

2022학년도 2학기-응용SW실무

# 3주차 : 프로젝트 계획하기

2022. 09. 19 (월)

Prepared by DaeKyeong Kim

Ph.D.

Dept. of CSE  
University of YUHAN



1

1교시: 애자일 프로젝트 지속적인 프로세스 개선

2

2교시: 시각적 관리 활용

3

3교시 : 유스케이스 다이어그램과 개념분석

프로젝트 계획하기

1교시 :

애자일 프로젝트  
지속적인 프로세스 개선





# 1-1. Review meetings

## 학습목표

- 이 워크샵에서는 애자일 Retrospective에 대해 알 수 있다.

## 눈높이 체크

- 애자일 Retrospective에 대해 알고 계신가요?



# 1-1. Review meetings

## 일일 스탠드 -업 회의

- 의의
  - 프로젝트 진행현황에 대한 이해
  - 지속적인 협업과 소통 강화
- 바람직한 스탠드 업
  - 상호 존중
  - 15분간 회의
- 전체 팀원이 모여 진척상황 공유
  - 어제 한 일
  - 오늘 할 일
  - 이슈



# 1-1. Review meetings

## 일일 스탠드 -업 회의 : 회의 준비

- 일일 스탠드 업 미팅은 모든 근무일 마다, 가급적이면 하루 업무 시작 시에 개최된다. 팀원 모두는 이에 참여 한다. 팀이 진행상황, 장애물, 다음 할 일을 보고할 기회이다.
- 팀 리더 준비
  - 리더는 회의를 중재하고 촉진하는 역할을 한다. 회의가 원활하게 진행되도록 다음을 준비한다.
    - 작업 현황판이 놓인 팀 내 공간
    - 타이머 또는 시계
    - 만일 회의에 참석할 수 없으면 적절한 사람 선정
- 팀원 준비 – 전문가들 포함
  - 기술적 및 팀 관점에서 공유해야 할 사항을 준비한다.
    - 한 일과 하려고 하는 일 공유
    - 장애물 공유



# 1-1. Review meetings

## 일일 스탠드 -업 회의 : 회의 시작

- 회의를 중재하고 있다면
  - 모두를 환영하라
  - 필요하다면, 모두에게 회의 권장사항과 엄격한 시간 엄수를 상기시키라



# 1-1. Review meetings

## 일일 스탠드 -업 회의 : 회의 진행

- 정시에 시작하기
- 정시에 마치기
- 15분으로 회의 마치기
- 한번에 한 사람 씩만 발언하기
- 주의 깊게 듣기
- 말하기 전에 생각하기
- 분명히, 간결하게, 긍정적으로 의견 말하기
- 토론에서 모든 아이디어들 수용하기
- 모두는 회의의 결정 존중하기
- 회의는 토론을 위한 안전한 장소일 것





# 1-1. Review meetings

## 일일 스탠드 -업 회의 : 회의 마치기

- 모두가 무엇을 해야할 지 이해했는가와 후속 조치(장애물의) 목록을 확실히 하고, 모두에게 감사하며 회의를 마쳐라.
- 회의 후
- 회의 후 새로운 방안들이 이터레이션 작업 현황판에 또는 제품 백로그를 위한 스토리 카드에 올려질 것이다. 팀 리더는 필요 시 장애물을 다른 팀에 보낸다.



# 1-1. Review meetings

## Sprint Review

- 목표
  - 모든 관심 있는 이해관계자들 앞에 완성된 결과 데모
- 방법
  - 참여자가 제품과 현 시점의 스프린트에 대해 이해
- 준비물
  - 완료된 실행 가능한 제품 일부
  - 스테이징 서버에서 데모 가능



# 1-2. Retrospective

## Sprint Retrospective

- 지난 스프린트에서 잘한 점과 잘못된 점을 공유
- 회고 미팅은 각 이터레이션 말에 한다. 팀에서 잘 진행된 점, 잘 진행되지 못한 점, 그리고 팀의 개선을 위한 아이디어에 대해 생각해 볼 기회이다.
  - 가치 있는 소프트웨어 전달
  - 지속 가능하게 작업
  - 팀원 서로와 사용자와 의사소통 잘 하기
  - 기술 향상시키기
  - 효과적으로 일하기
  - 개선전략 찾기
- 진행
  - 30~60분 정도
  - 모든 스프린트에서 수행



# 1-2. Retrospective

## 역할

- 회고의 참여자는 다음과 같다.
- 팀원, 전문가, 팀 리더 또는 (사용자를 대표하는) 제품 책임자.
- 잘된 것과 보다 잘될 수 있는 것에 대한 의견 제시
- 작업 프랙티스의 개선을 위한 아이디어 제시
- 다음 이터레이션에 이루어질 할 개선 사항에 대한 합의
- 팀 리더
- 미팅의 진행을 돕고 모든 사람에게 기회를 부여하고 사람들이 핵심에서 벗어나지 않게 하며 제한된 시간 내에 미팅 끝내기
- 합의된 개선 사항이 구현되도록 하기
- 역할자
- 퍼실리테이터: 회고의 진행이 매끄럽게 수행되도록 참가자들의 관심과 참여를 유도합니다.
- 참가자: 함께 프로젝트를 수행한 팀원들이 참여합니다.
- 적용상황
- 팀이 같은 실수를 반복한다고 느껴질 때
- 조직이 비슷한 상황을 반복해서 겪게 될 때
- 팀의 불만이나 제안이 리더에게 전달되지 않을 때

# 1-2. Retrospective

## 실천법





# 1-2. Retrospective

## 절차

- 사전 준비하기:
  - 회고에 참여할 사람들을 소집하고 회고할 수 있는 분위기를 만듭니다.
  - 경직되지 않은 열린 마음을 가질 수 있도록 합니다.
- 자료 모으기:
  - 기간 동안에 벌어졌던 다양한 자료들을 수집합니다. 시간이나 사건에 따라 모을 수 있습니다. 잘 했던 점, 아쉬웠던 점들을 나열합니다. 잘 했던 점을 더 잘하게, 혹은 미진했던 점을 극복할 수 있는 방법이 무엇인지를 모읍니다.
- 통찰 이끌어내기:
  - 이렇게 모여진 아이디어들을 바탕으로 투표를 하여 실제 다음 기간 동안 실행에 옮길 아이টে을 선정합니다.
- 무엇을 할지 결정하기:
  - 구체적이고, 측정할 수 있고, 달성 가능하며, 적절하고, 시기적절한(S.M.A.R.T) 목표를 세우도록 합니다.
- 회고 끝내기:
  - 회고에 참석한 사람들에게 감사를 표현합니다. 회고 자체를 더 잘하기 위한 방법을 나눕니다.



# 1-2. Retrospective

## 각 단계별 기법 예

단계	기법
체크인/회고 준비하기	<ul style="list-style-type: none"><li>① 이터레이션에 대한 감상 공유</li><li>② ESVP</li><li>③ 체크인</li></ul>
데이터 모으기	<ul style="list-style-type: none"><li>① 좋았던 점 / 나빴던 점 모으기</li><li>② 학습 매트릭스</li><li>③ 화남, 기쁨, 슬픔</li><li>④ 팀 만족도 그래프 그리기</li></ul>
통찰 이끌어내기	<ul style="list-style-type: none"><li>① 점 투표 하기</li></ul>
무엇을 할 지 결정하기	<ul style="list-style-type: none"><li>① SMART 한 회의</li><li>② EXACT 한 목표기준 세우기</li></ul>
회고 마무리하기	<ul style="list-style-type: none"><li>① 회고활동에 대한 회고</li><li>② 서로에게 감사하기</li><li>③ Follower 구하기</li></ul>



# 1-2. Retrospective

## 1. 체크인/회고 준비하기

- Time Box: 5분
- 수행 방법
  - 회의시작 전에 구성원들에게 특정 주제에 대해 짧게 돌아가며 이야기 합니다.
- 주제 예시
  - 지금 자신이 느끼고 있는 감정을 한 마디로 표현해 주세요
  - 지금 자신에게 가장 필요한 것을 한 마디로 표현해 주세요.
  - 최근 있었던 에피소드와 그로 인해 생긴 변화는 무엇이 있나요.
  - 요즘 자신이 가장 좋아하는 음식은 무엇인가요
  - 자신을 동물에 비유한다면 어떤 동물일까요.
- 주의할 점
  - 심리적인 장벽을 낮춰줄 뿐이지, 궁극적으로 모임이나 회의의 문제점을 해결해 주지 않습니다.
- 기대효과
  - 적극적으로 참여하지 않았던 사람에게 무조건 참여하는 기회를 줌으로써, 참여에 대한 두려움과 심리적 장벽을 없애 보다 회의에 적극적으로 참여하도록 합니다.
  - 모임이나 회의에서 보다 다양한 의견들이 나옵니다.





# 1-2. Retrospective

## 1. 체크인/회고 준비하기:이터레이션에 대한 감상 공유

- 사람들이 손쉽게 회고에 집중하기 위해, 분위기를 조성하는 활동으로, 지난 이터레이션 결과물(완료항목과 테스트 결과)에 대한 공유를 수행합니다.
- Time Box: 10분
- 수행 방법
  - 팀원 한 명씩 번갈아 가며, 이번 이터레이션에서 느꼈던 것을 짧게 이야기 하도록 합니다.
  - 내용은 자유롭게 말할 수 있도록 하되, 시간은 개인별로 1분 30초 이내로 제한합니다.
- 주의할 점
  - 내용 자체가 너무 부정적으로 흐르지 않도록 합니다.
  - 팀 내 인원에 대한 불만을 이야기 하지 않도록 합니다.
- 기대효과
  - 이터레이션을 수행하면서 느꼈던 감정과 데이터들을 정리할 수 있는 시간을 제공합니다.
  - 나 이외에 다른 팀원들이 지난 이터레이션 동안에 생각나는 내용을 공유합니다.



# 1-3. Retrospective 절차

## 1. 체크인/회고 준비하기: ESVP

- Time Box: 10분
- 수행 방법
  - 팀원 한 명 당 한 개의 포스트잇을 나누어 줍니다.
  - 참가자들은 익명으로 회고에 대한 자신의 태도를 ESVP 중 알파벳 하나로 적게 합니다.
    - E: Explorer (탐험가)  
새로운 아이디어나 직관을 쉽게 발견하는 사람. 가능한 모든 것을 배우고 싶어하는 사람
    - S: Shopper (쇼핑하는 사람)  
가능한 한 모든 정보를 둘러보고 그 중 유용한 아이디어를 취하고 행복해하는 사람
    - V: Vacationer (휴양객)  
현재 회고에 흥미는 없지만, 매일 반복되어 지겨운 일상에서 잠시나마 떠나있는 행복을 느끼는 사람
    - P: Prisoner (죄수)  
억지로 앉아 있다고 느끼고, 차라리 다른 일을 하고 싶어하는 사람
  - 결과를 칠판에 붙여, ESVP 가 각각 몇 명인지 팀원 전체가 알게 합니다.
  - “이 결과에 대해 어떻게 생각하나요?” 라는 질문으로 보다 심도 있는 대화로 진행하는 기초로 활용합니다.
- 기대효과
  - 이터레이션을 수행하면서 느꼈던 감정과 데이터들을 정리할 수 있는 시간을 제공합니다.
  - 나 이외에 다른 팀원들이 지난 이터레이션에 생각난 내용을 공유합니다.

# 1-3. Retrospective 절차

## 2. 데이터 모으기 : 좋았던 점 / 나빴던 점 모으기

- 지난 이터레이션 동안 일어난 일에 대해 팀원이 느낀 공통 분모를 찾는 활동
- Time Box: 20분
- 수행 방법
  - 참여 인원에게 두 가지 색깔의 포스트잇을 각각 3 장씩 나눠줍니다.
  - 참여 인원에게 이터레이션 동안 일하는데 도움이 되었던 요소와 방해가 되었던 요소
  - 가지를 각각 다른 색의 포스트잇에 적게 합니다.
  - 작성이 완료된 사람의 포스트잇부터, 왼쪽에는 도움이 된 요소, 오른쪽에는 방해가 되었던 요소로 벽에 붙이고 비슷한 내용끼리 그룹핑합니다.
  - 진행자는 그룹핑된 내용을 간략하게 정리하여 전체 인원과 공유합니다.
- 수행 팁
  - 인원이 많을 경우(10 명 이상) 시간을 단축시키기 위해, 포스트잇을 3 장 대신 2 장만 사용합니다.
  - 포스트잇이 많을 경우 벽에 붙이는 시간을 줄이기 위해, 팀원 중 한 명에게 도움을 요청합니다.
- 기대효과
  - 아주 가벼운 내용부터 무거운 내용까지 다양한 의견이 공유될 수 있는 장을 만듭니다.
  - 비슷한 내용이 많다면 꼭 개선해야 할 내용일 수 있습니다.

# 1-3. Retrospective 절차

## 2. 데이터 모으기 : 학습 매트릭스

- Time Box: 30분
- 수행 방법
  - 화이트 보드에 X/Y 축을 그리고, 왼쪽 위부터 잘된 점(+), 나빴던 점(-), 고칠 내용(C), 불확정/기타(?) 을 적습니다.
  - 진행자는 해당 내용에 대해 팀원들에게 설명하고 20 분 동안 자유롭게 화이트 보드에 와서 이터레이션에 대한 아이디어를 적도록 합니다.
- 수행 팁
  - 고칠 내용이 다음 이터레이션에 적용가능한지 팀에 스스로에게 물어보게 합니다.
- 기대효과
  - 팀 스스로 이슈 및 현안에 대해 자유롭게 공유할 수 있게 만들어 Self-Organizing을 연습할 수 있게 해줍니다.

+	-
C	?



# 1-3. Retrospective 절차

## 2. 데이터 모으기 : 화남, 기쁨, 슬픔

- Time Box: 20분
- 수행 방법
  - 참여자에게 각각 다른 색깔(빨간색, 노란색, 파란색)의 포스트잇 3장을 을 나눠줍니다.
  - 이터레이션 동안 기억날 만한 이벤트들을 상기시켜 줍니다.
  - 이터레이션 동안 가장 화가 났던 이벤트를 빨간색 포스트잇에 적고, 가장 기뻐던 이벤트를 노란색 포스트잇에 적고, 가장 슬펐던 이벤트를 파란색 포스트잇에 적게 합니다.
  - 진행자는 포스트잇을 수집하여 3 그룹으로 나눠 왼쪽, 가운데, 오른쪽에 각각 화남, 기쁨, 슬픔을 적습니다.
  - 진행자는 해당 내용을 리포팅하면서 각각 이벤트들에 대해 상기시킵니다.
- 수행 팁
  - 이 회고 방식은 부정적인 의견을 보다 많이 도출시키기 때문에, 긍정적인 내용을 나중에 이야기 해야 좋은 분위기로 나머지 회고를 진행할 수 있습니다. 따라서 슬픔 → 화남 → 기쁨 순서대로 리포팅하는 것이 좋습니다.
- 기대효과
  - 부정적인 감정을 긍정적인 감정과 적절히 균형을 유지하여, 문제를 팀이 직시하고 해결할 수 있는 장을 만듭니다.

# 1-3. Retrospective 절차

## 2. 데이터 모으기 : 팀 만족도 그래프 그리기

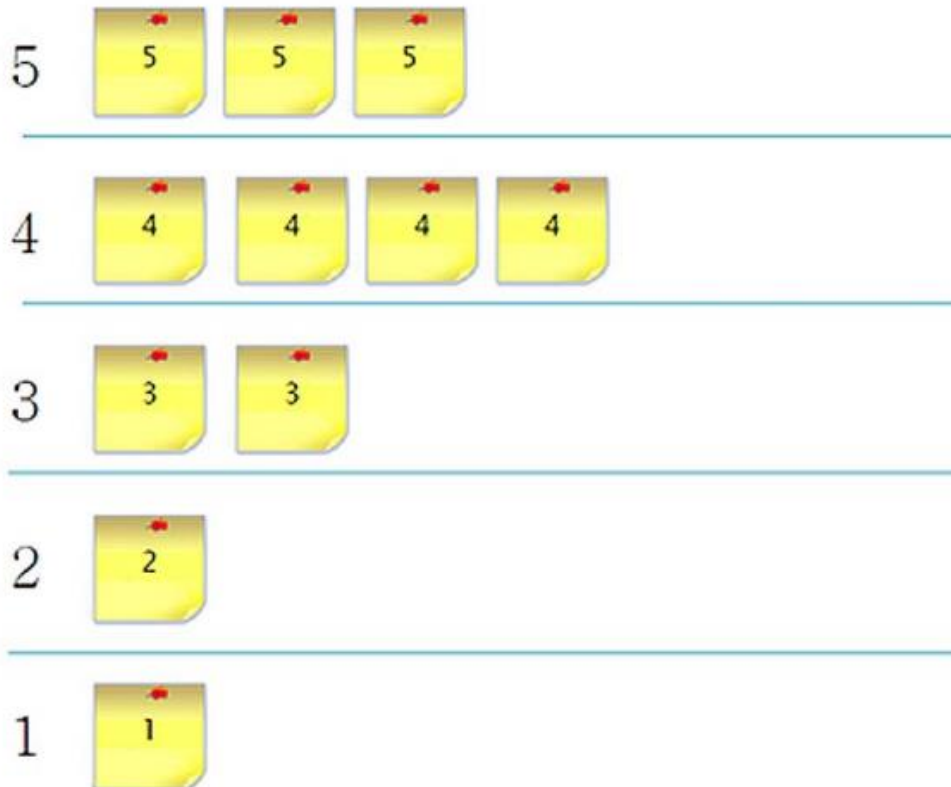
- Time Box: 20분
- 수행 방법
  - 참여 인원들에게 포스트잇 한 장씩을 나누어 줍니다.
  - A4 지에 다음이 적혀있는 종이를 나누어 주고, 포스트잇에 개인이 생각하는 팀의 수준을 숫자로 적게 합니다.
  - 각 레벨에 대한 정의를 모두에게 공유합니다.
  - Lv 5: 우리 팀은 세계 최고의 팀이다.
  - Lv 4: 나는 내가 이 팀의 일원이어서 기쁘고 우리 팀이 협력하는 방식에 만족한다.
  - Lv 3: 제법 만족한다. 우리는 협업을 잘하는 편이다.
  - Lv 2: 만족할 때도 있지만 그렇지 않을 때도 있다.
  - Lv 1: 불행하다. 팀 워크 수준이 불만족스럽다.
  - 벽에 별 수치를 알기 쉽게 숫자 별 일렬로 포스트잇을 나열하여 붙입니다.
  - 팀원들에게 5분 정도의 시간을 주고, 우리 팀 전체의 만족도가 현재 상태인 이유를 고민하게 합니다.
  - 팀원 한 명씩 차례로 우리 팀의 상태에 대한 원인을 추측한 내용을 발표하게 합니다.
- 수행 팁
  - 개선할 내용이 다음 이터레이션에 적용 가능한지 팀 스스로 정해야 합니다.
- 기대효과
  - 팀 스스로 이슈 및 현안에 대해 자유롭게 공유할 수 있게 만들어 Self-Organizing을 연습할 수 있게 해줍니다.



# 1-3. Retrospective 절차

## 2. 데이터 모으기 : 팀 만족도 그래프 그리기

팀워크 만족도





# 1-3. Retrospective 절차

## 3. 통찰 이끌어내기:점 투표하기

- 모아진 데이터를 이용하여 어떠한 주제에 대해 토의할 지 논의하여, 보다 개선된 팀이 되 기 위해 무엇을 해야 할 지에 관한 실천계획(Action Plan)을 만드는 활동
- Time Box: 10분
- 수행 방법
  - 점 스티커를 팀원들에게 나누어 줍니다.
  - 팀원들에게 마음 속으로 모아진 데이터에 대해 고민하도록 하고, 팀에 가장 도움이 되었던 것과 가장 방해가 되었던 것을 우선순위화 하도록 합니다.
  - 팀원들에게 우선순위가 높은 순으로 두 가지에 대해 모아진 데이터가 있는 곳에 직접 가서, 투표하도록 합니다.
  - 표가 많이 있는 도움이 된 내용과, 방해가 된 내용 2 가지 주제에 대해 리포팅 합니다.
  - 15분 간 팀원들에게 도움이 된 내용은 어떻게 더 도움이 될 지, 방해가 된 내용은 어떻게 개선할 지에 대해 토의를 합니다.
  - 결론을 각 그룹의 대표가 이야기하게 합니다.
- 수행 팁
  - 토의에 적절한 인원은 4 명 정도입니다. 6 인 이상이면 그룹으로 나누어 토의합니다.





# 1-3. Retrospective 절차

## 4. 무엇을 할 지 결정하기: SMART 한 회의

- Time Box: 20분
- 수행 방법
  - SMART 한 목표에 대해 설명합니다.
    - ✓ S: Specific (상세하고)
    - ✓ M: Measurable (측정 가능하고)
    - ✓ A: Attainable (달성 가능하고)
    - ✓ R: Relative (팀의 개선과 관련이 있으며)
    - ✓ T: Timely (이터레이션 안에 실행 가능해야 합니다)
  - 팀에게 15분의 시간을 주고, 주어진 데이터에 대해, 다음 이터레이션 동안 무엇을 SMART 하게 수행하면, 팀에 도움이 될 지 토의하도록 합니다.
  - 그룹 별 회의 결과를 그룹에서 한 명씩 타 팀과 공유할 수 있도록 리포팅합니다.
- 수행 팁
  - 인원이 4 인 이상이면 그룹으로 나누어 토의합니다.



# 1-3. Retrospective 절차

## 4. 무엇을 할 지 결정하기: EXACT한 목표기준 세우기

- Time Box: 20분
- 수행 방법
  - EXACT 한 목표에 대해 설명합니다.
    - ✓ EX: Exciting (긍정적이고 에너지가 넘치는 영감을 주고)
    - ✓ A: Assessable (성취한 것을 확인 가능하며)
    - ✓ C: Challenging (도전적인 목표이고)
    - ✓ T: Timely (기한이 있어야 합니다)
  - 팀에게 15분의 시간을 주고, 주어진 데이터에 대해, 다음 이터레이션 동안 무엇을 SMART 하게 수행하면, 팀에 도움이 될 지 토의하도록 합니다.
  - 그룹 별 회의 결과를 그룹에서 한 명씩 타 팀과 공유할 수 있도록 리포팅합니다.
- 수행 팁
  - 인원이 4 인 이상이면 그룹으로 나누어 토의합니다.

# 1-3. Retrospective 절차

## 5. 회고 마무리하기: 회고활동에 대한 회고 (+/델타(Delta))

- 수행한 회고를 마무리하고, 다음 이터레이션의 시작을 알리는 활동
- Time Box: 10분
- 수행 방법
  - 플립 차트에 T 자 형태의 표를 그리고 제한 시간을 알려줍니다.
  - 강점과 변화를 마음껏 말하도록 유도합니다.
  - 사람들의 피드백에 감사하며 더 발전시킬 것과 변화시킬 것을 정합니다.
- 수행 팁
  - T 자 형태의 표를 기준으로 피드백의 수를 제한하는 방법도 있습니다.
  - 변화시킬 것이 결정되었다면 담당자를 정해 추적하게 하는 것이 좋습니다.
- 기대효과
  - 회고를 끝내기 전에 다음 회고에서 이번 회고의 무엇을 유지하며 어떤 변화를 시도할지 결정할 수 있습니다.

회고 프로세스 개선	
+	△

## 5. 회고 마무리하기: Follower 구하기

- Time Box: 10분
- 수행 방법
  - 회고를 통해 결정한 사안의 중요성과 가치에 대해서 공유합니다.
  - 회고를 통해 결정한 사안에 대해 추적 관리할 담당자를 구합니다.
  - 자발적으로 문제를 추적 관리할 담당자가 결정되면 진행자는 일의 중요성을 다시 강조하며 모두 함께 담당자를 격려합니다.
- 수행 팁
  - 강요 되지 않도록 주의합니다.
- 기대효과
  - 팀원 사이에 자발적인 스폰서십을 구하여 자기조직적인 팀 분위기를 형성합니다.

# 프로젝트 계획하기

## 2교시 :

### 시각적 관리 활용





## 2-1. 작업 현황판

### 학습목표

- 이 워크샵에서는 작업 현황판에 대해 알 수 있다.

### 눈높이 체크

- 제품 백로그 , 스프린트 백로그 등 Story Board 와 Scrum Board, Kanban Board에 대해 알고 계신가요?



## 2-1. 작업 현황판

### 작업 현황판에 게시

- 두 이터레이션 계획 회의(팀 활동 계획) 중 두 번째로, 팀은 방금 이터레이션 백로그 내에 포함시키기로 합의한 스토리들을 어떻게 전달할지를 정한다. 이 회의에서 다음을 수행한다.
- 누가 어떤 활동에 대해 책임질 지와 이터레이션 백로그에서 어떤 스토리들에 책임을 맡을지 정하기
- 스토리를 완료하는 데 필요한 전문가 도움 확인하기
- 스토리들 간의 우선순위와 의존도에 기반해 이터레이션 기간 동안의 작업 일정 정하기
- 관련 사용자를 인수 테스트에 참여하도록 초대하기

## 2-1. 작업 현황판

### 작업 현황판에 게시

- 어떤 순서로 사용자 스토리들을 작업 현황판에 게시할 것인가?
- 처음에는 어떤 순서로 카드들을 작업 현황판에 올리지는 중요하지 않으며, 일정관리 하는 동안에 어떤 순서로 카드들을 작업할지 토론하게 될 것이다. 처음으로 그것들을 작업 현황판에 놓을 때, 전부 인터레이션 백로그 첫 세로줄에 넣으면 된다.

PIP인터레이션10:

10개의 스토리. 예상 속도= 32 스토리 포인트

재규 얼른 회복하기를 - 그림다!

인터레이션 백로그	개발	스토리 테스트	인수 테스트	배포
S2 <input type="checkbox"/>				
S6 <input type="checkbox"/>				
S8 <input type="checkbox"/>				
S1 <input type="checkbox"/>				
S7 <input type="checkbox"/>				





## 2-1. 작업 현황판

### 일정관리

- 일정관리는 팀이 얼마나 빨리 작업 하는지- 어떻게 작업 현황판을 가로질러 이동할지, 어떤 순서로, 그리고 누가 무엇을 하는지의 순서를 정하도록 돕는다. 종종, 특히 작은 팀에서, 이러한 일정관리는 비공식적으로 행해지며 전혀 문서화되지 않는다. 일정관리는 작업 현황판에 추가될 수 있다.
- 일정관리는 팀이 얼마나 빨리 작업 하는지- 어떻게 작업 현황판을 가로질러 이동할지, 어떤 순서로, 그리고 누가 무엇을 하는지의 순서를 정하도록 돕는다. 종종, 특히 작은 팀에서, 이러한 일정관리는 비공식적으로 행해지며 전혀 문서화되지 않는다. 일정관리는 작업 현황판에 추가될 수 있다.



## 2-2. Story Board

### 제품 백로그

업무 구분	상위 기능(에픽)	ID	스토리	스토리 점수
인터넷 서점 v1.0	사용자 로그인	A1	사용자는 회원 아이디와 비밀번호를 입력하고 로그인할 수 있다.	2
		A2	비회원은 회원 가입 없이 1회 로그인을 할 수 있다.	1
	도서 검색 (제목별, 저자별)	A3	사용자는 제목별, 저자별로 도서를 검색할 수 있다.	3
		A4	사용자는 선택한 책의 상세 정보를 볼 수 있다.	2
	도서주문	A5	사용자는 배송지 주소를 입력하여 책을 구입할 수 있다.	2
		A6	사용자는 검색한 도서 목록에서 원하는 도서를 장바구니에 담을 수 있다.	3
		A7	사용자는 자신의 장바구니를 조회하고 수량을 변경/삭제할 수 있다.	2



## 2-2. Story Board

### 스프린트 백로그

ID	사용자 스토리	완료조건	SP	스프린트 백로그			
				ID	작업	지원자	MD
1	사용자는 회원 아이디와 비밀번호를 입력하고 로그인할 수 있다 .	회원 아이디와 비밀번호가 불일치할 때는 메시지를 표시하여 다시 입력하게 한다.	2	1.1	웹페이지 설계		2
				1.2	사용자 암호 정책 설정과 구현		3
				1.3	패스워드 정책 검증 구현		2
				1.4	계정 잠금 처리 구현		3
				1.5	단위 테스트		0.5
2	사용자는 자신의 장바구니를 조회하고 수량을 변경 /삭제할 수 있다 .	특정 책의 수량 조절이 가능해야 한다.	2	2.1	웹페이지 설계		2
				2.2	장바구니 조회 구현		2
				2.3	장바구니 변경 구현		3
				2.4	단위 테스트		0.5
A				A.1	○○보고서 연동 API 개발	홍길동	3
				A.2	○○매뉴얼 개선 작업	박문수	3
				A.3	○○업무 개선 회의	팀전체	3
				A.4	코드 리뷰	팀전체	5




## 2-3. Scrum Board

### Scrum Board란?

- 두 이터레이션 계획 회의(이터레이션 백로그 선택) 중 첫 번째에서 제품 책임자와 팀은 이터레이션 동안 무엇이 수행될지에 동의한다. 회의에서 아래와 같은 일을 수행한다.
  - 어떤 사용자 스토리들이 요구되는지 확인하기
  - 스토리들 간 의존성 확인하기
  - 사용자 스토리에 배정될 스토리 포인트 추정하기
  - 사용자 스토리들의 우선순위와 팀의 속도에 기반해 이터레이션 내에 완성될 수 있는 사용자 스토리들 선택하기

# 2-3. Scrum Board

## Scrum Board란?

Product Backlog				SprintBacklog	To-do	In Progress	Done	Sprint Objective
ThemeA	Epic1	Story1	Task1	Story1		Task1		 Sprint BurnDownChart
			...					
		Story2	Task1	Story2	Task1			
			...					
	Epic2	Story1	Task1					
			...					
		Story2	Task1					
			...					
ThemeB	Epic1	Story1	Task1	Story1			Task1	
			...					
		Story2	Task1	Story2	Task1			
			...					
	Epic2	Story1	Task1					
			...					
		Story2	Task1					
			...					



## 2-4. Kanban Board

### Kanban Board란?

- 두 이터레이션 계획 회의(이터레이션 백로그 선택) 중 첫 번째에서 제품 책임자와 팀은 이터레이션 동안 무엇이 수행될지에 동의한다. 회의에서 아래와 같은 일을 수행한다.
  - 어떤 사용자 스토리들이 요구되는지 확인하기
  - 스토리들 간 의존성 확인하기
  - 사용자 스토리에 배정될 스토리 포인트 추정하기
  - 사용자 스토리들의 우선순위와 팀의 속도에 기반해 이터레이션 내에 완성될 수 있는 사용자 스토리들 선택하기

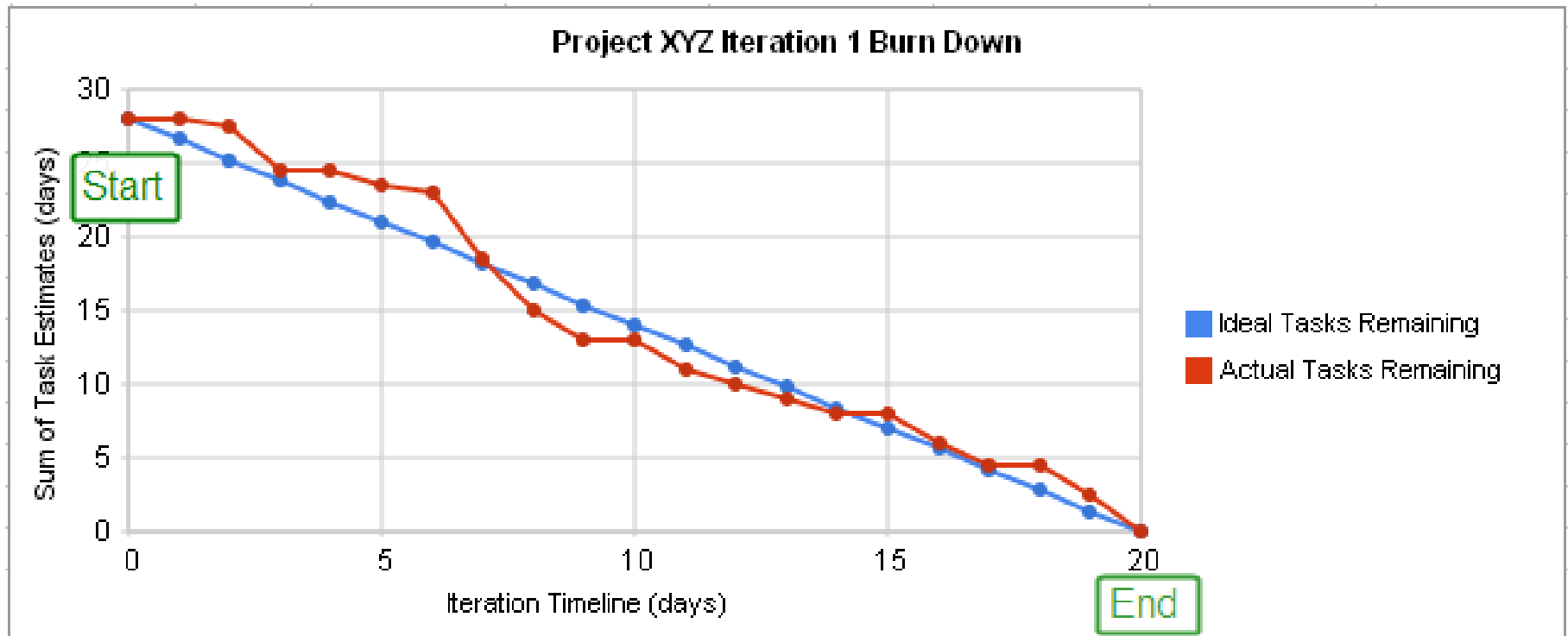
# 2-4. Kanban Board

## Kanban Board란?

Product Backlog				To-do	Development		Testing		Deployment	Done
					Ongoing	Done	Ongoing	Done		
ThemeA	Epic1	Story1	Task1			Task1				
			...							
		Story2	Task1	Task1	Task1					
			...							
	Epic2	Story1	Task1							
			...							
		Story2	Task1							
			...							
ThemeB	Epic1	Story1	Task1							
			...							
		Story2	Task1	Task1	Task1					
			...							
	Epic2	Story1	Task1							
			...							
		Story2	Task1							
			...							

## 2-5. Scrum Checklist(스크럼 체크리스트)

### Burndown Chart





# 프로젝트 계획하기

## 3교시 :

### 개념 분석과 유스케이스 다이어그램





## 3-1. SDLC와 소프트웨어 개발 방법론

### 학습목표

- 이 워크샵에서는 SDLC(Software Development Life-Cycle) 와 소프트웨어 개발 방법론의 관계에 대해 알 수 있다.

### 눈높이 체크

- SDLC(Software Development Life-Cycle) 에 대해 알고 계신가요?



## 3-1. SDLC와 소프트웨어 개발 방법론

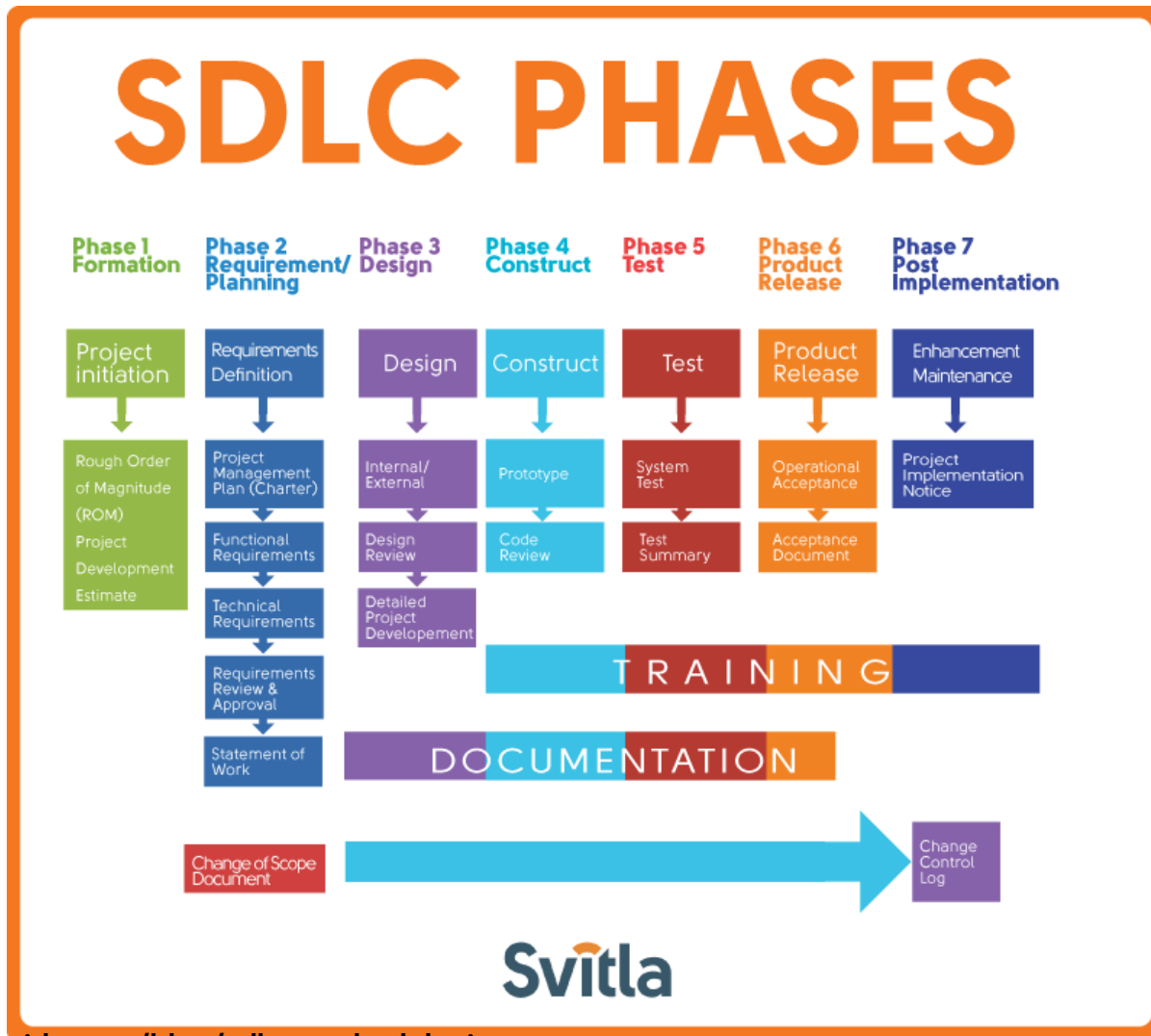
### 소프트웨어개발의 생명 주기 정리

- 소프트웨어 개발 수명 주기(Software Development Life-Cycle: SDLC)는 소프트웨어 개발 프로세스 중에 수행되는 일련의 다양한 활동. 소프트웨어개발의 생명 주기(SDLC)는 고객의 요구에 의해서 소프트웨어 시스템이 탄생하고, 가동, 운용되는 가운데에 유지 보수가 반복되고, 최종적으로 수명이 다하여 파기할 때까지의 전 공정을 체계화한 개념이다. 시스템이 개발될 때부터 운용과 보수를 거쳐 생애를 마칠 때까지 어떠한 순서를 밟는지에 대한 작업 프로세스를 모델화한 것이다.



# 3-1. SDLC와 소프트웨어 개발 방법론

## 소프트웨어개발의 생명 주기 정리



출처:

<https://svitla.com/blog/sdlc-methodologies>

# 3-1. SDLC와 소프트웨어 개발 방법론

## 소프트웨어개발의 생명 주기 정리

- 응용 소프트웨어 개발에 사용할 표준으로 활동 및 절차 수행에 필요한 기법과 표준, 산출물 표준 양식 및 작성 기법, 적용 도구를 정립할 수 있다.

## 소프트웨어개발 방법론의 종류

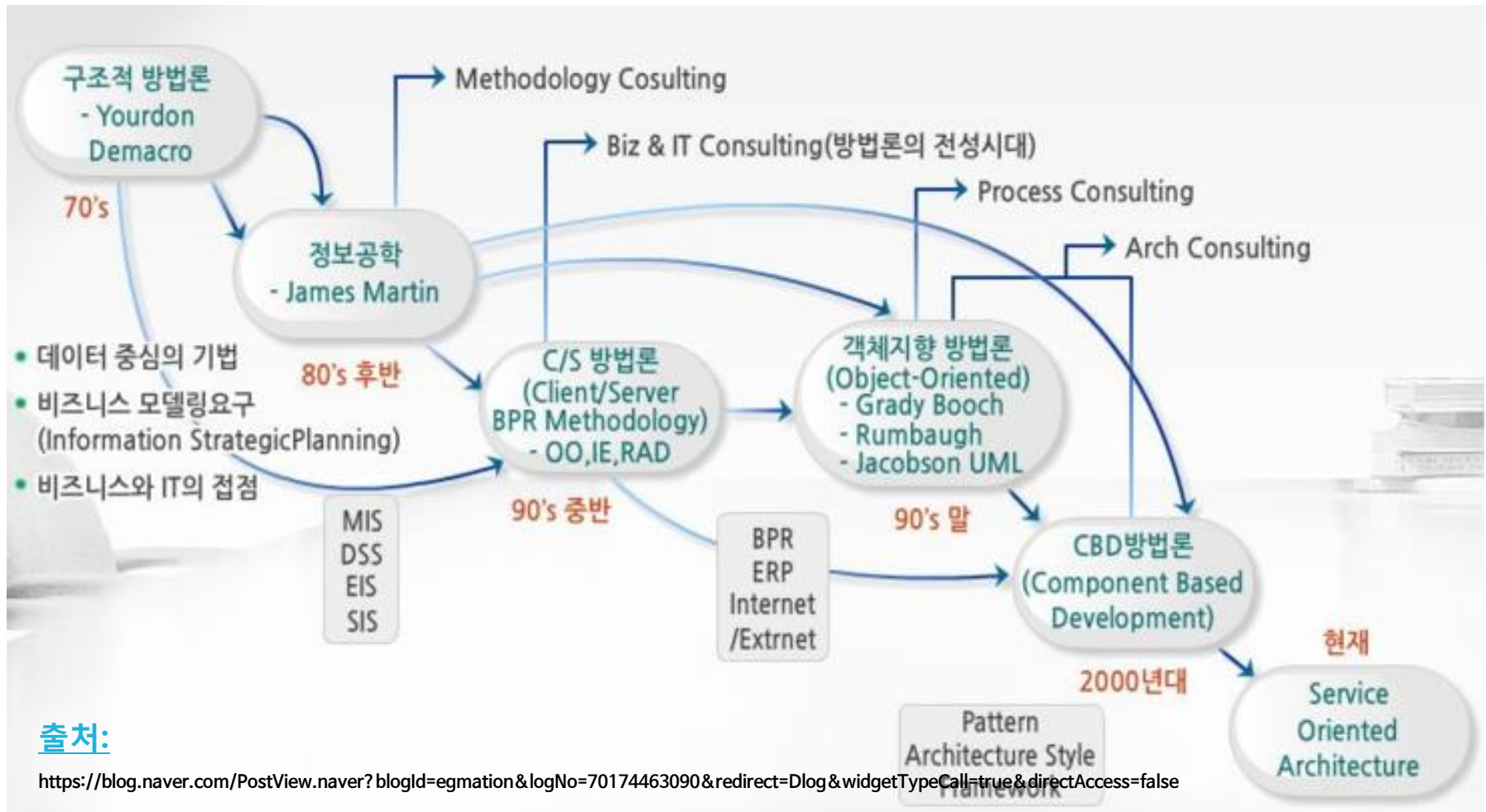
- 세대별 종류

세대	구분	내용
1970년대	구조적 방법론	구조화 프로그래밍
1980년대	정보공학 방법론	관리 절차와 작업 기법을 체계화
1990년대	객체지향 방법론	데이터와 그 데이터에 관련되는 동작을 모두 포함
2000년대	컴포넌트 기반 방법론	컴포넌트를 조립해서 새로운 응용 프로그램을 작성
2000년대	Agile 방법론	요구사항, 설계, 구현, 테스트의 과정으로 개발
2010년대	제품 계열 방법론	특정 제품에 수용하고 싶은 공통된 기능을 정의

# 3-1. SDLC와 소프트웨어 개발 방법론

## 소프트웨어개발 방법론 선정

- Agile 방법론과 함께 사용할 수 있는 개발 방법론



## 3-2. 애플리케이션 요구 사항 도출하기

### 학습목표

- 이 워크샵에서는 하드웨어, 장비에 내재될 수 있는 소프트웨어 요구 사항을 수집하고 명확한 애플리케이션 요구 사항이 무엇인지 파악하고 정의할 수 있다.
- 애플리케이션 요구 사항의 현황 파악을 통하여 문제 해결에 대한 영역을 도출할 수 있다.
- 애플리케이션 요구 사항의 도출된 결과를 도구를 활용하여 UML 다이어그램으로 표현 할 수 있다.

### 눈높이 체크

- UML 다이어그램에 대해 알고 계신가요?

## 3-2. 애플리케이션 요구 사항 도출하기

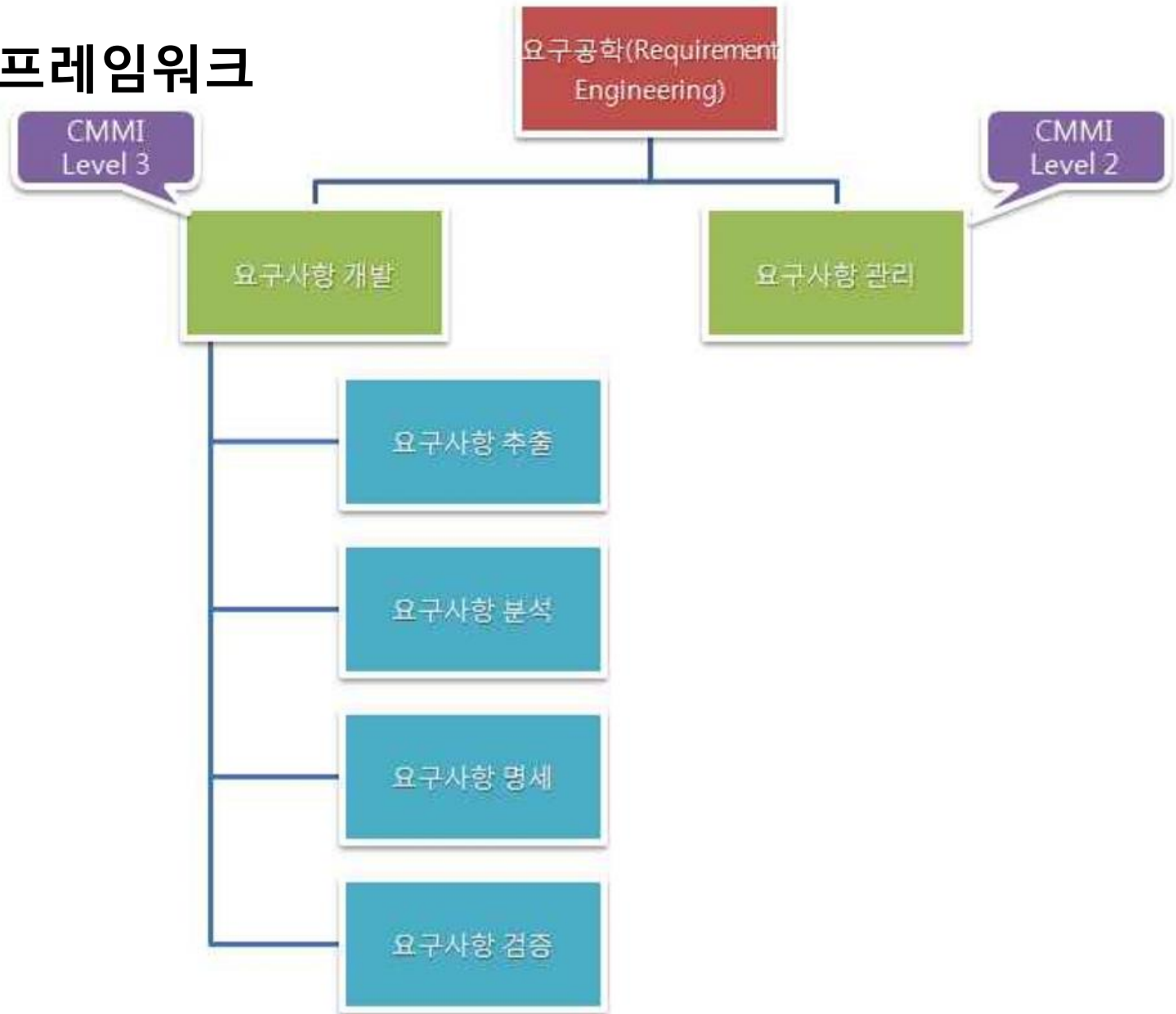
### 요구 공학(requirement engineering)

- 요구 공학이란 요구 사항 정의와 분석을 통해서 추출된 요구 사항의 추적 관리, 유지 등을 포함하여 요구 사항과 관련된 모든 활동과 원칙들에 대한 공학적 접근 기법이다.
- 요구 공학 절차는 크게 요구 사항 개발과 요구 사항 관리로 이루어진다. 요구 사항 개발 절차는 요구 사항 추출, 요구 사항 분석, 요구 사항 정의, 요구 사항 검증의 절차로 이루어지며 이를 통해 개발된 요구 사항을 관리하는 요구 사항 관리 절차가 있다.



## 3-2. 애플리케이션 요구 사항 도출하기

### 요구 공학 프레임워크



## 3-2. 애플리케이션 요구 사항 도출하기

### 요구 공학의 주요 프로세스

절차	설명	주요 기법
요구 사항 추출	제품 개발을 위한 사용자 요구 사항 및 제약 사항을 추출하는 절차	FGI, 브레인스토밍, 워크숍
요구 사항 분석	식별된 기능 및 비기능 요구 사항을 분석하여 중요도에 따라 우선순위를 부여하고 선별하는 일련의 절차	페르소나, AHP, 트리아제
요구 사항 정의/명세	분석된 요구 사항으로부터 요구 사항 명세서를 작성하는 일련의 절차	정형명세, UML
요구 사항 검증	요구 사항에 대한 유효성, 일관성, 완전성, 실현 가능성, 증명가능성 등을 검증하는 절차	V&V, Review, Inspection
요구 사항 관리	요구 사항 관리 계획 수립 및 요구 사항 변경 관리	기준선, 요구 사항 추적표

### 요구 사항의 종류

- 요구 사항은 시스템의 행위를 기술하는 기능 요구 사항과 시스템의 품질, 속성 혹은 제약 사항 등을 정의한 비기능 요구 사항으로 분류할 수 있다.

기능(Functional) 요구 사항	비기능(Non Functional) 요구 사항
시스템 행위가 어떻게 이루어지는가를 표현	시스템 결과에 의해 나타나는 전체적인 품질이나 속성 또는 기능적 요구 사항을 구현할 때 고려해야 하는 제약 사항을 정의
데이터 모델	성능: 응답 속도, 자원사용량
데이터 흐름 처리 모델	보안: 침입 대응, 사용자 인증
프로세스 모델	아키텍처: 확장성, 유연성
	안정성: 장애 대응, 서비스 연속성

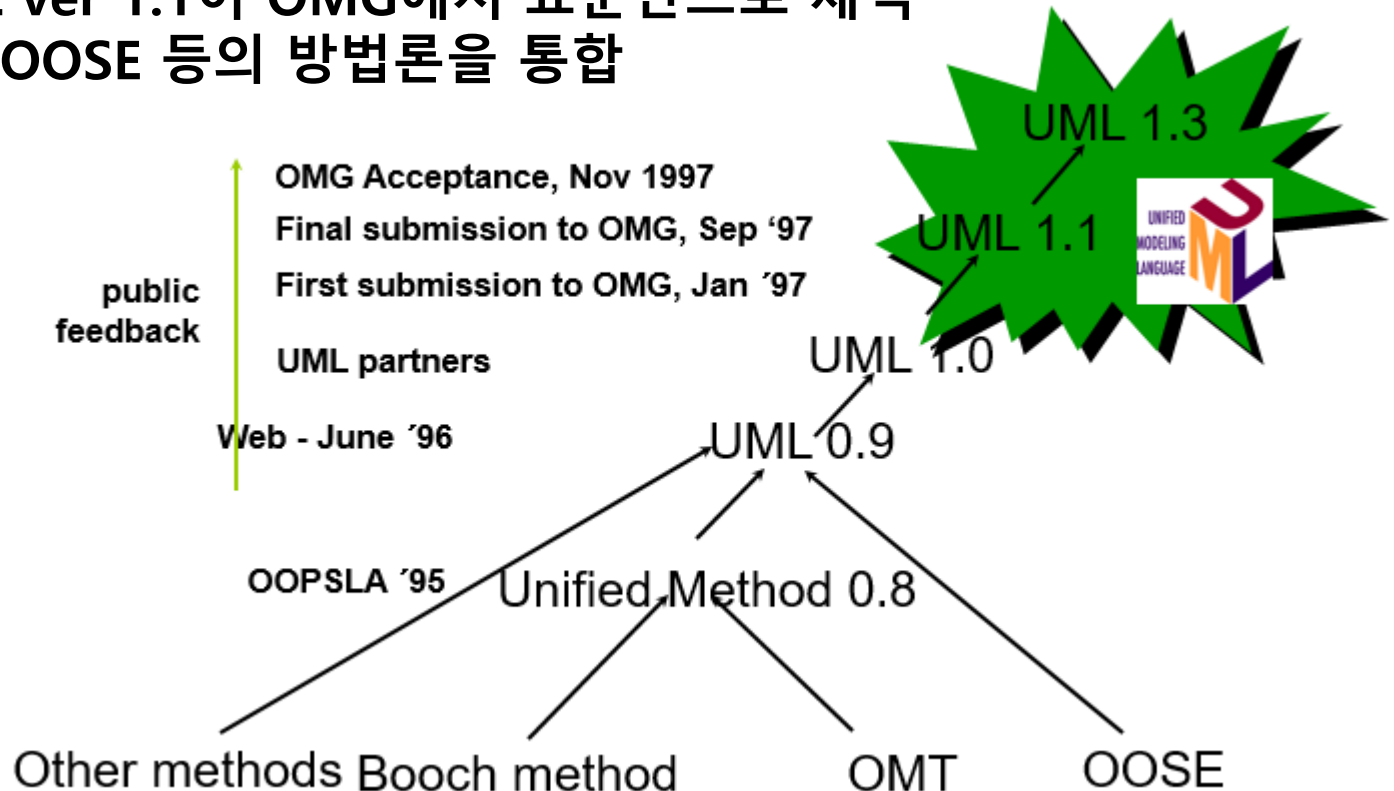
## UML과 Use Case 다이어그램

- UML(Unified Modeling Language)은 객체 기술 언어에 대한 표준화 기구인 OMG에서 제정한 객체 지향 분석, 설계용 모델링 언어이다. UML은 분석/설계자 관점인 Logical View, 개발자 관점인 Implementation View, 시스템 통합자 관점인 Process View, 시스템 엔지니어 관점의 Deployment View와 사용자 관점의 Usecase View로 구분할 수 있다.
- 각 View는 여러 가지 Diagram으로 구성되어지며 사용자 요구 사항에 대한 기술은 주로 Use Case 다이어그램을 이용한다.

## 3-2. 애플리케이션 요구 사항 도출하기

### UML과 Use Case 다이어그램

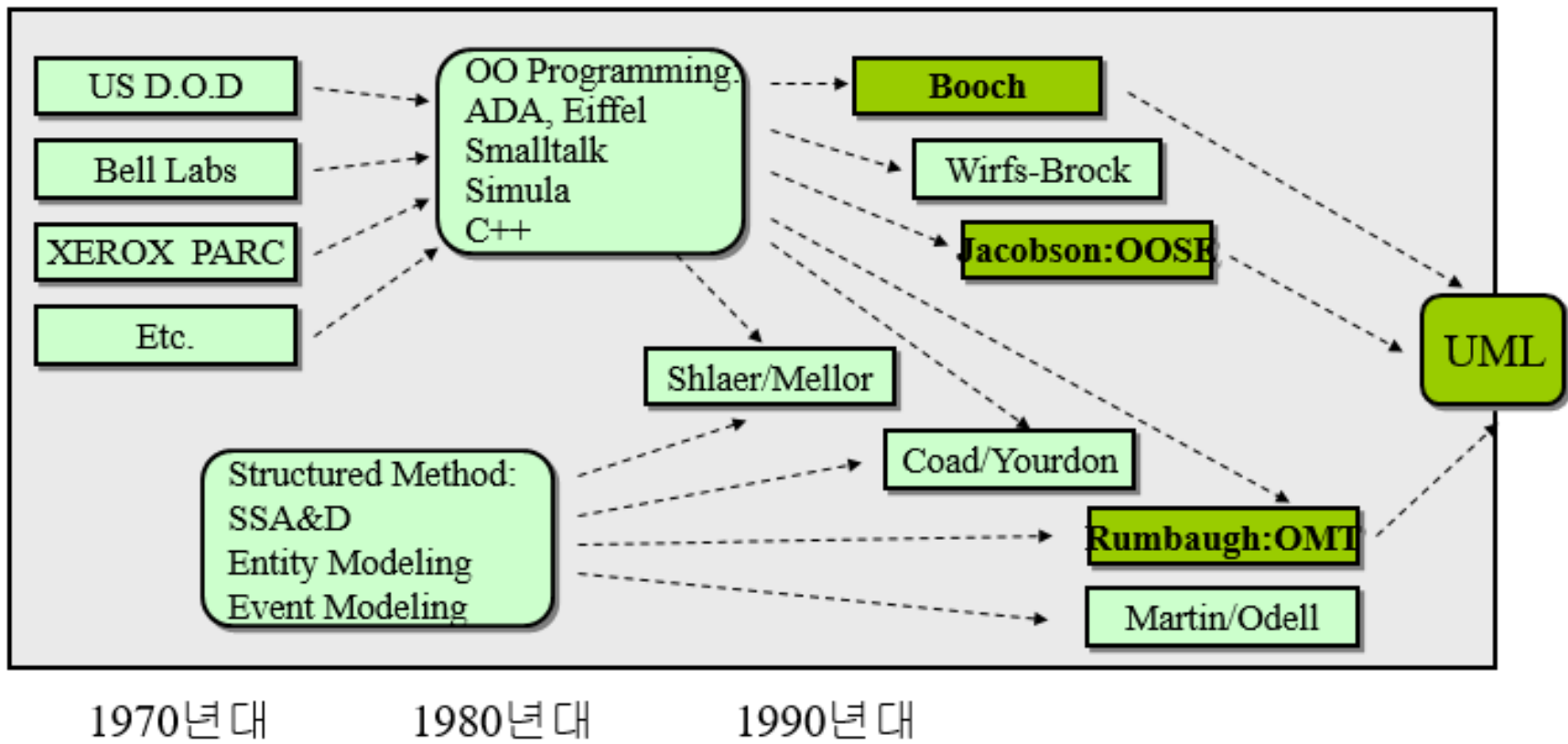
- UML(Unified Modeling Language)
  - OMT, Booch, OOSE 방법론을 통합하여 만든 모델링 개념의 공통 집합
  - 객체지향 분석 및 설계 방법론의 표준 지정을 목표로 제안
  - 97. 11월 UML ver 1.1이 OMG에서 표준안으로 채택
  - Booch, OMT, OOSE 등의 방법론을 통합



## 3-2. 애플리케이션 요구 사항 도출하기

### UML과 Use Case 다이어그램

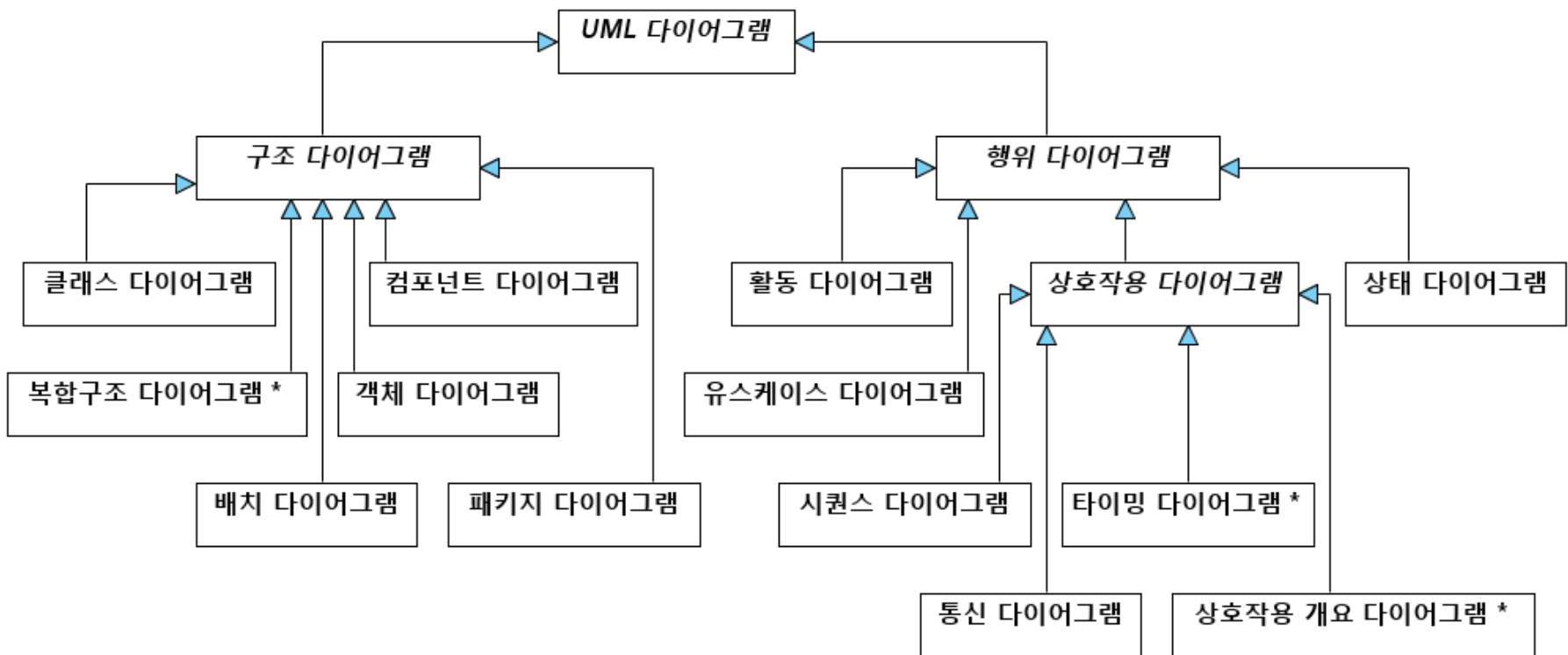
- UML의 역사



## 3-2. 애플리케이션 요구 사항 도출하기

### UML과 Use Case 다이어그램

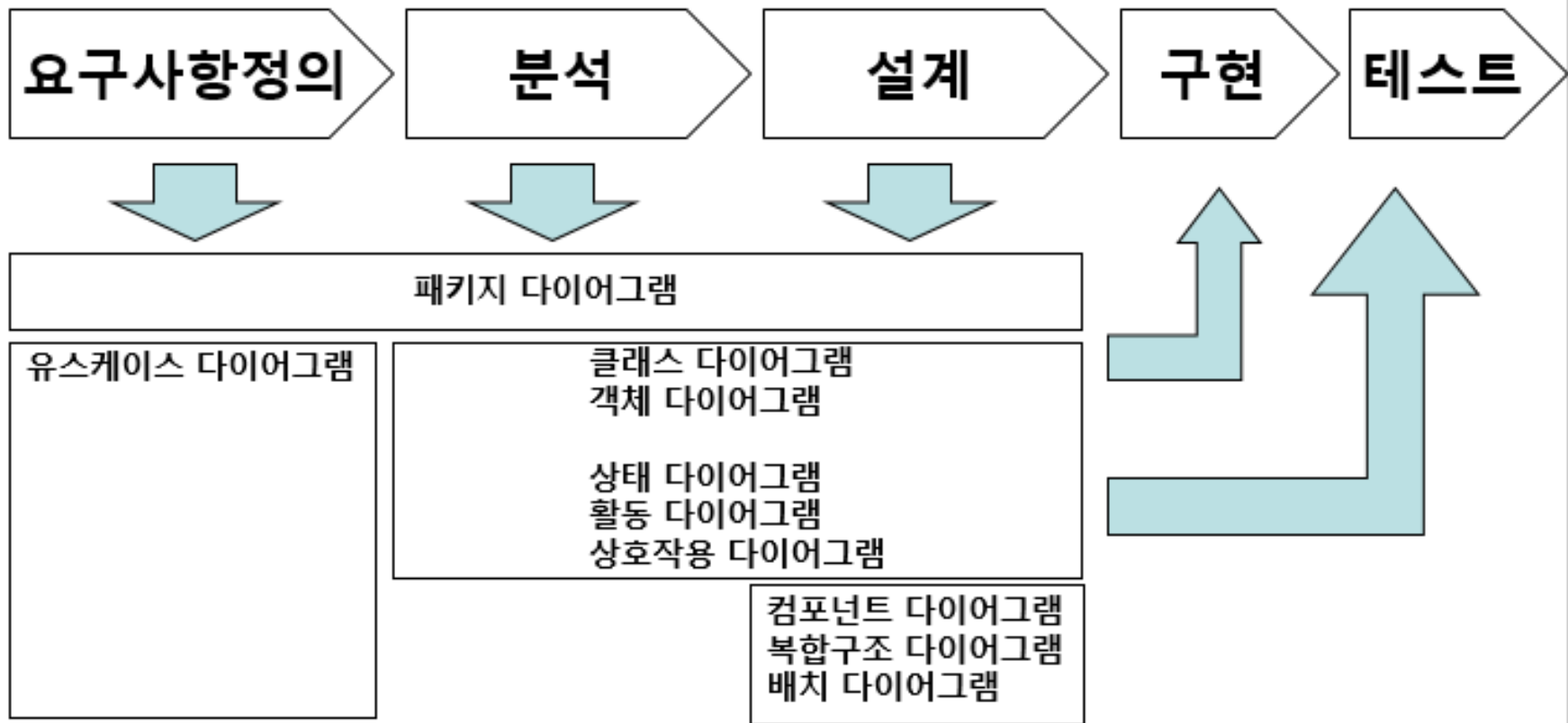
- UML 2.0 Diagram Taxonomy



## 3-2. 애플리케이션 요구 사항 도출하기

### UML과 Use Case 다이어그램

- 개발 단계 별로 주로 사용되는 다이어그램이 있음

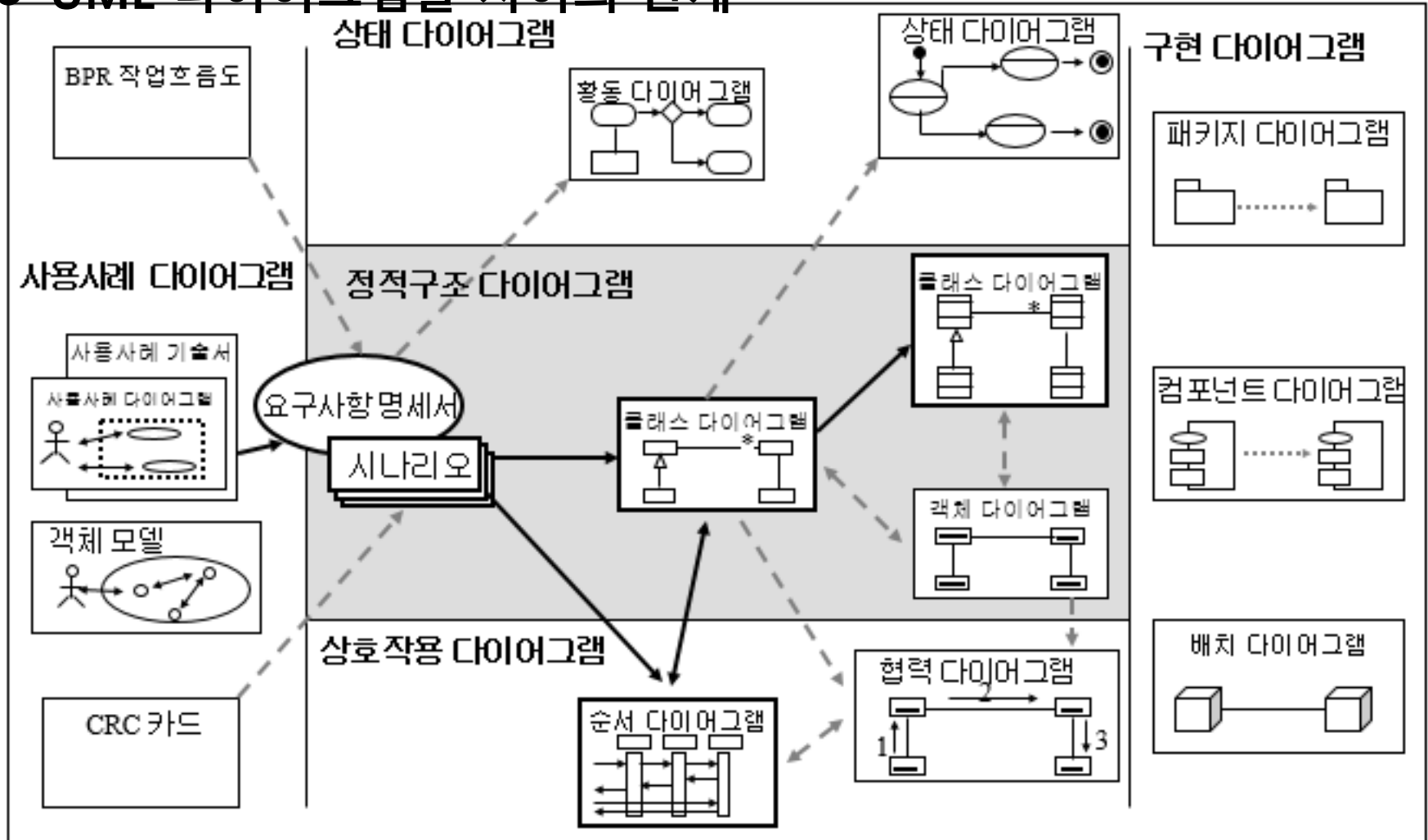




## 3-2. 애플리케이션 요구 사항 도출하기

### UML과 Use Case 다이어그램

#### UML 다이어그램들 사이의 관계



## 3-2. 애플리케이션 요구 사항 도출하기

### UML과 Use Case 다이어그램

#### ● 구조 다이어그램

다이어그램	내용	비고
클래스 다이어그램	시스템을 구성하는 클래스 표현	논리적 수준
객체 다이어그램	시스템을 구성하는 객체	
패키지 다이어그램	많은 수의 모델 요소(예, 클래스, 논리적 컴포넌트)들을 패키지를 이용하여 조직화함	
컴포넌트 다이어그램	시스템을 구성하는 논리적 컴포넌트 표현	
복합구조 다이어그램	논리적 컴포넌트의 내부를 파트(part)와 연결자(connector)로 표현	
배치 다이어그램	시스템을 구성하는 노드와 통신 경로, 배치되는 물리적 컴포넌트를 표현	물리적 수준

## 3-2. 애플리케이션 요구 사항 도출하기

### UML과 Use Case 다이어그램

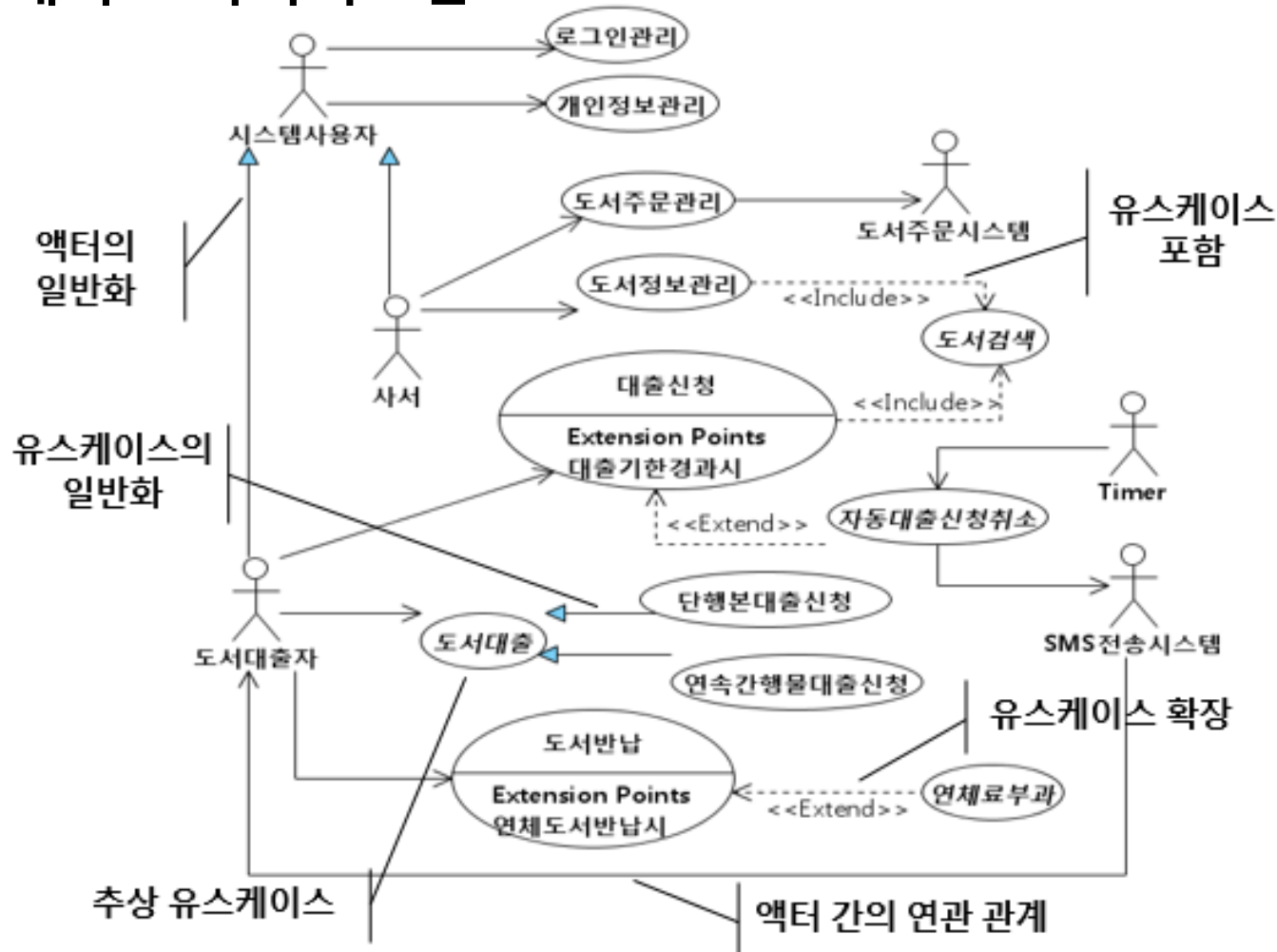
#### ● 구조 다이어그램

다이어그램	내용	비고
유스케이스 다이어그램	시스템의 외부 요소와 기능적 요구사항을 액터와 유스케이스로 표현	시스템의 행위
상태 다이어그램	개별 대상의 동적 행위를 상태와 전이로 표현	개별 구성 요소의 행위
활동 다이어그램	개별 대상의 동적 행위를 활동으로 표현	
시퀀스 다이어그램	상호작용을 구성 요소간의 시간적 순서에 따른 메시지 전달로 표현	구성 요소간의 상호작용
통신 다이어그램	상호작용을 구성 요소간의 관계를 바탕으로 둔 메시지 전달을 표현	
상호작용 개요 다이어그램	여러 상호작용의 관계를 상위 수준에서 표현	
타이밍 다이어그램	구성 요소의 상태 변화의 구체적인 시간으로 표현	

## 3-2. 애플리케이션 요구 사항 도출하기

### UML과 Use Case 다이어그램

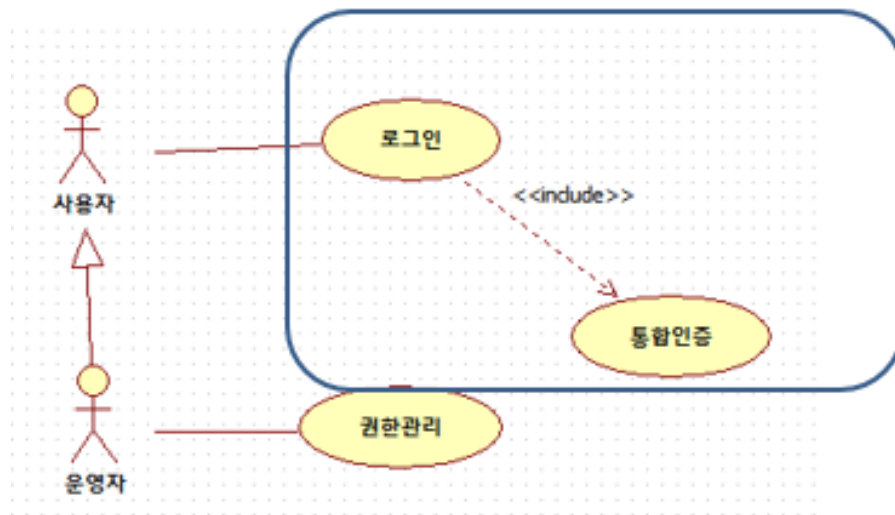
- 유스케이스 다이어그램



## 3-2. 애플리케이션 요구 사항 도출하기

### UML과 Use Case 다이어그램

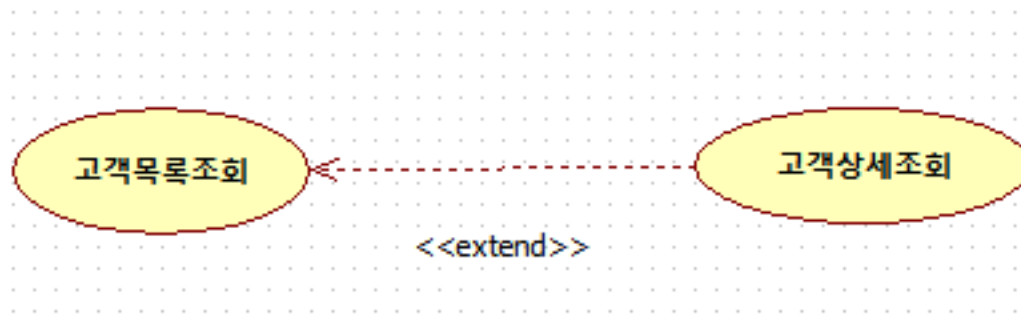
- 유스케이스 다이어그램
  - 포함(Include) : 하나의 유스케이스가 다른 유스케이스를 항상 포함한다는 의미
    - 유스케이스와 유스케이스 사이에 정의되는 관계
    - 한 유스케이스가 다른 유스케이스의 서비스 수행을 요청하는 관계
    - 즉, 한 유스케이스가 자신의 서비스 수행 도중에 다른 유스케이스의 서비스 사용이 필요할 때 정의 (서비스는 반드시 사용이 되어야 함)
    - 포함되는 유스케이스는 공통 서비스를 가진 존재



## 3-2. 애플리케이션 요구 사항 도출하기

### UML과 Use Case 다이어그램

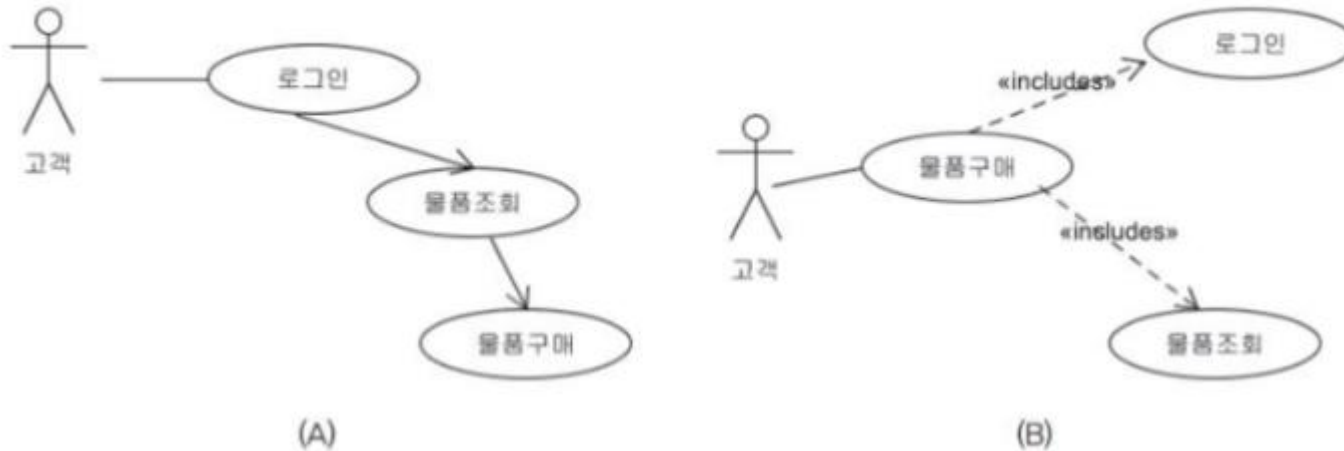
- 유스케이스 다이어그램
  - 확장(Extend): 하나의 유스케이스가 다른 유스케이스로 기능이 확장될 수 있다는 의미
    - 유스케이스와 유스케이스 사이에 정의되는 관계
    - 포함관계와 동일하게 서비스 수행을 요청하는 관계
    - 포함관계와 달리 서비스가 수행되지 않을 수 있음(선택적 유스케이스 관계이다. 필수적이지않고 옵션이라고 할수 있다.)
    - 수행 요청 조건을 확장한 Extention Point이라고 함
    - 표시방법은 포함(include)와 반대이다. (상속이 아니다!)



## 3-2. 애플리케이션 요구 사항 도출하기

### UML과 Use Case 다이어그램

- 유스케이스 다이어그램
- 잘못 작성된 유스케이스 다이어그램 예

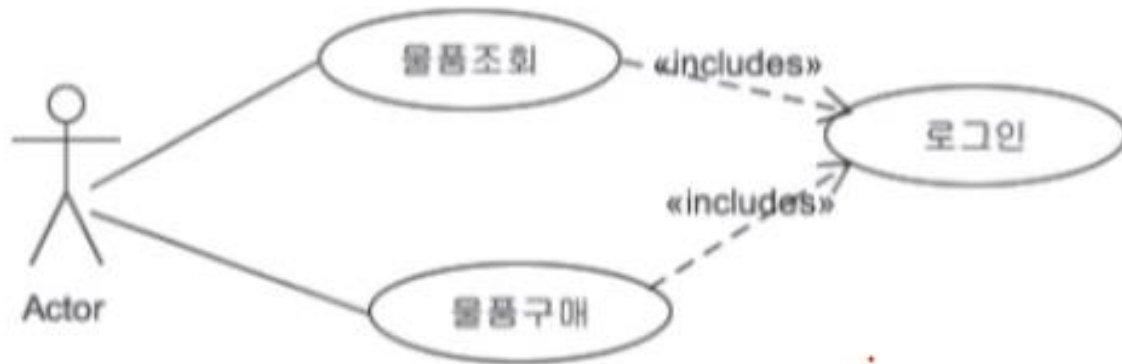


- ❖ 그림 A는 유스케이스 다이어그램을 흐름도처럼 그린 경우이다. 유스케이스 다이어그램을 작성할 때 흐름도를 작성하는 것 처럼 그려서는 안된다. 유스케이스간의 관계는 포함, 확장, 일반화 관계만이 존재하고 실행순서 관계는 나타내지 않는다는 점을 명확히 알아야 한다.
- ❖ 그림 B는 포함관계를 과도하게 사용하는 경우이다. 의도는 물품 구매가 로그인 단계와 물품조회 단계를 포함하기 때문에 포함관계로 연결하였으나 일반적으로 포함 관계는 여러개의 유스케이스가 서로 공통적인 부분을 갖고있는 경우에 중복된 부분을 유스케이스로 분리하여 표현하고자 할 때 사용된다. 예제처럼 물품 조회가 다른 물품 구매를 제외한 다른 유스케이스에서 전혀 사용되지 않는다면 이를 포함관계로 할 필요가 없다.

## 3-2. 애플리케이션 요구 사항 도출하기

### UML과 Use Case 다이어그램

- 유스케이스 다이어그램
  - 올바르게 수정된 유스케이스 다이어그램 예

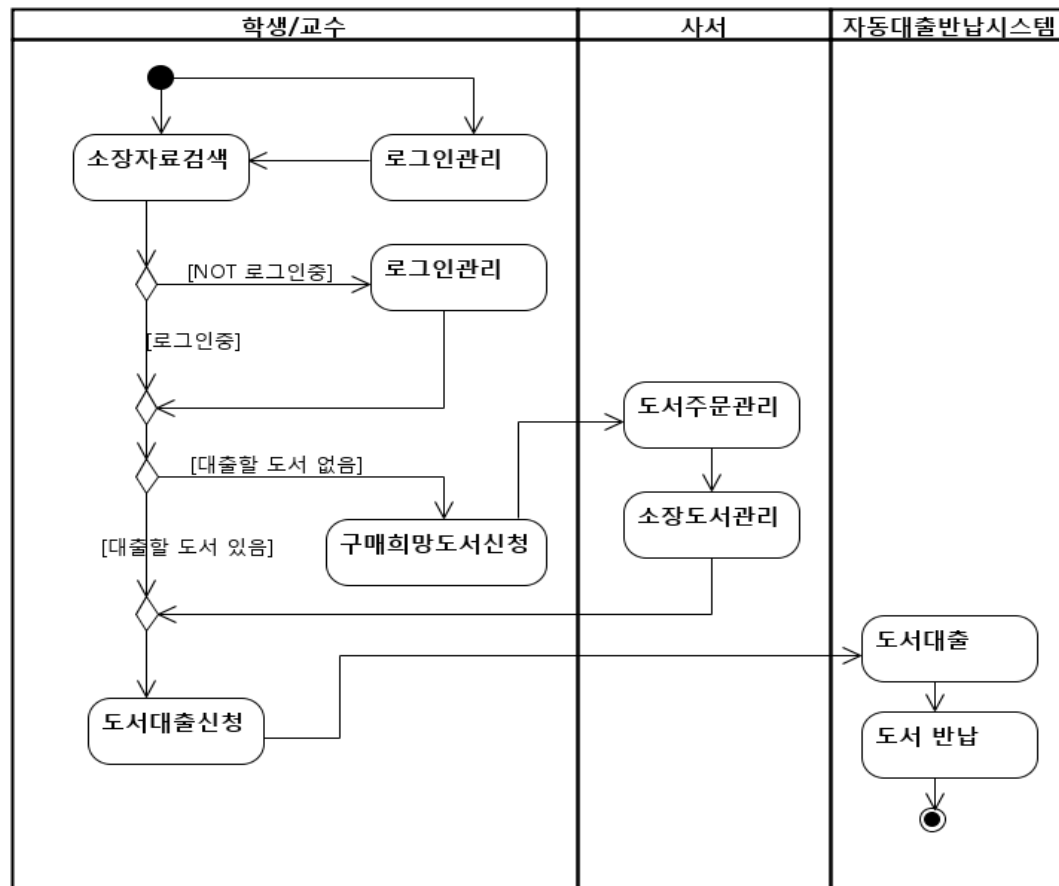




## 3-2. 애플리케이션 요구 사항 도출하기

### UML과 Use Case 다이어그램

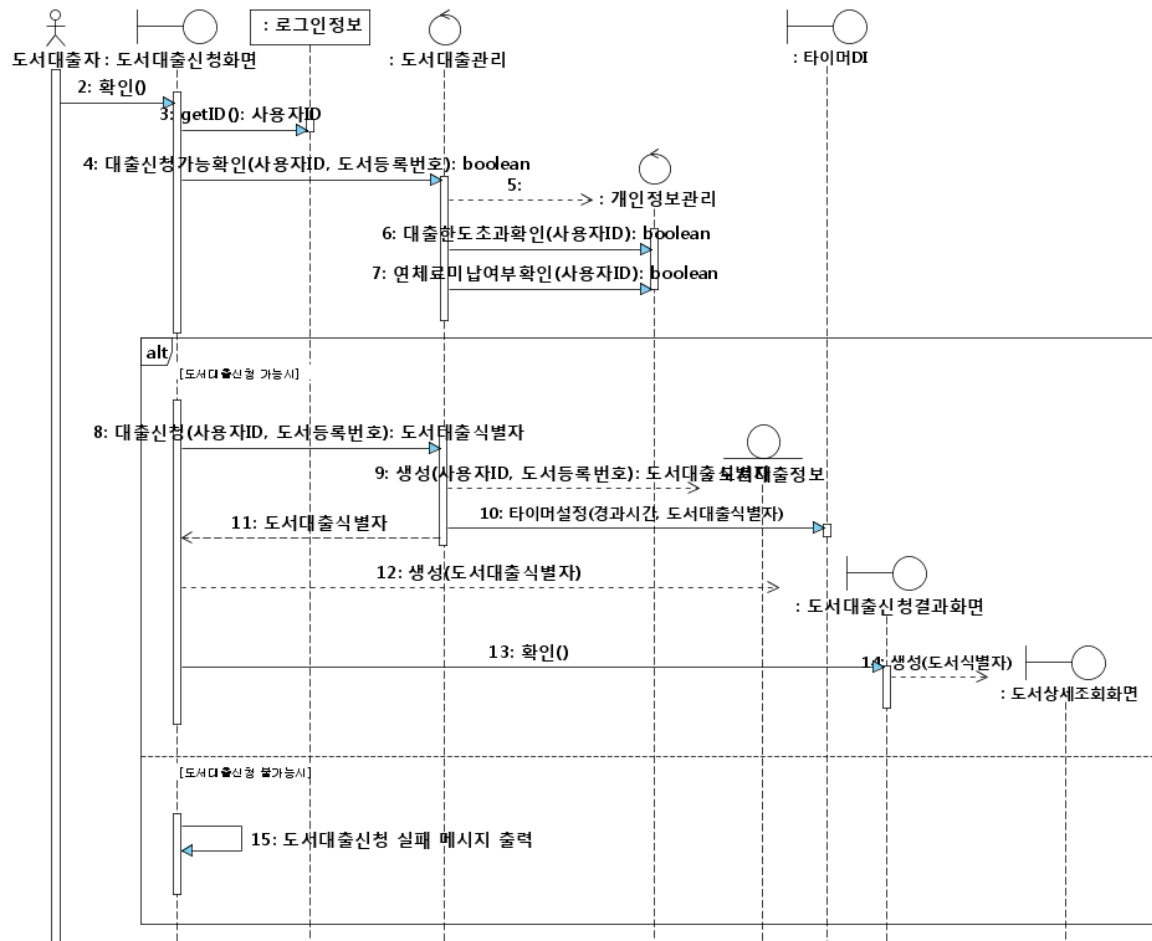
- 활동 다이어그램의 활용
  - 유스케이스 간의 선후관계 표현



## 3-2. 애플리케이션 요구 사항 도출하기

### UML과 Use Case 다이어그램

- 시퀀스 다이어그램의 활용
- 분석 유스케이스 실현: 도서대출신청 유스케이스






## 3-2. 애플리케이션 요구 사항 도출하기

### 분석 클래스 다이어그램을 이용한 개념 분석

- 분석 클래스 검증

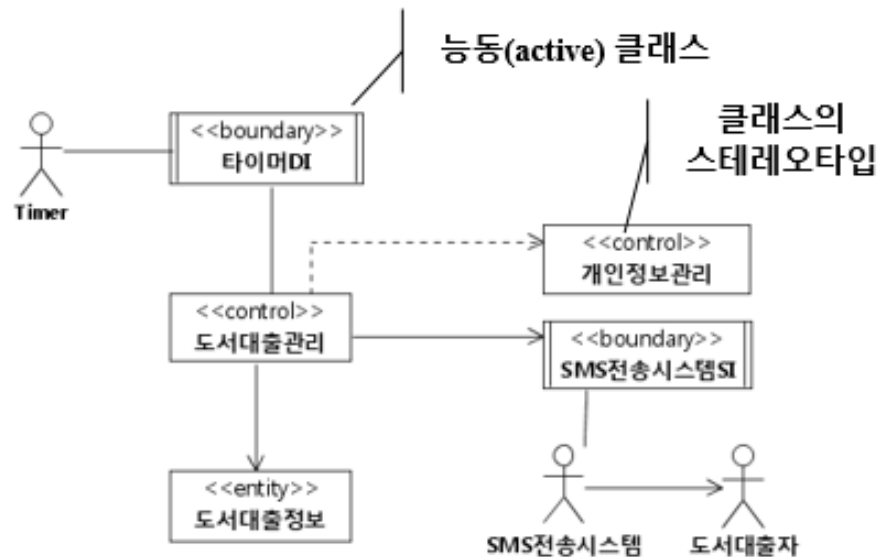
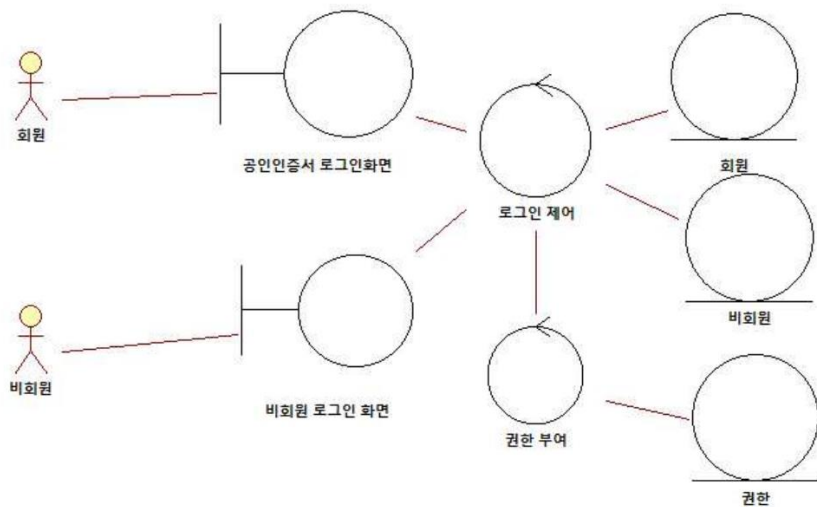
- 유스케이스마다 분석 클래스가 적절히 도출되었고, 제어 클래스의 도출 등이 충분하고 상세하게 도출되어 클래스의 역할, 클래스 간의 관계, 메시지 흐름 등을 확인할 수 있는지 검토한다.
- 스테레오 타입

역할구분	스테레오 타입	내용
경계 Boundary	<code>&lt;&lt;boundary&gt;&gt;</code> 	시스템과 외부 액터와의 상호작용을 담당하는 클래스
엔터티 Entity	<code>&lt;&lt;entity&gt;&gt;</code> 	시스템이 유지해야 하는 정보를 관리하는 기능을 전담하는 클래스
제어 Control	<code>&lt;&lt;control&gt;&gt;</code> 	시스템이 제공하는 기능의 로직 및 제어를 담당하는 클래스

## 3-2. 애플리케이션 요구 사항 도출하기

### 분석 클래스 다이어그램을 이용한 개념 분석

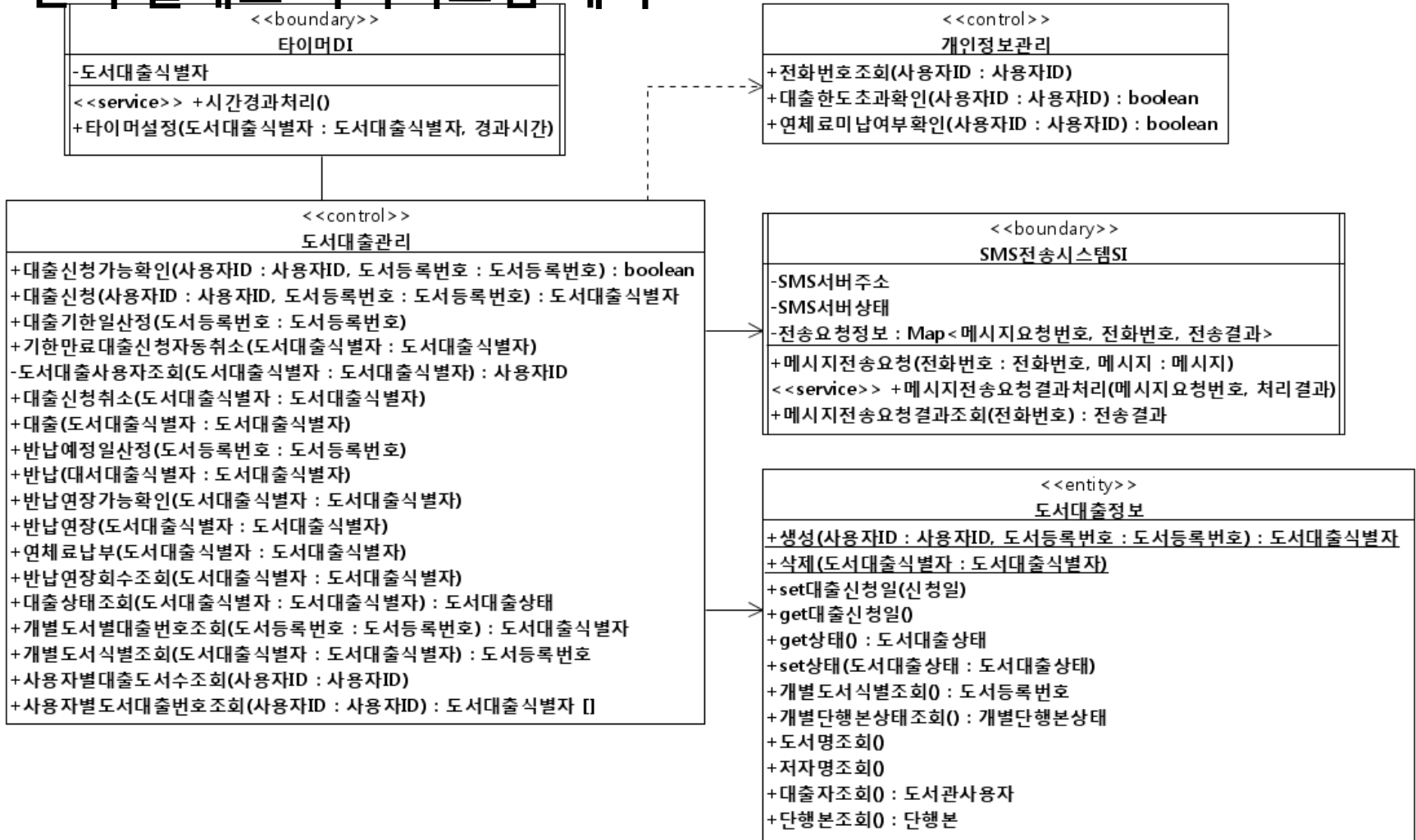
#### ● 분석 클래스 다이어그램 예시



## 3-2. 애플리케이션 요구 사항 도출하기

### 분석 클래스 다이어그램을 이용한 개념 분석

#### ● 분석 클래스 다이어그램 예시



Unit A

# 참고자료



## 문헌

1. <http://www.ncs.go.kr>
2. AgileSW개발101,nipa
3. 방법론 가이드,SW중소기업을 위한 경량 개발 방법론, 과학기술정보통신부, nipa
4. 김치수(2015). 『쉽게 배우는 소프트웨어 공학』. 한빛아카데미(주)
5. 김향곤, 소프트웨어개발 방법론, 대구 카톨릭대학 • 기술표준원(2013. 11)
6. 프로젝트 관리기준(ISO 21500) 이행가이드
7. 과학기술정보통신부 보도자료(2016. 9. 6)
8. 상용 소프트웨어 현재와 미래 조망
9. 보건복지부(2013).
10. 지식경제부(2010.2.26.). 소프트웨어 사업대가의 기준.
11. 정보통신기술진흥센터(2016. 9. 6). 글로벌 상용 소프트웨어 백서.
12. 특허청(2014). 소프트웨어개발 방법론.
13. 한국소프트웨어산업협회(2016). 소프트웨어 사업 대가 산정 가이드.
14. CMMI (CMMI(Capability Maturity Model Integration) 모델
15. ISO/ IEC 12207 모델 (소프트웨어 개발 생명주기 프로세스)
16. ISO/ IEC 14598 품질 평가 프로세스 • SPICE (ISO 15504) 모델 (프로세스 평가 표준)
17. 린 소프트웨어 개발의 적용
18. 속도 경쟁에서 승리하기
19. 메리 포펜딕, 톰 포펜딕 공저 / 엄위상, 심우곤, 한주영 공역 | 위키북스 | 2007년 08월 31일 | 원제 : Implementing Lean Software Development: From Concept To Cash
20. T 아카데미
21. <https://www.lesstif.com/jira/>
22. 제프랭어, 앤디 헌트, 데이브 토마스, 유동환 역,자바와 JUnit을 활용한 실용주의 단위 테스트,길벗

감사합니다.

❖ Mobile: 010-9591-1401

❖ E-mail: [onlooker2zip@naver.com](mailto:onlooker2zip@naver.com) / [dkkim@lklab.org](mailto:dkkim@lklab.org)

