An Automatic Visualization Mechanism Of Extracting Diverse Module Unit Level For Software Complexity

January 19th, 2016

Junsun Hwang
hwang@selab.hongik.ac.kr
SE Lab, Hongik University
Advisor: R. Youngchul Kim

Table Of Contents

0. Motivation

❖Why automatically extracting diverse module based on reverse engineering?

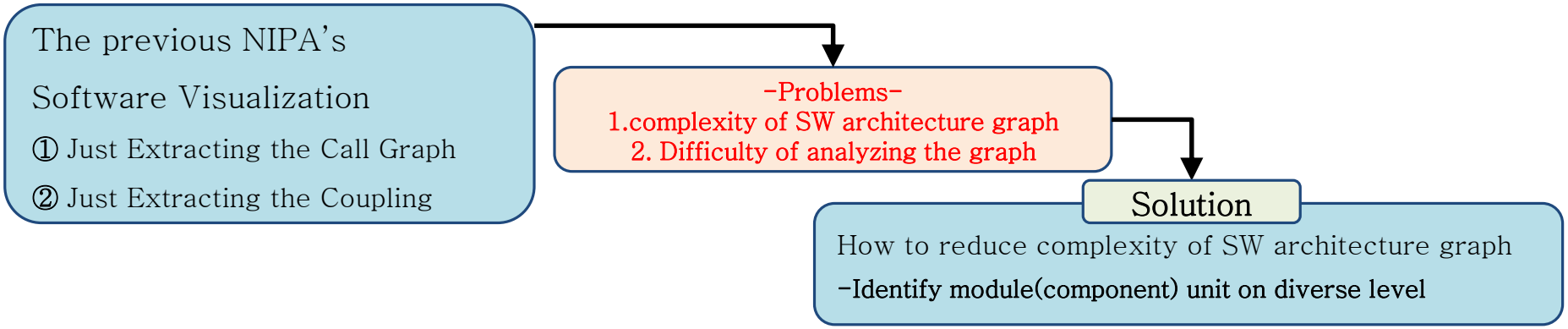
1. Need to identify software architecture in a system.
2. Need to identify internal structure of software.
3. Need to enhance reusable software of the existing system.
4. Effectively refactoring to identify module complexity on each level of a system.

I. Introduction

❖ Background of research

The importance of software is emphasized, but our domestic smaller firms have some difficulties in proper control of software quality under their developing environment due to invisibility and increasing complexity of software.

→ As referred to this paper, NIPA's method of software visualization is a method of visualizing source code of software in order to solve this problems.



❖ Purpose of research

1. Reduce complexity of graph when coupling is visualized.
2. Visualize software module in diverse component level.

II. Related Works

❖ SW Visualization

- A technique for visualizing the SW architecture to overcome the invisibility of the SW.
- To visualize each of the module as node and to visualize the relationship between modules as an edge.

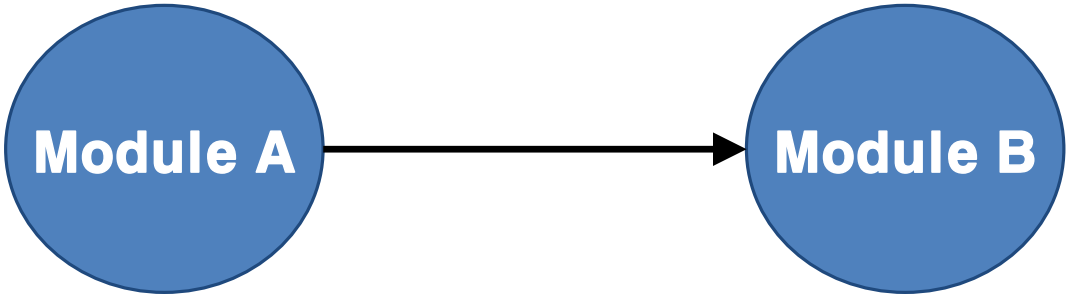


Figure 1. In case Module A refers Module B

II. Related Works

❖Extracting coupling between modules

- We have to minimize coupling for the high quality of the software.
- The lower coupling is, the more it is independency of module : and it is easy to reuse and maintain the module.
- The type of Coupling is all six such as Data, Stamp, Control, External, Common, Content.
- T he closer it approaches data coupling, the more it is becoming independent for higher software quality.

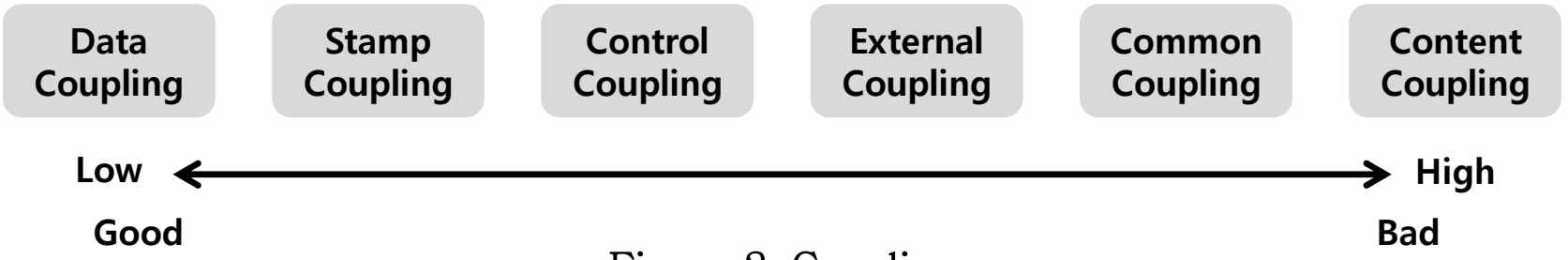


Figure 2. Coupling

II. Our Previous Approach

❖ The previous research: Extracting coupling between each modules.

- Constructed a tool-chain for SW Visualization.
- Just extract coupling between each module(class) of a system.

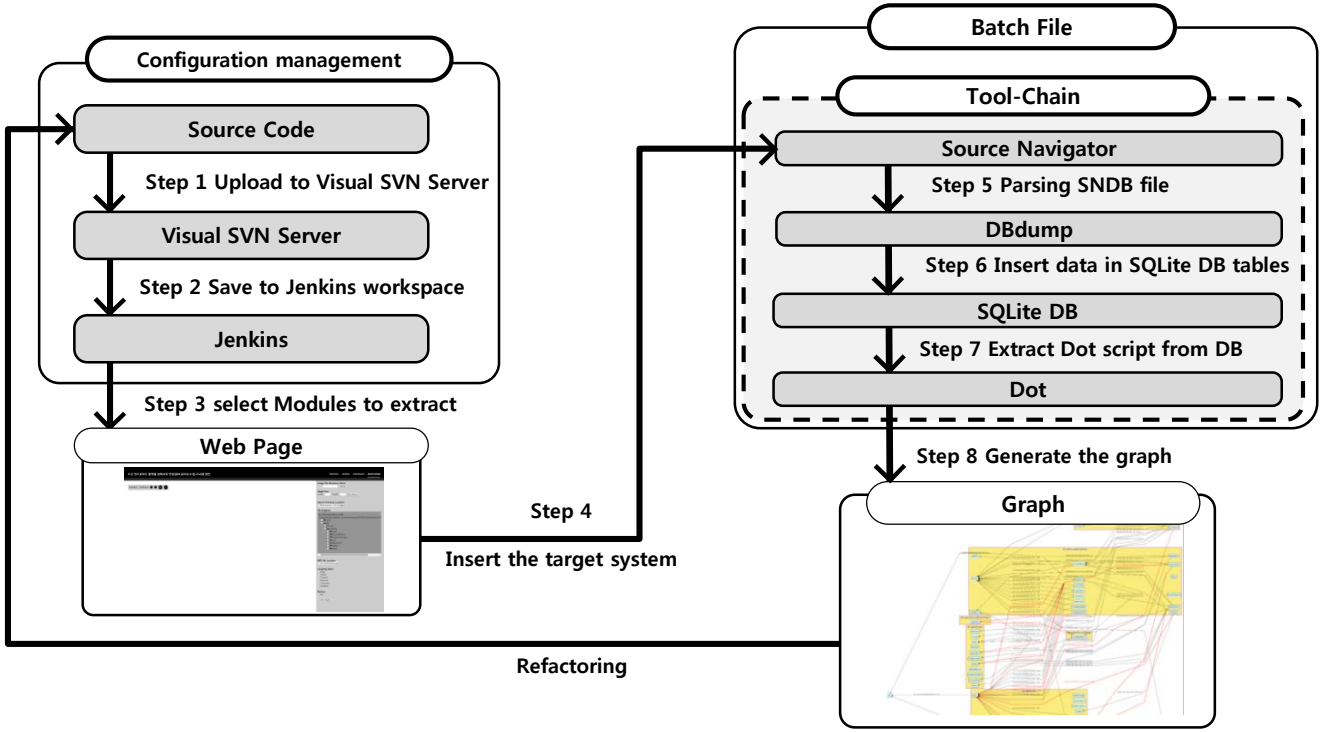


Figure 3. Tool-Chain Process to Extract Coupling

III. Our Extracting Mechanism Of Coupling Between Modules Per Diverse Component Level

- In this paper, we follow the same Tool-Chain Configuration as Figure 3 in order to extract the coupling between modules in various component level. But Tool-Chain is configured by changing the way of Step 3, Step 6, Step 7 in Figure 3.

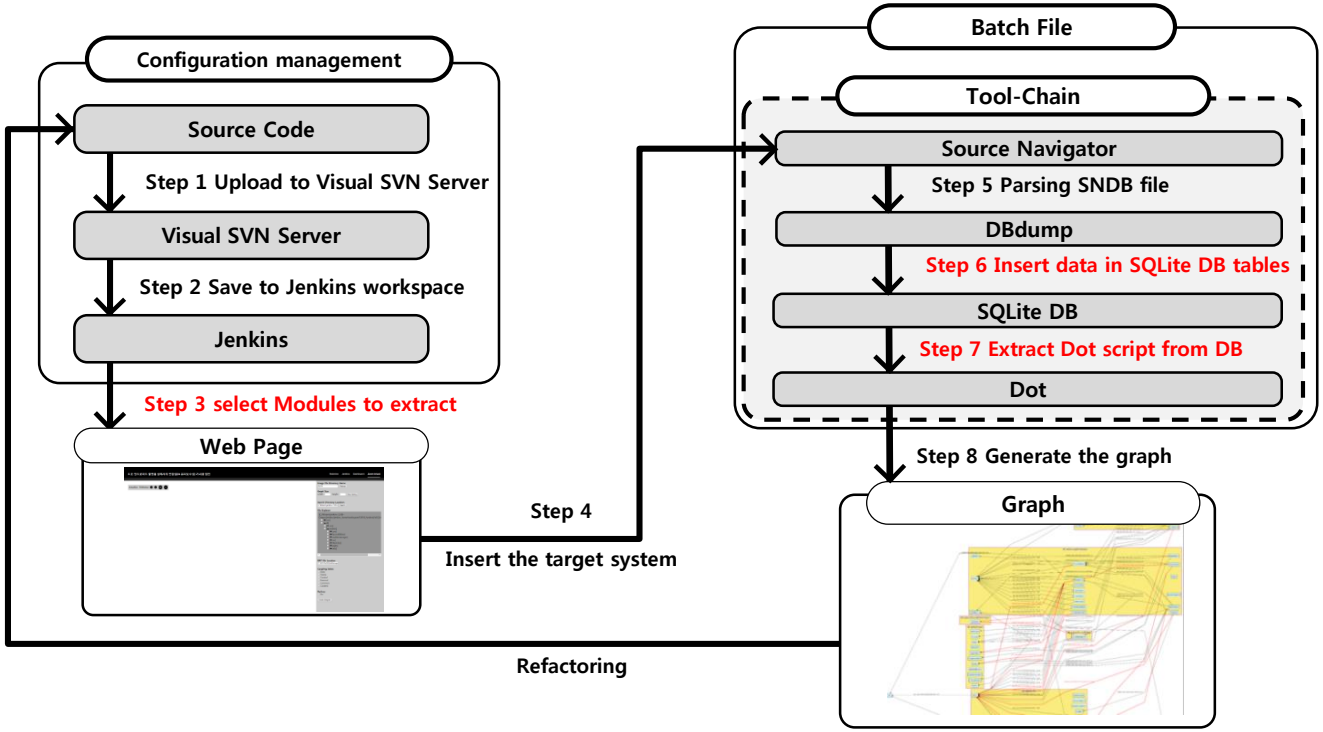


Figure 3. Tool-Chain Process to Extract Coupling

III. Our Extracting Mechanism Of Coupling Between
mponent Level

- of coupling in a web page by the

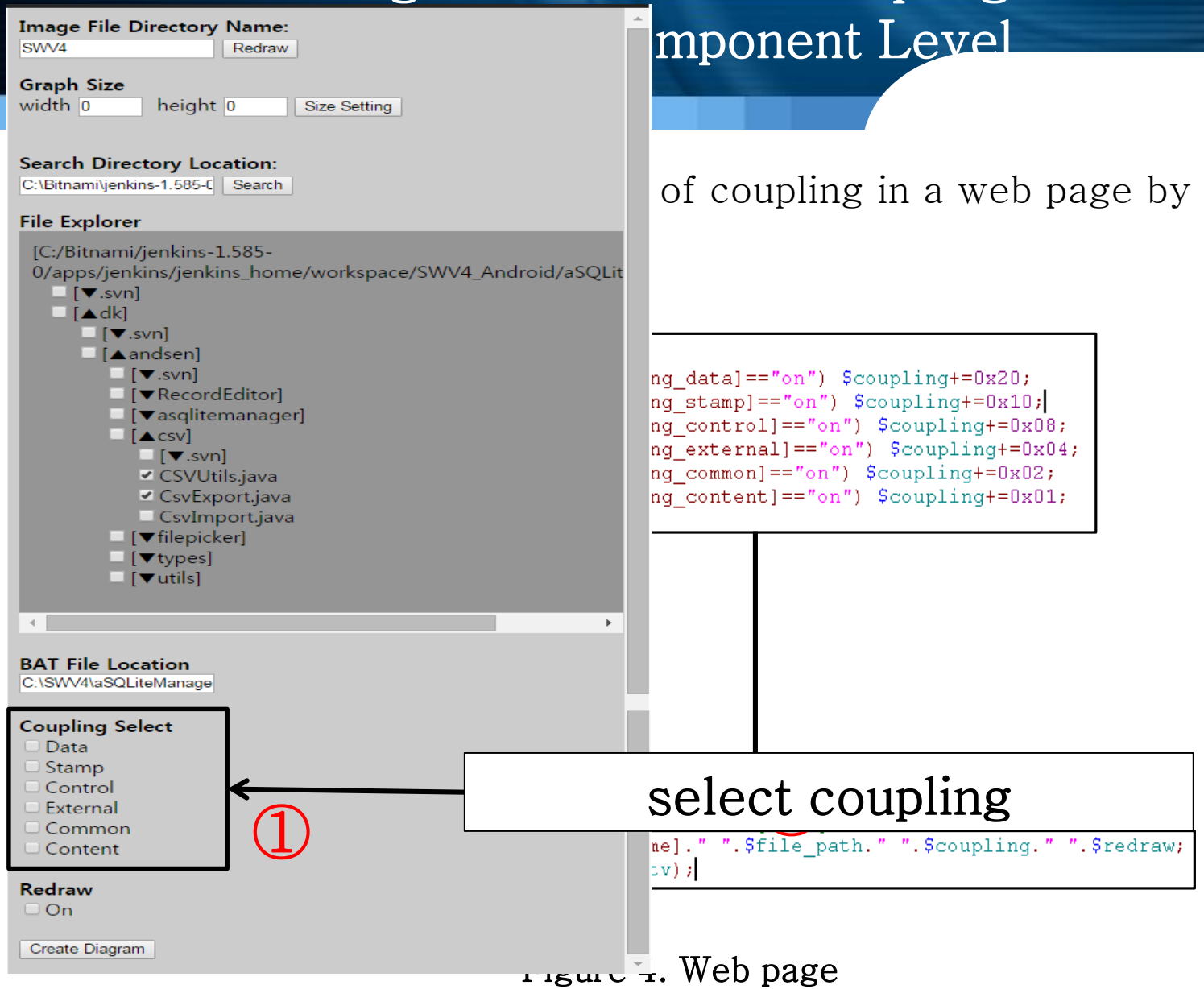


Figure 4. Web page

III. Our Extracting Mechanism Of Coupling Between Modules Per Diverse Component Level

❖In Step 6, change each part: query to define the module in the diverse component level.

1. Insert statement if the module is in a class unit.

```
module_mapping_creation = "insert into MODULE_MAPPING "
 +"select distinct '"+name+ "' as MODULE_NAME, TYPE.NAME as CLASS_NAME, 'null' as PACKAGE_NAME, 'file' as MODULE_TYPE, '"
 +architecture_name +"' as ARCHITECTURE_NAME, '" + project_name + "' as PROJECT_NAME from TYPE "
 + "where TYPE.PROJECT_NAME = '" + project_name +"' "
 + "and TYPE.ANNOTATE='class' "
 + "and not TYPE.ANNOTATE='method' "
 + "and TYPE.NAME = '"+name+"' "
 + "and TYPE.PACKAGE_PATH='"+module_path+"'";
```

2. Insert statement if the module is in a package unit.

```
module_mapping_creation = "insert into MODULE_MAPPING "
 +"select distinct '"+name+ "' as MODULE_NAME, '"+file_name+"' as CLASS_NAME, '"+package_name+"' as PACKAGE_NAME, 'package' as MODULE_TYPE, '"
 +architecture_name +"' as ARCHITECTURE_NAME, '" + project_name + "' as PROJECT_NAME from TYPE "
 + "where TYPE.PROJECT_NAME = '" + project_name +"' "
 + "and TYPE.ANNOTATE='class' "
 + "and not TYPE.ANNOTATE='method' "
 + "and TYPE.NAME = '"+file_name+"' "
 + "and TYPE.PACKAGE_PATH='"+module_path+"'";
```

III. Our Extracting Mechanism Of Coupling Between Modules Per Diverse Component Level

❖In Step 7, change each part: A query is changed to the four cases in order to extract the scores of coupling between modules, which had different component levels.

1. Select statement in case a module reference and the modules that are referenced are all ‘class’.

```
query = "select Caller_class, Callee_class, sum(data),sum(stamp),sum(control),sum(external),sum(common),sum(content) "
 + "from Coupling as CP "
 + "where CP.Caller_package='null' "
 + "and CP.Callee_package='null' "
 + "group by Caller_class, Callee_class";
```

2. Select statement in case of a module reference in a package unit.

```
query2 = "select Caller_package, Callee_class, sum(data),sum(stamp),sum(control),sum(external),sum(common),sum(content) "
 + "from Coupling as CP "
 + "where not CP.Caller_package='null' "
 + "and CP.Callee_package='null' "
 + "group by Caller_package, Callee_class";
```

III. Our Extracting Mechanism Of Coupling Between Modules Per Diverse Component Level

❖In Step 7, change each part: A query is changed to the four cases in order to extract the scores of coupling between modules, which had different component levels.

3. Select statement in case of the module that is referenced in a package unit.

```
query3 = "select Caller_class, Callee_package, sum(data),sum(stamp),sum(control),sum(external),sum(common),sum(content) "
 + "from Coupling as CP "
 + "where CP.Caller_package='null' "
 + "and not CP.Callee_package='null' "
 + "group by Caller_class, Callee_package";
```

4. Select statement in case a module reference and the module that is referenced are all ‘package’.

```
query4 = "select Caller_package, Callee_package, sum(data),sum(stamp),sum(control),sum(external),sum(common),sum(content) "
 + "from Coupling as CP "
 + "where not CP.Caller_package='null' "
 + "and not CP.Callee_package='null' "
 + "and not CP.Caller_package=CP.Callee_package "
 + "group by Caller_package, Callee_package";
```

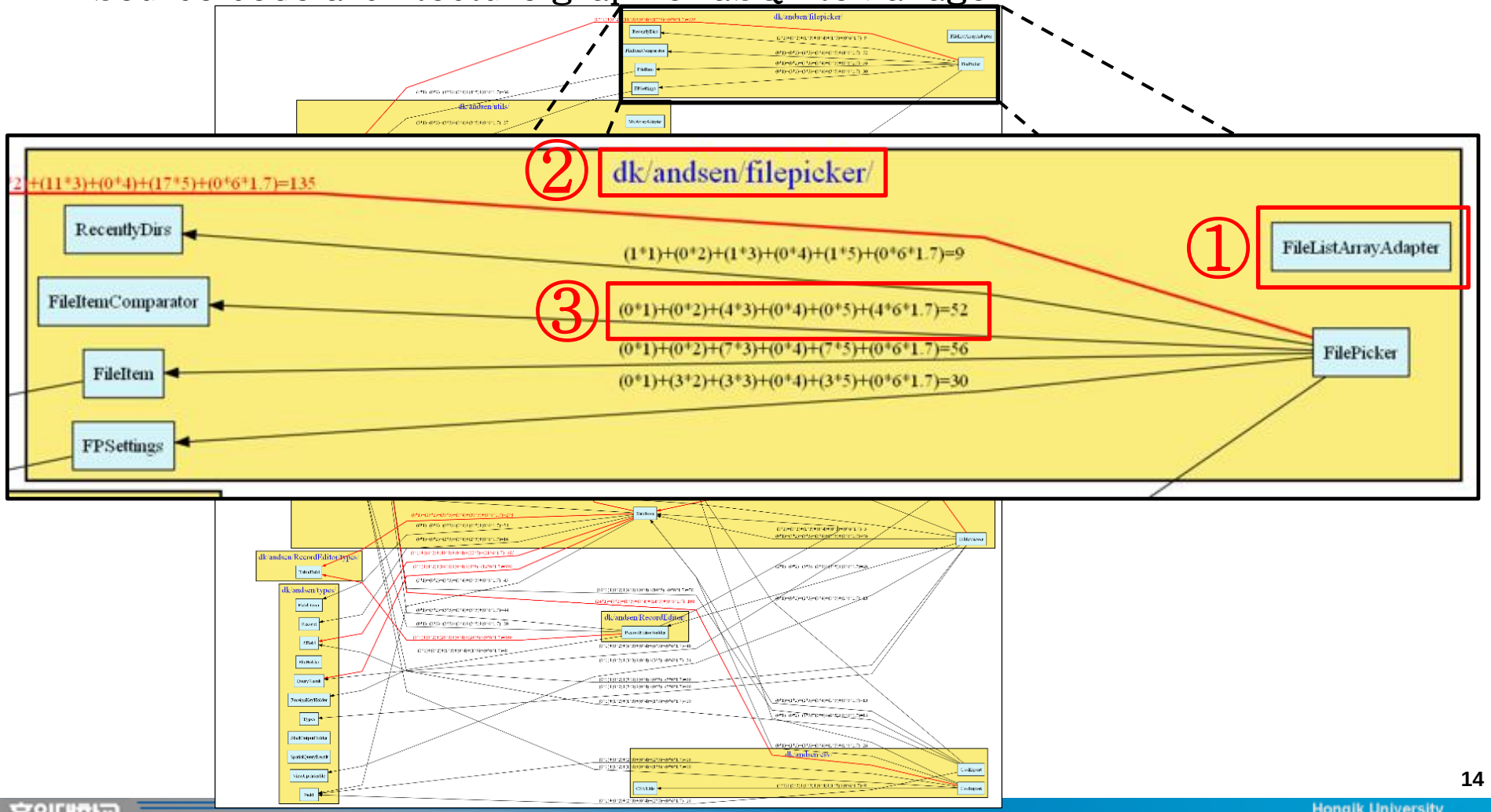
IV. Case Study

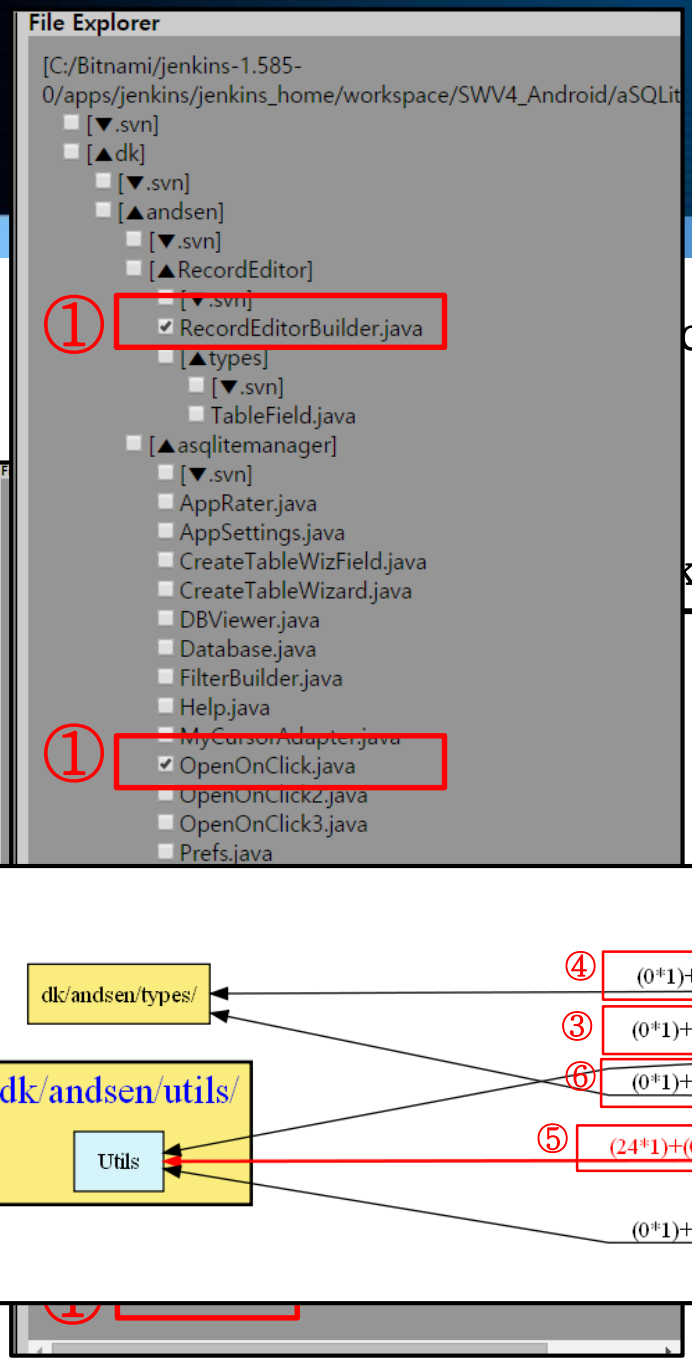
- Use aSQLiteManger which is open source Android application. In this research, the target system is aSQLiteManager.

aSQLiteManager.

IV. Case Study

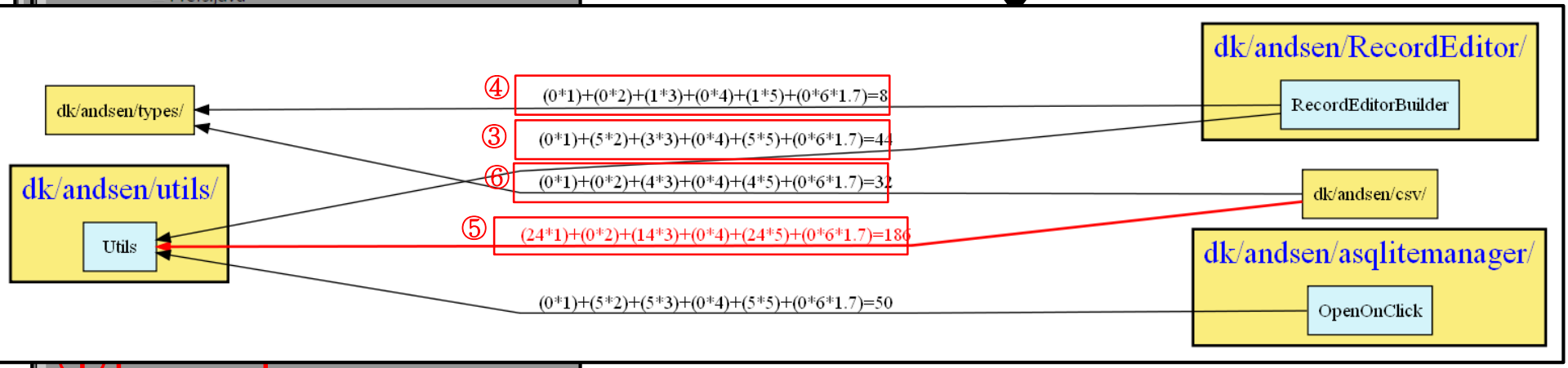
❖ Source code architecture graph of aSQLiteManager





ce code architecture defined by the module of

k on selective menu



V. Conclusion

❖Propose an automatic extraction mechanism for identifying diverse module based on reverse engineering

1. Identify software architecture in a system
2. Identify internal structure of software
3. Enhance reusable software of the existing system
4. Effectively refactoring to identify module complexity on each level of a system

Thank you.

hwang@selab.hongik.ac.kr