TEST NAME RECOMMENDATION

Final bachelor presentation

Christian Zürcher

11.06.2019

SUMMARY

- Motivation
- Tool
- Evaluation
  - Test comprehension
  - Name recommendation
- Conclusion

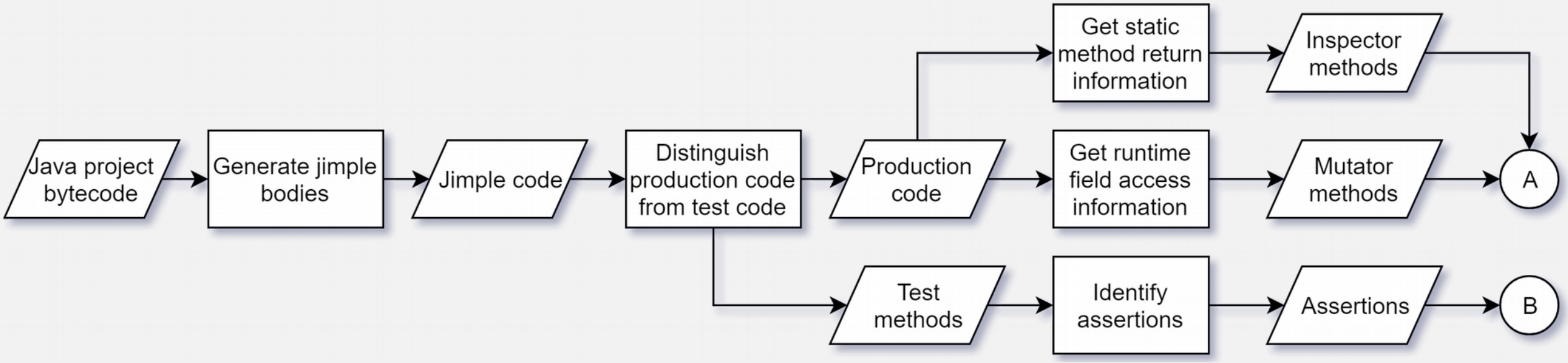
MOTIVATION

```
public void testRetreiveProxyIsNullAfterRemoveProxy() {
    IModel model = Model.getInstance();
    IProxy proxy = new Proxy("sizes", new String[]{"7","13","21"});
    model.registerProxy(proxy);
    IProxy removedProxy = model.removeProxy("sizes");
    assertEquals(removedProxy.getProxyName() , "sizes");
    proxy = model.retrieveProxy("sizes");
    assertNull("Expecting proxy is null", proxy);
}
```

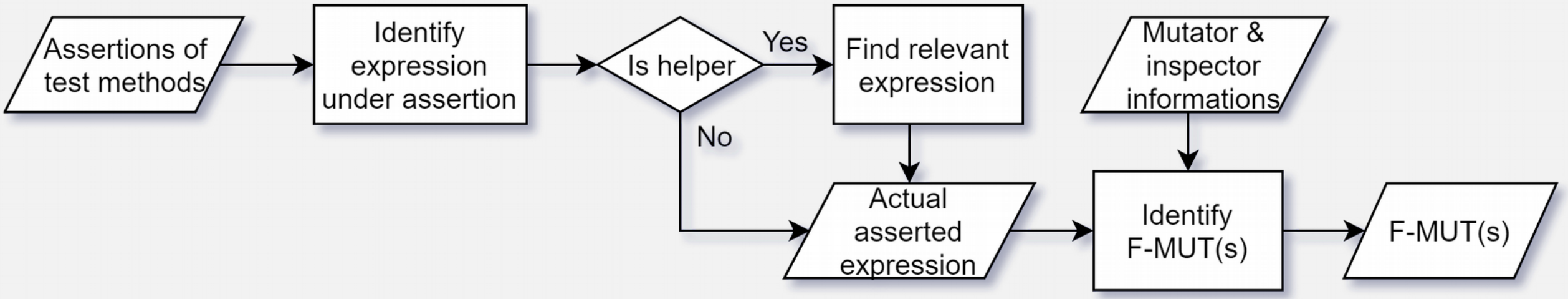
RESEARCH QUESTIONS

- Does the focal method under test information help to identify the different sections of a test case (setup, execution and oracle)?
- Can the focal method help to generate meaningful test names?

F-MUT ANALYSIS TOOL



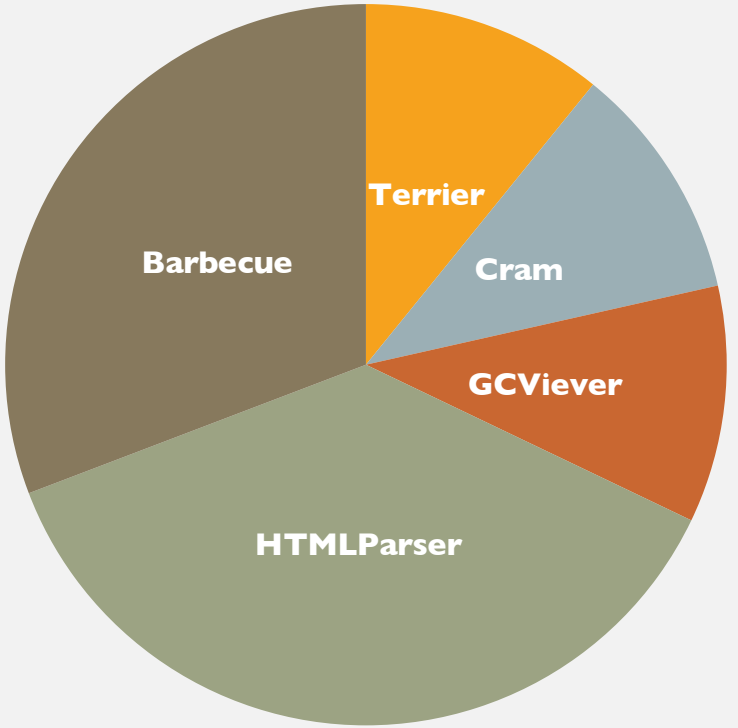
F-MUT ANALYSIS TOOL



EVALUATION

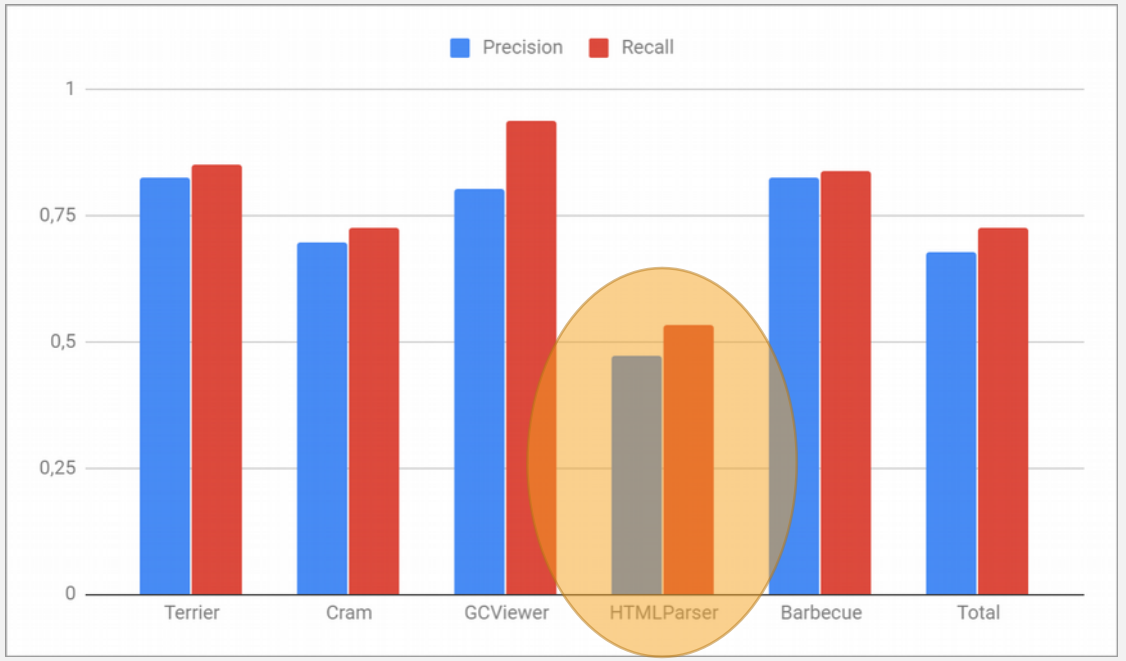
- Projects:
  - Barbecue (all test cases - 142)
  - HTMLParser (first 171)
  - GCViever (50 random)
  - Cram (50 random)
  - Terrier (50 random)

- Total of 463 test cases



F-MUT DETECTION RESULTS

- Precision: 67,7%
- Recall: 72,5%
- Success: 82,4%



ANALYSIS PROBLEMS: EXAMPLES

- F-MUT inside unrelated method

```
public void testParseParameters() {
    getParameterTableFor("a b = \"c\"");
    assertEquals("Value","c",((Attribute)(attributes.elementAt (2))).getValue ());
}
```

ANALYSIS PROBLEMS: EXAMPLES

- Asserting size of return object manually

```
public void testElements() throws Exception {
        StringBuffer hugeData = new StringBuffer();
        for (int i=0;i<5001;i++) hugeData.append('a');
        createParser(hugeData.toString());
        int i = 0;
        for (NodeIterator e = parser.elements();e.hasMoreNodes();) {
            node[i++] = e.nextNode();
        }
        assertEquals("There should be 1 node identified",1,i);
}
```

TEST STRUCTURE

```
public void test() {
    IModel model = Model.getInstance();
    IProxy proxy = new Proxy("sizes", new String[]{"7","13","21"});
    model.registerProxy(proxy);
    IProxy removedProxy = model.removeProxy("sizes");
    assertEquals(removedProxy.getProxyName() , "sizes");
    proxy = model.retrieveProxy("sizes");
    assertNull("Expecting proxy is null", proxy);
}
```

setup

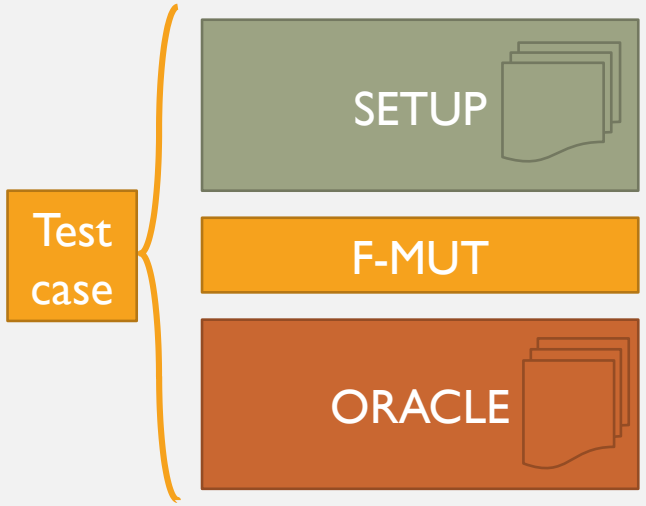
F-MUT

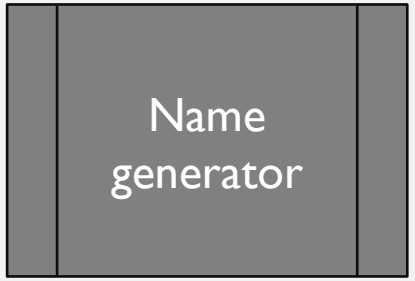
oracle

F-MUTS FOR NAMES

- How do we use the F-MUTs for naming

test*[F-MUT]_[setUp]_[oracle]*
test*[F-MUT]_[oracle]***When***[setUp]*
test*[oracle]***After***[F-MUT]***With***[setUp]*





test*[oracle]***After***[F-MUT]*
test*[oracle]***After***[F-MUT]****And*** *[oracle]***After***[F-MUT]*

NAME SUGGESTION (CONSTANT INCLUSION)

```
@Test public void testTwoOfThreeSentences() {
    Summariser s = new DefaultSummariser();
    String summary = s.generateSummary(doc1, new String[]{"lorem", "ipsum"});
    String expected = "Lorem Ipsum is simply dummy text of the printing and
        typesetting industry...Lorem Ipsum has been the industry's standard
        dummy text ever since the 1500s, when an unknown";
    assertEquals(expected, summary);
}
```

testGenerateSummaryEqualsLorem_Ipsum_is_simply_dummy_text_of_the_printing_and_typesetting_industry**TripleDot**Lorem_Ipsum_has_been_the_industrys_standard_dummy_text_ever_since_the_1500s**Comma**_when_an_unknown

NAME SUGGESTION (WEIRD JIMPLE CONVERSIONS)

```
assertFalse ( mod.hashCode() == mod2.hashCode())
```

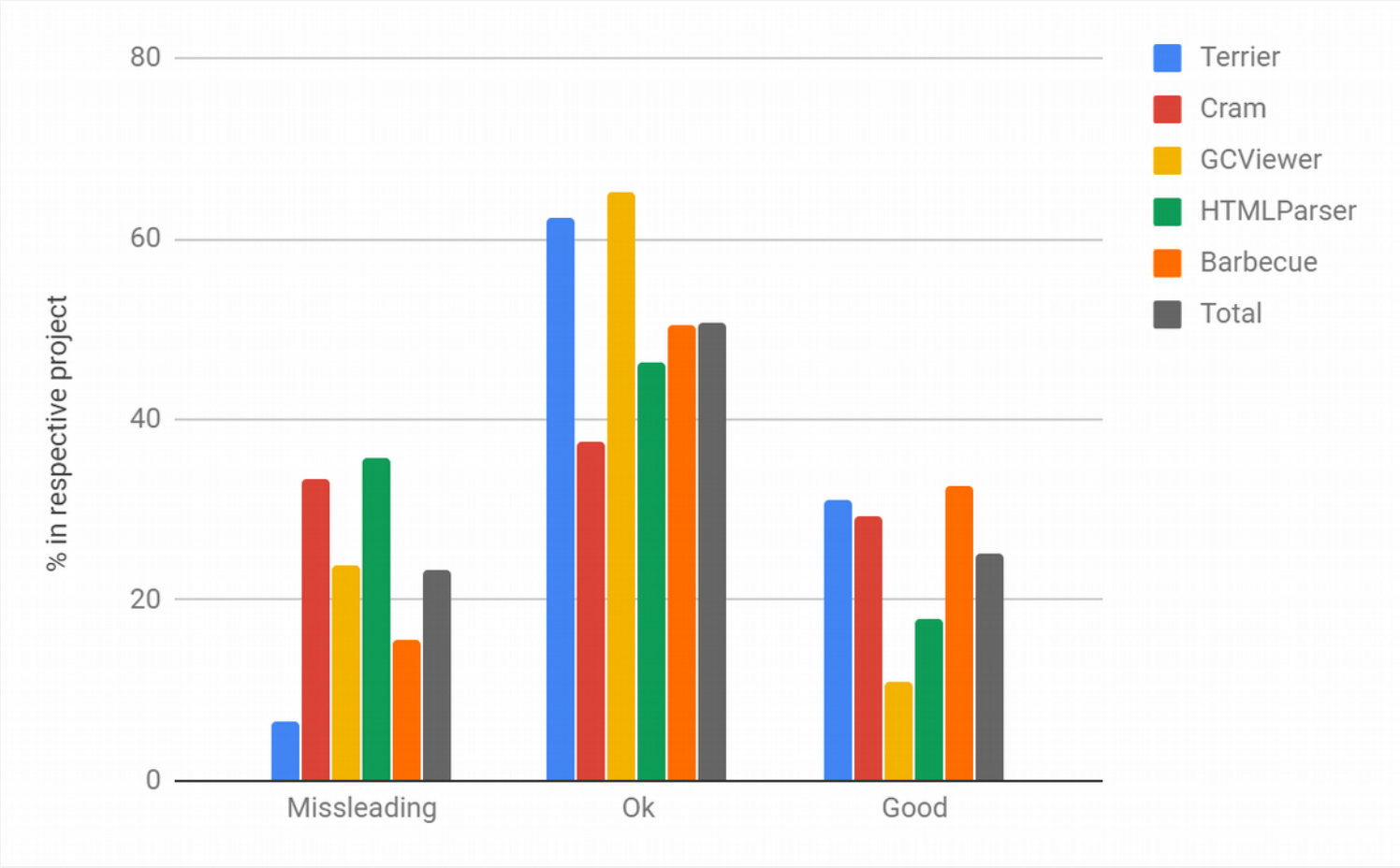
```
$stack7 = virtualinvoke mod.<net. sourceforge .barbecue.Module: int hashCode()>();
$stack8 = virtualinvoke mod2.<net. sourceforge .barbecue.Module: int hashCode()>();

if $stack7 != $stack8 goto label1 ;
        $stack9 = 1;
        goto label2 ;
label1 :
        $stack9 = 0;
label2 :
        staticinvoke <net. sourceforge .barbecue.ModuleTest: void assertFalse (boolean)>($stack9);
```

testHashCode**NotEqual**ToHashCodeIsFalse

instead of

testHashCode**Equal**ToHashCodeIsFalse

F-MUTS FOR NAMES



COMPARISON WITH "NAMEASSIST"

- Eclipse plugin with a "natural language program analysis" based approach
- Limited to single assertions

```
@Test public void test() throws Exception {
    servlet = new BarcodeServletMock();
    params.put("height", "200");
    params.put("width", "3");
    params.put("resolution", "72");
    req.setParameters(params);
    servlet.doGet(req, res);
    Barcode barcode = servlet.getBarcode();
    assertEquals(72, barcode.getResolution());
}
```

- "testDoGet"
- "testDoGetResolutionIs72"
- "testDoGetResolutionIs72WhenParamsResolutionIs72AndSettingParameters"

VS.

- testGetResolutionEquals72AfterDoGet

COMPARISON WITH "NAMEASSIST"

- Eclipse plugin with a "natural language program analysis" based approach
- Limited to single assertions

```
@Test public void test() throws Exception {
    servlet = new BarcodeServletMock();
    params.put("height", "200");
    params.put("width", "3");
    params.put("resolution", "72");
    req.setParameters(params);
    servlet.doGet(req, res);
    Barcode barcode = servlet.receiveBarcode();
    assertEquals(72, barcode.getRes());
}
```

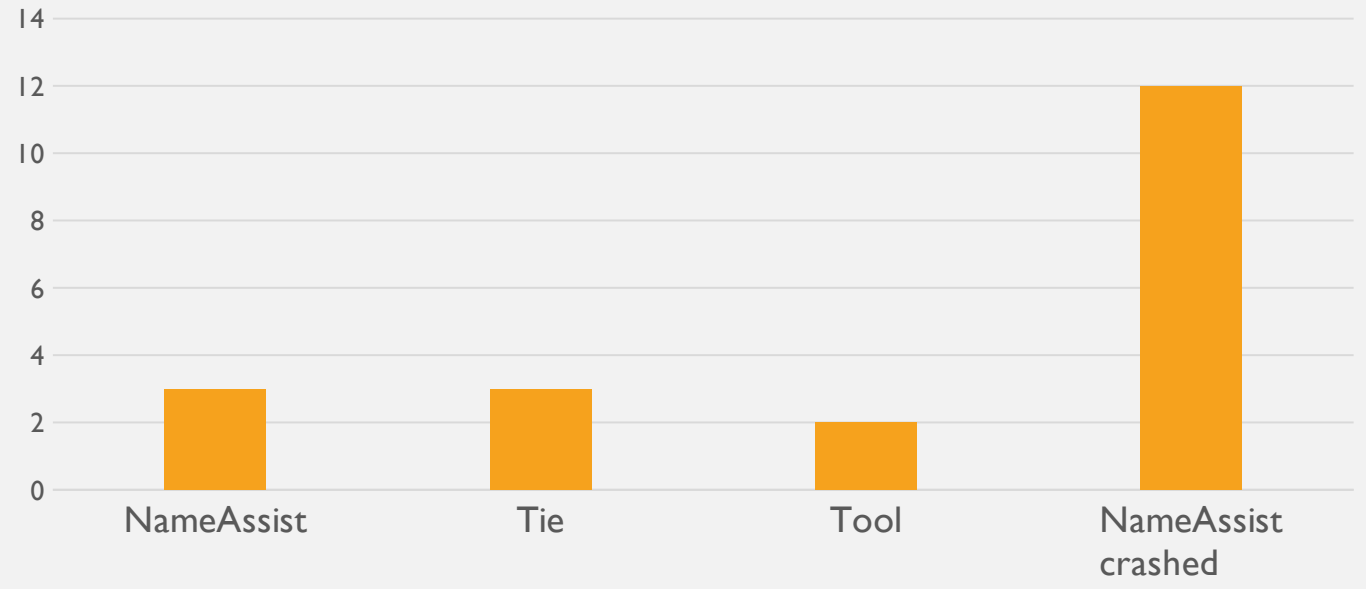
- "testReceiveBarcode"
- "testReceiveBarcodeResIs72"
- "testReceiveBarcodeResIs72WhenDoGet"

VS.

- testGetResEquals72AfterDoGet

COMPARISON WITH "NAMEASSIST"

- Eclipse plugin with a "natural language program analysis" based approach
- Limited to single assertions



CONCLUSION

- Results are promising
  - Over 80% for detected F-MUTs
  - Over 75% of acceptable names
- Human interpretation is still very important
- Suggestion tool

QUESTIONS