Extracting Architectural Model from Source Code

By:
Morteza Zakeri - Mohsen Amirian
Advanced Software Engineering Course
Iran University of Science and Technology
Winter 2017

Outline

- What is Software Reverse Engineering (SRE)?
- Two Different Dimensions
- Motivation
- SRE Tools
- Modularity Principles
- Extracting Architectural Model Step by Step
- Practical Case Study
- References

What is Software Reverse Engineering?

- Software Reverse Engineering (SRE) is the practice of analyzing a software system, either in whole or in part, to extract **design** and **implementation** information [1].

- [1] Cipresso, T. (2009). Software Reverse Engineering Education, (August), 120.

Two Different Dimensions

- Binary Code Reverse Engineering
  - To obtain source code from executable object.
- Source Code Reverse Engineering
  - To extract architectural features.

Motivation

1. **Old software systems** are often not documented or very less documentation is available.
   - Even in the systems where documentation is available there is no explicit mention of the architecture that the code possesses.

Motivation

2. **New changes** to the system need a knowledge of implicit architecture that the system possess.

Motivation

3. Legacy Transformation

- It is a tough task to convert a 10,000 line **COBOL** code to **C/C++** code if the programmer is unaware of the underlying architecture.
- As around **70%** of world's source code is in **COBOL** and in scientific communities FORTRAN has been the obvious choice [2].

- [2] *Ali, M. R. (2005). Why teach reverse engineering? SIGSOFT Softw. Eng. Notes, 30(4), 1–4. https://doi.org/10.1145/1082983.1083004*

Motivation

4. System evolution

- As system evolves , it tends to drift from it's original architecture.
- So it is very important to recover or reconstruct the architecture of the system in the spirit that new changes to the system do not affect the existing working model.

Motivation

5. Software Testing and Security

- Techniques are used to **debug** and find bugs and errors.
- Techniques are used to make sure that the system does not have any major **vulnerabilities** and security flaws.

Binary Code Reverse Engineering Tools

- Disassemblers
- Debuggers
- Hex Editors
- PE and Resource Viewer
- Example:
  - IAD Pro, OllyDBG, WinDBG, etc.

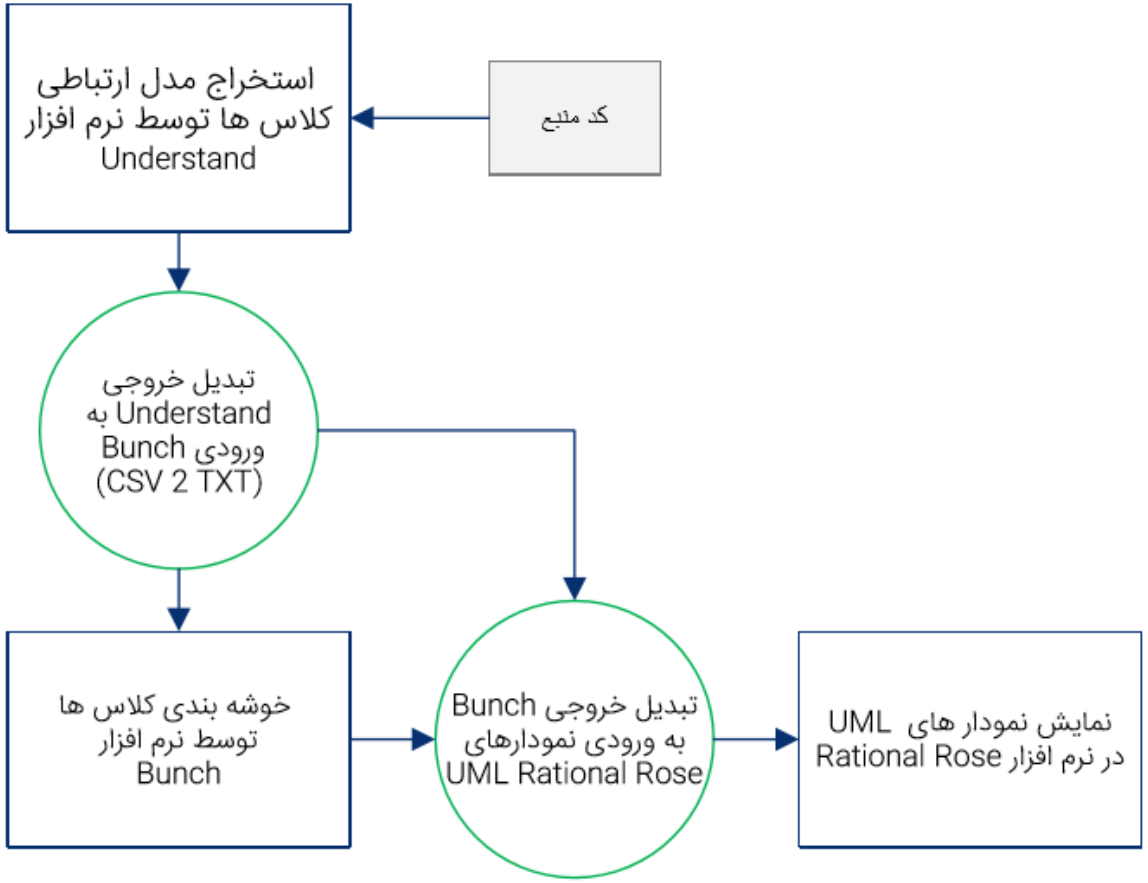
Source Code Reverse Engineering Tools

- Calculate some **software metrics.**
  - Lines of code (LoC)
  - Number of Class, Functions, Statements, etc.
  - **Visualize** source code architecture to optimize software design.
- Sometimes we need higher level of **abstraction**.
  - e.g. Component Diagram.

Modularity Principles

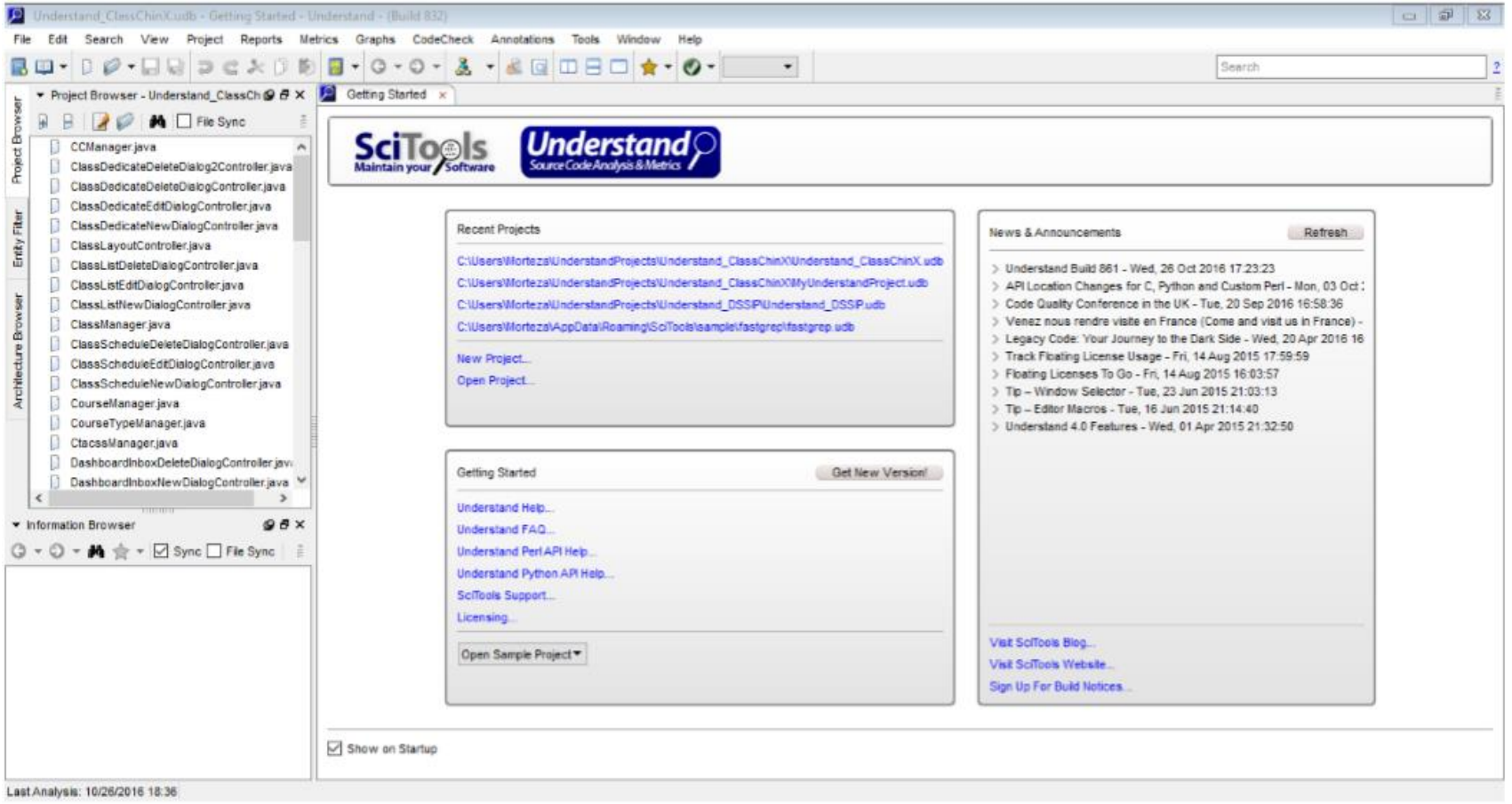
- Cohesion
  - Refers to the **degree** to which the elements of **a** module **belong together**.
- Coupling
  - The degree of **interdependence** between software modules.
- Good Design: High cohesion (↑), loose coupling (↓).

Reversing Steps



شکل ۲-۱ نمودار مراحل استخراج و نمایش معماری کد

Understand


File Edit Search View Project Reports Metrics Graphs CodeCheck Annotations Tools Window Help
Project Browser - Understand_ClassCh
File Sync
CCManager.java
ClassDedicateDeleteDialog2Controller.java
ClassDedicateDeleteDialogController.java
ClassDedicateEditDialogController.java
ClassDedicateNewDialogController.java
ClassLayoutController.java
ClassListDeleteDialogController.java
ClassListEditDialogController.java
ClassListNewDialogController.java
ClassManager.java
ClassScheduleDeleteDialogController.java
ClassScheduleEditDialogController.java
ClassScheduleNewDialogController.java
CourseManager.java
CourseTypeManager.java
CtacssManager.java
DashboardInboxDeleteDialogController.jav
DashboardInboxNewDialogController.java
Information Browser
Sync File Sync
Getting Started
SciTools Maintain your Software
Understand Source Code Analysis & Metrics
Recent Projects
C:\Users\Morteza\UnderstandProjects\Understand_ClassChinX\Understand_ClassChinX.udb
C:\Users\Morteza\UnderstandProjects\Understand_ClassChinX\MyUnderstandProject.udb
C:\Users\Morteza\UnderstandProjects\Understand_DSSP\Understand_DSSP.udb
C:\Users\Morteza\AppData\Roaming\SciTools\sample\fastgrep\fastgrep.udb
New Project...
Open Project...
News & Announcements
Refresh
Understand Build 861 - Wed, 26 Oct 2016 17:23:23
API Location Changes for C, Python and Custom Perl - Mon, 03 Oct
Code Quality Conference in the UK - Tue, 20 Sep 2016 16:58:36
Venez nous rendre visite en France (Come and visit us in France) -
Legacy Code: Your Journey to the Dark Side - Wed, 20 Apr 2016 16
Track Floating License Usage - Fri, 14 Aug 2015 17:59:59
Floating Licenses To Go - Fri, 14 Aug 2015 16:03:57
Tip – Window Selector - Tue, 23 Jun 2015 21:03:13
Tip – Editor Macros - Tue, 16 Jun 2015 21:14:40
Understand 4.0 Features - Wed, 01 Apr 2015 21:32:50
Getting Started
Get New Version!
Understand Help...
Understand FAQ...
Understand Perl API Help...
Understand Python API Help...
SciTools Support...
Licensing...
Open Sample Project
Visit SciTools Blog...
Visit SciTools Website...
Sign Up For Build Notices...
Show on Startup
Last Analysis: 10/26/2016 18:36

Bunch

- Bunch is a graph clustering tools.
- As part of Ph.D. thesis in Computer Science at Drexel University.
- By: **Brian Mitchell**
- Using **heuristic** searching such as
  - **Genetic Algorithm**,
  - **Hill Climbing**,
  - **…**

بررسی یک مثال ساده

- برنامه ای مشتمل بر ۵ کلاس **a, b, c, e, f** را در نظر بگیرید.
- با یک بررسی **دستی** ما اطلاعاتی از تعداد دستیابی های متدهای هر کلاس بـه متدهای کلاس دیگر به دست می آوریم. برای مثال مشاهده می کنیم که در کلاس a دو بار متدهای کلاس b فراخوانی شده است.
- ما این اطلاع را به صورت زیر نشان می دهیم:
  - a b 2

گام اول: استخراج مدل ارتباطی در قالب گراف وابستگی کلاسی

```
1  Dependent Class,a,b,c,e,f
2  a,,2,,,1
3  b,,,4,,
4  c,3,,,,
5  e,,,4,,
6  f,,2,,6,
```

شکل ۴-۱ خروجی نمونه از ابزار Understand در قالب CSV

گام اول: استخراج مدل ارتباطی در قالب گراف وابستگی کلاسی

	A	B	C	D	E	F
1	Dependent Class	a	b	c	e	f
2	a		2			1
3	b			4		
4	c	3				
5	e			4		
6	f		2		6	
7						

شکل ۴-۲ خروجی نمونه از ابزار Understand که در محیط Microsoft Excel باز شده است.

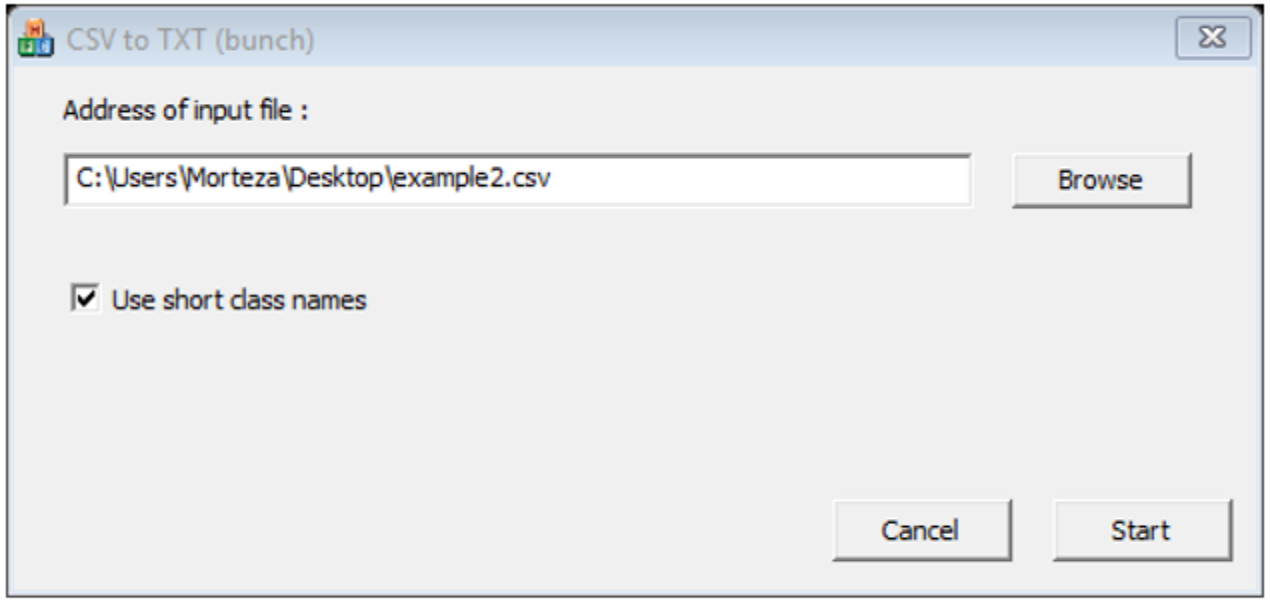
گام دوم: تبدیل خروجی Understand به ورودی Bunch

- در واقع خروجی گام قبل که گراف وزن دار جهت دار است، ورودی Bunch قرار می گیرد.

```
a b 2
a f 1
b c 4
c a 3
e c 4
f b 2
f e 6
```

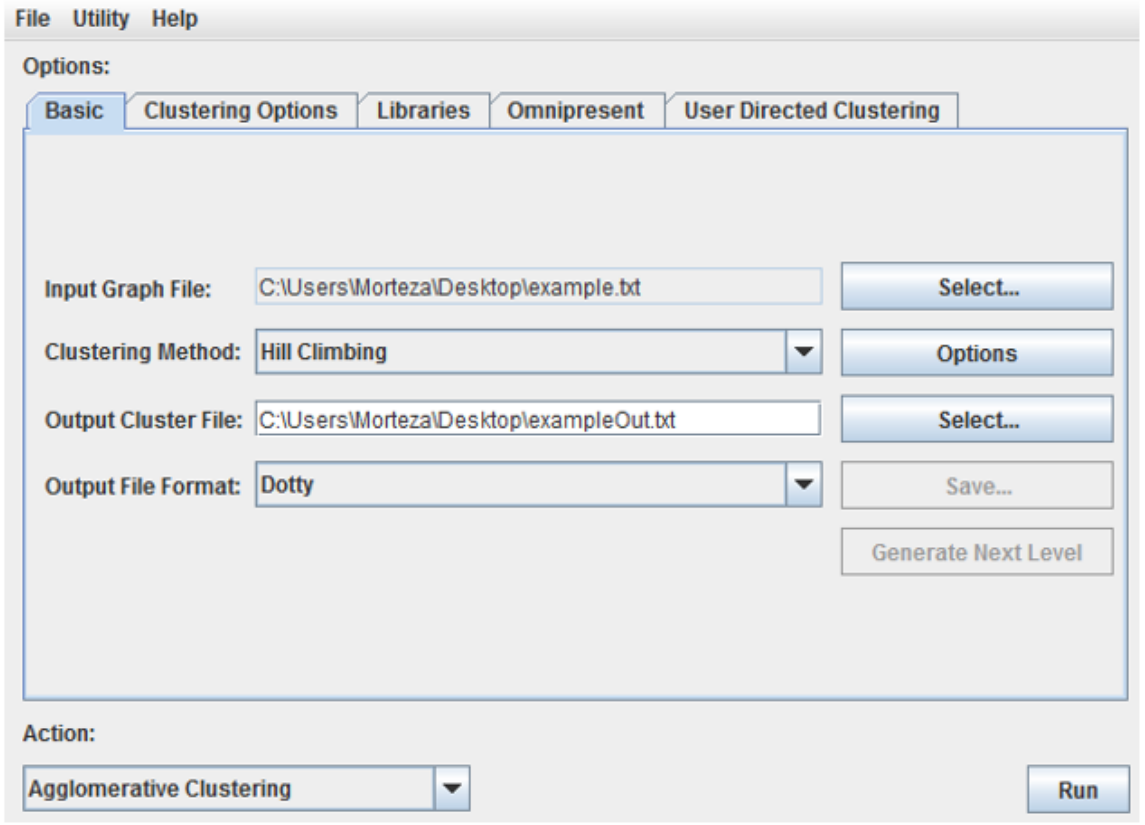
شکل ۴-۳ فرمت مناسب برای ورودی ابزار Bunch

گام دوم: تبدیل خروجی Understand به ورودی Bunch



شکل ۴-۴. ابزار و نحوه تبدیل خروجی Understand به ورودی Bunch

گام سوم: خوشه بندی



شکل ۴-۵ اتصال فایل ورودی به Bunch

Measuring Modularization Quality (MQ)

- **Basic MQ**
- **Inter-connectivity**
  - (i.e., connections between the components of two distinct clusters)
- **Intra-connectivity**
  - (i.e., connections between the components of the same cluster) [3].

Measuring Modularization Quality (MQ)

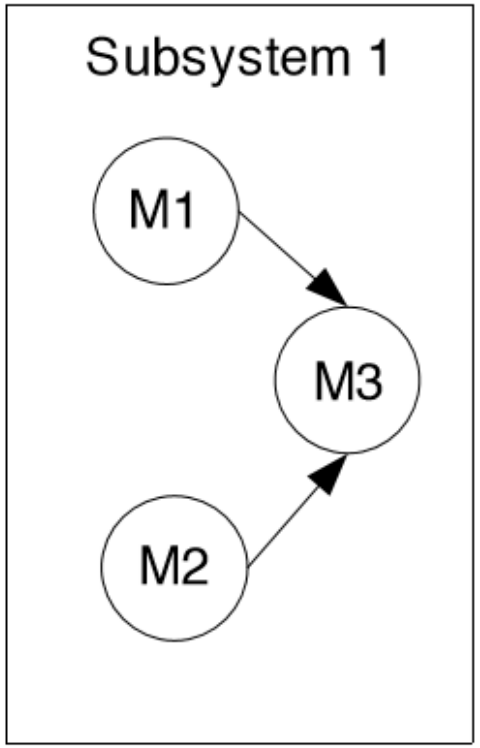
- Intra-connectivity measurement A_i of cluster i consisting of N_i components and μ_i intra-edge relations as:
- $A_i = \frac{\mu_i}{N_i^2}$

Measuring Modularization Quality (MQ)

- Inter-connectivity $E_{i,j}$ between clusters i and j consisting of N_i and N_j components, respectively, and with ε_i inter-edge dependencies as:

$$E_{i,j} = \begin{cases} 0 & \text{if } i = j \\ \frac{\varepsilon_{i,j}}{2N_iN_j} & \text{if } i \neq j \end{cases}$$

Measuring Modularization Quality (MQ)



Number of Modules in Subsystem: $N_i = 3$

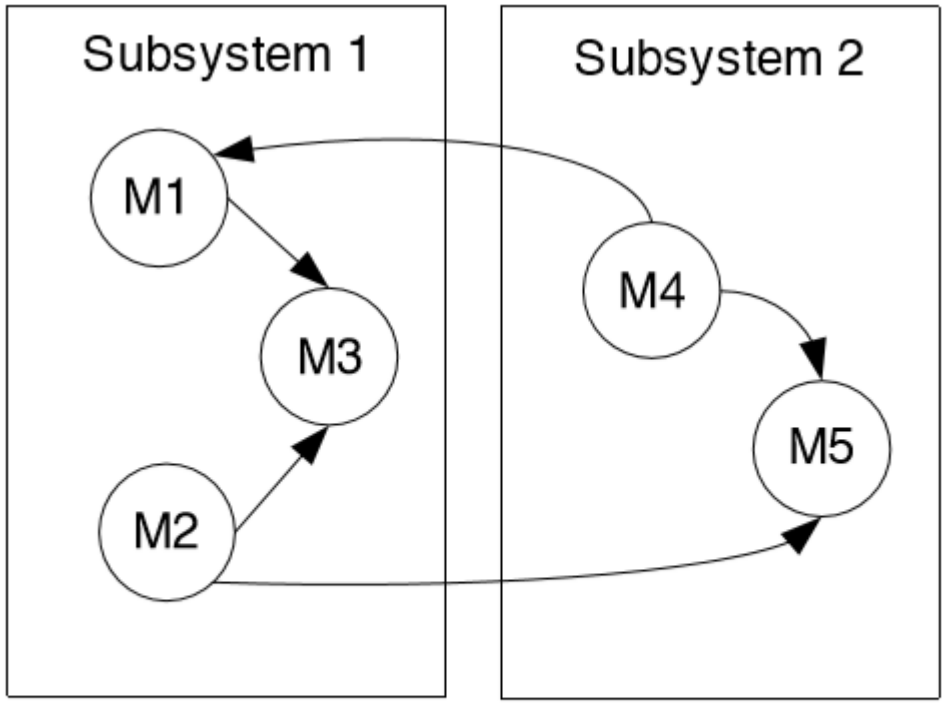
Number of Intra-Edge Relations: $\mu_i = 2$

Number of Inter-Edge Relations: $N_i^2 = 9$

$$A_1 = \frac{\mu_i}{N_i^2} = \frac{2}{9} = 0.2222...$$

Figure 3.4: Intra-Connectivity Calculation Example

Measuring Modularization Quality (MQ)



Number of Modules in Subsystem 1: $N_1 = 3$

Number of Modules in Subsystem 2: $N_2 = 2$

Number of Inter-Edge Relations: $\varepsilon_{1,2} = 2$

$$E_{1,2} = \frac{\varepsilon_{1,2}}{2N_1N_2} = \frac{2}{12} = 0.1666....$$

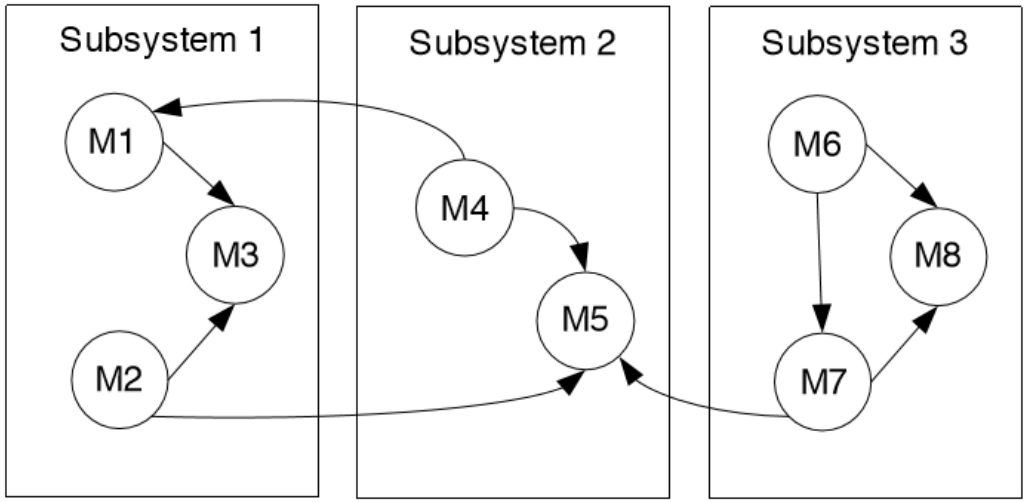
Figure 3.5: Inter-Connectivity Calculation Example

The BasicMQ Measurement

- The BasicMQ measurement demonstrates the **tradeoff** between inter-connectivity and intra-connectivity by rewarding the creation of **highly-cohesive** clusters, while penalizing the creation of too many inter-edges (k is number of subsystems):

$$BasicMQ = \begin{cases} \frac{1}{k}\sum_{i=1}^{k} A_i - \frac{1}{\frac{k(k-1)}{2}}\sum_{i,j=1}^{k} E_{i,j} & \text{if } k > 1 \\ A_1 & \text{if } k = 1 \end{cases}$$

The BasicMQ Measurement



$$MQ = \underbrace{\frac{\frac{2}{9}+\frac{1}{4}+\frac{3}{9}}{3}}_{\text{Average A}} - \underbrace{\frac{\frac{2}{12}+\frac{0}{18}+\frac{1}{12}}{3}}_{\text{Average E}} = \frac{5}{27} = 0.185....$$

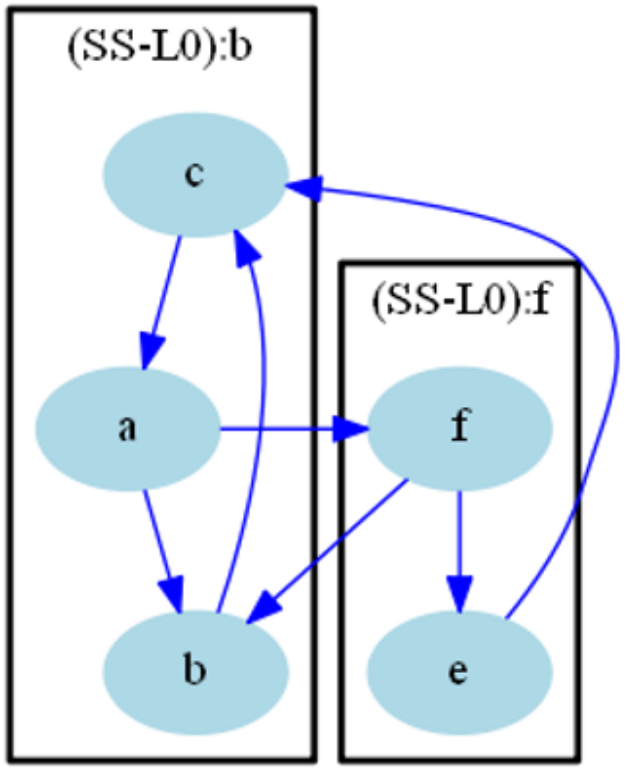
Figure 3.6: Modularization Quality Calculation Example

گام سوم: خوشه بندی

```
/* ------------------------------------------------------------ */
/* created with bunch v3 */
/* Objective Function value = 0.28*/
/* ------------------------------------------------------------ */

digraph G {
size= "10,10";
rotate = 90;
subgraph cluster0 {
label = "(SS-L0):b";
color = black;
style = bold;

"c"[label="c",shape=ellipse,color=lightblue,fontcolor=black,style=filled];
"e"[label="e",shape=ellipse,color=lightblue,fontcolor=black,style=filled];
"f"[label="f",shape=ellipse,color=lightblue,fontcolor=black,style=filled];
"a"[label="a",shape=ellipse,color=lightblue,fontcolor=black,style=filled];
"b"[label="b",shape=ellipse,color=lightblue,fontcolor=black,style=filled];
}
"b" -> "c" [color=blue,font=6];
"a" -> "b" [color=blue,font=6];
"a" -> "f" [color=blue,font=6];
"f" -> "b" [color=blue,font=6];
"f" -> "e" [color=blue,font=6];
"e" -> "c" [color=blue,font=6];
"c" -> "a" [color=blue,font=6];
}

```

شکل ۴-۸ خروجی تولید شده توسط Bunch

گام سوم: خوشه بندی (نمایش در Graphviz)

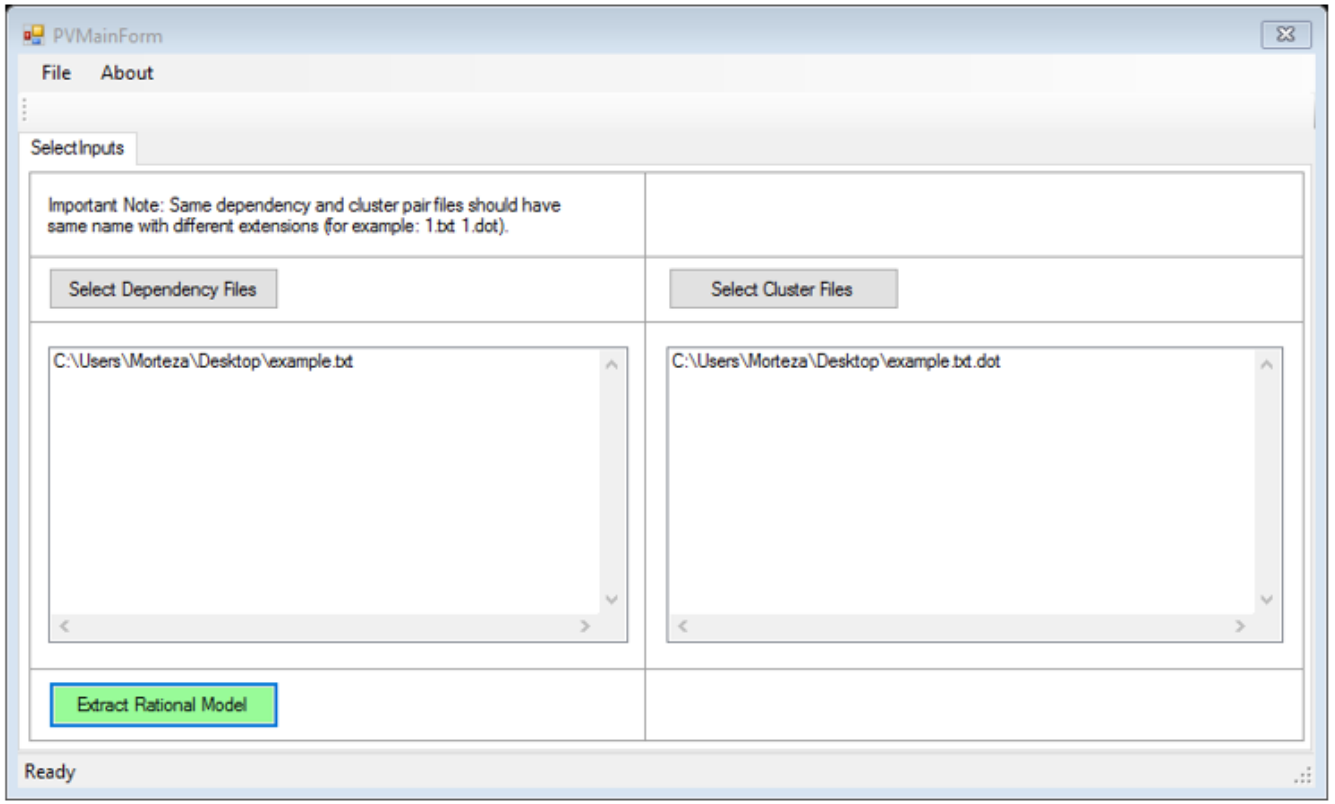


شکل ۴-۹ خروجی حاصل از خوشه بندی انجام شده توسط Bunch، نمایش داده شده در Graphviz

گام چهارم: استخراج طرح معماری

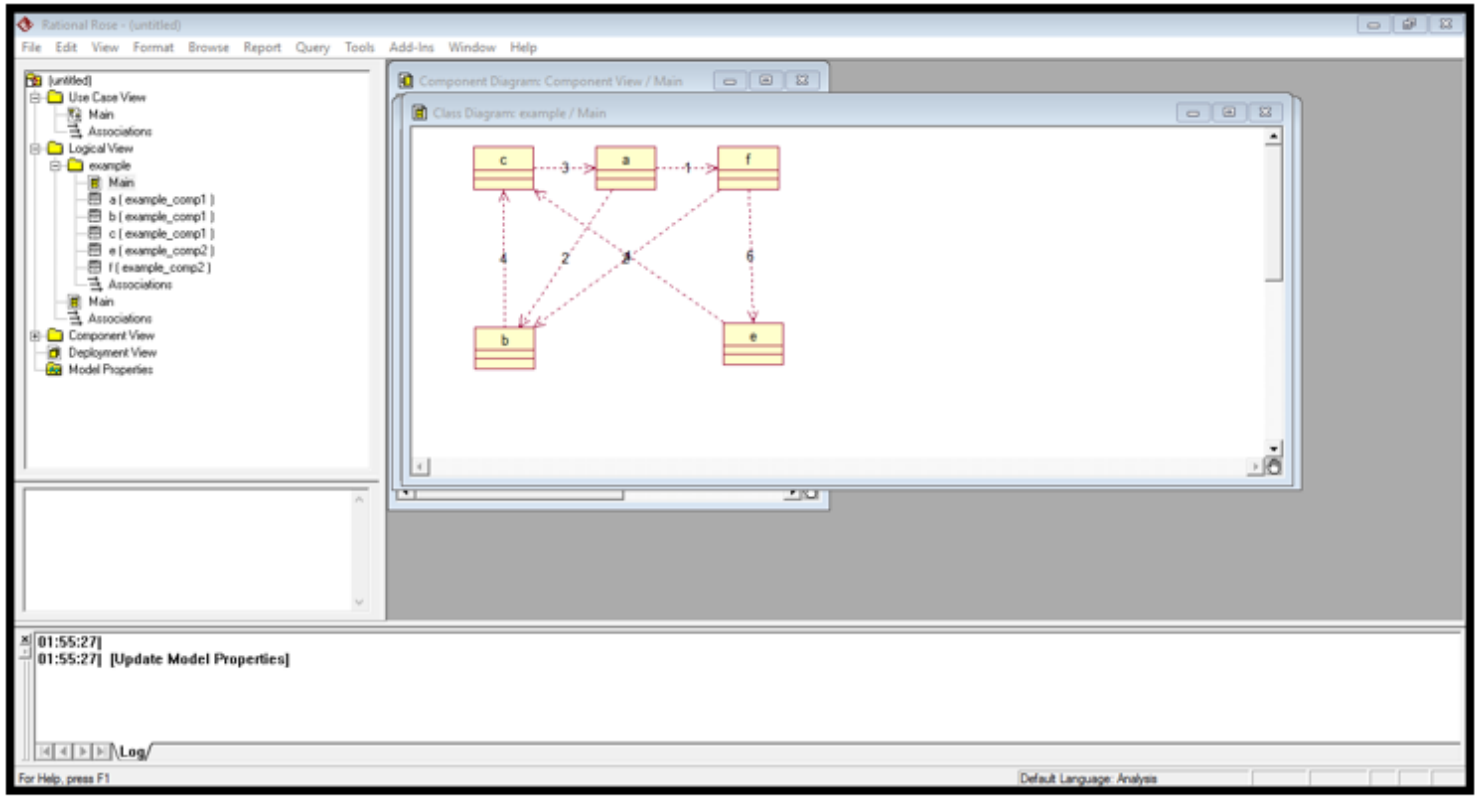
- IBM Rational Rose
  - *rationalrose.tlb*
  - *ProgramFiles\Rational\Rose\rationalrose.tlb*
  -

گام چهارم: استخراج طرح معماری (ابزار Package View)



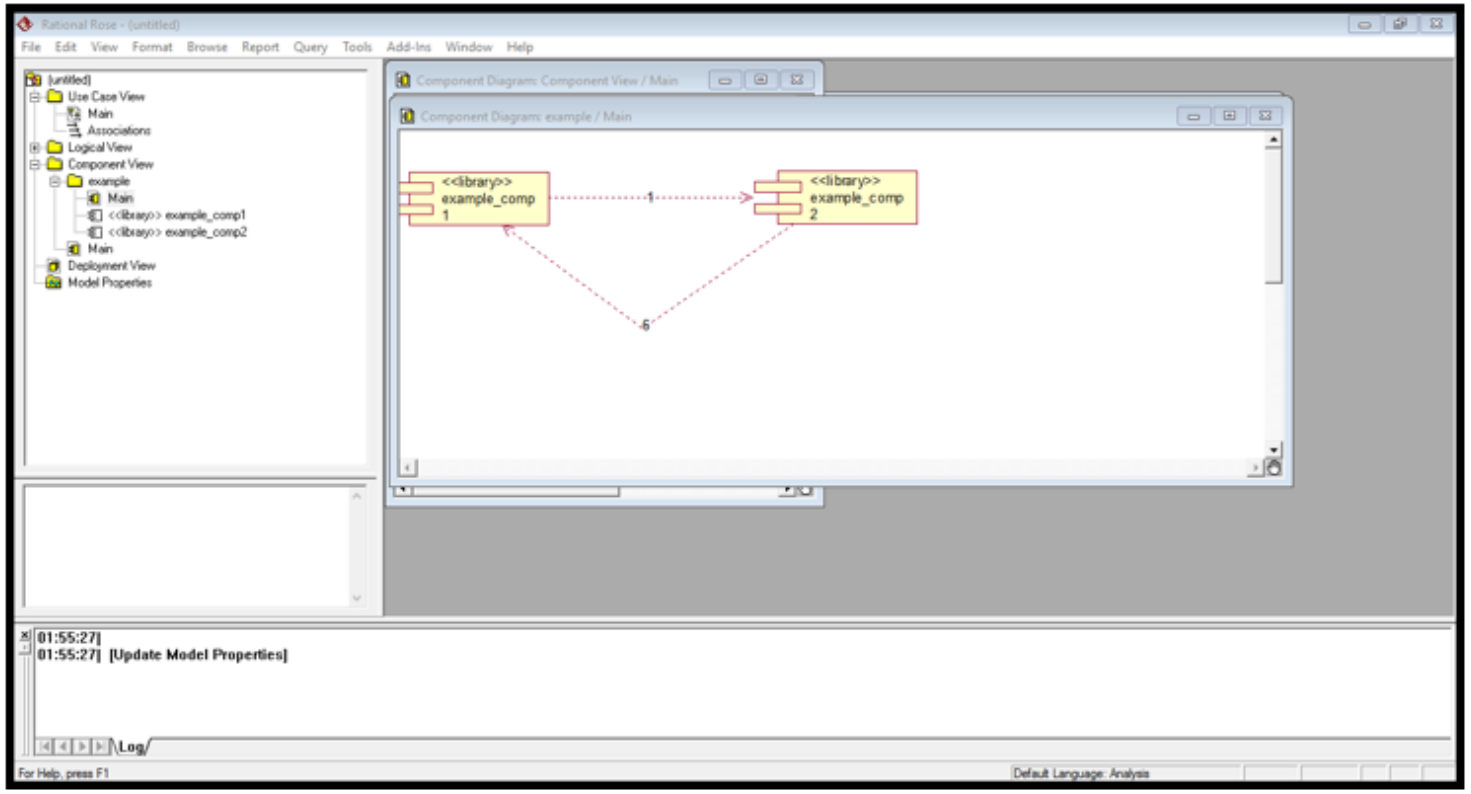
شکل ۴-۱۰ ابزار Package View

گام چهارم: نمایش در محیط عملیاتی Rational Rose



شکل ۴-۱۱ نمودار کلاس استخراج شده در محیط Rational Rose

گام چهارم: نمایش در محیط عملیاتی Rational Rose



شکل ۴-۱۲ نمودار قطعات استخراج شده در محیط Rational Rose

کارهای آتی

- تعمیم برنامه برای استخراج معماری برنامه های ساخت یافته
- تبدیل خودکار برنامه های ساخت یافته به شی گرا
- بررسی نحوه استخراج سایر نمودارهای طراحی از کد منبع
- نمایش در دیگر محیط های مشابه نظیر Visual Paradigm، Oracle Designer و ... .

References

- [1] Cipresso, T. (2009). Software Reverse Engineering Education, (August), 120.
- [2] *Ali, M. R. (2005). Why teach reverse engineering? SIGSOFT Softw. Eng. Notes, 30(4), 1–4. https://doi.org/10.1145/1082983.1083004*
- [3] Mitchell, B. S. (2002). A Heuristic Search Approach to Solving the Software Clustering Problem, (March).

Tools

- Understand
  - *https://scitools.com/*
- Bunch
  - *https://www.cs.drexel.edu/~spiros/bunch/*
- Graphviz
  - http://www.graphviz.org
- IBM Rational Rose Enterprise
  - http://www-03.ibm.com/software/products/en/enterprise

IUST Tools

- Understand 2 Bunch
- Bunch 2 Rational (Package Viewer)


Thank you for your attention!
Any question?
m-zakeri@live.com
mohsen_amirian@live.com