Elmar Juergens, Florian Deissenboeck,
Benjamin Hummel, Stefan Wagner

Do Code Clones Matter?

May 22nd, 2009
31st ICSE, Vancouver

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?

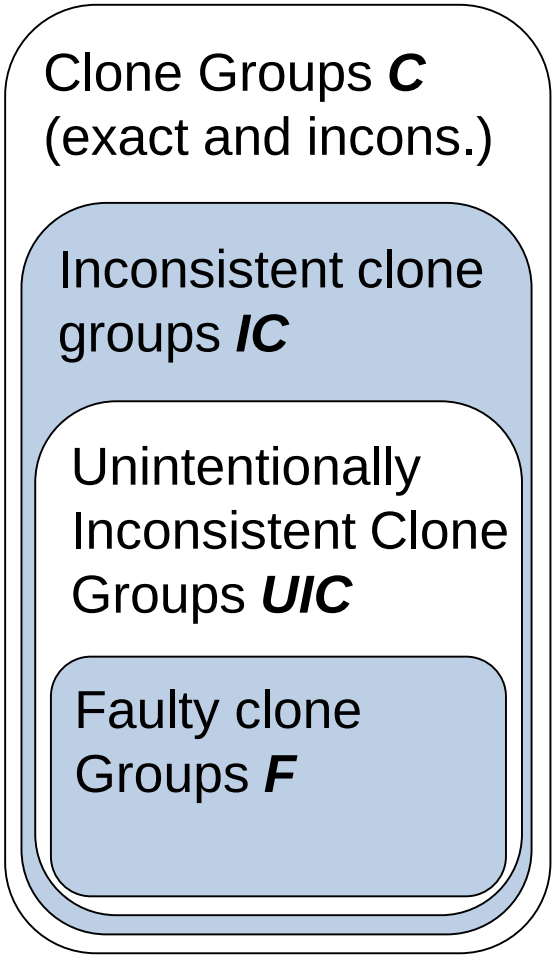
$$|IC| / |C|$$

RQ2: Are inconsistent clones created unintentionally?

$$|UIC| / |IC|$$

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

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



Study Design

Clone group candidate detection

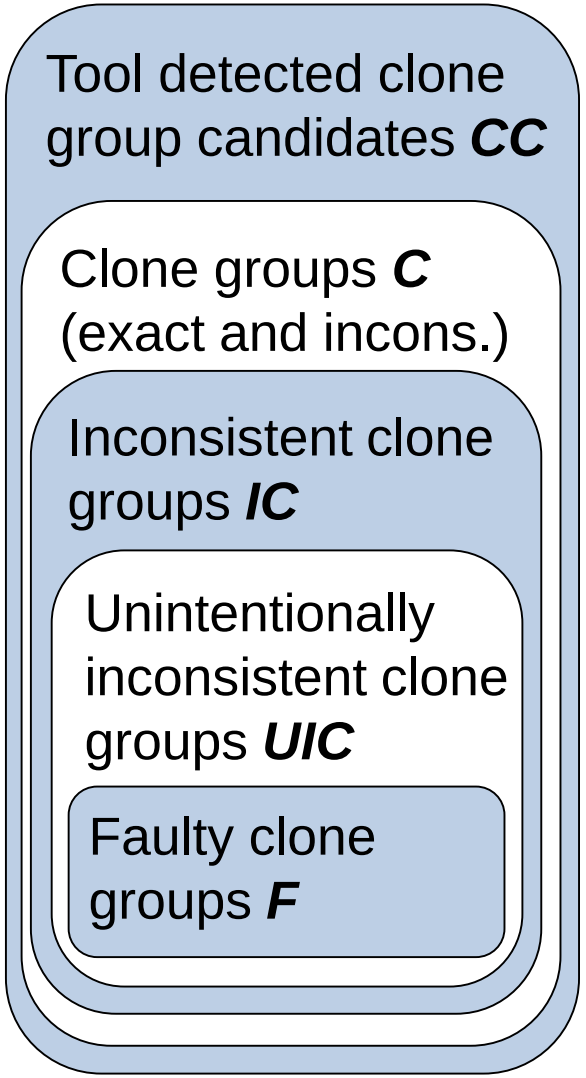
- Novel algorithm
- Tailored to target program → ***CC***

False positive removal

- Manual inspection of all inconsistent and ¼ exact CCs
- Performed by researchers → ***C, IC***

Assessment of inconsistencies

- All inconsistent clone groups inspected
- Performed by developers → ***UIC, F***



Detection of Inconsistent Clones

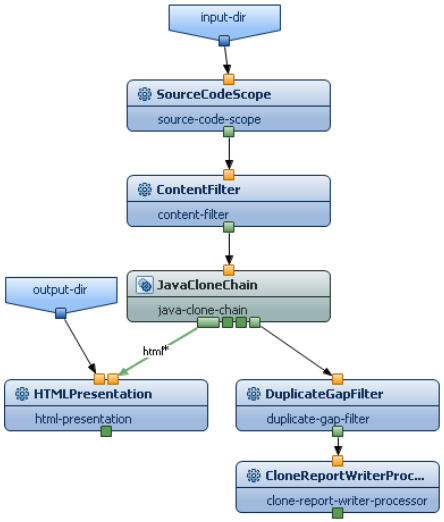
Approach

- 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

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

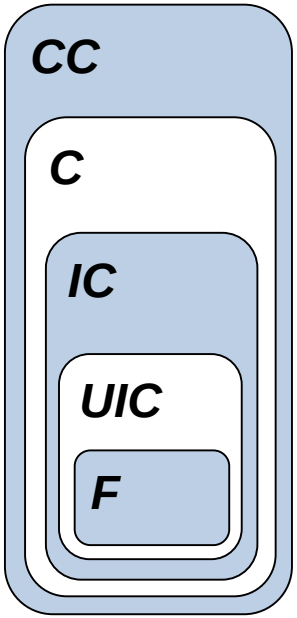
Munich Re International reinsurance company, 37.000 employees

LV 1871 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 |*C*|	286	160	326	352	303	**1427**
Inconsistent CGs |*IC*|	159	89	179	151	146	**724**
Unint. Incons. |*UIC*|	51	29	66	15	42	**203**
Faulty CGs |*F*|	19	18	42	5	23	**107**



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%**

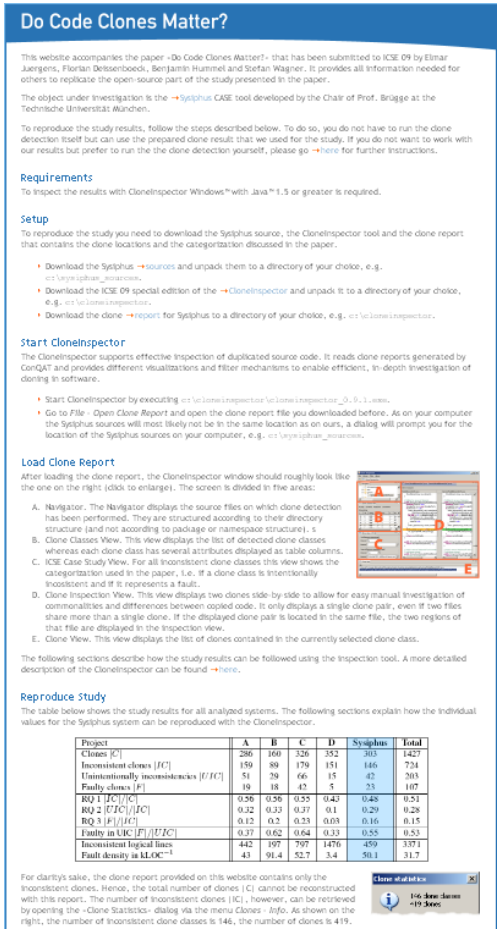
Threats to Validity

	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



Do Code Clones Matter?

This website accompanies the paper »Do Code Clones Matter?« that has been submitted to ICSE 09 by Elmar Juergens, Florian Deissenboeck, Benjamin Hummel and Stefan Wagner. It provides all information needed for others to replicate the open-source part of the study presented in the paper.

The object under investigation is the →Sysiphus CASE tool developed by the Chair of Prof. Brügge at the Technische Universität München.

To reproduce the study results, follow the steps described below. To do so, you do not have to run the clone detection itself but can use the prepared clone result that we used for the study. If you do not want to work with our results but prefer to run the the clone detection yourself, please go →here for further instructions.

Requirements

To inspect the results with CloneInspector Windows™ with Java™ 1.5 or greater is required.

Setup

To reproduce the study you need to download the Sysiphus source, the CloneInspector tool and the clone report that contains the clone locations and the categorization discussed in the paper.

- Download the Sysiphus →sources and unpack them to a directory of your choice, e.g. c:\sysiphus_sources.
- Download the ICSE 09 special edition of the →CloneInspector and unpack it to a directory of your choice, e.g. c:\cloneinspector.
- Download the clone →report for Sysiphus to a directory of your choice, e.g. c:\cloneinspector.

Start CloneInspector

The CloneInspector supports effective inspection of duplicated source code. It reads clone reports generated by ConQAT and provides different visualizations and filter mechanisms to enable efficient, in-depth investigation of cloning in software.

- Start CloneInspector by executing c:\cloneinspector\cloneinspector_0.9.1.exe.
- Go to *File - Open Clone Report* and open the clone report file you downloaded before. As on your computer the Sysiphus sources will most likely not be in the same location as on ours, a dialog will prompt you for the location of the Sysiphus sources on your computer, e.g. c:\sysiphus_sources.

Load Clone Report

After loading the clone report, the CloneInspector window should roughly look like the one on the right (click to enlarge). The screen is divided in five areas:

A. Navigator. The Navigator displays the source files on which clone detection has been performed. They are structured according to their directory structure (and not according to package or namespace structure). s
B. Clone Classes View. This view displays the list of detected clone classes whereas each clone class has several attributes displayed as table columns.
C. ICSE Case Study View. For all inconsistent clone classes this view shows the categorization used in the paper, i.e. if a clone class is intentionally inconsistent and if it represents a fault.
D. Clone Inspection View. This view displays two clones side-by-side to allow for easy manual investigation of commonalities and differences between copied code. It only displays a single clone pair, even if two files share more than a single clone. If the displayed clone pair is located in the same file, the two regions of that file are displayed in the inspection view.
E. Clone View. This view displays the list of clones contained in the currently selected clone class.

The following sections describe how the study results can be followed using the inspection tool. A more detailed description of the CloneInspector can be found →here.

Reproduce Study

The table below shows the study results for all analyzed systems. The following sections explain how the individual values for the Sysiphus system can be reproduced with the CloneInspector.

Project	A	B	C	D	Sysiphus	Total
Clones $\|C\|$	286	160	326	352	303	1427
Inconsistent clones $\|IC\|$	159	89	179	151	146	724
Unintentionally inconsistencies $\|UIC\|$	51	29	66	15	42	203
Faulty clones $\|F\|$	19	18	42	5	23	107
RQ 1 $\|IC\|/\|C\|$	0.56	0.56	0.55	0.43	0.48	0.51
RQ 2 $\|UIC\|/\|IC\|$	0.32	0.33	0.37	0.1	0.29	0.28
RQ 3 $\|F\|/\|IC\|$	0.12	0.2	0.23	0.03	0.16	0.15
Faulty in UIC $\|F\|/\|UIC\|$	0.37	0.62	0.64	0.33	0.55	0.53
Inconsistent logical lines	442	197	797	1476	459	3371
Fault density in kLOC^{-1}	43	91.4	52.7	3.4	50.1	31.7

For clarity's sake, the clone report provided on this website contains only the inconsistent clones. Hence, the total number of clones (C) cannot be reconstructed with this report. The number of inconsistent clones (IC), however, can be retrieved by opening the »Clone Statistics« dialog via the menu *Clones - Info*. As shown on the right, the number of inconsistent clone classes is 146, the number of clones is 419.

- **ConQAT**
  TECHNISCHE UNIVERSITÄT MÜNCHEN

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.