International Journal of Computational Intelligence Techniques
ISSN: 0976-0466 & E-ISSN: 0976-0474, Volume 3, Issue 1, 2012, pp.-56-59.
Available online at http://www.bioinfo.in/contents.php?id=36

COMPONENT-BASED ARCHITECTURE RECOVERY FROM OBJECT ORIENTED SYSTEMS USING EXISTING DEPENDENCIES AMONG CLASSES

SHIVANI BUDHKAR[1]* AND ARPITA GOPAL[2]

[1]Modern College of Engineering, Pune-05, India
[2]Sinhagad Institute of Business Administration and Research, Pune, India
*Corresponding Author: Email- shivanibudhkar@gmail.com

Received: February 27, 2012; Accepted: April 05, 2012

Abstract- Software Architecture modeling and representation is very important in software development process. Software Architecture provides high level view which is very useful in all phases of software life cycle. Component based software architecture is beneficial as it is useful for reusing system parts represented as components. Most of the existing object oriented systems do not have reliable software architecture and some legacy systems are designed without software architecture design phase. To deal with this problem we have proposed approach of architecture recovery which aims to extract component based architecture from existing object oriented system using existing dependencies among classes. In this paper we examine existing object oriented system to identify dependency among the classes using method coupling, inheritance coupling and composition coupling. The tool has been developed for this purpose. We evaluated the feasibility on Java software .
Keywords- Software Architecture, Architecture Recovery, Coupling, Component-based, Object-oriented, Dependency

Citation: Shivani Budhkar and Arpita Gopal (2012) Component-Based Architecture Recovery from Object Oriented Systems using Existing Dependencies among Classes. International Journal of Computational Intelligence Techniques, ISSN: 0976-0466 & E-ISSN: 0976-0474, Volume 3, Issue 1, pp.-56-59.

Introduction

Object-oriented development had not provided extensive reuse and computing infrastructures are evolving from mainframe to distributed environments, where object technology has not led to massive development of distributed systems. However, component-based technology is considered to be more suited for distributed system development due to its granularity and reusability. Component-based software architecture is a high level abstraction of a system using the architectural elements: components which describe functional computing, connectors which describe interactions and configuration which represents the topology of connections between components.

While recovering software Architecture from object oriented system different abstraction levels can be considered e.g. method level, variable level, object level and system level. Extensive literature research has justified these abstraction levels for software measures.[1] The definition of metrics on Object Oriented system elements are obtained by identification of different types of relationship between different classes and computation of their strengths[2]. Class coupling is one of the Object Oriented metric. Coupling is an indication of the connections between elements of the object oriented Design [3] and indicates dependencies among classes. It is important to identify coupling for creating components.

We need to determine precisely the dependency among classes and how to measure their strengths. The possible dependencies among Object Oriented system entities include inheritance, composition, aggregation and method invocations. So identifying these dependencies become the first step to recover software Architecture.

Well defined components designs are driven by a variety of factors e.g. the principles of cohesion and coupling are important factors for well-defined component design [4].Therefore in this study we are focusing on class coupling to identify well defined components.

Our approach of extracting component based architecture from

object oriented system consists of identifying dependencies among the classes then use Agglomerative hierarchical clustering algorithm to create components in component based system. The main advantage of this approach is automation level which decreases the need of human expertise which is expensive and is not always available.

The rest of the paper contains reviews research on component based software architecture recovery from object oriented system, overall process and tool, identifying dependencies from existing object oriented software for software architecture recovery, case study and results, as well as Conclusion and future work.

Related Work

Several authors through their work have proposed the extraction of architecture from an Object Oriented system [3]. Medvidovic, [5] proposes Focus, a guideline to a hybrid process which regroups classes and maps the extracted entities to a conceptual architecture obtained from an architectural style according to the human expertise. Chardigny,et al proposed ROMANTIC [6] approach which is quasi-automatic.Similar to our work Alae- Eddin et al recovered component based Architecture Via relational concept analysis[2]. 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.[7]Jong KObject Oriented k Lee et.al proposed a component identification method that considers class cohesion, class interaction coupling ,class static coupling.[4] Hassan Mathkour et al. demonstrates the generation of component based system from object oriented Design that has been achieved by developing a system recovery tool.[8] 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[9]. 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 [10.] Ghulam Rasool et al. conducted case study on six different types of software systems having source code in different programming languages using the architectural recovery framework [11].

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 provided and required interfaces to bind them together. Fig (1) shows overall approach for producing component based architecture.

i. **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.

ii. **Identify components-** A component is group of classes collaborating to provide a function of application [9].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 clustering algorithm allows grouping of classes of the application.

iii. **Identify the provided and required interfaces-** Identified group of classes working together will form components. We also need to identify required and provided interfaces to describe how they bind together.

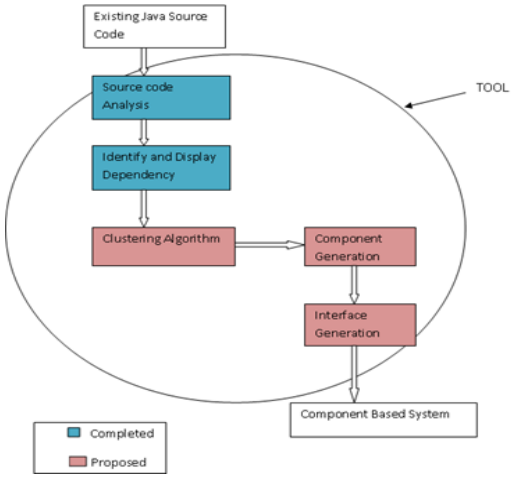


Fig. 1- Tool and Process overview

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. Similarity distance function is defined and threshold is decided.

Identifying dependencies for Architecture recovery

Coupling is qualitative measure of the degree to which classes are connected to one another. Coupling is an indication of the connections between elements of the object oriented Design [3]. It has been defined as a measure of the degree of interdependence between modules and the degree of interaction between modules. Other definition is "Coupling is a measure of the association, whether by inheritance or otherwise, between classes in a software product." [12]Though coupling is a notion from structured design, it is still applicable to object-oriented design at the levels of modules, classes and objects. We are concerned only with coupling between classes. Thus coupling indicates dependencies among classes. The possible dependencies between classes of object oriented system include inheritance, composition, method invocations etc. Here we are considering following important coupling dependencies as they are basis for identifying components from object oriented system.

1. **Inheritance coupling-** Inheritance coupling is coupling between generalized class (Super class) and its specialized classes (Sub classes) [13].
2. **Composition coupling-**When instance of one class is referred in another class, then we have composition coupling.
3. **Method coupling -** When methods of one class use methods of another class hierarchy, then we have method coupling between the classes.
4. **Integrated coupling**- It is a class's all three couplings inher-

itance coupling, composition coupling, method coupling.

Our approach of extracting component based architecture from object oriented system is based on the identification of source code entities and the relation between them. The entities and relations have to be extracted by source code analysis and identify dependencies between the classes. Fig (2) summarizes the process for identifying and displaying dependencies.

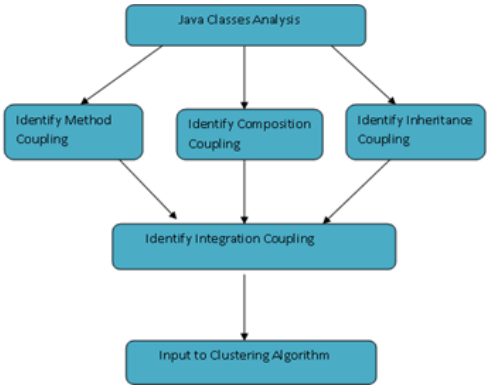


Fig. 2- Process for identify dependencies

Case study and Result

Our tool identifies inheritance coupling, method coupling and composition coupling from java source code. As a case study we used small java software 'Arithmetic24 Game', which is developed in Java by Huahai Yang [14]. It is a simulation of popular traditional card game.

Fig (3) and fig (4) shows all the classes extracted classes and different coupling tables as follows. Result shows most of the classes are placed in proper coupling tables. We have compared the result with the class diagram generated with the tool Enterprise Architecture [15].

Conclusion and future work.

An approach to recover component based architecture from object oriented system has been proposed in this paper. Relationships required to create components are extracted using tool we have developed. The tool is being further developed to automate the proposed approach. Initial experimental results from a case study were encouraging. The tool successfully extracts the classes and identifies coupling dependencies.

As a future work we are developing Hierarchical clustering algorithm to create components using identified dependencies.

Finally, further evaluation on larger and more complex programs is needed to assess how methodology scales to deal with real industrial scale.

References

[1] Frank Simon, Silvio Loffler, Claus Lewerentz. (1999) *AI Accepted for FESMA*, Amsterdam 4.

[2] Alae-Eddine El Hamdouni, Djamel Seriai A. and Marianne Huchard (2010) *7th International Conference on Concept Lattices and Their Applications, Seville: Spain,* 259-270.

[3] Roger S. Pressman, *Software Engineering A practitioner's Approach Sixth Edition*, McGraw Hill Publications.

[4] Jong kook Lee, Seung Jae Jung, Soo Dong Kim, Woo Hyun Jang, Dong Han Ham (2001) *Eight Asia pacific Software Engineering Conference(APSEC)* 1530-1362/01.

[5] Medvidovic N., Jakobac V. (2006) *Automated Software Eng.* 13(2), 225-256.

[6] Chardigny S., Seriai A., Oussalah M., Tamzalit D. (2008) *In: WICSA. IEEE Computer Society*, 285-288.

[7] Eunjoo Lee, Byungjeong Lee, Woochang Shin Chisu (2003) *27th Annual International Computer Software and Applications Conference*, 0730-3157/03.

[8] Hassan Mathkour, Ameur Touir, Hind Hakami, Ghazy Assassa (2008) *IEEE international conference on Signal Image technology and Internet Based systems,* 11-15.

[9] Simon Allier Salah Sadou, Houari Sahraoui and Regis Fleurquin (2011) *Ninth Working IEEE/IFIP Conference on Software Architecture,* 214-223.

[10] Suk Kyung Shin and Soo Dong Kim (2005) *The third ACIS international conference on Software Engineering Research, Management and Applications (SERA)* 274-281.

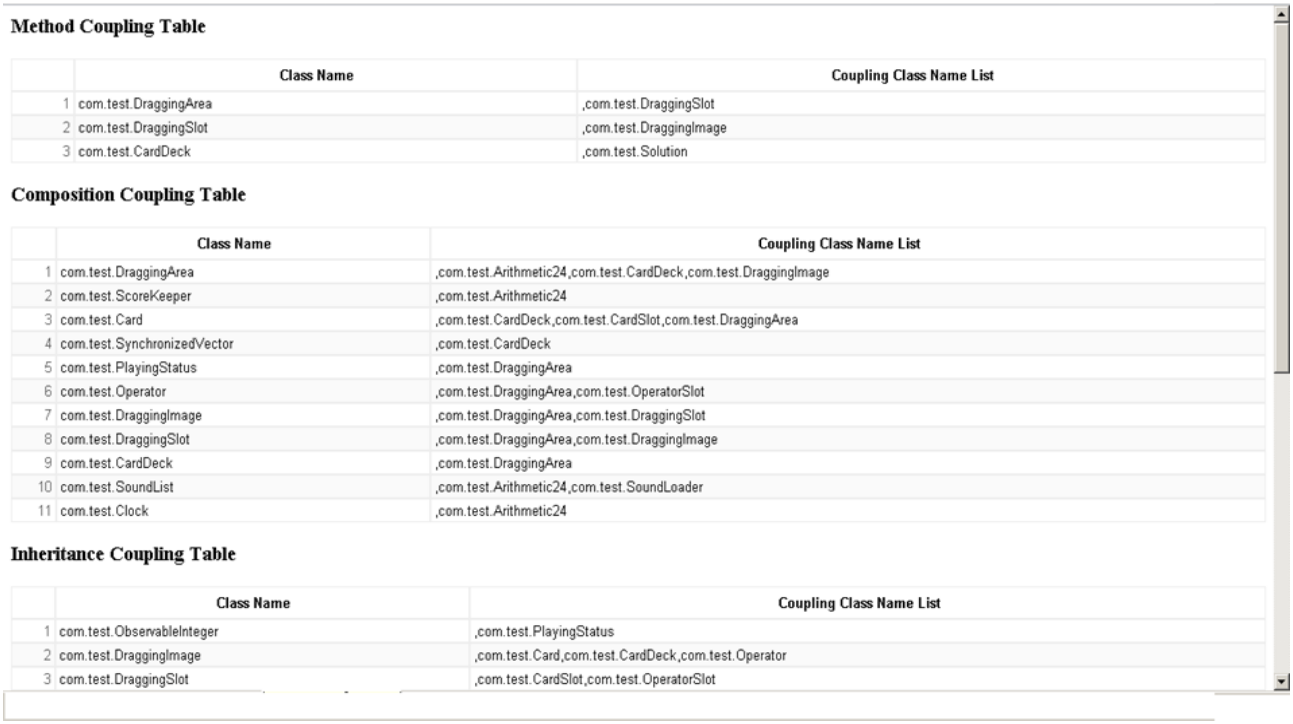
[11] Ghulam Rasool and Naddim Asif (2007) *International Journal of Computer, Information and Systems, Science and Engineering Summer,* 99-104.

[12] Chandrashekar Rajaraman, Michael R. Lyu, *Some Coupling Measures for C++ Programs*.

[13] Coad P., Yourdan E. (1991) *Object oriented Design, Princehall, Englewood cliffs, NJ.*

[14] *http://javaboutique.internet.com/arith24/*.

[15] Shivani Budhkar, Arpita Gopal (2011) *International Journal of Computer Applications,* 29(6), 36-43.



Method Coupling Table

	Class Name	Coupling Class Name List
1	com.test.DraggingArea	,com.test.DraggingSlot
2	com.test.DraggingSlot	,com.test.DraggingImage
3	com.test.CardDeck	,com.test.Solution

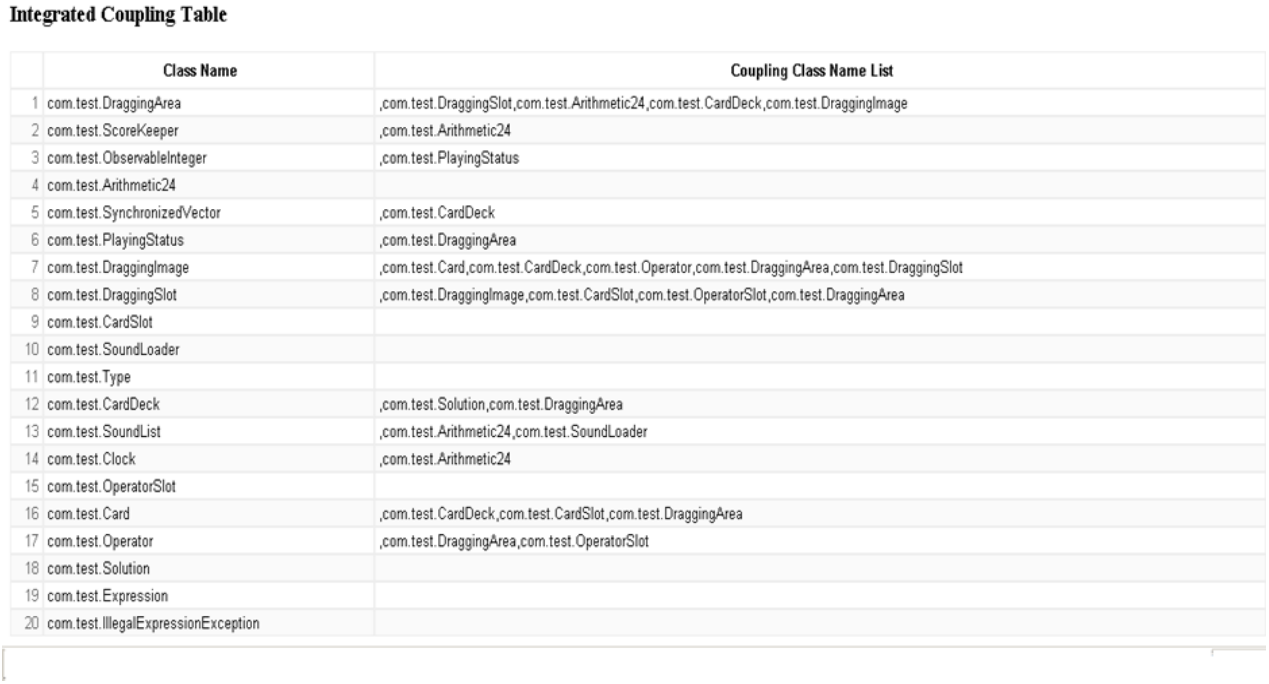
Composition Coupling Table

	Class Name	Coupling Class Name List
1	com.test.DraggingArea	,com.test.Arithmetic24,com.test.CardDeck,com.test.DraggingImage
2	com.test.ScoreKeeper	,com.test.Arithmetic24
3	com.test.Card	,com.test.CardDeck,com.test.CardSlot,com.test.DraggingArea
4	com.test.SynchronizedVector	,com.test.CardDeck
5	com.test.PlayingStatus	,com.test.DraggingArea
6	com.test.Operator	,com.test.DraggingArea,com.test.OperatorSlot
7	com.test.DraggingImage	,com.test.DraggingArea,com.test.DraggingSlot
8	com.test.DraggingSlot	,com.test.DraggingArea,com.test.DraggingImage
9	com.test.CardDeck	,com.test.DraggingArea
10	com.test.SoundList	,com.test.Arithmetic24,com.test.SoundLoader
11	com.test.Clock	,com.test.Arithmetic24

Inheritance Coupling Table

	Class Name	Coupling Class Name List
1	com.test.ObservableInteger	,com.test.PlayingStatus
2	com.test.DraggingImage	,com.test.Card,com.test.CardDeck,com.test.Operator
3	com.test.DraggingSlot	,com.test.CardSlot,com.test.OperatorSlot

Fig. 3- Method, Composition, Inheritance Dependency identified from our tool of Arithmetic24 Game software



Integrated Coupling Table

	Class Name	Coupling Class Name List
1	com.test.DraggingArea	,com.test.DraggingSlot,com.test.Arithmetic24,com.test.CardDeck,com.test.DraggingImage
2	com.test.ScoreKeeper	,com.test.Arithmetic24
3	com.test.ObservableInteger	,com.test.PlayingStatus
4	com.test.Arithmetic24	
5	com.test.SynchronizedVector	,com.test.CardDeck
6	com.test.PlayingStatus	,com.test.DraggingArea
7	com.test.DraggingImage	,com.test.Card,com.test.CardDeck,com.test.Operator,com.test.DraggingArea,com.test.DraggingSlot
8	com.test.DraggingSlot	,com.test.DraggingImage,com.test.CardSlot,com.test.OperatorSlot,com.test.DraggingArea
9	com.test.CardSlot	
10	com.test.SoundLoader	
11	com.test.Type	
12	com.test.CardDeck	,com.test.Solution,com.test.DraggingArea
13	com.test.SoundList	,com.test.Arithmetic24,com.test.SoundLoader
14	com.test.Clock	,com.test.Arithmetic24
15	com.test.OperatorSlot	
16	com.test.Card	,com.test.CardDeck,com.test.CardSlot,com.test.DraggingArea
17	com.test.Operator	,com.test.DraggingArea,com.test.OperatorSlot
18	com.test.Solution	
19	com.test.Expression	
20	com.test.IllegalExpressionException	

Fig. 4- Integrated coupling identified from our tool of Arithmetic24 Game software