ISSN No. 0976-5697

Volume 3, No. 4, July- August 2012

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Component evaluation and component interface identification from object oriented System

Shivani Budhkar*
Asst. Prof. MCA Department
P.E.S. Modern College of Engineering
Pune, India
shivanibudhkar@gmail.com

Dr.Arpita Gopal
Director, MCA
Sinhgad Institute of Business Administration and Research,
Pune, India
arpita.gopal@gmail.com

Abstract: Component based technology has proven to be more reusable and suitable for new computing environments than object oriented technology. Most of the existing object- oriented systems do not have reliable software architecture as system evolves. As High level software architecture is useful in all phases of software life cycle, it is important to reengineer object oriented system and recover component based architecture. For this purpose, we have developed a process and a tool which creates components from existing object oriented system. We have defined three steps to recover component based architecture .In this paper we will demonstrate how to extract interfaces among components and component evaluation while recovering component based architecture. We have evaluated feasibility of this tool on Java software.

Keywords: Component interface, component, Component Evaluation, coupling, cohesion

I. INTRODUCTION

Object oriented development has not provided extensive reuse. Component-based software architecture is a high level abstraction of a system. It has architectural elements: Components - which provide functionality, Connectors which describe interactions and Configuration which represents the topology of connections between components. This abstraction provides major advantages in the software life cycle like better abstraction capabilities, better flexibility for evolution and maintenance, better reusability as compared to object oriented paradigm. Hence, it is important to extract component based architecture from an object oriented system. Such architectures give better understanding of legacy object oriented code as stated in [1] and identified components can be packaged, integrated into component libraries for further reuse in other new applications [2].Similarly component interfaces can be created.

Component based software Engineering is being evolved; hence some characters of component paradigm are emerged. Component is cohesive to provide good services to user. Components have well defined interfaces. Size of components should be adequate for easy deployment and maintenance. Reusable components can be used in many different systems [6].

Using dependencies in existing object oriented system like composition, inheritance, method coupling etc., we derived input for agglomerative clustering algorithm, which creates components and then interface details are generated. We have developed a tool for identifying components and interfaces among components. We have evaluated our approach on small java program. Wherein four components were identified and interface details are shown.

The rest of the paper is organized as follows: In section II we will discuss about the tool and overview process for identifying components, interface generation and component evaluation. Case study and results from our tool are provided in section III. Section IV gives related work. Section V concludes and proposes idea for future.

II. OVERVIEW OF TOOL AND PROCESS

We have identified three steps to produce a component based architectural view from an object-oriented application in our approach –

i) Identify dependencies in existing object oriented system
ii) Identify components
iii) Identify the interfaces to bind them together. Following "Fig.1" shows overall approach for producing component based architecture.

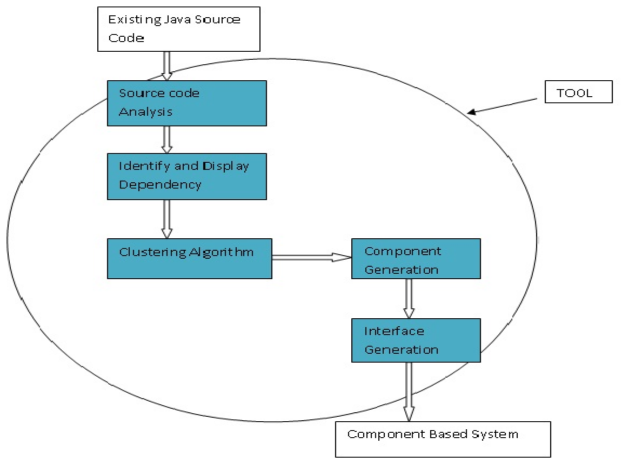


Figure1. Tool and Process

a. ***Identify dependencies in existing object oriented system:*** Our process is based on the identification of source code entities and the relationship between them. The list of possible relationships between object oriented systems includes inheritance, composition, invocation relationship etc. We have extracted inheritance coupling, composition coupling, method coupling and integrated coupling through our tool [3].

b. ***Identify components:*** - A component is group of classes collaborating to provide a function of application [4].We need to group the classes based on similarity to generate component based system from existing object oriented system. Each of the group becomes component. A hierarchical clustering algorithm allows grouping of classes of the application. We have identified components from clustering level through our tool [5].

c. ***Identify interfaces:*** - Identified group of classes working together will form components. We also need to identify interfaces to describe how they bind together.

A tool is being developed tool which will accept the user input of an existing java source code and then generates dependencies. The tool analyzes data represented through these dependencies. These are further taken as an input to Agglomerative clustering algorithm which creates components for component based system. Using these components, interface details are identified. Identified components are evaluated using component cohesion, component coupling, and component size metrics for quality of components. Here in this paper we focus on identifying interfaces and component evaluation.

A. Identify interfaces:

Component based system consists of components and interfaces. Component interfaces are the means by which components connect with each other. A component interface specifies the service that the component provides and requires. Among all of the methods in the component, only public methods used from outside provide services to other components or classes. Therefore we create a provide -interface that includes the public methods that exists in any of the component's classes and which are used by the outside of that component. Require-interface is the union set of every method in other components that is called by the component. To reduce cyclic dependency among components, we group these interfaces as packages. Our process of identifying interfaces and component evaluation is shown in below "fig. 2".

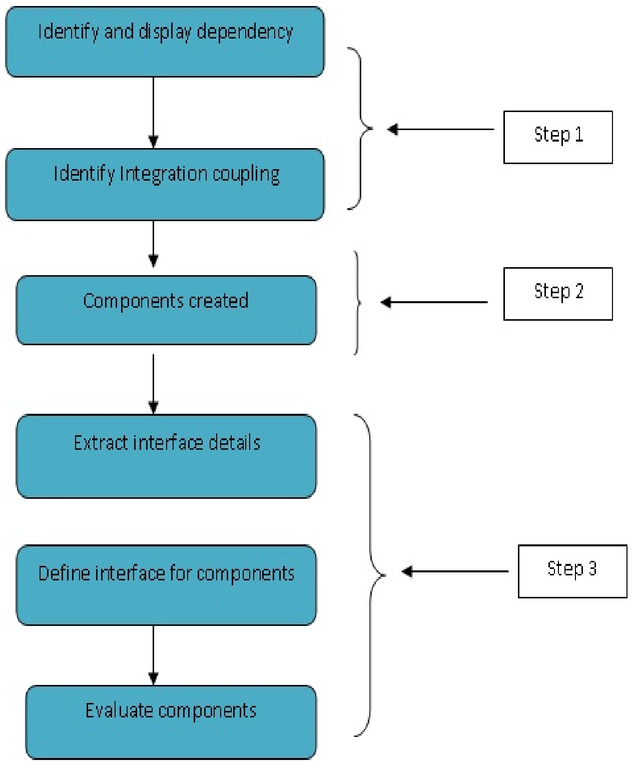

Identify and display dependency
Identify Integration coupling
Components created
Extract interface details
Define interface for components
Evaluate components
Step 1
Step 2
Step 3

Fig.ure 2: Process for identify interfaces and component evaluation

B. Component Evaluation:

The component evaluation of step 3 above accepts the results produced through clustering i.e. components created, interfaces details created (as shown in figure.3 - figure.6) as input and evaluates the quality of identified components. There are several evaluation criterions proposed to qualify clustering results. The basic quality metrics to evaluate software system are coupling and cohesion, which can cause serious impact on maintenance, evolution, and reuse. There should also be appropriate number of implementation classes in well organized components. Evaluation criteria for components used by us are size, coupling and cohesion.

a. Size:

In [7] Cui and Chae proposed size as evaluation criteria to show well organized components with appropriate number of implementation classes. So using size we evaluate clustering results. According to them sum of ratios of single class component, classes in largest component and other intermediate components should be 100%.

Ratio of Single class component=Number of Single class component/Total number of classes

Ratio of classes in largest component=Number of classes in the largest component/Total number of classes

Ratio of other intermediate components = Number of classes in intermediate components /Total number of classes

b. Coupling:

In component based system coupling shows how tightly one component is interacting with other components in the system. Coupled Component Ratio (CCR) is one of the metric for evaluating component coupling proposed by Cui and Chae[7].According to them two components are said to be coupled if there is connection between them and CCR is defined as Number of components coupled with particular component/(Total no. of components in system – 1). The CCR value of component lies between 0 and 1.Smaller the CCR value better the component is.CCR value 1 indicates that component is coupled with all other components in system.CCR value 0 indicates that component is entirely independent.

c. Cohesion:

Cohesion in component based system is how tightly classes are coupled within the component. Cohesion metric is used to measure quality of components for reusability and maintainability. We propose Component Cohesion Metric (CCM) as Number of component's self couplings/Total number of couplings of that component. Where total number of couplings of component = self coupling + coupling with other components within system. The value of CCM lies between 0 and 1. A higher CCM value indicates more similar behavior is grouped together i.e. more tightly coupled classes are grouped together.CCM value 1 indicates high cohesion within component.

III.RESULT AND CASE STUDY

We used small java software "Arithmetic24 Game" which is developed in Java by Huahai Yang [8] as a case study. It is a simulation of popular traditional card game.

We have developed tool for migration from object oriented system to component based system. Our tool accepts Java source code as input .The tool parses java code and shows inheritance coupling, method coupling and composition coupling and integrated coupling. Results showed most of the classes are placed in proper coupling tables [3] and four components are created and classes are kept in appropriate components [5]. Further "fig..3" to "fig.6" shows interface details created for these components. Using these details, interfaces among components can be created. "Table-I" show candidate components created along with respective classes for our case study "Arithmetic24 Game". Using interface details component diagram with dependencies is shown in "fig.7a". Components are evaluated for quality using metrics presented in sectionB.1 to B.3 and the results are shown in "fig.8" i.e. from "fig.3" and "Table I" largest components are component0 and component3 consisting of 8 classes each. So Ratio of classes in largest component0 =8/20 = 40% and Ratio of classes in largest component3 =8/20 = 40%. There is a single class component, component2, so Ratio of Single class component=1/20 = 5%.There is one intermediate component, component1, so Ratio of other intermediate components = 3/20 = 15%. Thus sum of these three ratios is 100%; it indicates all the classes in the software have been considered by three ratios. Also Result screen "fig.8" shows evaluation of components by coupling metric. Coupled component Ratio (CCR) for Component0 = 0.66, CCR for Component1=0.33, CCR for Component2=0.66, CCR for Component3=0.66.Agian from result screen "fig.8" shows evaluation of components by Component Cohesion Metric (CCM). CCM for Component0=0.6, CCM for Component1=0.25, CCM for Component2=0, CCM for Component3=0.25.

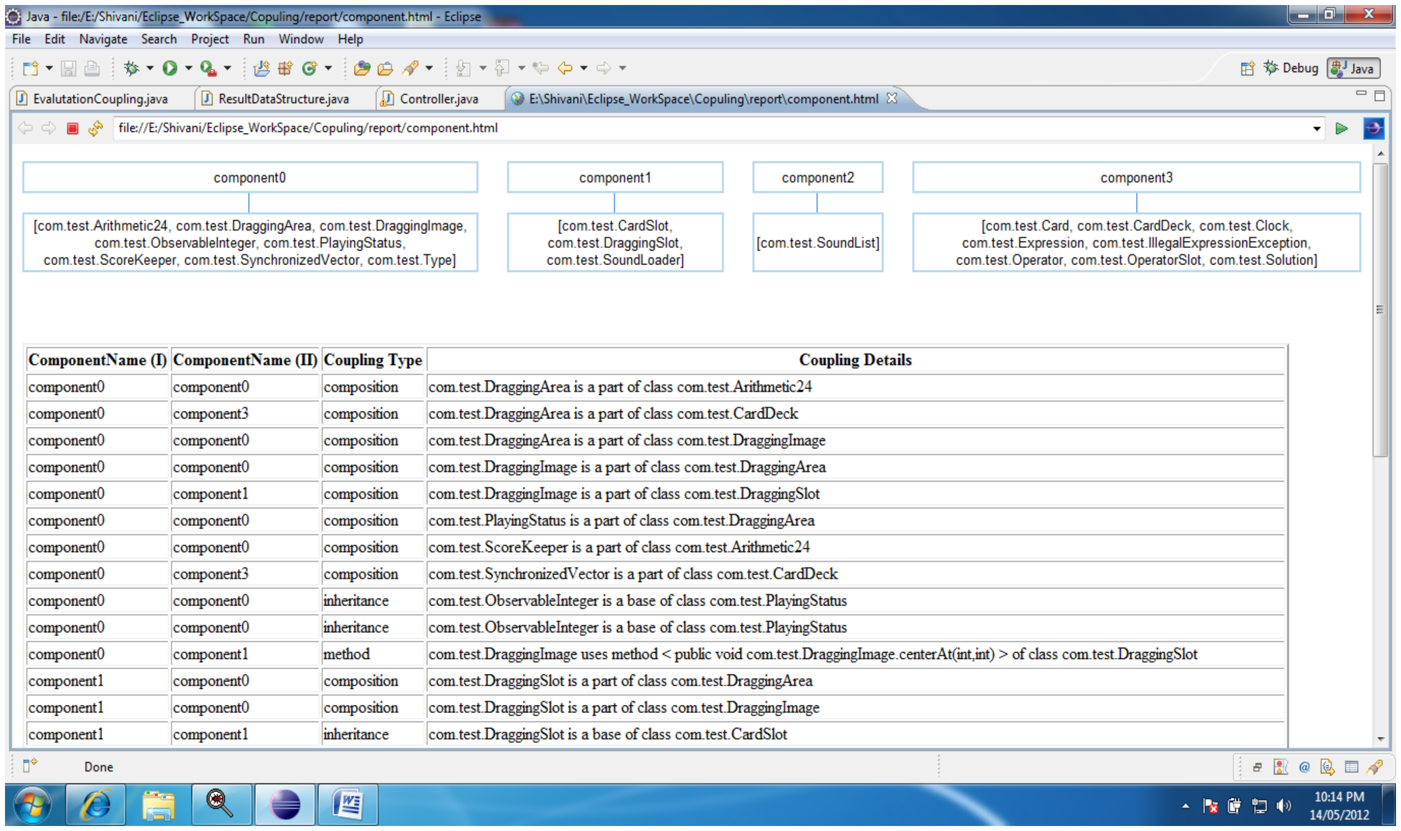
"Fig.7a" shows dependencies among components created through our tool. Component dependencies must be decreased. We decrease dependency by managing interfaces into another package. So using interface package, components with cyclic dependency can be removed, as shown in "Fig.7b".

We can create component packages and interface packages which will play role of required interface and provided interface. Deployment of components and interfaces will depend upon the framework you use.

We can conclude that our tool gives optimum results for component identification and interface generation.

Table I. Candidate component table for "Arithmetic24" game

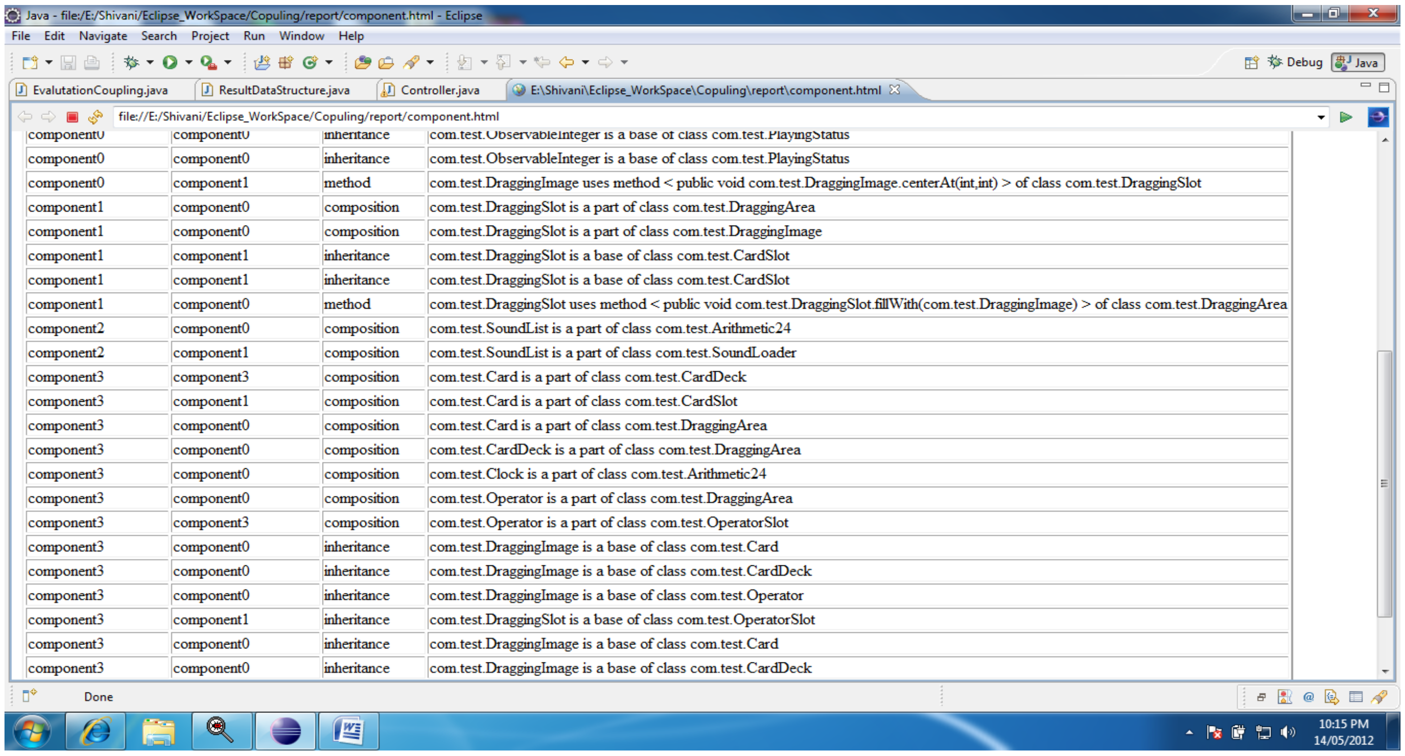
Candidate Component	**Classes**
Component0	Arithmetic24,DraggingArea,DraggingImage,ObservableInteger,PlayingStatus,ScoreKeeper,SynchronizedVector,Type
Component1	CardSlot, DraggingSlot, SoundLoader
Component2	SoundList
Component3	Card, CardDeck, Clock, Expression, IllegalExpressionException, Operator, OperatorSlot, Solution


Java - file:/E:/Shivani/Eclipse_WorkSpace/Copuling/report/component.html - Eclipse
File Edit Navigate Search Project Run Window Help
Debug Java
EvalutationCoupling.java
ResultDataStructure.java
Controller.java
E:\Shivani\Eclipse_WorkSpace\Copuling\report\component.html
file:///E:/Shivani/Eclipse_WorkSpace/Copuling/report/component.html
Done
10:14 PM
14/05/2012

component0	component1	component2	component3
[com.test.Arithmetic24, com.test.DraggingArea, com.test.DraggingImage, com.test.ObservableInteger, com.test.PlayingStatus, com.test.ScoreKeeper, com.test.SynchronizedVector, com.test.Type]	[com.test.CardSlot, com.test.DraggingSlot, com.test.SoundLoader]	[com.test.SoundList]	[com.test.Card, com.test.CardDeck, com.test.Clock, com.test.Expression, com.test.IllegalExpressionException, com.test.Operator, com.test.OperatorSlot, com.test.Solution]

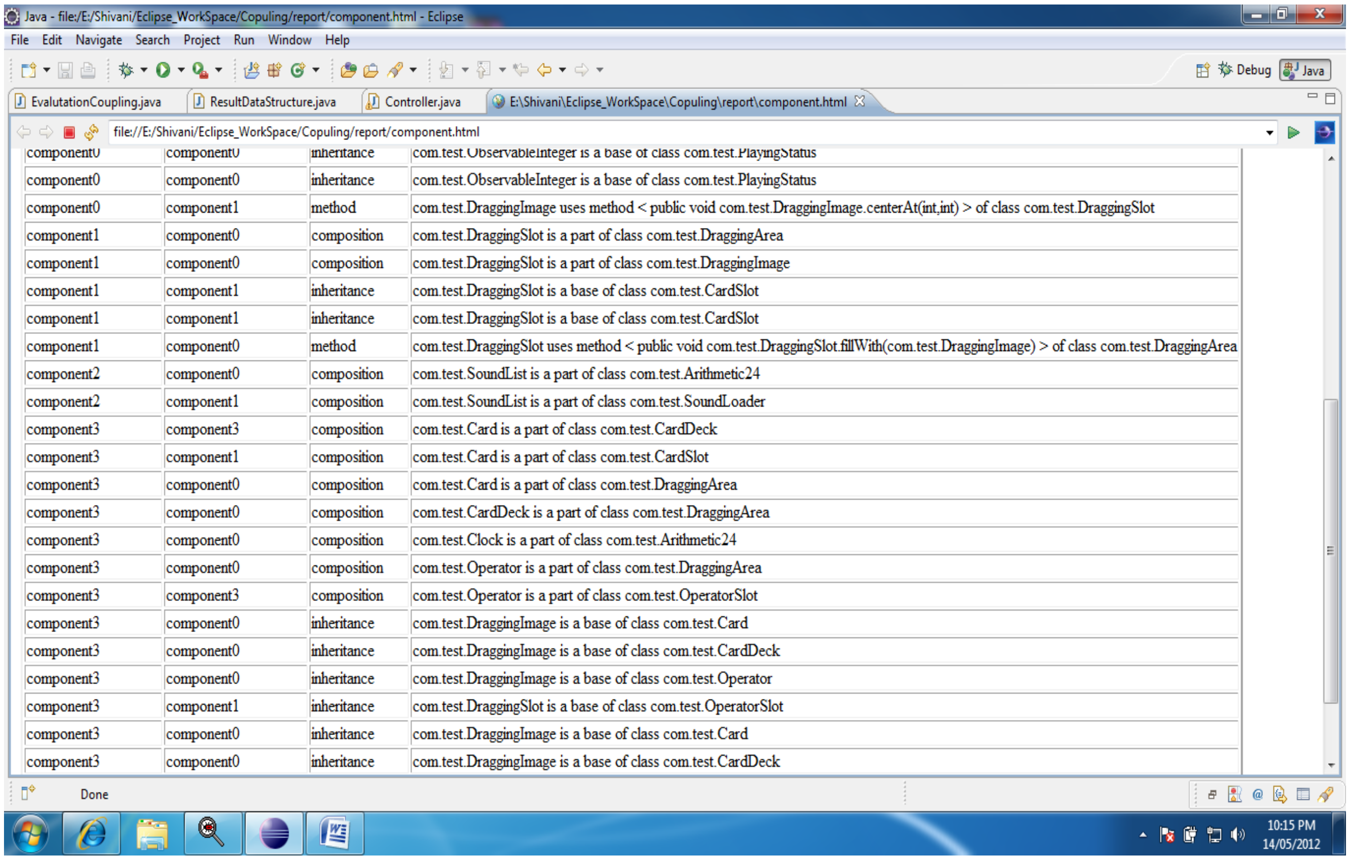
ComponentName (I)	ComponentName (II)	Coupling Type	Coupling Details
component0	component0	composition	com.test.DraggingArea is a part of class com.test.Arithmetic24
component0	component3	composition	com.test.DraggingArea is a part of class com.test.CardDeck
component0	component0	composition	com.test.DraggingArea is a part of class com.test.DraggingImage
component0	component0	composition	com.test.DraggingImage is a part of class com.test.DraggingArea
component0	component1	composition	com.test.DraggingImage is a part of class com.test.DraggingSlot
component0	component0	composition	com.test.PlayingStatus is a part of class com.test.DraggingArea
component0	component0	composition	com.test.ScoreKeeper is a part of class com.test.Arithmetic24
component0	component3	composition	com.test.SynchronizedVector is a part of class com.test.CardDeck
component0	component0	inheritance	com.test.ObservableInteger is a base of class com.test.PlayingStatus
component0	component0	inheritance	com.test.ObservableInteger is a base of class com.test.PlayingStatus
component0	component1	method	com.test.DraggingImage uses method < public void com.test.DraggingImage.centerAt(int,int) > of class com.test.DraggingSlot
component1	component0	composition	com.test.DraggingSlot is a part of class com.test.DraggingArea
component1	component0	composition	com.test.DraggingSlot is a part of class com.test.DraggingImage
component1	component1	inheritance	com.test.DraggingSlot is a base of class com.test.CardSlot

Figure 3. Components created and interface details among components


Java - file:/E:/Shivani/Eclipse_WorkSpace/Copuling/report/component.html - Eclipse
File Edit Navigate Search Project Run Window Help
Debug Java
EvalutationCoupling.java
ResultDataStructure.java
Controller.java
E:\Shivani\Eclipse_WorkSpace\Copuling\report\component.html
file:///E:/Shivani/Eclipse_WorkSpace/Copuling/report/component.html
Done
10:15 PM
14/05/2012

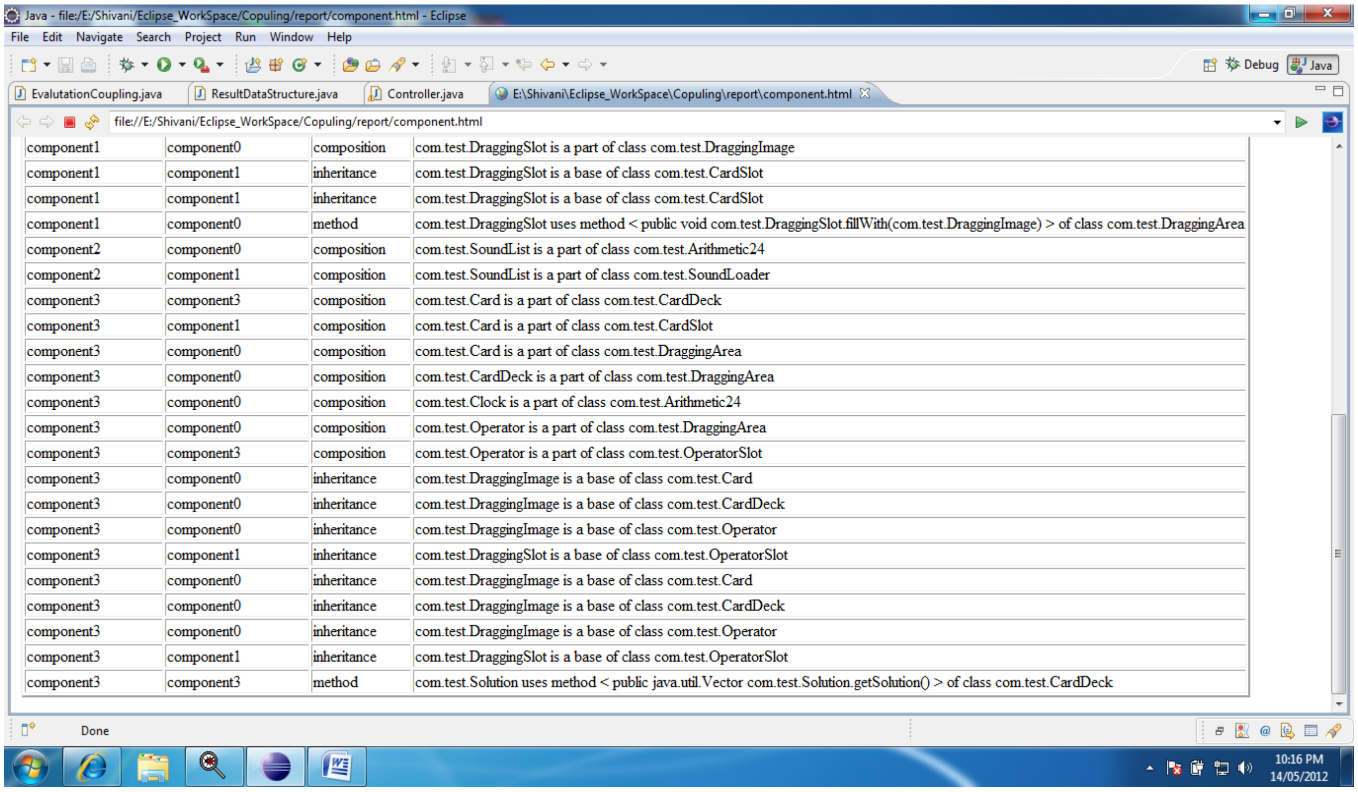
component0	component0	inheritance	com.test.ObservableInteger is a base of class com.test.PlayingStatus
component0	component0	inheritance	com.test.ObservableInteger is a base of class com.test.PlayingStatus
component0	component1	method	com.test.DraggingImage uses method < public void com.test.DraggingImage.centerAt(int,int) > of class com.test.DraggingSlot
component1	component0	composition	com.test.DraggingSlot is a part of class com.test.DraggingArea
component1	component0	composition	com.test.DraggingSlot is a part of class com.test.DraggingImage
component1	component1	inheritance	com.test.DraggingSlot is a base of class com.test.CardSlot
component1	component1	inheritance	com.test.DraggingSlot is a base of class com.test.CardSlot
component1	component0	method	com.test.DraggingSlot uses method < public void com.test.DraggingSlot.fillWith(com.test.DraggingImage) > of class com.test.DraggingArea
component2	component0	composition	com.test.SoundList is a part of class com.test.Arithmetic24
component2	component1	composition	com.test.SoundList is a part of class com.test.SoundLoader
component3	component3	composition	com.test.Card is a part of class com.test.CardDeck
component3	component1	composition	com.test.Card is a part of class com.test.CardSlot
component3	component0	composition	com.test.Card is a part of class com.test.DraggingArea
component3	component0	composition	com.test.CardDeck is a part of class com.test.DraggingArea
component3	component0	composition	com.test.Clock is a part of class com.test.Arithmetic24
component3	component0	composition	com.test.Operator is a part of class com.test.DraggingArea
component3	component3	composition	com.test.Operator is a part of class com.test.OperatorSlot
component3	component0	inheritance	com.test.DraggingImage is a base of class com.test.Card
component3	component0	inheritance	com.test.DraggingImage is a base of class com.test.CardDeck
component3	component0	inheritance	com.test.DraggingImage is a base of class com.test.Operator
component3	component1	inheritance	com.test.DraggingSlot is a base of class com.test.OperatorSlot
component3	component0	inheritance	com.test.DraggingImage is a base of class com.test.Card
component3	component0	inheritance	com.test.DraggingImage is a base of class com.test.CardDeck

Figure 4. Remaining Interface details among components -I


Java - file:/E:/Shivani/Eclipse_WorkSpace/Copuling/report/component.html - Eclipse
File Edit Navigate Search Project Run Window Help
Debug Java
EvalutationCoupling.java
ResultDataStructure.java
Controller.java
E:\Shivani\Eclipse_WorkSpace\Copuling\report\component.html
file://E:/Shivani/Eclipse_WorkSpace/Copuling/report/component.html
Done
10:15 PM
14/05/2012

component0	component0	inheritance	com.test.ObservableInteger is a base of class com.test.PlayingStatus
component0	component0	inheritance	com.test.ObservableInteger is a base of class com.test.PlayingStatus
component0	component1	method	com.test.DraggingImage uses method < public void com.test.DraggingImage.centerAt(int,int) > of class com.test.DraggingSlot
component1	component0	composition	com.test.DraggingSlot is a part of class com.test.DraggingArea
component1	component0	composition	com.test.DraggingSlot is a part of class com.test.DraggingImage
component1	component1	inheritance	com.test.DraggingSlot is a base of class com.test.CardSlot
component1	component1	inheritance	com.test.DraggingSlot is a base of class com.test.CardSlot
component1	component0	method	com.test.DraggingSlot uses method < public void com.test.DraggingSlot.fillWith(com.test.DraggingImage) > of class com.test.DraggingArea
component2	component0	composition	com.test.SoundList is a part of class com.test.Arithmetic24
component2	component1	composition	com.test.SoundList is a part of class com.test.SoundLoader
component3	component3	composition	com.test.Card is a part of class com.test.CardDeck
component3	component1	composition	com.test.Card is a part of class com.test.CardSlot
component3	component0	composition	com.test.Card is a part of class com.test.DraggingArea
component3	component0	composition	com.test.CardDeck is a part of class com.test.DraggingArea
component3	component0	composition	com.test.Clock is a part of class com.test.Arithmetic24
component3	component0	composition	com.test.Operator is a part of class com.test.DraggingArea
component3	component3	composition	com.test.Operator is a part of class com.test.OperatorSlot
component3	component0	inheritance	com.test.DraggingImage is a base of class com.test.Card
component3	component0	inheritance	com.test.DraggingImage is a base of class com.test.CardDeck
component3	component0	inheritance	com.test.DraggingImage is a base of class com.test.Operator
component3	component1	inheritance	com.test.DraggingSlot is a base of class com.test.OperatorSlot
component3	component0	inheritance	com.test.DraggingImage is a base of class com.test.Card
component3	component0	inheritance	com.test.DraggingImage is a base of class com.test.CardDeck

Figure.-5 Remaining Interface details among components-II


Java - file:/E:/Shivani/Eclipse_WorkSpace/Copuling/report/component.html - Eclipse
File Edit Navigate Search Project Run Window Help
Debug Java
EvalutationCoupling.java
ResultDataStructure.java
Controller.java
E:\Shivani\Eclipse_WorkSpace\Copuling\report\component.html
file://E:/Shivani/Eclipse_WorkSpace/Copuling/report/component.html
Done
10:16 PM
14/05/2012

component1	component0	composition	com.test.DraggingSlot is a part of class com.test.DraggingImage
component1	component1	inheritance	com.test.DraggingSlot is a base of class com.test.CardSlot
component1	component1	inheritance	com.test.DraggingSlot is a base of class com.test.CardSlot
component1	component0	method	com.test.DraggingSlot uses method < public void com.test.DraggingSlot.fillWith(com.test.DraggingImage) > of class com.test.DraggingArea
component2	component0	composition	com.test.SoundList is a part of class com.test.Arithmetic24
component2	component1	composition	com.test.SoundList is a part of class com.test.SoundLoader
component3	component3	composition	com.test.Card is a part of class com.test.CardDeck
component3	component1	composition	com.test.Card is a part of class com.test.CardSlot
component3	component0	composition	com.test.Card is a part of class com.test.DraggingArea
component3	component0	composition	com.test.CardDeck is a part of class com.test.DraggingArea
component3	component0	composition	com.test.Clock is a part of class com.test.Arithmetic24
component3	component0	composition	com.test.Operator is a part of class com.test.DraggingArea
component3	component3	composition	com.test.Operator is a part of class com.test.OperatorSlot
component3	component0	inheritance	com.test.DraggingImage is a base of class com.test.Card
component3	component0	inheritance	com.test.DraggingImage is a base of class com.test.CardDeck
component3	component0	inheritance	com.test.DraggingImage is a base of class com.test.Operator
component3	component1	inheritance	com.test.DraggingSlot is a base of class com.test.OperatorSlot
component3	component0	inheritance	com.test.DraggingImage is a base of class com.test.Card
component3	component0	inheritance	com.test.DraggingImage is a base of class com.test.CardDeck
component3	component0	inheritance	com.test.DraggingImage is a base of class com.test.Operator
component3	component1	inheritance	com.test.DraggingSlot is a base of class com.test.OperatorSlot
component3	component3	method	com.test.Solution uses method < public java.util.Vector com.test.Solution.getSolution() > of class com.test.CardDeck

Figure.- 6 Remaining interface details among components – III

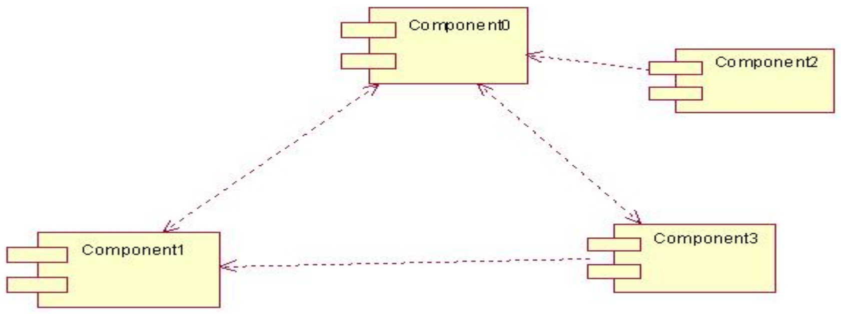


Figure.-7a UML Component Diagram for Arithmetic24 game

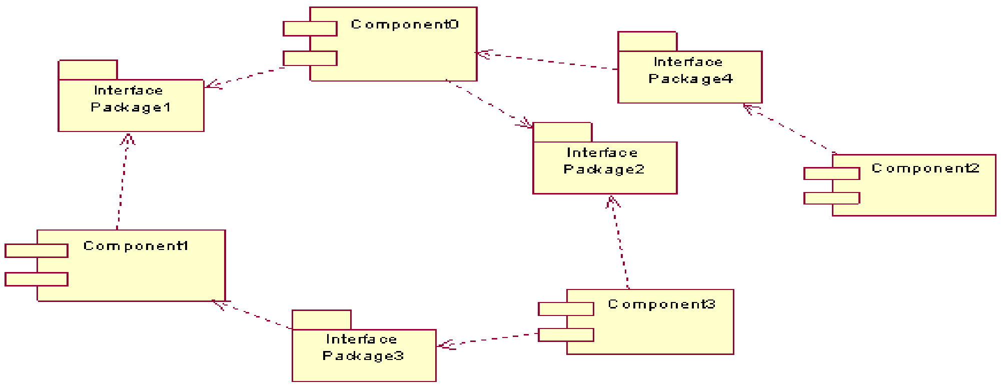
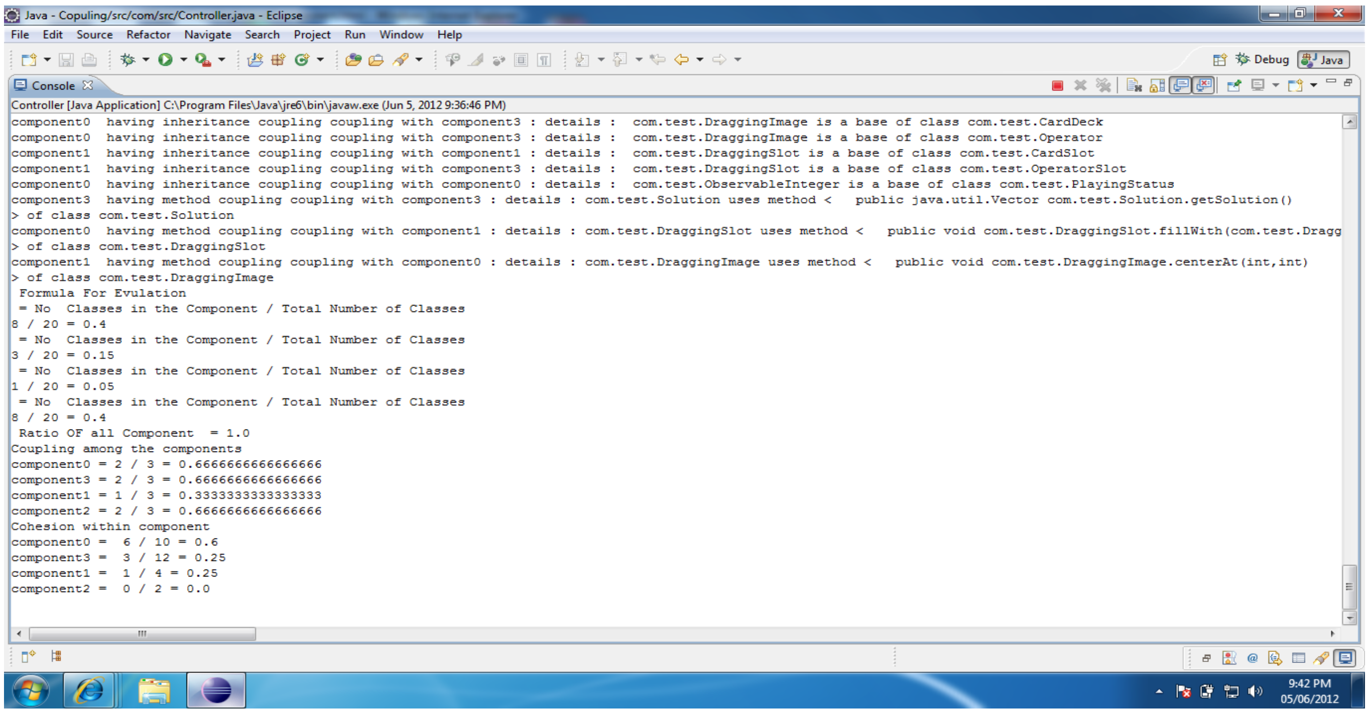


Figure.-7b UML Components with interfaces as packages for Arithmetic24 game


Java - Copuling/src/com/src/Controller.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Debug
Java
Console

```
Controller [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (Jun 5, 2012 9:36:46 PM)
component0  having inheritance coupling coupling with component3 : details :  com.test.DraggingImage is a base of class com.test.CardDeck
component0  having inheritance coupling coupling with component3 : details :  com.test.DraggingImage is a base of class com.test.Operator
component1  having inheritance coupling coupling with component1 : details :  com.test.DraggingSlot is a base of class com.test.CardSlot
component1  having inheritance coupling coupling with component3 : details :  com.test.DraggingSlot is a base of class com.test.OperatorSlot
component0  having inheritance coupling coupling with component0 : details :  com.test.ObservableInteger is a base of class com.test.PlayingStatus
component3  having method coupling coupling with component3 : details : com.test.Solution uses method <   public java.util.Vector com.test.Solution.getSolution()
> of class com.test.Solution
component0  having method coupling coupling with component1 : details : com.test.DraggingSlot uses method <   public void com.test.DraggingSlot.fillWith(com.test.Dragg
> of class com.test.DraggingSlot
component1  having method coupling coupling with component0 : details : com.test.DraggingImage uses method <   public void com.test.DraggingImage.centerAt(int,int)
> of class com.test.DraggingImage
 Formula For Evulation
 = No  Classes in the Component / Total Number of Classes
8 / 20 = 0.4
 = No  Classes in the Component / Total Number of Classes
3 / 20 = 0.15
 = No  Classes in the Component / Total Number of Classes
1 / 20 = 0.05
 = No  Classes in the Component / Total Number of Classes
8 / 20 = 0.4
 Ratio OF all Component  = 1.0
Coupling among the components
component0 = 2 / 3 = 0.6666666666666666
component3 = 2 / 3 = 0.6666666666666666
component1 = 1 / 3 = 0.3333333333333333
component2 = 2 / 3 = 0.6666666666666666
Cohesion within component
component0 =  6 / 10 = 0.6
component3 =  3 / 12 = 0.25
component1 =  1 / 4 = 0.25
component2 =  0 / 2 = 0.0
```

Figure.-8 component evaluation by using component size, component coupling and component cohesion metrics

IV. RELATED WORK

Chardigny,et al proposed ROMANTIC [9] approach which is quasi-automatic approach to extract component based software Architecture. Medvidovic [1] proposed Focus, which regroups classes and maps the extracted entities to a conceptual architecture obtained from an architectural style according to the human expertise. Jong Kook Lee et.al proposed a component identification method that considers class cohesion, class interaction coupling, and class static coupling [6]. Similar to our work Alae- Eddin et al recovered component based Architecture via relational concept analysis [10]. Using Annealing simulation algorithm and concept of lattice Eunjoo Lee et.al presented a reengineering process of migrating existing object oriented system into components that are domain specific functional units [11]. Suk Kyung Shin and Soo Dong Kim proposed techniques for transforming Object Oriented Design into Component Based Design using Object-Z specification. Also proposes set of rules for transforming Object Oriented Design to Component Based Design [12]. Simon Allier et.al developed automatic approach for migration from object oriented to component based system which uses Execution traces to extract data and uses clustering algorithm for component identification [4].

In our approach and tool automation level is higher which decreases the need of human expertise which is expensive and is not always available.

V. CONCLUSION AND FUTURE WORK

In this study, we have developed a tool which accepts object oriented java source code and migrates into component based system. The tool identifies different kind of dependencies among the classes then uses clustering algorithm to identify components. Interfaces details of the extracted components are identified by tool using which, interface packages can be defined and components are evaluated based on component quality metrics size, component coupling and component cohesion. We used this tool for "Arithmetic24" game written in Java and showed it is applicable to object oriented system. The tool has successfully extracted four components and interface details. Thus we have satisfactory results.

Our future work will focus on evaluation of larger and more complex programs by the tool to show how methodology scales to deal with real industrial scale.

VI. REFERENCES

[1] Medvidovic, N., Jakobac V, "Using software evolution to focus architectural recovery", Automated Software Eng. 13(2), 225–256 (2006)

[2] Hironori Washizaki and Yoshiaki Fukazawa, "A technique for automatic component extraction from object oriented programs by refactoring", Sci. Comput. Program,56(1-2):99-116,2005

[3] Shivani Budhkar and Dr.Arpita gopal, "Component Based Architecture recovery from object oriented system using existing dependencies", International Journal of Computational Intelligence Techniques , Volume 3, Issue 1, 2012, pp.-56-59.

[4] Simon Allier Salah Sadou,Houari Sahraoui and Regis Fleurquin, "From Object-Oriented Applications to Component–Oriented Applications via Component oriented Architecture", Ninth Working IEEE/IFIP Conference on Software Architecture,2011 pg- 214-223

[5] Shivani Budhkar and Dr. Arpita Gopal, "Component identification from existing object oriented system using Hierarchical clustering", IOSR Journal of Engineering, May. 2012, Vol. 2(5) pp: 1064-1068

[6] Jong Kook Lee, Seung Jae Jung, Soo Dong Kim, Woo Hyun jang, Dong Han Ham, " Component Identification Method with Coupling and Cohesion" ,Eighth Asia-Pacific Software Engineering Conference(APSEC'01) 1530-1362/01

[7] Jian Feng Cui,Heung Seok Chae, "Applying agglomerative hierarchical clustering algorithms to component identification for legacy systems", Information and Software technology 53(2011)601-614

[8] http://javaboutique.internet.com/arith24/

[9] Chardigny, S., Seriai, A., Oussalah, M., Tamzalit, D," Extraction of component-based architecture from object-oriented systems", In: WICSA. pp. 285–288. IEEE Computer Society(2008)

[10] Alae-Eddine El Hamdouni, A.Djamel Seriai1, and Marianne Huchard, "Component based architecture recovery From OO systems via relational Concept Analysis" ,CLA10 7th International Conference on Concept Lattices and Their Applications, Sevilla : Spain (2010)pg- 259-270

[11] Eunjoo Lee,Byungjeong Lee,Woochang Shin, Chisu,"A reverse engineering Process for Migrating from object oriented Legacy system to a component based system" ,In proceedings of 27th Annual International Computer Software and Applications Conferen*ce* 2003, 0730-3157/03

[12] Suk Kyung Shin and Soo Dong Kim," A Method to Transform Object Oriented Design into Component-Based Design using Object-Z", The third ACIS international conference on Software Engineering Research, Management and Applications (SERA'05) pg- 274 - 281