

개발자 오리엔테이션

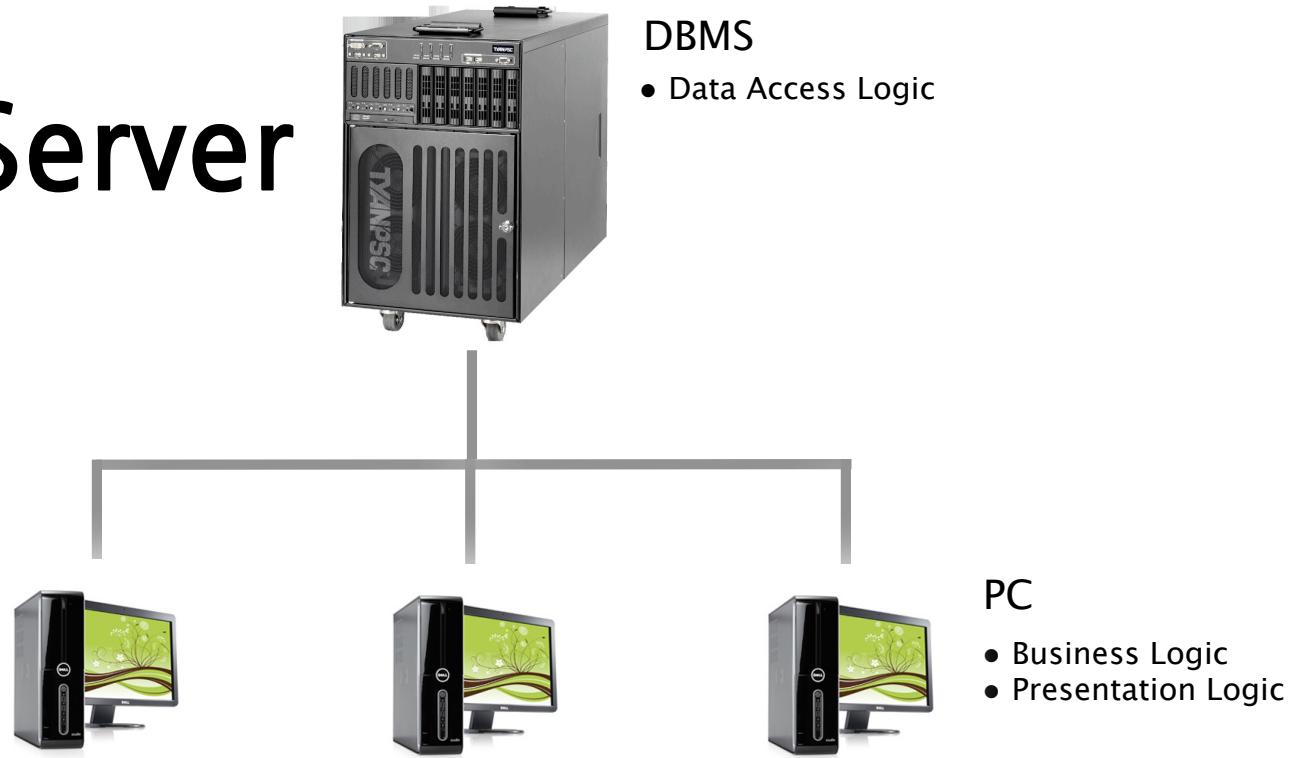
향후 유망 분야의 필요 기술

1990s

- Windows 3.0 릴리즈(1990.5.22)
- Windows 3.1 릴리즈(1992.4.6)
- Windows 95 릴리즈(1995.8.24)
- LAN/WAN(노벨 네트워어)
- Internet (천리안, 1994)

1990s

Client/Server



- 4GL(PowerBuilder, Delphi, VB)
- OOP, C/C++
- Windows API, MFC, ODBC API
- HTML, CGI programming

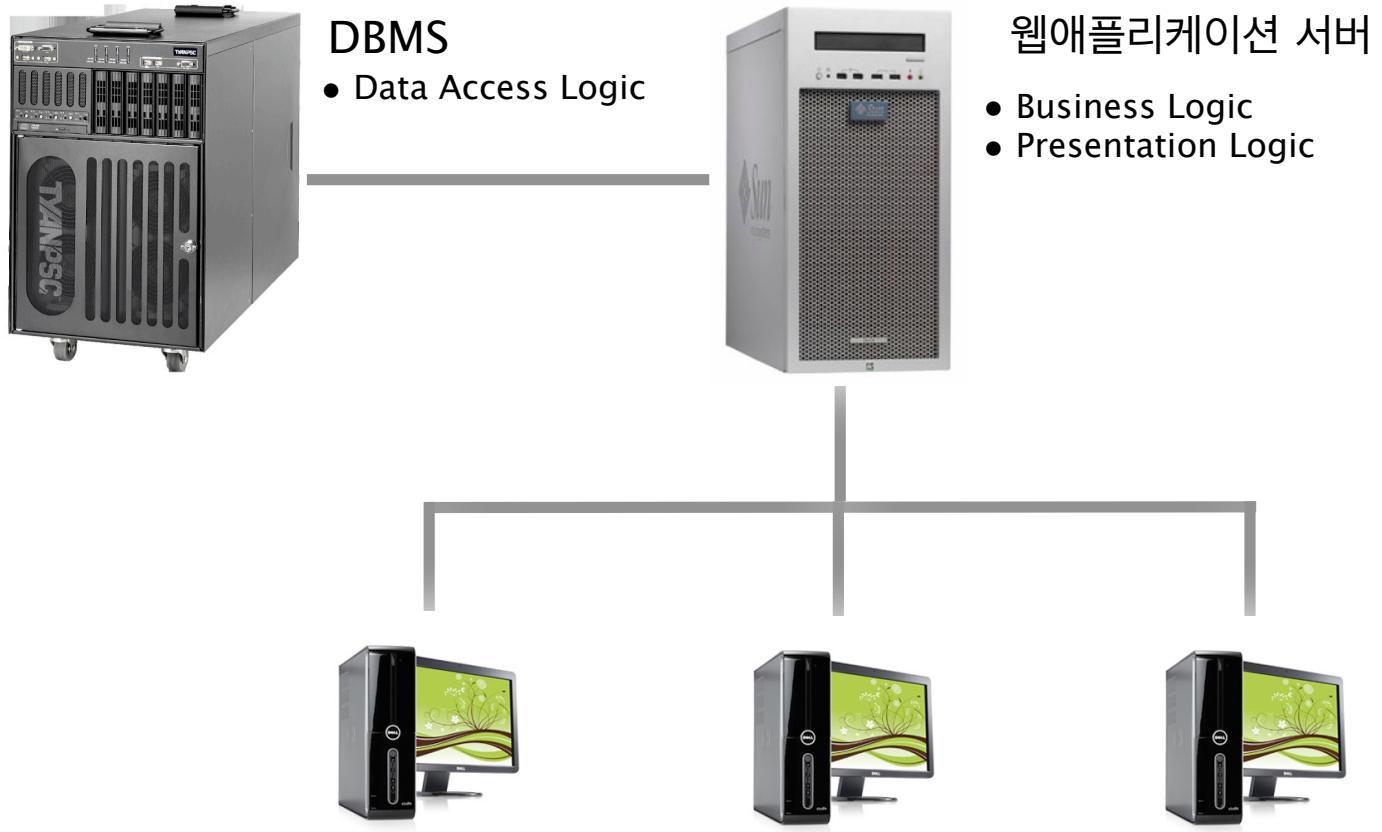
2000s

상반기

- 초고속망 등장
- 글로벌 비즈니스 가속화
 - 경쟁 심화
 - 비즈니스 프로세스의 잣은 변화
 - 시스템의 잣은 변경

1990s

2000s
상반기



- 웹 프로그래밍(Perl, PHP, ASP, Servlet/JSP)
- 분산 컴퓨팅(RMI, EJB, Web-Service)
- POJO 컨테이너(Spring Framework)
- Agile 개발 방법론(Extreme Programming)

1990s

2000s
상반기

1999 ~ 2001 창업!

실패

- 개발 팀원의 역량 부족
- 프로젝트 관리의 미숙
- 비즈니스 모델의 부재

Refactoring

요구사항수집

분석

설계

구현

테스트

배포

Refactoring

- 코드의 구조를 개선 → 유지보수 향상
- 클래스/메서드 분리, 메서드 이동, 메서드 이름 변경, ...
- Refactoring : Improving the Design of Existing Code
 - Martin Fowler 외, Addison-Wesley

UML

Unified Modeling Language

모델링



Three Amigos

- Grady Booch
- James Rumbaugh
- Ivar Jacobson

UML

Unified Modeling Language

- | | |
|-----------|-------------------|
| • 1997년 | UML 1.0 초안 발표 |
| • 1997년 말 | OMG 표준 모델링 언어로 채택 |
| • 1998년 | UML 수정안 발표 |
| • 현재 | UML 2.0 |

모델링



Three Amigos

- Grady Booch
- James Rumbaugh
- Ivar Jacobson

UML

Unified Modeling Language

- Rational Rose Rational → IBM
- Together TogetherSoft → Borland
- Visual Paradigm
- Enterprise Architect

...

Design Pattern

요구사항수집

분석

설계

구현

테스트

배포

문제 해결을 위한 설계 방법

이미 여러 시스템 개발에 사용되어
효과가 검증된 방법

Design Pattern *Best Practices!*

문제 해결을 위한 설계 방법



Design Pattern

- 건축학 영역에서 크리스토퍼 알렉산더가 처음으로 고안
- Design Pattern : Elements of Reusable Object-Oriented Software(GoF)
- GoF(Gang of Four)
 - Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides

요구사항수집

분석

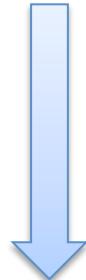
설계

구현

테스트

배포

문제 해결을 위한 설계 방법



Design Pattern

- 객체 생성: Singleton, Builder, Factory Method, ...
- 객체 구조: Adapter, Decorator, Façade, Flyweight, ...
- 객체 행위: Chain of Responsibility, Command, Iterator, Observer, ...

Applying UML & Patterns

[Graig Larman, Prentice Hall PTR]



Unified Process

프로젝트
시작

종료

요구사항수집

분석

설계

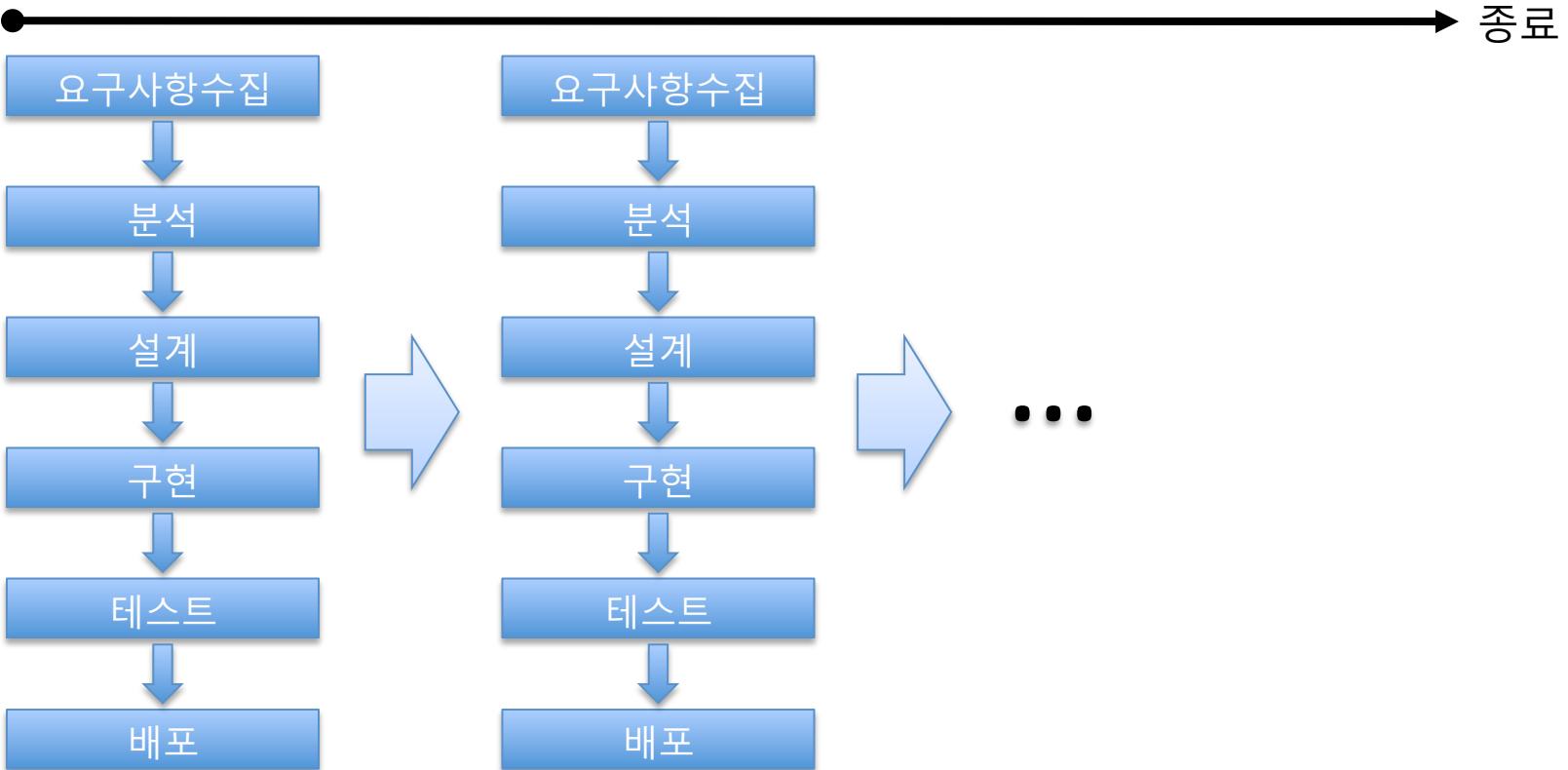
구현

테스트

배포

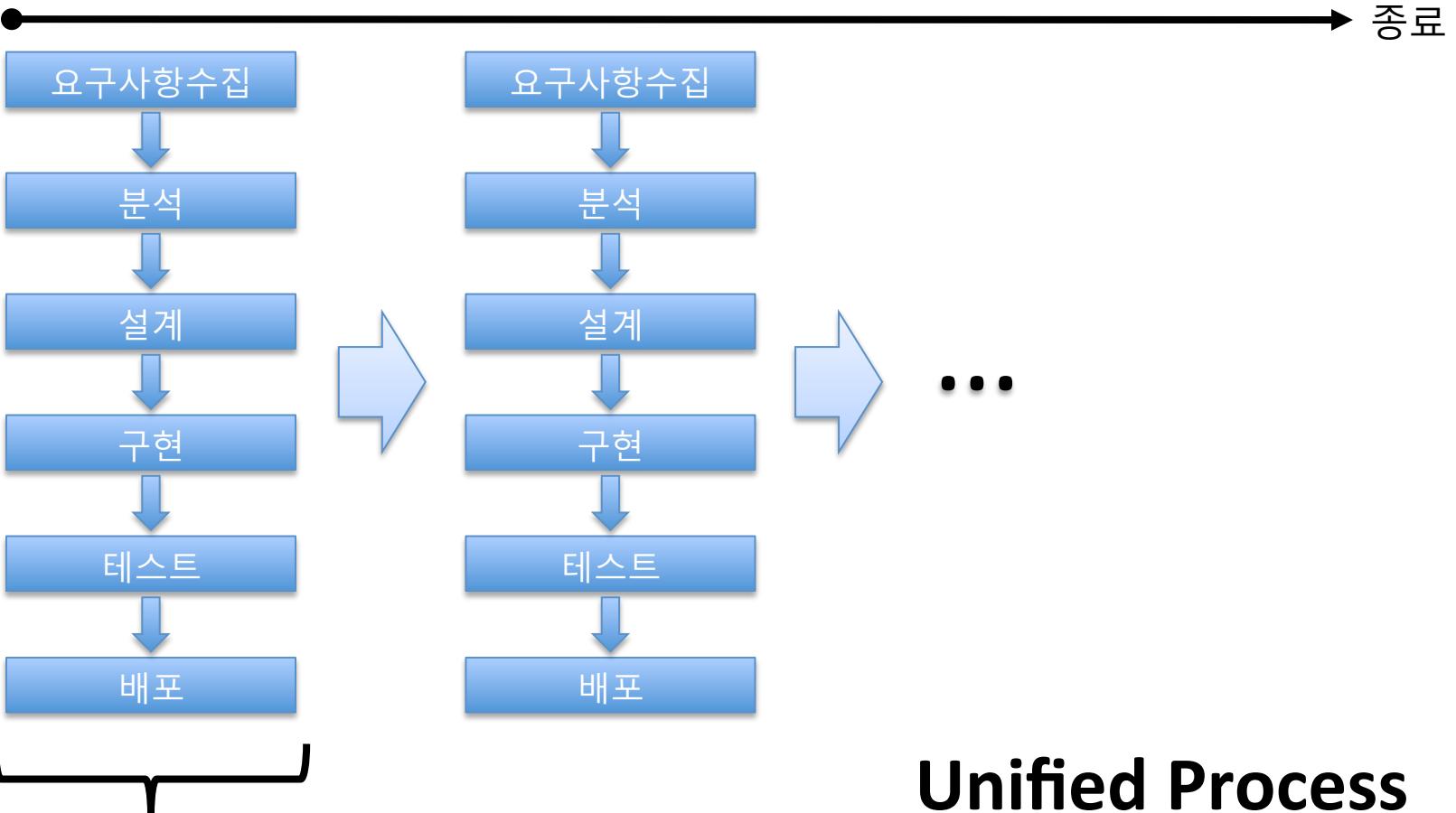
폭포수 모델

프로젝트
시작



Unified Process

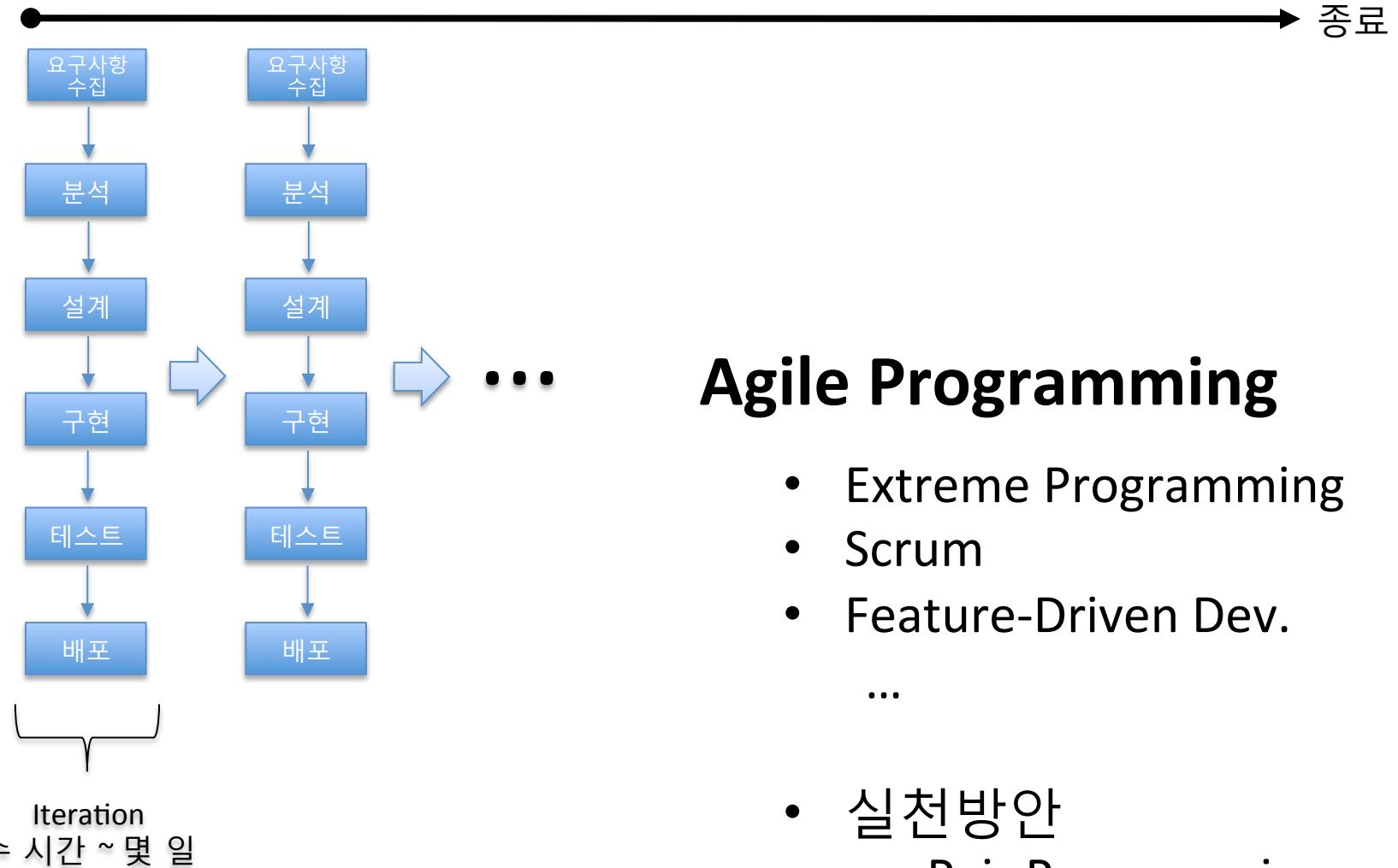
프로젝트
시작



Iteration
2주 ~ 6주

Unified Process

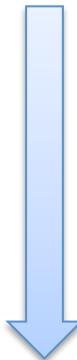
프로젝트
시작



Agile Programming

- Extreme Programming
- Scrum
- Feature-Driven Dev.
- 실천방안
 - Pair Programming
 - Test-Driven Dev.

Unified Process



Use-case 명세 기반

OOA/D

Object-Oriented Analysis & Design

- 객체 식별
- 객체 협력 모델링
- 메서드 식별

Case Study

AirDisk 서비스

자료 공유 서버

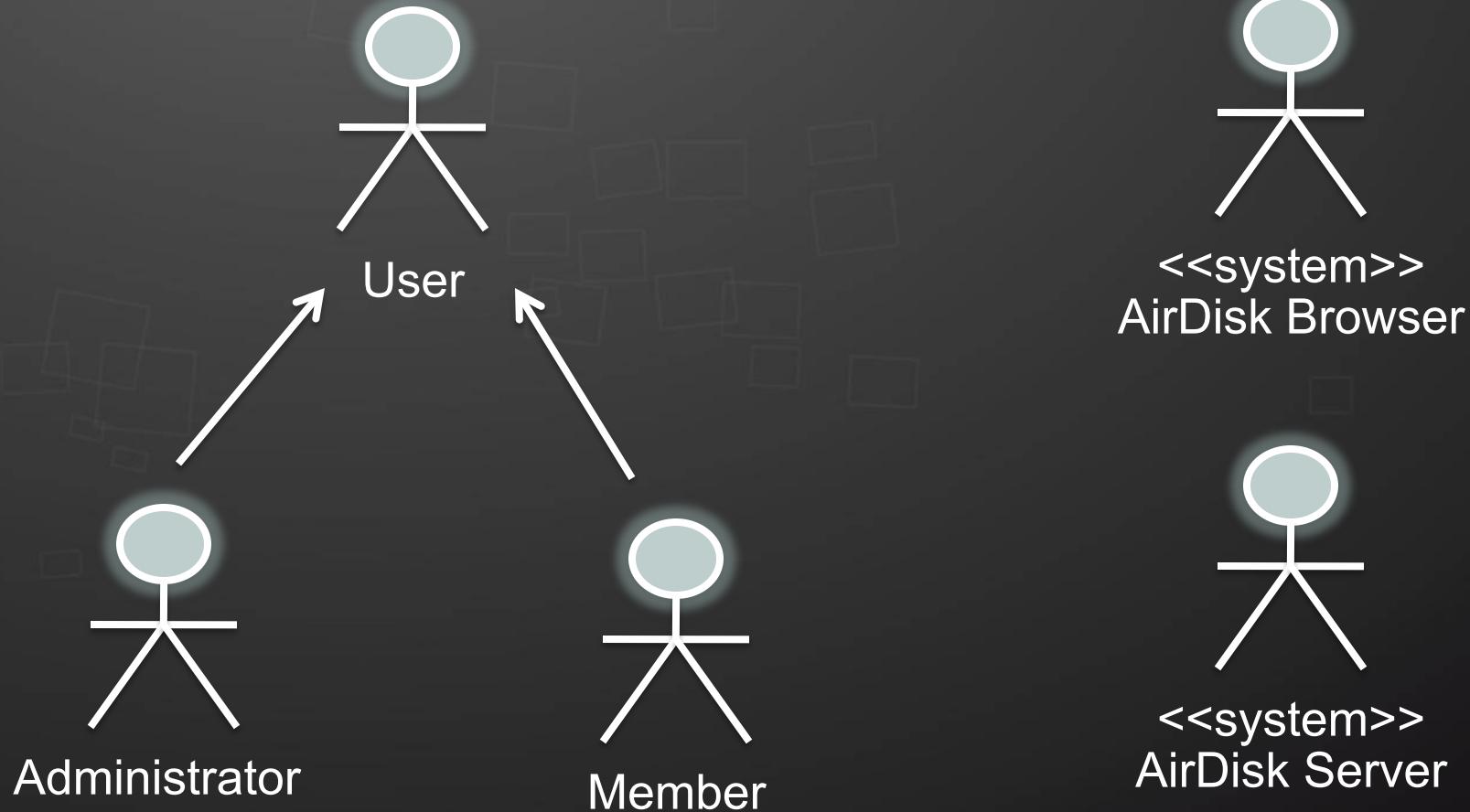
자료 공유 서버



- 개방형 웹하드 및 자료실
- 보안 기술“AirTicket”
- 친숙한 UI
- 다양한 플랫폼 지원
- 손쉬운 통합
- Open Source Project

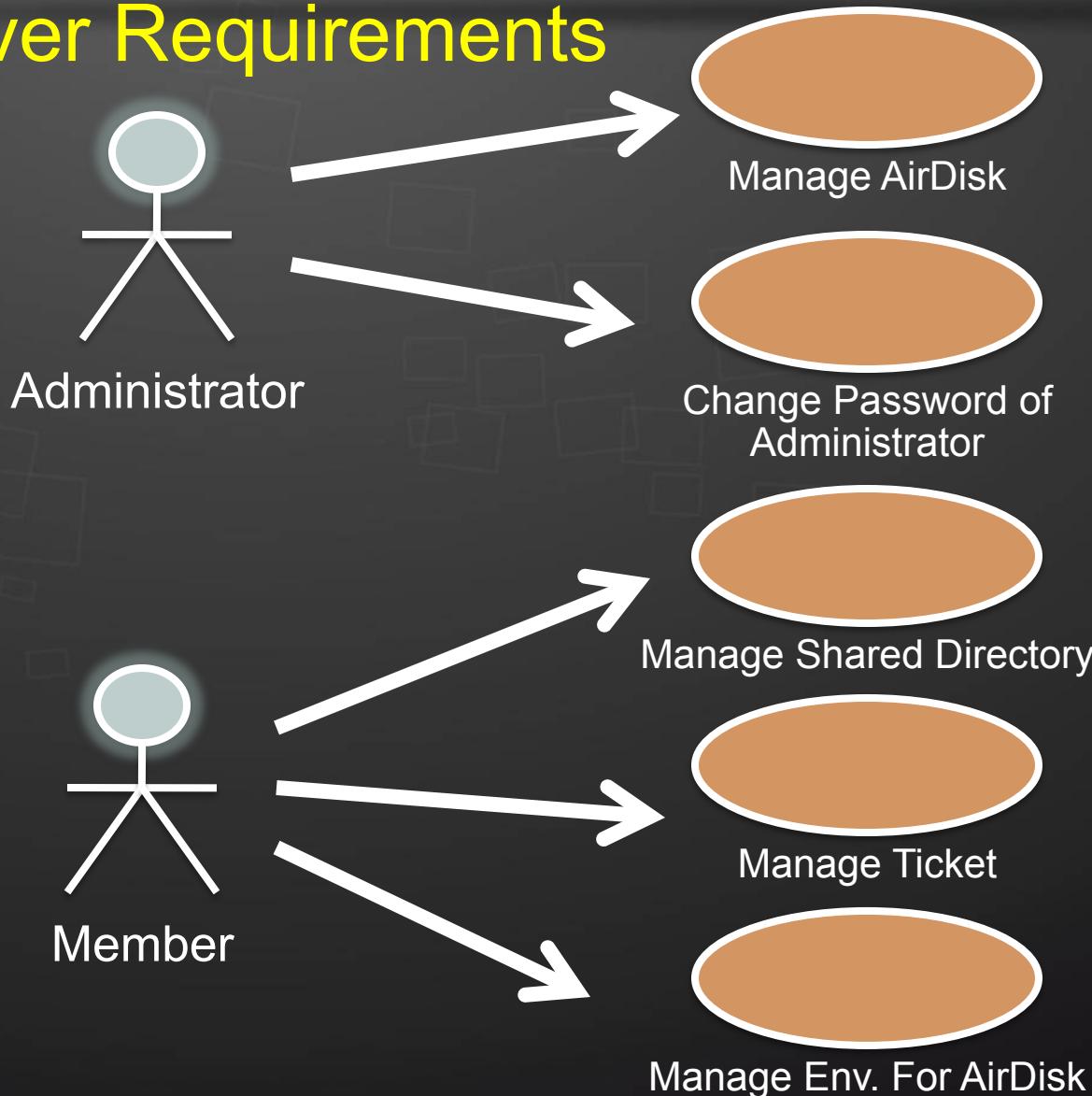
단계1. 요구사항 식별 – Actor 식별

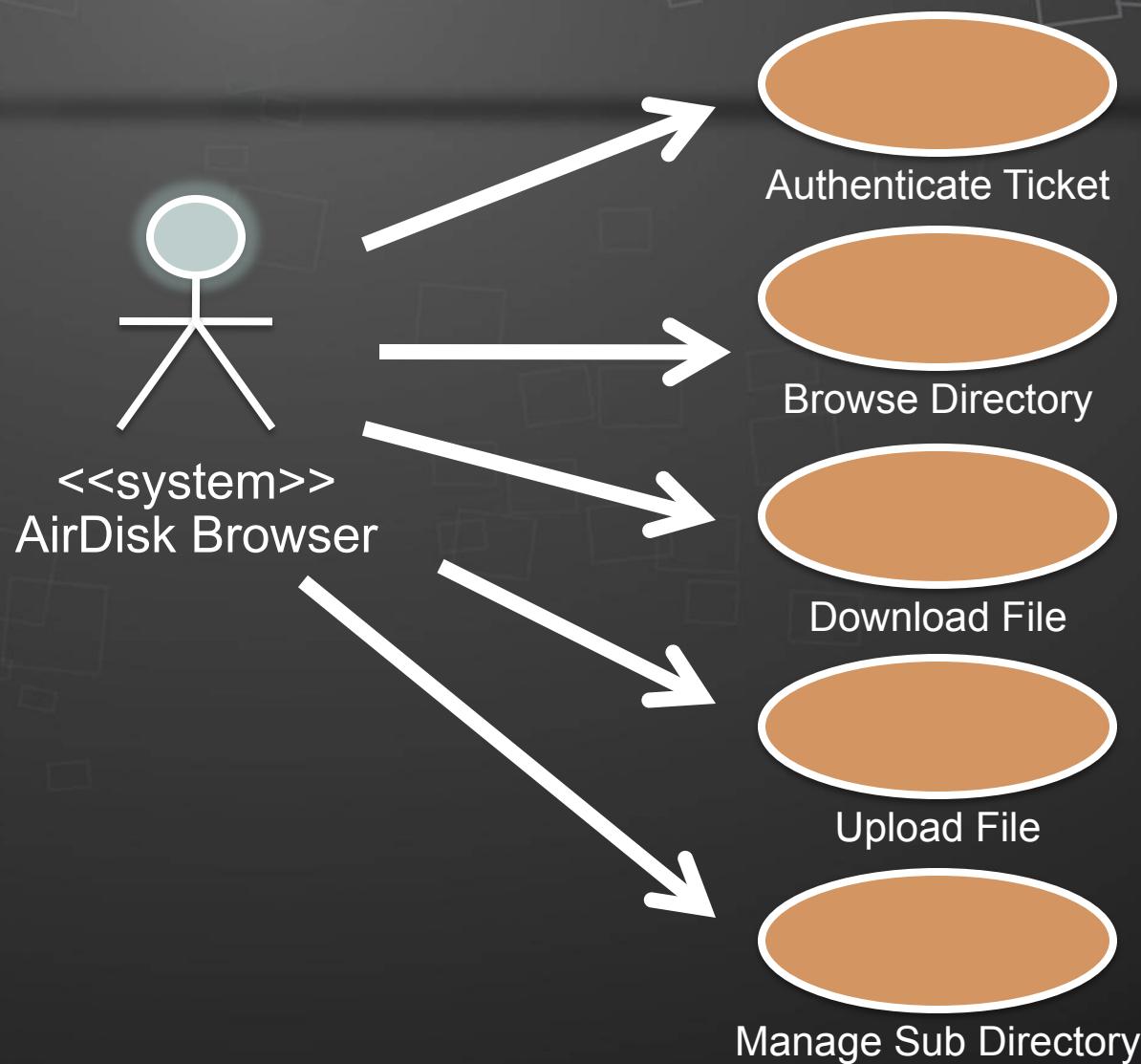
<http://www.airdisk.org/airdisk>



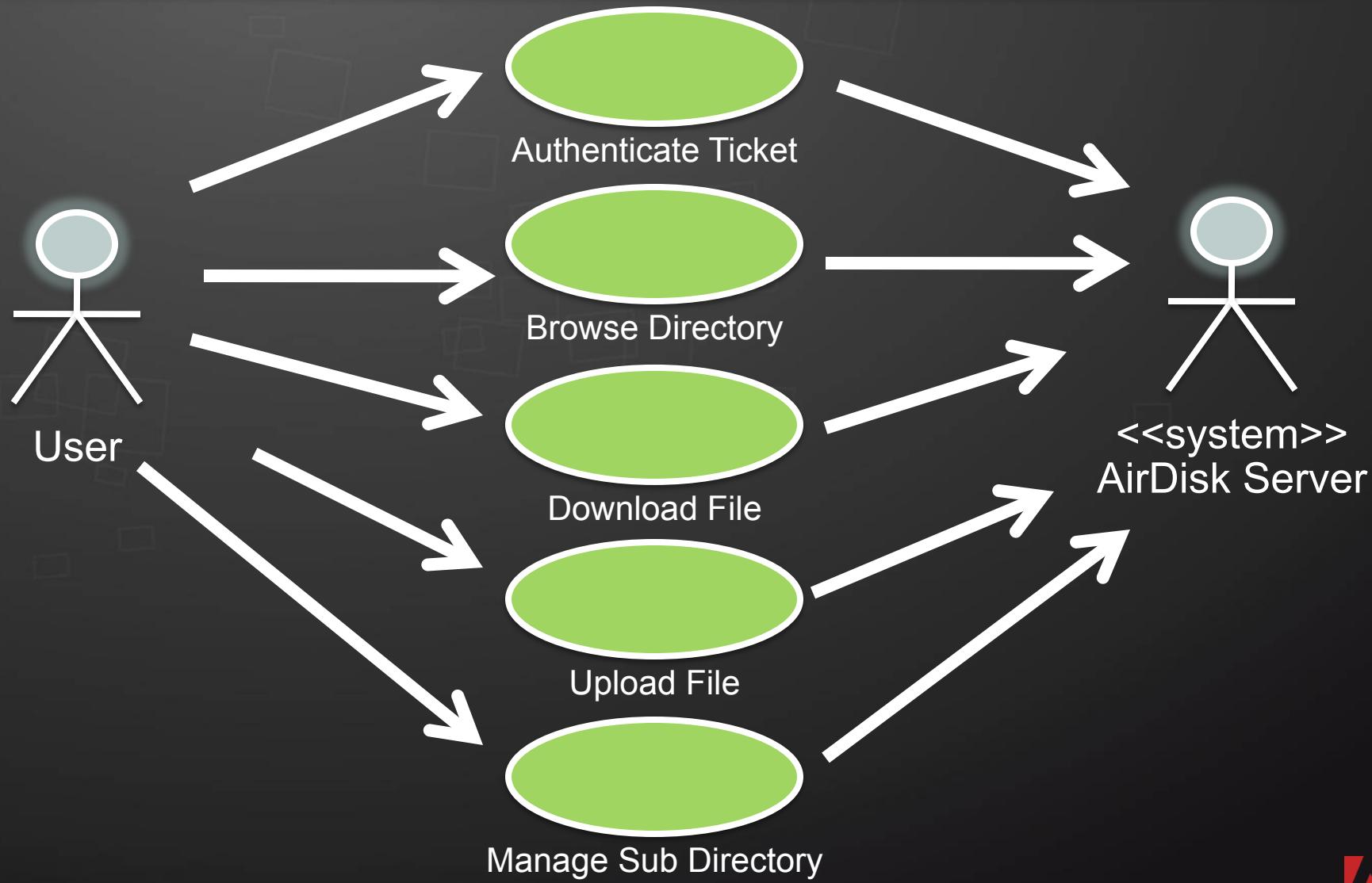
단계2. 요구사항 식별 – Use-case 식별

Server Requirements





Client Requirements



단계3. 요구사항 정의 – Use-case 명세서

Use-case name

Actor

Pre-condition

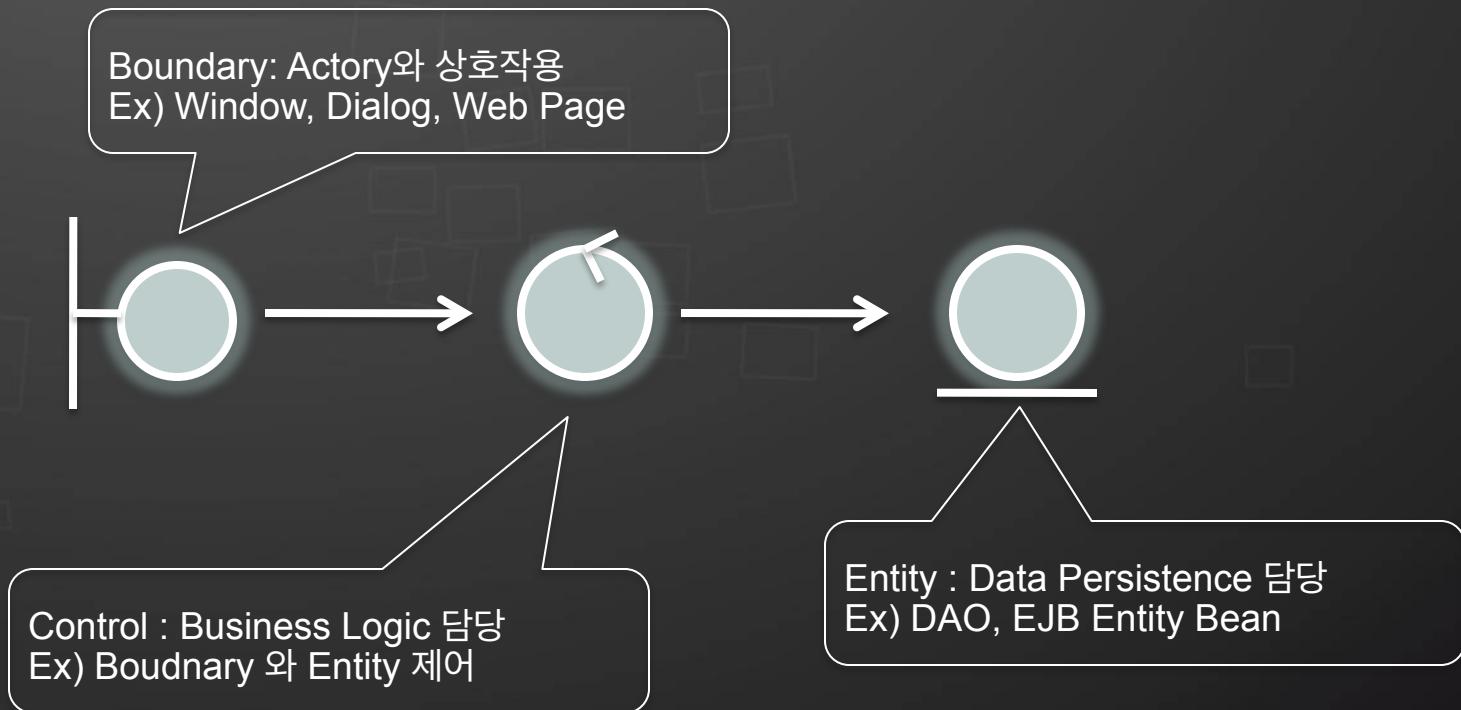
Successful Flow

Alternate Flow

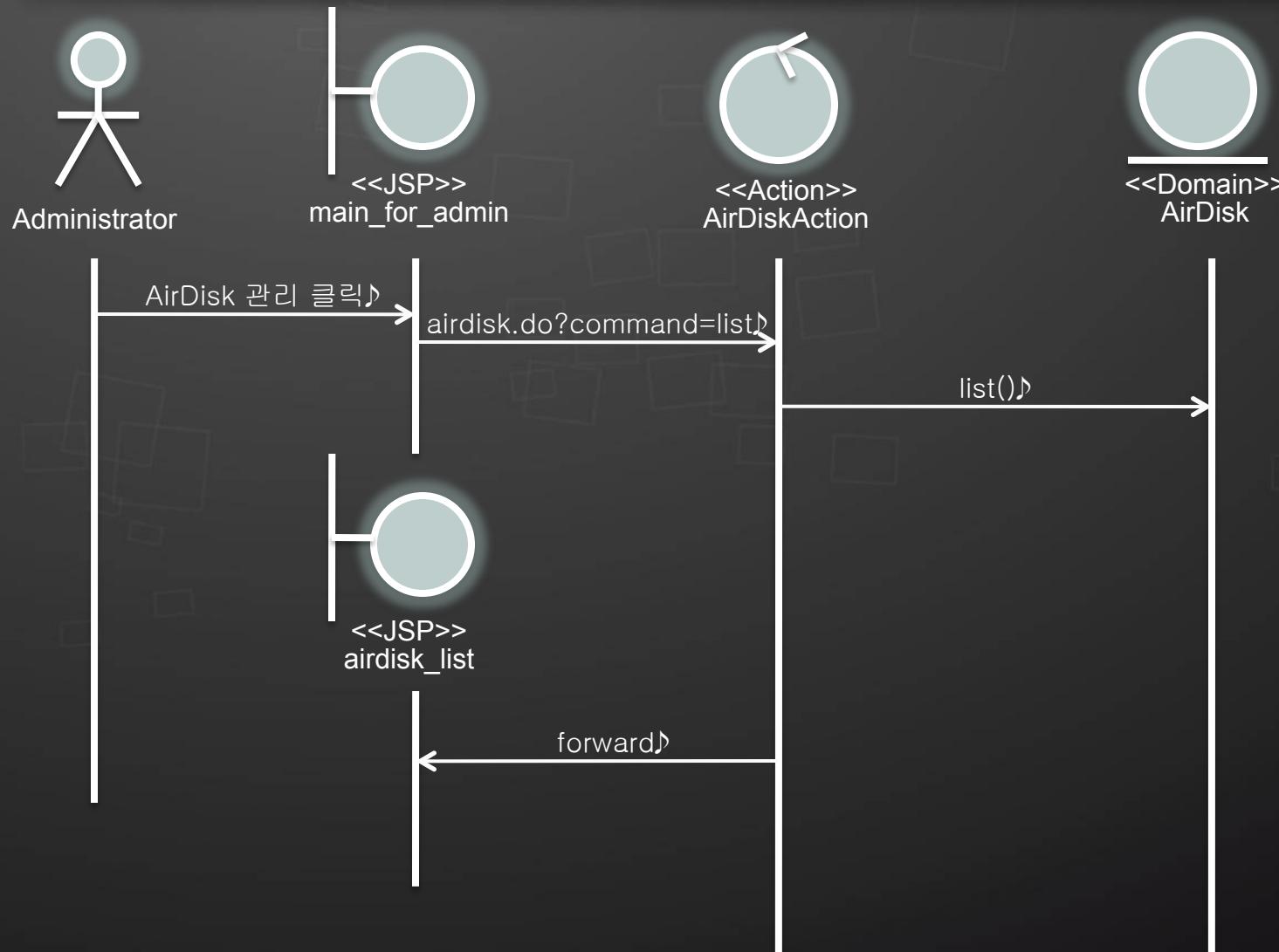
Post-condition

단계4. Robustness 분석 – Use case 별

Use case에 참여하는 객체들



단계5. 설계 – Use case 별



1990s

2000s
상반기

2000s 하반기

- 개방, 참여, 공유
- RIA, UX
- iPhone 1세대 등장(2007.6.29)
- 첫 안드로이드폰(2008.10)
- SNS의 부상

1990s

2000s
상반기

2000s
하반기



1990s

2000s
상반기

2000s
하반기

2010s

상반기

- 첫 iPad(2010.4.3)
- Galaxy Tab(2010.9)
- Wearable Computing
- Smart TV
- N-Screen
- HTML5
- Tizen 1.0 릴리즈(2012.4.30)
- Firefox OS 1.0(2013.2.21)
- 클라우드 서비스
- 빅데이터

1990s

- Hybrid App(PhoneGap, HTML5 + CSS + JavaScript)
- N-Screen → 반응형 웹
- Smart Embedded System 포팅

2000s
상반기

2000s
하반기

2010s
상반기



Samsung
GALAXY Camera



1990s

- SaaS, PaaS, IaaS
- Open API



2000s

상반기



2000s

하반기

2010s

상반기

Application



Platform



Infrastructure



Phones



Tablets

Cloud Computing

1990s

2000s
상반기

2000s
하반기

2010s
상반기

정형 데이터

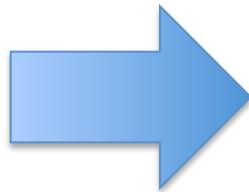
- 센서 신호, 로그, 매출, 재고 등

반정형 데이터

- XML, HTML 등

비정형 데이터

- 소셜 컨텐트, 메일, 블로그
게시판, 그림, 영상, 위치, 로그 등



빅 데이터 분석

- Text mining
- Opinion mining
- Social network analysis
- Cluster Analysis

- 빅데이터 분산 처리(Hadoop, NoSQL)
- 분석 방법: 통계학, 데이터 마이닝, 자연어 처리, 패턴 인식
- 활용 예) 2008년 미국 대통령 선거
 - 유권자를 인종, 종교, 나이, 가구형태, 소비수준, 과거투표여부, 구독하는 잡지, 마시는 음료 → 유권자 지도 → 맞춤형 선거

Framework

- 다양한 디자인 패턴과 라이브러리들을 사용
- 문제 해결을 위한 기본 기능 및 공통, 고정 기능을 구현
- 변경하거나 추가할 기능은 프레임워크 규칙에 따라 개발함
- '**여러 애플리케이션에서 공유할 수 있는, 재활용 가능하고 보편적인 구조 제공**' (Ralph E. Johnson 외, Desiging Reusable Classes 중에서)

Class와 CBD, Framework의 비교

Class

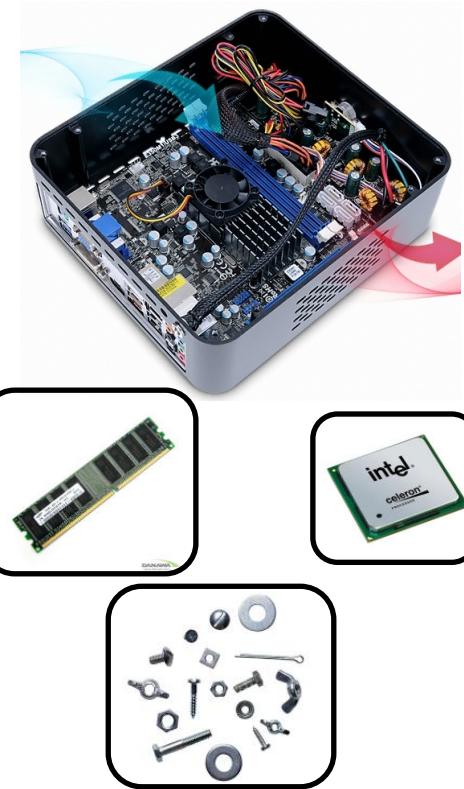


CBD

Component-Based Development



Framework



Class와 CBD, Framework의 비교

	Class	CBD Component-Based Development	Framework
개발속도	느림	빠름	매우 빠름
숙련도	고급	중급	초급
안정성	개발자의 역량에 따라 결정	검증된 컴포넌트 사용으로 어느 정도 안정성 보장	개발자의 역량에 상관없이 최소한의 품질을 보장
재사용	다양한 도메인의 시스템 개발에 사용 가능	컴포넌트 재사용	재사용 높음

Framework

빈 관리 컨테이너

- 객체의 생성 관리
- 의존 객체 주입(DI = IoC)
- 메서드 호출 전/후에 필터링(AOP)
- 데이터 검증, 타입 변환, 테스팅
- Spring Framework

빈 관리 컨테이너

Framework

웹 MVC 아키텍처

- MVC 아키텍처 구축
- 모델, 뷰 기반 기술 제공
- 다양한 뷰 프레임워크와 연동
- 웹 리소스 관리
- Spring MVC, Struts

웹
MVC

빈 관리 컨테이너

Framework

데이터 처리

- 트랜잭션 관리
- 객체 관계 맵퍼 지원
- SQL 맵퍼 지원
- Hibernate, JDO, JPA, iBatis

웹
MVC

데이터
처리

빈 관리 컨테이너

HTML5

- HTML : UI 레이아웃
- CSS : UI 스타일 제어
- JavaScript : UI 컴포넌트 제어, 사용자 이벤트 처리
- DOM API : HTML, XML 엘리먼트 제어
- AJAX : 비동기 서버 애플리케이션 요청

jQuery

- Cross browsing 지원
- 짧은 코드, 많은 기능 구현
- CSS 기반 UI 컴포넌트 제어
 - 모바일 UI 지원

The Business Model Canvas

Designed for:

Designed by:

On: Day Month Year

Iteration: No.



Cost Structure

What are the most important costs inherent in our business model?
Which Key Resources are most expensive?
Which Key Activities are most expensive?

클라우드 시스템 임대비

인건비

방법론 및 템플릿 라이센스 비

Revenue Streams

For what value are our customers really willing to pay?
For what do they currently pay?
How are they currently paying?
How would they prefer to pay?
How much does each Revenue Stream contribute to overall revenue?

타당화
Annual fee
Usage fee
Subscription fee
Lending/Renting/Licensing
Royalty fees
Advertising

판권료
One-time fee
Product feature dependent
Customer segment dependent
Volume dependent

판권료
One-time fee
Product feature dependent
Customer segment dependent
Volume dependent

년간 서비스 이용료 (개인, 기업)

회의 템플릿 판매



- UP/Agile 개발 프로세스
- UML
- OOA/D
- Design Pattern

- Framework
- Web Server Application
- Refactoring
- HTML5 Programming
- jQuery

JUnit

- 테스팅 자동화 프레임워크
- 독립적으로 수행되는 단위 테스트
- 단위 테스트 별 오류 식별 및 보고

- UP/Agile 개발 프로세스
- UML
- OOA/D
- Design Pattern
- Framework
- Web Server Application
- Refactoring
- HTML5 Programming
- jQuery
- JUnit

ANT

- 복잡하고 유동적인 빌드 프로세스를 제어에 적합
- 빌드→테스트→배치 프로세스에 사용
- 크로스 플랫폼 지원

- UP/Agile 개발 프로세스
- UML
- OOA/D
- Design Pattern

- Framework
- Web Server Application
- Refactoring
- HTML5 Programming
- jQuery

- Junit
- ANT

Maven

- 정형화된 구조의 빌드 도구
- 프로젝트 관리 도구
- 라이브러리 자동 다운로드

- UP/Agile 개발 프로세스
- UML
- OOA/D
- Design Pattern

- Framework
- Web Server Application
- Refactoring
- HTML5 Programming
- jQuery

- Junit
- ANT
- Maven

Continuous Integration

- 팀의 구성원들이 작업한 내용을 최소 일 단위로 통합 수행
- 통합이 수행될 때마다 자동 빌드 도구를 통하여 자동으로 테스트 되고, 검증되고, 배포됨
- 오류 발생 시 자동으로 보고
- Hudson, Jenkins

- UP/Agile 개발 프로세스
- UML
- OOA/D
- Design Pattern
- Framework
- Web Server Application
- Refactoring
- HTML5 Programming
- jQuery
- Junit
- ANT
- Maven
- CI

형상관리

- 소프트웨어 개발 과정에 만들어 지는 산출물의 체계적인 관리
- 소스의 버저닝을 통해 소프트웨어의 가시성과 추적성 부여
→ 품질보증
- 팀원들의 작업 결과물 공유 및 변경, 병합
- CVS → SVN → git

- UP/Agile 개발 프로세스
- UML
- OOA/D
- Design Pattern
- Framework
- Web Server Application
- Refactoring
- HTML5 Programming
- jQuery
- JUnit
- ANT
- Maven
- CI
- SVN, git