دانشکده مهندسی کامپیوتر

گروه مهندسی نرم افزار

عنوان پروژه:

روش گام به گام استخراج مدل معماری از متن کد برنامه

پیش نویس اول پروژه کلاسی **شماره ۱** درس کامپایلر پیشرفته

دانشجو:

مرتضی ذاکری

استاد:

دکتر سعید پارسا

پاییز ۱۳۹۵

فهرست مطالب

فهرست شکل ها

۱ مقدمه

مهندسی معکوس کد و استخراج معماری از کد منبع برنامه (به بیان دقیق تر استخراج مدل ارتباطی کلاس ها و تعیین طرح معماری نرم افزار) عمدتا با دو رویکرد کلی متفاوت صورت می پذیرد. نخست، تولید مستندات از متن برنامه موجود و از پیش نوشته شده، به منظور نگهداشت و توسعه آن و دوم، کشف ساختار کدهای حجیم به منظور درک عملکرد آن ها و تأثیرشان بر روی سیستم های کامپیوتری. ساختار برنامه هایی نظیر بد افزارها، آنتی ویروس ها و غیره.

آن چه در این تمرین به آن خواهیم پرداخت ارایه یه راهکار **گام به گام** ساده جهت استخراج ارتباط بین کلاس ها، خوشه بندی[۱] و نیز نمایش آن ها در محیط عملیاتی است. این راهکار در واقع **شرح استفاده سلسله مراتبی از مجموعه ای از ابزارها** برای محقق سازی هدف ذکر شده است. در واقع ما از مجموعه ای از ابزارها بهره می گیریم تا به یک خروجی کاربردی از ساختار یک کد موجود دست یابیم.

در ادامه به معرفی هر یک از ابزارها، توضیح روش استفاده از آن ها و گام های پنجگانه روش نام برده می پردازیم. در ابتدا مراحل را با یک مثال فرضی حاوی داده های غیر واقعی و ساده پیگیری نموده، نتایج به دست آمده را تحلیل می کنیم. سپس بر روی یک کد واقعی روش پیشنهادی را اجرا می کنیم. علت استفاده از مثال اولیه درک کلی جریان کاری که قرار است انجام شود خواهد بود. در غیر این صورت ممکن است روش کمی پیچیده به نظر آید. همچنین بررسی تکمیلی استخراج مدل معماری از کد واقعی در نظر گرفته شده در نسخه دوم این پیش نویس منتشر می شود.

۲ طرح اولیه مسئله و راه حل

فرض کنید کد منبع یک برنامه بزرگ در قالب تعداد زیادی کلاس به صورتی کاملا نامرتب در دسترس است. چگونه می توان ارتباط بین این کلاس ها پی برد؟ چگونه می توان کلاس های مرتبط را در یک خوشه[۲] قرار داد؟ برای پاسخ به این سوال ها نیازمند ابداع یک یا

[۱] Clustering
[۲] Cluster

چند متریک[۳] هستیم که بتوان بر حسب آن تصمیم اتخاذ کرد. در واقع با اندازه گیری پارامترهای متریک طرح شده به پاسخ می رسیم.

۲ـ۱ گراف وابستگی کلاس ها

بدیهی ترین پارامتر قابل محاسبه از روی فایل های حاوی کد منبع یک برنامه، تعداد دفعاتی است که هر فایل به دیگری ارجاع دارد. موردهای بسیاری را می توان به عنوان مصداق واژه ارجاع پذیرفت. از جمله فراخوانی تابع یک کلاس[۴] در کلاس دیگر، ارث بری، تعریف و نمونه سازی شی (ترکیب و تجمع) و غیره. حاصل محاسبه کمی این پارامترها را می توان به صورت این متریک تعمیم داد که به کلاس در رویایی با کلاس دیگر به چه اندازه ای درگیر ارتباط است. به این ترتیب می توان یک **گراف وزن دار جهت دار** از پویش کد منبع برنامه، تولید کرد که گره های آن مبین کلاس ها و یال های آن مبین ارتباطات بین آن ها، وزن هر یال بیان کننده میزان ارتباط و بلاخره جهت هر یال نشان دهنده طرف وابستگی است. تولید این گراف با استفاده از تکنیک گرامرهای ویژه و با نوشتن Action های مناسب برای گرامر زبان مورد نظر و سپس پویش متن برنامه امکان پذیر است.

۲ـ۲ خوشه بندی

مرحله بعد خوشه بندی کد منبع موجود است. تصور کنید گراف وابستگی تعداد بسیار زیادی از کلاس ها داده شده است. احتمالا درک چنین گرافی از درک کد برنامه برای شما زمان گیر تر خواهد بود (!). در نتیجه به سطح بالاتری از طبقه بندی یعنی خوشه بندی کلاس ها و سپس محاسبه ارتباطات بین خوشه ها نیازمندیم. با آن که ممکن است صورت مسئله در این قسمت ساده تر به نظر برسد، اما در این جا با مسیر های متعدد انتخاب رو به رو هستیم و روش قطعی برای نیل به خوشه بندی به همان دقتی که احتمالا کد اولیه خوشه بندی بوده است، وجود ندارد. به نظر می آید روش های جست و جوی آوینی[۵] در هوش مصنوعی، نظیر الگوریتم ها ژنتیک و تپه نوردی مؤثر واقع شود و تخمین دقیقی از نحوه خوشه بندی پیشنهاد کند. بنابراین در این مرحله به ابزاری برای خوشه بندی بر اساس این روش ها نیازداریم.

---

[۳] Metric

[۴] در مفهوم عام تر می توان از واژه **فایل** استفاده کرد، ممکن است کد منبع شی گرا نباشد و واژه کلاس بی معنی شود، با این حال در این تمرین رویکرد ما محدود به زبان های شی گرا می باشد.

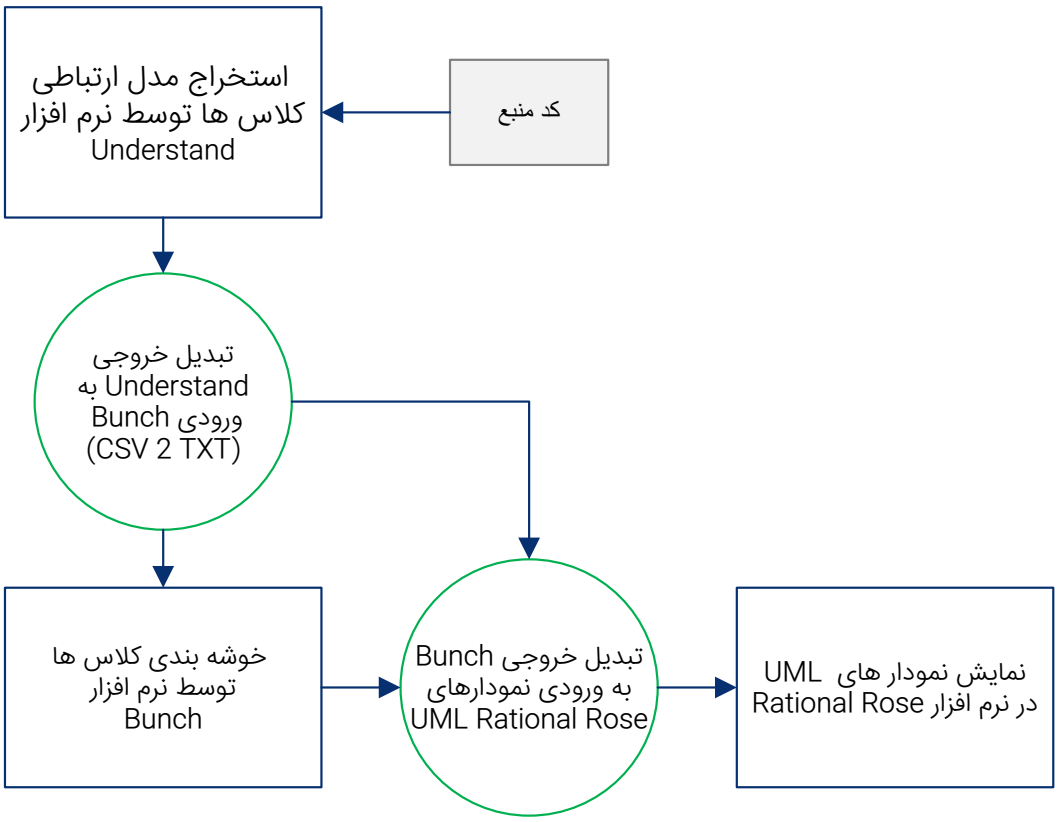
[۵] Heuristic

۲ـ۳ انتقال و عملیاتی سازی

خروجی حاصل از مرحله خوشه بندی مجددا می تواند به صورت گراف وزن دار جهت داری باشد که آن را گراف وابستگی خوشه ها می نامیم. در واقع عدد وزن یال ها مجموع اعداد ارتباط کلاسه های داخل دو خوشه ی مجزا است. هرچند چنین خروجی ارزشمند است اما قابل بهره برداری نیست. به منظور عملیاتی سازی استفاده از خوشه ها می توان آن ها را به یک محیط طراحی و توسعه نرم افزار انتقال داده که در آن این اجزا به صورت فعال[۶] و تعاملی[۷] در دسترس هستند. پس مرحله سوم انتقال و نمایش در محیط عملیاتی است.

۲ـ۴ نمودار مراحل استخراج معماری

مراحل شرح داده شده را می توان در نمودار زیر به صورت خلاصه وار دید. مستطیل ها گام های اصلی و دایره ها گام های فرعی مراحل انجام کار را نشان می دهند.



شکل ۲-۱ نمودار مراحل استخراج و نمایش معماری کد

[۶] Active

[۷] Interactive

۳ معرفی ابزار

در این قسمت ابزارهای مورد نیاز معرفی می شوند. برای هر سه مرحله اصلی شرح داده شده در بخش قبلی ابزار وجود دارد. این ابزارها منحصر به فرد نیستند و ساخت نوع جدید از هر کدام ارزش علمی و عملی بالایی دارد. ما در این تمرین روش خود را بر مبنای سلسله ابزارهای زیر ارایه می دهیم. برای آن که خروجی هر ابزار قابل استفاده در ورودی ابزار بعدی باشد، نیازمند نوشتن تکه برنامه هایی هستیم که این تطابق را بر عهده دارند. بنابراین دو ابزارک واسط (بین هر دو مرحله یکی)، نیز در این جا معرفی می شوند. در پایان کار یک ابزار هدایت گام به گام نوشته شده است که دنبال کردن این سناریو را تسهیل می کند.

ابتدا نیازمند ابزاری برای استخراج گراف وابستگی از متن کد برنامه هستیم. ابزار Scientific Tool works Understand که در بخش ۳ـ۱ معرفی شده، علاوه بر امکان نام برده، امکانات اضافه تری نیز دارد. پس از آن از ابزار Bunch برای خوشه بندی استفاده می کنیم. در پایان نیز نمودارهای کلاس و قطعه را در IBM Rational Rose تولید می کنیم.

۳ـ۱ ابزار Scientific Tool works Understand

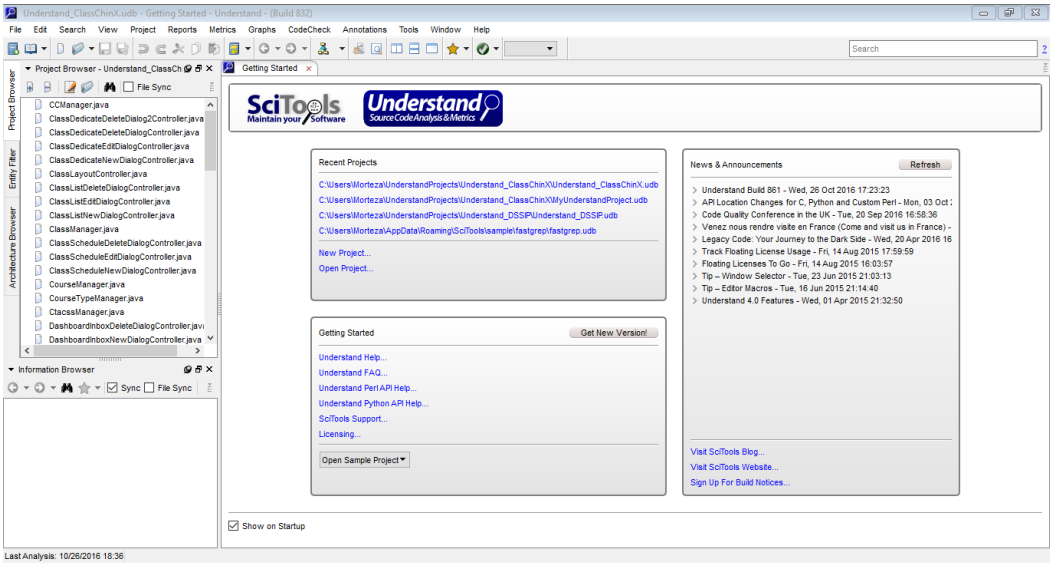
نرم افزار Scientific Tool works Understand یک ابزار تجزیه و تحلیل استاتیک برای حفاظت، اندازه گیری و تجزیه تحلیل انتقادی برای پایگاه های بزرگ کد (های برنامه نویسی) است. این برنامه توانایی شناخت پلت فرم های متقابل، پشتیبانی از چندین زبان و همچنین محیط های برنامه نویسی (محیط توسعه مجتمع مثل ویژوال استادیو)، تعمیر و نگهداری کد را دارد. این نرم افزار برای کمک به حفظ و ایجاد کد منبع طراحی شده که این کد های منبع می توانند شامل کد های منبع JOVIAL, Java, FORTRAN, C#, C++ و یا Delphi/Pascal با شند. این برنامه توانایی شناخت روابط و ساختار های موجود در پروژه های نرم افزاری را دارد. چند نمونه پارامترهای اصلی که این برنامه بررسی می کند عبارت است از:

- تعداد کلاس
- تعداد فایل
- تعداد خط
- تعداد خط های خالی
- تعداد خط های کد

- تعداد خط های توضیحات
- تعداد خط های غیر فعال
- تعداد Declarative Statement
- تعداد Executable Statement
- تعداد Ratio Comment to

علاوه بر آن Understand پارامتر های پیشرفته تری را نیز بررسی و تحلیل می کند، از جمله:

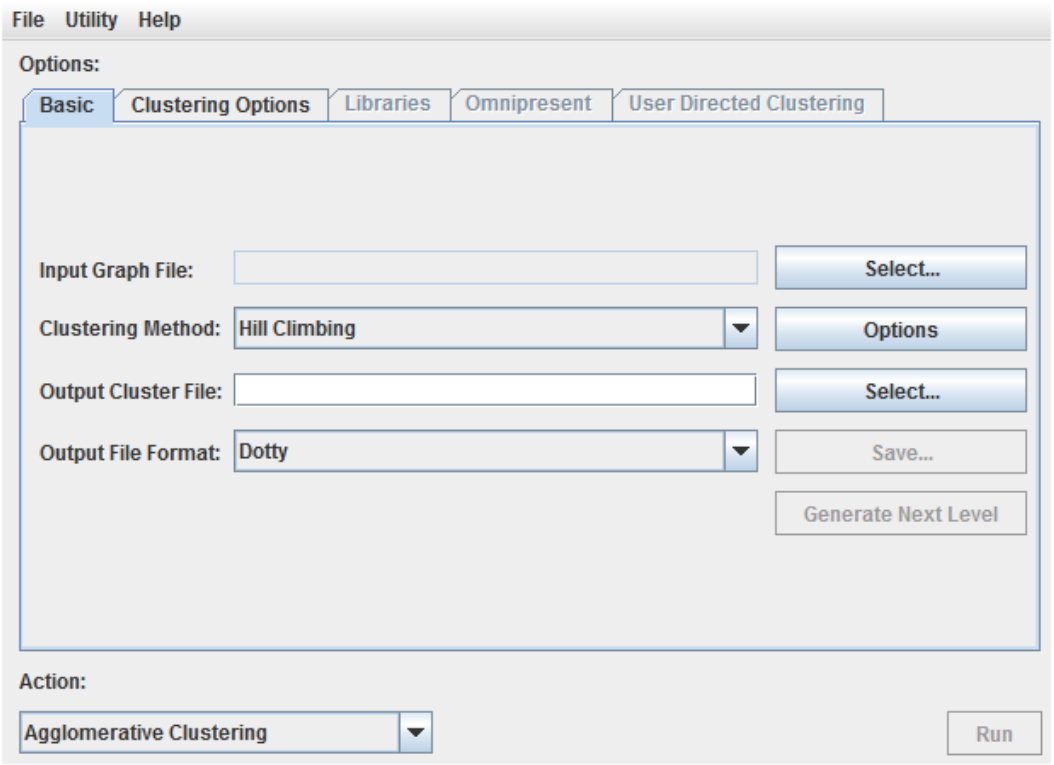
- Cyclomatic Complexity
- گره ها
- کلاس Coupling
- درصد عدم انسجام
- تعداد راه
- Max Inheritance
- تعداد کلاس پایه
- تعداد کلاس ارث برنده
- تعداد تابع های instance
- وزن تابع در هر کلاس



شکل ۳-۱ نرم افزار Understand

۳ـ۲ ابزار Bunch

نرم افزار Bunch که حاصل کار تز دکتری آقای Brian S. Mitchell از دانشگاه Drexel است در واقع یک جست و جوی مبنی بر روش های آوینی هوش مصنوعی را برای خوشه بندی بهتر انجام می دهد. اطلاعات بیش تر را در صفحه اینترنتی این ابزار به نشانی https://www.cs.drexel.edu/~spiros/bunch/ یافت.



شکل ۳-۲ نرم افزار Bunch

۳ـ۳ ابزار IBM Rational Rose Enterprise

یکی از ابزارهای مدلس ازی نرم افزار از طریق عمومی ترین زبان مدل سازی (UML) نرم افزار Rational Rose Enterprise می با شد. این نرم افزار سرعت و دقت عمل مدل سازی محصول را بالا می برد. به کمک این نرم افزار می توانید به راحتی به آنالیز و مدل سازی محصولات تهیه شده به زبان های C++، ANSI C++، Visual C++، CORBA، Java، Visual Basic بپردازید. به علاوه می توانید برنامه های تحت وب تان را مدل سازی کنید. این نرم افزار

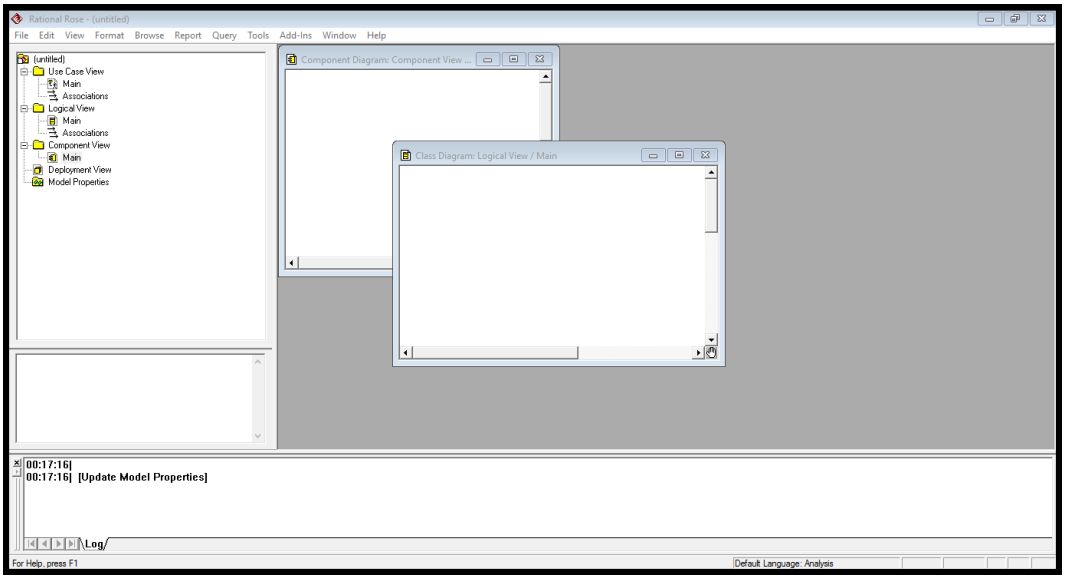
با سـایر ابزار های تولید شـده توسـط کمپانی IBM Rational در زمینه ی توسـعه ی چرخه ی حیات نرم افزار هماهنگی دارد. که از این ابزار ها می توان به نرم افزار های SCC-compliant version control system و IBM Rational Clear Case اشاره کرد.

به کمک این نرم افزار می توان به مدل سازی بر ا ساس UML به منظور طراحی پایگاه داده پرداخت. این نرم افزار از سـیسـتم عامل ویندوز پشـتیبانی می کند. از دیگر امکانات آن ساخت DTD با فرمت XML به منظور تو سعه ی مح صول می با شد. نمودارهای موجود در این نرم افزار به شرح زیر می باشند:

- Class
- Component
- Deployment
- Sequence
- State Chart
- Use Case
- Collaboration
- Physical Storage Physical Data / Table

قابلیت های کلیدی نرم افزار IBM Rational Rose Enterprise:

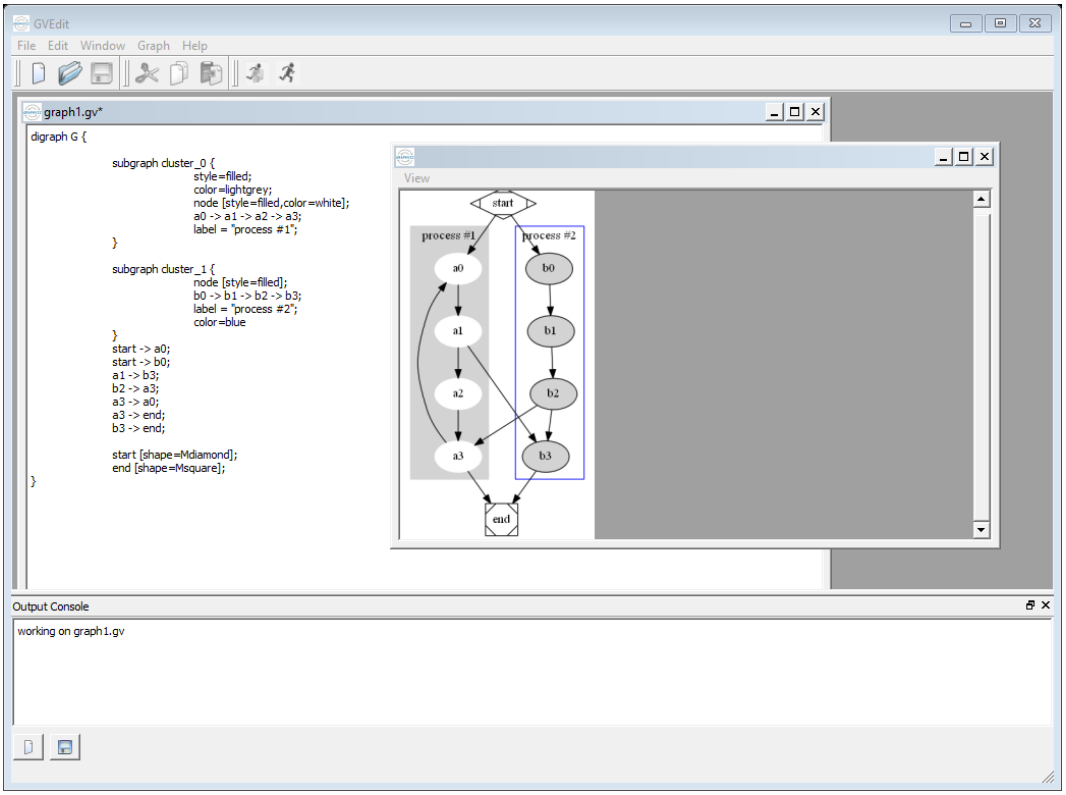
- مدل سازی محصول بر اساس زبان UML
- آنالیز و مدل سازی محصولات تهیه شده به زبان های C++، ANSI C++، Visual C++، CORBA، Java، Visual Basic
- سازگاری با سایر نرم افزارهای تولید شده در زمینه ی توسعه ی نرم افزار
- ساخت DTD با فرمت XML
- پشتیبانی از تکنولوژی RUP
- مدل سازی به منظور طراحی پایگاه داده
- پشتیبانی از سیستم عامل ویندوز



شکل ۳-۳ نرم افزار IBM Rational Rose

۳ـ۴ ابزار Graphviz

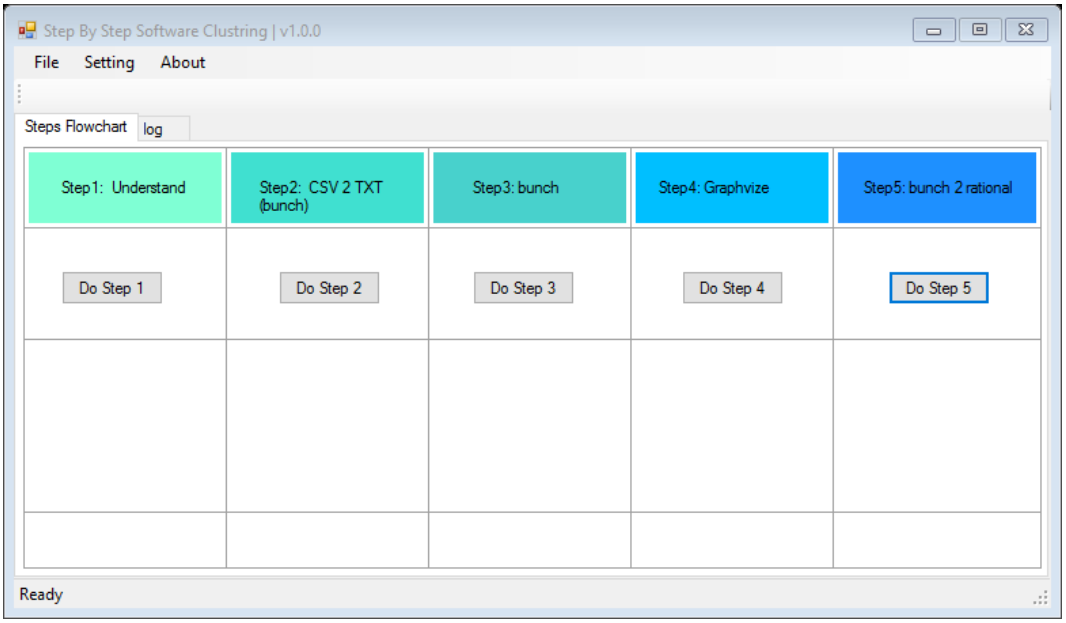
Graphviz یک ابزار مصور سازی ساختارهای گراف گون است. اگر چه در این مسئله ما با این ابزار ســـر و کار نداریم، ولی در حقیقت خروجی ابزار Bunch کدی اســـت که به عنوان ورودی Graphviz توسط آن قابل اجراست.



شکل ۳-۴ ابزار Graphviz

۳ـ۵ ابزار جامع گام به گام استخراج معماری کد

این ابزار جهت تسهیل انجام مراحل شرح داده شده در نمودار شکل ۲-۱ تولید شده است و حاوی دو **ابزارک واسط** جهت تبدیلات بین مرحله ای است. در گام چهارم عنوان شده در این ابزار، Graphviz گنجانده شده است که جزو مراحل کار ما نیست. توجه شود که گام های موجود در این نرم افزار دقیقا متناسب با مراحل پنجگانه شرح داده شده در بخش قبلی نیستند ولی در مجموع همین مراحل را طی می کنند.



شکل ۳-۵ ابزار Step By Step Software Clustring

۴ بررسی یک مثال فرضی

۴ـ۱ ورودی مسئله: کد منبع

فرض کنیم برنامه ای مشتمل بر ۵ کلاس a, b, c, e, f به ما داده شده است. با یک بررسی دستی ما اطلاعاتی از تعداد دستیابی های متدهای هر کلاس به 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

در ادامه سه گام اصلی که به منظور استخراج معماری این کد بایستی طی شود را شرح می دهیم.

۴ـ۲ گام اول: استخراج گراف وابستگی کلاسی

به همین ترتیب کلیه ارتباطات بین کلاسی را استخراج می کنیم. برای این کار از ابزار Understand استفاده می کنیم که خروجی در قالب فایل CSV می دهد. این فایل را می تواند به صورت متنی مشاهده کرد و یا با نرم افزار Excel باز کرد که به صورت مرتبی اطلاعات را

نشـــان می دهد. یک خروجی نمونه که کاملا فرضـــی و بر مبنای اطلاعات غیر واقعی اســـت در شکل های زیر نشان داده شده است.

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

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

	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.

۴ـ۳ گام دوم: خوشه بندی

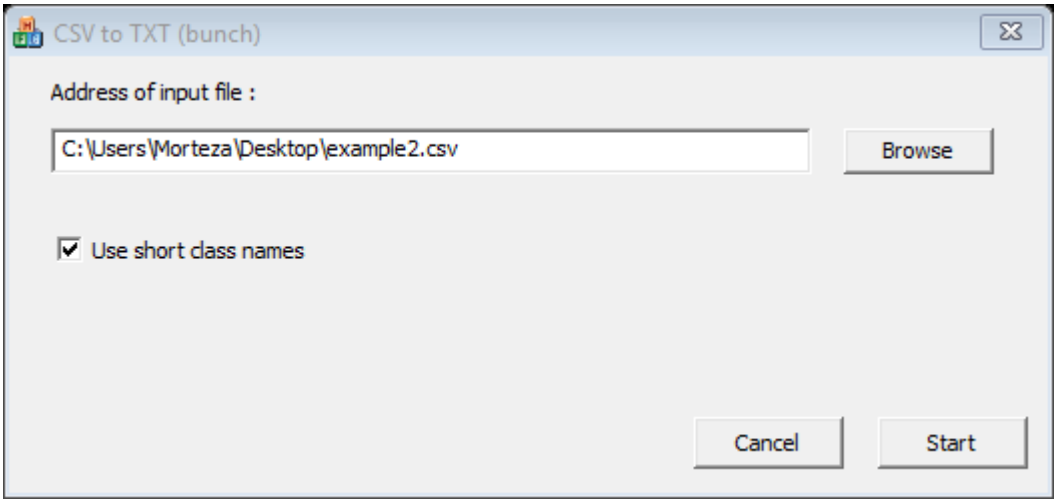
حال برای خوشه بندی این ۵ کلاس به وسیله ابزار Bunch (دقت کنید که در این مثال به دلیل *کم بودن کلاس ها* ممکن است بتوان با یک نگاه ساده خوشه بندی خوبی را تشخیص داد، اما در پروژه های واقعی انجام چنین کاری تقریبا غیر ممکن اســت)، باید این فایل خروجی را به فرمت ورودی ابزار Bunch تغییر دهیم. فرمت ورودی Bunch به شکل زیر است:

```
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

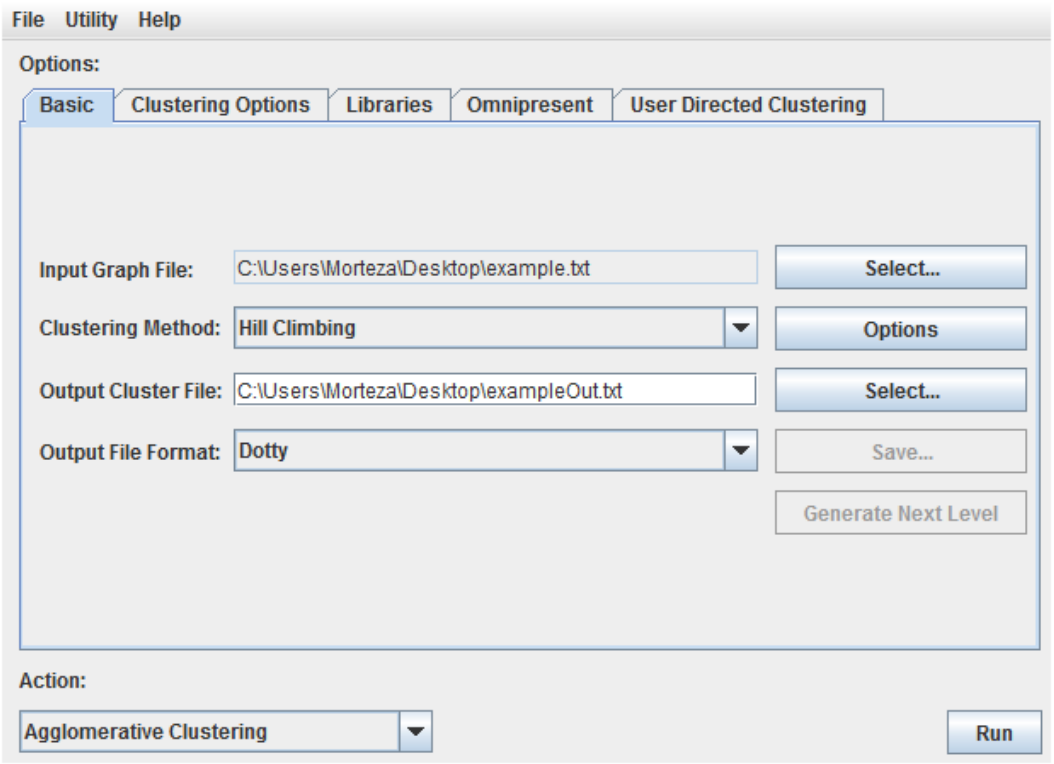
که همان نحوه اولیه نمایش اطلاعات توسط خودمان بود. در واقع این نمایش به ساده ترین شکل، توصیف یک **گراف وزن دار جهت دار** می باشد. به عنوان نمونه سطر اول فایل می گوید از a به b یالی با وزن 2 وجود دارد.

حال با ا ستفاده از ابزار CSV 2 TXT موجود در گام ۲ برنامه Step By Step Software Clustring می توان خروجی شکل ۴-۳ را از ورودی شکل ۴-۱ تولید کرد (شکل ۴-۴).

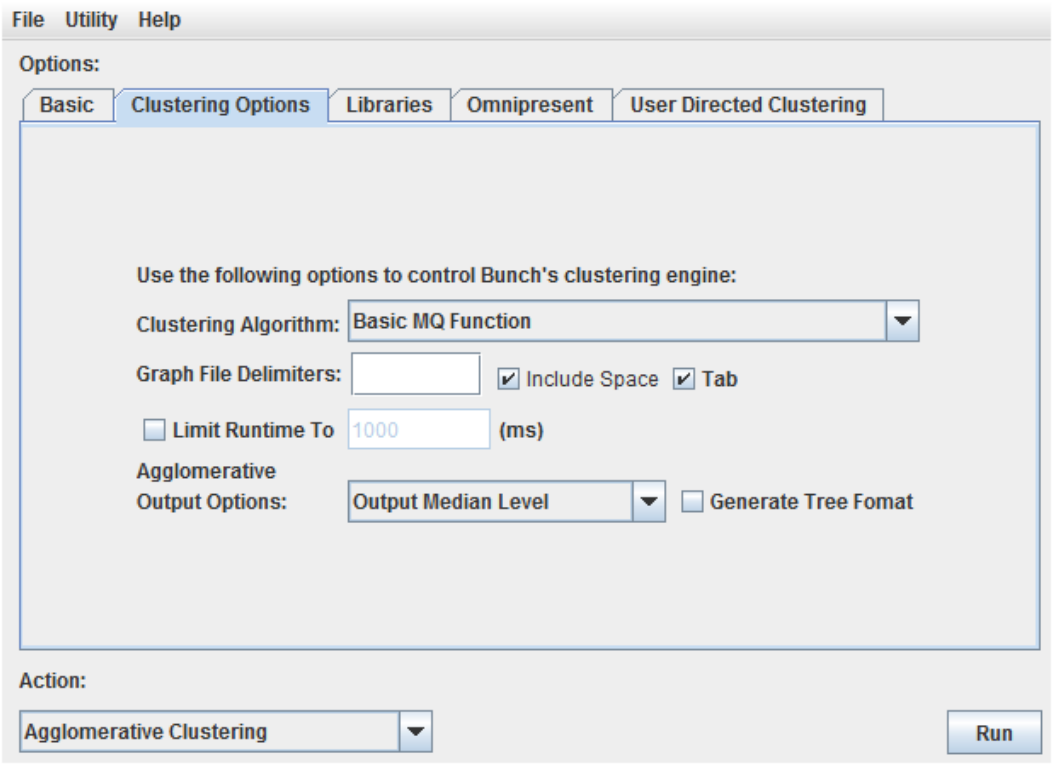


شکل ۴-۴. ابزار و نحوه تبدیل خروجی Understand به ورودی Bunch

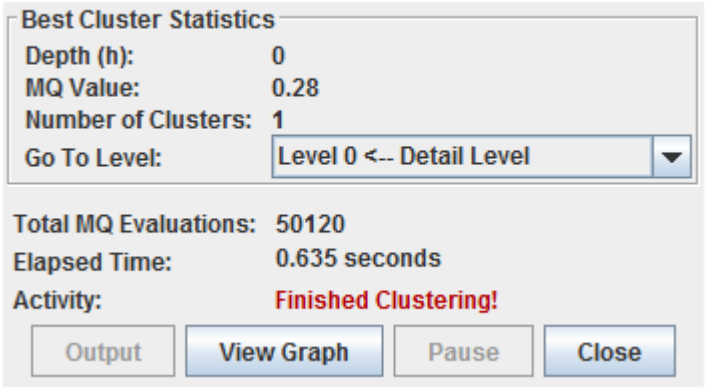
در مرحله بعدی با توجه به شکل ۴-۵ و شکل ۴-۶ که نحوه تنظیمات ابزار Bunch را نشان می دهد. این برنامه را روی ورودی شکل ۴-۳ اجرا کرده و خروجی نمایش داده شده در شکل ۴-۸ را به دست می آوریم که توسط نرم افزار Graphviz قابل تفسیر است. می توان یک شمای گرافیکی از خو شه بندی انجام شده را که تو سط Graphviz تولید شده در شکل ۴-۹ دید.



شکل ۴-۵ اتصال فایل ورودی به Bunch



شکل ۴-۶ تعیین تابع ارزیابی کیفیت در Bunch



شکل ۴-۷ اجرای موفقیت آمیز Bunch

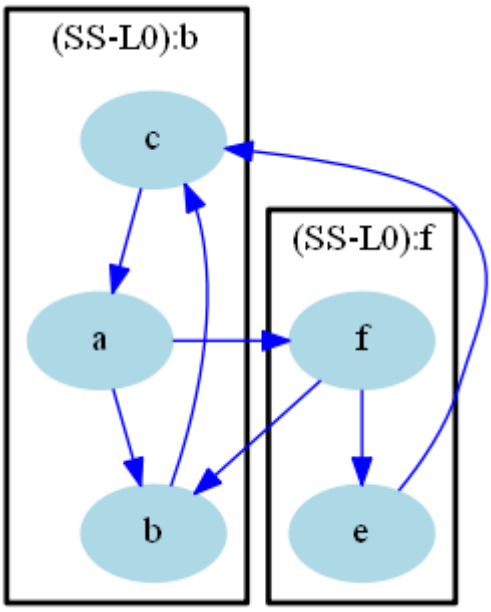
```
/* ------------------------------------------------------------ */
/* created with bunch v3 */
/* Objective Function value = 0.28*/
/* ------------------------------------------------------------ */

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



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

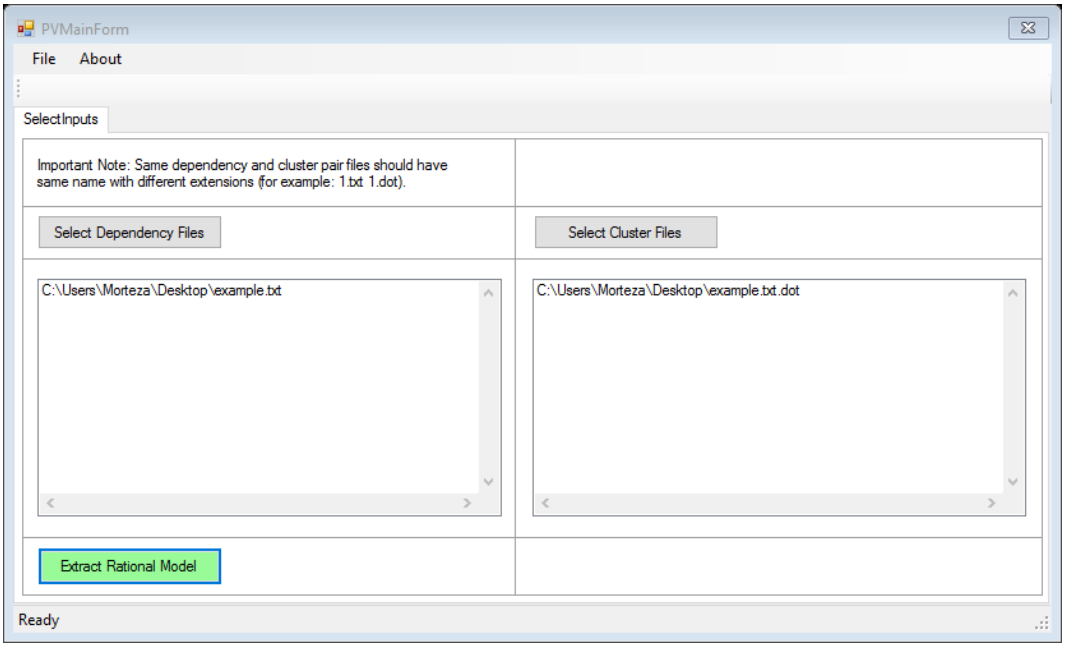
چنان چه از نمایش گرافیکی شـــکل ۴-۹ بر می آید، Bunch توانســـت دو خوشـــه را تشخیص دهد یعنی کلاس های برنامه فرضی ما را با توجه به حد اتصالات به دو مجموعه {a, b, c} و {f, e} افراز کرد. می توان الگوریتم های Bunch را تغییر داد و نتایج متفاوتی به دســـت آورد و بهترین نتیجه ممکن را برگزید. با این حال به دلیل وجود جســـت و جوهای مبنی بر هوش مصنوعی قطعیت در کار نیست.

۴ـ۴ گام سوم: نمایش در محیط Rational Rose

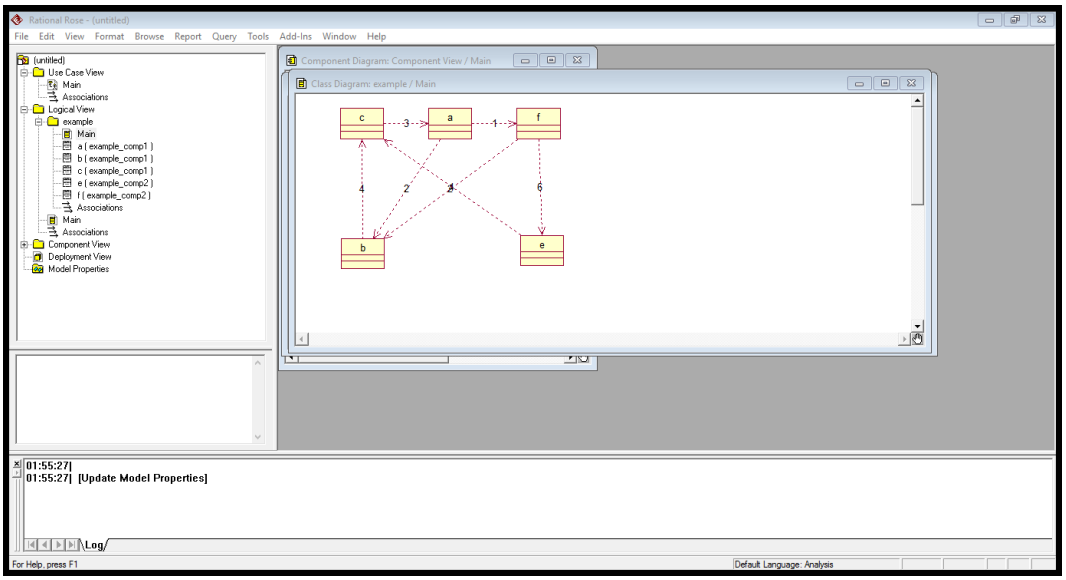
در قسمت پایانی ما علاقمندیم تا خروجی حاصل از گراف وابستگی و نیز خوشه بندی را به محیط Rational Rose انتقال دهیم. این انتقال با اســـتفاده از ابزار موجود در گام ۵، به عنوان Package View صـورت می گیرد که اسـتفاده از آن بسـیار سـاده بوده و شـبیه Bunch است.

ورودی این ابزار هر دو خروجی حاصـــل از گام های اول و دوم اســـت، یعنی هم فایل TXT. اولیه حاوی گراف وابســـتگی کلاس ها و هم فایل خروجی Bunch (Dotty.) را به عنوان ورودی به ابزار Package View می دهیم و نمودارهای UML را به عنوان خروجی نهایی در Rational Rose مشـــاهده می کنیم. ابزار Package View برای تولید و نمایش خروجی از کتابخانه های Rational Rose شرکت IBM استفاده می کند که برای اجرای صحیح آن بایستی IBM Rational Rose از قبل بر روی سیستم نصب گردیده باشد.

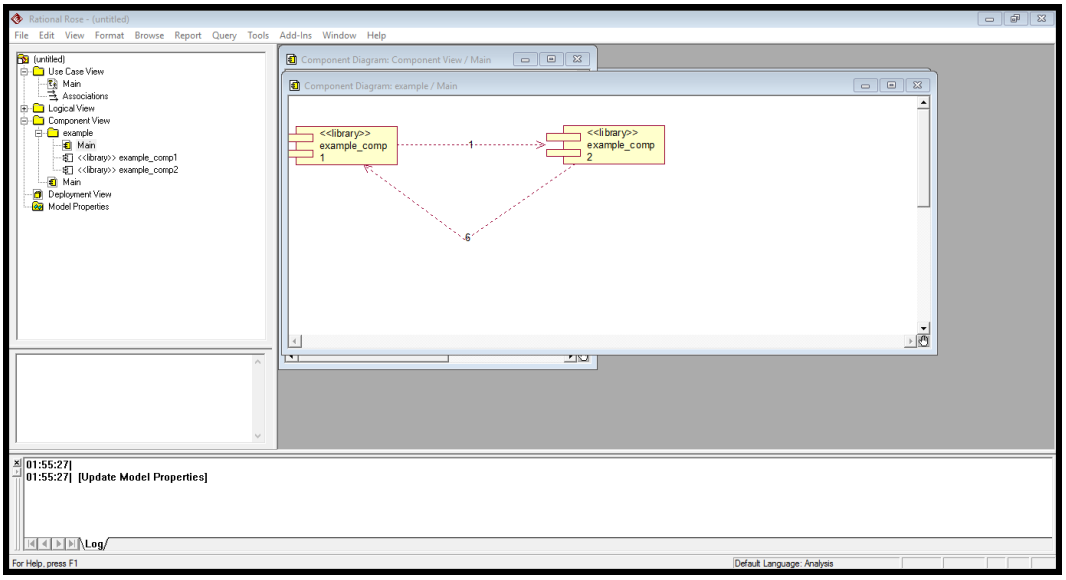
در شکل ۴-۱۰ ابزار Package View و نحوه استفاده از آن در این مثال نشان داده شده اســت. در شــکل ۴-۱۱ نمودار کلاس اســتخراج شــده از متن برنامه در محیط Rational Rose نشان داده شده است و بلاخره در شکل ۴-۱۲ نمودار قطعات استخراج شده از متن برنامه توسط ابزار Bunch که در این محیط عملیاتی نگاشت شده، به تصویر کشیده شده است.



شکل ۴-۱۰ ابزار Package View



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



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

بدین ترتیب مراحل ا ستخراج معماری از متن کد برنامه برای این مثال فر ضی خاص و نحوه استفاده زنجیری از ابزارهای گوناگون معرفی شده در بخش های قبل، به پایان می رسد و می توانیم این مرا حل را روی یک کد واقعی اجرا کنیم که در بخش بعدی به آن خواهیم پرداخت.

۵ استخراج مدل معماری نرم افزار کلاس چین

نرم افزار کلاس چین یک برنامه متن باز اســـت که جهت برنامه ریزی و چیدمان مکان کلاس های درسی دانشکده فنی و مهندسی دانشگاه اراک، تهیه شده است. این برنامه به زبان جاوا نوشته شده است و متن کد آن از وب سایت Github قابل دسترسی است. در این قسمت قصد داریم تا با روش گام به گام طرح شده معماری این کد را ا ستخراج نماییم. با توجه به آن که تعداد کلاس های این برنامه زیاد اســـت (بیش از ۱۰۰ کلاس) به روش دســـتی نمی توان معماری را استخراج نمود و بایستی از روش گفته شده مراحل را پیگیری نمود.

**در نسخه دوم این پیش نویس توضیحات این بخش کامل می گردد.

۶ منابع و ماخذ

***