Reverse Engineering Java Code to Class Diagram: An Experience Report

Shivani Budhkar
Assistant Professor
Modern College of Engineering, Pune
Maharashtra, India

Dr. Arpita Gopal
Professor & Director
SIBAR, Pune
Maharashtra, India

ABSTRACT
Software Engineering research and industry recognize the need for practical tools to support reverse engineering activities. Most of the well-known CASE-tools nowadays support reverse engineering in some way or other. Reverse engineering is first step towards software Architecture recovery. The most commonly used standard today is Unified Modeling Language to depict the architecture and design of an application. An UML class diagram describes the architecture of object oriented programs. Class diagram captures the essence of its design. Most of the existing systems do not have reliable software architecture and some legacy systems are designed without software architecture design phase. By using reverse engineering tools we can generate class diagram as part of software architecture recovery.

In this paper we assess capabilities of software reverse engineering tools to generate class diagram from java source code.

General Terms
Software Architecture, CASE tools, UML, class diagram

Keywords
Reverse Engineering tools, EA, Reverse, Rose, NOC, NOA, NGR

1. INTRODUCTION
In the software engineering domain, UML [10] has grown to a widely known and readily used graphical standard for representing various aspects of an object oriented software system. Its class diagram has proven to be particularly useful. They are the kind of specification that is to be reverse engineered in this work, with goal of architecture recovery.

Reverse engineering is first step towards software Architecture recovery. Software reverse engineering tools help in software architecture recovery.

An UML class diagram describes the architecture of object oriented programs. Most of the object oriented systems do not have reliable system's architecture or designed without software architecture design phase. If such systems needs to be migrated into component based system, reverse engineering is the first step. Most of the CASE tools help in reverse engineering with most of the languages. For migrating into component based system, class diagram need to be recovered from the object oriented code. It shows classes and relationships between them. It is needed for creating components.

In this paper, we exercised four different reverse engineering tools to generate class diagram from given source codes. Our aim was to assess capabilities of reverse engineering tools to generate class diagram from java source code, as it will help in reconstructing the software architecture from existing implemented software. The idea behind choosing these tools was assessing different kinds of tools like commercial, non commercial, open source tools that support reverse engineering of java code. We have chosen tools Rose [6], ArgoUML[1], Reverse[5], Enterprise Architecture[4]

We have examined these tools on simple java based programs. One of the experimented source code is 'Arithmetic24 Game', which is developed in Java by Huahai Yang [3]. It is a simulation of popular traditional card game.

2. RELATED WORK
There is range of existing tools that can contribute to reverse engineering of java code. Some of the researchers have developed their own tools to generate class diagrams. Various empirical studies on comparisons of reverse engineering and information extracting tools have been presented [[7]. [2]

Martin Keschenau [9] compared his own developed tool class2Uml with non- commercial tools like ArgoUML, Fujaba, Java2Uml, Idea etc. He identified query methods, association cardinalities and tried to recover more than just classes, methods.

Yann Gael [12] compared his tool PTIDEJ with WOMBLE proposed by Jackson and Waingold a tool for lightweight extraction of object models i.e. class diagram, CHAVA proposed by Korn et al., a reverse engineering tool dedicated to Java applets, which reproduces exactly a program static model [7].

Ralf Kollmann [11] examined Rose, Idea , Together and Fujaba tools from industry and research and found that despite of the constantly ongoing development, the advanced reverse engineering strategies have not been considered in these tools. Industrial and open source CASE tools such as AgroUML, Rational ROSE offer reverse engineering capabilities, but their capabilities are very limited. They do not clearly define binary class relationships. They distinguish graphically the association, aggregation, and composition relationships, but their reverse engineering algorithms produce erroneous or inconsistent relationships[13] Also all the relationships are not properly extracted. These tools take a very concrete approach.

3. A CASE STUDY

3.1 Examined Tools

Four tools were selected in this study as they support java reverse engineering. These were –

Rational rose - Rational Rose [6] is a widely used commercial UML modeling tool. Rational Rose offer reverse engineering capabilities, but their capabilities are very limited. Rational Rose supports reverse engineering of, Java software systems. When reverse engineering a Java program, Rose constructs a tree view that contains classes, interfaces, and association found at the highest level. Methods, variables etc. are nested under the owner classes. Rose also constructs (on demand) a class diagram representation of the extracted information and generates a default layout for it. Additionally, Rose automatically constructs a package hierarchy as a tree view. Rose is able to reverse engineer the information from the source code (.java files), byte code (.class files), jar files, or packed zip files. In Rose, the Java reverse engineering module can be given instructions on files, directories, packages, and libraries to be examined.

ArgoUML - ArgoUML[1] is a widely used open source tool for UML modeling tool. ArgoUML provides a modular reverse engineering framework. Currently Java source code is provided by default and there are modules for Java Jar and class file import. Similar to Rose ArgoUML constructs a tree view that contains classes, interfaces, and association found at the highest level. Methods, variables etc. are nested under the owner classes. Using Drag and drop facility user can create class diagram. Reverse engineering capability of the tool is very limited as it cannot extract association and interface.

Reverse - Reverse is non commercial tool to convert java code to class diagram developed by Neil Johan. [5] .User needs to select main java file and tool automatically displays class diagram. Tool has extracted limited classes, but no interfaces. Hence, realization relationships have not been extracted. It was successful in identifying most of the associations.

Enterprise Architecture (EA) - Enterprise Architecture (EA) [4] is widely used commercial UML modeling tool. Tool generates tree view of classes and methods. Variables are nested under methods. EA's current reverse engineering capabilities can only reverse engineer UML semantics such as class diagrams and associations.

3.2 Examined Model Properties

Number of Classes (NOC) -This is a general measure for the overall size of a software module. Therefore, high NOC values may indicate a more detailed representation.

Number of Associations (NOA) - NOA is a metric measure of interconnectedness in a module. In reverse engineering it is important to understand how classes are connected.

Number of Generalization relationship (NGR) – It models "is a" and "is like" relationships, enabling you to reuse existing data and code easily. It is a generalization / specialization relationship between classes, which helps to measure how tightly coupled classes are. From reverse engineering point of view it will help for concluding component structure.

Handling of Interfaces - An interface is a specifier for the externally-visible operations of a class, component, or other classifier (including subsystems) without specification of internal structure [10]. In UML diagrams, interfaces are drawn as classifier rectangles (with a stereotype << Interface >>) or as circles. The interfaces are attached by a dashed generalization arrow to classifiers that support it, known as realization relationship [10]. This indicates that the class provides (implements) all of the operations of the interface. The circle notation is used when the operations of the interface are hidden. A class that uses or requires the operations supplied by the interface may be attached to the circle by a dashed arrow pointing to the circle. From the reverse engineering point of view, generation of such dependencies is important for understanding the usage of interfaces and for concluding component structures and dependencies (e.g., to abstract class diagrams to a component diagram). Furthermore, different ways of handling interfaces have impact on the NOC
metric and possibly on the readability of the respective class diagram[11].

Role Names - The function of role names at association ends is comparable to that of attribute names in the sense of giving to an association between classes a meaningful descriptor, which depends on the end it's attached to. Therefore in reverse engineering, role names can hold relevant additional information about the system infrastructure. We examine, whether role names are used and if, what kind of information they represent. [11]

4. RESULTS FROM THE TOOLS

The elements found by the CASE tools are summarized in table 1 and shown in Figure 1 - 4.

Classes - Rose, ArgoUML, and Enterprise Architecture were able to find 19 classes. when applied to the "*.java" files of Arithmetic24.Tool reverse was able to find out only 17 classes as it accepts only main java file for reverse engineering. The name compartment of the class reverse engineered by all the four tools contains the name of the actual class. Both the attributes and operations compartments contain the names, types and visibility (public, private or protected) for Rose, EA, and Reverse. But ArgoUML could not identify visibility.

Associations - Total number of associations found by Rose and EA was 12.ArgoUML could not find any association. Reverse found 18 associations and showed mutually dependent classes with red dashed line. Associations are directed in all cases except for ArgoUML. The roll is named by the variable itself. Rose and EA could produce roll names. For tool 'Reverse', associations are directional but do not specify any roles.

None of the tool can specify multiplicities.

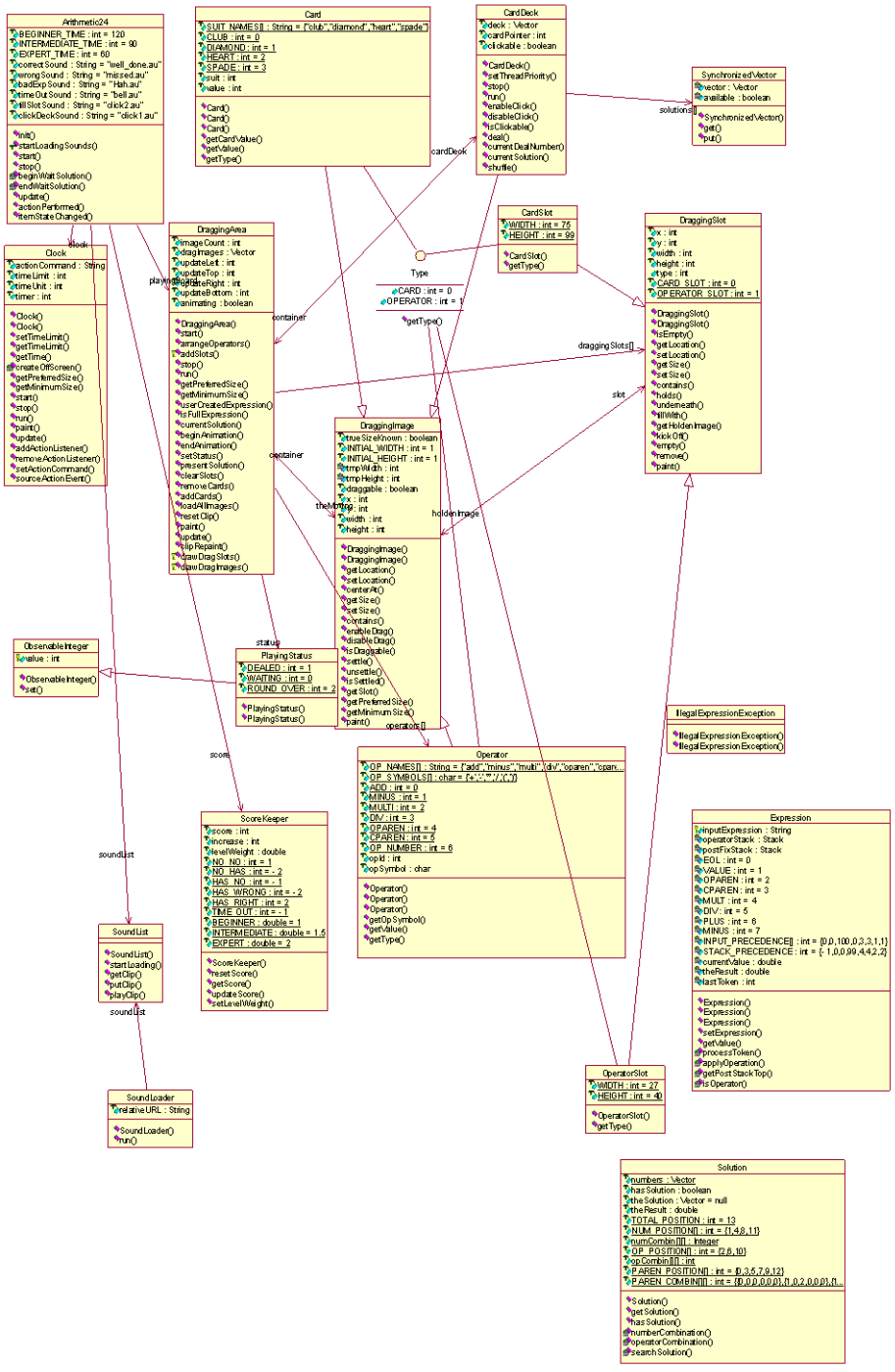

Figure 1: **Class Diagram of Arithmetic24 game from Rose**

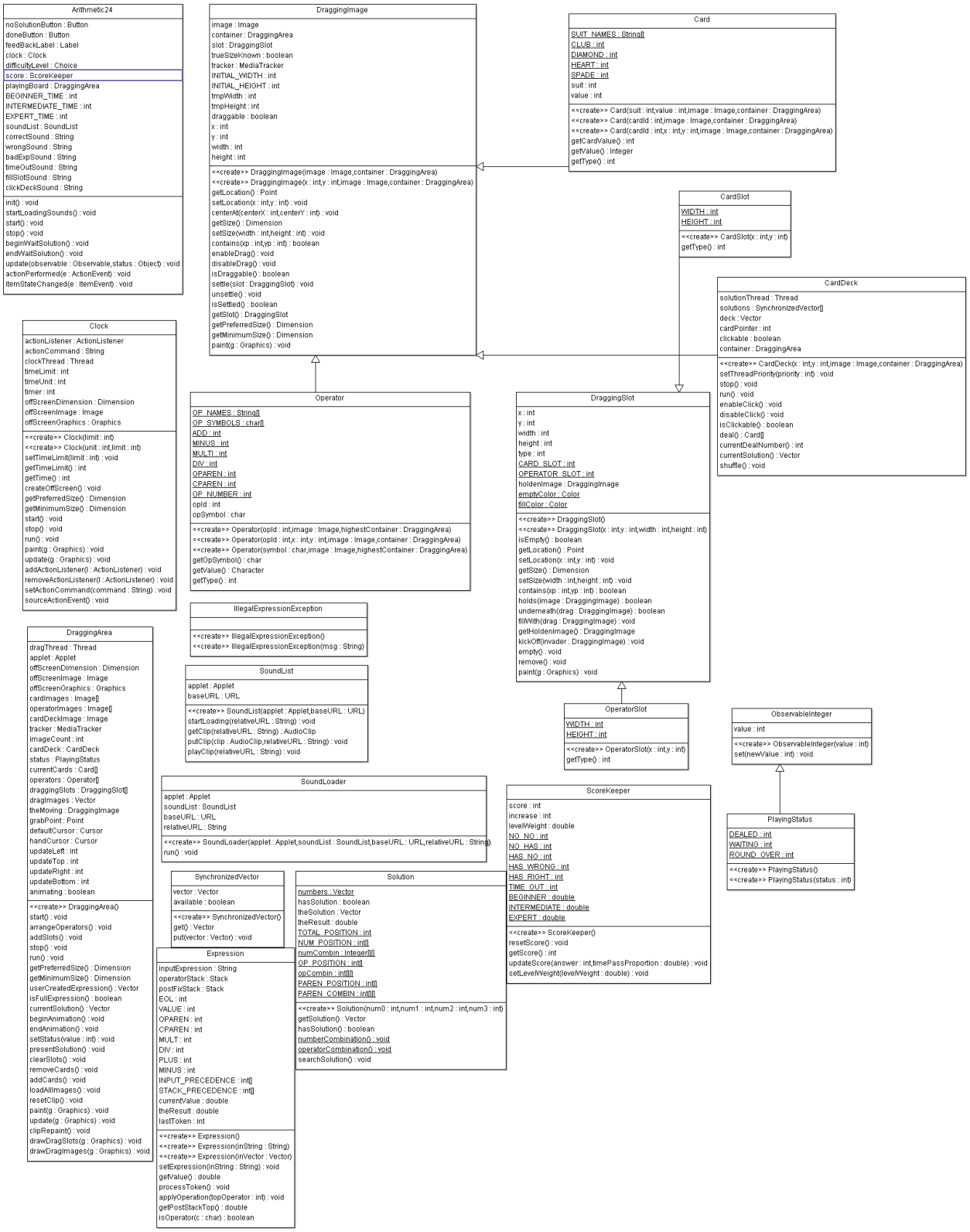

Figure 2: **Class Diagram of Arithmetic24 game from ArgoUML**

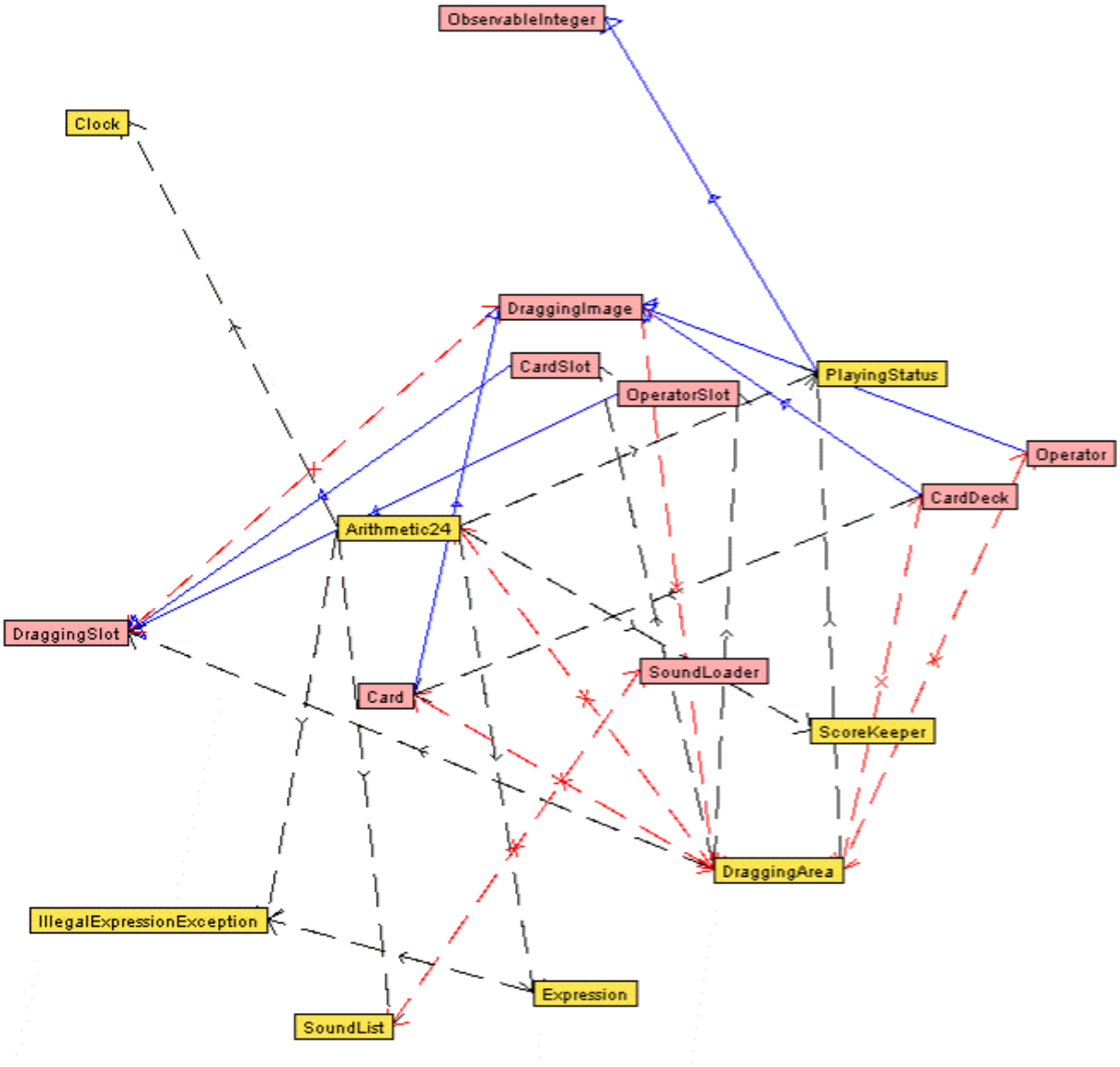

Figure 3: Class Diagram of Arithmetic24 game from Reverse

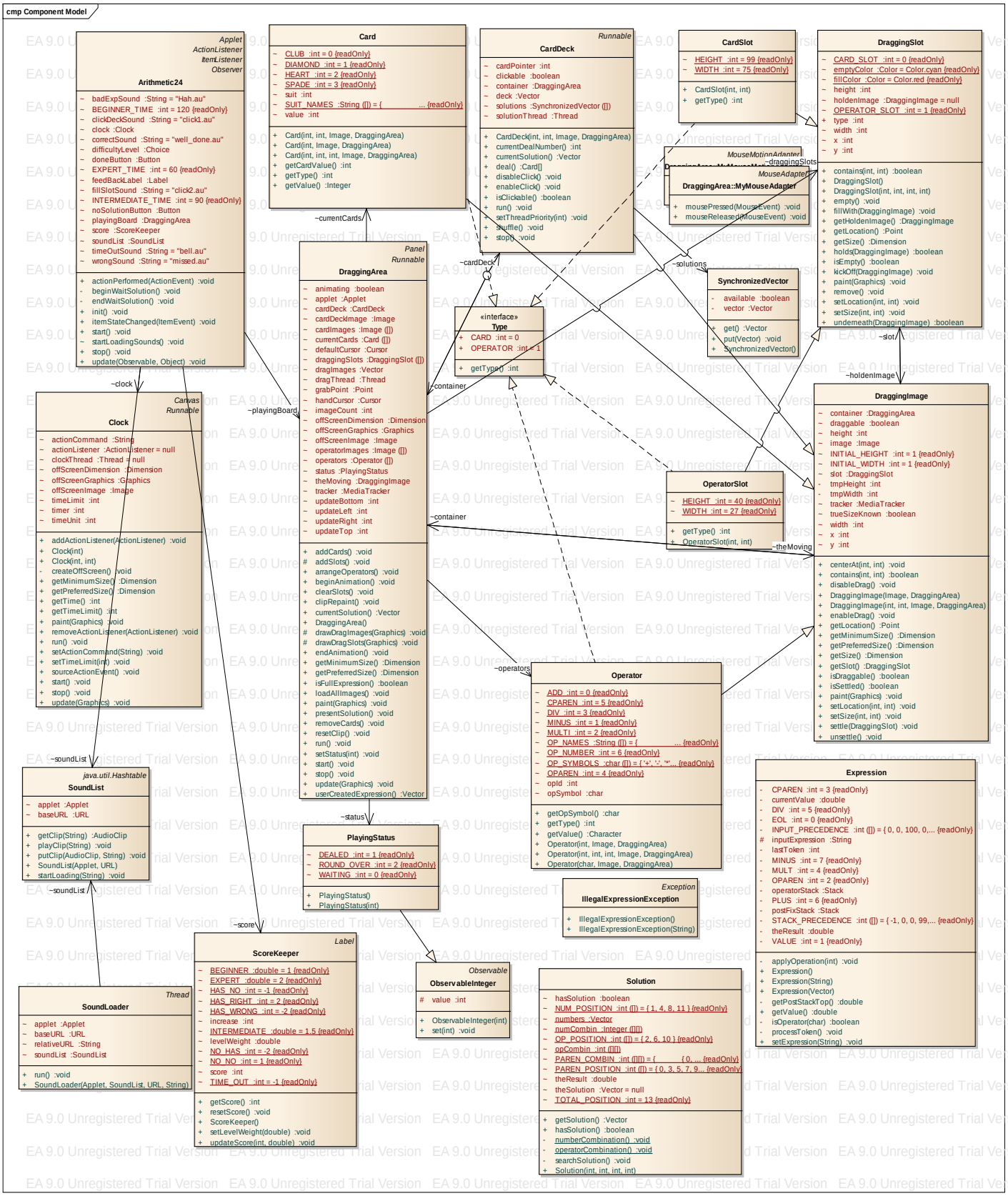

Figure 4: Class Diagram of Arithmetic24 game from EA

Table 1: Elements found by CASE Tools

Tools	No. of Classes	No. of Association	No. of Roll names	No. of Interfaces	No. of Generalization	No. of Realizations
Rational rose	19	12	14	**1**	**6**	**4**
ArgoUML	19	0	0	**0**	**6**	**0**
Reverse	17	18	0	**0**	**6**	**0**
EA	19	12	15	**1**	**6**	**4**

Generalization - All the four tools were able to recognize generalization relationships. All of them found total 6 such relationships

Handling of Interfaces - Rose uses a circle to illustrate interfaces in the class diagram. The (abstract) methods of the interfaces are written below the circle, separated with two horizontal lines, which is not the style recommended in UML. EA illustrates them using class rectangles with a <<Interface>> stereotype shown above the interface name. This notation is also available as an option in Rose.

Both Rose and EA found one interface in the Arithmetic24 core package. Both connect the interfaces to the classes that support them, that is, the classes that implement the abstract methods defined in the interfaces. Rose does that with a solid line (a realization relationship). EA uses a dashed line with a triangle at the end pointing to the interface (similar to the inheritance notation).

However, neither Rose nor EA were able to generate any dependencies between interfaces and the classes that use them (typically shown with a dashed arrow from a class pointing to the interface). This is an obvious limitation to understanding the roles of the interfaces. Further, interface dependencies are needed for abstracting a class diagram into a component diagram, understanding the interaction among different components, etc. Tools 'Reverse' and ArgoUML could not able to extract interface.

5. SUMMARY AND CONCLUSION

In the present study we have assessed capabilities of four software reverse engineering tools that generate class diagram from java source code. We have found that, even, if most of the classes are identified with simpler relationships, tools have limited capabilities for identifying indirect or complex relationships.

The motivation for this paper is to find out to what extent tools support reverse engineering. In the present study, four tools have been compared with regards to their reverse engineering capabilities. We have carried out manual comparisons. The manual comparison is needed to understand the interpretations and mappings used to generate a class diagram.

Examination of the features of some well-known tools ([1], [4], [5], [6]) has shown that it is difficult to reverse engineer class diagrams elements beyond the class boxes themselves or simpler relationships. Additionally, not all the classes, interfaces, multiplicity are properly recovered from the code. We have experimented above tools on simple java based programs .We have observed that, for the same source code different tools gives different results. The same was found in case of 'Arithmetic24 Game'. [3]
We observed that most of the classes are extracted by all the four tools but all the relationships have not been extracted properly. The simpler inheritance, associations and realization relationships were extracted. Few of the classes remained unrelated to any of the classes in the diagram; even if source code shows the classes are related. ArgoUML and Reverse were unable to extract interfaces and realization relationships.

All the above tools, except 'Reverse' needs to drag and drop the classes to complete class diagram, once reverse engineering is complete. Reverse automatically generates the class diagram but all classes are not extracted.

Our examination shows that although all tools provide reverse engineering facility, most of them are able to recognize basic features like classes and associations .ArgoUML could not show that. Several advanced concepts like multiplicity, inverse association etc have not been addressed in these tools.

6. REFERENCES

[1] ArgoUML, http://argouml.tigris.org/ is a widely used open source tool for UML modeling tool.

[2] B. Bellay and H. Gall. A comparison of four reverse engineering tools. In 4th Working Conference on Reverse Engineering, pages 2–11. The Netherlands, 1997.

[3] http://javaboutique.internet.com/arith24/ :- 'Arithmetic 24 Game', which is developed in Java by Huahai Yang. It is a simulation of popular traditional card game.

[4] http://www.sparxsystems.com/products/ea/downloads.html :- Enterprise Architecture (EA) is widely used commercial UML modeling tool.

[5] http://www.neiljohan.com/projects/reverse/ :- Reverse is non commercial tool to convert java code to class diagram developed by Neil Johan.

[6] IBM Rational Rose Enterprise Edition

[7] Jeffrey Korn, Yih-Farn Chen and Eleftherios Koutsofios. Chava: Reverse Engineering and Tracking of Java Applets

[8] M. Armstrong and C. Trudeau. Evaluating Architectural Extractors. In 5th Working Conference on Reverse Engineering, pages 30–39. Hawaii, USA, 1998.

[9] Martin Keschenau. Reverse Engineering of UML specifications from java programs, Student Research Competition (OOPSLA'04)

[10] OMG: UML specification version 1.4

[11] Ralf Kollmann , Petri Selonen , Eleni Stroulia , Tarja Syst¨a_ , Albert Z¨undorf_A Study on the Current State of the Art in Tool-Supported UML-Based Static Reverse Engineering In Elizabeth Burd and Arie van Deursen, editors, Proc. 9thWorking Conference on Reverse Engineering. IEEE, Los Alamitos, 2002

[12] Yann-Ga¨ el Gu´eh´eneuc. A reverse Engineering Tool for Precise class diagram.(2004)

[13] Yann-Ga¨ el Gu´eh´eneuc and Herv´e Albin-Amiot, Recovering Binary Class Relationships: Putting Icing on the UML Cake- OOPSLA'04, Oct. 24-28, 2004, Vancouver, British Columbia, Canada.