소프트웨어 개발 방법론

SOFTWARE ENGINEERING

3.6 익스트림 프로그래밍(XP)

- XP는 애자일 소프트웨어 개발 방법론 중 가장 많이 알려진 방법이다.
- XP 자체는 기존의 방법론에 비교해 볼 때 매우 가벼운 기법이며 실용성(pragmatism)을 강조한 것이라고 볼 수 있다.
- XP의 목표: '고객에게 최고의 가치를 가장 빨리'
- XP는 의사소통(communication), 단순함(simplicity), 피드백(feedback), 용기(courage), 존중(respect) 등 5가지의 가치에 기초
- XP는 개발 속도를 높이는 가속 기술이며, 그 중심은 단순한 디자인 정신, 테스트 우선 프로그래밍, 리팩토링이라 할 수 있다.

사용자 스토리(user story)

- **사용자 스토리**(user story)를 만들어 고객과 직접 대면하며 이야기한다.
- 사용자 스토리는 고객이 원하는 기능을 짧게 표현해 놓은 것
- 해당 기능에 대해 간략하게 설명하거나 기능을 대표하는 키워드를 포함하는 짧은 문장이 포함
- 사용자의 요구사항은 언제든지 변할 수 있으며, 사용자 조차도 자신의 요구사항을 정확히 알지 못하는 경우가 대부분이라는 것을 가정
- 결국 개발 초기에 요구사항을 구체적으로 정의하는 단계를 거치지 않고 사용자와 개발자가 지속적으로 대화하며 사용자가 원하는 요구사항을 이끌어 내는 방식을 사용한다.

사용자 스토리 예

❖ 인터넷 쇼핑몰 프로젝트의 사용자 스토리 예시

- 관리자는 카테고리를 새로 등록하거나 수정 또는 삭제한다.
- 회원은 카테고리를 선택하여 카테고리에 속한 상품의 목록을 조회한다.
- 회원은 상품을 장바구니에 담거나 이미 담긴 상품을 장바구니에서 삭제한다.

사용자 스토리 카드 예

스토리ID	M102	작성일자	2014-08-22
우선순위	상☑ 중☐하☐	추정	1주
담당개발자	홍길동		
스토리 쇼핑몰 회원은 카테고리를 선택하여 카테고리에 속한 상품의 목록을 조회한다.			
비고 하위카테고리가 존재하는 카테고리에는 상품이 포함되지 않는다. 최하위 카테고리를 선택한 경우에만 상품 목록이 조회되어야 한다.			

3.6.1 좋은 사용자 스토리

- 독립적이다 (Independent)
  - 사용자 고유의 비밀번호가 필요하다.
  - 사용자가 비밀번호를 잊었을 때 이를 찾을 수 있는 기능이 필요하다.
- 협상 가능하다 (Negotiable)
  - 사용자는 고유의 비밀번호를 가지며, 이를 잊었을 경우 찾을 수 있는 기능이 필요하다.
- 사용자와 고객에게 가치가 있다 (Valuable)
- 추정 가능하다 (Estimable)
- 작다 (Small)
- 테스트 가능해야 한다 (Testable)

유스케이스와의 차이점

- **유스케이스명**: 상품목록조회
- **개요**: 인터넷 쇼핑몰 사용자는 특정 카테고리를 선택하여 해당 카테고리의 상품목록을 조회한다. 상품의 목록은 사용자가 최하위 카테고리를 선택하는 경우에만 조회된다.
- **주 행위자**: 회원
- **전제**: 사용자는 시스템에 로그인하고, 상품조회 기능을 실행한다.
- **성공조건**: 선택된 최하위 카테고리에 포함된 상품들의 목록이 나타난다.

- **기본흐름**
  1. 최상위 카테고리 목록이 사용자에게 보여진다.
  2. 조회하고자 하는 카테고리를 선택한다.
  3. 선택된 카테고리의 하위 카테고리들을 보여준다.
  4. 최하위 카테고리를 선택할 때까지 2~3번 흐름을 반복한다.
  5. 선택된 최하위 카테고리에 포함된 상품들의 목록을 보여준다.

- **대안흐름**
  - 카테고리 선택 과정에서 상위 카테고리 목록으로 돌아가고자 하는 경우
    1. 카테고리 조회 중에 '위로' 버튼을 선택한다.
    2. 상위 카테고리 목록을 보여준다.

3.6.2 XP의 테스트

❖ XP 프로세스에서 이루어지는 테스트의 특징

1) 테스트와 관련된 활동은 프로젝트의 시작에서부터 요구사항 분석 단계까지 지속된다.
2) 테스트를 작성하는 작업은 요구사항을 밝히는 고객과 함께 협동하여 수행한다.
3) 테스터와 개발자는 적대적 관계가 아닌 협력 관계를 유지해야 한다.
4) 프로그램을 작게 나누어 테스트를 자주 수행한다.

테스트 케이스 예

❖ 요구사항을 포함하여 고객이 작성한 테스트 케이스 예

- 사용자가 선택한 카테고리가 하위 카테고리를 갖는 경우 하위 카테고리의 목록을 보여준다.
- 최하위 카테고리를 선택한 경우 카테고리에 속한 상품 목록을 보여준다.
- 임의의 시점에서 조회중인 카테고리의 상위 카테고리 목록을 조회할 수 있어야 한다.

❖ 테스트 케이스 작성 시점

(1) 고객과 개발자가 스토리에 대해 토론하는 과정에서 도출된 세부 사항을 기록하기 위해
(2) 스토리의 구현을 시작하기 전 개발자가 스토리를 명확하게 이해하고자 할 때
(3) 프로그래밍 중 또는 그 이후라도 스토리에 필요한 새로운 테스트를 발견할 때

3.6.3 사용자 스토리를 작업으로 분해

- 사용자스토리를 개발 작업으로 분해하고 구현에 필요한 노력과 자원을 추정
- 스토리 카드는 작업(업무)들로 분해되고, 각 작업은 구현의 기본 단위이다.
- 이는 개발자가 스토리를 구현하는 과정에서 해야 할 임무를 명확히 하기 위해 필요하다.
- 스토리를 작업으로 분류하는 또 다른 목적은 스토리를 구현하는데 필요한 일정 계획을 세우기 위함
- 개발자가 해야 할 일의 목록을 구체적으로 작성하고 나면 개발자는 각각의 작업에 소요되는 시간을 더 정확하게 추정할 수 있게 된다.
- 작업에 소요되는 시간의 합계는 스토리를 구현하는데 소요되는 시간

사용자 스토리와 테스트 케이스

- **스토리**
  - 회원은 카테고리를 선택하여 카테고리에 속한 상품의 목록을 조회한다.

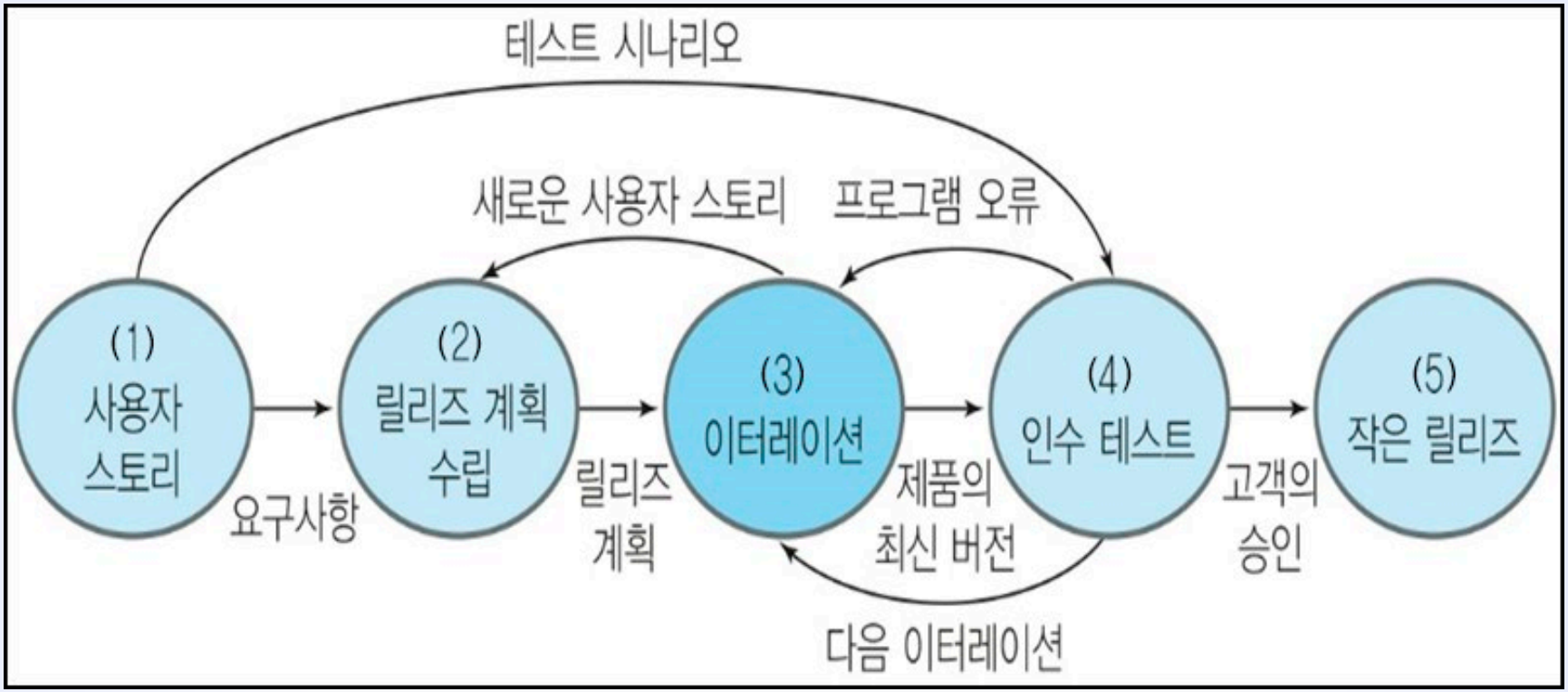
- **테스트**
  - 사용자가 선택한 카테고리가 하위 카테고리를 갖는 경우 하위 카테고리의 목록을 보여준다.
  - 최하위 카테고리를 선택한 경우 카테고리에 속한 상품 목록을 보여준다.
  - 임의의 시점에서 조회중인 카테고리의 상위 카테고리 목록을 조회할 수 있어야 한다.
  - 하위 카테고리가 존재하지 않는 최하위 카테고리를 선택하였으나 카테고리에 속한 상품이 존재하지 않는 경우 상품이 없음을 사용자에게 알린다.

요구되는 작업 정리

- **카테고리 목록 조회**
  - 최초 화면 조회 시에는 최상위 카테고리의 목록을 출력(상위카테고리명=null)
  - 사용자가 선택한 카테고리의 카테고리명을 입력 받아 하위 카테고리의 목록 출력
  - 상위 카테고리 목록을 조회하는 기능

- **상품 목록 조회**
  - 사용자가 선택한 카테고리의 하위 카테고리 목록을 조회한 결과가 null인 경우 최하위 카테고리로 판단
  - 최하위 카테고리가 선택되면 카테고리명을 입력 받아 카테고리에 속한 상품의 목록 출력
  - 최하위 카테고리에 속한 상품의 목록이 비어 있는 경우 상품 목록 출력 위치에 메시지 출력 “등록된 상품이 없습니다.”

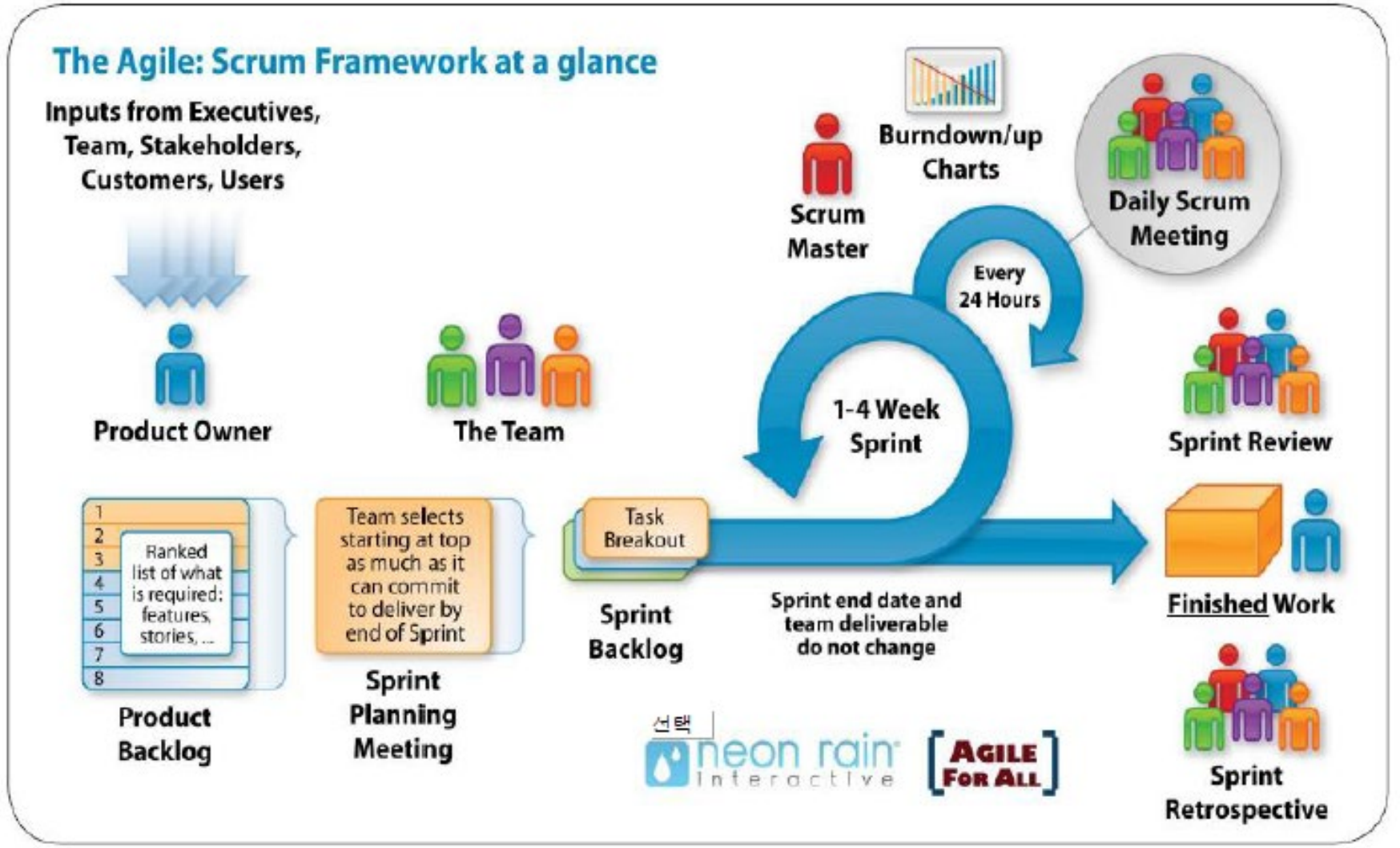
3.6.4 XP 개발 프로세스



3.6.5 XP의 가치

- 의사소통(communication)
- 단순함(simplicity)
- 피드백(feedback)
- 용기(courage)
- 존중(respect)

SCRUM의 프로세스


The Agile: Scrum Framework at a glance
Inputs from Executives, Team, Stakeholders, Customers, Users
Product Owner
The Team
Scrum Master
Burndown/up Charts
Daily Scrum Meeting
Every 24 Hours
1-4 Week Sprint
Sprint Review
Ranked list of what is required: features, stories, ...
Product Backlog
Team selects starting at top as much as it can commit to deliver by end of Sprint
Sprint Planning Meeting
Task Breakout
Sprint Backlog
Sprint end date and team deliverable do not change
Finished Work
Sprint Retrospective
neon rain Interactive
AGILE FOR ALL

SCRUM의 구성요소

구성요소	설명
Product Backlog	- 제품에 대한 기능 요구사항에 대한 우선순위 목록 - 프로젝트 기간 내 발생하는 변경 사항 목록 포함 - 시스템에서해결해야하거나, 시스템에포함되어야할 기능, 특성과기술에대한 모든기술나열 - 프로젝트가 진행되면서 진화되고 변경
Sprint Backlog	- Sprint 기간에 수행되어야 하는 작업 목록 - Product Backlog 내에서 선택하여 진행 - 해당Sprint 기간에수행되어야하는Task 목록으로 Sprint 기간동안 개발가능한기능의목록을ProductBacklog에서선택
Sprint	- 통상 4~6주(30일)정도의 Time Box 성격을 가진 개발 반복 주기 - 각 Sprint 단계 종료 시 새로운 기능이 추가된 실행 가능한 제품 인도
Daily Scrum Meeting	- 매일 15분 내외의 짧은 회의로 진척 사항 및 작업 계획 확인 -SCRUM Master는진척사항검토, 정상적종료를 방해하는 위험및작업 계획확인
Retrospective	- Scrum팀에서 운영중인 사항 리뷰 및 개선 미팅
Burndown Chart	- 한 Sprint에 대한 작업 완료 추이 차트(소멸차트)

SCRUM 구성원의 역할

구성원	역할
Product Owner	Product Backlog작성, 관리 및 우선순위 결정 고객,관리자,팀원들과 협업을 통해 목표 설정
Scrum Master	일정 추진 최종 결정권자로 팀원에게 업무, 권한부여 프로젝트 관리자로서 팀원을 코칭, 문제 상황 해결
Scrum Team	4~7명의 개발자로 구성되며,Product Backlog로부터 선택한 Sprint Backlog를 Sprint 기간 내 구현

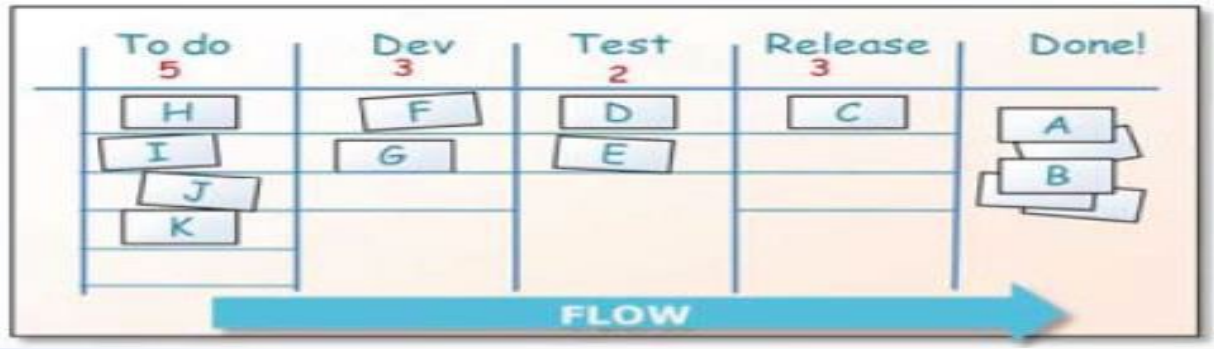
SCRUM 적용 현황 및 고려 사항

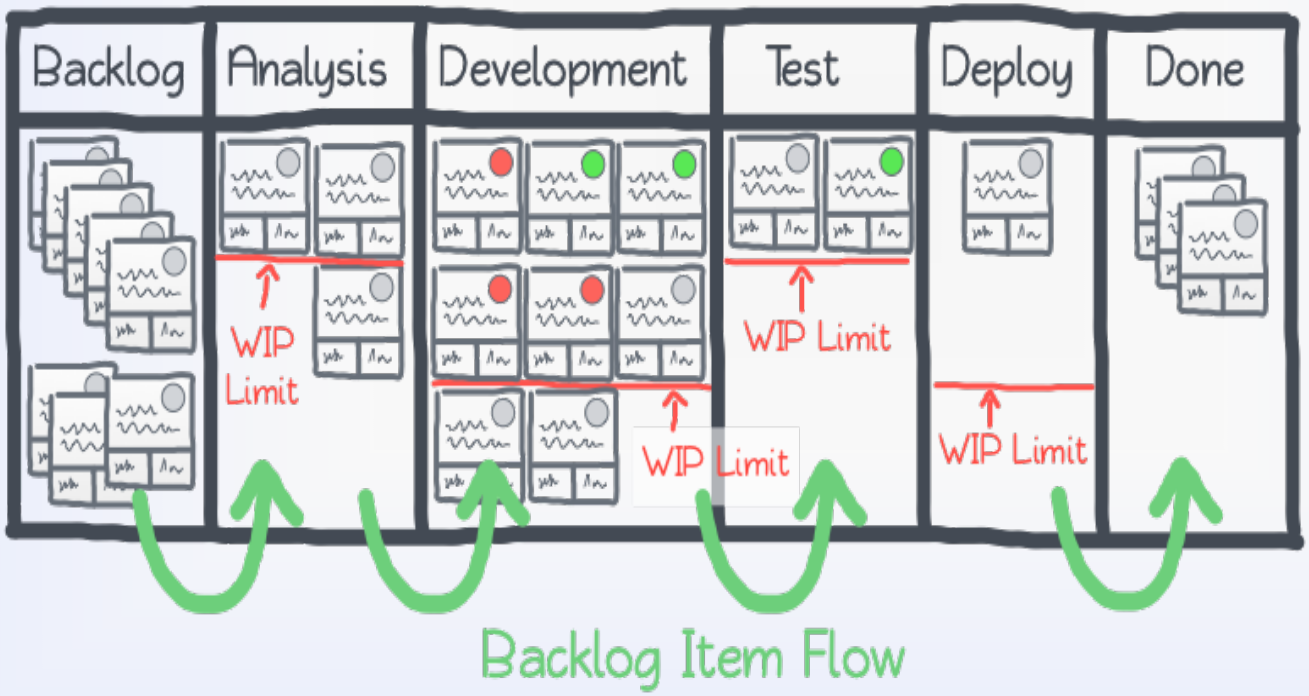
적용 현황	고려사항
- 금융, 보험 등 보수적인 산업에 있는 기업과 Google, Yahoo, MS 등도 SCRUM 을 선택 - Agile 방법론을 실행하는 팀 중 약50 %가 Scrum을 사용하고, 20%가 Scrum을 XP 구성 요소와 함께 사용하고, 12%가 XP만을 사용	- 전체 프로젝트에 대한 큰Mile Stone은 기존 개발방법론을 사용하고, SCRUM 방법론은 각 단계에 대한 Task를 위해서 사용 - 어떤 전문화된 시스템이나 도구가 아니라, 팀이 최대한의 효율을 낼 수 있는 프로세스가 중요

KANBAN의 개요

- **Kanban의 정의**
  - Workflow를 표현하는 **Kanban보드를 통해, 개발공정을 시각화** 하고, 작업제한, 소요시간 최적화 기법을 통해, 적시개발(JIT - just in time Development) 를 지원하는 **Agile방법론**

Kanban의 개념도

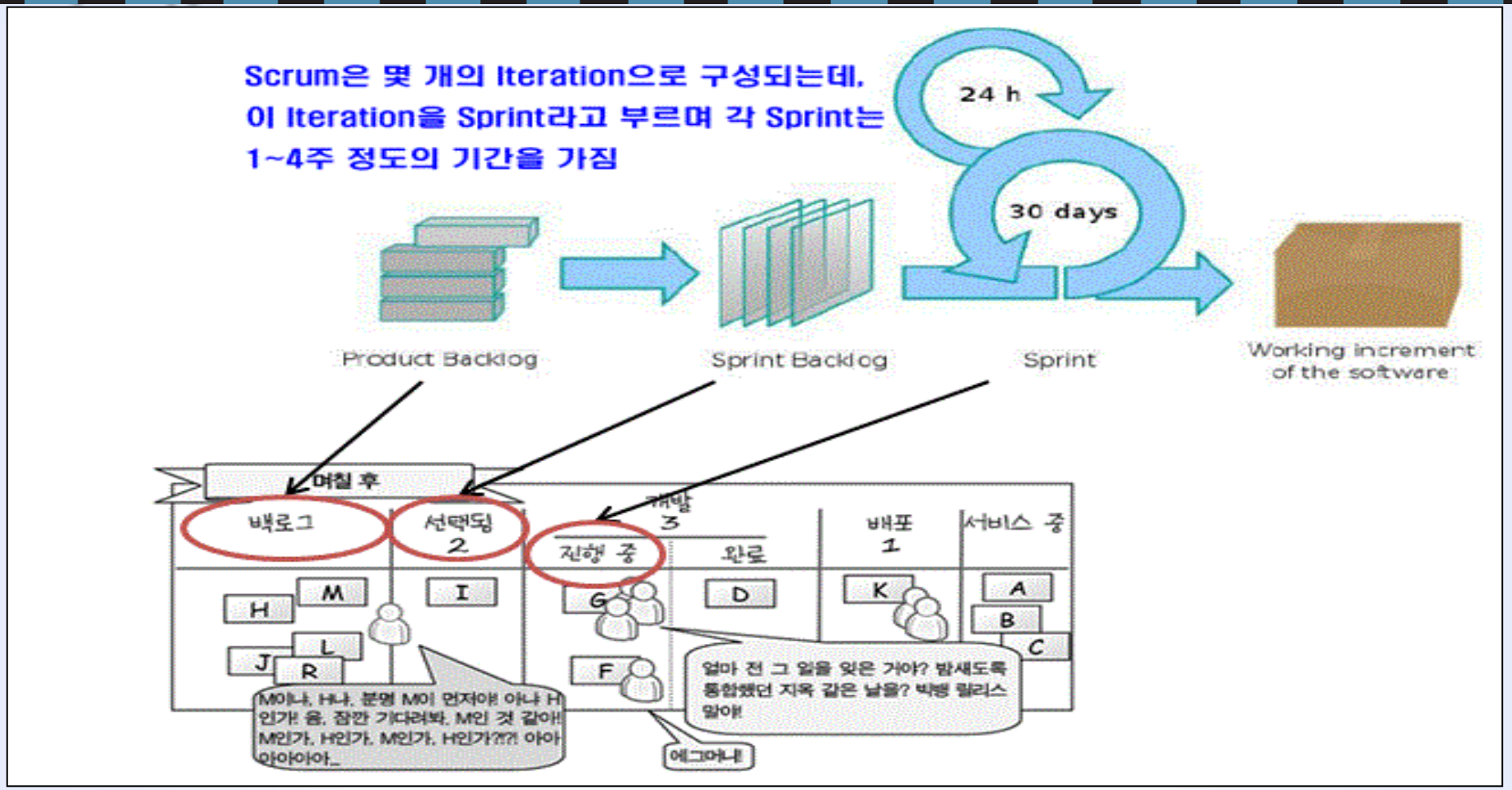

To do
5
Dev
3
Test
2
Release
3
Done!
H
I
J
K
F
G
D
E
C
A
B
FLOW


Backlog
Analysis
Development
Test
Deploy
Done
WIP Limit
WIP Limit
WIP Limit
WIP Limit
Backlog Item Flow

KANBAN의 구성요소

구성요소	상세내용	산출물
Kanban Board	프로세스를 기재한 Board와 스토리카드를 이용해 업무흐름 제어	스토리카드
Process	실제 업무가 이루어지는 단계 및 업무 수행을 통한 산출물 작성	업무성과
Work Queue (대기행렬)	대기행렬, 개발 대기, 테스트 대기, 배포/릴리즈 대기과정	Work Queue List
총 주기 시간 (total cycle time)	총 작업의 수행시간. 개별업무의 Cycle Time의 합으로 구성	Total Cycle Time

Kanban과 SCRUM 관계



- SCRUM가 Kanban 모두 테스트 주도(Test-drive) 개발 혹은 지속적 통합(CI)과 같은 애자일이 고려된 기법을 사용할 수 있음
- SCRUM 의 워크플로우가 KANBAN 과 융합함으로써 워크플로우의 가시화가 가능함

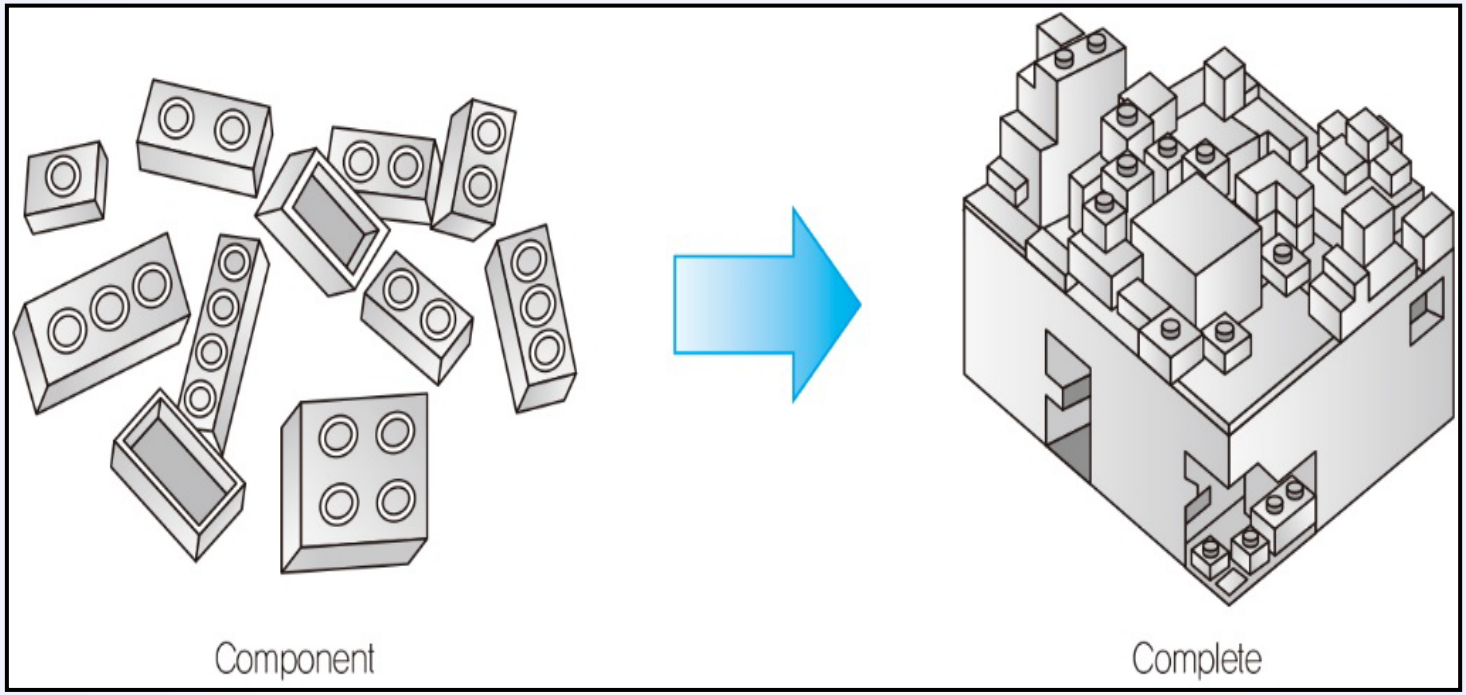
3.7 컴포넌트 기반 (CBD) 개발 방법론

- 소프트웨어 개발도 부품을 사다가 조립(plug-in)하여 만들 수 있지 않을까?
- 부품 조립 방법을 택하면 좋은 품질의 소프트웨어를 빠른 시간 안에 만들 수 있지 않을까?
- 소프트웨어의 경우 이런 재사용 가능한 부품을 컴포넌트(component)라고 부른다.
- 컴포넌트는 특정한 기능을 수행하기 위해 독립적으로 개발되고, 잘 정의된 인터페이스를 가지며, 다른 부품과 조립되어 응용 시스템을 구축하기 위해 사용되는 소프트웨어 부품(단위)
- 재사용 가능한 컴포넌트를 기반으로 소프트웨어를 개발하는 방법론이 CBD(component-Based Development) 방법론이다.

컴포넌트

- 컴포넌트는 하나 이상의 프로그램들을 하나의 단위로 관리하는 패키지
- 컴포넌트는 독립된 배포 단위이다.
- 컴포넌트가 다른 프로그램 또는 다른 컴포넌트와 상호 작동할 수 있는 유일한 방법은 잘 정의된 인터페이스이다.
- 컴포넌트는 세부적인 내부의 구현 사항(예: 구현 언어, 알고리즘 등)들을 외부로부터 감추고 제공되는 인터페이스를 통해 외부와 소통할 수 있도록 만들어져 있다.
- 다른 컴포넌트와의 조립을 위해 컴포넌트는 다른 컴포넌트의 인터페이스와 연결되어야 한다.
- 컴포넌트는 실제 구동될 수 있도록 만들어진 단위이며, 동적으로 바인드할 수 있도록 실행시간(run-time)에 인터페이스를 통해 접근이 가능하다.
- 컴포넌트는 일반적으로 잘 정의된 아키텍처 상에서 특정한 기능을 수행하며 독립적이면서 대체 가능한 시스템의 부분을 의미한다.

컴포넌트 조립



컴포넌트의 장점

- 복잡한 소프트웨어 시스템을 보다 쉽게 관리할 수 있다.
- 교체하기 쉽고 재사용하기 쉬워 개발 기간과 비용을 줄일 수 있다.
- 셋째, 기존의 검증된 컴포넌트를 사용하여 높은 품질의 소프트웨어를 만들 수 있다.
- CBD 방법론을 적용하면 컴포넌트 단위의 재사용을 가능하게 하여 객체지향 개발 기법의 구현코드(소스코드) 레벨 재사용에 대한 단점을 보완할 수 있다.
- 컴포넌트 단위의 재사용이란 컴포넌트 자체가 실행 가능한 모듈이 되어 구현코드에 대한 별도의 해석이나 컴파일 과정이 필요 없다는 것을 의미한다.
- 컴포넌트 기반 개발 방법론은 부품 조립식 소프트웨어 개발을 지원한다.

CBD 과정

- 컴포넌트를 만드는 컴포넌트 개발단계(CD: Component Development)

- 이미 개발된 컴포넌트를 사용하여 새로운 소프트웨어를 만드는 컴포넌트 기반 소프트웨어 개발단계(CBSD: Component Based Software Development)

3.7.1 컴포넌트 개발(CD)

- 컴포넌트 개발단계에서는 완전한 소프트웨어 시스템을 만드는 것이 아니라 해당 도메인에 대한 분석의 결과 재사용 가능한 부품을 만드는 것이다.
- 도메인 영역에서 재사용이 가능한 기능적인 요구사항이 무엇인지를 명확하게 정의하는 것이 필요
- 이를 바탕으로 기능적인 요소들을 담당하는 컴포넌트를 추출해 내는 작업이 따르게 된다.
- 컴포넌트의 제작이 이루어지면 이를 저장하고 관리하기 위한 컴포넌트 저장소가 필요하게 된다.
- 컴포넌트 저장소는 단순히 파일시스템이나 데이터베이스와 같이 컴포넌트 자체를 저장하기 위한 공간이 아니라 제작된 컴포넌트들을 분류하고, 그들 간의 관계에 대한 정보까지 제공해줄 수 있어야 한다.
- 뿐만 아니라 특정 컴포넌트에 대한 변경 이력이 발생할 경우 그에 따른 변경 관리도 수행할 수 있어야 한다.

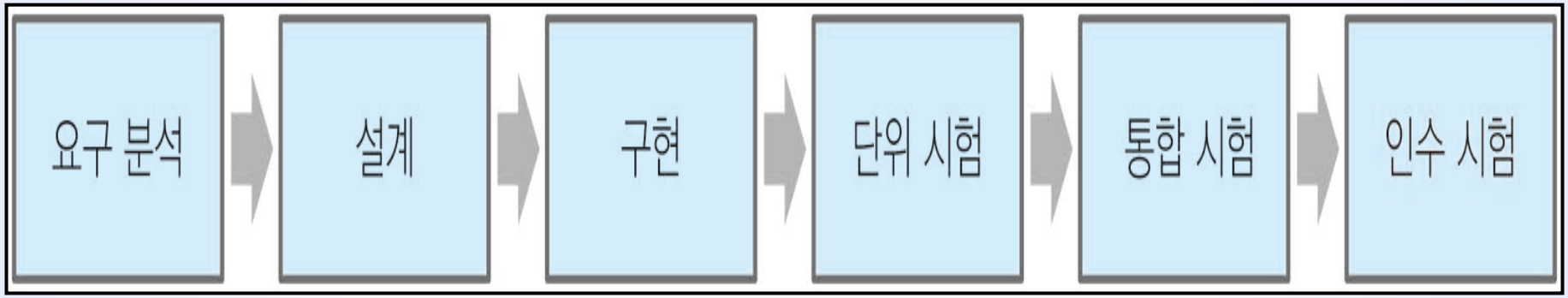
3.7.2 컴포넌트 기반 소프트웨어 개발(CBSD)

- 컴포넌트 기반 소프트웨어 개발은 이미 개발된 컴포넌트를 조립하여 소프트웨어 시스템을 개발하는 과정

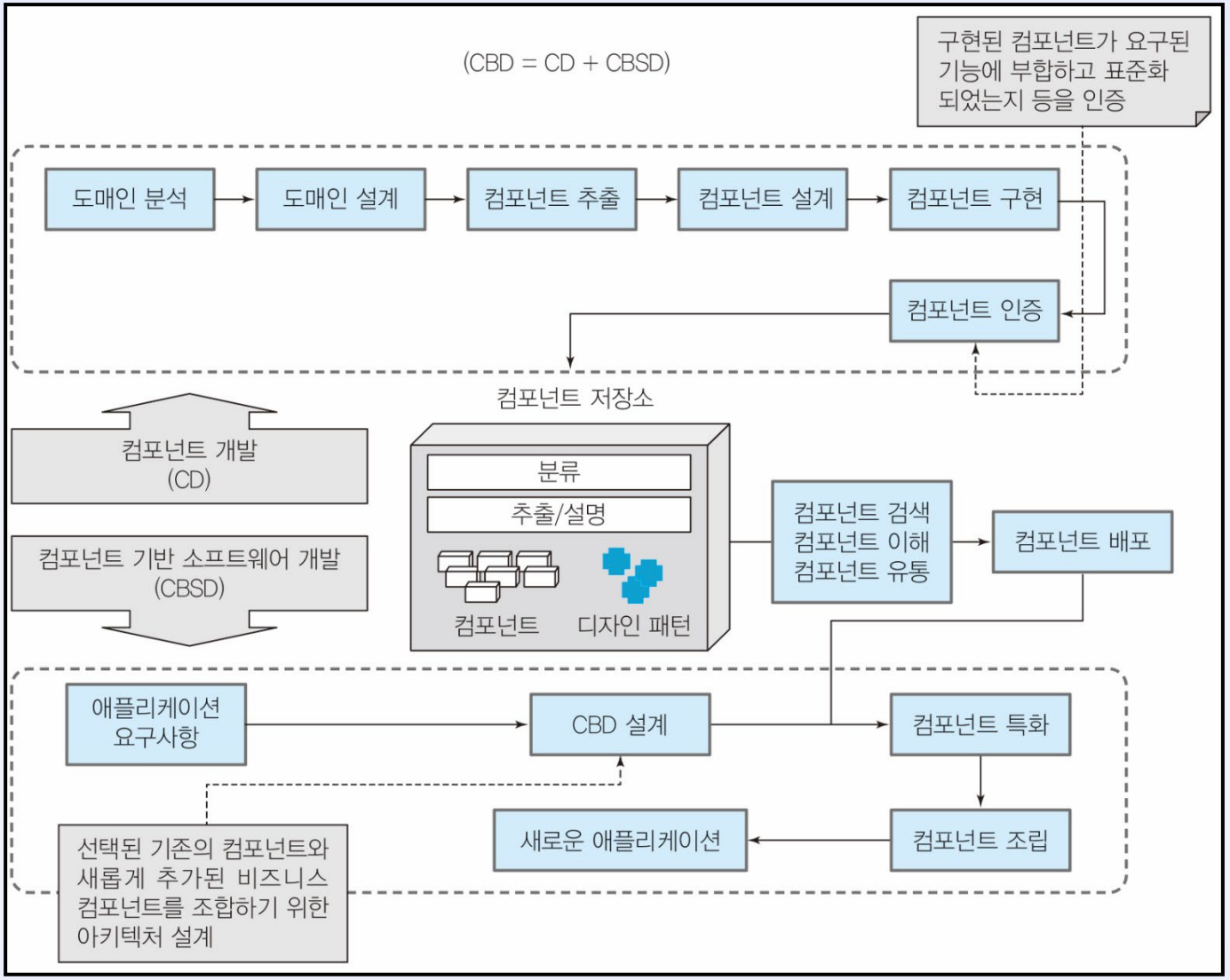
- 이 과정에서는 개발하고자 하는 애플리케이션의 요구사항을 정의하고 그에 따른 적절한 아키텍처의 설계가 이루어져야 한다.

- 아키텍처의 설계가 이루어지면 그 위에 조립하고자 하는 컴포넌트들을 획득해야 하는데, 기존에 개발된 컴포넌트들을 그대로 활용할 수도 있고 기능의 추가나 변경이 필요한 경우는 요구에 따라 커스터마이징한 후 사용할 수도 있다.

전통적 개발 프로세스



컴포넌트 기반 개발 방법


(CBD = CD + CBSD)
구현된 컴포넌트가 요구된 기능에 부합하고 표준화 되었는지 등을 인증
도매인 분석
도매인 설계
컴포넌트 추출
컴포넌트 설계
컴포넌트 구현
컴포넌트 인증
컴포넌트 저장소
분류
추출/설명
컴포넌트
디자인 패턴
컴포넌트 개발 (CD)
컴포넌트 기반 소프트웨어 개발 (CBSD)
컴포넌트 검색
컴포넌트 이해
컴포넌트 유통
컴포넌트 배포
애플리케이션 요구사항
CBD 설계
컴포넌트 특화
새로운 애플리케이션
컴포넌트 조립
선택된 기존의 컴포넌트와 새롭게 추가된 비즈니스 컴포넌트를 조합하기 위한 아키텍처 설계

CBD

- CBD란 느슨한 결합도(loosely-coupled)와 큰 입자(coarse-grained)의 특징을 갖는 컴포넌트를 기반으로 소프트웨어 시스템을 개발함으로써
- 고객의 요구변화에 신속하고 유연하게 대처하고자 하는 것을 목표로 하는 방법론이다.
- 또한, CBD 방법론을 기반으로 하는 소프트웨어 개발은 각 프로세스마다 특정 산출물을 가지게 되며 이 산출물들을 통해 중복투자 감소 및 유지보수성 향상을 달성한다.
- CBD의 주 이점인 재사용성에 대한 연구는 1980년대 말 객체지향방법론을 기반으로 계속 발전해 왔다.

CBD의 미래

- CBD는 소프트웨어 개발 패러다임 진화의 최첨단에 위치하며 높은 품질의 소프트웨어를 신속하고 효과적으로 개발할 수 있는 방법
- 결론적으로, 컴포넌트 기반 소프트웨어 개발이란 독립적인 기능을 담당하는 다양한 컴포넌트 소프트웨어의 집합으로부터, 해당 업무의 수행에 필요한 기능을 담당하는 하나 이상의 컴포넌트를 결합하여 해당 업무를 위한 소프트웨어를 개발하는 기술을 말한다.
- 이러한 컴포넌트 기반 개발은 과거 구조적 방법이나 객체지향 기술이 제대로 해결하지 못한 개발 생산성, 소프트웨어 재사용성, 시스템 유지보수성을 향상시킬 수 있는 대안으로 주목을 받고 있다.
- 이외에도 요구사항 획득 및 다른 소프트웨어의 생산, 납기 지연, 비용 초과 등 소프트웨어 위기를 초래한 고질적인 문제들을 해결할 수 있는 방안으로 인식되고 있다.

SCRUM과 CBD 비교

비교항목	SCRUM	CBD
정의	프로토타입을 반복 점진적으로 개발하는 Agile 방법론	기 개발된 모듈을 조립하여 새로운 시스템 구축 방법론
특징	반복,점진적 프로토타입개발	컴포넌트(모듈)검색 및 조립
확장방법	Sprint backlog iteration 기법	모듈간호출 인터페이스 사용
장점	빠른 개발 주기 프로젝트초반 요구사항 명확	재사용성 극대화 장기적 관점 개발비 감소
단점	대형 프로젝트 적용 어려움 산출물 중시 사업에 부적합	공용 모듈 개발,유통에 한계 개발 전문업체가 부족

3.8 소프트웨어 제작 방법의 공통점

- 시스템 제작의 공통점은 다음의 3가지로 요약되어 설명될 수 있다.

시스템의 **정의(definition)** 단계
시스템의 **개발(development)** 단계
시스템의 **유지보수(maintenance)** 단계

- 소프트웨어 개발에서도 앞의 어느 패러다임을 선택하든 이 세 가지는 중요한 의미를 갖는다.

시스템의 정의 과정

- 앞의 요구사항 분석 과정에 해당
- 사용자의 관점에서 시스템이 제공해야 하는 기능, 데이터, 인터페이스 정의
- 사용자에게 **무엇**(what)을 제공할 것인가에 초점을 맞춘다.
- 시스템 정의 과정: **사용자 관점**, 시스템의 **논리적**(logical) **관점**

시스템 개발 과정

- 시스템이 제공해야 하는 무엇(what)을 어떻게(how to) 만족시킬 수 있을 것인가 규명
- 개발 과정: **엔지니어**의 관점, 시스템의 **물리적**(physical) 관점
- 시스템 개발 과정은 **설계, 구현, 시험**의 과정
- 개발자는 요구사항을 만족시키기 위해 소프트웨어를 어떻게 설계할지, 어떤 프로그래밍 언어를 사용하는 것이 좋을지, 시험은 어떻게 할지 등에 관심을 가짐

시스템 유지보수 과정

- 시스템이 개발된 후 오류의 수정, 환경 변화, 기능 향상 요구 등과 연관되어 발생하는 **변화**(change)에 초점을 맞춘다.
- 수정적 유지보수, 적응적 유지보수, 완벽적 유지보수, 예방적 유지보수
- 시스템 변경의 경우에 따라 재 요구사항 분석, 재 설계, 재 프로그래밍, 재 시험의 과정이 필요하게 되고 이에 따라 관련된 문서의 갱신을 수반한다.

소프트웨어 개발의 무질서 극복

- 무질서는 사용자 관점(what)과 엔지니어 관점(how to)을 구별하지 못하는 데에서 출발
- 무질서는 소프트웨어의 품질과 유지보수, 문서 관리에 치명적인 영향을 준다.
- 엔지니어의 관점보다 **사용자의 관점**에 우선 순위를 두어야 한다.
- 시스템 개발 초기 정의 과정에서 충분한 분석이 이루어지고, 구체적인 목표가 확립되고, 사용자의 동의를 끌어내어 사용자들이 원하는 좋은 제품을 만들 수 있는 기반을 마련하여야 한다.
- **구체적인 목표의 확립**은 기술력과 품질의 향상은 물론 사용자 만족도를 증진시킬 수 있는 첫째 조건