<u>Elmar Juergens</u>, Florian Deissenboeck,
Benjamin Hummel, Stefan Wagner

Do Code Clones Matter?

May 22nd, 2009
31st ICSE, Vancouver

TECHNISCHE
UNIVERSITÄT
MÜNCHEN

Code Clone

```
// 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().resetCurrentRenderer();
    }
    if (!found && nomsg != null && nomsg.length() > 0) {
        res.append(HTML.italics(nomsg));
    }
    return res.toString();
}
```

```
// 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().resetCurrentRenderer();
    }
    if (!found && 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?

$$|IC| / |C|$$

RQ2: Are inconsistent clones created unintentionally?

$$|UIC| / |IC|$$

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

$$|F| / |IC|, |F| / |UIC|$$

Clone Groups **C**
(exact and incons.)

Inconsistent clone groups **IC**

Unintentionally Inconsistent Clone Groups **UIC**

Faulty clone Groups **F**

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**

Tool detected clone group candidates **CC**

Clone groups **C** (exact and incons.)

Inconsistent clone groups **IC**

Unintentionally inconsistent clone groups **UIC**

Faulty clone groups **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
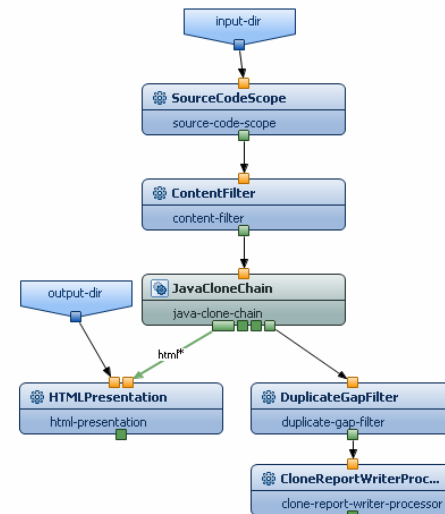
Study Objects

System	Organization	Language	Age (years)	Size (kLoC)
A	Munich Re	C#	6	317
B	Munich Re	C#	4	454
C	Munich Re	C#	2	495
D	LV 1871	Cobol	17	197
Sysiphus	TUM	Java	8	281

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)

Project	A	B	C	D	Sysip.	Sum
Clone groups $\lvert C \rvert$	286	160	326	352	303	**1427**
Inconsistent CGs $\lvert IC \rvert$	159	89	179	151	146	**724**
Unint. Incons. $\lvert UIC \rvert$	51	29	66	15	42	**203**
Faulty CGs $\lvert F \rvert$	19	18	42	5	23	**107**

CC

C

IC

UIC

F
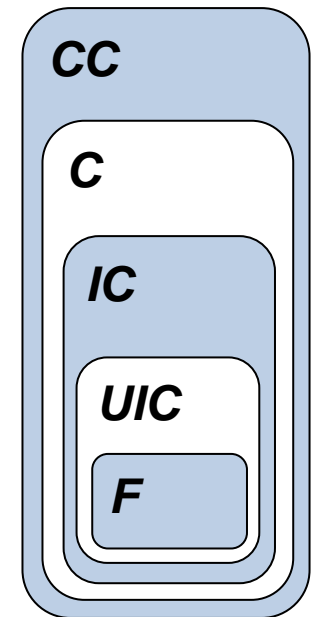
Discovered Faults

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 …?

System	A	B	C	D	Sysip.	Mean
RQ1 |IC| / |C|	56%	56%	55%	43%	48%	**52%**
RQ2 |UIC| / |IC|	32%	33%	37%	10%	29%	**28%**
RQ3 |F| / |IC|	12%	20%	23%	3%	16%	**15%**
|F| / |UIC|	37%	62%	64%	33%	55%	**50%**

Threats to Validity

	Threat	Mitigation
Construct	• Analysis of latest version instead of evolution.	• All inconsistencies of interest, independent of creation time.
Internal	• Developer review error	• Conservative strategy only makes positive answers harder
	• Clone Detector Configuration	• Validated during pre-study
External	• System selection not random (impact on transferability)	• 5 different dev. organizations • 3 different languages • Technically different

Study Replication

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

http://www.conqat.org

- 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.