OOAD 개요

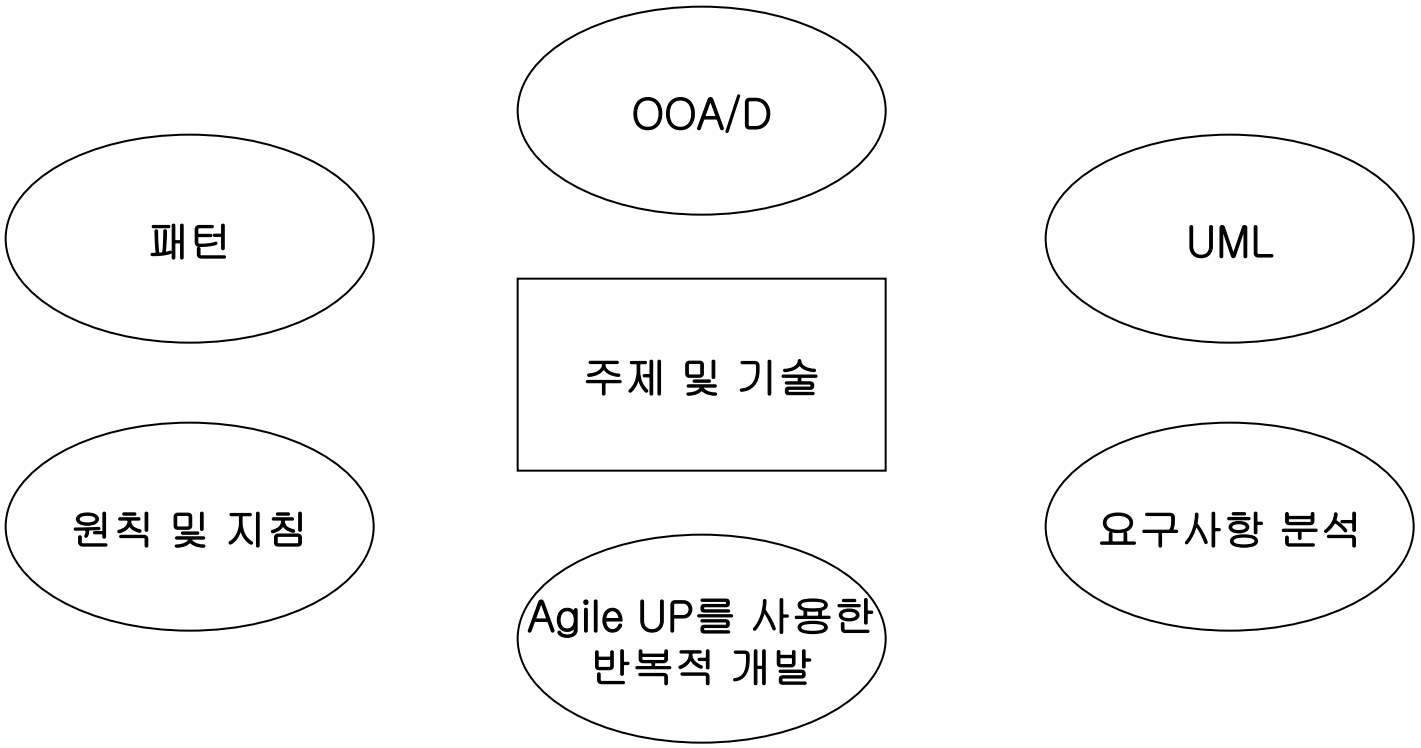
Objectives

- 분석과 설계의 구분
- OOA/D의 정의
- 간단한 예시

학습 내용

- UML vs. 객체지향적으로 생각하기
- 객체지향 설계 : 원칙 및 패턴
- 사례연구
- 유스케이스
- 반복적 개발,Agile 모델링, Agile UP
- 기타 다른 기술

Applying UML and Pattern in OOA/D



"망치를 가졌다고 건축가가 되는 것은 아니다"

가장 중요한 학습 목표

- 중요하고 기본적인 객체지향설계 분석의 목적은 소프트웨어 객체에 “책임”을 기술적으로 할당하는 것

- 객체를 설계하고 책임을 할당하는 9가지 원칙: GRASP

분석 및 설계란?

- 분석은 문제 및 요구사항들을 조사하는데 초점을 두는 것이며 그 해결책에 초점을 두는 것이 아님.
- What!

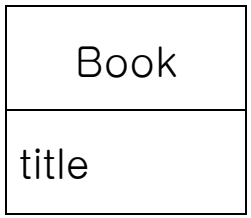
- 설계는 요구사항들을 달성하기 위한 개념적인 해결책을 강조하며, 그 설계에 대한 구현을 강조하는 것이 아님.
- How!

객체지향 분석 및 설계란

도메인 개념

문제영역의 객체와 개념을 포착

도메인 개념의 시각화



소프트웨어 객체들을 기술하고 요구사항을 수행하기 위해 그 객체들이 어떻게 협력하는 지를 정의

객체지향 프로그래밍 언어로 표현

```
public class Book
{
private String title;

public Chapter getChapter(int) {...}
}
```

설계 객체를 구현

다른 예제

domain concept

representation in an object-oriented programming language

Plane
tailNumber

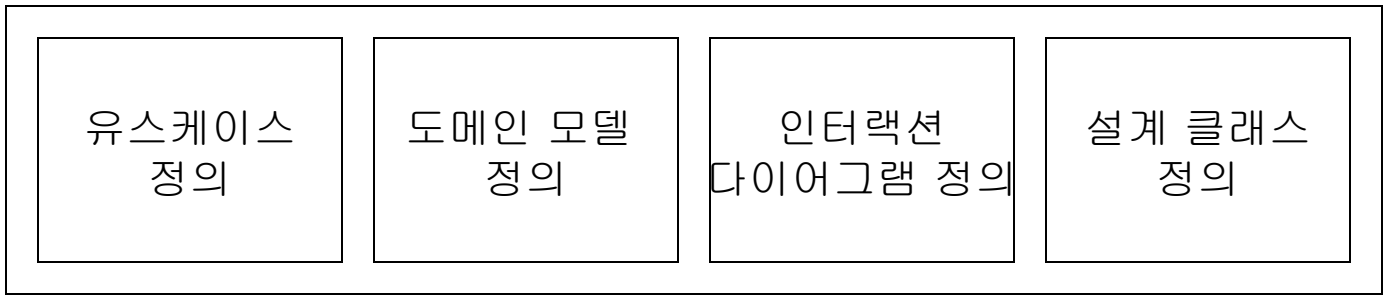
visualization of domain concept

```
public class Plane
{
private String tailNumber;

public List getFlightHistory() {...}
}
```

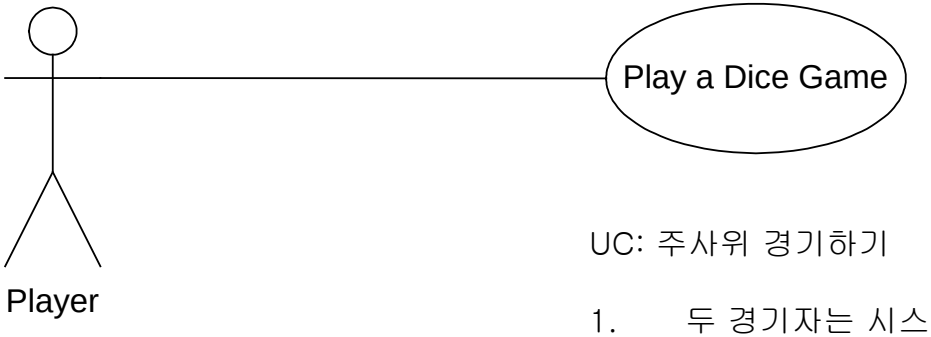
간단한 예제

- 주사위 게임 시뮬레이션 분석 설계
  - 두 사람의 경기자는 2개의 주사위를 굴려 두 수를 얻도록 요청한다.
  - 시스템은 두 사람의 결과를 비교하여, 다음의 규칙으로 승패를 결정하여 출력한다.
    - 두 수가 같으며 높은 숫자일 경우 승리
    - 두 수의 합이 더 큰 경우 승리
  - A birds-eye view of OOA/D



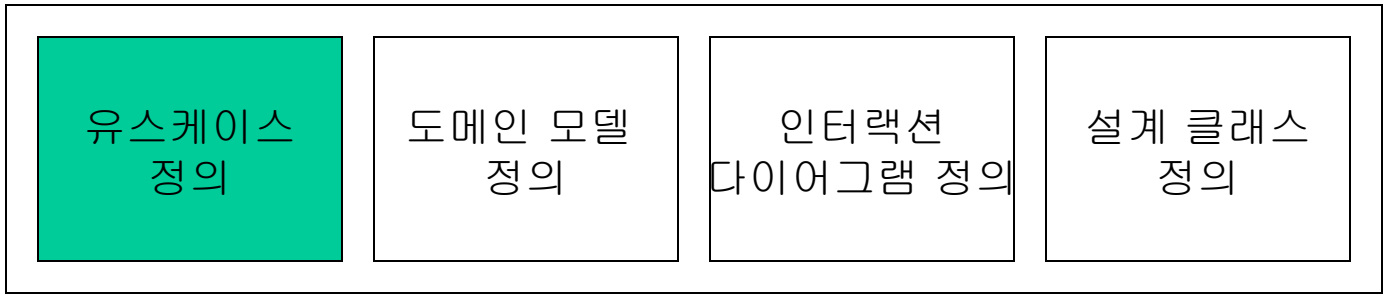
사용사례 작성 - 요구분석

- 요구분석은 쓰임새들로 기술될 수 있는 관련영역의 프로세스를 기술하는 것을 포함.



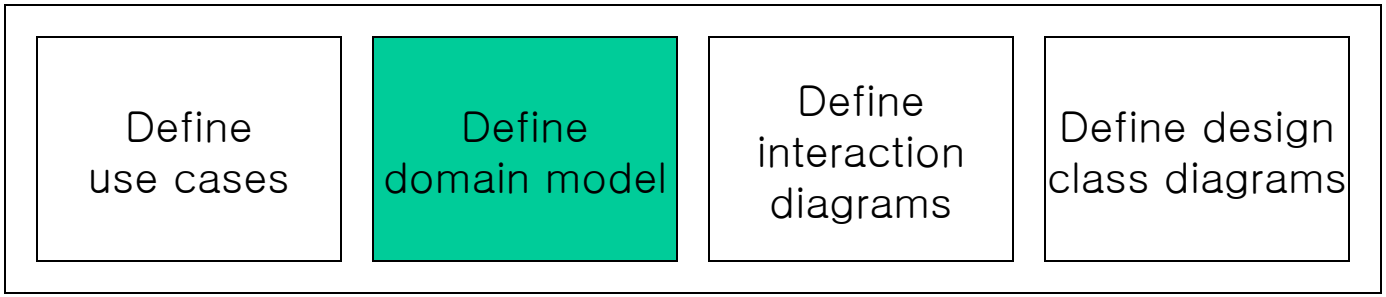
UC: 주사위 경기하기

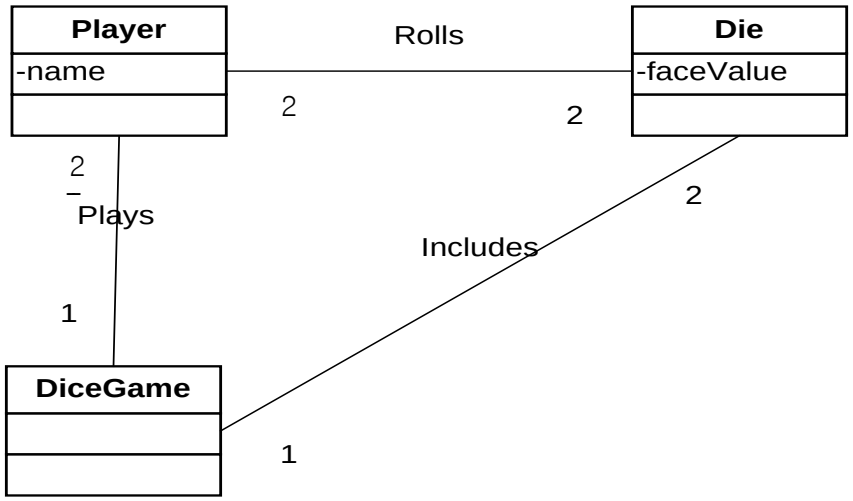
1. 두 경기자는 시스템에게 주사위 두 개를 던지도록 요구한다.
2. 시스템은 경기자에게 윗면의 값들을 보여주고 승패를 알려준다.
3. 시스템은 게이머들에게 다시 할 것인지를 묻고 1부터 반복하거나 종료한다.



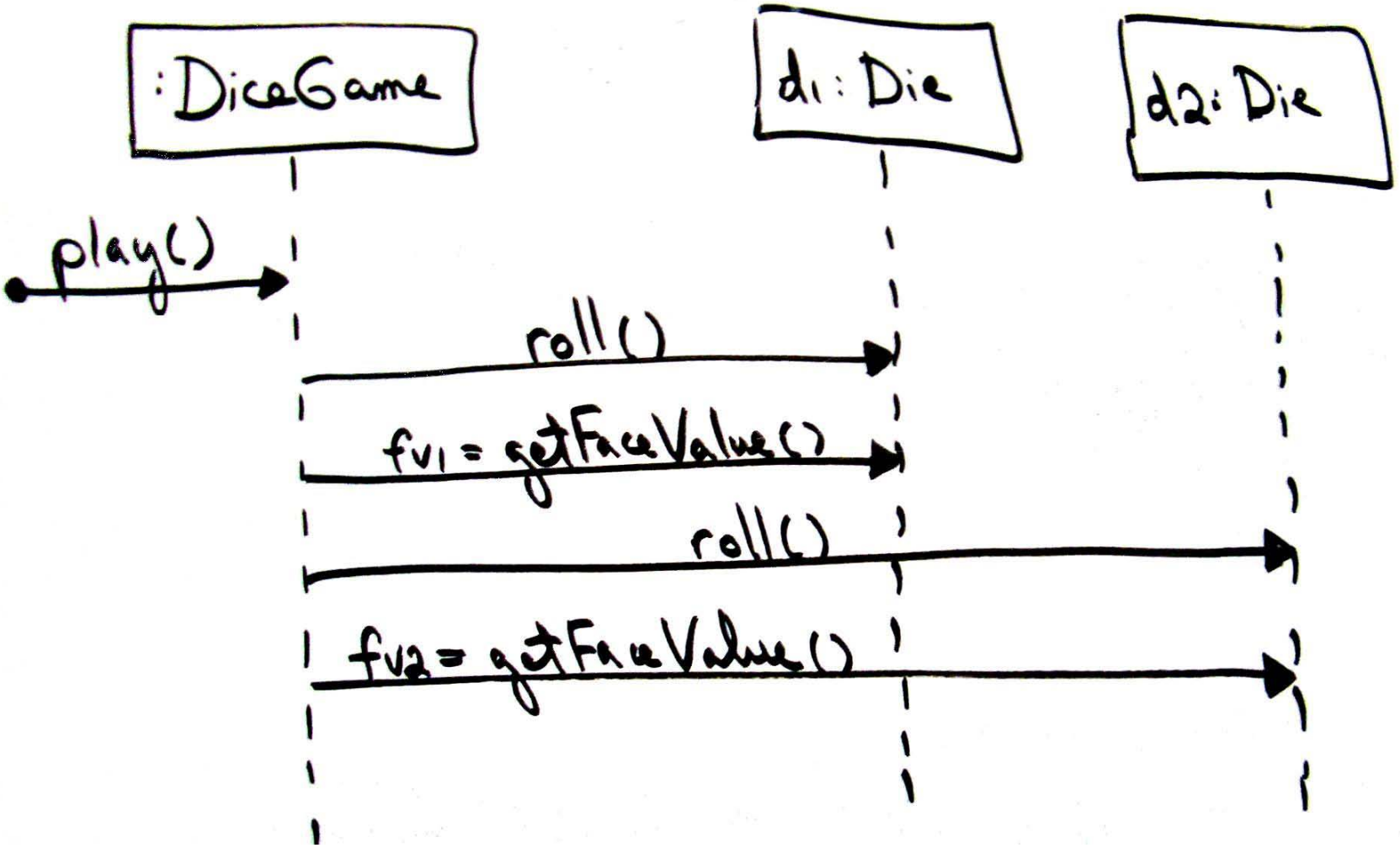
문제 영역 분석 및 표현

- 객체지향분석은 영역객체를 분류하는 관점에서 관심영역을 기술하는 것과 연관



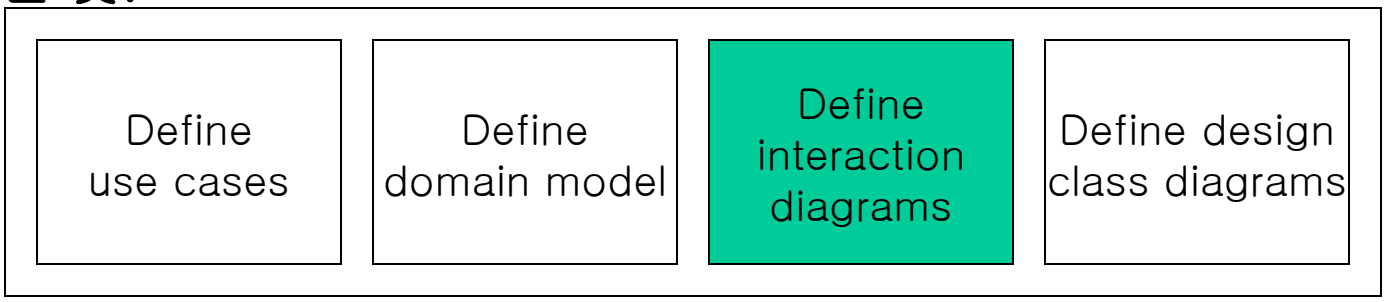


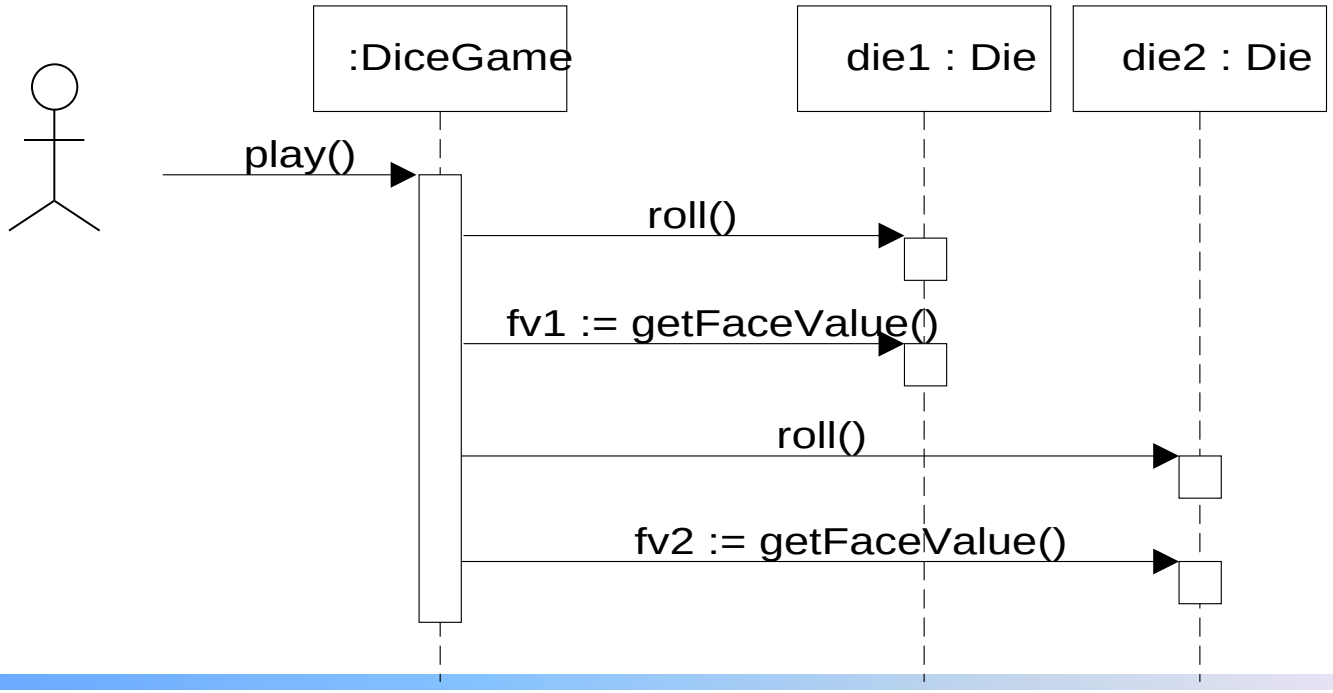
상호협력 관계 표현


:DiceGame
d1:Die
d2:Die
play()
roll()
fv1 = getFaceValue()
roll()
fv2 = getFaceValue()

Define Interaction Diagrams

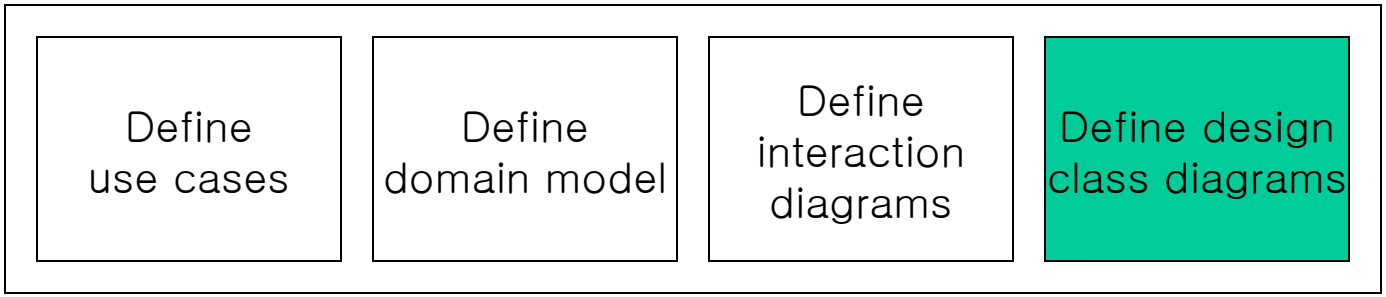
- 객체지향설계는 소프트웨어 객체들과 그들간의 협력관계를 정의하는 것.



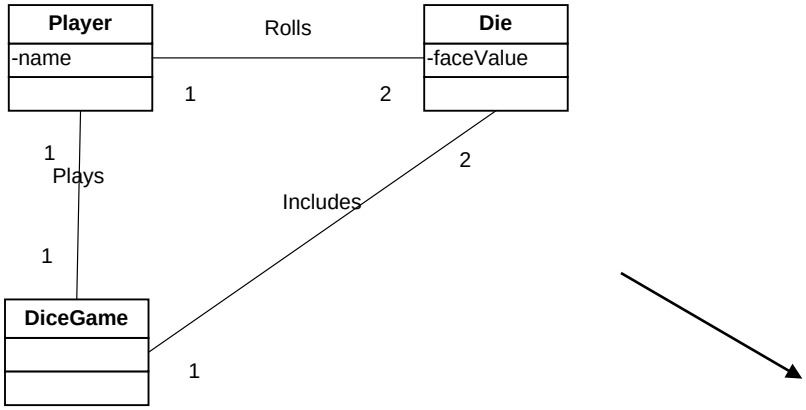


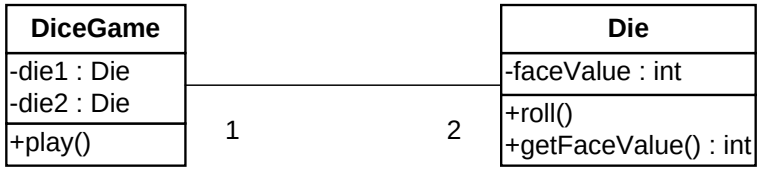
Define Design Class Diagrams

- Static view of the class definitions



도메인 모델 vs. 구현 모델





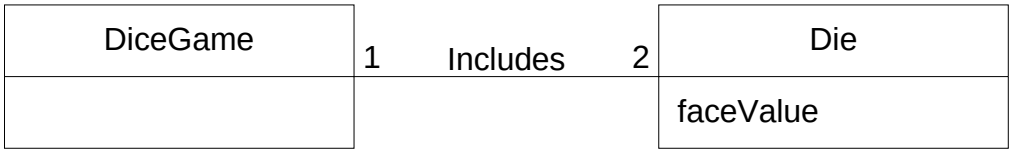
1.6 UML이란 무엇인가?

- OMG (Object Management Group) 에서 정의한 객체 표현 방법

- UML 적용하는 세 가지 방범
  - 스케치로서의 UML
  - 청사진으로서의 UML (코드의 표현, 설계의 표현)
  - 프로그래밍 언어로서의 UML : 코드 자동생성

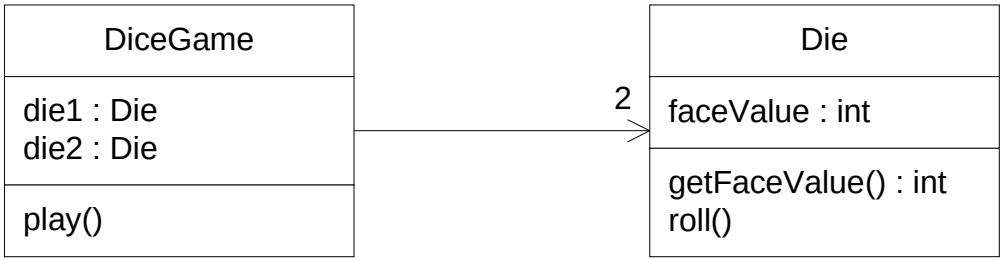
- UML 적용하는 세 가지 관점
  - 개념적 모델 관점
  - 명세(소프트웨어) 관점
  - 구현 관점

UML의 서로 다른 관점들



Conceptual Perspective (domain model)

Raw UML class diagram notation used to visualize real-world concepts.



Specification or Implementation Perspective (design class diagram)

Raw UML class diagram notation used to visualize software elements.

UML1과 UML2

- 1997년 UML 1.0 발표
- 2005년 UML 2.0 -> 시스템공학적 관점까지 확장
- OMG (Object Management Group) 의 객체 표현에 관한 세계 표준
- 현재에도 계속해서 발전 중
- OOA/D의 역사
  - 1960/1970: Simula, Smalltalk
  - 1980
    - Grady Booch: Object-Oriented Design
    - Ivar Jacobson
    - Jim Rumbaugh
  - 1997: UML1.0
  - http://www.omg.org
  - http://www.uml.org

Larman's Process

1. Use Cases
2. Conceptual Model
3. System Sequence Diagram
4. System Contracts
5. Collaboration Diagram
6. Class Diagram
7. Code

Homework#1

- UML 도구를 사용하여 주사위 게임에 쓰인 UML 도해를 그리고, 구현모델을 C++ 또는 Java 를 통해 구현하여라.
  - 단, 주사위의 roll() 함수에서 랜덤넘버를 생성할 때 rand() 함수를 사용하여라.

- Java 또는 C++ 프로그래밍을 위한 통합개발도구는 무엇을 사용하고 있는가?

- UML 그리기 도구는 어떤 것들이 있는지 조사하시오.

The UML

- The method war; Booch, OMT, OOSE, Fusion, Coad/Yourdon
- Standard at OMG 1997
- A standard for graphical notation, not method

Further Reading

- Summary of UML
  - UML Distilled, by Martin Fowler
- Introduction to UP
  - The Rational Unified Process, by Philippe Kruchten
- Detailed discussion of UML
  - The Unified Modeling Language Reference Manual and The Unified Modeling Language User Guide, by Three Amigo
- Current version of UML
  - www.omg.org
  - www.celigent.com/uml
- Software pattern
  - Design Pattern, by Gamma, Helm, Johnson, and Vlissides

Case Study: POS

The NextGen POS System

- 판매시점관리(Point-Of-Sale) 시스템

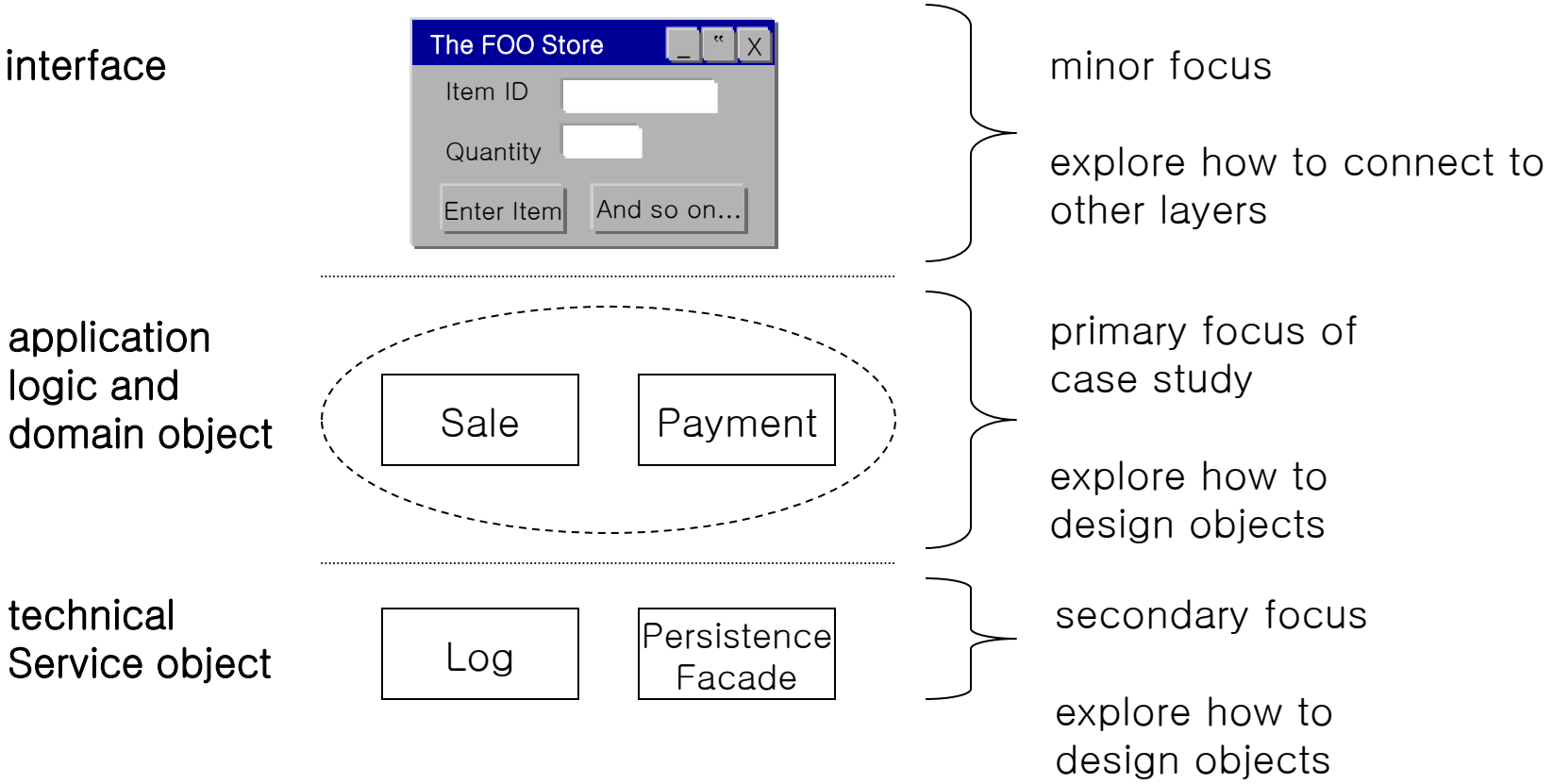
•POS 시스템은 주로 소매점에서 사용되는 매출등록과 결재처리를 위한 애플리케이션이다. 시스템은 컴퓨터나 바코드 스캐너와 같은 하드웨어 부품과 시스템에서 동작하는 소프트웨어로 구성된다. 시스템은 제3자가 제공하는 세금계산기나 재고관리 애플리케이션 같은 다양한 서비스 애플리케이션과 인터페이스 한다. 이 시스템은 비교적 고장으로부터 안전해야 한다. 즉, 재고관리 애플리케이션 같은 원격서비스가 일시적으로 사용 불능이더라도, 업무가 중단되어서는 안 된다.

•POS 시스템은 점차적으로 다양한 클라이언트 측의 터미날과 인터페이스를 지원해야 한다. 여기에는 신-클라이언트 웹 브라우져, 자바 Swing 그래픽 사용자 인터페이스를 사용하는 일반적인 개인용 컴퓨터, 터치 스크린, 무선 PDA 등이 포함된다.

•더욱이, 우리는 상업적인 목적으로 POS 시스템을 만들고 있다. 우리는 시스템을 각기 다른 업무규칙과 요구사항을 갖고 있는 고객에게 판매할 것이다. 그러므로, 우리는 유연성과 변형성을 제공해주는 매커니즘이 필요할 것이다.

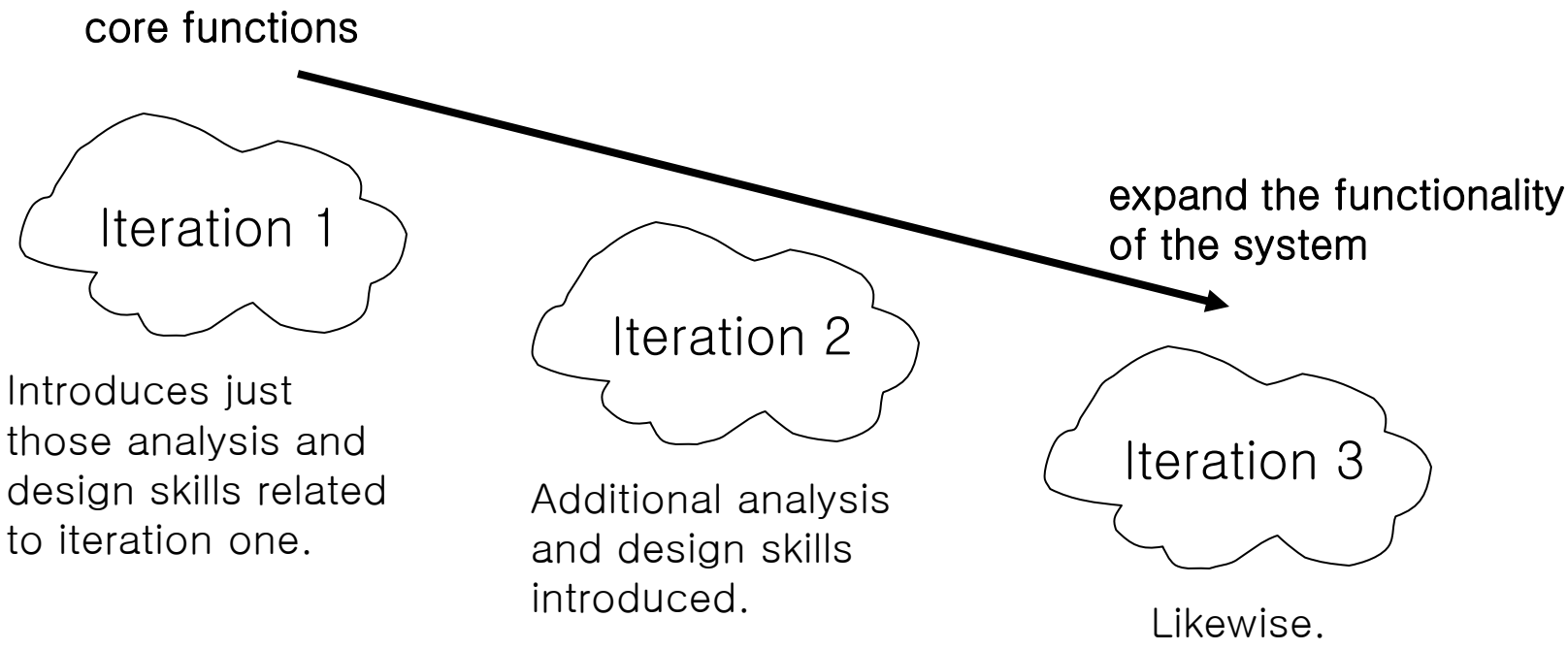
•우리는 반복적인 개발 전략에 따라, 요구사항, 객체지향 분석과 설계, 구현을 수행하게 될 것이다.

Architectural Layers and Case Study Emphasis

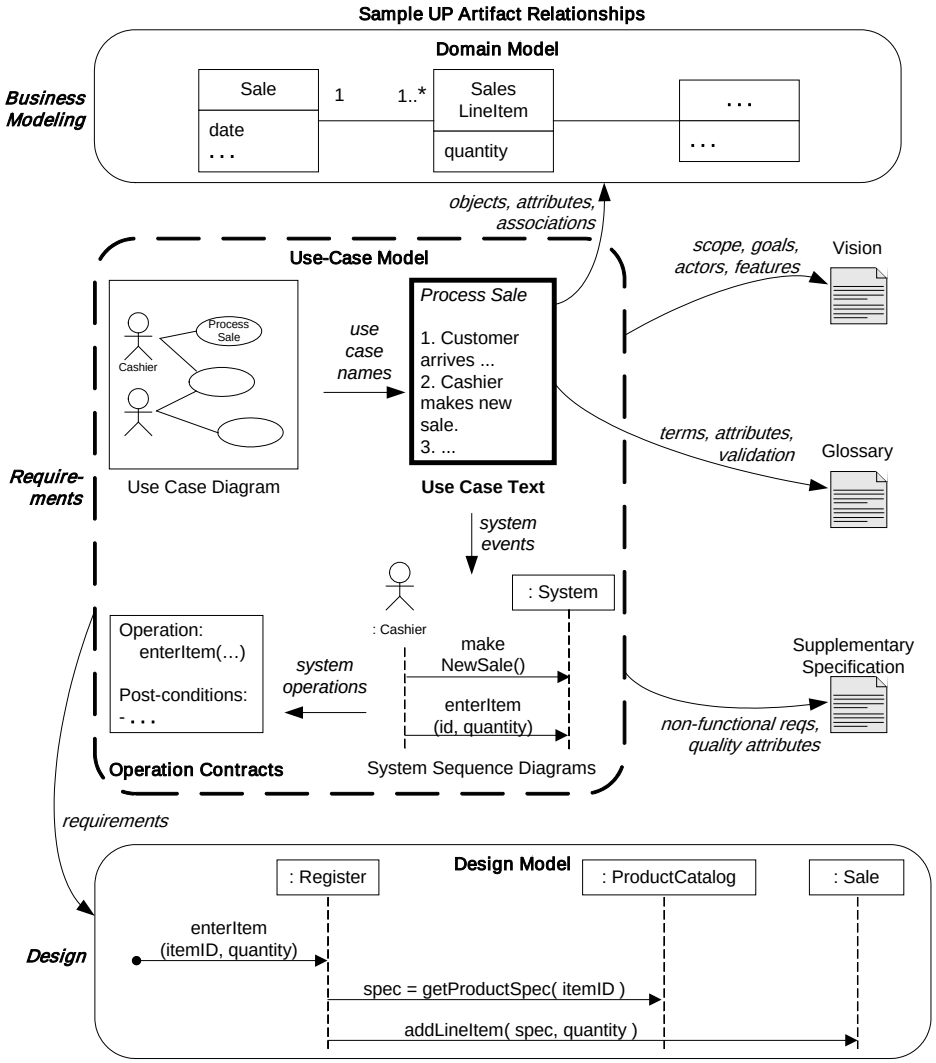


The Book's Strategy

- Iterative learning and development



Agile UP


Sample UP Artifact Relationships
Domain Model
Business Modeling
Sale
date
...
1
1..*
Sales LineItem
quantity
...
...
objects, attributes, associations
Use-Case Model
scope, goals, actors, features
Vision
Process Sale
Cashier
use case names
Process Sale
1. Customer arrives ...
2. Cashier makes new sale.
3. ...
terms, attributes, validation
Glossary
Require-ments
Use Case Diagram
Use Case Text
system events
: System
: Cashier
Operation:
enterItem(...)
Post-conditions:
- ...
system operations
make NewSale()
enterItem (id, quantity)
Supplementary Specification
non-functional reqs, quality attributes
Operation Contracts
System Sequence Diagrams
requirements
Design Model
: Register
: ProductCatalog
: Sale
Design
enterItem (itemID, quantity)
spec = getProductSpec(itemID)
addLineItem(spec, quantity)

객체 설계로 전이
(Transition to Object Design)

Objectives

- 동적/정적 객체 설계 모델링을 이해한다.
- 다이어그램을 그리기 위해 Agile 모델링 또는 UML CASE 도구를 사용해 본다.

소개

- 일반적인 객체 설계 방법
  - 1. 코드(code): 코딩하면서 동시에 설계하기 (자바, C#), 리팩토링 (refactoring)을 같이 하면 이상적. 정신적인 모델에서 코드로.
  - 2. 그리고 난 후 코드화하기(draw, then code) : UML 다이어그램을 화이트보드나 UML CASE 도구를 이용하여 그림. 그리고, 이를 개발도구 (Eclipse,Visual Studio 등) 텍스트 코드로 작성한다.
  - 3. 그림만 그리기(only draw): CASE 도구는 다이어그램만 그려주면 코드를 생성한다.
- ‘그린 후 코딩’ 방법이 가장 적절
  - 현재로선 가능한 간단하게 그린 후 코드로 바꾸는 것이 Agile 방법론에서 사용됨

1. Agile 모델링과 간단하게 UML 그리기

- Agile 모델링
  - 다이어그램을 그리는 오버헤드를 가급적 줄이면서 문서화보다는 의사소통을 위해 모델링한다!
  - 도구
    - 화이트보드 또는 흰색 접착용지
    - 마커 펜, 디지털 카메라, 프린터
  - 민첩한 모델링은 다음을 포함
    - 다른 사람과 함께 모델링한다.
    - 여러 모델을 병렬로 생성한다. 즉, 교류도를 5분간 생성하고 다른 벽면에는 이를 반영한 클래스도를 5분간 그린다.
  - XP Agile 방법론
    - 흰색 접착 용지 (또는 화이트 보드 활용)
    - 이를 디지털카메라로 찍어 웹(예:wiki (www.wiki.org))에 업로드한다.

2. UML CASE 도구

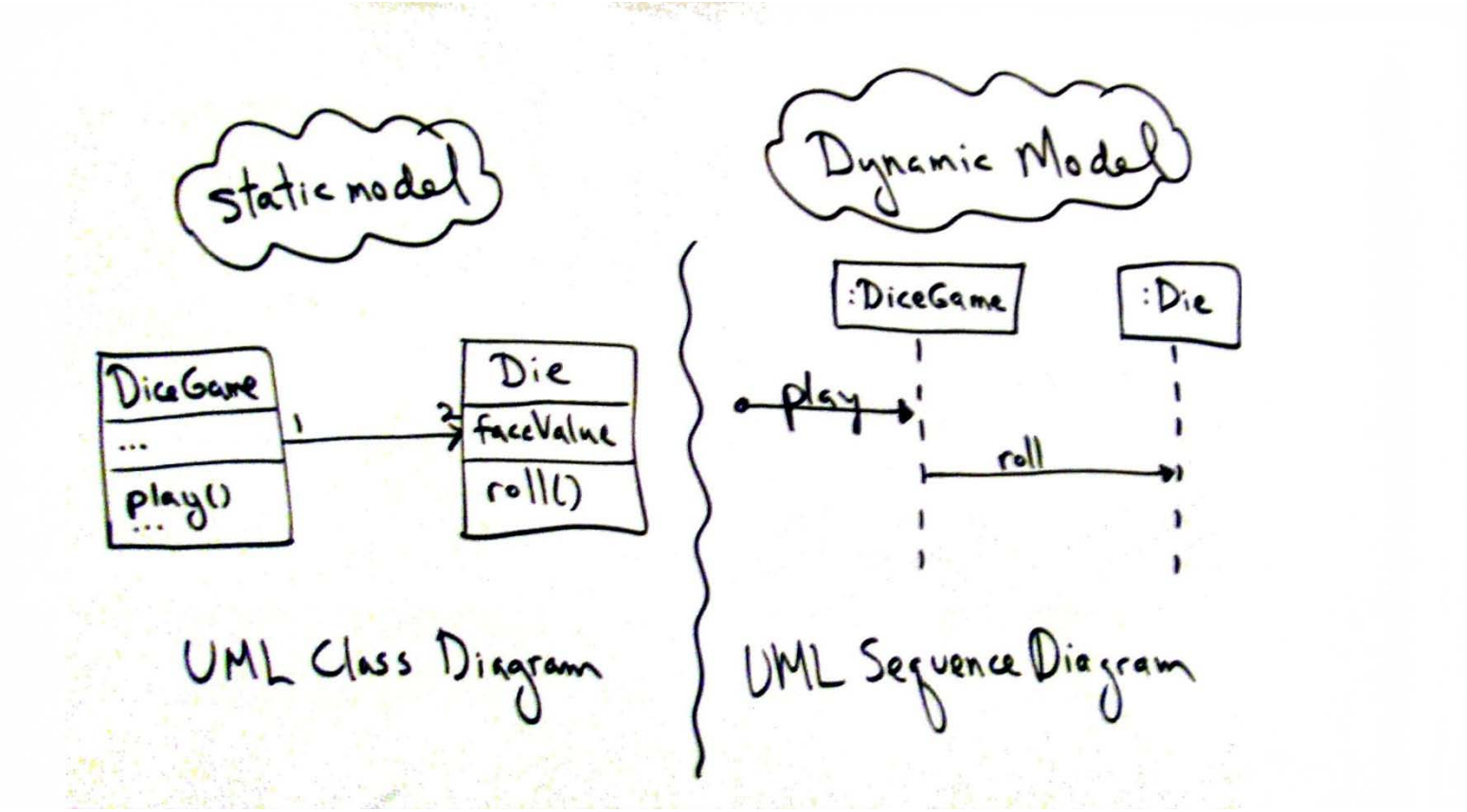
- CASE (Computer Aided Software Engineering) 도구도 매우 유용
- 무료에서부터 수천만원짜리까지
- 선택 가이드라인
  - Eclipse나 Visual Studio 같은 대중적인 텍스트 기반 IDE와 같이 사용할 수 있는 (연동되는) UML CASE 도구를 선택하라.
  - 클래스 다이어그램 뿐만 아니라 인터랙션 다이어그램까지 역공학을 할 수 있는 UML 도구를 선택하라.
- 대부분,
  - IDE에서 잠깐 코딩하고 이를 역공학 버튼을 눌러 UML 그림을 살펴봄.
  - 벽을 사용하는 Agile 방법과 도구를 사용하는 방법론은 상호보완적

3. 코딩 전에 UML을 그리는데 소요되는 시간

- 가이드라인
  - 3주 정도의 타임박스를 가진 반복개발일 경우, 초기 객체 설계 중 어렵고 창조성이 필요한 부분을 위해 UML로 벽에 그리거나 CASE도구를 이용해 몇 시간에서 길어도 하루 정도의 시간을 보내라.
  - 만약 스케치를 한다면 디지털 카메라를 사용하여 사진을 찍고 인쇄하며, 만들어진 UML 다이어그램을 사용하여 남은 기간에 코딩으로 전환해본다. 하지만 코드의 마지막 설계에는 개선될 수 있음을 인정한다.
- 늘어나는 클래스들...
  - 코딩된 것 것을 역공학하여 인쇄한 다음 계속 설계한다.

4. 객체 설계하기: 정적 모델링과 동적 모델링이란?

- 정적 모델: 클래스도 – 객체의 속성과 연산, 객체 사이의 연관
- 동적 모델: 순차도/협력도 – 객체들간의 상호작용 (호출순서)



동적 객체 모델링

- 필요한 객체, 객체간의 상호작용 식별
- 가이드라인
  - 클래스도에 집중하기보다는 인터랙션 다이어그램을 위해 시간을 많이 보낸다.
  - 이 가이드라인을 무시하는 것은 매우 흔하지만 “좋지 않는 실행기술”이다.
- 책임주도 설계 (RDD: Responsibility-Driven Design) 와 GRASP (General Guideline for Responsibility Assignment Software Pattern) 원칙
- UML
  - 순차도 (협력도) : 순차도, 협력도 == 인터랙션 다이어그램
  - 상태도 (State Machine Diagram)
  - 활동도 (Activity Diagram)

정적 객체 모델링

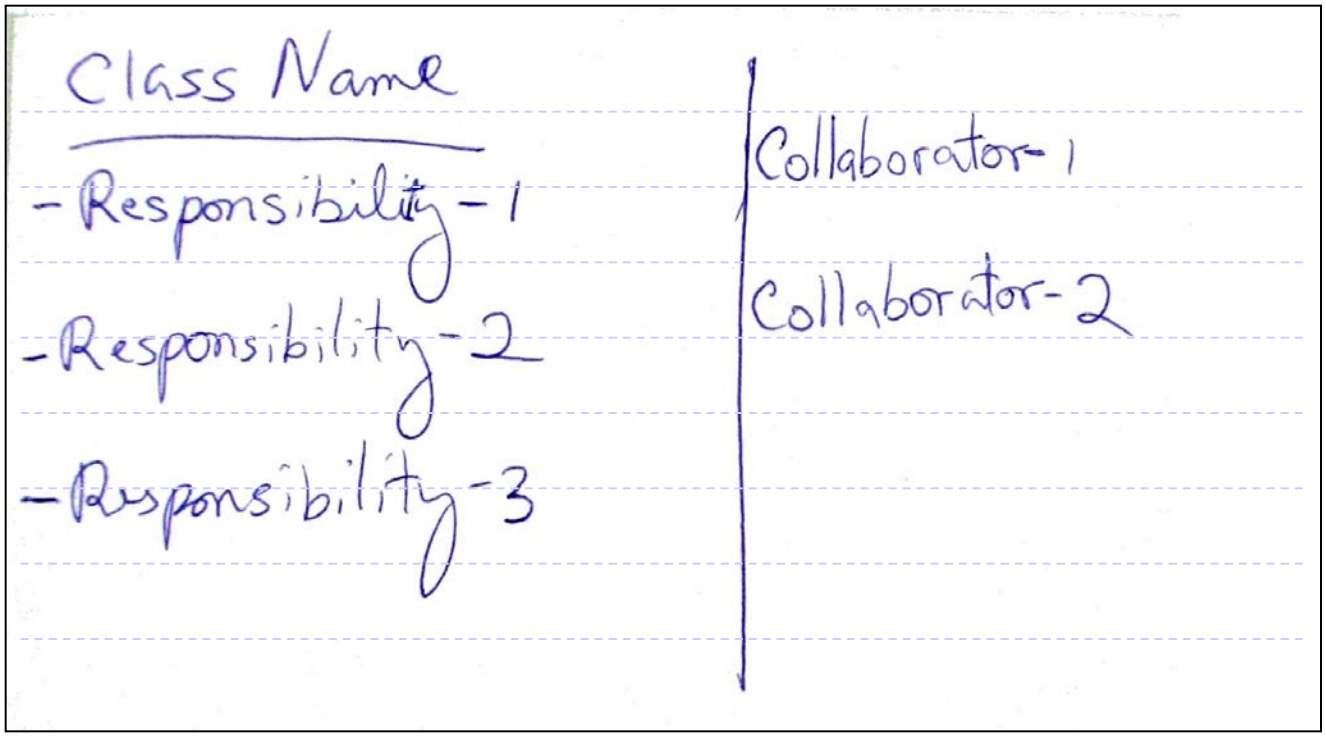
- UML 클래스도 사용
  - 먼저 동적 모델을 한 후에 클래스도 작성
  - Agile 방법론: UML 클래스도와 인터랙션 다이어그램을 병렬로 작성
- 패키지도, 배치도를 병행해서 사용
  - 패키지도: 클래스를 군으로 분류
  - 배치도: 물리적인 위치를 표시

5. UML표기 기술과 비교한 객체 설계 기술의 중요성

- 객체 설계 기술 vs. UML 표기 기술
  - UML 다이어그램은 해당 설계에 대한 의사결정 내용이 반영된다.
  - 객체설계 기술은 UML 다이어그램을 어떻게 그리는가를 아는 것이 아니라, 무엇을 그리는가에 대한 것을 아는 것이다.
  - 근본적인 객체 설계는 다음과 같은 지식이 요구된다.
    - 책임할당의 원칙
    - 디자인 패턴

6. 기타 객체 설계 기법: CRC 카드

- Class – Responsibility – Collaboration 카드
- Memo Card를 사용
- Roll Playing



CRC 카드의 네 가지 작성 예