객체지향소프트웨어공학

5주차 : UML 기반 시스템 분석 설계

I-01. UML과 방법론 소개

1. 왜 모델링이 필요한가?
2. 비주얼 모델링(Visual Modeling)
3. UML의 역사
4. UML 소개
5. 아키텍처 명세 기법
6. 개발 프로세스(Process)
   - 구조적 방법론과 폭포수 모델(Waterfall Model)
   - UP(Unified Process) 방법론

1장 목차

1. 왜 모델링이 필요한가?
2. 비주얼 모델링(Visual Modeling)
3. UML의 역사
4. UML 소개
5. 아키텍처 명세 기법
6. 개발 프로세스(Process)
   - 구조적 방법론과 폭포수 모델(Waterfall Model)
   - UP(Unified Process)

왜 모델링이 필요한가?

- 시스템의 규모가 확대되면서 다양한 모델링 필요
- 모델이란?
  - “실체를 정형적으로 정리한 대상체 ”
- 모델링은 왜 필요할까?
  - 개발하고자 하는 시스템을 상호 간에 더욱더 잘 이해하기 위함
- UML을 지원하는 도구의 이용 목적
  - 프로젝트를 추진하는 사람이나 조직마다 추구하는 바가 다름
    - 비용 절감 중점
    - 효율성에 근간한 시간의 절감 추구
    - 시스템의 무결성 추구

왜 모델링이 필요한가?

- 모델링 목적
  - 현상태의 시스템이나 우리가 원하는 형태로 시스템을 비주얼화
  - 시스템의 구조나 행위 명세화
  - 시스템을 구축하는 안내가 될 기본 틀 제공
  - 결정 사항에 대한 문서화

비주얼 모델링(Visual Modeling)

- 비주얼 모델링
  - 실세계(Real space)의 관념들을 둘러싼 것들로 구성된 모델을 이용하여 문제를 가시화함으로써 이해하는 방법
- 모델
  - 매우 복잡한 문제와 구조의 중요하지 않은 부분을 걸러내고 핵심 부분만을 묘사하여 문제를 보다 쉽게 이해할 수 있도록 하는 추상화(抽象畵)
- '추상'의 의미
  - 특별한 이야기나 의미를 지니지 않고, 선, 색, 형태 등의 그림의 구성 요소로만 어떤 생각이나 감정을 표현한 것

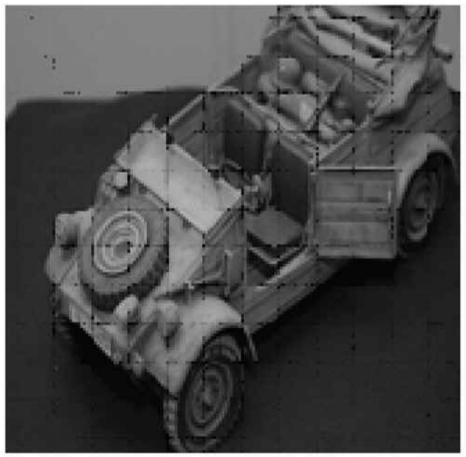
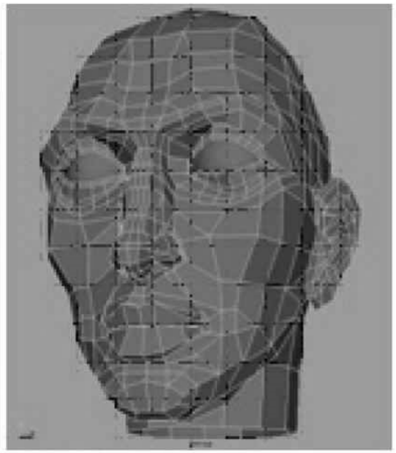
비주얼 모델링(Visual Modeling)

- 전산학에서 추상화
  - 실체의 대상을 구성하고 있는 요소들로부터 특징을 추출하여 단순화 시키는 과정
  - 객체지향 세계
    - 어떠한 객체(사물)가 있다면 그것을 관찰해서 필요한 정보만 추출
    - 객체의 속성과 객체의 동작 포함
    - 정보를 추출하여 하나의 논리적인 객체를 만들고 이것을 다시 추상화하여 사용자 정의 형태의 클래스 만듦
  - 복잡한 시스템을 만들기 위해, 개발자는 반드시 시스템을 다른 측면에서 바라본 것을 추상화해야 하고, 이것을 정확한 표기법으로 모델을 만들어야 함
  - 모델이 시스템 요구사항을 만족하는지 검증 필요
  - 모델을 실제로 구현하기 위해 점진적으로 자세한 부분 추가

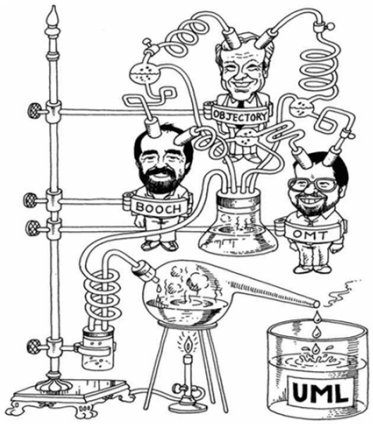
비주얼 모델링(Visual Modeling)

- 모델링의 4가지 기본 원칙
  - 첫째, 생성하고자 하는 모델의 선택은 문제 영역을 어떻게 공략할 것인가와 해결안을 어떻게 형상화할 것인가에 대해 큰 영향
  - 둘째, 각각의 모델들은 다른 수준(level)의 정밀도에 따라 표현
  - 셋째, 최상의 모델은 실체와 연결
  - 넷째, 단일 모델만으로는 부족

- 다양한 형태의 모델


UML의 역사



- UML 삼총사
  - 제임스 럼바(James Rumbaugh) : OMT
  - 그래디 부치(Grady Booch) : Booch method
  - 이바 야콥슨(Ivar Jacobson) : OOSE

- 제임스 럼바 : OMT(Object Modeling Technique)
  - 제임스 럼바가 근무했던 GE(General Electronic)에서 개발된 방법론
  - 객체 모델링(Object Modeling)
    - 시스템에서 필요한 객체 찾아내고 객체 속성과 객체 간의 관계 규명
  - 동적 모델링(Dynamic Modeling)
    - 객체 모델링에서 나타낸 객체의 행위와 상태를 포함하는 수명주기를 나타냄
  - 기능 모델링(Functional Modeling)
    - 각 객체 변화로 인해 다른 상태로 전이되었을 때 수행되는 동작 기술

UML의 역사

- 그래디 부치 : 부치 방법론(Booch method)
  - 시스템은 몇 개의 뷰(View)로 분석된다고 생각하여 뷰를 모델 다이어그램으로 표현
  - 거시적(Macro) 개발 프로세스와 미시적(Micro) 개발 프로세스 모두 포함
  - 단계적 접근과 자동화 도구 지원
  - 분석보다는 설계 쪽에 더 많은 중점
  - 객체지향 방법론에 대한 광범위한 이론적 배경 제시

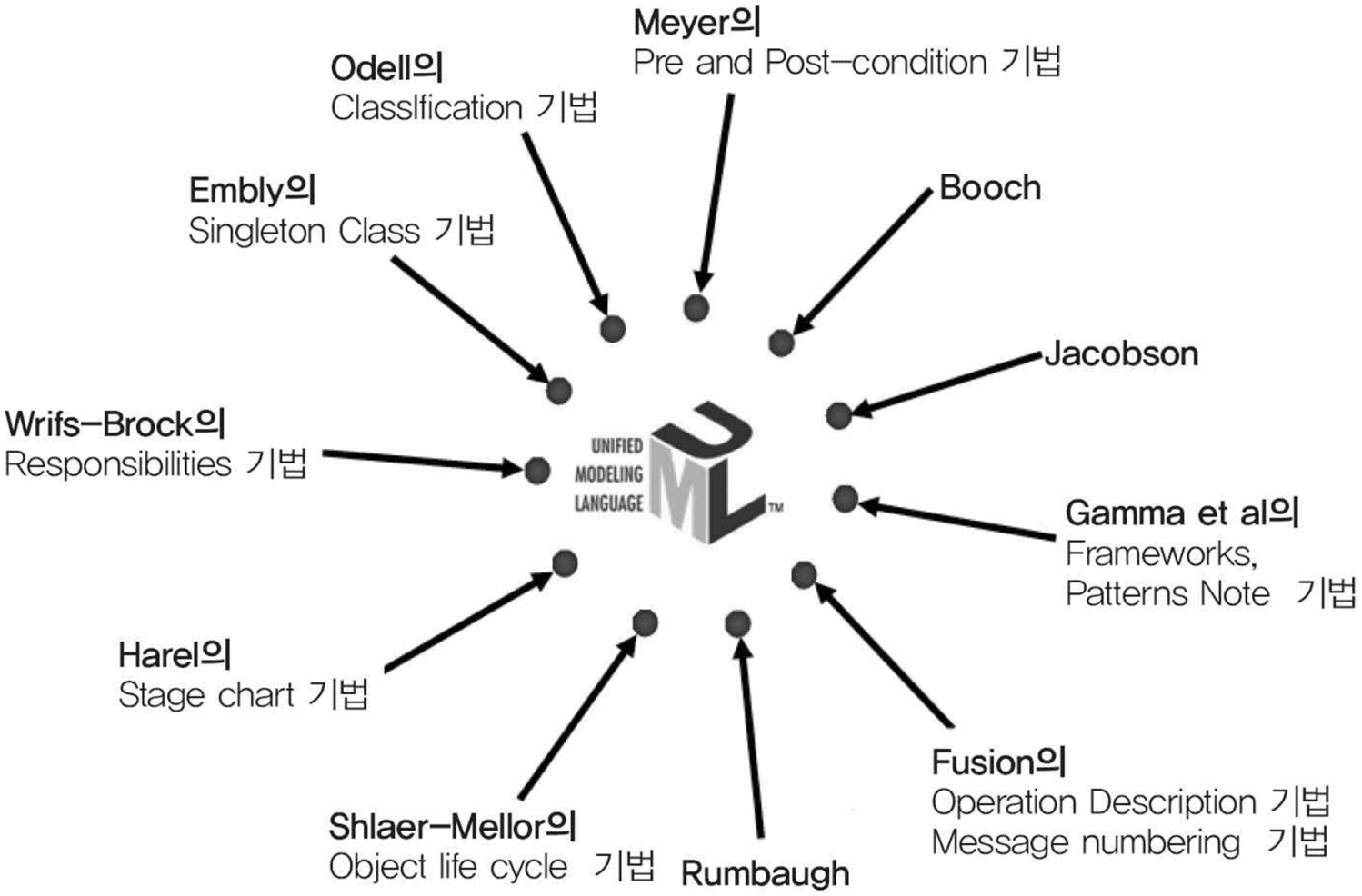
- 이바 야콥슨 : OOSE(Object-Oriented Software Engineering)
  - 유스케이스(Use case) 강조
  - 유스케이스
    - 외부 액터(External actor)와 상호작용하는 시스템 요구사항 정의
    - 개발, 시스템 테스트, 시스템 검증 단계에 사용
  - 방법론이 복잡하고 내적인 연관성이 강해서 초보자에게는 어려움
  - 대규모 시스템 개발에 적합

UML의 역사

- UML(Unified Modeling Language)
  - 표기법에 관련한 방법론 전쟁을 끝맺게 함
  - 객체지향 시스템의 개발 중에 생기는 산출물들을 명세화, 시각화, 문서화에 사용되는 언어
  - Booch, OMT, Objectory 표기법뿐만 아니라 여러 방법론 학자들이 주장했던 최고 개념 간의 통합 의미
  - 사용자 경험에 기반을 둔 객체지향 분석 설계 분야에 사실상의 표준 제공
  - 분석과 설계의 산출물을 표준화하려는 시도로써, 의미론적 모델, 문장론적인 표기법과 다이어그램 등 표준화

UML의 역사

- UML(Unified Modeling Language) 통합된 표기법



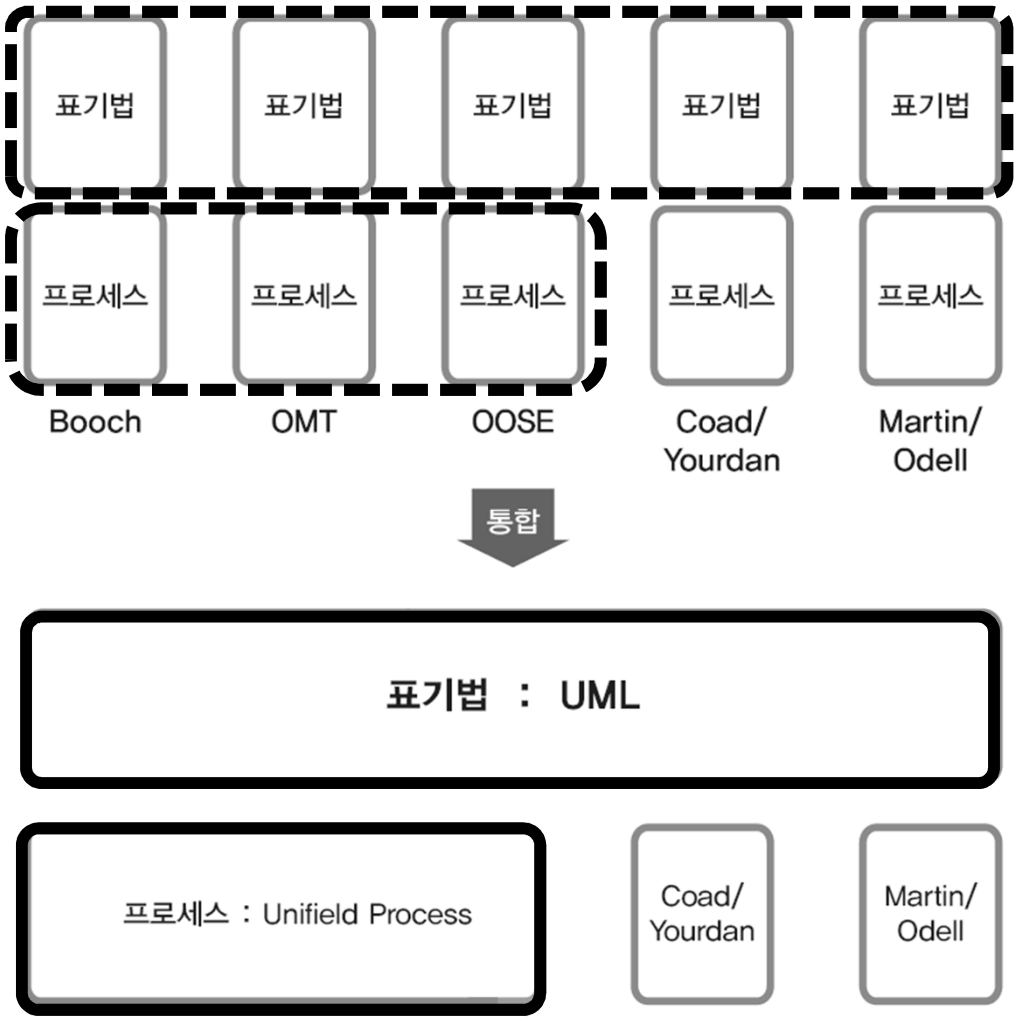
UML 소개

- UML
  - 소프트웨어 중심 시스템과 관련한 산출물을 가시화하고, 명세화하며, 시스템 구축 및 문서화에 사용
  - 시스템 모델링에 적합
  - 기업 IT 시스템, 분산처리의 웹 기반 애플리케이션과 실시간 임베디드 시스템(Real-time Embedded system)까지 표현 가능
  - 표현력이 매우 풍부한 언어로써, 시스템들을 개발하고 업무에 배치하는 데 필요한 모든 관점 다룸

- UML을 효과적으로 익히기
  - UML 구성요소
  - 작성 방법 및 규칙
  - 공통 메커니즘

UML 소개

- 표기법(Notation)과 프로세스(Process)의 통합



- UML 특징
  - 가시화
  - 명세화
  - 구축
  - 문서화

- 가시화
  - 바로 코딩을 하는 것에 있어서의 문제점
    - 작성된 개념 모델을 다른 사람에게 전달 시 오류가 생기기 쉬움
      - 참여한 모든 사람이 같은 언어를 사용하지 않기 때문
    - 소프트웨어 시스템에는 이해할 수 없는 것들이 있어서 문자 위주의 프로그래밍 언어로는 모델을 만드는데 한계
    - 만일 개발자가 모델을 전혀 기록하지 않았다면, 그 정보는 영원히 없어질 것이고, 그 개발자가 전직하고 나면 부분적으로만 다시 모델을 만들 수 있음

- 가시화(계속)
  - UML 모델 작성
    - 모델로 의사 소통을 용이하게 하자는 것
    - UML은 그래픽 언어
  - UML로 모델 작성
    - 다른 개발자 또는 개발도구까지도 명백하게 그 모델에 대한 해석 가능
- 명세화
  - 정확하고, 명백하며, 완전한 모델을 만드는 것
  - 특히 UML은 분석 및 설계와 구현 시점의 모든 중요한 결정에 대한 명세서를 다룸
  - 이러한 결정들은 소프트웨어 중심 시스템을 개발하고 배치할 때 정함

- 구축
  - UML은 시각적인 모델링 언어!
    - 모델들은 다양한 프로그래밍 언어와 직접 연결 가능
  - 그래픽으로 가장 잘 표현되는 것은 UML로 처리
  - 문자 위주로 가장 잘 표현되는 것은 프로그래밍 언어로 표현
  - 순공학(Forward Engineering) 지원
    - UML 모델로부터 대상 프로그래밍 언어의 코드를 생성하는 것
  - 역공학(Reverse Engineering) 지원
    - 구현된 코드를 분석하여 UML 모델 생성
  - 왕복공학(Round-trip Engineering)
    - 그래픽과 문자 중 어느 하나의 판단으로 일을 할 수 있음을 의미
    - 도구는 두 가지 관점을 일치시켜 봄

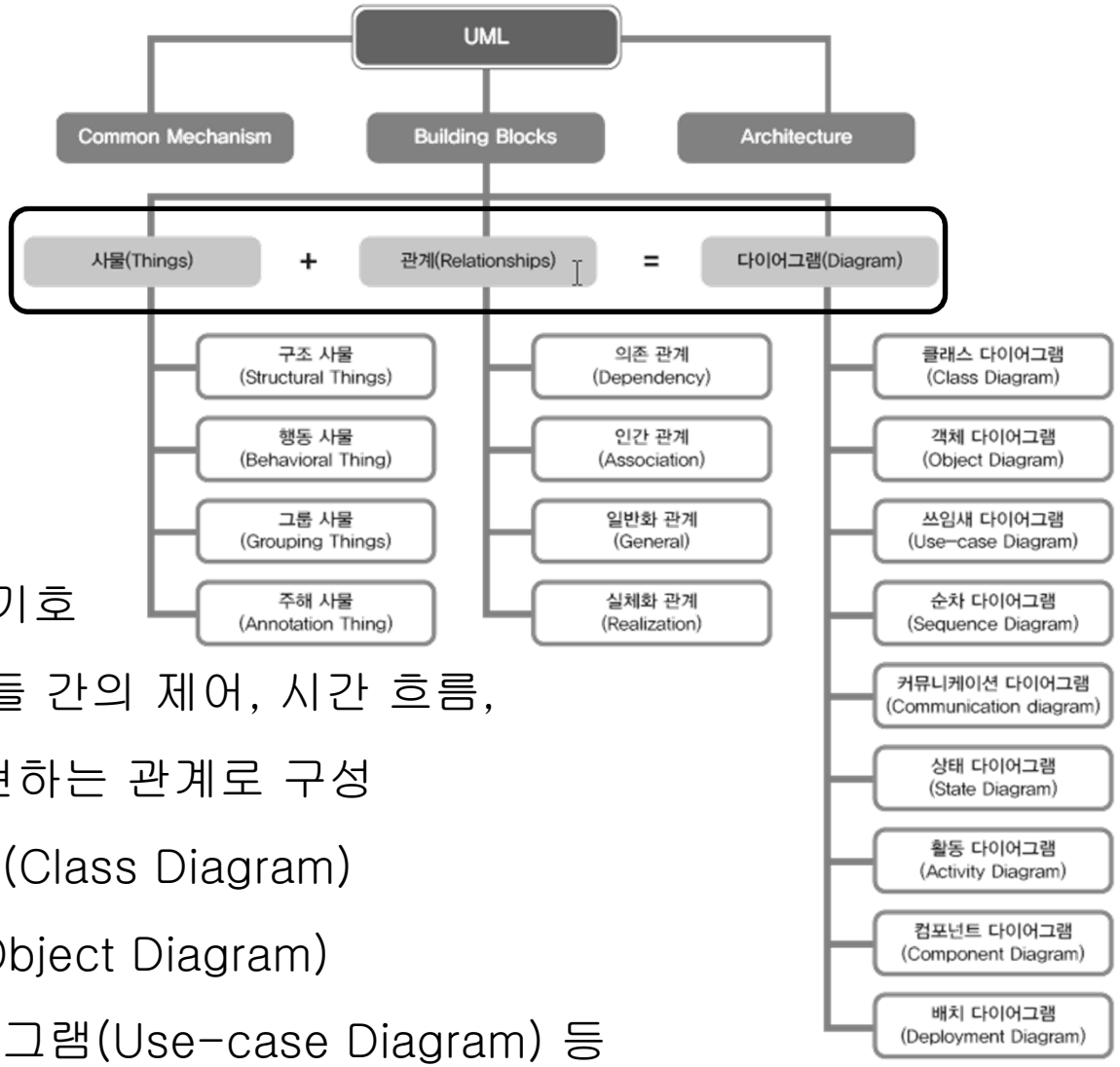
- 문서화
  - 산출물의 포함 요소
    - 아키텍처 설계
    - 소스 코드
    - 프로젝트 계획 테스트
    - 프로토타입(Prototype : UP에서는 사용자 경험 모델이라 함)
    - 배포판(Releases)
  - 산출물
    - 프로젝트 산출물
    - 개발 도중이나 배치 후에 시스템에 대한 통제, 평가, 의사소통에 중요

UML 소개

- 문서화(계속)
  - UML
    - 시스템 아키텍처와 모든 상세 내역에 대한 문서화
    - 요구사항 표현
    - 시스템 시험 언어
    - 프로젝트 계획과 배포관리 활동의 모델링 언어
  - 모델링 언어
    - 소프트웨어 청사진을 위한 표준 언어

UML 소개

- 문서화(계속)
  - UML의 구성요소



  - 다이어그램 : 9개
    - 의미를 갖고 있는 기호
    - 사물(things)과 이들 간의 제어, 시간 흐름, 이벤트 흐름을 표현하는 관계로 구성
    - 클래스 다이어그램(Class Diagram)
    - 객체 다이어그램(Object Diagram)
    - 유스케이스 다이어그램(Use-case Diagram) 등

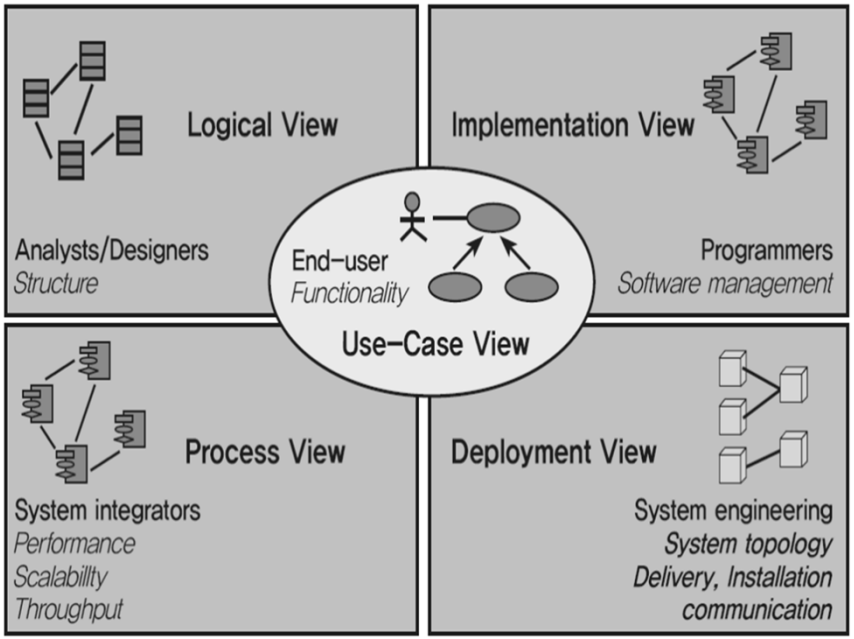
아키텍처 명세 기법

- 시스템 아키텍처
  - 가장 중요한 산출물
  - 프로젝트 생명주기에 걸쳐 반복적이고 점진적인 시스템 개발을 제어가능 해야 함

- 아키텍처의 중요한 결정 사항
  - 소프트웨어 시스템의 구성
  - 시스템을 구성하는 구조 요소들과 그것들 간의 인터페이스 선택
  - 요소들 간의 협력으로 명세화되는 행동
  - 점진적으로 더 큰 서브 시스템으로 구조 요소와 행동 요소 결합
  - 이러한 구성을 좌우하는 아키텍처 양식, 예컨대 정적 및 동적 요소들과 그것이 갖는 인터페이스와 그것들이 갖는 협력과 결합

아키텍처 명세 기법

- 소프트웨어 아키텍처
  - 구조와 행동에 연관
  - 용도, 기능성, 성능, 탄력성, 재사용성, 경제성 및 기술적 제약과 타협
  - 미학적인 것도 관련



- UML 이용한 소프트웨어 중심 시스템의 아키텍처
  - 상호 연결된 5개의 뷰
  - 뷰(View)
    - 시스템 조직과 구조 투영
    - 시스템의 특정한 관점에 초점
  - UML의 아키텍처 : 4+1뷰

아키텍처 명세 기법

- 유스케이스 뷰(Use-case View)
  - 유스케이스로 구성
  - 유스케이스는 최종 사용자, 분석가, 테스트 담당자가 보게 되는 시스템의 행동 설명
  - 소프트웨어 시스템의 구성을 명세화하기보다는 시스템 아키텍처를 구체화하는 요인들을 명세화
  - UML을 이용
    - 정적인 관점 : 유스케이스 다이어그램으로 포착
    - 동적인 관점 : 시퀀스 다이어그램(Sequence Diagram)과 커뮤니케이션 다이어그램 (Communication Diagram1)으로 대표되는 인터액션 다이어그램(Interaction Diagram), 상태 다이어그램(State Diagram)과 액티비티 다이어그램(Activity Diagram)으로 포착
  - 사용자 요구사항을 정의하는데 많은 공헌

아키텍처 명세 기법

- 논리 뷰(Logical View)
  - 문제 영역과 해법의 어휘를 형성하고 있는 클래스, 인터페이스, 협력으로 되어 있음
  - 주로 시스템의 기능적인 요구사항 지원 → 시스템이 최종 사용자에게 해주어야 하는 서비스 의미
  - UML 이용
    - 정적인 관점 : 클래스 다이어그램, 오브젝트 다이어그램으로 포착
    - 동적인 관점 : 커뮤니케이션 다이어그램, 스테이트 다이어그램, 액티비티 다이어그램으로 포착

- 프로세스 뷰(Process View)
  - 시스템의 동시성과 동기화 메커니즘을 형성하고 있는 쓰레드(thread)와 프로세스로 구성
  - 주로 시스템의 성능, 신축성, 처리능력 다룸
  - UML 이용
    - 정적 및 동적인 관점을 설계 뷰의 경우와 같은 종류의 다어어그램으로 포착
    - 쓰레드와 프로세스를 나타내는 액티브 클래스에 초점

아키텍처 명세 기법

- 구현 뷰(Implementation View)
  - 물리적인 시스템을 조립하고 배포하는데 사용되는 컴포넌트와 파일들로 구성
  - 주로 시스템 배포판의 형상관리 다룸
  - 배포판
    - 다소 독립적인 컴포넌트와 파일들로 구성
    - 실행 시스템의 제작을 위해서 다양한 방법으로 조립 가능
  - UML 이용
    - 정적인 관점 : 컴포넌트 다이어그램으로 포착
    - 동적인 관점 : 콜래보래이션 다이어그램, 스테이트 다이어그램, 액티비티 다이어그램으로 포착

아키텍처 명세 기법

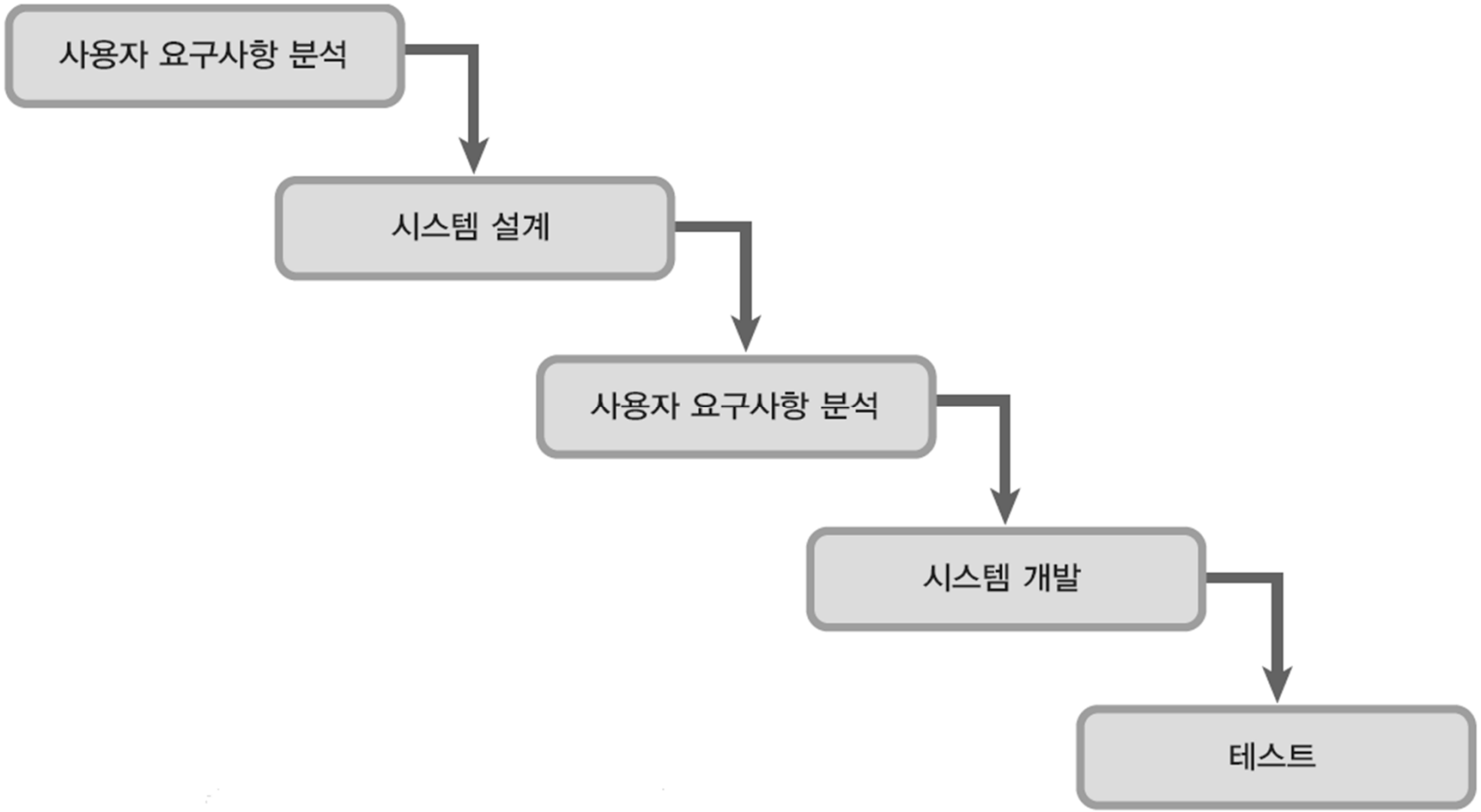
- 배치 뷰(Deployment View)
  - 시스템이 실행하는 하드웨어 형태를 형성하고 있는 노드(node)로 구성
  - 주로 물리적인 시스템을 구성하고 있는 각 부분들의 분산 형태와 설치 다룸
  - UML 이용
    - 정적인 관점 : UML을 이용하여 디폴로이먼트 다이어그램(Deployment Diagram)으로 포착
    - 동적인 관점 : 커뮤니케이션 다이어그램, 스테이트 다이어그램, 액티비티 다이어그램으로 포착

개발 프로세스(Process)

- 소프트웨어 개발 프로세스
  - 사용자 요구사항을 소프트웨어 시스템으로 구현하기 위한 일련의 활동
- 프로젝트 수행 시 적용할 수 있는 개발 프로세스
  - 폭포수형 모델부터 객체지향적인 프로세스에 이르기까지 여러 가지 존재
  - 각 프로세스는 장단점 모두 갖고 있음

구조적 방법론과 폭포수 모델(Waterfall Model)

- 구조적 방법론과 폭포수 모델(Waterfall Model) 개요



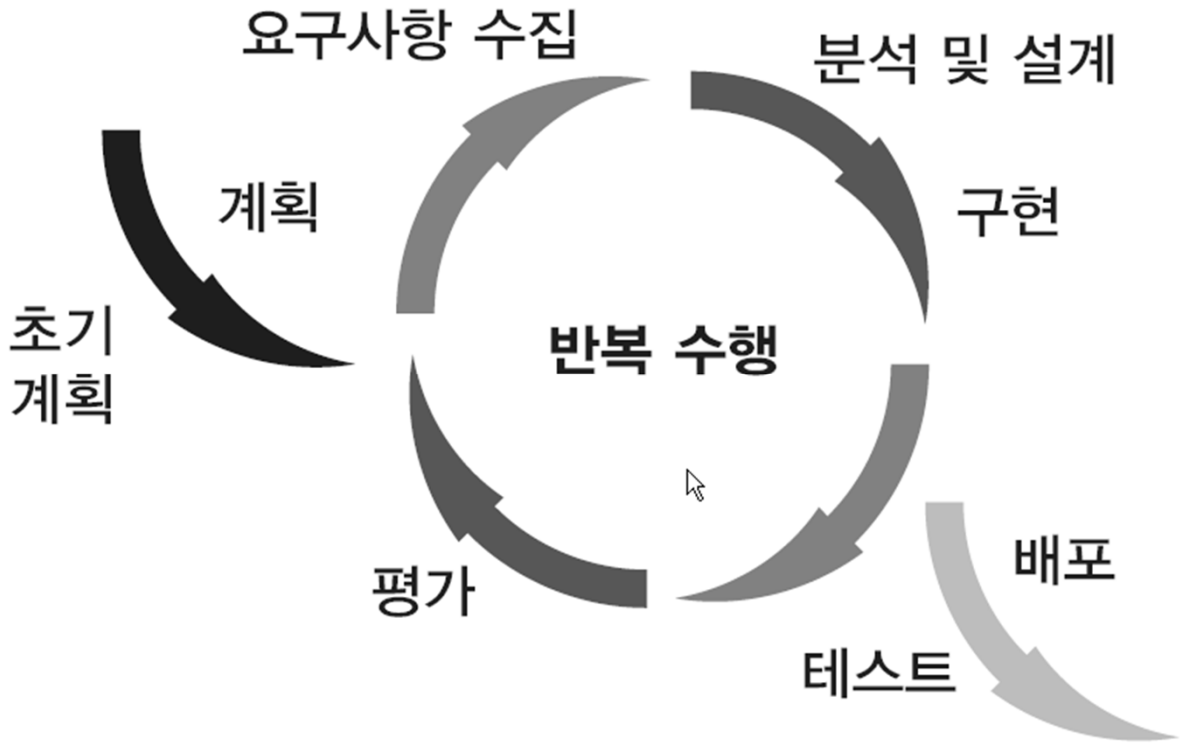
구조적 방법론과 폭포수 모델(Waterfall Model)

- 구조적 방법론과 폭포수 모델(Waterfall Model) 개요
  - 요구사항 분석 → 시스템 설계 → 시스템 개발 → 시스템 테스트→ 개발된 시스템 배포
- 구조적 방법론(Yourdon)
  - 업무 영역에 대하여 기능 중심으로 분해
  - 업무 프로세스를 행위 위주로 분석/설계
  - 개발 프로세스
    - 폭포수 모델 사용
  - 주요 산출물
    - DFD(Data Flow Diagram), 모듈 명세서(Module Specification)
  - 개발 하위 단계로부터 상위 단계로 거슬러 올라갈 수 없도록 설계
  - 수 많은 프로젝트에서 사용되어 정형화되고 검증된 방법론

구조적 방법론과 폭포수 모델(Waterfall Model)

- 폭포수 모델의 단점
  - 프로젝트를 진행할수록 단계를 거꾸로 거슬러 올라갈 필요성이 커진다는 점
  - 착수 단계
    - 요구사항 수집 관련 작업 이후 사용자들이 내용을 전부 읽을 수 없을 만큼 엄청난 분량의 요구사항 명세서 및 관련 산출물 작성
    - 사용자들로부터 이를 확인하는 서명 받음
  - 설계 단계
    - 시스템 아키텍처 결정 → 만약 이 작업을 진행하는 동안 새로운 문제가 발생하면 다시 사용자에게 가서 그 문제에 대해 이야기를 하고 새로운 요구사항 도출 → 다시 분석 단계로 되돌아 감
    - 몇 번의 앞뒤 단계로 왔다 갔다 하기를 반복한 후 개발 단계로 이동
  - 개발 단계
    - 코딩 시 일부 설계했던 내용이 구현하기에 불가능하다는 사실을 깨닫고 다시 설계 단계로 돌아가서 그 문제를 다시 검토

- 폭포수 모델의 단점(계속)
  - 테스트 단계
    - 요구사항 명세가 충분히 상세하지 않았을 뿐만 아니라 잘못 해석하고 있었다는 사실 알게 됨 → 분석 단계로 되돌아가 요구사항 다시 검토



UP(Unified Process)

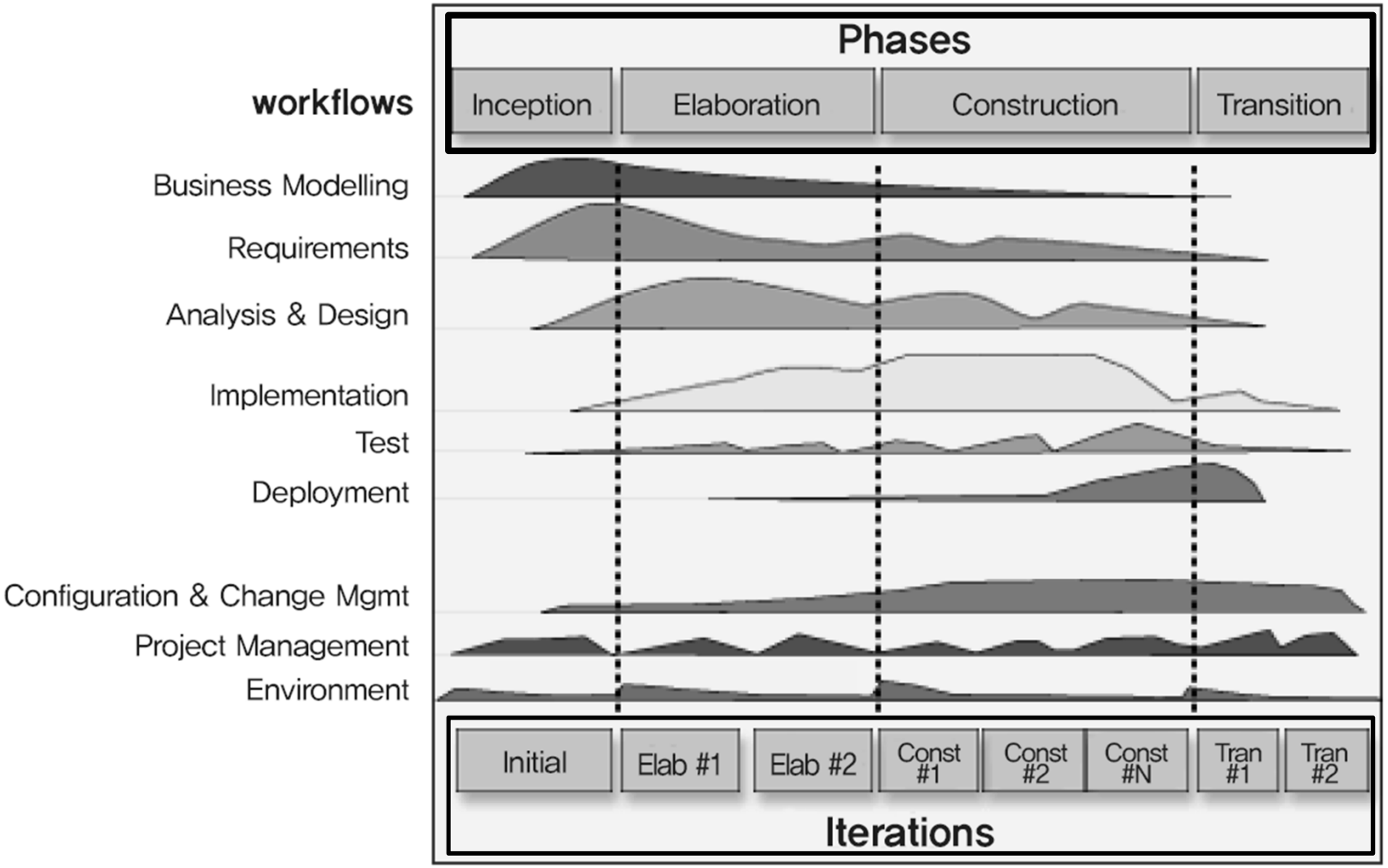
- UP(Unified Process) 개요
  - 반복적인(Iterative) 개발 방법론
    - 작업들을 계속해서 반복적으로 수행
    - 객체지향 프로세스에서는 분석, 설계, 개발, 테스트, 배포의 각 단계를 소규모 단계로 나누어 그 안에서 개발 업무를 반복 수행

- UP(Unified Process)
  - 프로젝트는 일련의 소규모 폭포수형 모델이 합쳐진 것!
  - UP를 구성하는 주요 철학 : 6 best practice
    - 유스케이스(Use-case) 중심
      - 유스케이스를 주요 산출물로 사용
      - 시스템에 요구되는 행동을 찾아냄
      - 시스템 아키텍처를 검증 및 확인
      - 테스트하고 프로젝트 이해당사자 간에 의사소통 하는 것
    - 아키텍처 중심
      - 시스템 아키텍처를 주요 산출물로 사용
      - 개발 중인 시스템을 개념화, 구축 및 관리하며, 친화시키는 것 의미

- UP(Unified Process)
  - UP를 구성하는 주요 철학 : 6 best practice(계속)
    - 반복적이고 점진적
      - 반복 프로세스 : 실행 배포판을 관리하는 것
      - 점진적 프로세스
        - » 시스템 아키텍처를 지속적으로 통합하여 배포판을 만드는 것
        - » 새로운 배포판은 이전 것에 대한 개정판
      - 반복 및 점진적 프로세스 :
        - » 위험관리 중심적인 것
        - » 프로젝트를 성공시키기 위해서 가장 심각한 위험을 조기에 공략하고 줄이는 데 초점을 두고 새로운 배포판을 개발하는 것

UP(Unified Process)

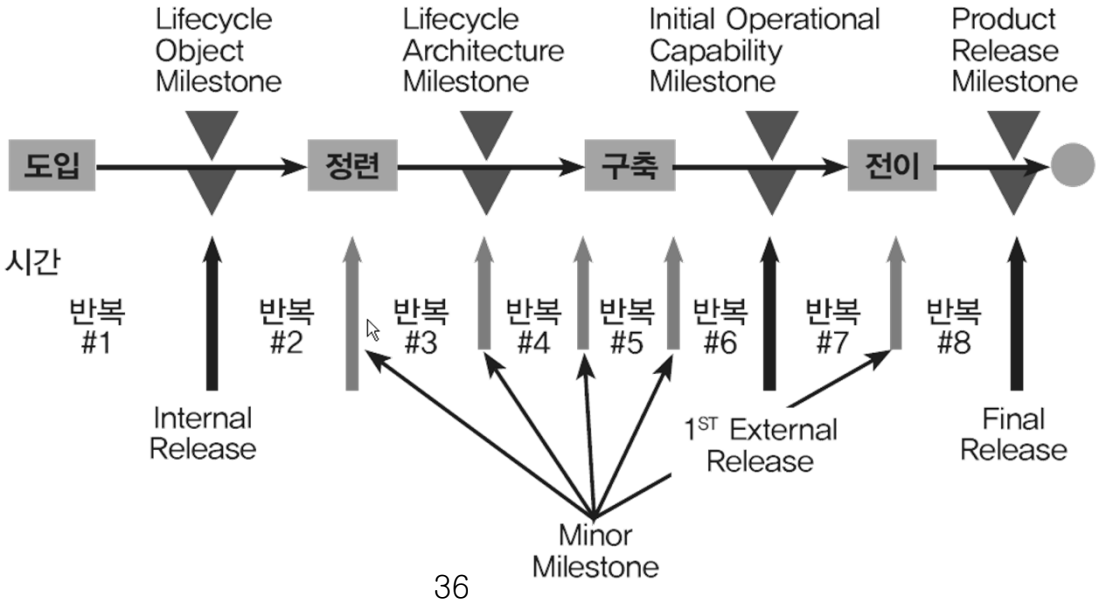
- UP의 개발생명주기



- UP의 개발생명주기(계속)
  - 도입 단계
    - 프로젝트의 시작
    - 정보를 수집하고 개념에 대한 교정 작업 수행
    - 도입 단계의 마지막에는 프로젝트를 계속 진행할 것인지 중단할 것인지 결정
  - 정련 단계
    - 유스케이스가 보다 자세하게 작성되고 시스템의 아키텍처 결정
    - 일부 분석과 설계, 코딩, 테스트 계획 포함
  - 구축 단계
    - 대부분의 코딩 작업 진행
  - 전이 단계
    - 사용자들에게 시스템을 배포하기 전에 최후로 준비하는 단계
    - 프로젝트 관리자가 앞에서 제시한 사항들을 평가하여 만족할 만한 평가를 얻었다면 프로젝트는 다음 단계로 넘어감

UP(Unified Process)

- UP의 개발생명주기
  - 생명주기(Life Cycle)와 이정표(Milestone)
    - 각 단계의 구분은 시간의 연속선상에서 주요한 이정표(Milestone)에 따라 분류
    - 각 단계는 프로세스에 있는 두 이정표 간의 기간
      - 정의가 명확한 목적이 선정
      - 산출물 완성
      - 다음 단계로의 이동 여부 결정



- UP(Unified Process)의 단계
  - 도입 단계(Inception Phase)
    - 프로세스의 첫 번째 단계
    - 개발의 근간이 되는 아이디어가 도출
    - 적어도 내부적으로는 정련 단계로 진입할 수 있을 만큼 충분한 근거가 되도록 함
    - “이런 (무엇인가 어떤 기능을 수행하는) 시스템이 있다면 훨씬 더 좋아지지 않을까?”라는 물음에서 시작
    - 비즈니스 모델링과 함께 시작
      - 원하는 시스템의 업무 내용을 분석
      - 시스템의 특징을 상위 수준에서 결정하고 그 내용을 문서화
      - 비즈니스 유스케이스, 비즈니스 액터, 비즈니스 유스케이스 다이어그램 작성
      - 워크플로우를 모델링하는 액티비티 다이어그램을 작성할 수도 있음

- UP(Unified Process)의 단계
  - 도입 단계(Inception Phase) (계속)
    - 정보들이 모두 준비되면 개발할 시스템을 분석하는 작업으로 넘어 감
      - 경영진에게는 예측한 결과 제공
      - 프로젝트를 지원할 수 있도록 CASE 도구를 이용하여 액터와 유스케이스를 생성
      - 유스케이스 다이어그램 작성
    - 조사 작업이 완료되고 경영진이 정련 단계를 수행할 인적, 물적 지원을 승인하면 도입 단계 종료
    - 반복 계획(Iteration Plan)
      - 반복 작업을 어떻게 수행할 것인지 계획
      - 어떤 유스케이스가 어떤 반복 작업 동안 구현될 것인가를 결정하는 계획
      - 이 계획을 통해서 어떤 유스케이스를 먼저 작업해야 하는지 파악
      - 유스케이스 간의 종속성 고려

- UP(Unified Process)의 단계
  - 도입 단계(Inception Phase) (계속)
    - 비즈니스 유스케이스 모델 구축
      - CASE 도구 이용
      - 비즈니스 유스케이스 모델에는 비즈니스 유스케이스(Business Use Case), 비즈니스 액터(Business Actor), 비즈니스 작업자(Business Worker)가 포함
    - 어떤 유스케이스와 액터가 필요한지 결정
      - CASE 도구 이용
        - » 유스케이스와 액터를 문서화
        - » 관계를 보여주는 다이어그램 작성
      - 유스케이스 다이어그램
        - » 시스템의 특징을 포괄적으로 잘 보여주는지 확인할 수 있도록 사용자들을 위한 프레젠테이션용으로도 사용 가능

UP(Unified Process)

- UP(Unified Process)의 단계
  - 정련 단계(Elaboration Phase)
    - 제품(Product) 비전과 아키텍처가 정의
    - 시스템의 요구사항이 명료하게 표현
    - 우선 순위와 기준선이 설정
    - 시스템의 요구사항
      - 총체적인 비전으로부터 상세한 평가 기준까지를 범위로 할 수 있음
      - 각 요구사항은 특정의 기능적 행동과 비기능적 행동을 명세화하며 테스트에 대한 기준 제공
    - 일부 계획안과 분석, 아키텍처 설계가 포함
    - 반복 계획에 따라 현재의 반복 작업에서 각 유스케이스에 대해 상세 분석
    - 프로젝트의 몇 가지 서로 다른 측면들이 포함
      - 개념에 대한 코딩 검증
      - 테스트 항목의 결정
      - 설계 시의 의사 결정
    - 프로젝트의 아키텍처적인 기초 설정에 초점을 맞춤

- UP(Unified Process)의 단계
  - 정련 단계(Elaboration Phase)
    - 주요 작업
      - 유스케이스를 상세화
      - 유스케이스의 하위 수준 요구사항에는 유스케이스를 실행시키는 프로세스의 흐름이 포함
      - 시퀀스 다이어그램, 콜래보래이션 다이어그램, 스테이트 다이어그램 포함
    - 유스케이스가 상세화된 형태로 수집된 요구사항
    - 소프트웨어 요구사항 기술서(SRS : Software Requirement Specification)라고 불리는 문서에 수록
    - SRS
      - 시스템 요구사항에 대한 모든 상세 내역 포함
    - 초기에 예측했던 사항을 좀 더 정제
    - SRS를 검토하여 유스케이스 모델의 품질을 고급화
    - 위험 요소들을 조사하는 작업

UP(Unified Process)

- UP(Unified Process)의 단계
  - 정련 단계(Elaboration Phase)
    - CASE Tool
      - 시퀀스 다이어그램, 콜래보래이션 다이어그램 작성 시 이용
    - 앞으로 개발할 객체들을 보여주는 클래스 다이어그램 설계
    - 상세 분석 단계 종료
      - 높은 위험성과 아키텍처적으로 중요한 유스케이스가 완전히 상세화
      - 사용자에게 승인
      - 개념에 대한 검증 완료
      - 초기 클래스 다이어그램까지 완성

UP(Unified Process)

- UP(Unified Process)의 단계
  - 구축 단계(Construction Phase)
    - 소프트웨어
      - 실행 가능한 아키텍처 기준선으로부터 사용자 쪽으로 전이될 준비 갖춤
    - 프로젝트에 대한 업무 요구 별로 요구사항과 평가 기준이 끊임없이 재검사
    - 위험 요소들을 적극적으로 공략하는 데 적합하도록 자원이 할당
    - 시스템의 나머지 부분들을 분석, 설계, 구축
      - 상세 분석 단계에서 얻은 아키텍처를 기초로 하여 시스템의 나머지 부분 구축
    - 구축 단계의 작업
      - 아직 남아있는 요구사항에 대한 결정
      - 소프트웨어의 개발 및 테스트
    - CASE 도구
      - 시스템에 대한 기본 골격 코드 생성
      - 구축 단계의 초기 시점에서 컴포넌트와 컴포넌트 다이어그램 작성 필요

- UP(Unified Process)의 단계
  - 구축 단계(Construction Phase)
    - 컴포넌트를 만들고 서로의 종속성을 결정하면 코드 생성 시작
    - 코드 생성 기능
      - 설계에 기반하여 생성된 코드 획득
      - 업무에 관련된 코드까지 생성해 주는 것은 아님
      - 프로그래밍 언어에 대단히 종속적
      - 일반적으로 클래스 선언과 속성 선언, 가시성 선언(public, private, protected), 함수 원형이나 상속과 같은 내용 등 포함
    - 코드 완성
      - 완성된 코드가 표준과 설계 기준을 제대로 따랐는지 기능은 제대로 작동하는지 확인하기 위해 개발자들은 서로 코드를 검토
      - 코드 검토가 끝나면 객체들은 품질 관리 검토
      - 구축 단계에서 새로운 속성 및 함수가 추가되었거나 객체 간 상호작용 중 변경된 것이 있다면 역공학(Reverse Engineering)을 통해서 모델 반영
    - CASE 툴 이용
      - 시퀀스 및 콜래보래이션 다이어그램, 클래스 다이어그램, 스테이트/액티비티 다이어그램, 컴포넌트 다이어그램 작성
      - 객체 설계에 따라서 컴포넌트를 생성해내는데 도구 이용

UP(Unified Process)

- UP(Unified Process)의 단계
  - 전이 단계(Transition Phase)
    - 소프트웨어가 사용자에게 이전
    - 소프트웨어 개발 프로세스가 여기서 끝나는 경우는 거의 없으며, 이 단계에서도 시스템은 지속적으로 개선되고, 결함 제거가 수행되고, 이전의 배포판에 없는 특성이 추가
    - 완성된 소프트웨어 제품이 사용자 그룹에게 양도되는 시점
    - 이 단계에서 해야 할 작업
      - 최종적으로 소프트웨어 제품을 완성
      - 마지막 테스트를 수행
      - 사용자 문서 작업을 완료
      - 사용자 교육 준비
      - 소프트웨어 요구사항 기술서(SRS), 유스케이스 다이어그램, 클래스 다이어그램, 컴포넌트 다이어그램, 디플로이먼트 다이어그램 모두 최종 변경 사항을 반영하도록 수정
    - CASE 도구 이용
      - 주로 소프트웨어 개발이 완료되었을 때 모델을 수정하는 용도로 사용
      - 특히 컴포넌트와 배치 다이어그램에 대한 수정 작업이 주로 발생

01장 요약

- 모델링이란?
- UML과 UP방법론의 특징과 장단점
- 구조적 방법론과 객체지향 방법론의 차이점
- UML의 뷰
- UP방법론의 각 단계

강의 계획 피드백 (5주차)

주차	강의주제	강의내용	과제	평가
1주차	객체지향 패러다임	과목 소개 및 객체지향 방법론의 전반적인 개요		
2주차	프로젝트 관리1	프로젝트 계획 및 팀 편성/프로젝트 과제 제시		
3주차	소프트웨어 개발방법론과 UML	기존의 소프트웨어 개발방법론과 객체지향방법론 차이점 이해	과제1 : 프로젝트 현장 및 계획서 제출(5)	
4주차	Use Case와 UML	UML 특성 이해		
5주차	UP(Unified Process) 방법론	UP 방법론 이해		
6주차	비즈니스 모델링 및 요구사항 정의	사례를 통한 비즈니스 모델링 및 요구사항 정의 방법 이해	과제2 : 요구사항 정의 결과 제출(5)	
7주차	분석 모델링 및 UML 다이어그램(분석)	객체지향 분석 방법 이해 및 분석용 UML 다이어그램 작성 방법 이해		
8주차	분석 결과 문서화 및 설계 모델링	분석 산출물 작성 방법 및 객체지향 설계 방법 이해	과제3 : 분석 결과 제출(10)	
9주차	UML 다이어그램(설계)	설계용 UML 다이어그램 작성 방법 이해		
10주차	객체 설계	객체설계 및 세분화		
11주차	설계 결과의 문서화 및 프로젝트 관리 2	시스템 설계 결과의 문서화 방법 이해 및 형상관리/검증과 확인 방법 이해	과제4 : 설계 결과 제출(10)	
12주차	시스템 구현	객체지향 프로그래밍의 기본 개념 및 기법		
13주차	시스템 테스트 및 구현/시험 결과의 문서화	객체지향 테스팅 기법 및 구현/시험 산출물의 문서화 방법 이해	과제5 : 구현/시험 결과 및 유지보수 계획 제출(20)	
14주차	프로젝트 관리3	소프트웨어 품질관리와 프로세스 개선 방법 이해		
15주차	최종 결과 문서화 및 발표	최종 산출물 문서화 방법 이해 및 개발 결과 발표	과제6 : 최종보고서 제출 및 발표(10)	

다음 주(5주차) 강의계획

- 교재#2의 2장 꼭 읽어오기
- 팀 프로젝트 진행 : 계획단계-분석단계
  - 프로젝트 헌장(PC) 수정제출 : 최종 제출(2016.09.28, 18:00)
  - 프로젝트 관리 계획서(PMP) :
    - 1차 제출(2016.09.28, 18:00)
    - 최종 제출(2016.10.05, 18:00)
  - 요구사항 수집 및 요구사항 정의 : 1차 제출(2016.10.05, 18:00)

☞ 다음 주(6주차)는 **교재#2의 2장 강의**