Evaluating Code Duplication Detection Techniques

Filip Van Rysselberghe and Serge Demeyer
Lab On Re-Engineering
University Of Antwerp

Towards a Taxonomy of Clones in Source Code: A Case Study

Cory J. Kapser and Michael W. Godfrey
Software Architecture Group
University of Waterloo

Duplicated Code (a.k.a. code clone)

- Code duplication occurs when developers systematically copy previously existing code which solved a problem similar to the one they are currently trying to solve.
- Typically 5% to 10% of code, up to 50%.
- Variety of reasons duplication occurs.

Associated Problems

- Errors can be difficult to fix.
- Change in requirements may be difficult to implement.
- Code size unnecessarily increased.
- Can lead to unused, dead code.
- Can be indicative of design problems.
- Bugs may be copied as well.

Evaluating Duplicated Code Detection Techniques

- Authors set out to evaluate the qualities of several clone detection techniques and determine where they fit best into the software maintenance process.
- Compares 3 representative techniques on 5 small to medium size cases.

Duplication Detection Techniques

- Authors suggest there are three groups of methods of detecting duplicated code:
  - String based
  - Token based
  - Parse-tree based

Research Structure

- Goal
- Questions
- Experimental Setup

Selected Cases

- ScoreMaster
- TextEdit
- Brahms
- Jmocha
- JavaParser of JMetric

Results: Portability

- Simple line matching most portable.
- Parameterized line matching and suffix tree matching are fairly portable.
- Metric based matching least portable.

Results: What Kind of Matches Found?

- Metrics based approach find function block duplication.
- Simple string matching finds equal lines.
- Parameterized line matching finds duplicated lines.
- Suffix tree matching finds duplicated series of tokens.

Results: Accuracy

- Number of false matches:
  - Parameterized suffix tree matching and simple line matching find no false matches.
  - Parameterized line matching finds few false matches.
  - Metrics based matching finds many false positives when applying metrics to block fragments, only a few when applying to methods.

Results: Accuracy

- Number of useless matches:
  - Both parameterized methods returned low amounts of useless matches.
  - Metrics found more useless matches, 133 out of 138 in TextEdit when applying metrics to methods.
  - Simple line matching finds many, 229 useless matches in TextEdit.

Results: Accuracy

- Number of recognizable matches
  - Metric fingerprints is very high.
  - Parameterized matching techniques return less recognizable matches.
  - Simple string match returns the lowest.

Results: Performance

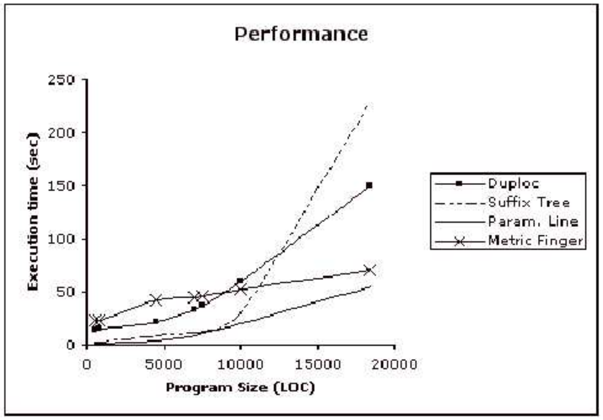


Figure 4. Performance of the different techniques

Conclusions

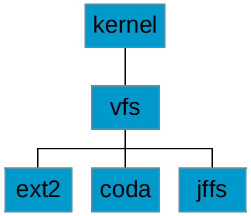
- Based on comparing the 3 representative duplication detection techniques, the following conclusions were drawn:
  - Simple line matching is suitable for problem detection and assessment.
  - Parameterized matching will work well with fine-grained refactoring tools.
  - Metric Fingerprints will work well with method level refactoring techniques.
- Have shown that each technique has specific advantages and disadvantages.
- Have laid the ground work for a systemic approach to detecting and removing clones.

Toward a Taxonomy of Clones

- Aim to profile cloning as it occurs in the real world and generate a taxonomy of types of code duplications.
- This will give us insight into how and why developers duplicate code, and aid the effort in developing clone detection techniques and tools.

The Study

- Performed on the Linux kernel file-system subsystem.
  - Consists of 538 .c and .h files, 279,118 LOC.
  - 42 file system implementations.
  - Layered design.



Study Methods

- Used parameterized string matching and metrics based detection to gather clones.
- Manually inspected clones returned from the detection tools and created the current taxonomy.
- Generated scripts to classify each clone into one of clone types, and again manually inspected these results.

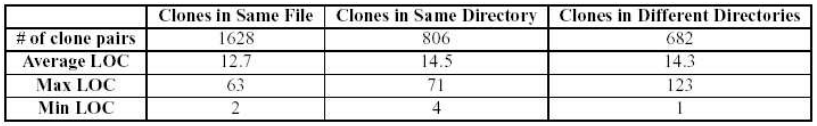
Taxonomy of Clones

- Duplicated blocks within the same function.
- Cloned blocks across functions, files and directories.
- Similar functions, same file.
- Functions cloned between files in the same directory.
- Functions cloned across directories.
- Cloned files.
- Initialization and finalization clones.

Results

- 12% of the Linux kernel file-system code is involved in code duplication.
- Detected 3116 clone pairs, with an average length is 13.5 lines.
- 78% of cloning occurs in the same directory.

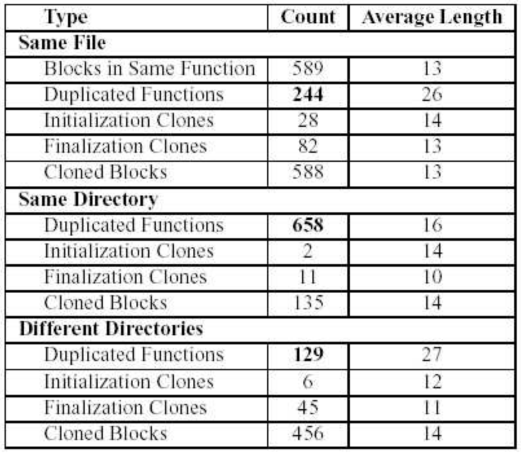
Locality of Clone Pairs



	Clones in Same File	Clones in Same Directory	Clones in Different Directories
# of clone pairs	1628	806	682
Average LOC	12.7	14.5	14.3
Max LOC	63	71	123
Min LOC	2	4	1

Table 1: Profiles of cloning locality — All clones

Frequency of Clone Types



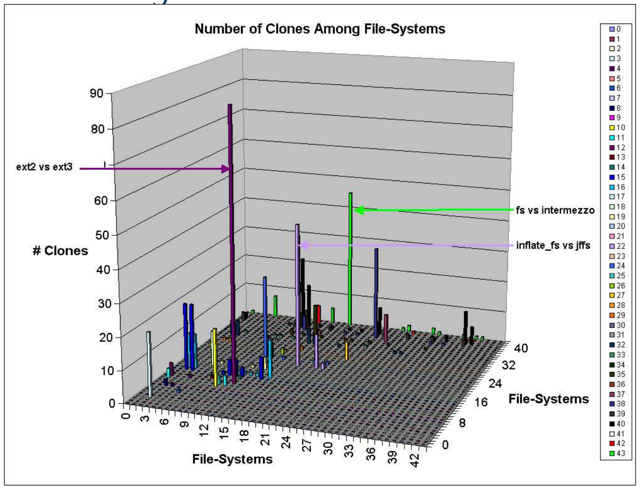
Type	Count	Average Length
Same File		
Blocks in Same Function	589	13
Duplicated Functions	**244**	26
Initialization Clones	28	14
Finalization Clones	82	13
Cloned Blocks	588	13
Same Directory		
Duplicated Functions	**658**	16
Initialization Clones	2	14
Finalization Clones	11	10
Cloned Blocks	135	14
Different Directories		
Duplicated Functions	**129**	27
Initialization Clones	6	12
Finalization Clones	45	11
Cloned Blocks	456	14

Table 2: Frequency of various clone categories — Parametric String Match

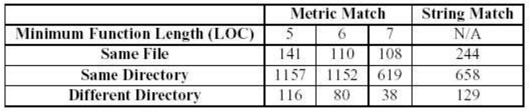
Families of File Systems

- ext2 and ext3 highly related.
- Intermezzo cloned much from the main file-system code and Coda.
- Jffs has cloned much from inflate_fs, most of the clones were put into 1 file.

Visualization of Cloning Without Showing Same Directory Clones

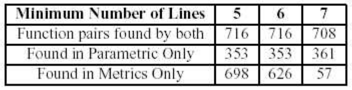


Metrics Vs. String Matching



	Metric Match			String Match
Minimum Function Length (LOC)	5	6	7	N/A
Same File	141	110	108	244
Same Directory	1157	1152	619	658
Different Directory	116	80	38	129

Table 3: Number of function clones found in metrics based clone detection and parameterized string match



Minimum Number of Lines	5	6	7
Function pairs found by both	716	716	708
Found in Parametric Only	353	353	361
Found in Metrics Only	698	626	57

Table 4: Comparison of # of function clones found by the two clone detection algorithms

Conclusions

- We have begun to build a taxonomy of code clones in software.
- Cloning activity in the Linux kernel file-system subsystem is at a non-trivial rate.
- Cloning most commonly occurs within a subsystem.
- Parameterized string matching provides an interesting and powerful method for function duplication detection.
- 3D visualization provided an interesting method of viewing clones amongst subsystems.

Importance of this Work

- Lots of clone detection methods out there, few comparisons.
- What we catch and what we miss is unclear.