Eclat Tool Analysis

Team 3
Christine Ryu
Dahlia Bock
Faustinus Gozali

About Eclat

- Goal: Help discover errors and write new unit tests for Java classes.
- When running Eclat on a Java class, one would need to input two things:
  - The class(es) that is being tested
  - An example test suite, i.e. maybe a simple program that demonstrates the use of the class
- From that Eclat will automatically generate new test inputs different from the ones given in the example test suite.

Setting up Eclat

1. Download diakon.jar and eclat.jar
2. Add diakon.jar and eclat.jar to classpath

```
C:\15-413>set CLASSPATH=%CLASSPATH%;c:\15-413\daikon.jar;c:\15-413\eclat.jar
```

3. Create JunitTest to run against files
4. Run it to generate new inputs

```
C:\15-413>java eclat.textui.Main generate-inputs --create-regression-suite --tes
t CRAP/InputParser.java --test CRAP/OpenEnded.java --test CRAP/SingleChoice.java
 CRAP.JunitTest
```

How we applied Eclat

- Our project is in J2ME
  - Revised code to rely on JAVA instead of J2ME
- Could not test User Interface or interactive content
- Decided to test:
  - Parsing an SMS String
  - Creating a new Question object
  - Set/Get attributes of the question

```
/* ------------------------- TESTING QE ---------------------------------*/
    String qString = "How are you today";
    String qType = "QE";
    String UserId = "1111";
    String QuestionId = "2222";

    String testMsg = UserId + "*" + QuestionId + "*" + qType + qString;

    InputParser parser = new InputParser();
    Question q = parser.parsePayload( testMsg);

    if( !qString.equals(q.getQuestion()) ) {
        System.out.println("Question string not set properly");
    }
    if( !qType.equals(q.getQType()) ) {
        System.out.println("Question type not set properly");
    }
    if( !UserId.equals(q.getUserID()) ) {
        System.out.println("User ID is not set properly");
    }
    if( !QuestionId.equals(q.getQuestionID()) ) {
        System.out.println("Question ID not set properly");
    }

    System.out.println("---- Finished testing new Question creation ----");


    qString = "A new question here";
    q.setQuestion(qString);

    if( !qString.equals(q.getQuestion()) ) {
        System.out.println("Question string not set properly");
    }

    qType = "NM";
    q.setQType(qType);
    if( !qType.equals(q.getQType()) ) {
        System.out.println("Question type not set properly");
    }

    UserId = "1123";
    q.setUserID(UserId);
    if( !UserId.equals(q.getUserID()) ) {
        System.out.println("User ID is not set properly");
    }

    QuestionId = "5813";
    q.setQuestionID(QuestionId);
    if( !QuestionId.equals(q.getQuestionID()) ) {
        System.out.println("Question ID not set properly");
    }

    System.out.println("----Finished testing set methods ----");
/* ------------------------- TESTING SC ---------------------------------*/
```

```
C:₩WINDOWS₩system32₩cmd.exe
Observing CRAP.JunitTest's values as it executes.
Entering Daikon to detect invariants over observations.
Daikon version 4.1.7, released November 1, 2005; http://pag.csail.mit.edu/daikon
.
Processing trace data; reading 1 dtrace file:
[9:05:15 PM]: Finished reading JunitTest.dtrace.gz
Creating implications
Exiting Daikon.
Instrumenting sources for runtime invariant checking.
Compiling instrumented sources.
Generating inputs.


Constructing new inputs (round 1)...

Max. possible inputs for this round: 1
.
Constructed 1 new inputs (out of 1 possible).


Constructing new inputs (round 2)...

Max. possible inputs for this round: 8
iiiiiii
Constructed 7 new inputs (out of 8 possible).


Constructing new inputs (round 3)...

Max. possible inputs for this round: 8

Constructed 0 new inputs (out of 8 possible).


Constructing new inputs (round 4)...

Max. possible inputs for this round: 8

Constructed 0 new inputs (out of 8 possible).
Done generating inputs.

Created file containing inputs in text format: C:₩15-413₩eclat-misc₩JunitTest.al
linputs.txt.zip
This file contains ALL the inputs generated,
in case you want to see them.
Creating JUnit class.

JUnit test suite has been created: C:₩15-413₩eclat-src₩eclatgen₩problems₩EclatTe
st.java
```

Results and Analysis

- Eclat produces eclat-src, eclat-scratch and eclat-misc
- eclat-src/
  - Contains the JUnit suites that Eclat generated to test the stack
  - Determines the minor and major preconditions and postconditions for each method in each class which is used to check for violations of the conditions
- eclat-misc/
  - Contains a human-readable listing of all the inputs that Eclat generated

Results and Analysis

- We were able to:
  - See what unit tests could be run
  - The minor/major pre/post-conditions
  - Possible inputs for our program
- We were unable to:
  - Run the test suites and see exactly where the program failed
  - Run for any interactive input

Results Example

```
================================================================================
round: 2
id: 1096
INPUT CLASSIFIED AS <illegal>
================================================================================

  1. CRAP.InputParser var12 = new CRAP.InputParser();

  2. CRAP.Question var1027 = var12.parseMultipleChoice((java.lang.String)null);


  ------------------------------------------------------------------------
  Prep code evaluation (lines 1 through 1).

  EXCEPTIONS: none.
  INVARIANT VIOLATIONS:
  none.

  ------------------------------------------------------------------------
  Test expression evaluation (line 2).

  EXCEPTIONS:     java.lang.NullPointerException
  INVARIANT VIOLATIONS:
  none.

  ------------------------------------------------------------------------
  Explanation:
  No violations, but found an exception or error. Since one of
  the arguments was null, I will classify this input as
  illegal.
```

Benefits

- Requires a downloading of only 2 relatively small files to run
- Generates new inputs and the results of executing those inputs on the class that is being tested
  - Generates inputs that the developer may not have thought of
  - Shows how extensively Eclat is testing the class and which types of inputs will cause errors/bugs.
  - Can aid in the creation of new unit tests
- Uses a small list of commands to generate the new test inputs
  - Though a short list, it covers a significant number of different options that a user can add when running the tool. Each extra option is documented in the Eclat Manual.

Drawbacks

- Not easy to use for all platforms
  - Requires Java 1.5, but Andrew uses 1.4
  - Tutorial provides Unix commands, but is hard to follow from a Windows Command line
- Documentation is sparse
  - Overlooks issues that a user might face regarding setting classpath variables
- Cannot handle User Interface testing
  - Does not deal with interactive input

Scope of Applicability

- Assumes that javac and UNIX commands are available for use
  - Easy to use for Unix command line with Java 1.5
- Non-interactive JAVA programs

Conclusion!

- Eclat is helpful in generating possible inputs for an application
  - Helps to classify as illegal/fault-revealing/normal execution so we can deal with it accordingly
- Difficult to use unless UNIX and Java 1.5 are readily available