


A Tool for Reversing Java Programs under Test to UML Sequence Diagrams

Siros Supavita Natachart Latheppitak
Narakorn Engsuwan Taratip Suwannasart

Faculty of Engineering, Chulalongkorn University, Thailand

A Tool for Reversing Java Programs under Test to UML Sequence Diagrams

1




Agenda

- Introduction
- Our Approach
 - Instrumentation
 - Diagram Generation
- Evaluation
- Conclusion & Future Works

A Tool for Reversing Java Programs under Test to UML Sequence Diagrams

2




Introduction

- In software development, specification and implementation are not usually in sync.
- Changes are sometimes applied directly to the implementation.
- Some software, especially legacy ones, may not have specification in the first place.

A Tool for Reversing Java Programs under Test to UML Sequence Diagrams

3




Reverse Engineering

- Reverse engineering is an essential process to solve this problem.
- Reverse engineering is the process of analyzing a subject system to create representations of the system at a higher level of abstraction.

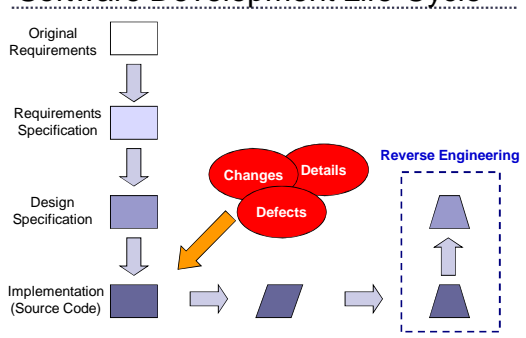
E.J. Chikofsky, J.H. Cross II, "Reverse Engineering and Design Recovery: A Taxonomy in IEEE Software", IEEE Computer Society, January 1990, pp. 13-17.

A Tool for Reversing Java Programs under Test to UML Sequence Diagrams

4




Software Development Life-Cycle



A Tool for Reversing Java Programs under Test to UML Sequence Diagrams

5



Static Analysis

- Most of the time, reverse engineering relies on static analysis.
- Static analysis is performed on structures of the subject software, particularly source code.
- However, it cannot reveal behavioral features, e.g. execution scenario, infeasible path etc.

A Tool for Reversing Java Programs under Test to UML Sequence Diagrams

6

Dynamic Analysis

- Dynamic analysis is usually performed on the subject software in a form of execution (possibly symbolic execution).
- Dynamic behavior of the subject software is the major target of this type of analysis.

A Tool for Reversing Java Programs under Test to UML Sequence Diagrams 7

Static vs. Dynamic Analysis

A Tool for Reversing Java Programs under Test to UML Sequence Diagrams 8

Our Approach

- Reverse engineering using dynamic analysis
- Reversing program execution scenarios into UML Sequence Diagrams

A Tool for Reversing Java Programs under Test to UML Sequence Diagrams 9

Our Approach

A Tool for Reversing Java Programs under Test to UML Sequence Diagrams 10

Instrumentation

- Binary level instrumentation
- Capture method calls into Message Sending Sequence
- Using AOP (AspectJ)

A Tool for Reversing Java Programs under Test to UML Sequence Diagrams 11

Instrumentation using AOP

A Tool for Reversing Java Programs under Test to UML Sequence Diagrams 12

Example of AspectJ Instrument

```

pointcut normalCall(Object obj) :
    call(* *(..) && target(obj) && exclusion());

before(Object obj): normalCall(obj){
    // Create a method call
    // Add to the current context

    // Create a new context for calls
    // activated by this call
}

after(Object obj) returning :
normalCall(obj){

    // Close the current context

    // Restore the previous context
}
    
```

A Tool for Reversing Java Programs under Test to UML Sequence Diagrams 13

Example of Instrumented Code

```

Service a;
Service b;

public void doSomething() {

    Instrument.beforeNormalCall();
    a.callService();
    Instrument.afterNormalCall();

    Instrument.beforeNormalCall();
    b.callService();
    Instrument.afterNormalCall();

}
    
```

A Tool for Reversing Java Programs under Test to UML Sequence Diagrams 14

Diagram Generation

- A "pic" file is generated from Message Sending Sequence.
- GNU plotutils "pic2plot" generates image files (GIF, SVG etc.) from the pic file <http://www.gnu.org/software/plotutils/>
- UMLGraph macro (by D. Spinellis) helps in drawing Sequence diagram elements (actor, object, life line etc.)

A Tool for Reversing Java Programs under Test to UML Sequence Diagrams 15

Diagram Generation Process

A Tool for Reversing Java Programs under Test to UML Sequence Diagrams 16

Example of a portion of pic file

```

.PS
copy "sequence.pic";
spacing = 0.15;

actor(A,"");
object(S1,":MainController");
placeholder_object(IN0);

step();
message(A,S1,"doSth(int,String)");
active(S1);
cmessage(S1,IN0,"test : B");
message(S1,IN0,"doSth(int)");
active(IN0);
step();
return_message(IN0,S1,"return int");

...
    
```

A Tool for Reversing Java Programs under Test to UML Sequence Diagrams 17

Evaluation

- The tools are tested with different types of method calls (instance method call, static method call, constructor call, and polymorphic method call).
- The generated diagrams accurately represent method call interactions of the subject software.

A Tool for Reversing Java Programs under Test to UML Sequence Diagrams 18

Example Program

```

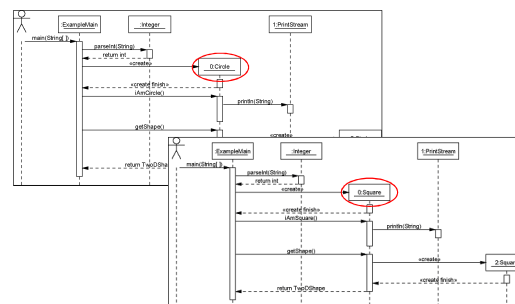
public class ExampleMain {
    public static void main(String[] args) {
        TwoDShape tdsTemp;
        switch(Integer.parseInt(args[0])){
            case 0 : {
                tdsTemp = new Circle(3.0,3.0,3.0);
                ((Circle)tdsTemp).iAmCircle();
                break;
            }
            case 1 : {
                tdsTemp = new Square(3.0,3.0,3.0);
                ((Square)tdsTemp).iAmSquare();
                break;
            }
            default : {
                tdsTemp = new Triangle(3.0,3.0,3.0);
                ((Triangle)tdsTemp).iAmTriangle();
                tdsTemp.area();
                break;
            }
        }
        tdsTemp.getShape();
    }
}

```

A Tool for Reversing Java Programs under Test to UML Sequence Diagrams

19

Example of Generated Diagrams



A Tool for Reversing Java Programs under Test to UML Sequence Diagrams

20

Conclusion & Future Works

- The tool can generate UML Sequence diagrams representing dynamic behavior of the subject software, including polymorphic interaction.
- The approach and the tool complement static analysis techniques to achieve more comprehensive reverse engineering result.
- In future, details such as loop or condition, which require more advanced instrumentation technique, may be captured and presented in the generated diagrams.

A Tool for Reversing Java Programs under Test to UML Sequence Diagrams

21