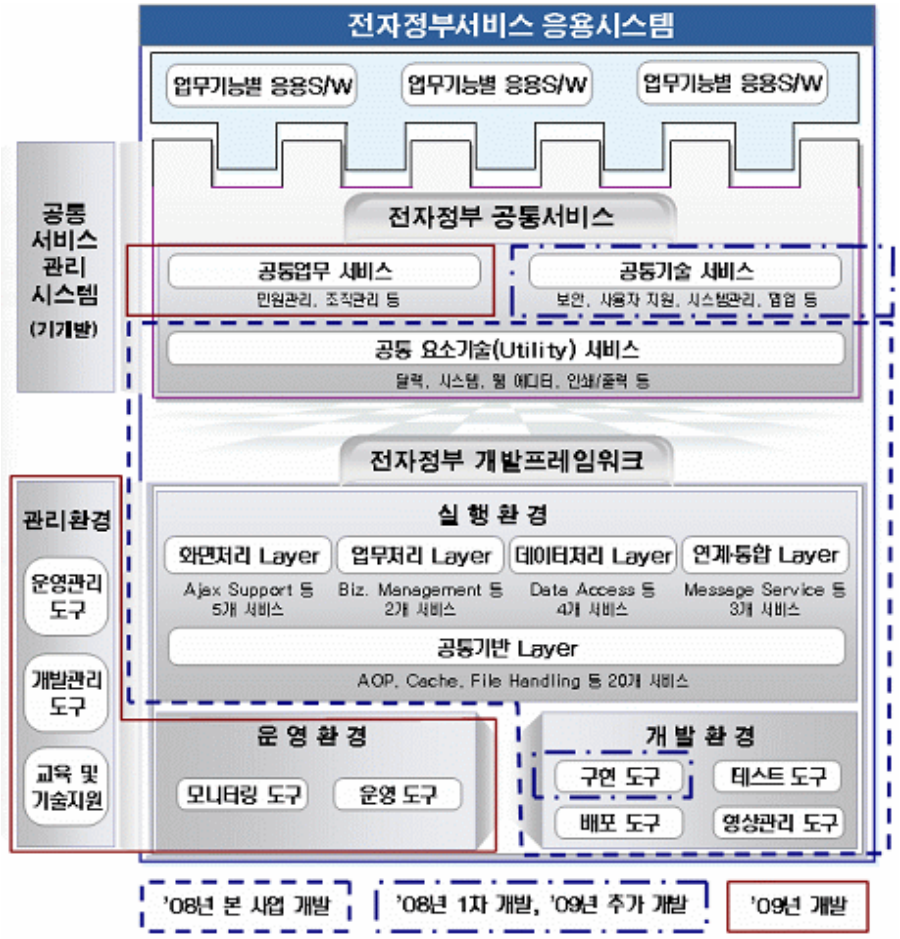
Diagram명	특징	정적/동적 구분
Use Case Diagram	사용자의 시스템에 대한 기능적 요구사항을 표현	Static
Activity Diagram	사용자의 업무흐름 또는 시스템 내부 로직의 분기 등의 흐름을 가시화	Dynamic
Sequence Diagram	Use Case가 동작 시 관련 객체들이 메시지를 주고 받는 것을 시간의 흐름에 따라 표현	Dynamic
Collaboration Diagram	Use Case가 동작 시 관련 객체들이 메시지를 주고 받는 것을 관계 중심으로 표현	Dynamic
Class Diagram	도출된 클래스의 내부 구조와 연관관계를 표현	Static
Statechart Diagram	시스템의 주요 상태와 상태를 변화시키는 이벤트를 표현	Dynamic
Object Diagram	클래스들이 실체화된 객체와 그 관계를 표현	Static
Component Diagram	개발 모듈의 파일 단위와 호출 관계를 표현	Static
Deployment Diagram	개발 모듈의 설치 Node의 위치와 Node간 관계를 표현	Static

- 하나의 시스템을 견고히 설계하기 위해서는 다양한 관점(View)의 도면이 필요
- 일반적인 정보처리 시스템에서는 Use Case, Sequence, Class Diagram이 가장 빈번하게 사용되며 가장 중요

1) 4+1 View


Logical View
End User
시스템의
기능적인 요구사항
Implementation View
Programmers
구현 관점의
컴포넌트 구성
Use Case View
Process View
System Integrators
실행 시 성능을
고려한 프로세스 구성
Deployment View
System Engineering
물리적인
시스템의 구성 및
분산,인도,설치 표현

1) 전자정부 표준 개발프레임워크 아키텍처



가. 실행환경 구성

서비스그룹	설명
Presentation Layer	프로그램, 사용자간의 Interface를 담당. 사용자 화면 구성, 사용자 입력 정보 검증 수행 (Spring MVC, Ajax Tags 등)
Business Logic Layer	프로그램의 업무 로직을 담당. 업무흐름제어, 에러처리 수행(Spring WebFlow 등)
Persistence Layer	프로그램에서 사용하는 데이터에대한 제어를 제공 (iBatis, Hibernate 등)
Integration Layer	타시스템과의 연동 기능을 제공 (Apache CXF 등)
Foundation Layer	각 Layer에서 공통적으로 활용되는 공통 기능 제공 (Spring, Apache Commons 등)

나. 개발환경 구성

서비스그룹	설명
Implementation Tool	업무 프로그램 구현을 지원 하는 도구 (Eclipse, AmaterasUML 등)
Test Tool	구현된 업무 프로그램의 테스트하는 도구 (Junit, EasyMock 등)
Deployment Tool	구현 완료된 업무 프로그램을실행 환경에 배포 가능한 형태로 패키징하고 패키징 파일을실행환경에 배포하는 도구 (Maven, Hudson 등)
Configuration & Change Tool	형상 및 변경관리 지원 도구 (Subversion 등)

1) 전자정부 표준 개발프레임워크 아키텍처(계속)

응용SW구성의 기반

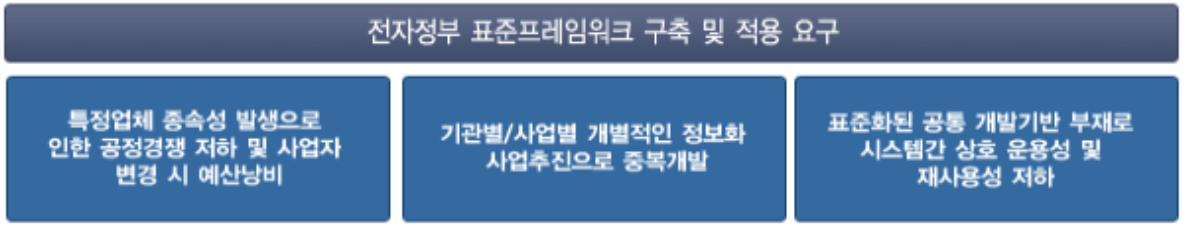
- 전자정부 표준 프레임워크는 응용SW의 구성기반이 되며 응용SW실행 시 필요한 기본 기능을 제공하는 환경

개발프레임워크의 표준

- 전자정부 서비스의 품질향상 및 정보화 투자 효율성 향상을 위해 개발 프레임워크 표준을 정립

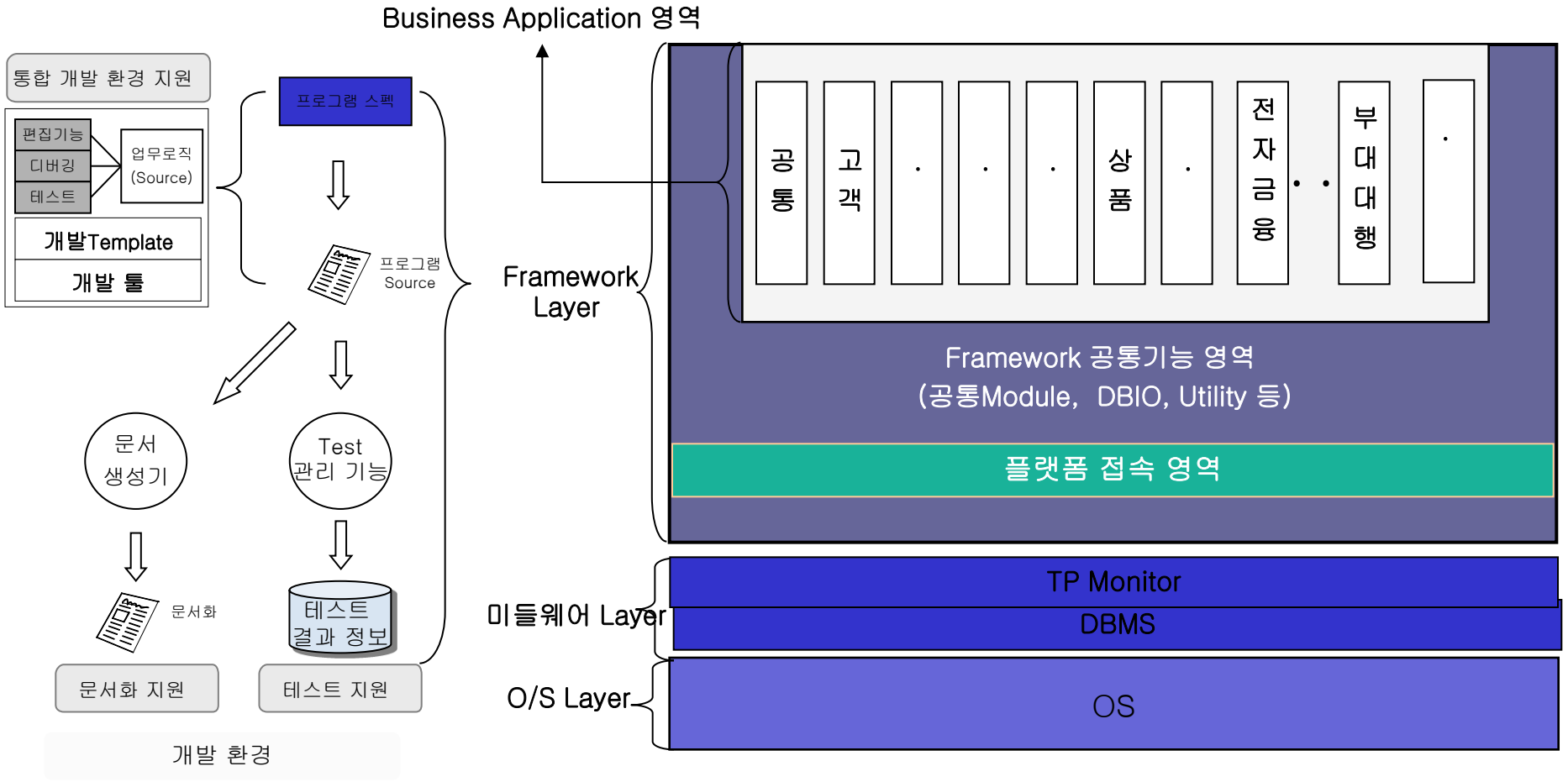
표준화, 재사용성 향상

- 개발 프레임워크 표준 적용을 통한 응용 SW의 표준화 및 품질과 재사용성 향상을 목표



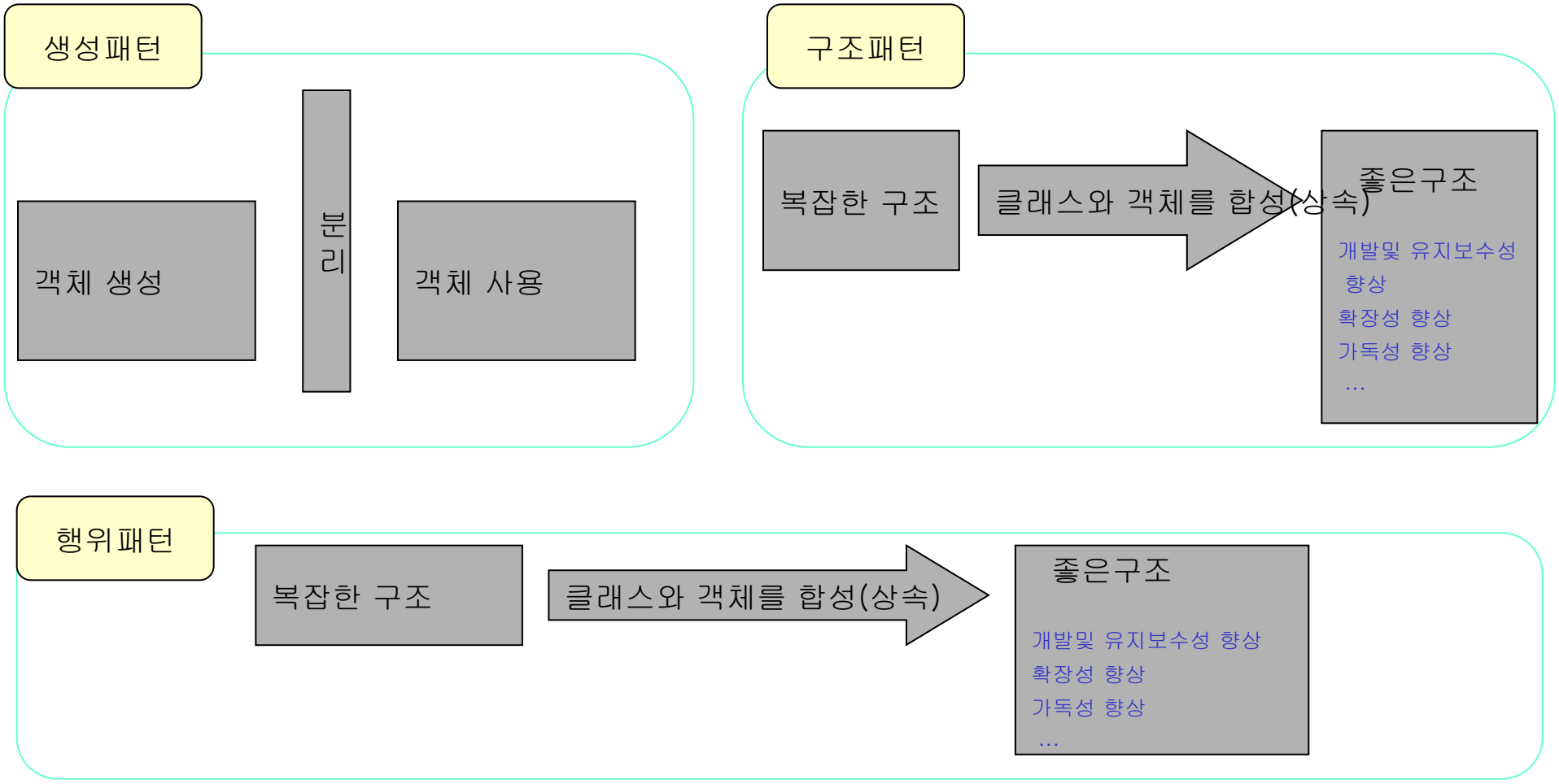
1) 어플리케이션에서 프레임워크 의 위치

Framework는 미들웨어 상에서 Business Application을 개발, 운영하기 용이하도록 기본적이고 공통적인 기능을 제공함으로써 개발의 생산성, 표준화된 Application , 운영의 용이성 등을 제공



1) 디자안패턴

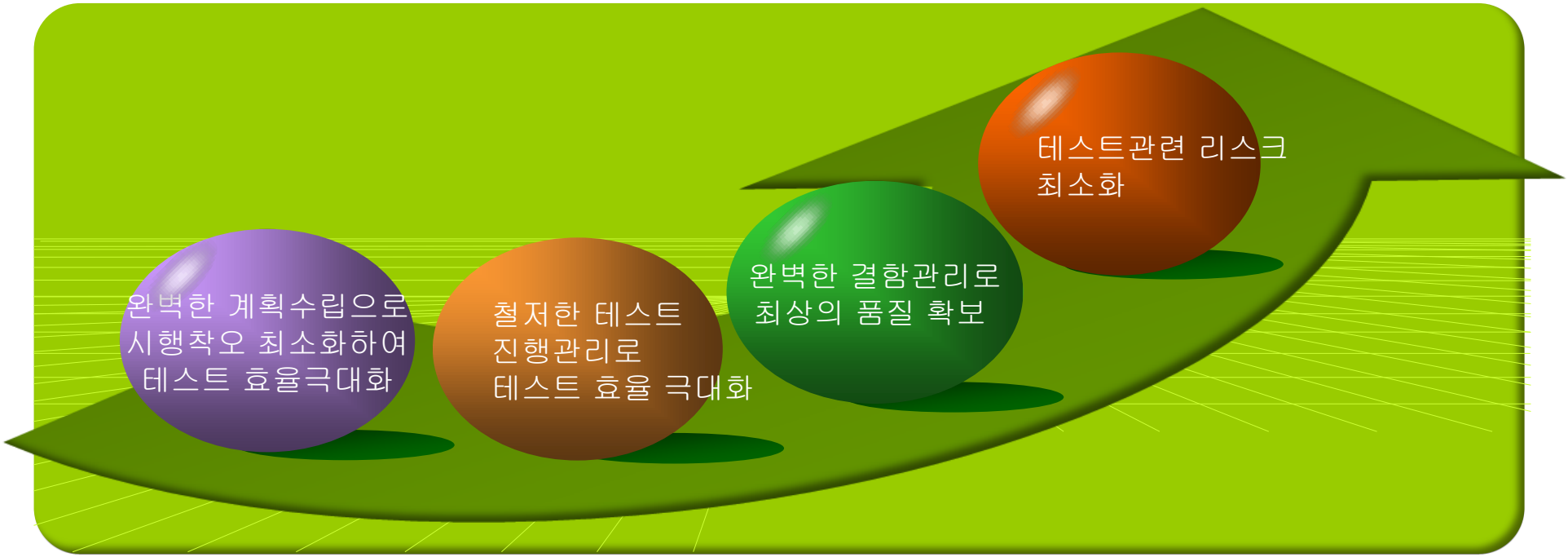
디자인패턴은 개체지향의 여러 이론들을 시스템의 구조속에서 실제로 구현되도록 만든 여러 유형들의 집합임



Ⅴ. 시험계획 및 실시 핵심 기법

1) 마스터 테스트 플랜

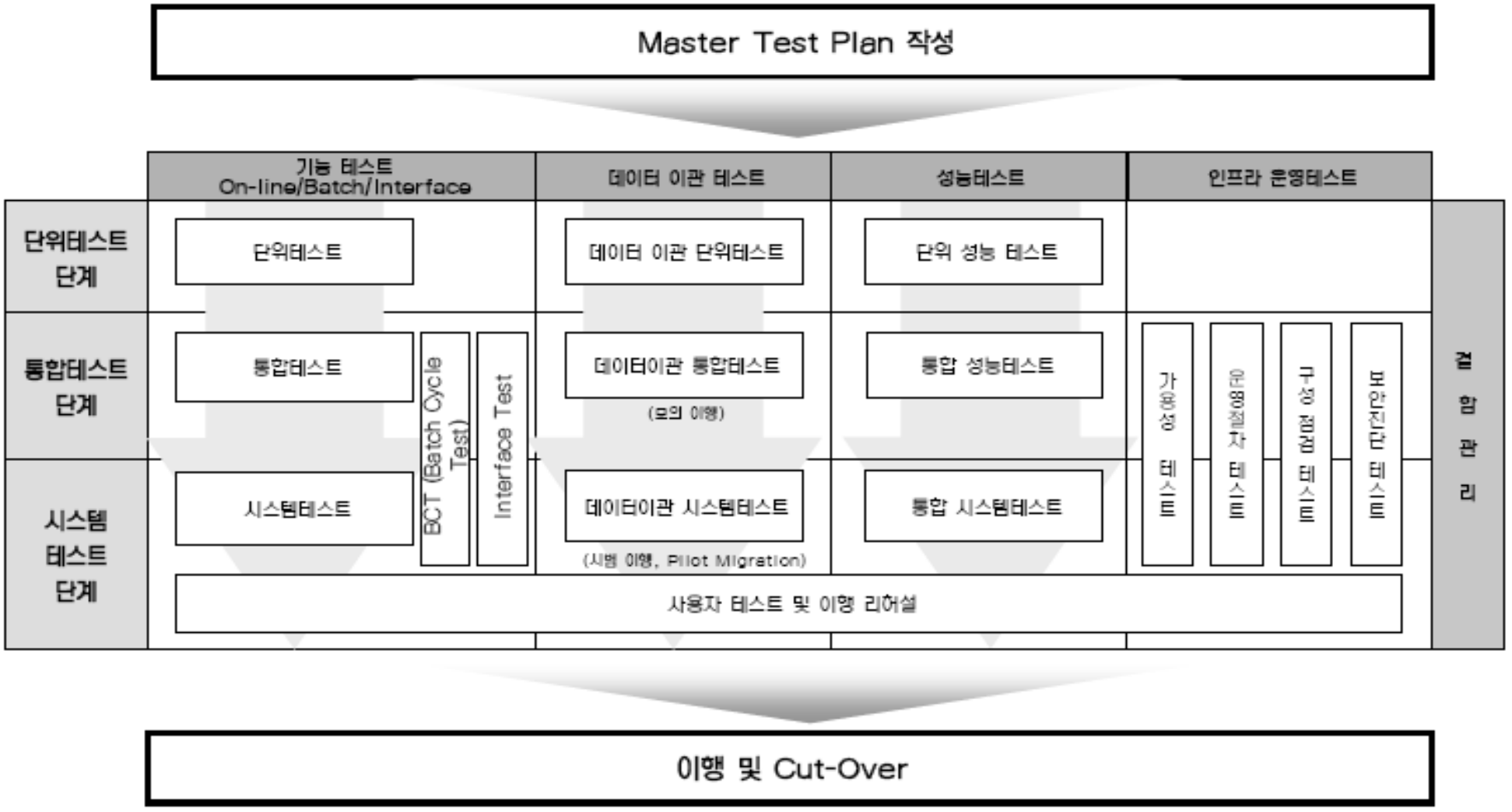
프로젝트의 유형별/단계별 테스트수행 효율 극대화를 통해 품질 ,일정 및 비용 면에서 프로젝트 목표를 달성하고 성공적인 Go-Live로 연결하기 위해 마스터 테스트 플랜을 수립



- 구축 프로젝트의 테스트 전략과 원칙을 제시하여, 모든 프로젝트 멤버들이 공유하여 같은 기준으로 유형별 테스트 진행 (프로젝트 고유특성은 반영함)
- 명확히 정의된 유형별 테스트 범위 및 테스트 Level, Type에 따른 테스트 수행
- 테스트 단계에서 프로젝트 Risk를 조기에 발견하고 대응할 수 있는 체계 구축
- 테스트 조직 구성 및 명확한 R&R 부여

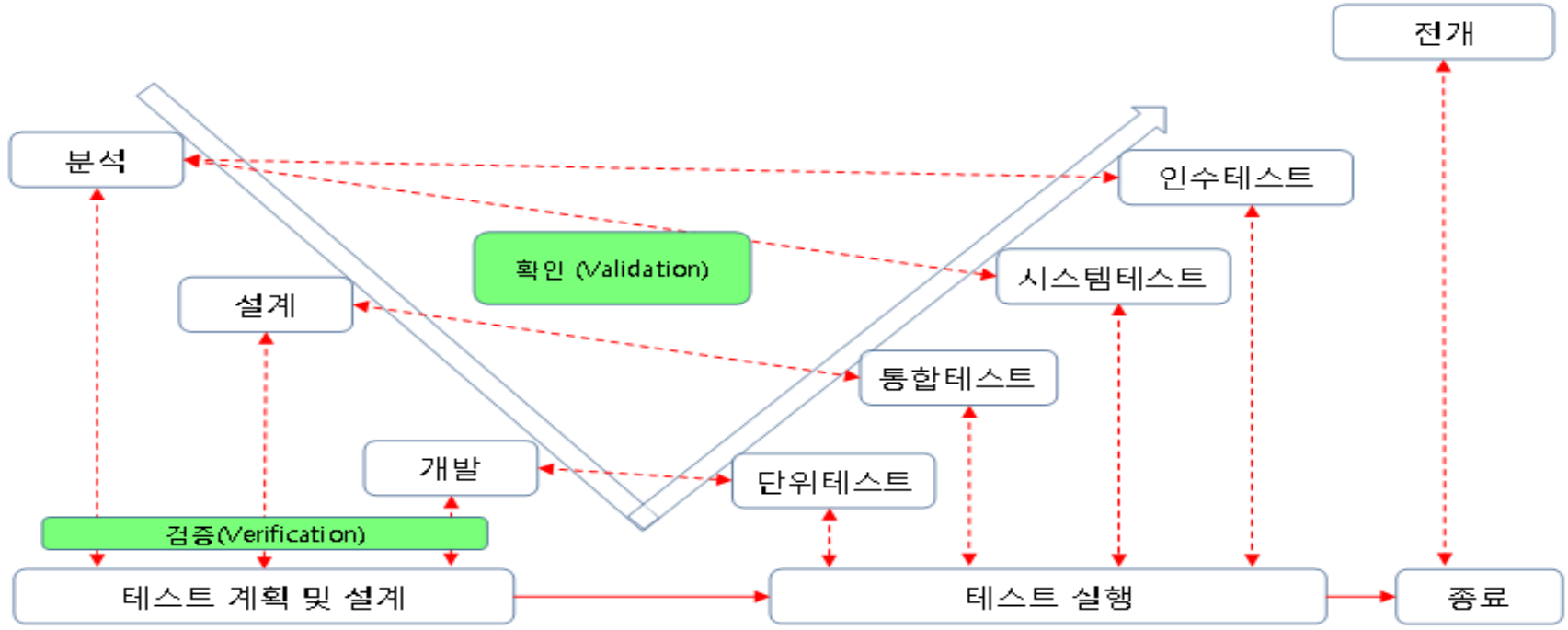
1) 마스터 테스트 플랜 (계속)

- **MTP 구성도**



1) 테스트의 V-Model

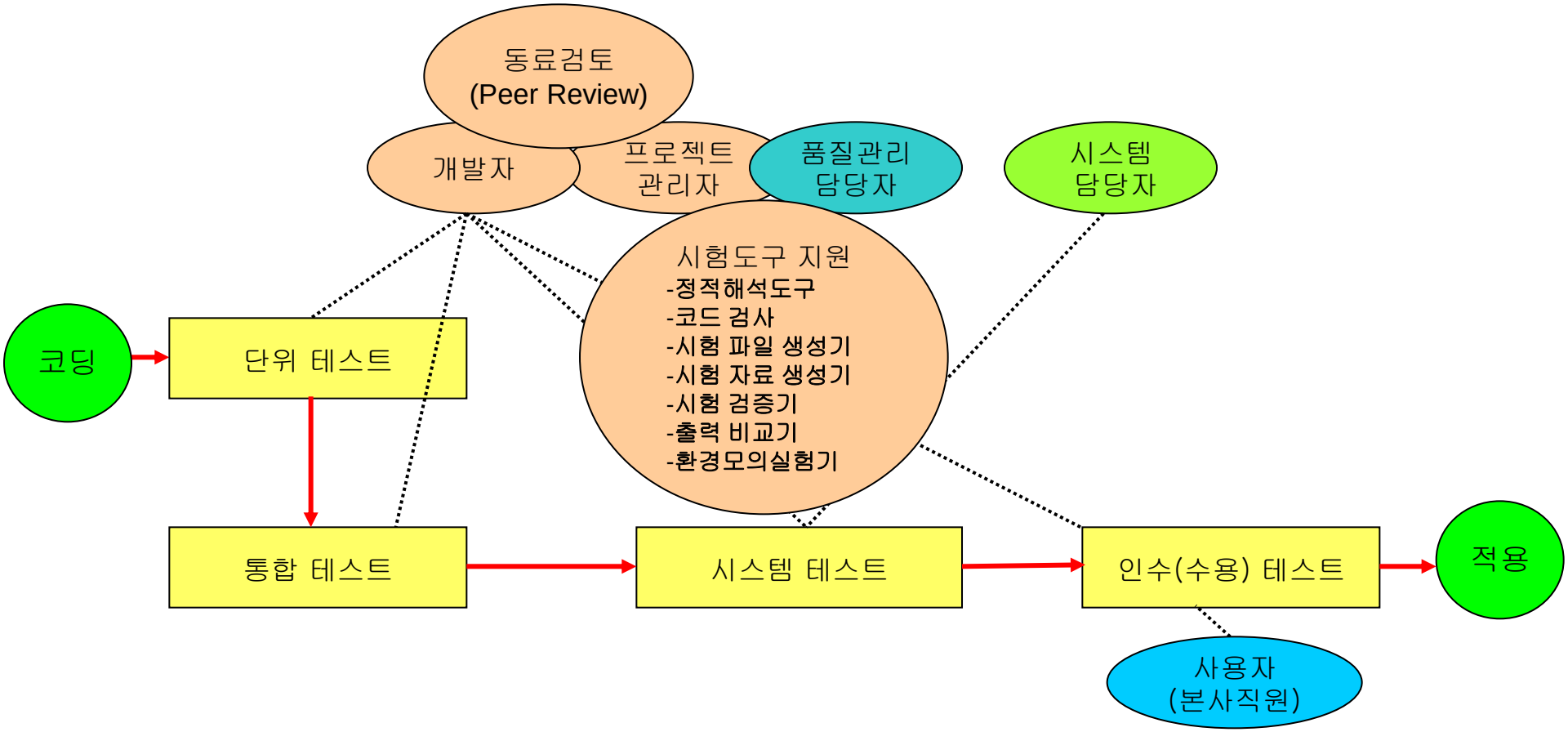
V-Model은 소프트웨어 전 개발주기에 걸쳐서 지속적으로 테스트를 수행해야 한다는 테스트 라이프 사이클 개념이 반영되었으며, 분석/설계 단계에 테스트 계획 및 설계를 완료하고, 개발단계 이후에는 테스트 실행에만 집중함



- Human testing(Verification)과 Computer-based testing(Validation)을 통해서 단계별 결함을 검출하고 이후단계로의 전이를 예방하기 위해서 개발 단계와 테스트를 Mapping해놓은 것
- 개발(분석/설계) 산출물과 테스트계획/설계 산출물과의 검증(Verification)활동을 수행하며, 단계별 테스트를 수행하면서 확인(Validation)활동을 수행함

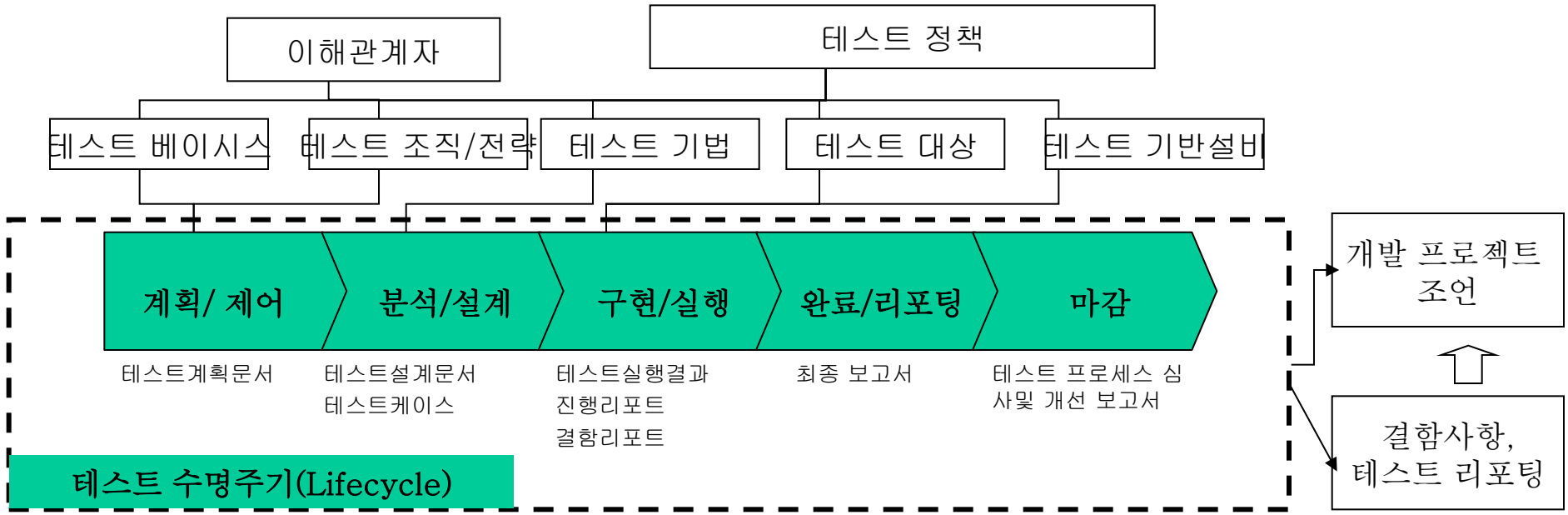
1) 테스트 분류

테스트는 단위테스트, 통합 테스트, 시스템 테스트, 인수 테스트의 단계를 수행함



1) 테스트 프로세스 연관

테스트 프로세스는 테스트 수명주기 활동을 중심으로 모든 관련 구성요소와 관련된 사항과 활동을 포함한다



- 테스팅은 다양한 구성요소를 테스트 수명주기와 프로세스를 중심으로 조정하고 관리하는 것을 포함하며, 테스팅을 통하여 발견한 결함과 관련 정보를 바탕으로 정량적으로 개발 프로젝트에 조언한다

1) 테스트 케이스

특정한 프로그램 경로를 실행해 보거나 특정한 요구사항에 준수하는지를 확인하기 위해 개발된 입력 값, 실행 조건, 그리고 예상된 결과의 집합이다

필 드 명	필 드 설 명
화면명(프로그램명)	화면명에 해당하는 화면ID를 적는다. 배치 프로그램에 해당하는 프로그램ID를 적는다.
요구사항 ID	분석/설계 단계에서 생성된 요구사항 ID 를 표기한다.
요구사항 내용	요구사항 내용을 간략히 기술한다.
관련프로그램	요구사항과 관련된 프로그램을 기술한다
테스트케이스 명	테스트케이스 명은 해당 테스트케이스를 수행하여 달성하고자 하는 목표와 관련이 있다. 목표를 간단히 줄여서 목표로 정한다.
테스트 내용	테스트 내용을 적어야 하는 이유는 테스트케이스는 축적되어, 향후 누구라도 실행이 가능해야 하기 때문이다. 따라서, 제3자가 이해할 수 있도록 객관적으로 기술되어야 한다. 테스트케이스에 대한 구체적인 설명을 기술하도록 한다.
테스트 데이터	제3자가 수행하기 위해서는 테스트데이터도 구체적인 값이 표기되어야 한다
예상 결과	확인내용:테스트케이스를 실행시켰을 경우, 성공을 확인할 수 있는 값이나 메시지에 대해 기술한다
환경설정	테스트 수행할 때 필요한 하드웨어나 소프트웨어 환경
특수절차요구	테스트 케이스 수행 시 특별히 요구되는 절차

1) 테스트설계 기법

블랙박스 테스트, 화이트 박스 테스트를 통하여 커버리지를 충족하는 테스트 케이스를 도출할 수 있다.

1. 블랙박스 테스트

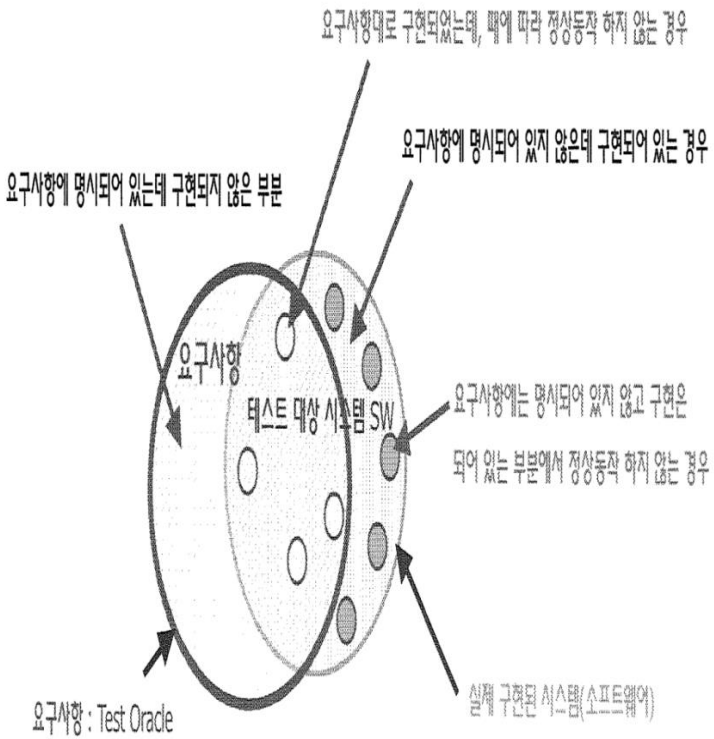
구 분	내 용
동등 분할 기법	- 다양한 입력 조건의 TEST 사례들을 선정하여 TEST - 예) 1~100 범위에서 x <0, 0, x, 100, x >100으로 구분하여 TEST
경계값 분석 기법	- 경계값을 기준으로 결과의 정확성을 TEST - 예) 1~100 범위에서 x=0, x=100, x=-1, x=200등으로 TEST
원인,결과 그래프 기법	- 입력값이 출력값에 미치는 영향을 그래프로 표현하여 오류검출
오류예측 기법	- 간과할 수 있는 오류들을 감각과 경험으로 검출하는 기법 - 예) 입력 값없이 확인하거나, TEXT 입력란에 숫자입력 등

2. 화이트 박스 테스트

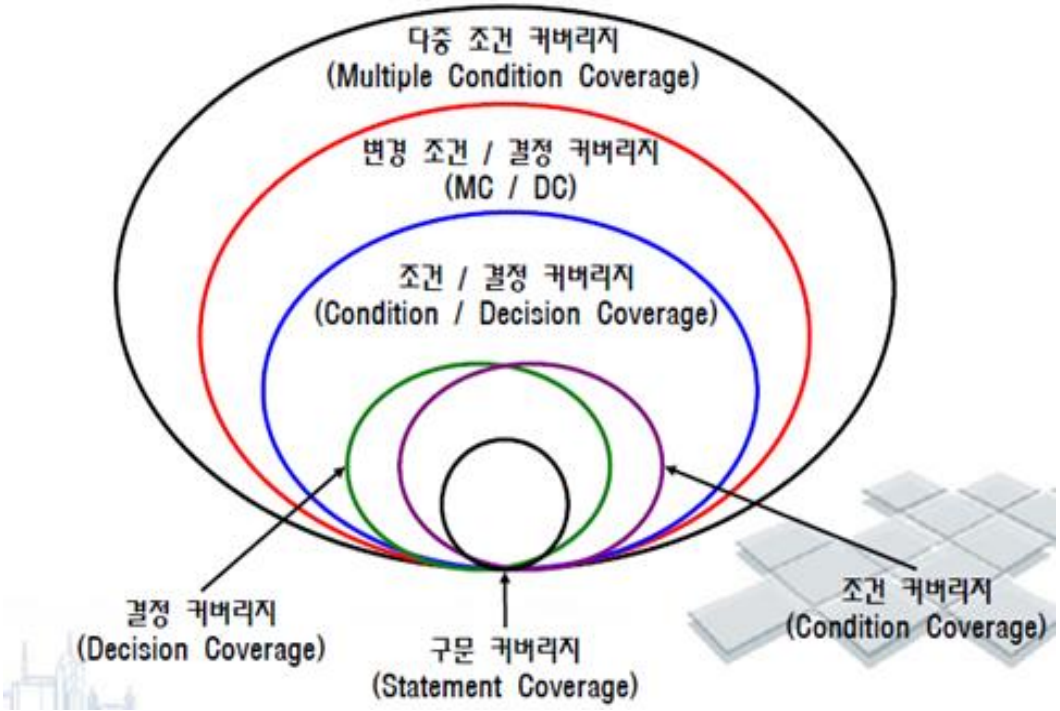
구 분	내 용
구조적 기법 (Structural test)	- 프로그램의 논리적인 복잡도를 측정하여 평가
루프 TEST (Loop test)	- 프로그램의 Loop 구조에 국한하여 실시하는 기법 - 초기화 값, 인덱스와 증가분, Loop의 경계 값 등의 오류 검출

1) 테스트설계 기법(계속)

테스트 커버리지를 최대화하기 위해서 프로그램 내부 구조에 존재하는 모든 경로를 실행 하도록 테스트 케이스를 설계 할 수 있다



테스트 커버리지를 최대화 한다
최대한 많은 수의 결함을 찾아낸다



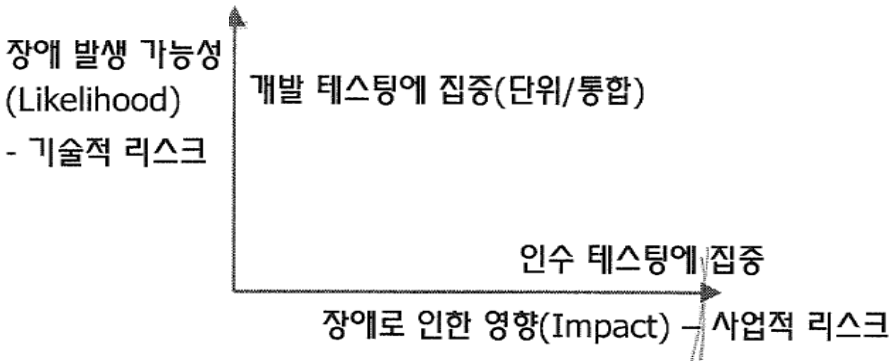
- 제어흐름 테스트기법을 통해 구문 커버리지와 결정 커버리지를 달성하고 최소 비교 테스트 기법을 통해 변경조건/결정 커버리지를 달성할 수 있음

1) 리스크 기반 테스트

리스크에 대한 정량적 분석을 통하여 우선순위가 높은 부분에 테스팅 자원이 집중하여 전체적인 비즈니스 영향을 최소화 하는 것이 중요하다.

리스크요소 / 리스크아이템	장애발생가능성					영향		
	복잡성	새로운기능	상호관계	크기	기술 난이도	사용자 중요도	사용빈도	민원소지
리스크아이템1	1	3	1	3	3	1	1	1
리스크아이템2	2	3	1	5	5	5	5	5

이해관계자가 리스크 테이블 작성
리스크레벨 : 9=심각, 5=높음, 3=보통, 1=낮음, 0=없음

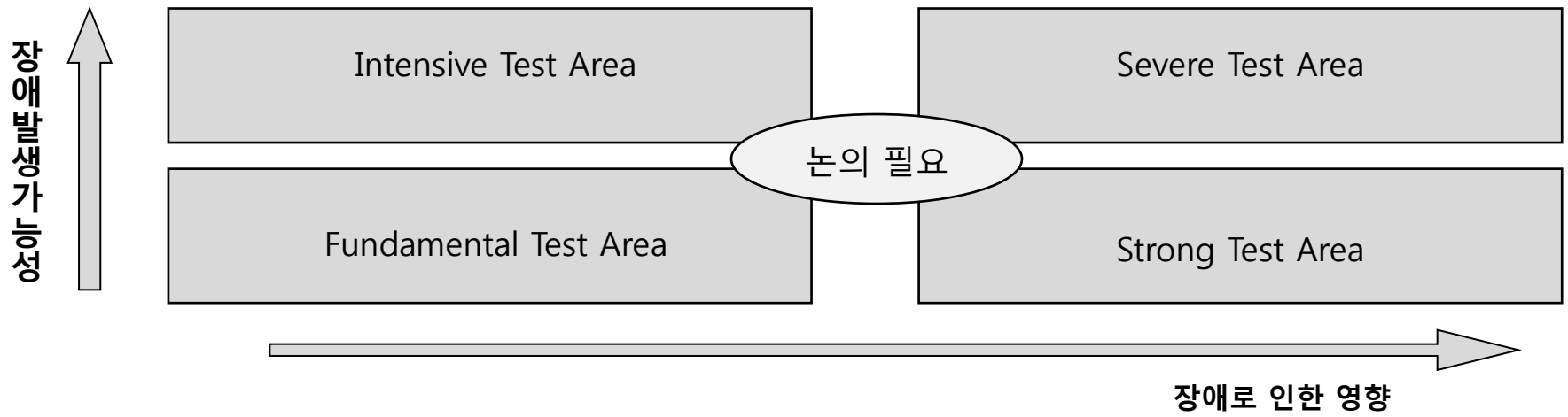


리스크를 기술적 리스크와 사업적리스크로 분리

- 기술적 리스크는 단위/통합 테스팅과 관련됨
  (장애발생 가능성 관리)

- 사업적 리스크는 인수 테스팅과 관련됨
  (발생한 장애로 인한 영향관리)

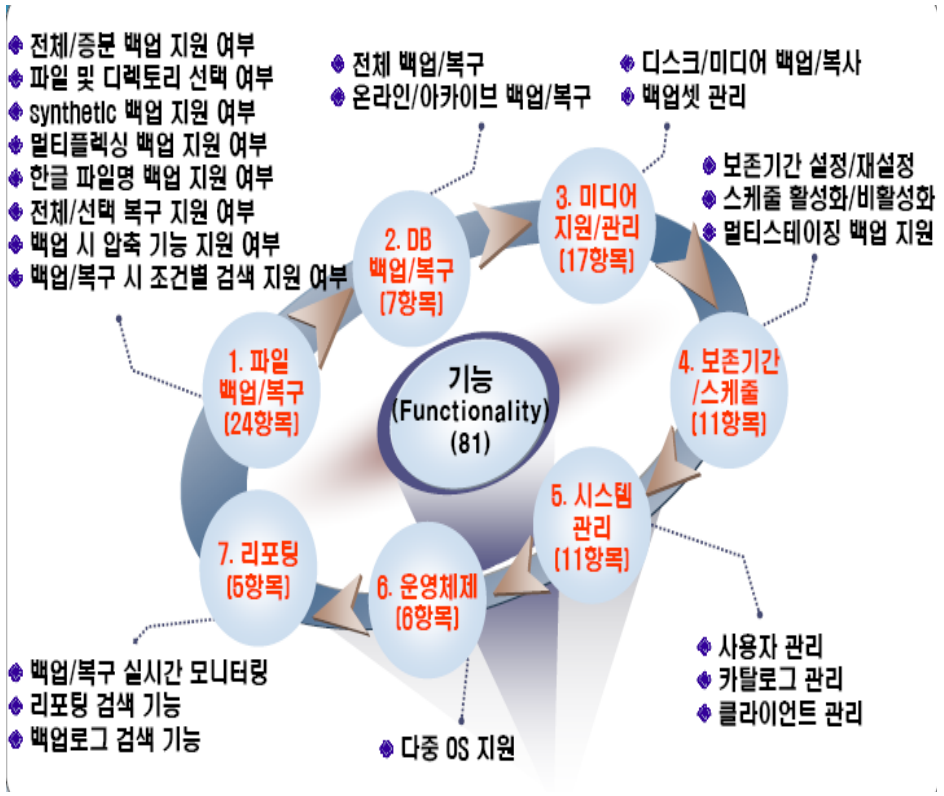
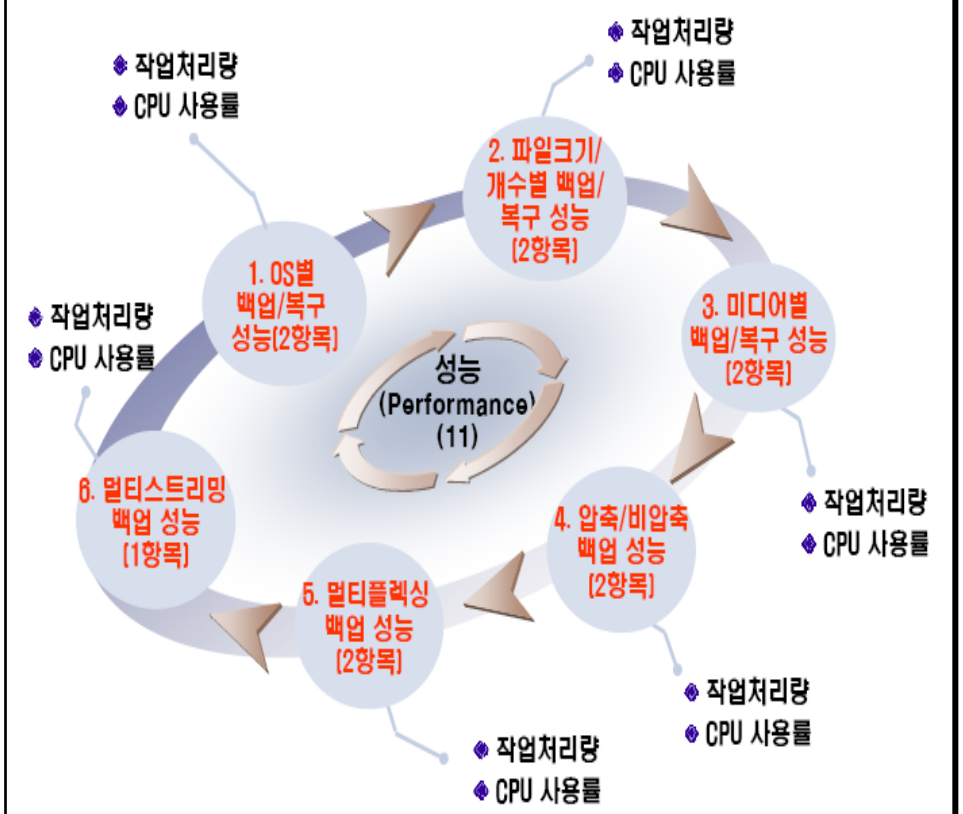
1) 리스크 기반 테스트(계속)



리스크 레벨	우선 순위	설계기법	완료조건
Severe Test Area	1	-유스케이스 테스팅 -페어와이즈 + 경험적 포함 -경계 값 분석 -탐색적 테스트	-모든 가능한 테스트 케이스 수행 -Workaround결함 50개 이하 -재테스팅 3회/ Full리그레션 테스팅
Strong Test Area	2	-유스케이스 테스팅 -페어와이즈 -경계 값 분석	-모든 가능한 테스트 케이스 수행 -Workaround결함 100개 이하 -재테스팅 2회/ Full리그레션 테스팅
Intensive Test Area	3	-유스케이스 테스팅 -동등 분할	-90% 이상의 기능 테스트 케이스 수행 -재 테스트 1회/Partial 리그레션 테스팅 완료
Fundamental Test Area	4	-유스케이스 테스팅(기본흐름만)	-80% 이상의 기능 테스트 케이스 수행 -재 테스트 1회/확인 테스팅 완료

1) BMT(BenchMark Test)

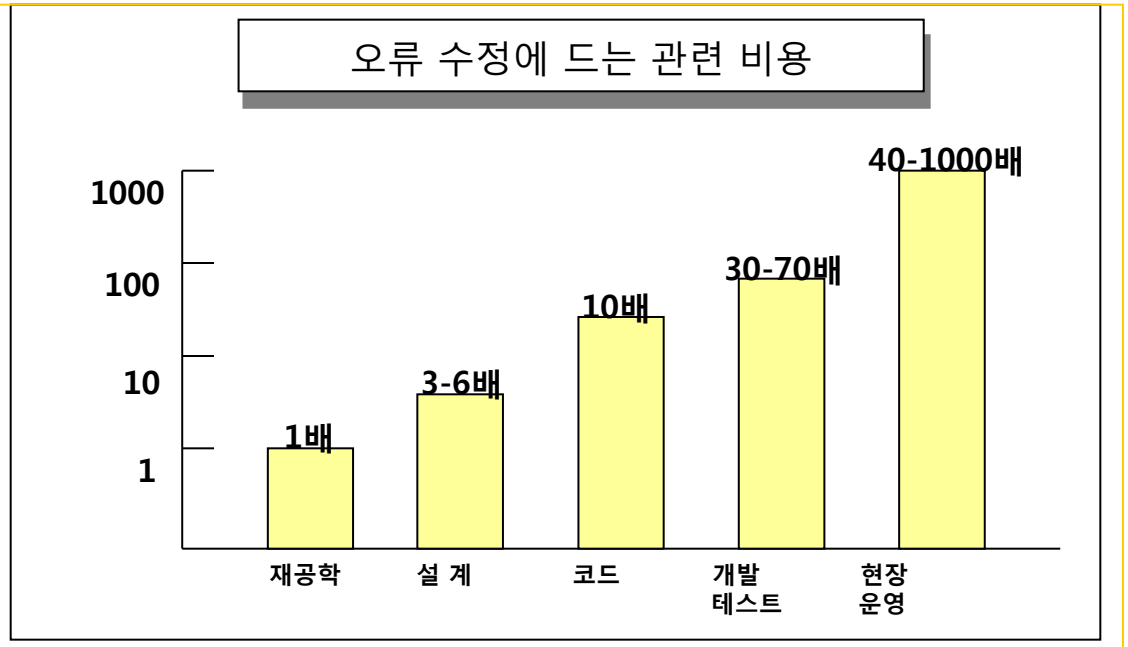
업체 및 솔루션과 장비 선정을 가격만 가지고 입찰하는 게 아니라 여러 제공 기능이나 유지보수 능력 등을 비교 평가하기 위하여 BMT를 실시한다

기능별 평가항목	성능별 평가항목
	

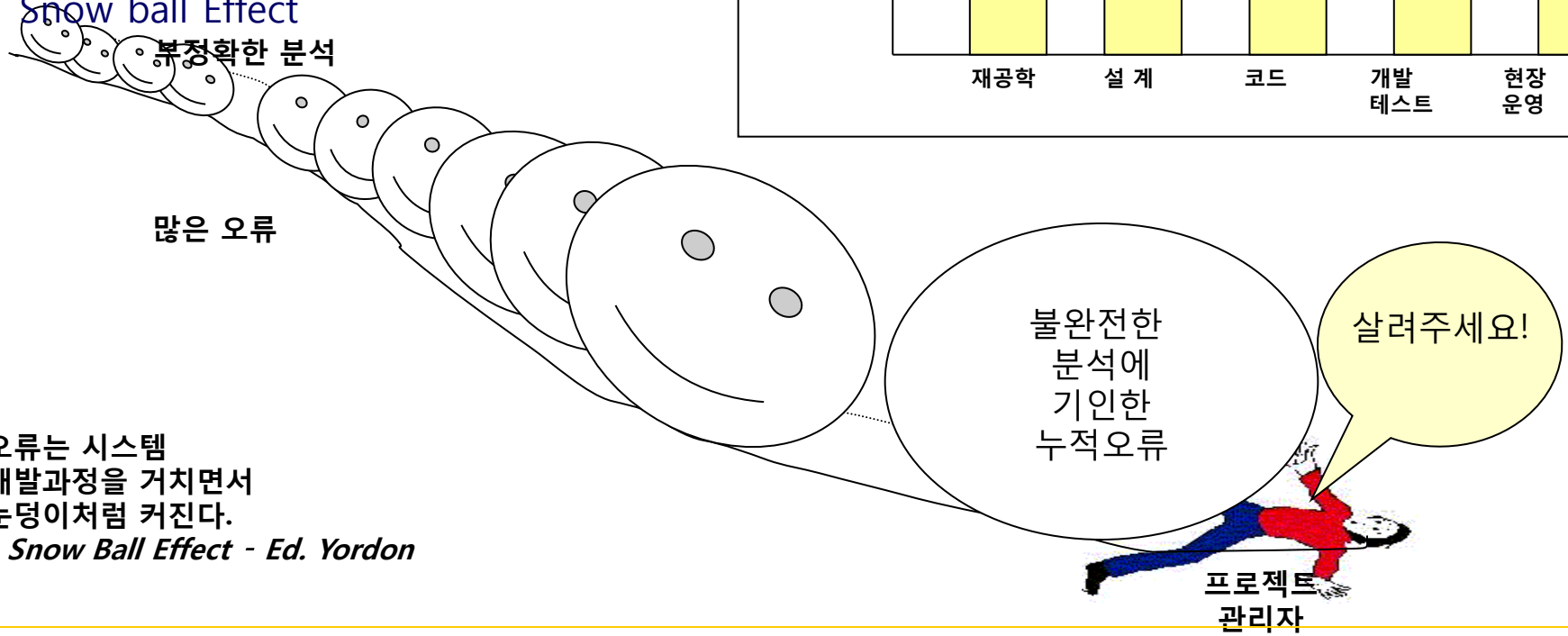
1) BMT(BenchMark Test)(계속)

절차	상세 내용
BMT 요청	- 대상 업체 선정 후 관련 업체에게 BMT 관련 HW/SW 요청, 일정 및 기간 통보
BMT 평가항목 도출	- 기능 관련 항목 : 기본 기능, 적합성, 정확성, 유연성, 상호호환성, 보안성 - 성능 관련 항목 : 신뢰성, 효율성, 장애 복구성, HW/SW 성능, HA 여부 등
BMT 평가방법 선정	- 평가 도구, 평가 체크리스트, 평가 등급, 평가 횟수 선정, 평균값/최대값 중 선택 - BMT 평가 방법 선정에 따른 BMT 시나리오 구성
BMT 시험환경 구성	- 운영환경과 관련 없는 별도의 Test Bed 구성, 인원 및 일정에 따른 시험 환경 구성
BMT 수행	- 세부 평가 항목별로 요청된 기본 기능의 존재 여부 확인 - 평가 내용에 따라 정상 동작 여부, 최적의 성능 발휘 여부 확인
BMT 결과 확인	- 결과 자료 확인, 선정 이유, 공정성 여부 확인화여 결과 공표 후 우수 사업자 선정

1) 소프트웨어 테스트의 경제성



Snow ball Effect



오류는 시스템
개발과정을 거치면서
눈덩이처럼 커진다.
- Snow Ball Effect - Ed. Yordon

1) 소프트웨어 검토

소프트웨어 공학 과정에서의 여과기로, 즉 개발기간 중 제거할 수 있는 결점들을 찾는 과정이다

❖ 검토의 필요성 (Freedman and Weinberg)
- 기술적인 일은 당연히 검토를 필요로 하며 오류의 원인은 사람이다.
- 개발자가 자신의 몇몇 오류를 찾아내는데 능숙하지만 대부분의 오류는 다른 사람들보다 오히려 찾기 어렵다.

❖ 소프트웨어 오류의 비용 영향
1) 기술검토의 이익은 후속단계 이전에 오류를 조기에 발견 하는 것
2) 연구단체의 통계치 (대형 프로젝트)
- 설계활동이 모든 오류의 50~60% 차지
- 기술검토는 설계오류의 75% 지적 효과

❖ 결함증폭 모델
- 소프트웨어 공학 과정의 설계 및 코딩단계에서 결함의 발생과 검출을 나타내는데 이용

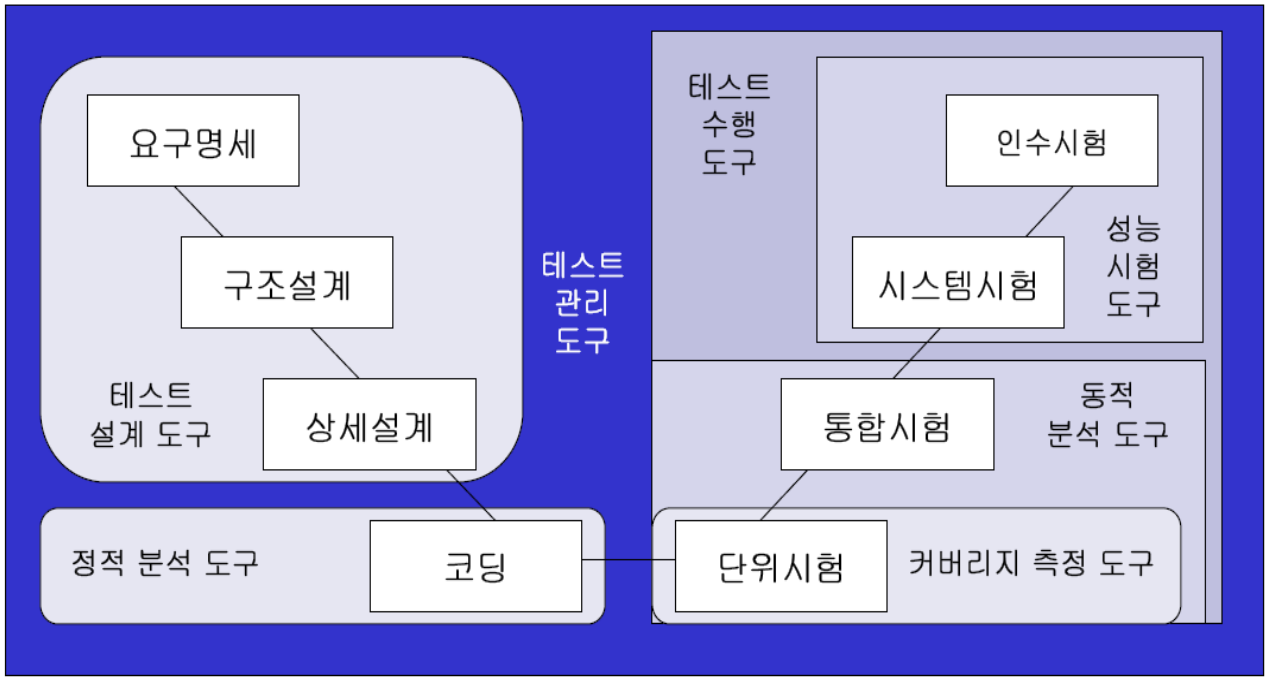
1) 결함관리 방안

단위 테스트 단계에서 결함관리는 테스트 수행 중 발생하는 결함을 등록하고 이에 대한 분석 및 오류 수정을 통하여 프로그램의 품질을 높이고자 하며, 결함관리 절차별로 파악되는 정보의 관리를 통해 결함해결의 진척관리와 사전 예방 활동을 강화하고자 함

단계정의	주요 수행 TASK	담당자
결함발견 및 등록	• 각 테스트 영역별 발견된 결함에 대해 결함을 등록	테스트담당자
결함 등록 확인	• 등록된 결함을 PL, 개발자가 확인하고, 조치 예정일 및 조치자 재분배를 진행	PL
결함조치 및 완료 등록	• 조치예정일 이전까지 등록된 결함 조치를 완료하고 결함조치 사항을 등록 (결함 발생후 1주일 이내를 조치완료일로 정함)	개발자
결함 보완 여부 확인	• 조치 완료된 결함에 대해 재테스트 수행을 통해 완료 여부를 점검	PL
결함 진행 관리	• 등록된 결함의 현황과 조치 완료를 확인하고, 결함 보완을 독려함	PM

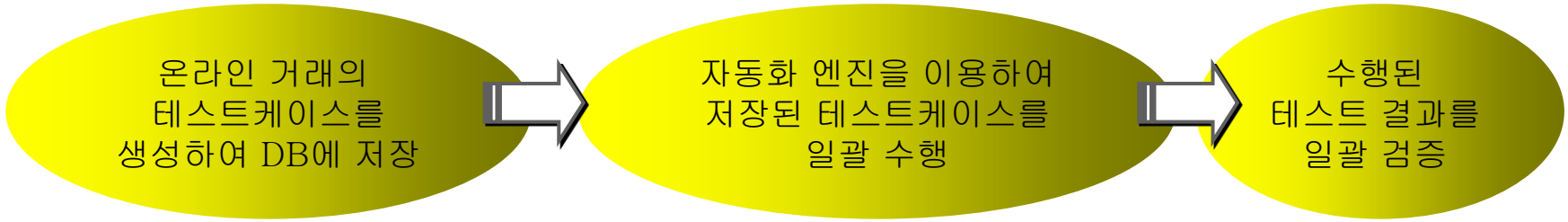
1) 테스트 자동화

테스트의 효율성과 효과성을 위한 테스트 자동화 설계영역에는 명세기반 테스트설계도구, 코드기반 테스트 설계 도구, 구현 영역에는 정적, 동적분석도구, 커버리지 측정도구가 있다



테스트 수행 도구
✓성능 시험 도구(시스템, 인수)
✓동적 분석 도구(통합)

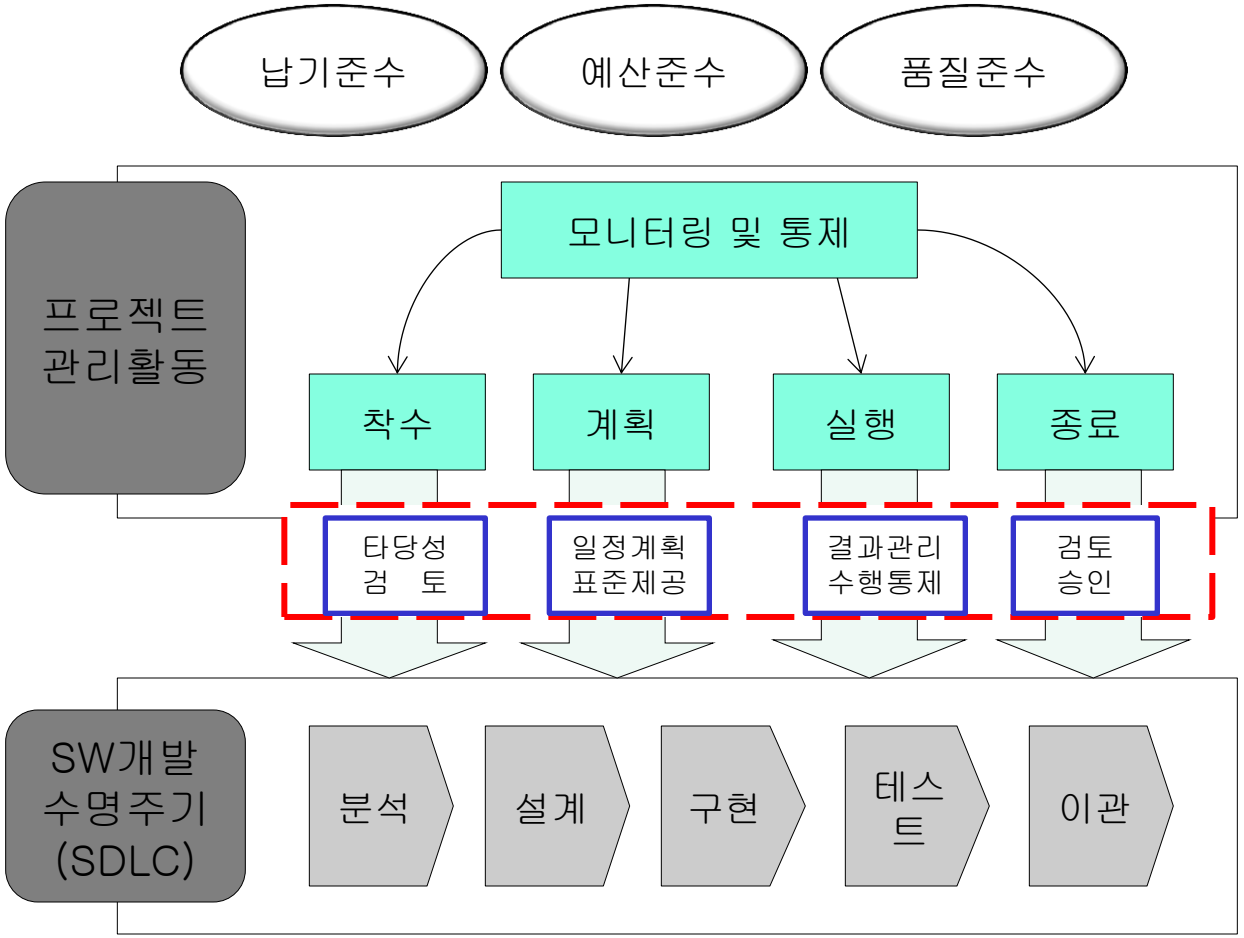
커버리지 측정 도구(단위)
✓테스트 관리 도구
✓테스트 설계 도구
✓정적 분석 도구(코딩)



Ⅵ. SW개발방법론 확대전략

1) 관리방법론 (Management Methodology) 개요

납기준수, 예산준수, 품질준수 등의 성공적인 SW개발을 수행하기 위해서는 최종산출물을 만들기 위한 활동들에 대한 관리가 필수적이며, 이에 대한 모델이 관리방법론 임



전체와 부분관리의 원리

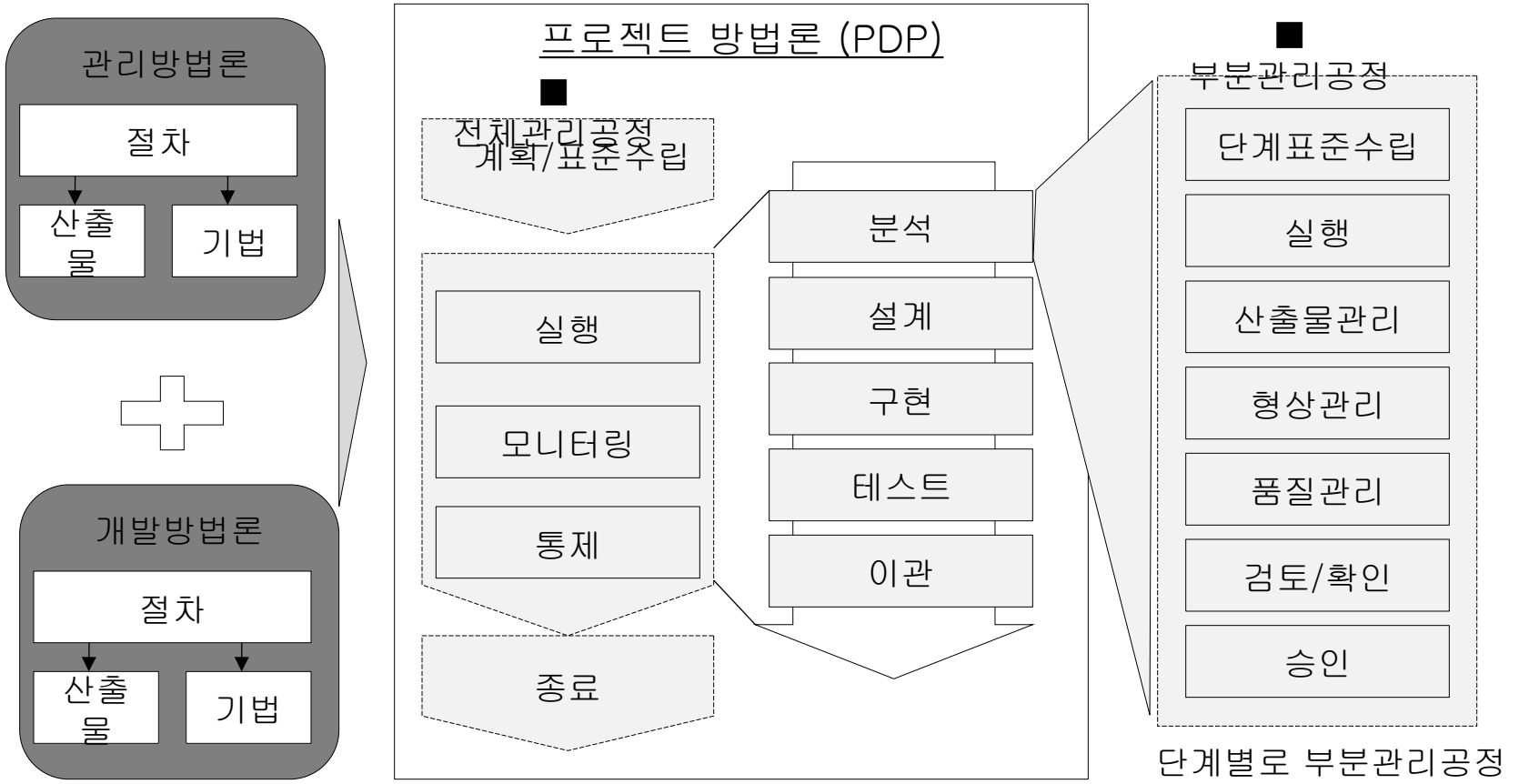
프로젝트 관리활동은 전체 개발수명주기에 적용하며, 각 단계별로도 별도의 관리활동 적용

관리방법론

프로젝트 관리 활동의 절차, 기법, 산출물 등 지식을 모델화한 것

2) 관리방법론과 개발방법론의 통합

SW개발 프로젝트에서 방법론을 적용하기 위해서는 개발방법론과 관리방법론을 통합하여 프로젝트에 최적화된 방법론을 수립하여야 함



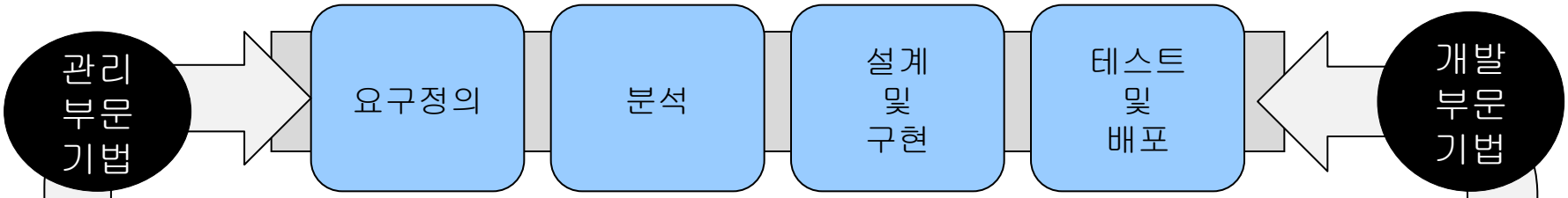
* PDP (Project Defined Process)

단계별로 부분관리공정 수행
(단계별 수행절차는 동일)

1) Agile Process 의 IT조직 적용 동향

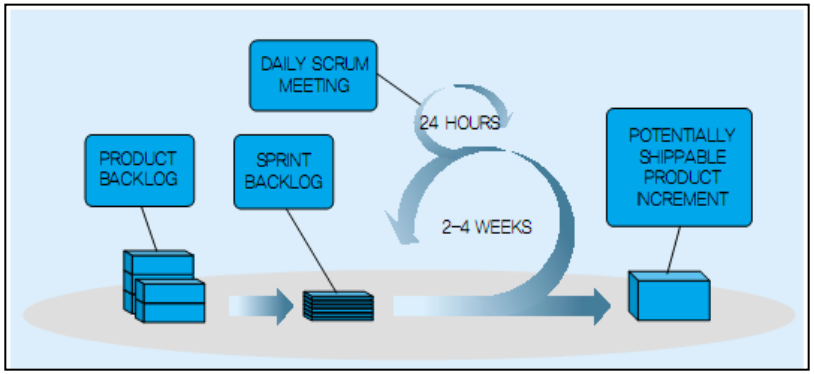
최근 Agile 방법론을 적용하는 조직들은 기존 개발방법론(정보공학, 객체지향 등)을 기반으로 하고, Agile 방법론의 장.단점을 고려하여 부분적인 관리 및 개발기법으로 적용하는 추세임

■ 기존 객체지향 방법론

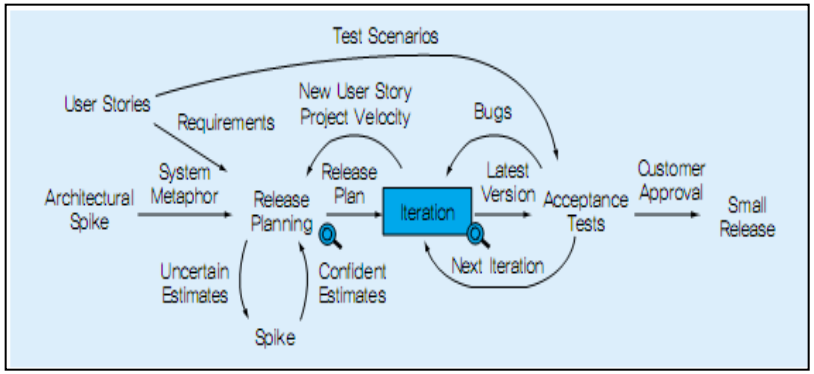


1. 기능/개선점 별 우선순위
2. 2~4주 개발주기 → 실제 동작하는 결과
3. 개발 주기 별 적용기능, 개선 목록 제공
4. 매일 SCRUM 회의 (진척관리 등)
5. 항상 팀 단위 고려
6. 열린 공간 유지 (원활한 의사소통)

1. 공통 코딩 가이드라인
2. 코드 리팩토링
3. 코드 회귀테스트 (단위 테스트 등)
4. 지속적 통합
5. 고객 참여
6. 테스트 주도 개발
7. 짝 프로그래밍



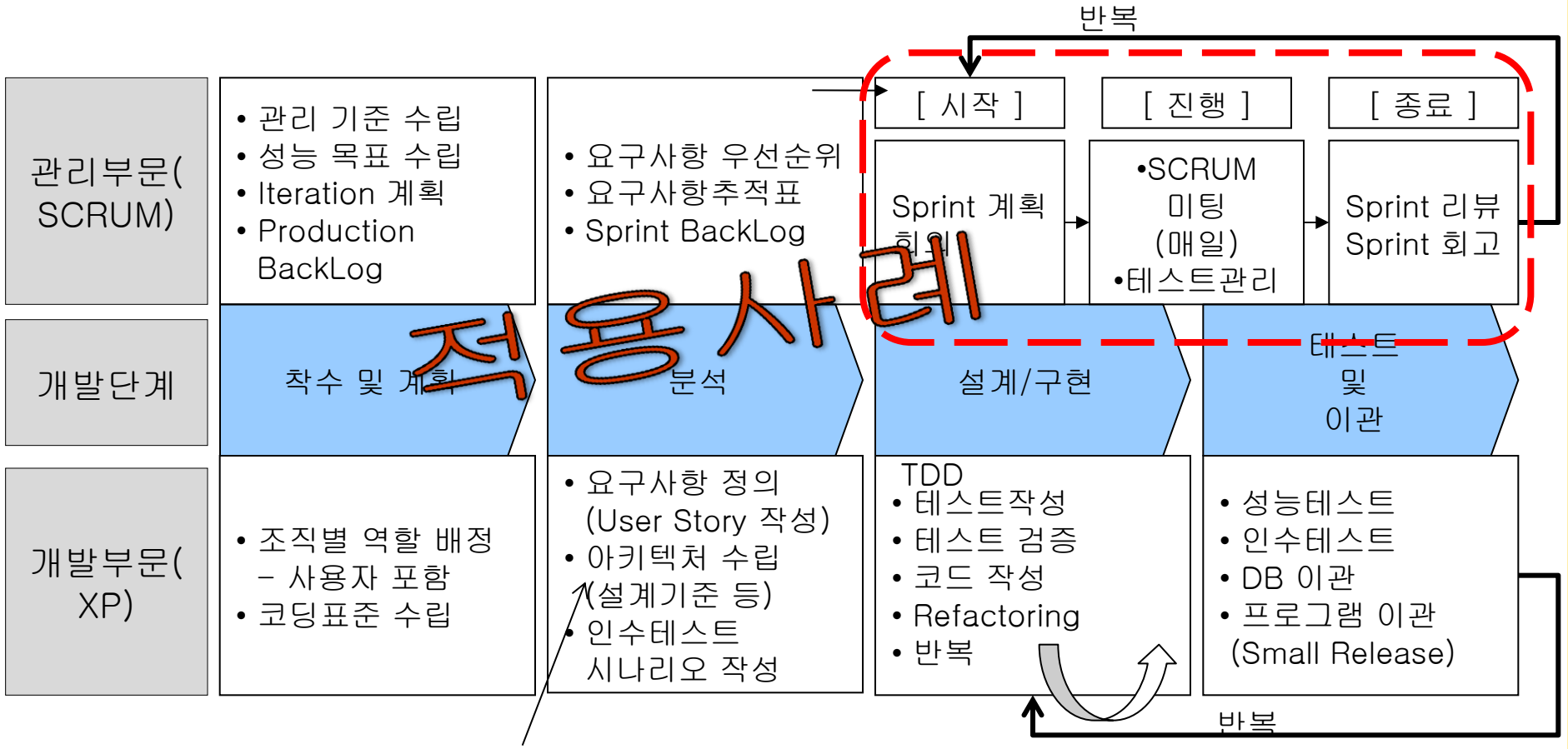
■ Agile Process – 스크럼(SCRUM) 방법론



■ Agile Process – XP (eXtreme Programming) 방법론

2) Agile Process 의 개발방법론 적용

전체 SDLC에 Agile 방법론(기법)을 포함하여 개발방법론을 표준화하는 경우, 기존 관리 활동에는 SCRUM의 기법을, 분석 및 설계, 테스팅에는 XP의 기법을 포함하여 각 타스크별 수행기법, 산출물을 조정하여 개발 방법론을 수립할 수 있음

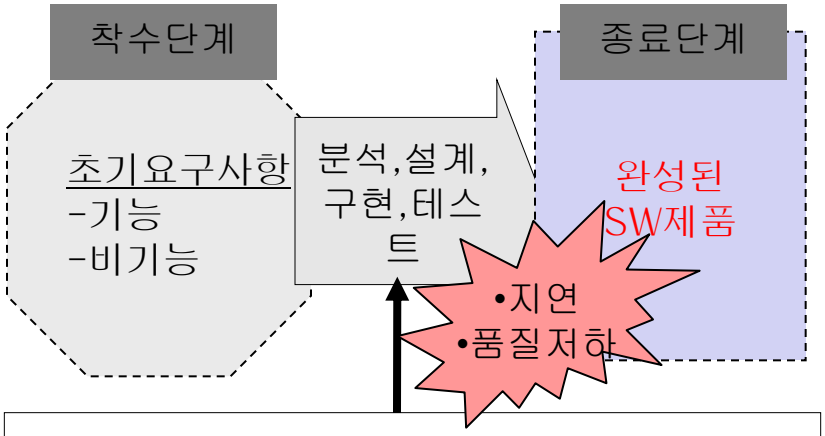


아키텍처를 수립한 이후 TDD를 적용한 Small Release 반복

1) 요구사항 감리의 개요

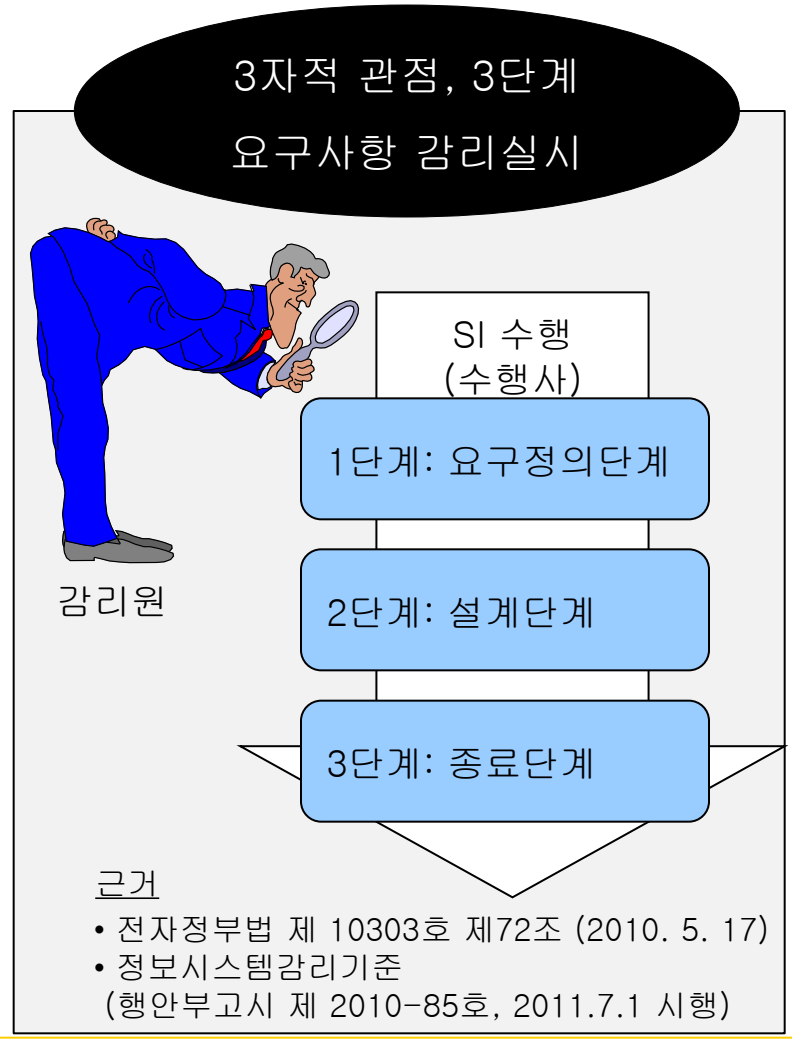
공공기관 프로젝트의 성공적인 SW개발을 보장하기 위해서 계약된 과업내용이 적정하게 개발에 반영되었는지를 제 3자적인 관점에서 점검하는 감리활동을 의미함

요구사항 불일치로 프로젝트실패 증가



- 불분명한 요구사항 (점진적 명확화)
- 잦은 요구사항 변경
- 관리되지 않는 요구사항
- 현업, 사용자의 무성의

➔ 프로젝트 성공을 위한 요구사항 관리 필요



2) 요구사항 감리위한 개발방법론 적용

공공 프로젝트의 성공적인 수행을 위해서는 요구사항 감리에 대한 준비가 필요하며, 이를 위해서는 조직의 개발 표준모델인 방법론에 대응을 위한 작업 절차, 기법(도구), 산출물의 적용이 필요함

■ 단계	■ 주요 감리 관점	■ 수행조직의 대응 활동
1단계 요구정의단계	• 과업내용의 요구사항 누락여부 (완전성) • 요구사항이 추적가능 여부 점검 (추적성) • 설계단계 검사기준서 작성가능 적정성 여부 점검 (검증가능성)	계약된 과업내용 반영여부 점검문서를 발주자 확인 거쳐 감리제출 • 과업내용서(과업지시서) • 사업수행계획서 • 회의록 • 과업 대비표 등
2단계 설계단계	• 설계산출물의 과업내용 반영 여부 점검 • 과업대비표가 세부 검사항목별로 적합 판정 가능 여부(구체성)	관리 측면 준비 • 계약서 • 과업 대비표 (요구사항 추적표) – 설계산출물과 매핑여부 체크 개발 측면 준비 • 설계단계 산출물
3단계 종료단계	• 감리대상 사업 검사 전까지 세부 검사항목별 과업내용 이행 여부 점검 • 과업범위 및 요구사항의 설계반영 및 구체화 여부 검토.확인 • 관련 법령 등 규정 및 지침등의 준수 여부 • 감리기준에서 정한 사항	관리 측면 준비 • 계약서 • 과업 대비표 (요구사항 추적표) – 구현과 테스트 매핑여부 체크 개발 측면 준비 • 요구사항추적표 기반 개발산출물 • 테스트 결과물 – 품질목표 검증 포함

작업절차, 기법(도구), 산출물 표준화

적용

개발 및 관리 방법론

반드시 수행해야 하는 절차로서 방법론에 포함

예)
- 산출물 계획 수립시 과업대비표 매핑 활동
- 단계 완료 이전 해당단계 수행결과의 과업대비표 확인활동

※ 참고 : 정보화사업 감리점검 수행가이드 (2011.07)

3) 요구사항 감리위한 개발방법론 적용

■ 핵심 고려사항

1단계
요구정의단계
(분석 말)

- 요구사항의 크기는 기능점수 (FP, Function Point) 단위로 나눌 것을 권고
- 요구사항 작성 품질은 TTAK.KO-11.0103 소프트웨어 요구사항 품질 평가 항목을 만족하도록 작성할 것 을 권고
- 과업 대비표 : ① 식별자를 이용하여 과업내용을 추적할 수 있도록 작성, ② 과업대비 변경과 요구사항정의의 변경 일치토록 함

2단계
설계단계
(설계 말)

- 과업 대비표 및 검사기준서
  ① 구체적인 수준의 준비 ② 요구정의단계 이후에 변경된 내용의 반영
- 설계 산출물 : 과업내용을 적정하게 반영하였는지를 점검
  ① 방법론상에 정의된 산출물의 작성 여부 ➔ 결과 누락여부 확인
  ② 과업대비표와 검사기준서의 요구사항ID와 설계산출물 매핑

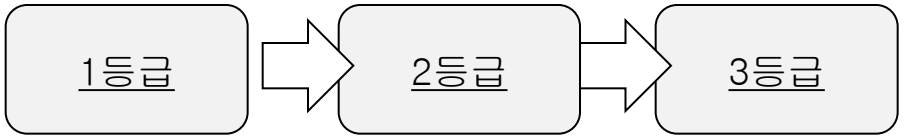
3단계
종료단계
(통합시험 이후
~ 검사이전)

- 과업 진척현황 : ① WBS의 결과 산출물을 작성 ② 과업대비표에 단위, 통합시험 테스트 실시 여부 확인 가능토록 매핑
- 산출물 : 과업내용 이행여부 점검을 할 수 있도록 ① 산출물 물량과 ② 점검방법(테스트 환경) 등을 제시할 수 있도록 준비

1) SW 프로세스 품질인증의 필요성

국내 SW중소기업의 개발역량 향상을 위해 SP(Software Process) 품질인증제도가 구축되었으며 공공 프로젝트의 제안평가요소에 포함되어 IT중소기업에서는 SP 품질인증 획득이 필요함

■ SP 품질인증 등급



초보단계 수행

프로젝트지원 프로세스

조직지원 프로세스 량강화

■ SP 품질인증 영역 과 평가항목

영역	평가항목
① 프로젝트관리	프로젝트 계획, 통제, 관리
② 개발관리	요구사항관리, 분석, 설계, 구현
③ 지원영역	품질보증, 형상관리, 측정 및 분석
④ 조직관리	조직프로세스관리, 기반구조관리, 구성원 교육
⑤프로세스개선	정량적 프로세스 관리, 문제해결, 프로세스 개선 관리

2등급

3등급

■ SP 품질인증의 근거

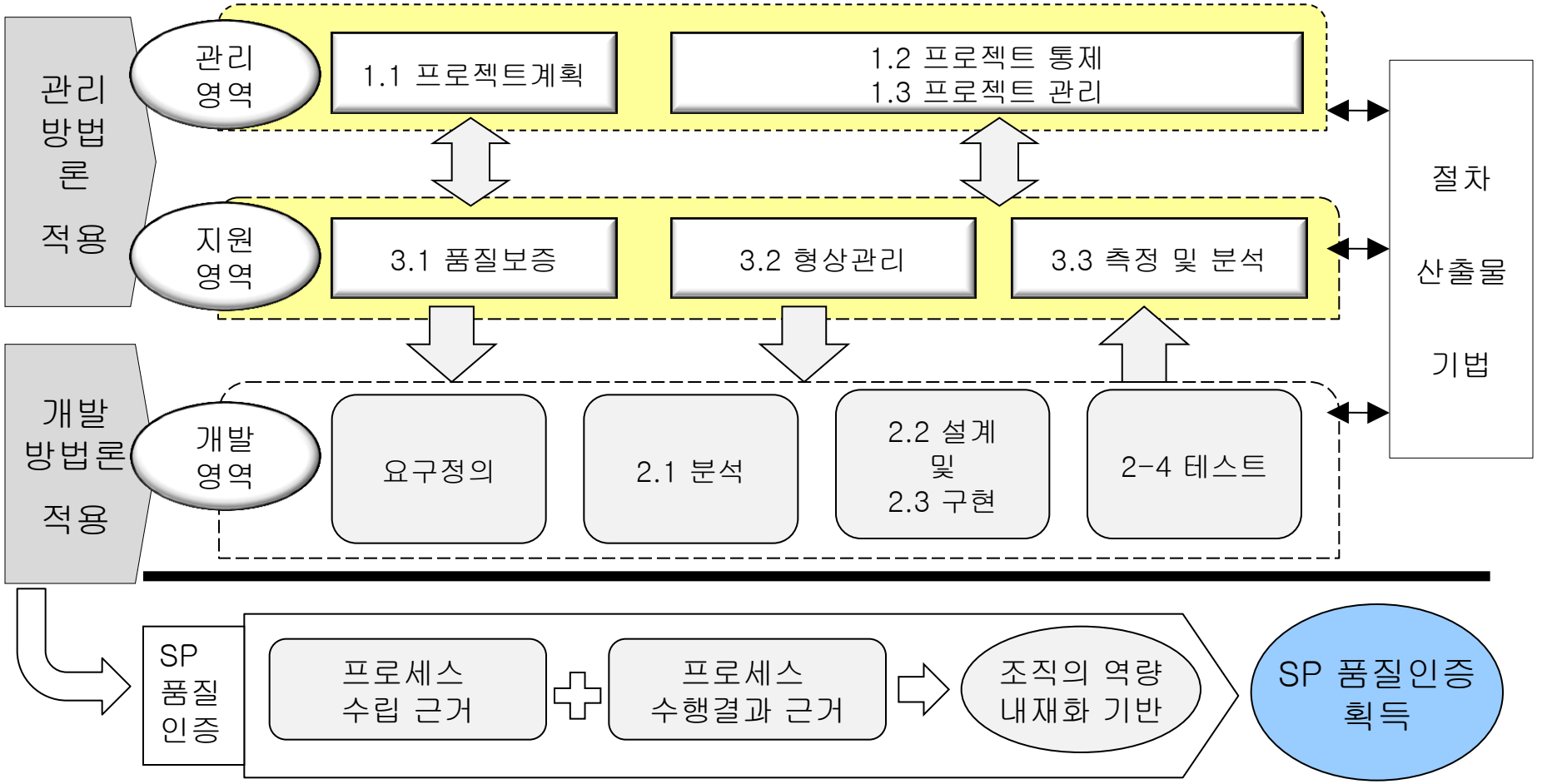
- 소프트웨어 산업진흥법 제 2조, 제33조 ('07.12)
- 소프트웨어 산업진흥법 시행령 제 16조의 2 내지 제 16조의 4 ('08.08)
- 소프트웨어 산업진흥법 시행규칙 제9조의 내지 11조 ('08.09)
- 소프트웨어 프로세스 품질인증 운영지침 ('08.19)

- 공공 프로젝트 제안평가 요소
  - 소프트웨어 기술성 평가기준
('11. 7. 22, 지식경제부 2011-148호)

2) SP품질인증을 위한 개발방법론 활용

SP인증의 프로세스 영역의 절차, 산출물, 기법을 관리방법론과 개발방법론에 포함하여 조직의 표준모델을 수립함으로써, SP품질인증의 근거와 내재화의 기준으로 활용할 수 있음

■ 개발방법론 적용



1) PIMS (Personal Information Management System) 개요

개인정보를 개인정보생명주기에 따라 보호하기 위한 관리과정, 보호대책, 보호기술을 수립하고, 지속적인 유지관리 요구가 급증하고 있음

■ 필요성

정보통신 서비스 급격한 증가 (웹 서비스, 행정 전산 정보 등)

→

침해/피해 급증 ➔파급효과 전사회적확산

←

개인정보 활용 의존도 높아짐 (개인화 서비스, 위치정보 CRM 등)

■ 구축 및 운영 요구

공공부문

개인정보보호에 관한 법률

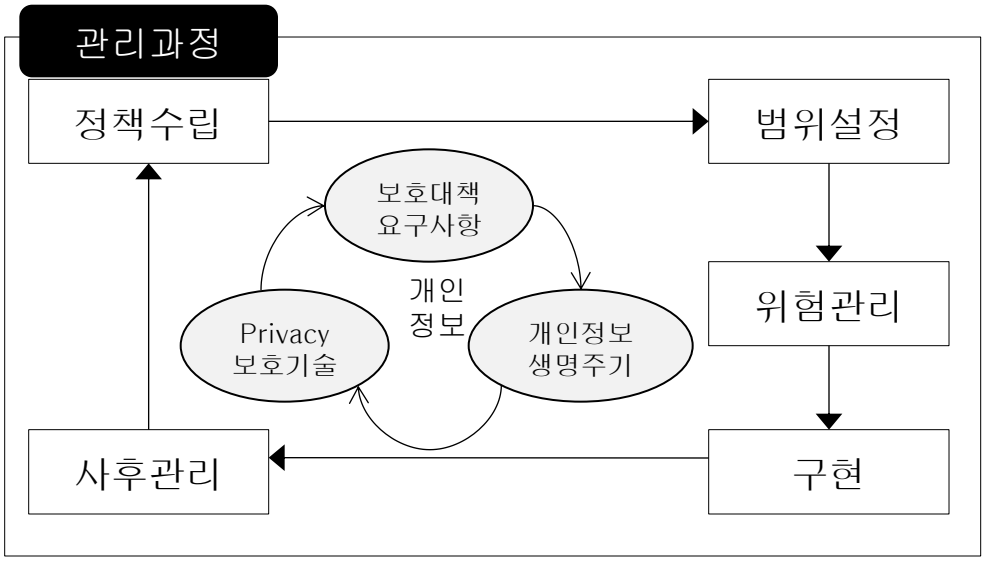
민간부문

정보통신망이용촉진 및 정보보호 등에 관한 법률

OECD

개인정보보호 8원칙

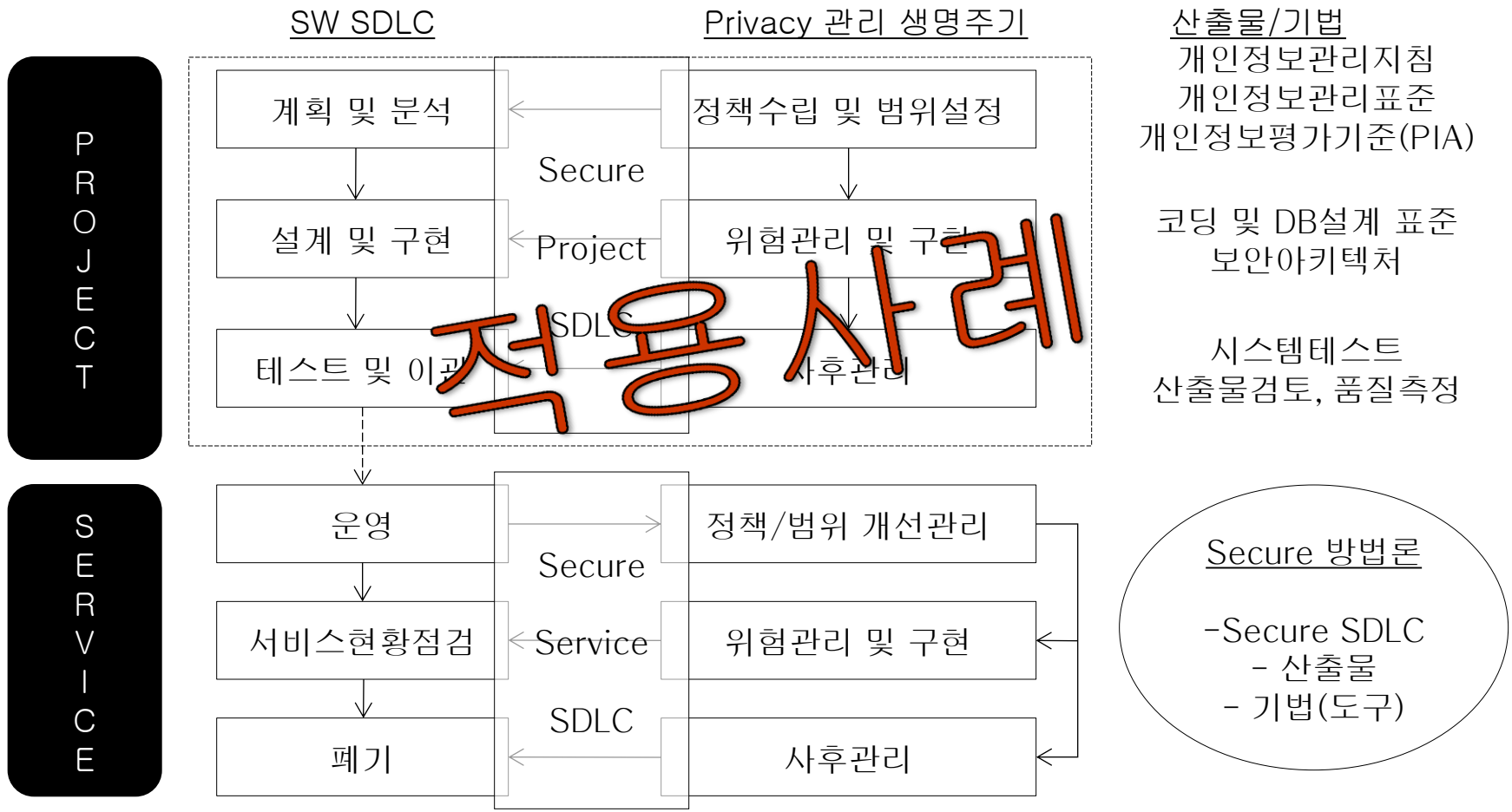
■ PIMS 체계 수립 4요소



1) PIMS (Personal Information Management System) 개요

2) 개발방법론과 보안(PIMS 구축)과의 연계적용방안

PIMS(개인정보보호체계)의 지속적인 유지관리를 위해서는 개발라이프사이클(SDLC)에 개인정보를 보호하기 위한 체계수립 단계와 각 단계별 요구사항을 추가 Secure SDLC를 수립하고 수행절차, 기법(도구), 산출물 등을 체계화하여 표준으로 적용



붙임. 참고서적