E. Jürgens, B. Hummel, F. Deissenboeck, M. Feilkas

Static Bug Detection Through Analysis of Inconsistent Clones

19. February 2008

Redundancy in Software

Eclipse 3.2.2, ClassFile.java, 12 instances

```
this.contents[localContentsOffset++] = (byte) (251 + numberOfDifferentLocals);
this.contents[localContentsOffset++] = (byte) (offsetDelta >> 8);
this.contents[localContentsOffset++] = (byte) offsetDelta;
int index = currentFrame.getIndexOfDifferentLocals(numberOfDifferentLocals);
int numberOfLocals = currentFrame.getNumberOfLocals();
for (int i = index; i < currentFrame.locals.length && numberOfDifferentLocals > 0; i++) {
  if (localContentsOffset + 6 >= this.contents.length) {
    resizeContents(6);
  }
  VerificationTypeInfo info = currentFrame.locals[i];
  if (info == null) {
    this.contents[localContentsOffset++] = (byte) VerificationTypeInfo.ITEM_TOP;
  } else {
    switch(info.id()) {
      case T_boolean :
      case T_byte :
      case T_char :
      case T_int :
      case T_short :
        this.contents[localContentsOffset++] = (byte) VerificationTypeInfo.ITEM_INTEGER;
        break;
      case T_float :
        this.contents[localContentsOffset++] = (byte) VerificationTypeInfo.ITEM_FLOAT;
        break;
      case T_long :
        this.contents[localContentsOffset++] = (byte) VerificationTypeInfo.ITEM_LONG;
        i++;
        break;
      case T_double :
        this.contents[localContentsOffset++] = (byte) VerificationTypeInfo.ITEM_DOUBLE;
        i++;
        break;
      case T_null :
        this.contents[localContentsOffset++] = (byte) VerificationTypeInfo.ITEM_NULL;
        break;
      default:
        this.contents[localContentsOffset++] = (byte) info.tag;
        switch (info.tag) {
          case VerificationTypeInfo.ITEM_UNINITIALIZED :
            int offset = info.offset;
            this.contents[localContentsOffset++] = (byte) (offset >> 8);
            this.contents[localContentsOffset++] = (byte) offset;
            break;
          case VerificationTypeInfo.ITEM_OBJECT :
            int indexForType = constantPool.literalIndexForType(info.constantPoolName());
            this.contents[localContentsOffset++] = (byte) (indexForType >> 8);
            this.contents[localContentsOffset++] = (byte) indexForType;
        }
    }
    numberOfDifferentLocals--;
  }
}
break;
```

```
localContentsOffset += 2; // leave two spots for number of locals
int numberOfLocalEntries = 0;
numberOfLocals = currentFrame.getNumberOfLocals();
int numberOfEntries = 0;
int localsLength = currentFrame.locals == null ? 0 : currentFrame.locals.length;
for (int i = 0; i < localsLength && numberOfLocalEntries < numberOfLocals; i++) {
  if (localContentsOffset + 3 >= this.contents.length) {
    resizeContents(3);
  }
  VerificationTypeInfo info = currentFrame.locals[i];
  if (info == null) {
    this.contents[localContentsOffset++] = (byte) VerificationTypeInfo.ITEM_TOP;
  } else {
    switch(info.id()) {
      case T_boolean :
      case T_byte :
      case T_char :
      case T_int :
      case T_short :
        this.contents[localContentsOffset++] = (byte) VerificationTypeInfo.ITEM_INTEGER;
        break;
      case T_float :
        this.contents[localContentsOffset++] = (byte) VerificationTypeInfo.ITEM_FLOAT;
        break;
      case T_long :
        this.contents[localContentsOffset++] = (byte) VerificationTypeInfo.ITEM_LONG;
        i++;
        break;
      case T_double :
        this.contents[localContentsOffset++] = (byte) VerificationTypeInfo.ITEM_DOUBLE;
        i++;
        break;
      case T_null :
        this.contents[localContentsOffset++] = (byte) VerificationTypeInfo.ITEM_NULL;
        break;
      default:
        this.contents[localContentsOffset++] = (byte) info.tag;
        switch (info.tag) {
          case VerificationTypeInfo.ITEM_UNINITIALIZED :
            int offset = info.offset;
            this.contents[localContentsOffset++] = (byte) (offset >> 8);
            this.contents[localContentsOffset++] = (byte) offset;
            break;
          case VerificationTypeInfo.ITEM_OBJECT :
            int indexForType = constantPool.literalIndexForType(info.constantPoolName());
            this.contents[localContentsOffset++] = (byte) (indexForType >> 8);
            this.contents[localContentsOffset++] = (byte) indexForType;
        }
    }
    numberOfLocalEntries++;
  }
  numberOfEntries++;
}
```

Detection of clone-related bugs

Counter-Measures

Existing Approaches

- *Preventive*: Detect clones, refactor to remove them
- *Constructive*: Support impact analysis with clone detection, linked editing

Drawbacks

- Large restructurings are costly, error prone, typically infeasible in practice
- Constructive measures are no help for existing inconsistencies

Contribution: Analytic Approach

Tool support for cost-efficient identification of inconsistencies in existing software

+ Immediately applicable to detect existing bugs
+ Complementary to existing approaches

  (both clone management and bug detection)

Approach

1 Detect Clones

2 Locate Gaps

3 Filter (Heuristic)

4 Manual inspection of remaining inconsist.

```
protected Document generateAntTask() {
  try {
    DocumentBuilderFactory factory =
      DocumentBuilderFactory.newInstance();
    Document doc =
      factory.newDocumentBuilder().newDocument();
    Element root = doc.createElement("project");
    root.setAttribute("name", "build");
    root.setAttribute("default", "feature_export");
    doc.appendChild(root);

    Element target = doc.createElement("target");
    target.setAttribute("name", "feature_export");
    root.appendChild(target);

    Element export = doc.createElement("pde.exportFeatures");
    export.setAttribute("features", getFeatureIDs());
    export.setAttribute("destination", fPage.getDestination());
    String filename = fPage.getFileName();
    if (filename != null)
      export.setAttribute("filename", filename);
    export.setAttribute("exportType", getExportOperation());
    export.setAttribute("useJARFormat",
      Boolean.toString(fPage.useJARFormat()));
    export.setAttribute("exportSource",
      Boolean.toString(fPage.doExportSource()));
    return doc;
  } catch (DOMException e) {
  } catch (FactoryConfigurationError e) {
  } catch (ParserConfigurationException e) {
  }
  return null;
}
```

```
protected Document generateAntTask() {
  try {
    DocumentBuilderFactory factory =
      DocumentBuilderFactory.newInstance();
    Document doc =
      factory.newDocumentBuilder().newDocument();
    Element root = doc.createElement("project");
    root.setAttribute("name", "build");
    root.setAttribute("default", "plugin_export");
    doc.appendChild(root);

    Element target = doc.createElement("target");
    target.setAttribute("name", "plugin_export");
    root.appendChild(target);

    Element export = doc.createElement("pde.exportPlugins");
    export.setAttribute("plugins", getPluginIDs());
    export.setAttribute("destination", fPage.getDestination());
    String filename = fPage.getFileName();
    if (filename != null)
      export.setAttribute("filename", filename);
    export.setAttribute("exportType", getExportOperation());
    export.setAttribute("useJARFormat",
      Boolean.toString(fPage.useJARFormat()));
    export.setAttribute("exportSource",
      Boolean.toString(fPage.doExportSource()));
    target.appendChild(export);
    return doc;
  } catch (DOMException e) {
  } catch (FactoryConfigurationError e) {
  } catch (ParserConfigurationException e) {
  }
  return null;
}
```

Bug 155070 in Eclipse 3.2.2

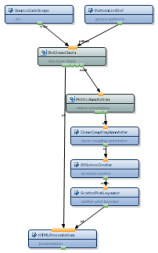
Case Study

Münchener Rück
Munich Re Group

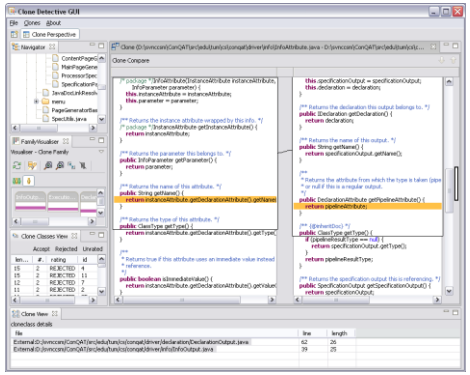
Tools

Detection

conQAT



Inspection



Analyzed System

- Business Information System, C#
- ~215 kLoC hand-written code

Results

- 106 inconsistencies (272 code fragments)
- 67 false positives (64%)

  (Order of commutative statements, Inconsistent use of optional braces, …)

- 11 Probable Bugs (i.e. missing null checks)
- 8 Style issues (unused / commented code)
- 20 open inconsistencies (under investigation)

=> Interesting results for developers

Summary

Future Work

- More and larger industrial case studies
- Approximate Suffix-Tree based clone detection
- Release of tools under Apache License

Contribution

- Novel approach to detect clone related bugs
- Scalable, multi-language, requires no additional information
- Complementary to existing clone detection and bug detection research
- Working tools

Contact

Elmar Juergens **juergens@in.tum.de**