2020년 한국인터넷정보학회 추계학술발표대회

역공학 기반 UML 설계 복원을 위한 소프트웨어 가시화

이원영[1], 정세준[1], 권하은[2], 김영철[1]

홍익대학교 소프트웨어 공학 연구실[1], 한국정보통신기술협회[2]

CONTENTS

Hongik Univ. Selab

1) 연구 목적

객체지향 코드 품질 측정 방법이 불충분하여 절차식 패러다임을 객체지향 품질 패러다임으로 수정하여 사용

객체지향 패러다임에 맞는 설계 복잡도 논의 필요

설계도 비교 검증을 통한 요구사항 충족 확인

소프트웨어 개발 라이프 사이클 前 단계에서의 고품질화

설계 단계 리팩토링(Refactoring)을 통한 '설계→구현 →설계' 순환적/반복적인 소프트웨어 개발

모델 복잡도 해소하여 모델 이해도(Understandability), 수정용이성(Updateability) 향상

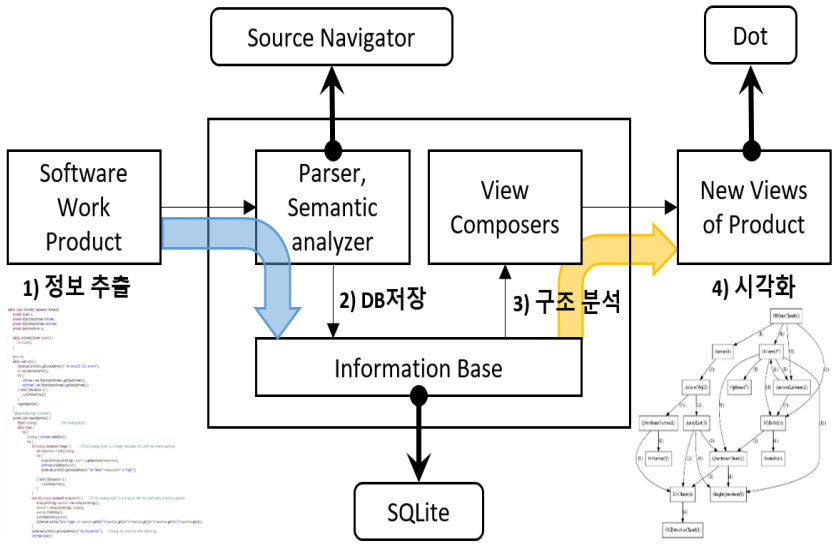
UML 다이어그램 가시화

소프트웨어 개발/유지보수 비용과 시간 단축

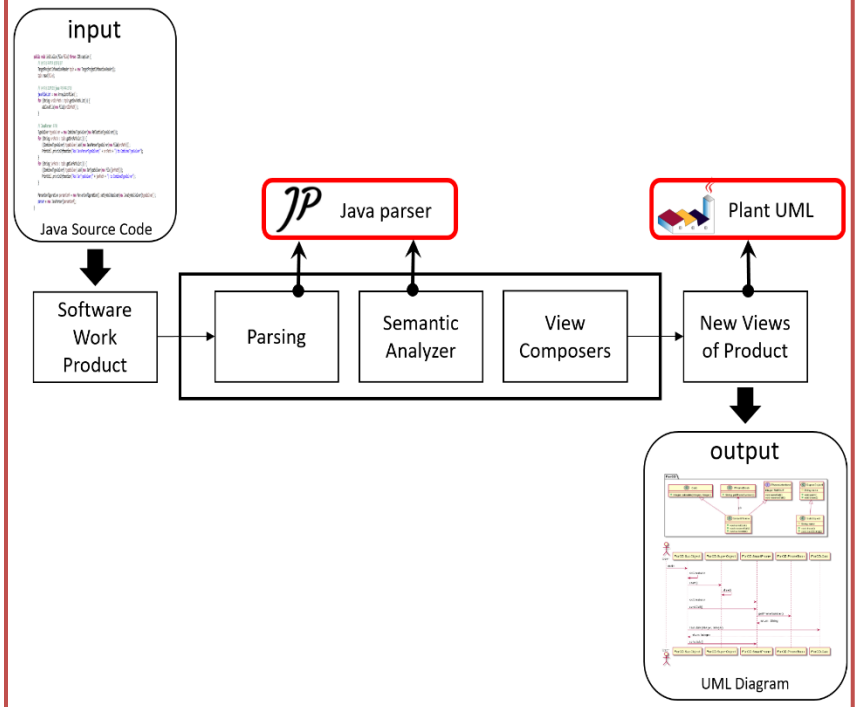
02

1) 기존 Tool-chain

기존 Tool-chain



새로운 Tool-chain



2) 개선 사항들

① 분석 도구의 개선

기존 Tool-chain에서 사용한 Source Navigator

- CMD 명령어 커맨드를 설정하여 Process 클래스로 OS 실행
- C와 Java 언어 모두 분석하지만 **Java 분석은 미비**
- 느리지만 단순한 구조

```
String snCmd = snPath + " -b --import " + fileListPath
        + " --databasedir " + sndbDirPath + " --project "
        + projectPath + "\\";
Process p = Runtime.getRuntime().exec(snCmd);
p.waitFor();
for (int i = 0; i < 15; i++) {
    Thread.sleep(3000);
    afterFileCnt = sndbDir.list().length;
    if (afterFileCnt - beforeFileCnt >= MIN_SNDB_FILE_CNT) {
        Thread.sleep(3500);
        break;
    }
```

생성되는 SNDB 파일들과 데이터베이스 테이블

새로운 Tool-chain에서 사용하는 Java parser

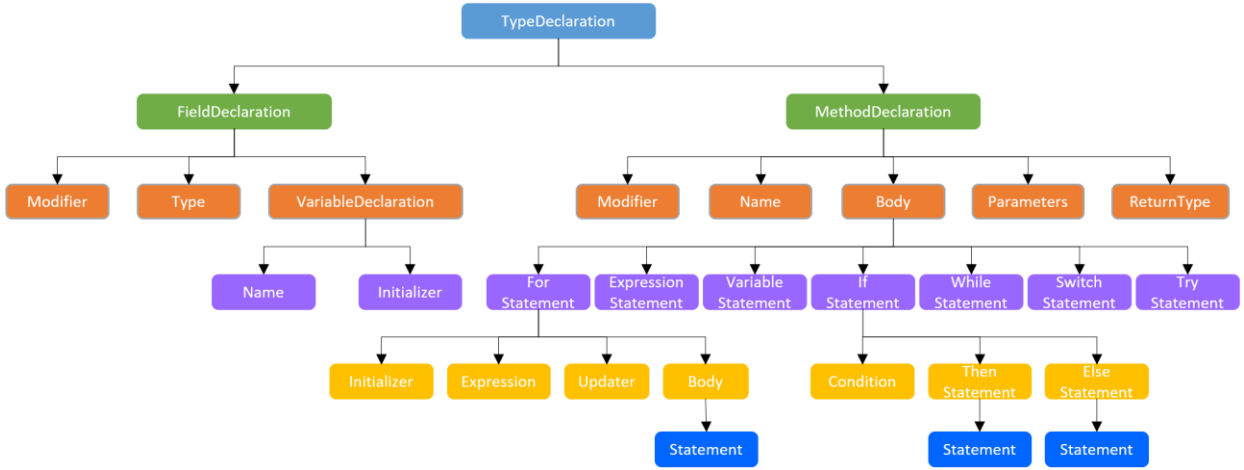
- 초기화한 parser를 자바에서 실행
- Java만 분석
- 복잡하지만 빠른 구조

```
TypeSolver typeSolver = new CombinedTypeSolver(new ReflectionTypeSolver());
for (String srcPath : tpir.getSrcPathList()) {
    ((CombinedTypeSolver) typeSolver).add(new JavaParserTypeSolver(new File(srcPath)));
    PrintUtil.printlnInfomation("Add JavaParserTypeSolver(" + srcPath + ") to CombinedTypeSolver");
}
for (String jarPath : tpir.getJarPathList()) {
    ((CombinedTypeSolver) typeSolver).add(new JarTypeSolver(new File(jarPath)));
    PrintUtil.printlnInfomation("Add JarTypeSolver(" + jarPath + ") to CombinedTypeSolver");
}

ParserConfiguration parserConf = new ParserConfiguration().setSymbolResolver(new JavaSymbolSolver(typeSolver));
parser = new JavaParser(parserConf);

compUnit = parser.parse(ParseStart.COMPILATION_UNIT, new StreamProvider(new FileReader(javaFile)))
        .getResult().get();
```

2) 개선 사항들

② 간결한 분석 단계

기존 Tool-chain의 분석 단계	새로운 Tool-chain의 분석 단계
• 윈도우 프로세스 사용 요구 • Target Source 경로, Source Navigator 경로, 분석 파일 저장경로, DB파일경로를 읽어 Target File 검색하여 실행 • Source Navigator.exe 프로세스 실행 • Source Navigator 결과물 **SNDB 저장** • Dbdump.exe 프로세스로 SNDB 파일을 읽고 DB 생성	• Java만 있으면 작동 가능 • Target Source 경로, jar파일 경로를 읽어 Java parser 객체를 생성 및 초기화하여 실행 • Java parser 객체로 분석하여 **AST 객체 생성** • 데이터베이스 생성단계 생략



Java AST 구조

2) 개선 사항들

③ 분석 데이터 저장 단계 개선

기존 Tool-chain의 저장 단계	새로운 Tool-chain의 저장 단계
• 윈도우 환경과 **쿼리문 사용** 요구 • Database에 연결하여 DB 생성 • **SNDB 결과물의 각 파일별로 테이블 생성** • Dbdump.exe 프로세스로 SNDB를 읽어 생성한 데이터베이스 테이블에 데이터 삽입	• Java만 있으면 작동 가능 • Java parser 객체가 생성한 ComplilationUnit 객체를 Visitor로 순환하며 ModelManager에 데이터 저장 • **Java parser 라이브러리에서 제공하는 Visitor로 AST 객체를 순환**하며 ModelManager에 데이터 저장 • 데이터베이스 저장 단계 생략

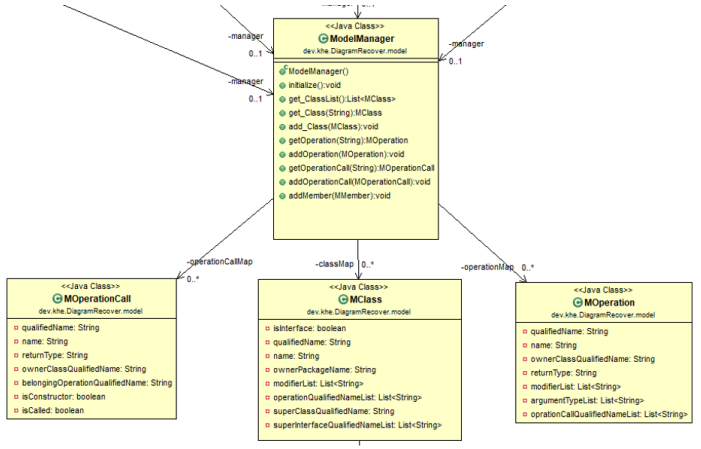
```
Statement stat = conn.createStatement();

stat.executeUpdate("drop table if exists SNDB_CL;");
stat.executeUpdate("create table SNDB_CL (NAME TEXT,"
        + "START_LINE_NO TEXT," + "PATH TEXT,"
        + "END_LINE_NO TEXT," + "ATTRIBUTES TEXT" + ");");
stat.executeUpdate("drop table if exists SNDB_IV;");
stat.executeUpdate("create table SNDB_IV (CLASS_NAME TEXT,"
        + "VARIABLE_NAME TEXT," + "START_LINE_NO TEXT,"
        + "PATH TEXT," + "END_LINE_NO TEXT," + "ATTRIBUTES TEXT"
        + ");");
stat.executeUpdate("drop table if exists SNDB_LV;");
stat.executeUpdate("create table SNDB_LV (VARIABLE_NAME TEXT,"
        + "START_LINE_NO TEXT," + "PATH TEXT,"
        + "END_LINE_NO TEXT," + "ATTRIBUTES TEXT" + ");");
stat.executeUpdate("drop table if exists SNDB_MD;");
stat.executeUpdate("create table SNDB_MD (CLASS_NAME TEXT,"
        + "METHOD_NAME TEXT," + "START_LINE_NO TEXT,"
        + "PATH TEXT," + "END_LINE_NO TEXT," + "ATTRIBUTES TEXT,"
        + "RETURN_TYPE TEXT," + "ARGUMENT_TYPES TEXT,"
        + "ARGUMENT_NAMES TEXT" + ");");
stat.executeUpdate("drop table if exists SNDB_BY;");
stat.executeUpdate("create table SNDB_BY (REFERRED_CLASS_NAME TEXT,"
        + "REFERRED_SYMBOL_NAME TEXT,"
        + "REFERRED_TYPE TEXT,"
        + "REFER_CLASS_NAME TEXT,"
        + "REFER_SYMBOL_NAME TEXT,"
        + "REFER_TYPE TEXT,"
        + "ACCESS TEXT,"
        + "LINE_NO TEXT,"
        + "PATH TEXT,"
```

```
or (File file : workspaceDir.listFiles()) {
    if (file.getName().contains(projectName)) {
        ArrayList<String> list = new ArrayList<String>();

        try {
            String dbdumpCmd = "\"" + dbdumpPath + "\"  \"" + file.getAbsolutePath() + "\"";
            Process p = Runtime.getRuntime().exec(dbdumpCmd);
            BufferedReader br = new BufferedReader(new InputStreamReader(p.getInputStream()));
            String line;
            while ((line = br.readLine()) != null) {
                list.add(line);
            }
            br.close();
        } catch (IOException e) {
            System.out.println("fail!");
            e.printStackTrace();
        }

        if (file.getName().contains(".cl")) {
            insertCl(conn, list);
        } else if (file.getName().contains(".iv")) {
            insertIv(conn, list);
        } else if (file.getName().contains(".lv")) {
            insertLv(conn, list);
        } else if (file.getName().contains(".md")) {
            insertMd(conn, list);
        } else if (file.getName().contains(".by")) {
            insertBy(conn, list);
        } else if (file.getName().contains(".in")) {
            insertIn(conn, list);
        }
```

SNDB 결과물 파일별 쿼리문

```
@Override
public void visit(MethodDeclaration n, StringBuilder arg) {
    // TODO Auto-generated method stub
    if (n.findAncestor(MethodDeclaration.class).isPresent()) {
        return;
    }

    MOperation operation = new MOperation();
    operation.setQualifiedName(n.resolve().getQualifiedName());
    operation.setName(n.getName().toString());
    operation.setOwnerClassQualifiedName(
            operation.getQualifiedName().substring(0, operation.getQualifiedName().lastIndexOf(".")));
    // operation.setReturnType(n.getType().toString());
    operation.setReturnType(TypeUtil.getTypeStr(n.resolve().getReturnType()));

    NodeList<Modifier> modifierList = n.getModifiers();
    Modifier m = null;
    for (int i = 0; i < modifierList.size(); i++) {
        m = modifierList.get(i);
```

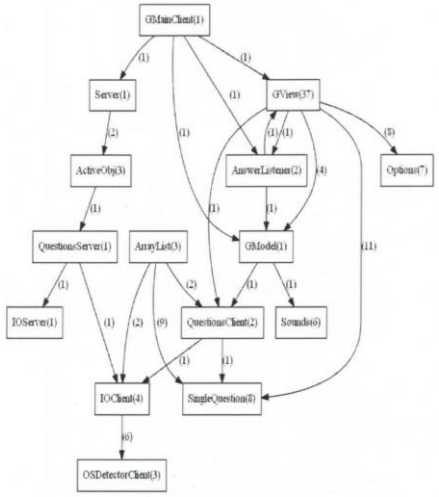
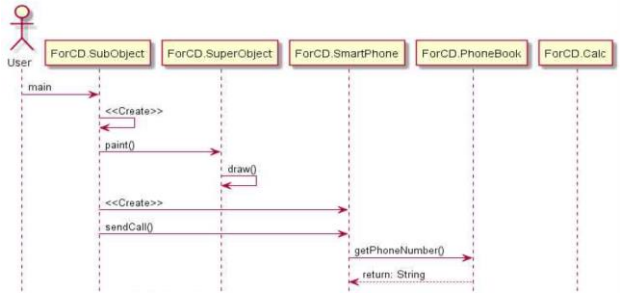
Java AST 순회



ModelManager 저장구조

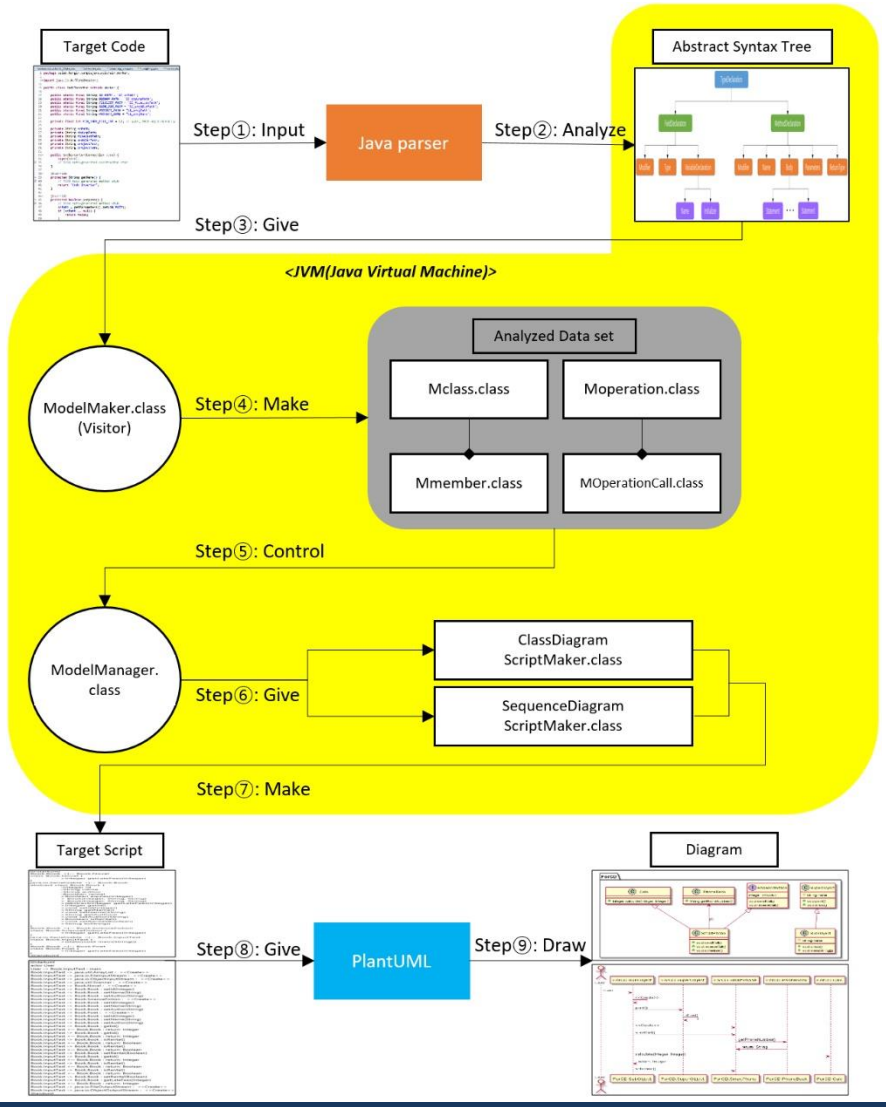
2) 개선 사항들

④ 출력 도구 단일화

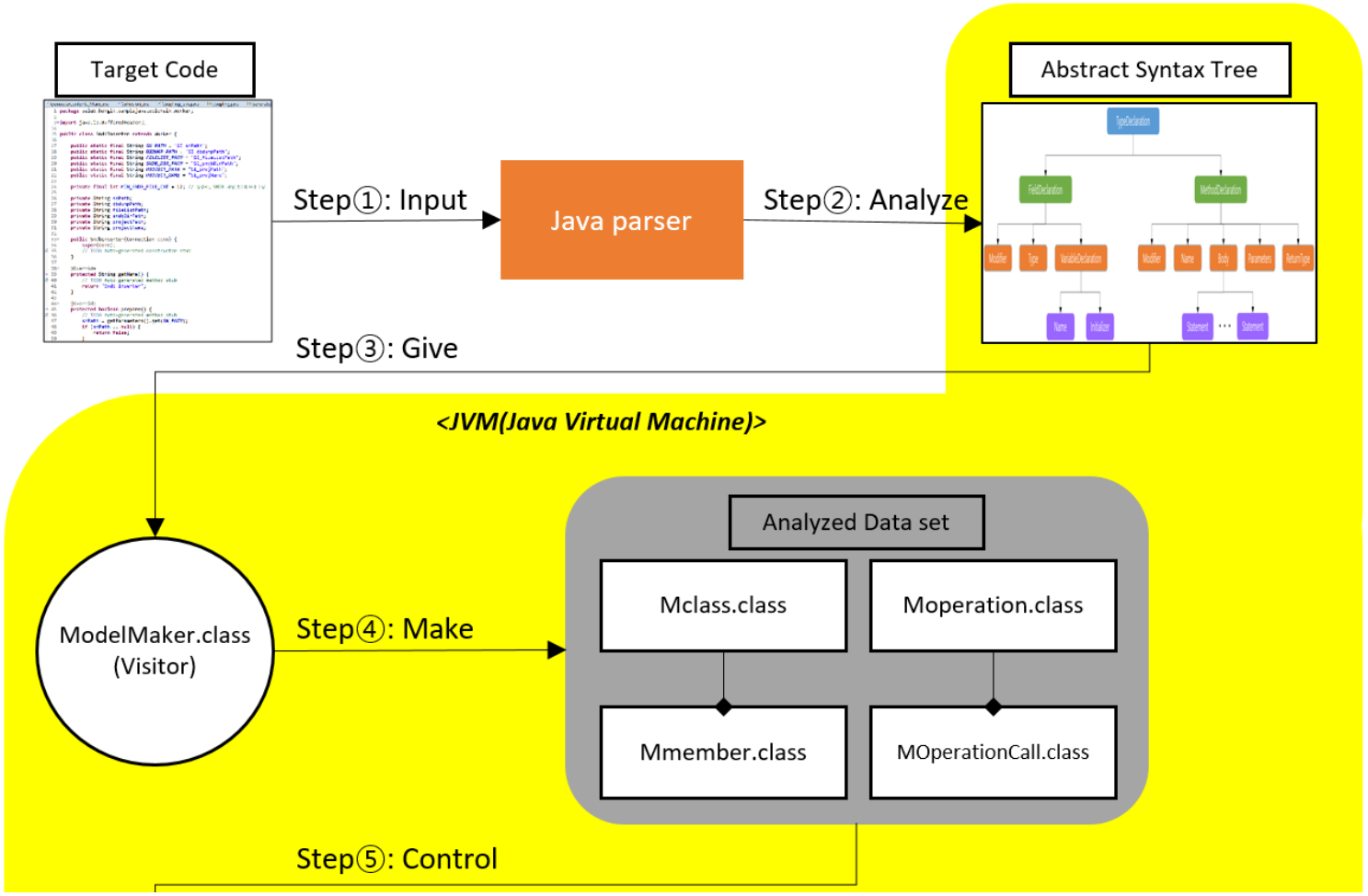
기존 Tool-chain의 가시화 단계	새로운 Tool-chain의 가시화 단계
• 윈도우 환경과 쿼리문 사용 요구 • **Dot.exe** 프로세스를 통해 그래프 출력하지만 순차 다이어그램은 그리기 어려움 • 데이터베이스 필요 • 생성하고자 하는 그래프에 맞게 쿼리문을 통해 데이터를 가져옴	• Java만 있으면 작동 가능 • **PlantUML** 오픈소스를 사용하여 다양한 UML 다이어그램 출력 • 데이터베이스 불필요 • 생성하고자 하는 그래프 종류에 맞게 원하는 데이터를 객체로부터 가져옴
 함수 호출 그래프	@startuml actor User User -> ForCD.SubObject : main ForCD.SubObject -> ForCD.SubObject : <<Create>> ForCD.SubObject -> ForCD.SuperObject : paint() ForCD.SuperObject -> ForCD.SuperObject : draw() ForCD.SubObject -> ForCD.SmartPhone : <<Create>> ForCD.SubObject -> ForCD.SmartPhone : sendCall() ForCD.SmartPhone -> ForCD.PhoneBook : getPhoneNumber() ForCD.SmartPhone <-- ForCD.PhoneBook : return: String 순차 다이어그램 스크립트  순차 다이어그램

03

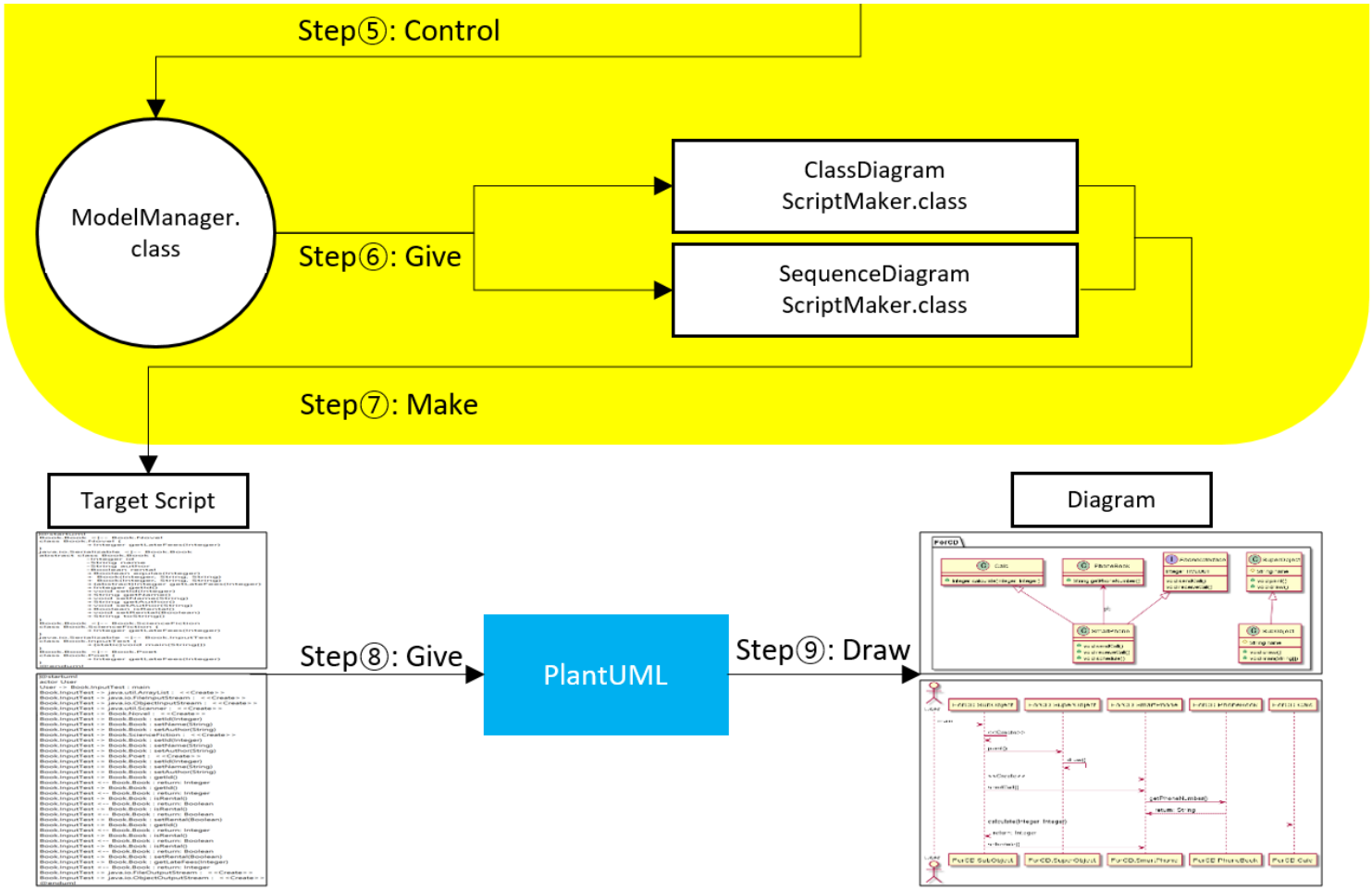
1) 역공학 기반 UML 복원 Tool-chain 전체 구성도



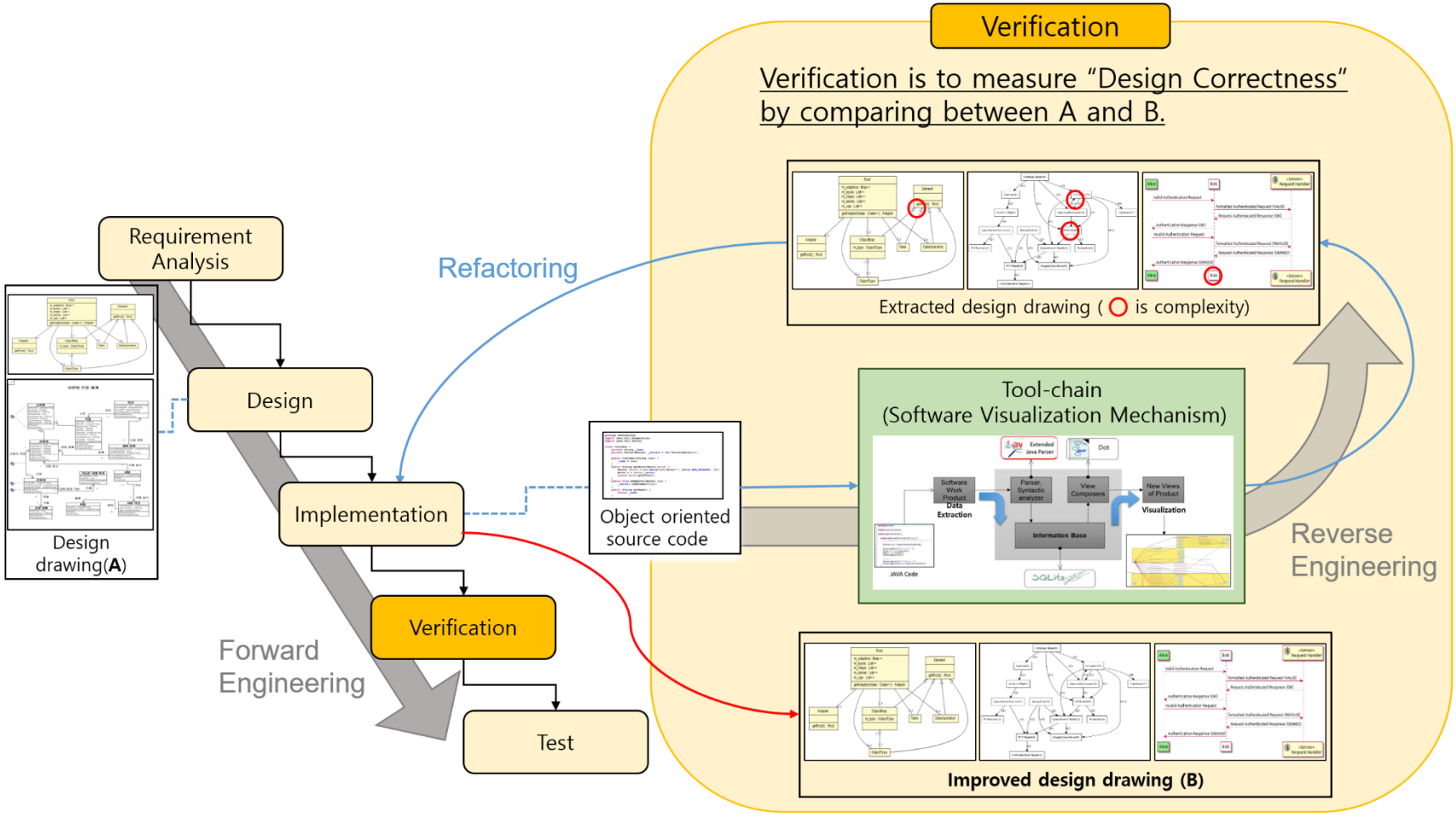
2) 설계 복원 방안


Target Code
Step①: Input
Java parser
Step②: Analyze
Abstract Syntax Tree
Step③: Give
<JVM(Java Virtual Machine)>
ModelMaker.class
(Visitor)
Step④: Make
Analyzed Data set
Mclass.class
Moperation.class
Mmember.class
MOperationCall.class
Step⑤: Control

2) 설계 복원 방안


Step⑤: Control
ModelManager.
class
Step⑥: Give
ClassDiagram
ScriptMaker.class
SequenceDiagram
ScriptMaker.class
Step⑦: Make
Target Script
Step⑧: Give
PlantUML
Step⑨: Draw
Diagram

3) 설계 검증 방안

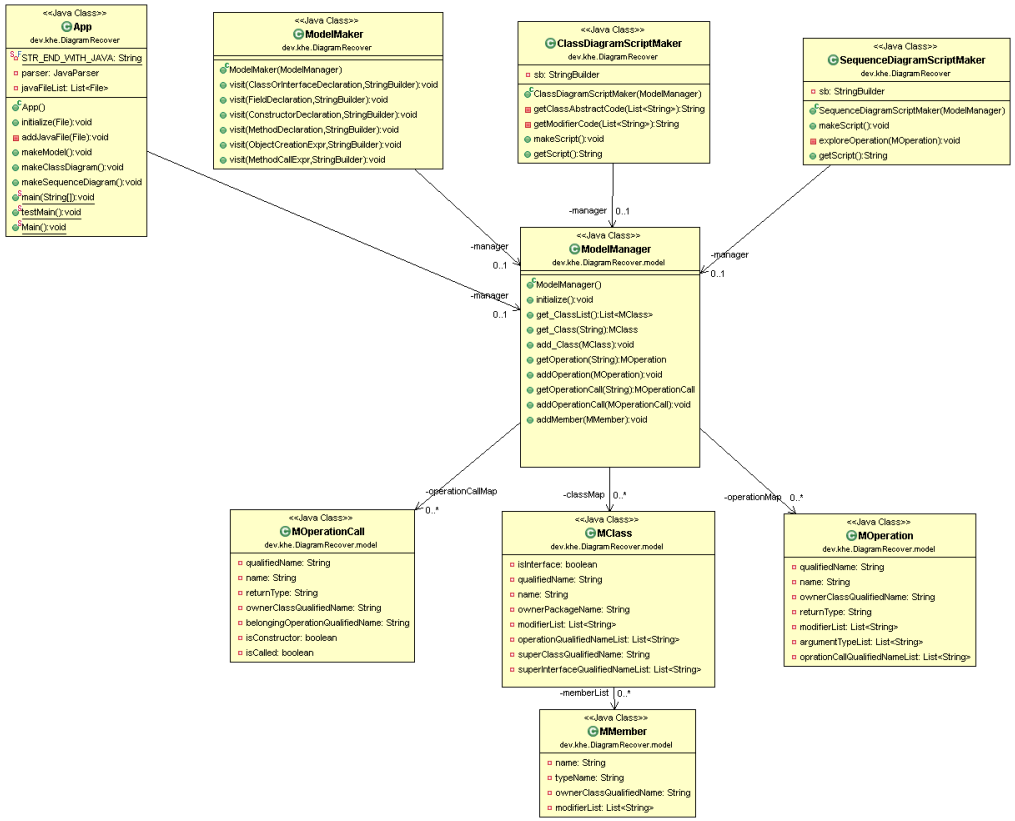

Verification
Verification is to measure "Design Correctness" by comparing between A and B.
Extracted design drawing (○ is complexity)
Requirement Analysis
Design
Implementation
Verification
Test
Refactoring
Forward Engineering
Design drawing(A)
Object oriented source code
Tool-chain
(Software Visualization Mechanism)
Reverse Engineering
Improved design drawing (B)

1) Tool-chain 적용: 도서관리 시스템

- Step ①: Java source code 입력

```
public class InputTest implements Serializable {

    public static void main(String[] args) {
        ObjectInputStream in = null;
        ObjectOutputStream out = null;
        try {
            ArrayList<Book> bookList = null;
            try {
                in = new ObjectInputStream(new FileInputStream("data.bin"));
                bookList = (ArrayList<Book>) in.readObject();
            } catch (IOException e) {
                bookList = new ArrayList<Book>();
            }
            int num;

            do {
                Scanner scan = new Scanner(System.in);

                System.out.println("-----------------");
                System.out.println("Select the work");
                System.out.println("-----------------");
                System.out.println("1. Add a book");
                System.out.println("2. Delete a book");
                System.out.println("3. Search a book");
                System.out.println("4. Borrow a book");
                System.out.println("5. Return a bokk");
                System.out.println("6. Exit");
                System.out.println("-----------------");

                num = scan.nextInt();
                scan.nextLine();
                switch (num) {

                case 1:
                    System.out.println("Input your id");
                    int id = scan.nextInt();
                    scan.nextLine();

                    System.out.println("Input book title");
                    String name = scan.nextLine();

                    System.out.println("Input book author");
                    String author = scan.nextLine();

                    System.out.println("1.novel 2.science 3.poet");
                    int category = scan.nextInt();
                    scan.nextLine();
                    Book b;

                    switch (category) {
                    case 1:
                        b = new Novel();
                        b.setId(id);
                        b.setName(name);
                        b.setAuthor(author);
                        bookList.add(b);
                        break;

                    case 2:
                        b = new ScienceFiction();
                        b.setId(id);
                        b.setName(name);
                        b.setAuthor(author);
                        bookList.add(b);
                        break;

                    case 3:
                        b = new Poet();
                        b.setId(id);
                        b.setName(name);
                        b.setAuthor(author);
```

- Step ②: Java parser를 통해 데이터 분석
- Step ③: 분석데이터를 ModelManager에 데이터 저장



ModelManager 구조

1) Tool-chain 적용: 도서관리 시스템

- Step ④: PlantUML을 통해 클래스 다이어그램과 순차 다이어그램 스크립트 작성

```
@startuml
Book.Book <|-- Book.Novel
class Book.Novel {
        +Integer getLateFees(Integer)
}
java.io.Serializable <|-- Book.Book
abstract class Book.Book {
        -Integer id
        -String name
        -String author
        -Boolean rental
        +Boolean equlas(Integer)
        + Book(Integer, String, String)
        + Book(Integer, String, String)
        +{abstract}Integer getLateFees(Integer)
        +Integer getId()
        +void setId(Integer)
        +String getName()
        +void setName(String)
        +String getAuthor()
        +void setAuthor(String)
        +Boolean isRental()
        +void setRental(Boolean)
        +String toString()
}
Book.Book <|-- Book.ScienceFiction
class Book.ScienceFiction {
        +Integer getLateFees(Integer)
}
java.io.Serializable <|-- Book.InputTest
class Book.InputTest {
        +{static}void main(String[])
}
Book.Book <|-- Book.Poet
class Book.Poet {
        +Integer getLateFees(Integer)
}
@enduml
```

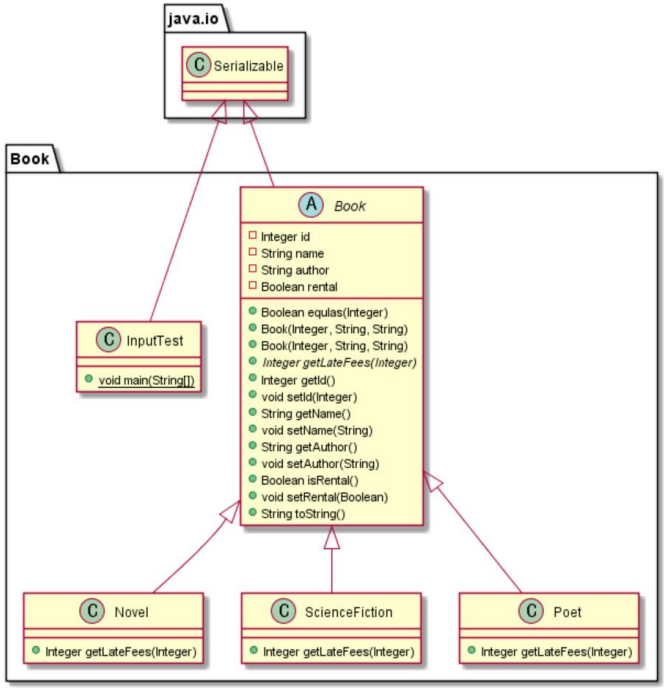
클래스 다이어그램 스크립트

```
@startuml
actor User
User -> Book.InputTest : main
'Book.InputTest -> java.util.ArrayList :  <<Create>>'
'Book.InputTest -> java.io.FileInputStream :  <<Create>>'
'Book.InputTest -> java.io.ObjectInputStream :  <<Create>>'
'Book.InputTest -> java.util.Scanner :  <<Create>>'
loop [num = 1~5]
    alt [num = 1]
        alt [num = 1]
            Book.InputTest -> Book.Novel :  <<Create>>
            Book.InputTest -> Book.Book : setId(Integer)
            Book.InputTest -> Book.Book : setName(String)
            Book.InputTest -> Book.Book : setAuthor(String)
        else [num = 2]
            Book.InputTest -> Book.ScienceFiction :  <<Create>>
            Book.InputTest -> Book.Book : setId(Integer)
            Book.InputTest -> Book.Book : setName(String)
            Book.InputTest -> Book.Book : setAuthor(String)
        else [num = 3]
            Book.InputTest -> Book.Poet :  <<Create>>
            Book.InputTest -> Book.Book : setId(Integer)
            Book.InputTest -> Book.Book : setName(String)
            Book.InputTest -> Book.Book : setAuthor(String)
        end
    else [num = 2]
        Book.InputTest -> Book.Book : getId()
        Book.InputTest <-- Book.Book : return: Integer
    else [num = 4]
        loop
            alt [id] '???'
                Book.InputTest -> Book.Book : getId()
                Book.InputTest <-- Book.Book : return: Integer
                alt [rental]
                    Book.InputTest -> Book.Book : isRental()
                    Book.InputTest <-- Book.Book : return: Boolean
                else [!rental]
                    Book.InputTest -> Book.Book : isRental()
                    Book.InputTest <-- Book.Book : return: Boolean
                    Book.InputTest -> Book.Book : setRental(Boolean)
                end
            end
        end
```

순차 다이어그램 스크립트

1) Tool-chain 적용: 도서관리 시스템

- Step ⑤: PlantUML을 각 스크립트를 자동으로 UML 설계도면 가시화



복원된 클래스 다이어그램

복원된 순차 다이어그램

1) 연구 결과

실제 운영되는 코드를 통한 최신 설계도 생산이 가능하여 유지보수 용이

설계 구조의 개선, 설계 복잡도, 성능 분석 가능

소프트웨어 설계 복잡도 가시화를 통해 모듈 간 복잡도 측정

모듈 결합력을 낮추고 응집도를 높여 복잡도에 대한 소프트웨어 품질 향상

소프트웨어 설계 검증을 통해 기존에 발견하지 못한 문제점 해결함으로써 잠재적인 오류 제거

개발과정에서 코드 정적 분석과 설계 검증을 통해 빠른 유지보수 가능

오픈 소스 기반 코드 정적 분석기를 통해 기업의 소프트웨어 자산 구축 용이, 인력 문제 해소, IT 벤처/중소기업의 유지보수 지원도구로 활용, 비용 및 시간 절감

THANK YOU