Code Clone

```java
// Utilities for arrays of elements
public String showElements(ModelElement[] elements, String nomsg) {
    boolean found = false;
    StringBuffer res = new StringBuffer();
    if (elements != null) {
        Index.getInstance().setCurrentRenderer(
            FlatReferenceRenderer.getInstance());
        for (int i = 0; i < elements.length; i++) {
            ModelElement el = elements[i];
            res.append(showElementLink(el)).append(HTML.LINE_BREAK);
            found = true;
        }
        Index.getInstance().setCurrentRenderer();
    }
    if (!found && nomsg != null && nomsg.length() > 0) {
        res.append(HTML.italics(nomsg));
    }
    return res.toString();
}
```
Agenda

Related Work
Empirical Study
Detection of inconsistent clones
Conclusion
Related Work

Indicating harmfulness

[Lague97]: *inconsistent evolution* of clones in industrial telecom. SW.

[Monden02]: *higher revision number* for files with clones in legacy SW.

[Kim05]: substantial amount of *coupled changes* to code clones.

[Li06], [SuChiu07] and [Aversano07], [Bakota07]: discovery of bugs through search for inconsistent clones or clone evolution analysis.

⇒ Indication for increased maintenance effort or faults

Doubting harmfulness

[Krinke07]: inconsistent clones hardly ever become consistent later.

[Geiger06]: Failure to statistically verify impact of clones on change couplings

[Lozano08]: Failure to statistically verify impact of clones on changeability.

⇒ Does not confirm increased maintenance effort or faults
Related Work (2)

Limitations of previous studies

• Indirect measures (e.g. stability of cloned vs. non-cloned code) used to determine effect of cloning are inaccurate
• Analyzed systems are too small or omit industrial software

This Work

• Manual inspection of inconsistent clones by system developers
  ⇒ No indirect measures of consequences of cloning
• Both industrial and open source software analyzed
• Quantitative data
**Terminology**

**Clone**
- Sequence of normalized statements
- At least one other occurrence in the code

**Exact clone**
- Edit distance between clones = 0

**Inconsistent clone**
- Edit distance between clones > 0 & below given threshold

**(Inconsistent) Clone Group**
- Set of clones at different positions (with at least 1 inconsistent clone)
- Semantic relationship between clones
Research Questions

RQ1: Are clones changed inconsistently?

\[ \frac{|IC|}{|C|} \]

RQ2: Are inconsistent clones created unintentionally?

\[ \frac{|UIC|}{|IC|} \]

RQ3: Can inconsistent clones be indicators for faults in real systems?

\[ \frac{|F|}{|IC|}, \frac{|F|}{|UIC|} \]
Study Design

Clone group candidate detection
- Novel algorithm
- Tailored to target program $\rightarrow CC$

False positive removal
- Manual inspection of all inconsistent and ¼ exact CCs
- Performed by researchers $\rightarrow C, IC$

Assessment of inconsistencies
- All inconsistent clone groups inspected
- Performed by developers $\rightarrow UIC, F$
Detection of Inconsistent Clones

Approach

- Token based for easy adaptation to new (incl. legacy) languages
- Suffix tree of normalized statements
- Novel edit-distance based suffix tree traversal algorithm
- Scalability: 500 kLOC: 3m, 5.6 MLOC: 3h

Implementation

- Detection steps implemented as pipeline
- Configurable for project-specific tailoring
- Implemented as part of ConQAT clone detection infrastructure
## Study Objects

<table>
<thead>
<tr>
<th>System</th>
<th>Organization</th>
<th>Language</th>
<th>Age (years)</th>
<th>Size (kLoC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Munich Re</td>
<td>C#</td>
<td>6</td>
<td>317</td>
</tr>
<tr>
<td>B</td>
<td>Munich Re</td>
<td>C#</td>
<td>4</td>
<td>454</td>
</tr>
<tr>
<td>C</td>
<td>Munich Re</td>
<td>C#</td>
<td>2</td>
<td>495</td>
</tr>
<tr>
<td>D</td>
<td>LV 1871</td>
<td>Cobol</td>
<td>17</td>
<td>197</td>
</tr>
<tr>
<td>Sysiphus</td>
<td>TUM</td>
<td>Java</td>
<td>8</td>
<td>281</td>
</tr>
</tbody>
</table>

International reinsurance company, 37,000 employees

Munich-based life-insurance company, 400 employees

Sysiphus: Open source collaboration environment for distributed SW development. Developed at TUM.
## Results (1)

<table>
<thead>
<tr>
<th>Project</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>Sysip.</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone groups $</td>
<td>C</td>
<td>\rangle$</td>
<td>286</td>
<td>160</td>
<td>326</td>
<td>352</td>
</tr>
<tr>
<td>Inconsistent CGs $</td>
<td>IC</td>
<td>\rangle$</td>
<td>159</td>
<td>89</td>
<td>179</td>
<td>151</td>
</tr>
<tr>
<td>Unint. Incons. $</td>
<td>UIC</td>
<td>\rangle$</td>
<td>51</td>
<td>29</td>
<td>66</td>
<td>15</td>
</tr>
<tr>
<td>Faulty CGs $</td>
<td>F</td>
<td>\rangle$</td>
<td>19</td>
<td>18</td>
<td>42</td>
<td>5</td>
</tr>
</tbody>
</table>
Do Code Clones Matter?

Discovered Faults

System-Crash or Data Loss  17
• Exceptions
• Erroneous transaction handling

Unexpected user-visible behavior  44
• Wrong messages
• Inconsistent behavior in similar dialogs/forms

Unexpected non-user visible behavior  46
• Resource management
• Exception handling / log messages
## Results (2)

RQ1: Are clones changed inconsistently?
RQ2: Are inconsistent clones created unintentionally?
RQ3: Can inconsistent clones be indicators for faults …?

Can *unintentionally* incons. clones be indicators …?

<table>
<thead>
<tr>
<th>System</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>Sysip.</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>RQ1</td>
<td>56%</td>
<td>56%</td>
<td>55%</td>
<td>43%</td>
<td>48%</td>
<td>52%</td>
</tr>
<tr>
<td>RQ2</td>
<td>32%</td>
<td>33%</td>
<td>37%</td>
<td>10%</td>
<td>29%</td>
<td>28%</td>
</tr>
<tr>
<td>RQ3</td>
<td>12%</td>
<td>20%</td>
<td>23%</td>
<td>3%</td>
<td>16%</td>
<td>15%</td>
</tr>
<tr>
<td></td>
<td>37%</td>
<td>62%</td>
<td>64%</td>
<td>33%</td>
<td>55%</td>
<td>50%</td>
</tr>
</tbody>
</table>
## Threats to Validity

<table>
<thead>
<tr>
<th>Threat</th>
<th>Mitigation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Construct</strong></td>
<td></td>
</tr>
<tr>
<td>• Analysis of latest version instead of evolution.</td>
<td>• All inconsistencies of interest, independent of creation time.</td>
</tr>
<tr>
<td><strong>Internal</strong></td>
<td></td>
</tr>
<tr>
<td>• Developer review error</td>
<td>• Conservative strategy only makes positive answers harder</td>
</tr>
<tr>
<td>• Clone Detector Configuration</td>
<td>• Validated during pre-study</td>
</tr>
<tr>
<td><strong>External</strong></td>
<td></td>
</tr>
<tr>
<td>• System selection not random (impact on transferability)</td>
<td>• 5 different dev. organizations</td>
</tr>
<tr>
<td></td>
<td>• 3 different languages</td>
</tr>
<tr>
<td></td>
<td>• Technically different</td>
</tr>
</tbody>
</table>
Do Code Clones Matter?

http://wwwbroy.in.tum.de/~ccsm/icse09

- Version of ConQAT used for the study
  (includes both detection and inspection tools)

- Source code and all results for Sysiphus

Apache License

ABAP, Ada, C#, C/C++, COBOL, Java, VB, PL/I

IDE Integration, Visualizations, …
Summary

Clone detection
• Scalable algorithm for inconsistent clone detection.
• Open source implementation (ConQAT).

Consequences of code cloning on program correctness
• Inconsistent clones constituted numerous faults in productive software.
• Every second unintentional inconsistency constitutes a fault.
Conclusion

Code clones do matter.