Beyond Promise

# 금융 IT의 새로운 길 : 모델중심개발

2012.12.12

LG CNS 최종태 부장

LG CNS

# AS- IS does matter!

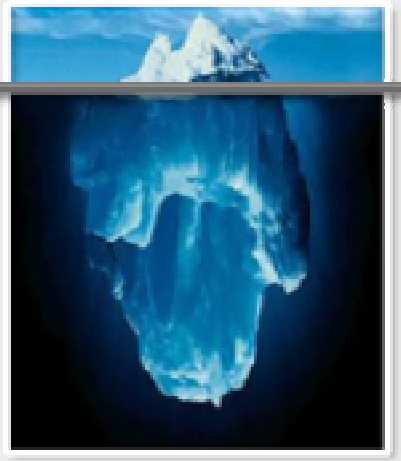
과거의
금융IT 시스템

업무요건의
퇴적

미래의
금융IT 시스템

TO-BE

AS-IS



M&A

규제 대응

다양한 제휴

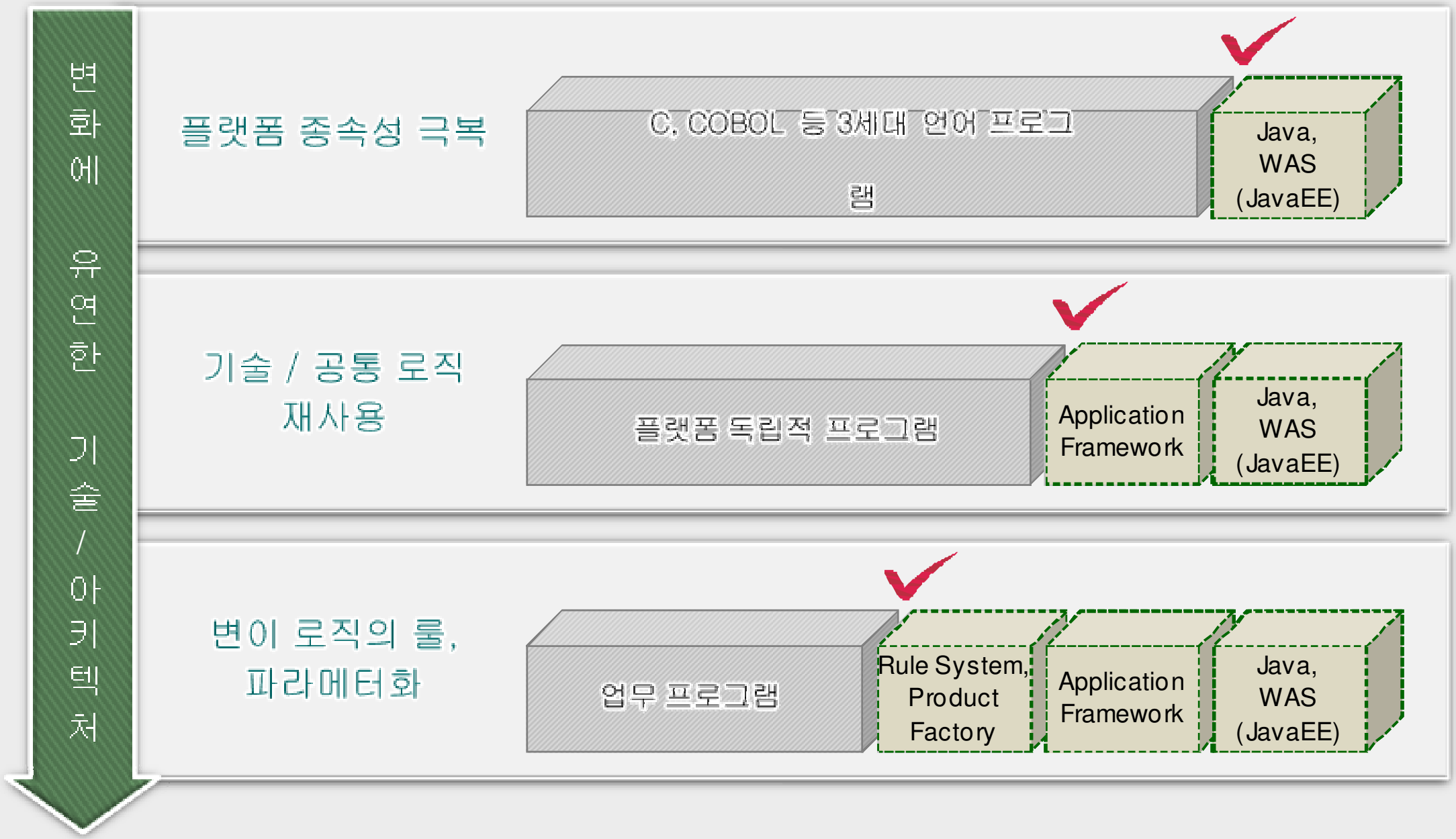
상품 트랜드

업계 BP 적용…

다음 차세대
규모, 기간, 비용은
어는 정도일까?

유지보수는?

# Separation of concern(Technology)

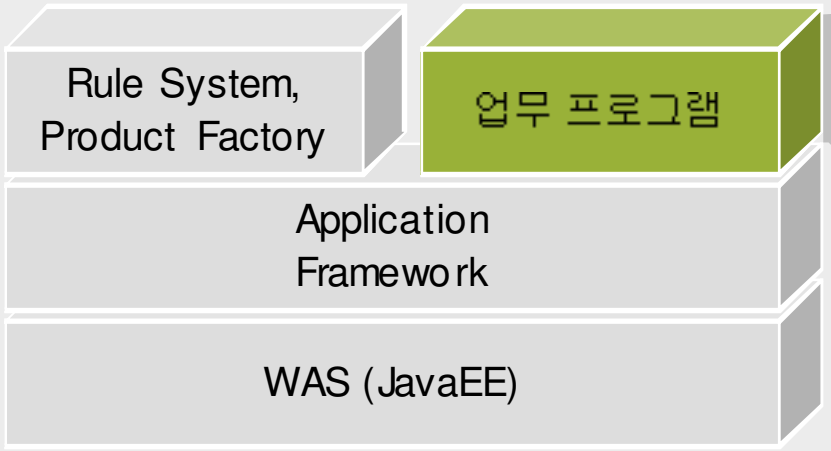


# 분리의 한계

기술 분리는 복잡성을 해결하지만 분리물은 여전히 남아 있음. 또한 업무프로그램은 여전히 기술과 업무가 분리되지 않고 있음

[ 지난 노력들의 결과

]




복잡성 해결

- 전문처리, 트랜잭션 처리 등의 기술/공통 처리가 간편해짐
- 프로그램 수정 없이도 변화가 많은 업무규칙 및 신상품 출시에 대응하기 쉬어짐

[ 지난 노력들의 한계

]

Framework 등 알아야 할 것이 여전히 많다.

개발자/유지보수 담당자

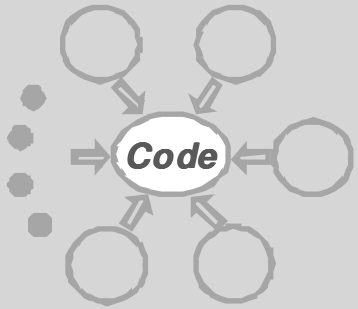
업무요건은 여전히 Java(프로그램언어)와 SQL로 구현해야 한다.

# 금융IT의 새로운 길 : 모델중심 개발

분리의 한계를 넘어서기 위해서

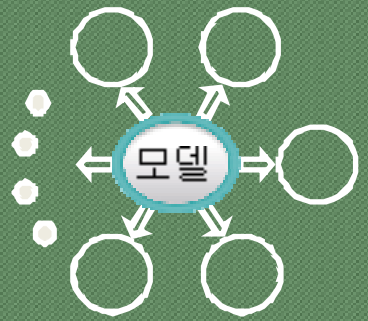
## 우리의 관심에서 소스코드를 사라지게 하자

### Code Centric

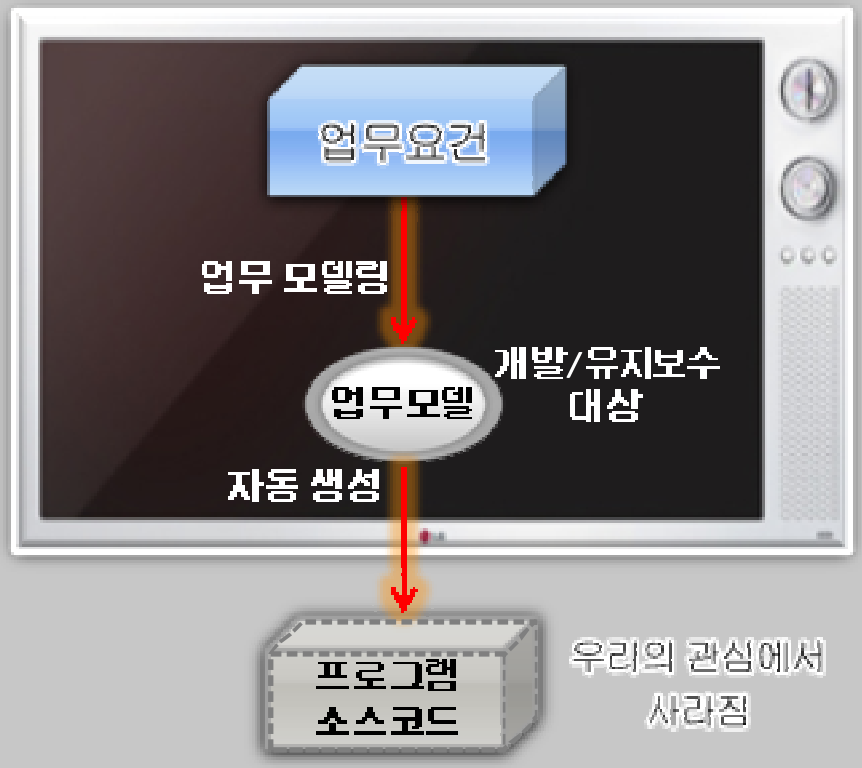



SW 이해 및 변경을 코드 중심으로 수행

### Model Driven




기술적 요소와 상관없이 업무 요건을 담고 있는 업무모델을 중심으로 개발/유지보수



# MDD 소개 동영상

# MDD 프로세스


분석단계
설계단계
테스트단계
N회 반복
논리모델
UML : 시퀀스 다이어그램
DSL : 논리표현식
용어표준
코드생성
자동화도구
코드표준
소스코드
단위
테스트
UML : Unified modeling language
DSL : Domain specific language

# 왜 이제야 MDD인가?

1884년 : 토마스 파커

1996년 : GM EV1
– 주행거리 120Km

2012년 : 테슬라 모델 S
- 주행거리 426Km


장애 해결(비용, 주행거리, 충전)
가능성 높아짐

# 왜 이제야 MDD인가?

## CASE : Computer-aided software engineering, 1990년대

- 표준의 부족 : CASE Tool간 호환 어려움
- 단기적 성과 부재 : 높은 초기 투자비용 대비 단기 성과 미미
- 장기간 숙련 필요 : 단기간에 숙련이 어려워 프로젝트 적용 기피

## MDA 표준 스펙 발표 by OMG, 2001년

- 아키텍처 표준

## Eclipse 도구 플랫폼 배포, 2004년

- 도구 표준

## MDD : Model-driven development, 현재

- 표준 존재 : OMG MDA, Eclipse
- 단기적 성과 창출 : 합리적 초기 투자비용, 단기 성과 향상
- 숙련 기간 단축 : 도구 편의성 향상
- 기술 성숙 : 모델 중심의 완전한 설계, 소스생성 및 디버깅

# MDD 적용 사례(1/2)

[ 전북은행 프로젝트 개요 ]

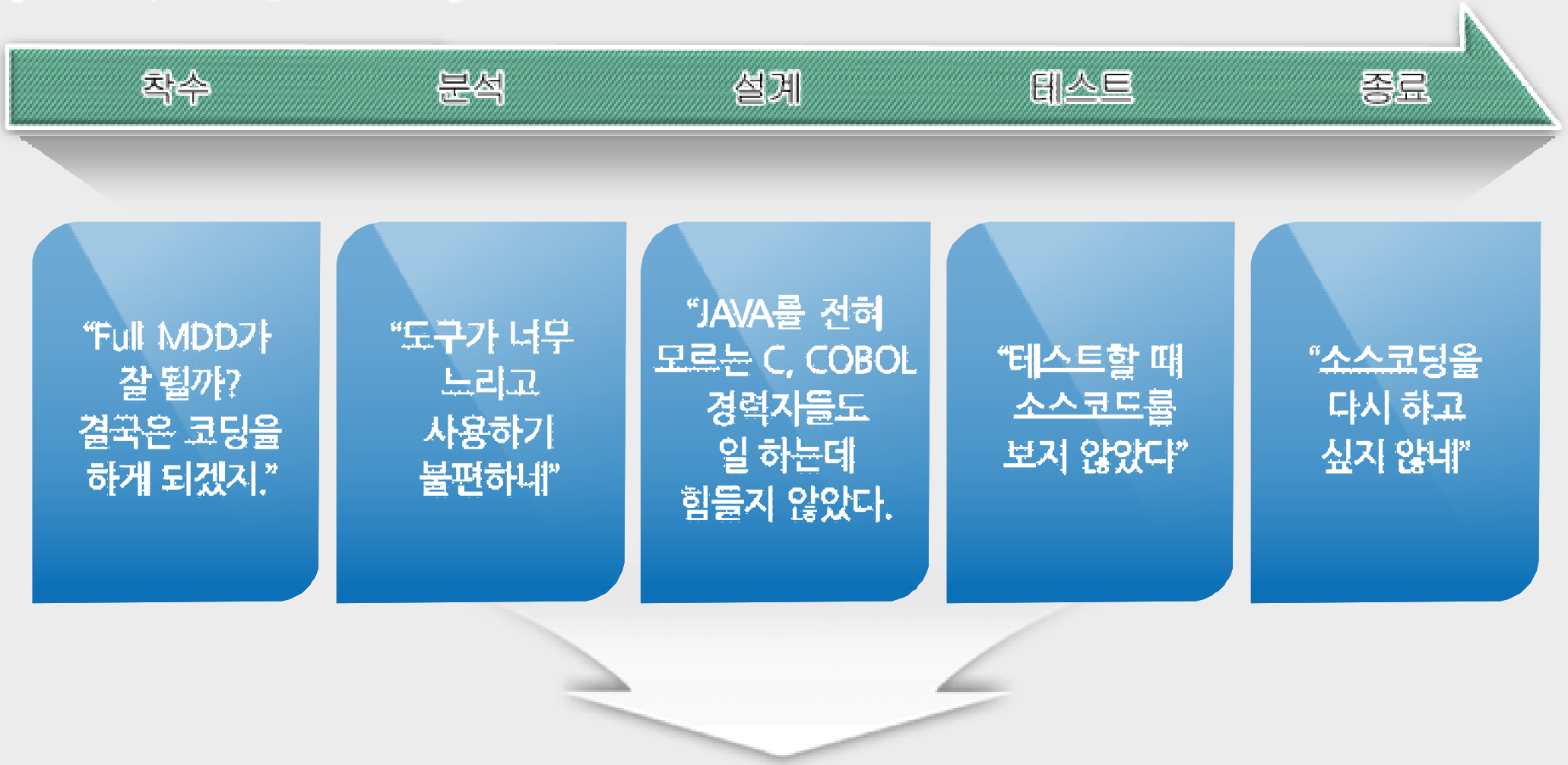
| 구축기간 | 수행범위 | 특징 | 기대효과 |
| --- | --- | --- | --- |
| 12년 2월 ~ 13년 9월 (20개월) | 차세대 뱅킹 시스템 구축 | 국내 최초 Java 기반 은행 기간계 차세대 | 소스 자동생성을 통한 품질 및 생산성 향상 |
| 납기보다 1주일 앞서 오픈 | 시스템간 인터페이스 구축 | DevOn MDA 적용을 통한 100% 소스 자동 생성 (온라인, 배치, 센터컷) | 모델과 소스의 일치로 향후 유지보수 및 안정화 기여 |
| | 인프라 아키텍처 구축 | | |

성공적 구축

"금융 시스템 차세대에 본격적으로 적용할 정도로 성숙"

# 국내 MDD 적용 사례(2/2)

[ 프로젝트 현장의 소리 ]




"현장의 개발자들도 만족할 정도로 성숙"

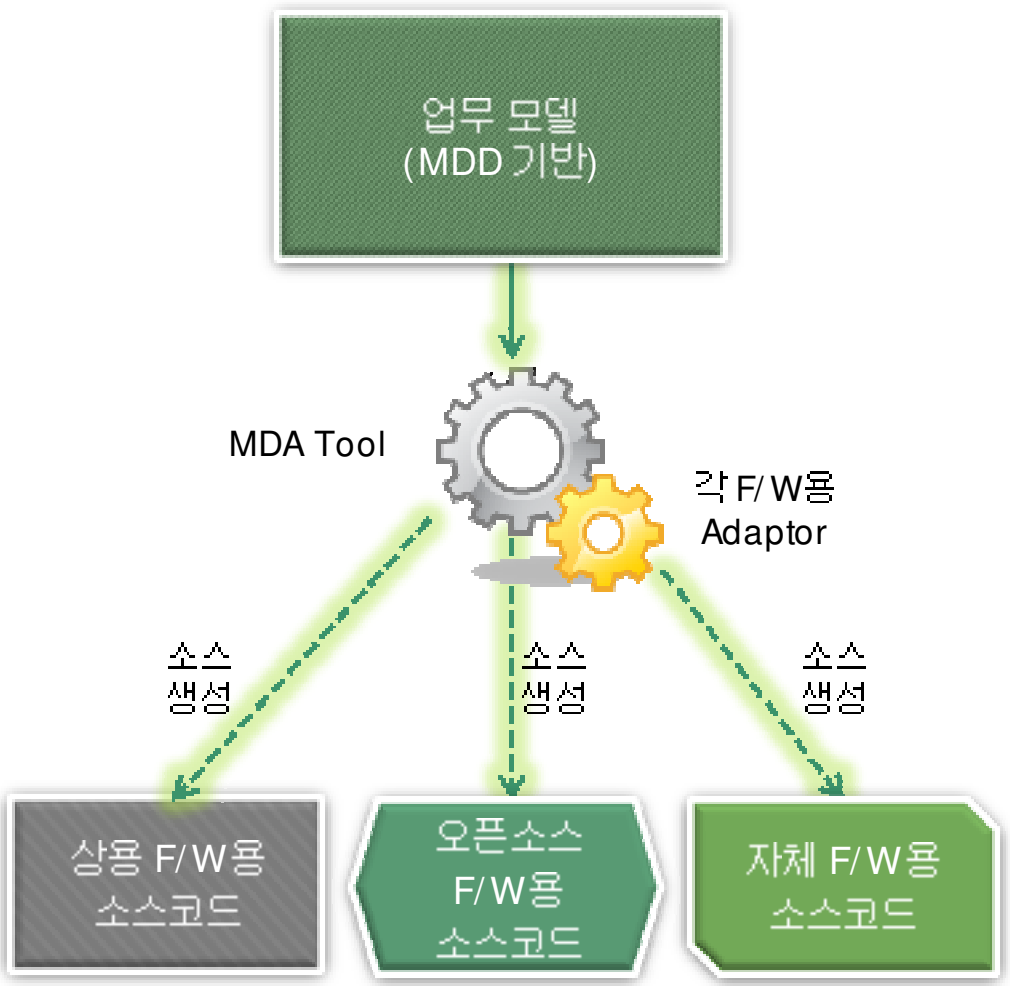
# 변화들

| 구분 | Code-centric | Model-driven |
| --- | --- | --- |
| 코딩 | 수작업 | 자동 |
| 산출물 | 수작업 | 자동 |
| 공정 | 개발 | 설계 |
| 자산 | 소스 | 모델 |
| 전문성 | 언어 | 업무 |
| 문자 | 알파벳 | 한글 |
| 표현 | 문자 | 그림 |
| 플랫폼 | 의존적 | 독립적 |

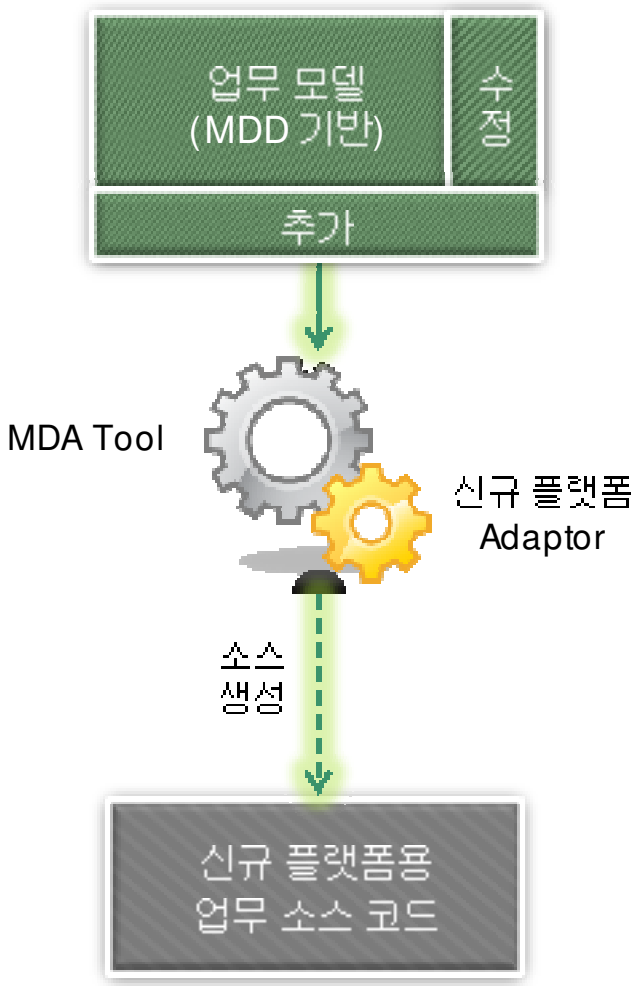
기대효과

# 기술변화에 유연 : 유연성과 이식성

## Framework 변화에 유연




## 모델 재사용



# 품질 향상 : 이해 용이성, 의사소통 명확성

코드

모델(다이어그램)

사용 용어

알파벳 약어
(retrieveJntNetwkFileTpcdList)

VS

한글 표준용어
(공동망파일구분코드목록조회)

처리 흐름

모듈 처리 흐름, 데이터 조작,
계산식 혼재

VS

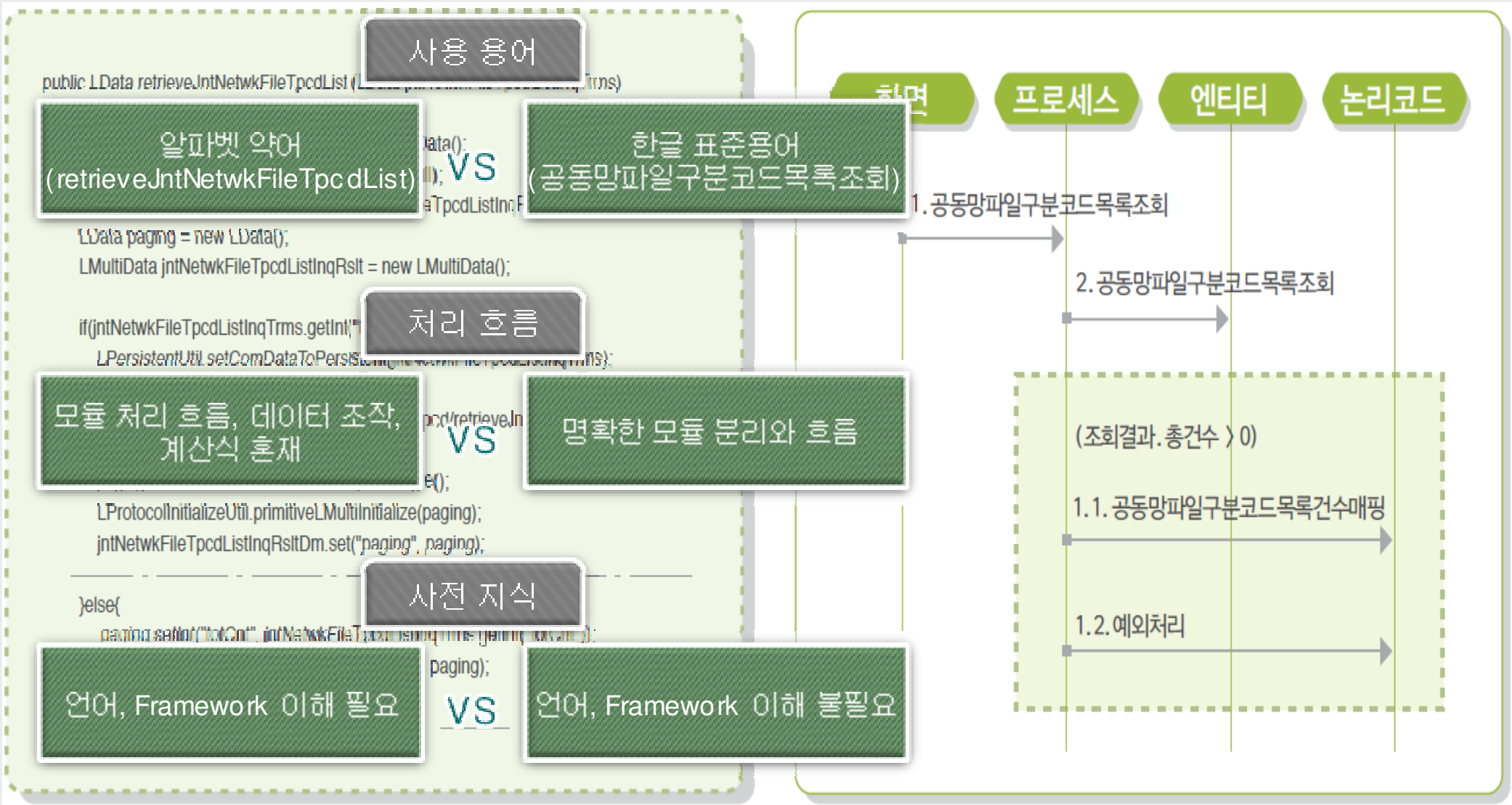
명확한 모듈 분리와 흐름

사전 지식

언어, Framework 이해 필요

VS

언어, Framework 이해 불필요



# 품질 향상 : 표준화

## 코드중심

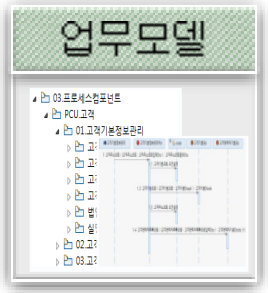

## MDD

**용어 표준 사용여부 체크**

- 도메인 : ex) 일자, 금액, 코드
- 단어 : ex) 고객, 계약
- 용어 : ex) 계약일자, 고객코드

용어 표준

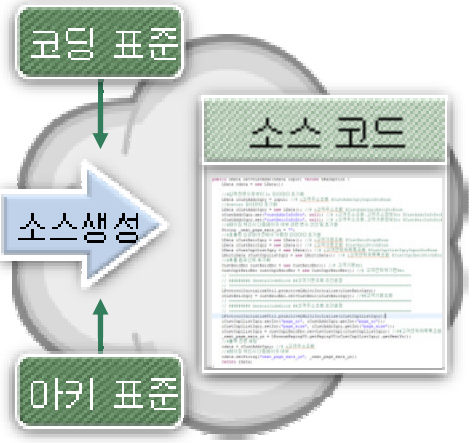
모델표준




**모델 표준 사전 점검 항목**

- 컴포넌트 호출 규칙 (프로세스 → 곰들 → 데이터 등)
- 컴포넌트, 오퍼레이션 등 명명 규칙
- 표준 용어 적용 여부 등

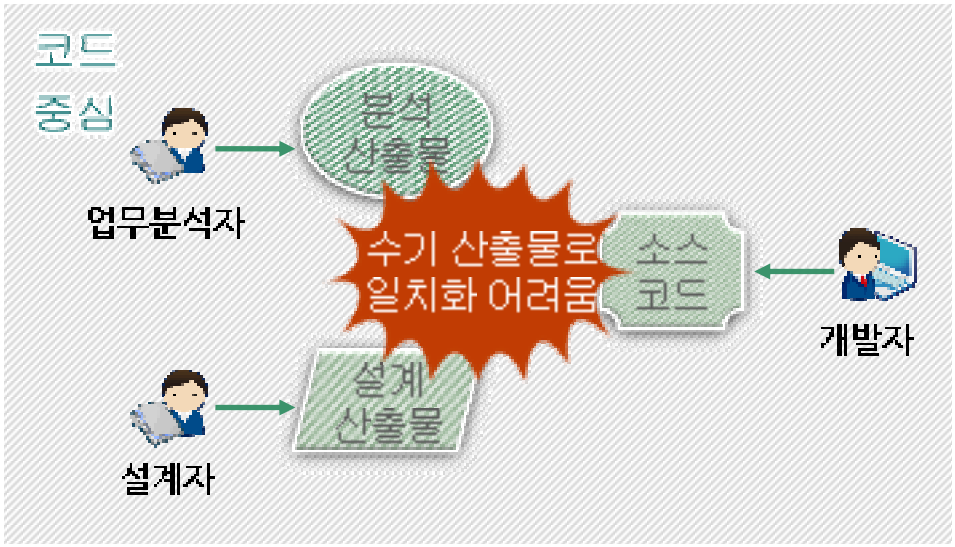
코딩 표준 자동 반영

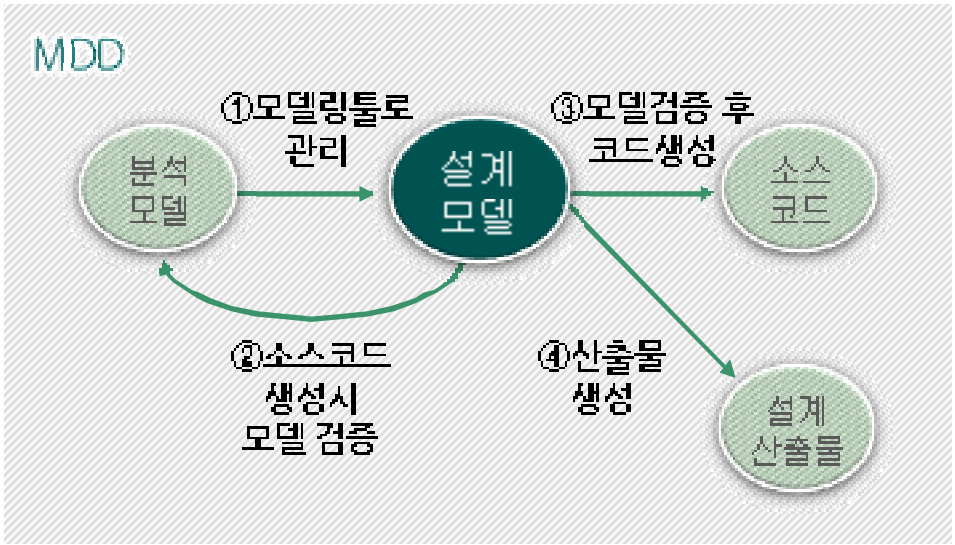



아키 표준 자동 반영

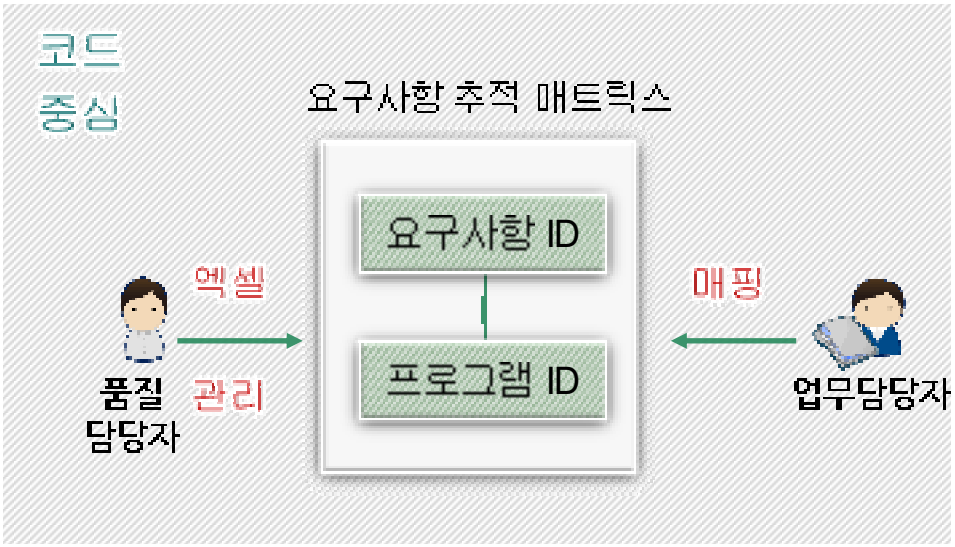
# 품질 향상 : 일치성, 추적성

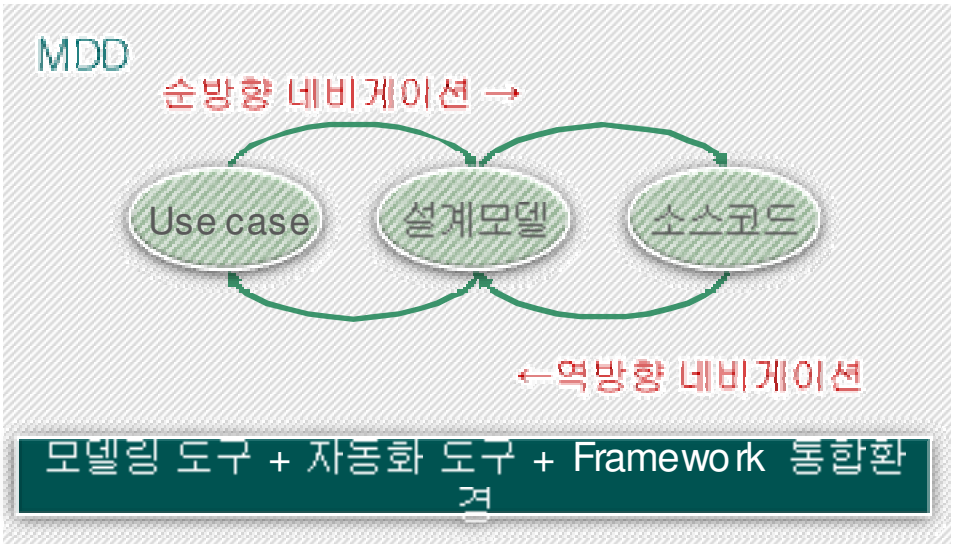
## 왜 산출물과 소스코드가 같지 않은가?




## 요구사항을 잘 반영했는지 어떻게 아나?



# 생산성 향상

## ✔ 어플리케이션 개발 시간 50% 단축을 기대한다.

***Air France*** Does not yet have sufficient experience to fully demonstrate the advantage of its approach ;
However, it anticipates many benefits :

줄탁…

A decrease of up to 50% in the time it takes developers to create applications.
실탁…

## ✔ 개발시간을 반 또는 1/4 단축한다.

Reduced project costs and time-to-market

“ MDD allows us to halve or quarter development time compared with a traditional approach”
***( Fiducia, financial service firm)***

## ✔ 개발자 생산성이 1M/M 당 코드라인 수 기준으로 50% ~ 90% 높다.

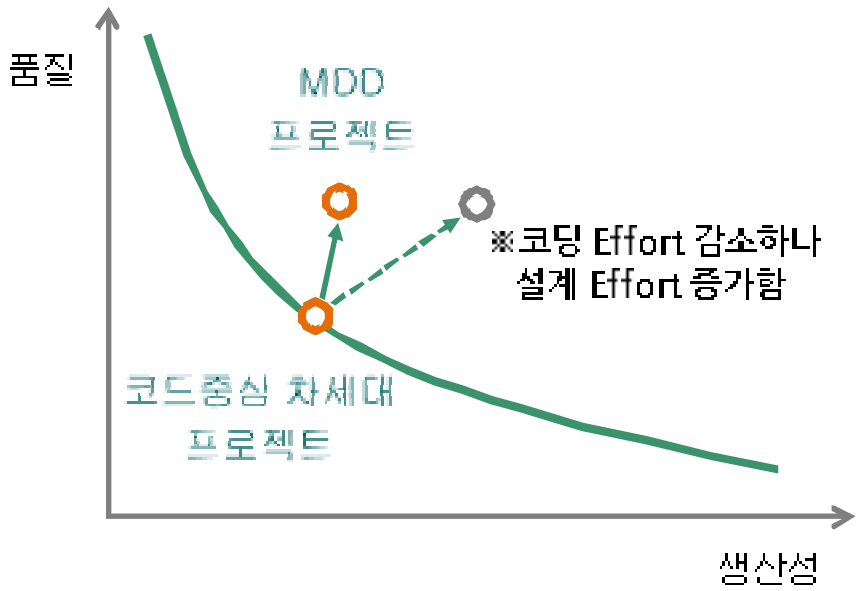
Increased application developer productivity

“Our clients in the embedded , real time, and high performance systems market have increased productivity by 50% to 90%, as measured in sources lines of code per person per month.”
***( Pathfinder, technology vender)***

포레스터 리서치 : 2007. 2008

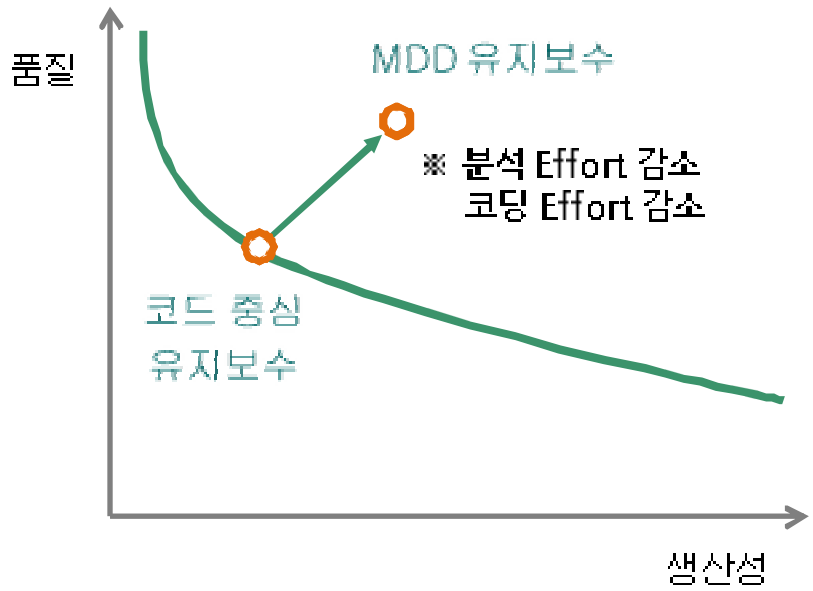
# 생산성/품질 향상 : 개발 vs 유지보수

## 개발 프로젝트 품질 향상



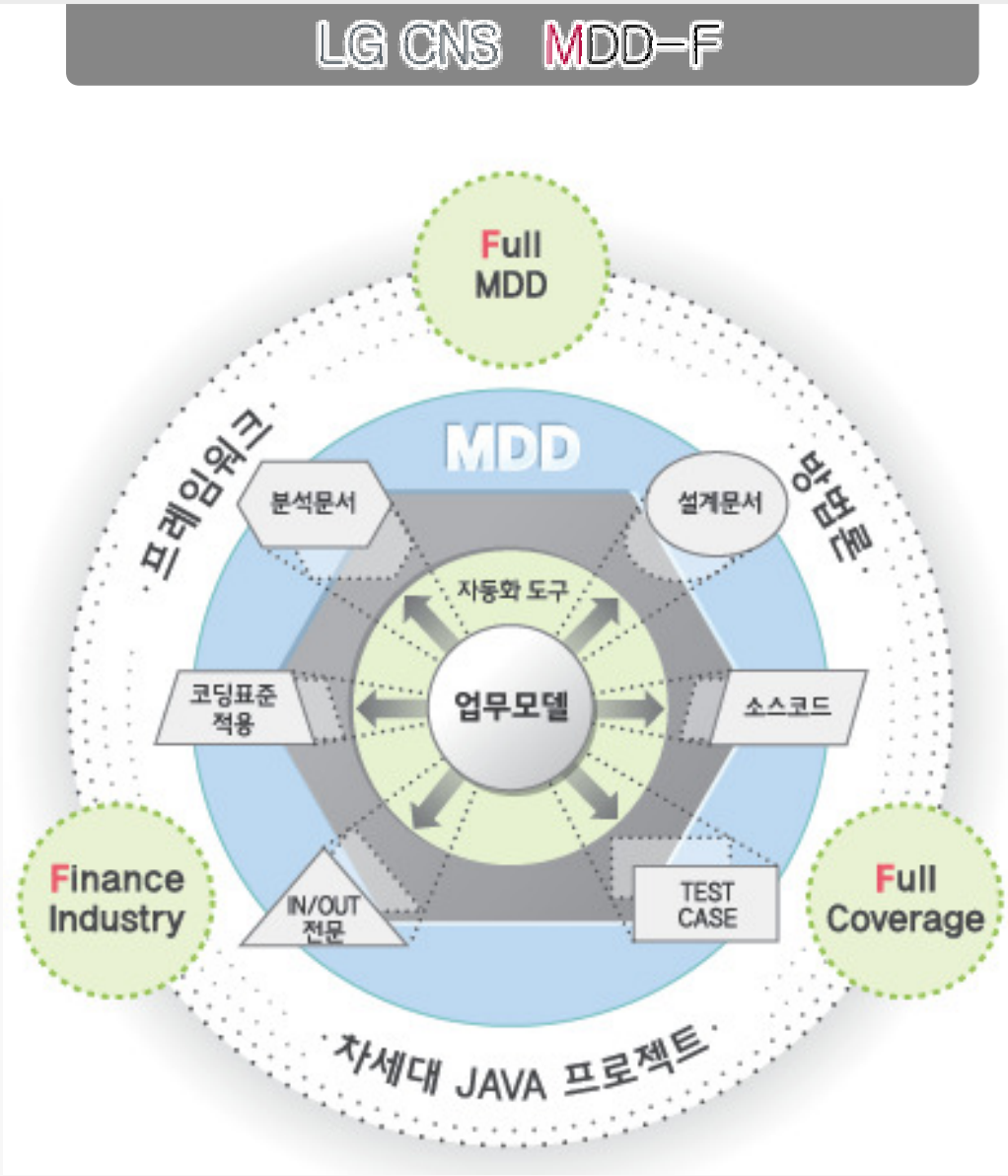

- 코드중심 : 생산성과 품질의 균형 모색
  - 높은 수준의 품질을 달성하기 위한 비용 부담이 큼
- MDD : 품질 대폭 향상, 생산성 약간 향상
  - 산출물, 소스코드 자동생성으로 인한 품질 향상 및 생산성 향상

## 유지보수 단계 품질/생산성 향상




- 개발 프로젝트와 달리 생산성도 향상
  - 업무분석이 용이함 : 상담, 분석시간 단축
  - 소스코드 자동생성 : 코딩 시간 대폭 감소

# LG CNS 모델중심개발




## LG CNS MDD-F란?

- Model-driven development for finance industry
- 방법론, 자동화 도구, 프레임워크가 결합된 개발 체계

## F의 의미

- Full MDD : 소스코드 100% 자동생성
- Full Coverage : 온라인, 배치, C/C

## 구성요소

- 방법론 : UML 및 DSL 기반 모델링 방법
- 자동화 도구 : 모델로부터 소스코드와 산출물 생성
- Framework : 금융 특화 framework

# 요약

- 금융 시스템 차세대에 본격적으로 적용할 정도로 성숙
- 개발 공정에서는 품질 향상, 유지보수 단계에서는 품질에 더해 생산성 향상도 기대
- “기술중심에서 업무중심으로” 일하는 방식이 변화

# 감사합니다!

LG CNS 금융플랫폼팀장 최종태 부장

jtchoi@lgcns.com

010-7640-4993