S/W 개발의 새로운길

- MDD 모델중심 개발 -

LG CNS

목 차

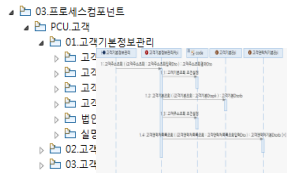
MDD(Model Driven Development) 개념(1/2)

소프트웨어 개발 방식은 기계 중심적에서 인간이 이해하기 쉬운 방향으로 진화하고 있으며 현재는 Model 단계로 진입하고 있습니다.

S/W 개발 방식의 변환

Model Driven Architecture
(2000년 OMG에서 제정)

Model Driven Engineering
(1990년 연구시작)



추상화Level (이해용이성)

Next

C / Java

COBOL

두 번째

Assembly

첫 번째

Binary

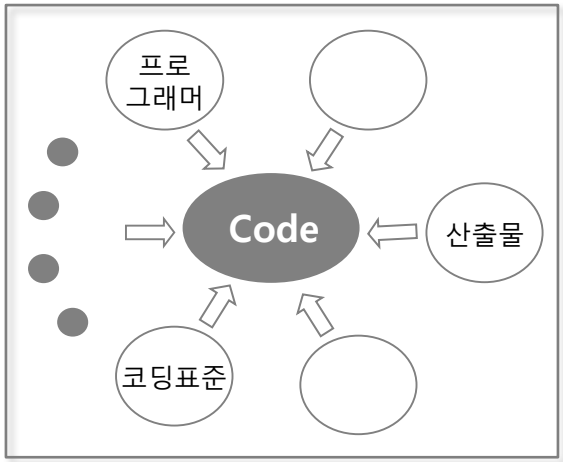
```
public IData recvCustAddr(IData input) throws IException {
    IData rdata = new IData();

    //입력전문으로부터 In 파라미터 초기화
    IData iCustAddrIqry = input; //# 고객주소조회 ECustAddrIqryInputDtoEnum
    //#return 파라미터 초기화
    IData oCustAddrIqry = new IData(); //# 고객주소조회 ECustAddrIqryOutDtoEnum
    oCustAddrIqry.set("custAddrInfoDto", null); //# 고객주소조회.고객주소정보Dto ECustAddrInfoDtoEnum
    oCustAddrIqry.set("custBasicInfoDto", null); //# 고객주소조회.고객기본정보Dto ECustBasicInfoDtoEnum
    //#페이징 처리시 다음페이지 여부 관련 변수 선언 및 초기화
    String next_page_exis_yn = "";
    //#호출된 오퍼레이션에 서 사용된 파라미터 초기화
    IData iCustBasicIqry = new IData(); //# 고객기본조회 ECustBasicIqryEnum
    IData oCustBasicIqry = new IData(); //# 고객기본조회 ECustBasicOutDtoEnum
    IData iCustCopliListIqry = new IData(); //# 고객연락처목록조회 ECustCopliListIqryInputDtoEnum
    IMultiData oCustCopliListIqry = new IMultiData(); //# 고객연락처목록조회 ECustCopliBasicOutDtoEnu
    //#호출 컴포넌트 초기화
```

a = 5 + 1;

```
Data
var1 DWORD 1
var2 DWORD 5
.code
MOV eax, var1
ADD eax, var2
```

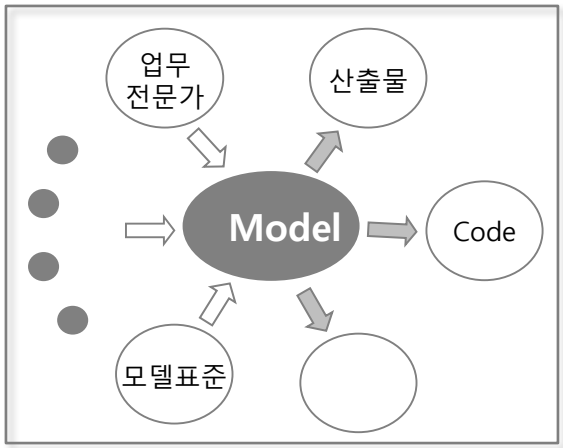
```
01010101011101010101010
10101011000001000111
```

Code VS Model



Code Centric Development

- 고품질 소스 코드를 위한 집중
  - 코딩 잘하는 프로그래머
  - 고품질 소스를 위한 산출물
  - 고품질 소스를 위한 코딩표준



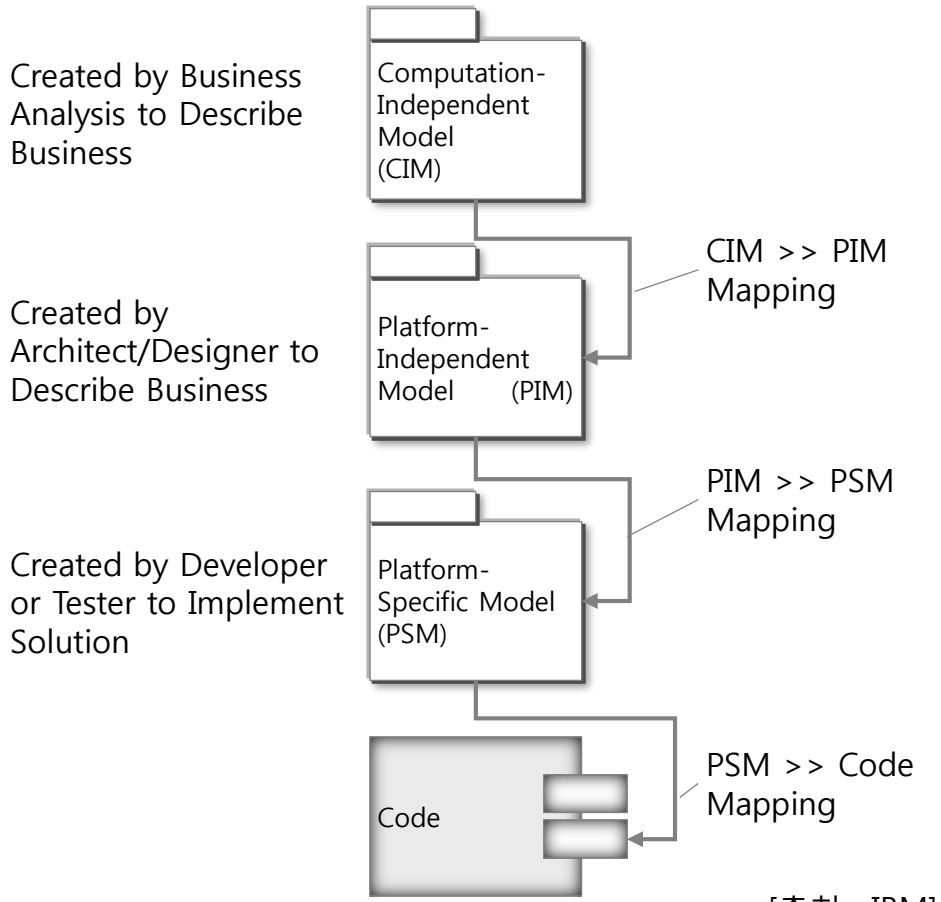
Model Driven Development

- 고품질 모델 집중을 통한 고품질 소스 생성
  - 프로그래머 보다 업무전문가
  - 코드 집중 보다 모델 집중
  - 코딩 표준 보다 모델표준

MDD(Model Driven Development) 개념(2/2)

MDD기반 개발시 주요기술은 OMG에서 정의한 MDA이며 핵심개념은 모델분리와 모델간 매핑을 통한 기술변화 유연성, 신속성입니다.

모델 단계와 소스 변환 절차



[출처 : IBM]

MDA (Model-Driven Architecture)™

MDD 사상을 실제 프로젝트에서 효과적으로 적용하기 위해 OMG에서 정의한 제반 기술 표준 및 기술 기반

* **OMG** (Object Management Group) : 1989년 3Com, HP, Sun Microsystems, Unisys 등 8개 회사가 주축이 되어 소프트웨어 산업에서 특정 Vendor에 의존하지 않는 객체관련 기술을 표준화하는 비영리단체

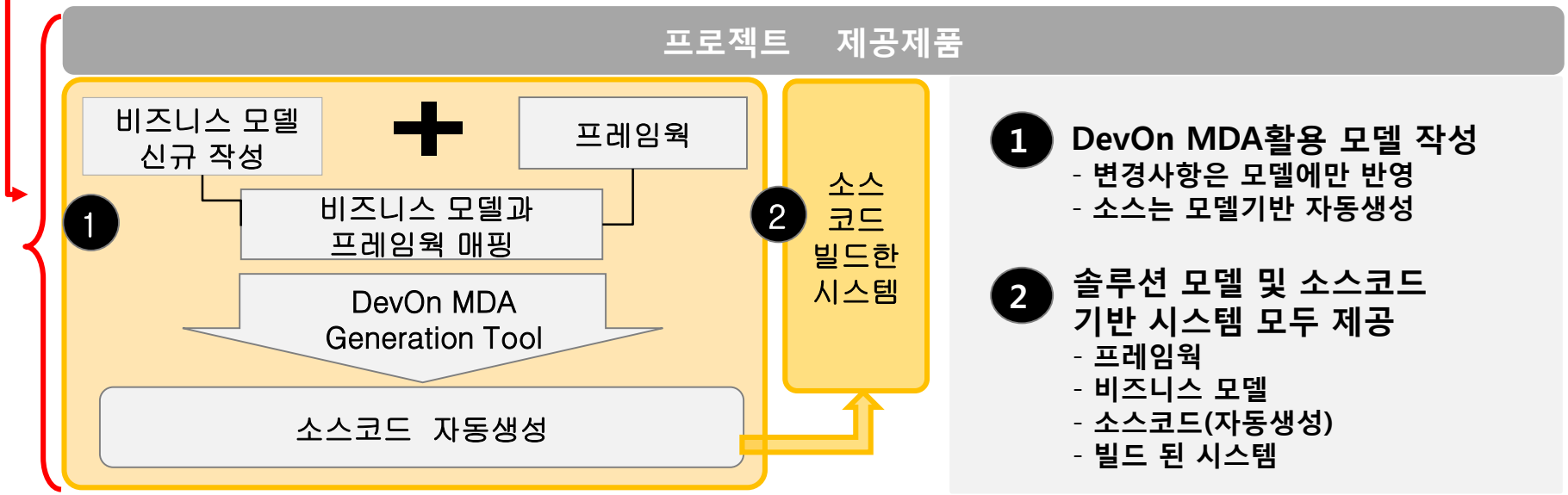
MDA 핵심 개념

매핑 구분	매핑 핵심 기술		설명
	표준화	자동화	
CIM >> PIM	모델표준		CIM(현업용)에서 - 연계시스템 반영, - 관련S/W 매핑
PIM >> PSM	용어표준 아키표준	자동매핑 산출물생성	PIM에서 - 언어특징반영(J2EE, .Net등) - 모델 영어로 변경(한국내)
PSM >> code	코딩표준	소스생성 테스트전문생성	PSM(물리모델)에서 - 해당 언어로 코딩

MDD 적용 – MDD적용수준

MDD적용방식은 적용 수준에 따라 3단계로 구분되며 신한카드의 경우 2단계, 전북은행의 경우 3단계가 적용된 사례입니다.

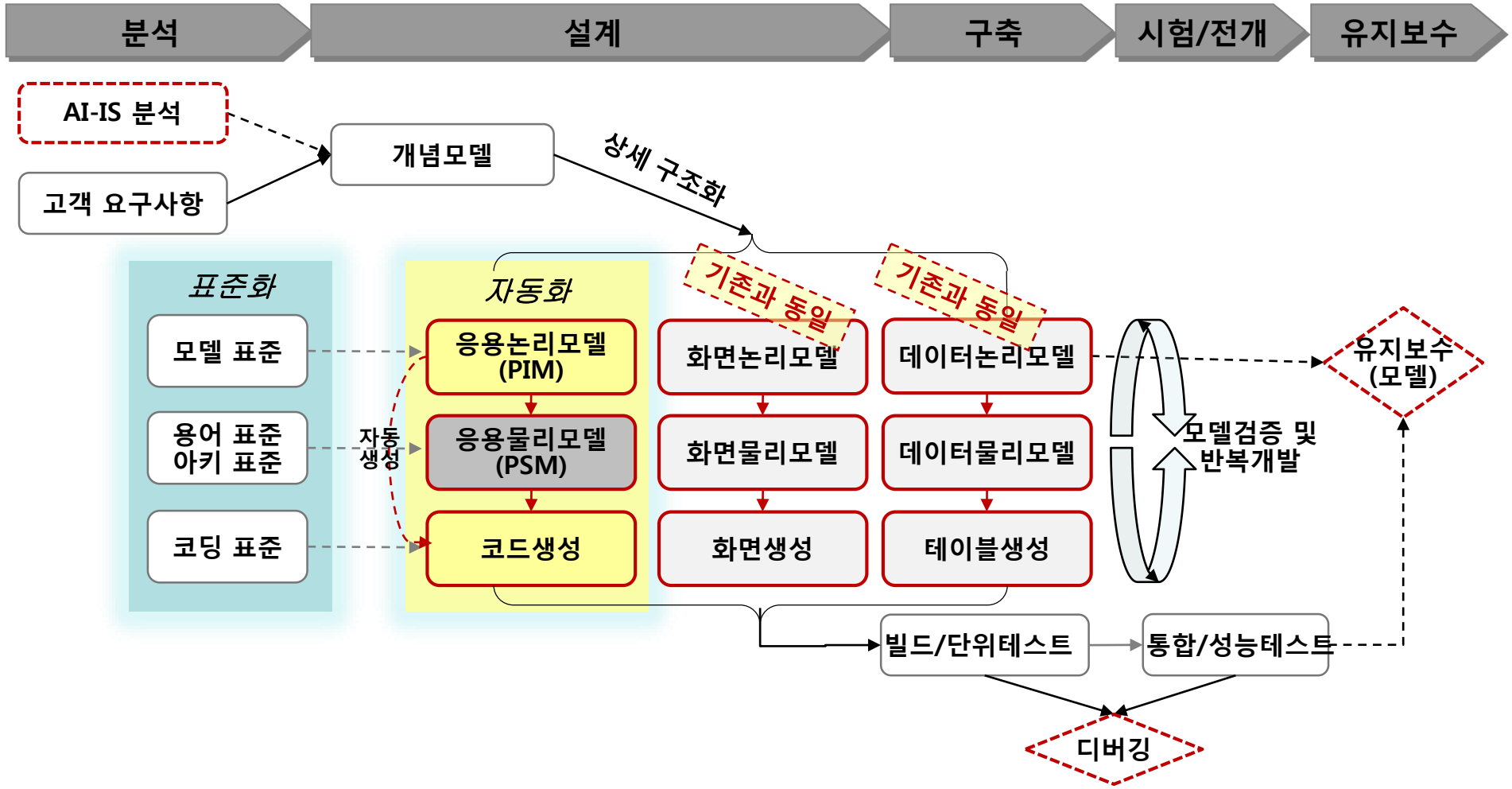
구분	수준	비고
Level 1	**모델링 도구를 활용**하여 모델링을 수행하는 수준	모델링 도구를 활용하는 대부분의 프로젝트가 해당 소스코드의 개발이 전적으로 사람에게 의존
Level 2 **(신한카드)**	모델로부터 **소스코드를 부분적으로 자동생성** (20~30%의 프로그램의 큰 흐름 대상)	운영단계에서 소스를 수정하며 모델과 소스가 이중 관리되며 이를 통해 모델-소스의 일치성 보장이 어려움
Level 3 **(전북은행)**	**모델로부터 100% 소스와 산출물을 생성**하는 수준	관리는 모델로 일원화되고 모델-소스-산출물 일치



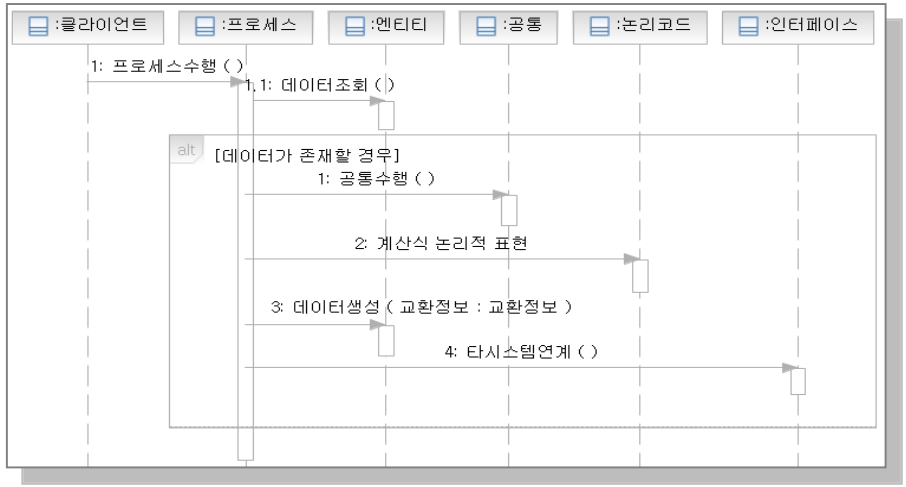
MDD적용 - 3단계 수준의 MDD 적용 방식

MDD 적용은 설계단계부터 구축단계에 걸쳐 응용분야의 표준화와 자동화 기반으로 반복적 개발을 통한 고품질,고효율 S/W를 개발 합니다

프로젝트에서 MDD 적용



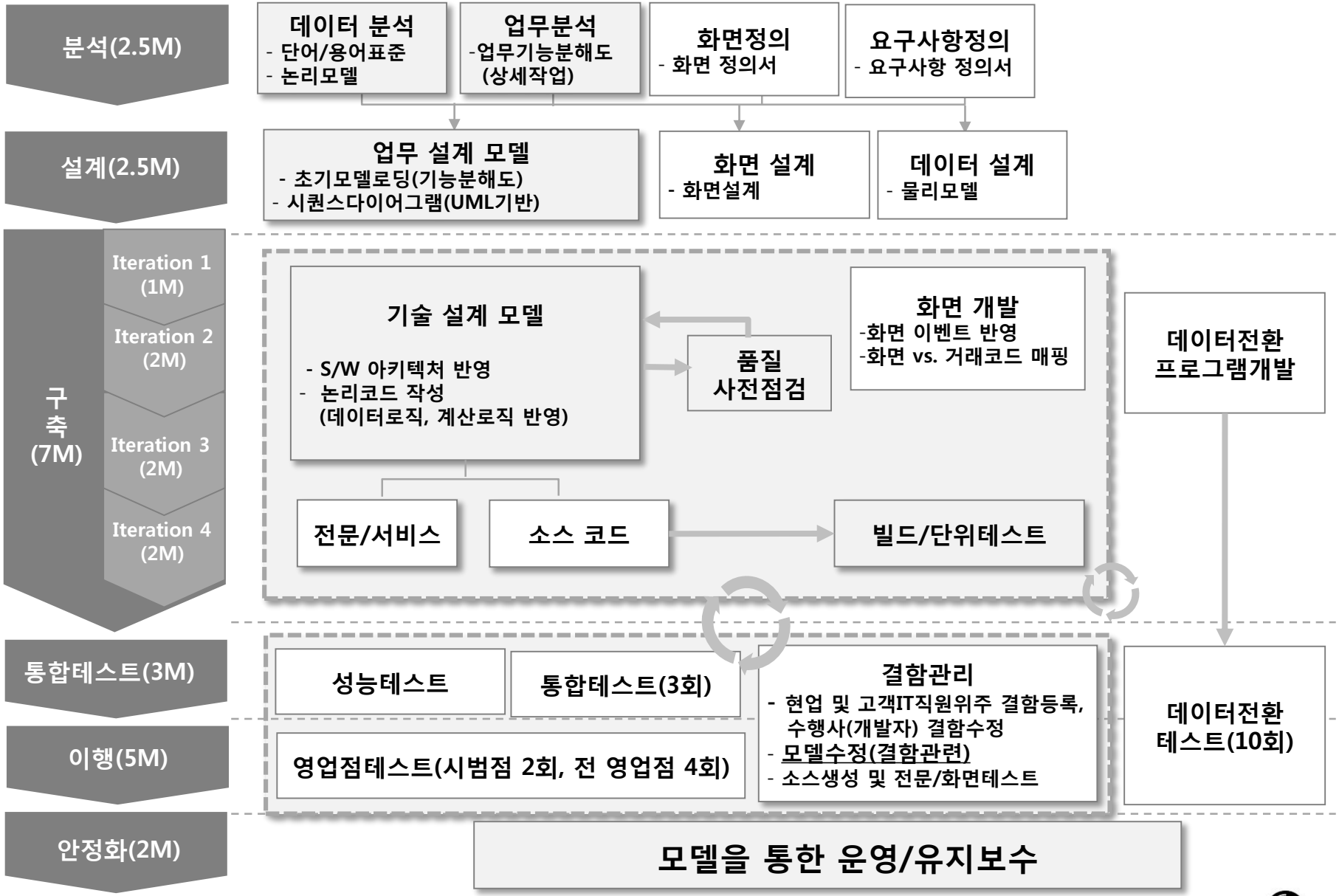
MDD적용 - 논리모델

구분	사례	설명	비고
전체 로직		**전체 프로세스 로직은 모델로 표현함.** **- 다른 모듈 호출** **- 분기/반복 등의 제어** **- 전체적인 수행 순서**	기존 방식 작성 수준 → 이후 물리 설계와 코딩은 수작업
데이터 처리	[개인고객정보] [t_개인고객정보]; [t_개인고객정보] = [개인고객처리더미].[개인고객정보];	**데이터 처리는 논리 표현식으로 표현함** **- 지역변수 선언** **- getter, setter 표현**	모델기반 개발방식 추가 작성 → 이후 물리 설계와 코딩 자동화
계산식	[이자계산일수] = [일수정보].[기간일수]; [적용금리] = [여신계좌금리조회결과].[적용금리] [정상이자] = [대출잔액]*[이자계산일수]*[적용금리]/365/100;	**각종 계산식은 논리 표현식으로 표현함**	

전북은행 사례 - 개요

사 업 명	**전북은행 차세대 시스템 구축 프로젝트**
사업 기간	**2012.02.01 ~ 2013.11.29 (22개월 – 안정화 2개월 포함)**
사업범위	**- 차세대 Core Banking 시스템 구축** 공통, 수신, 여신, 외환, 카드, 계리, 대외접속, 대행부대, 전자금융, 예산자산, 정보계 등 **- Core banking 연관시스템 고도화** 인사관리, 경영관리, 유가증권 시스템 구축 등
사업환경 및 특성	▪ **국내 최초 제1금융권 Java기반 차세대시스템 구축** ▪ **기술 인력 소싱의 어려움** - 은행권 사업은 대부분 C기반 – Java기술을 보유한 은행업무전문가/개발자가 거의 없음 (프로젝트 투입인력 중 Java기술 유경험자 비율 : 6.4%) - 고객사 IT인력 중 JAVA 경험인력이 없음 ▪ **LG CNS : 기존의 경험자산 존재** - 금융 차세대구축 경험 업무 노하우 + MDD기반의 Banking Asset보유 **LG CNS의 뱅킹 솔루션을 참조 + Full MDD방식 적용 (적용level3)**
적용효과	▪ **품질향상** - 오픈 후 결함율 0.003개/FP - 모델의 통제성을 활용한 프로젝트 기본 품질 확보 ▪ **기술 독립성 확보** - COBOL 및 C 기술 기반의 운영인력 → MDD기반의 Java활용 운영인력으로 자연스럽게 전환 ▪ **생산성 향상** - 프로젝트 생산성 향상으로 1week 조기 open및 조기 안정화 (기존의 은행차세대 프로젝트 대비 개발생산성 6.13% 향상)

전북은행 사례 – 프로젝트 공정



S/W 개발시 MDD 기대효과

MDD방식으로 S/W 개발시 표준화로 품질 향상, 자동화로 개발자 효율향상, 가시화로 의사소통 향상의 효과가 기대 됩니다.

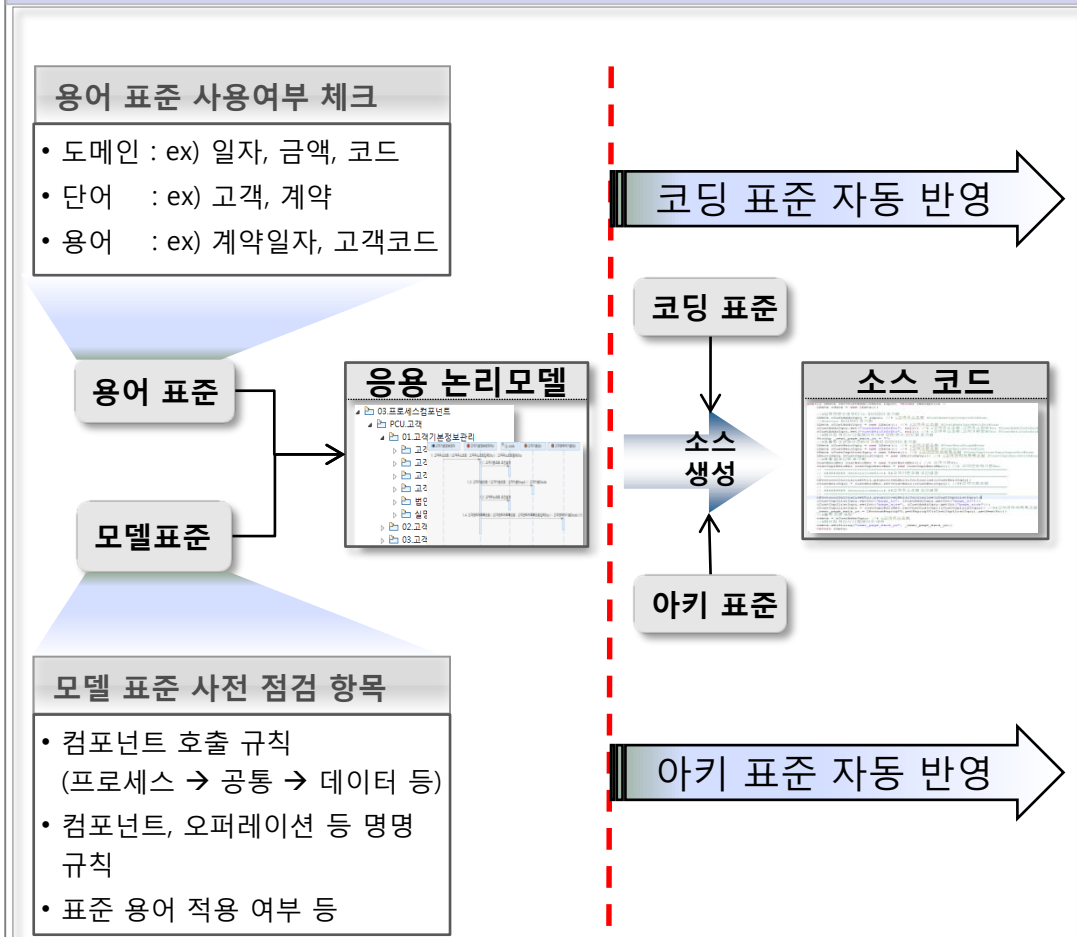
MDD S/W 개발시 기대효과

	S/W개발시 현황 및 문제점	*MDD 핵심 요소*	*기 대 효 과*
표준화	• 표준 미존재 보다 미준수 확인이 곤란 • 표준 변경시 일관성 있는 반영 곤란 • 체계적 설계 표준 부재(코딩에 집중)	**모델 표준화** **용어 표준화** **코딩 표준화** **아키텍처 표준화**	**품질 상향 평준화** • 모델만으로 표준 준수 여부 확인 • 모델반영을 통해서만 표준 추가/변경
자동화	• 선코딩후 산물출 생성에 시간 과다 소요 • 신기술에 미숙한 금융업무 전문가 참여 필수	**표준 준수 자동화** **산출물 생성 자동화** **소스 생성 자동화**	**개발자 효율 향상** • 문서작업 보다 설계/코딩에 집중 • 경험많은 업무전문가 적극참여 유도
가시화	• 문서 보다는 말을 통한 의사소통 • 코딩상의 단순한 오류 빈번히 발생	**다이어그램 을통한…** **자연어(한글) 을통한…**	**의사 소통 향상** • 체계적 모델을 통한 설계자 개발자간 소통 • 자연어(한글)로 인한 단순오류 감소

MDD를 통한 표준화 – 표준 준수 및 모델 점검

MDD방식은 표준 제정보다 표준 준수를 강제화 하여 일관되고 신속하게 대응하게 합니다.

표준 적용 및 모델 점검



기대효과 – 품질 상향 평준화

표준 준수 강제화

- 메타 연계를 통한 용어 표준 준수
- 모델 점검기능을 통한 모델 표준 준수
- 자동 생성시 코딩,아키 표준 자동 반영

표준변경시 신속하고 유연한 대응

- PGM 유형 변경/추가 시 표준 변경으로 일괄 대응
- 성능 문제 발생시 각표준 수정후 일괄 생성으로 신속 대응(공통모듈)

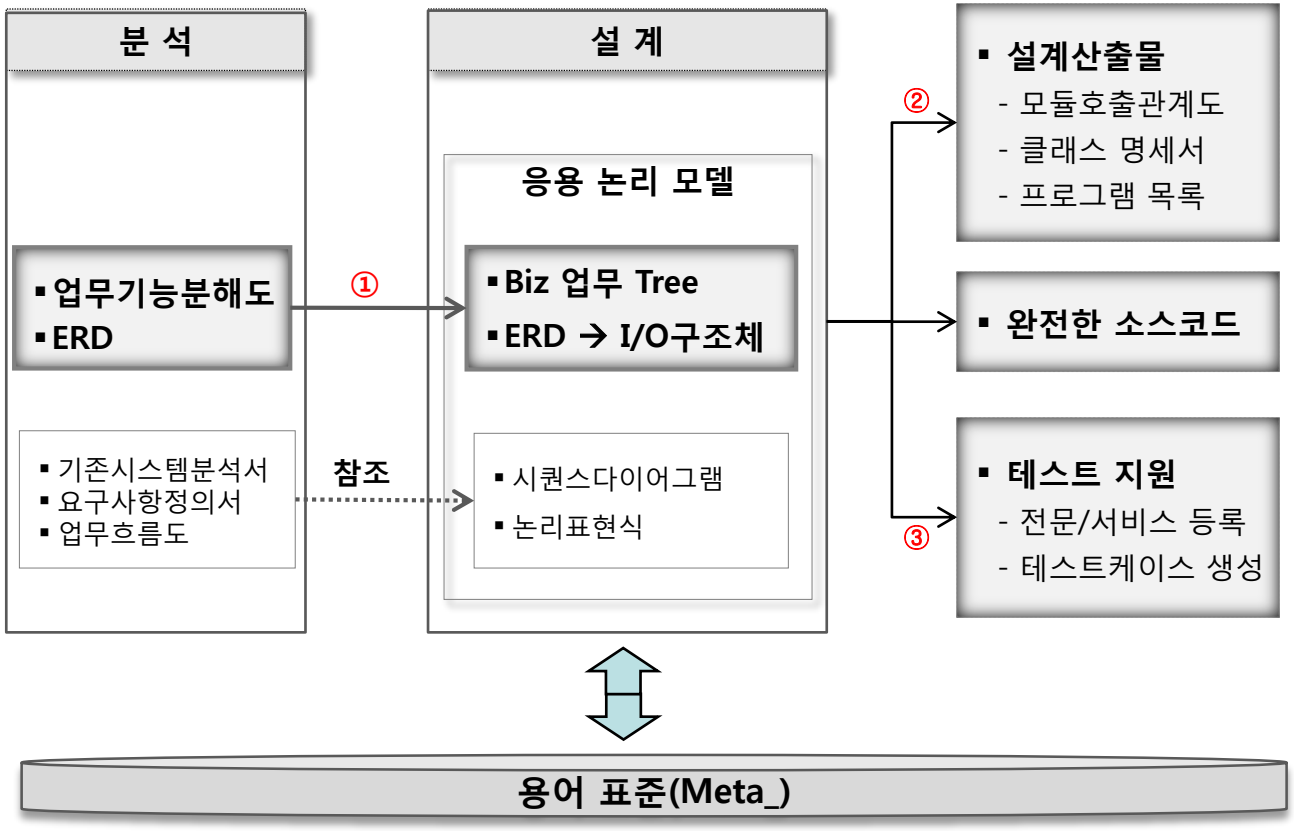
표준화된 모델로 동일 설계 품질 유지

- 유형별 설계모델패턴 표준화
  ex) 온라인유형, 배치 유형등

MDD를 통한 자동화 – 모델/산출물 자동 생성

완전한 소스생성은 기본이며 추가적으로 분석단계 산출물을 이용한 모델 자동생성, 테스트 단계 서비스/전문 자동 등록을 통해 작업효율을 극대화할 수 있습니다.

소스코드 및 산출물 자동생성



기대효과 – 개발자 효율 향상

① 모델 자동생성을 통한 효율화

- 분석 산출물 Loading하여 초기 모델 자동생성(컴포넌트, DTO)
  → 수작업 Effort 감소

② 산출물 자동생성을 통한 효율화

- 소스코드와 일치된 최신 산출물 수시 생성
  → 모델, 소스, 산출물이 항상 일치
  → 수작업 Effort 감소

③ 서비스/전문 자동생성을 통한 효율화

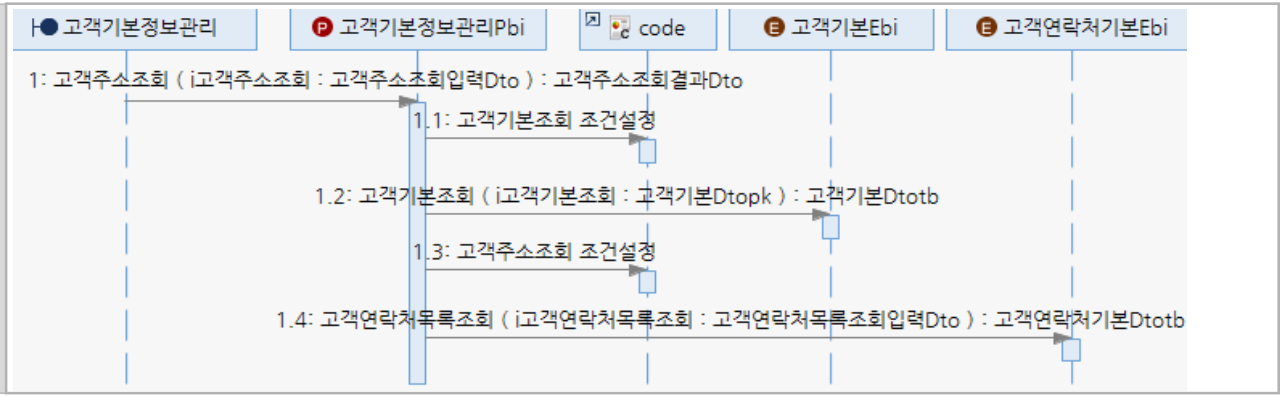
- 테스트를 위한 서비스/ 전문 자동 생성
  → 수작업 Effort 감소
  → 수작업 오류 감소

MDD를 통한 가시화 – Text가 아닌 다이어그램

업무 흐름이 Sequence Diagram으로 표현되어 Text 기반의 소스코드 보다 업무 흐름 파악이 용이하며, 프로그램 언어에 대한 지식이 없어도 업무 흐름 파악이 가능합니다.

시퀀스 다이어그램 vs 소스코드

시퀀스 다이어그램



소스 코드

```
public LData retvCustAddr(LData input) throws LException {
    LData rdata = new LData();

    //#입력전문으로부터 In 파라미터 초기화
    LData iCustAddrIqry = input; //# i고객주소조회 ECustAddrIqryInputDtoEnum
    //#return 파라미터 초기화
    LData rCustAddrIqry = new LData(); //# r고객주소조회 ECustAddrIqryRsltDtoEnum
    rCustAddrIqry.set("custAddrInfoDto", null); //# r고객주소조회.고객주소정보Dto E
    rCustAddrIqry.set("custBsicInfoDto", null); //# r고객주소조회.고객기본정보Dto E
    //#페이징 처리시 다음페이지 여부 관련 변수 선언 및 초기화
    String _next_page_exis_yn = "";
    //#호출된 오퍼레이션에서 사용된 파라미터 초기화
    LData iCustBsicIqry = new LData(); //# i고객기본조회 ECustBsicDtopkEnum
    LData rCustBsicIqry = new LData(); //# r고객기본조회 ECustBsicDtotbEnum
    LData iCustCnplListIqry = new LData(); //# i고객연락처목록조회 ECustCnplListIqr
    LMultiData rCustCnplListIqry = new LMultiData(); //# r고객연락처목록조회 ECustC
    //#호출 컴포넌트 초기화
```

```
CustBsicEbc custBsicEbc = new CustBsicEbc(); //# 고객기본Ebi
CustCnplBsicEbc custCnplBsicEbc = new CustCnplBsicEbc(); //# 고객연락처기본Ebi
// ================================================================
// ######## GeneralCodeBlock ##고객기본조회 조건설정
// ================================================================
LProtocolInitializeUtil.primitiveLMultiInitialize(iCustBsicIqry);
rCustBsicIqry = custBsicEbc.retvCustBsic(iCustBsicIqry); //##고객기본조회
// ================================================================
// ######## GeneralCodeBlock ##고객주소조회 조건설정
// ================================================================
LProtocolInitializeUtil.primitiveLMultiInitialize(iCustCnplListIqry);
iCustCnplListIqry.setInt("page_no", iCustAddrIqry.getInt("page_no"));
iCustCnplListIqry.setInt("page_size", iCustAddrIqry.getInt("page_size"));
rCustCnplListIqry = custCnplBsicEbc.retvLstCustCnpl(iCustCnplListIqry); //##고객연락처목록조회
_next_page_exis_yn = LRownumPagingVO.getPagingVO(rCustCnplListIqry).getNextYn();
//#출력 전문 세팅
rdata = rCustAddrIqry; //# r고객주소조회
//#페이징 처리시 다음페이지 여부
rdata.setString("next_page_exis_yn", _next_page_exis_yn);
return rdata;
```

기대효과 – 의사소통 향상

① 다이어그램을 통한 직관적 업무 파악 가능

② 신규 개발자 업무 인수인계 시 의사소통 원활

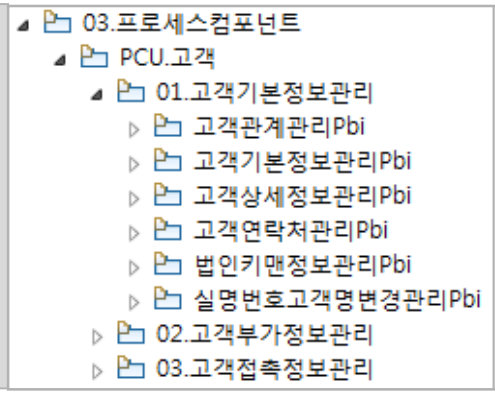
③ 다이어그램 활용한 현업 참여를 통해 요구사항 조기 명확화

MDD를 통한 가시화 – 한글 기반 작성

한글로 작성된 모델은 의사소통 증진 및 이해도 향상으로 업무파악 시간 감소, 오류 감소 등의 효과가 있으며, 현업의 참여가 가능할 수 있습니다.

한글 기반 모델 작성

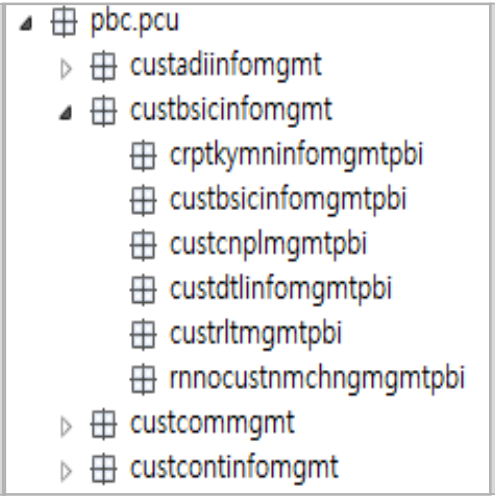
업무 구조



논리 표현식

```
[이자계산일수] = [일수정보].[일수];
[적용금리] = [여신계좌금리조회].[적용금리];
[정상이자] = [대출잔액]*[이자계산일수]
                *[적용금리]/365/100;
```

패키지 구조



소스 코드

```
intCalcDays = daysInfo.getBigDecimal("days");
aplyIntr = lnbzAccoIntrIqry
                .getBigDecimal("aplyIntr");
nrin = loanBal.multiply(intCalcDays)
          .multiply(aplyIntr)
          .divide(new BigDecimal("365"),
                50, RoundingMode.HALF_EVEN)
          .divide(new BigDecimal("100"),
                50, RoundingMode.HALF_EVEN);
```

기대효과 – 이해도 향상

① **업무 파악 시간 감소**

② **한글로 인한 가독성 향상으로 이해도 향상**

③ **Syntax Error 등의 오류 가능성 감소**