Structured Review of the Evidence for Effects of Code Duplication on Software Quality

Wiebe Hordijk, María Laura Ponisio, Roel Wieringa
University of Twente, The Netherlands
hordijkwtb|m.l.ponisio|roelw@ewi.utwente.nl

Abstract

This report presents the detailed steps and results of a structured review of code clone literature. The aim of the review is to investigate the evidence for the claim that code duplication has a negative effect on code changeability. This report contains only the details of the review for which there is not enough place to include them in the companion paper published at a conference (Hordijk, Ponisio et al. 2009 - Harmfulness of Code Duplication - A Structured Review of the Evidence).

1 Introduction

Duplication of source code is an important factor that is suspected to affect the quality of systems in terms of changeability and the number of errors. We want to investigate how duplication affects quality. There is a vast body of research about code duplication, and in this review we aggregate the current knowledge about the effects of duplication on changeability and error levels.

1.1 Problems

There is a lot of literature about code duplication, but only a few studies have attempted to investigate if and how duplication actually has a negative effect on changeability and error levels. Therefore it is not known if duplication is harmful, and if so, under what circumstances. This is a problem for researchers because many investigations are based upon the assumption that clones are harmful, and if this assumption is false, the value of the research would be called into doubt.
For practitioners this lack of knowledge about harmfulness of duplication is a problem because they do not know if they should invest effort in avoiding or removing clones, and if so, how to prioritize those efforts between different kinds of clones. Based on practitioners' reports and our own experience, we see that little use is made of clone detectors in practice. We think that solid knowledge about the harmfulness of clones would make such tools more attractive to practitioners.

1.2 Contributions

This study is a structured review of the evidence in code clone literature for harmfulness of duplication. The contributions are in the conference paper.

2 Methodology

We gathered information only from primary research, not from empirical observations. We have followed a method described by Kitchenham's general procedure for performing systematic reviews Kitchenham 2007 - Procedures for Performing Systematic Reviews. Even though the entire investigation is not completely repeatable, as human judgment is involved in interpreting articles, Kitchenham's method makes steps of the process as repeatable as possible. The following sections summarize our steps.

2.1 Framing the Research Questions

To frame research questions in such a way that they can be used to drive a structured review, Kitchenham suggests to use the PICOC format which we apply here to our research.

- Population: our population consists of software systems.
- Intervention: the intervention is the presence of duplication in a system. This is more like a disease when compared to medical research than a cure; our research is not evaluating treatments, but investigating how bad the disease is.
- Comparison: we compare software systems with duplication against software systems without or with less duplication.
- Outcome: the outcome of duplication is a reduction of changeability of the software system, or such is the hypothesis.
- Context: the context in which the above hypothesis holds, consists of context factors that are as yet not well understood. They include the sizes of clones and the refactorability of clones. Some context factors are mentioned in the primary sources, and they are discussed in the conference paper.

2.2 Identification of research

We searched a number of literature sources with several search criteria, aimed at finding a set of articles with the most complete possible coverage of the field of code clones. We chose the criteria to reflect the research questions stated in §1.1. We searched the following databases: DBLP, ACM Portal, CiteSeer and Scopus with the following search terms: "code clone", "clones", "code | duplication" (because "duplication" yields too many false hits), on December 17, 2008. We discarded articles that were not about code clones; examples include compiler optimization, set theory and DNA research. After our extensive searches we have validated the completeness of the search actions by looking for references in the selected papers to other papers that were not present in our sample but that would pass our search criteria. We found only 2 such references, which were workshop papers. Altogether, this yielded 153 papers. To our knowledge we have thus exhausted all available evidence in the period under review.

2.3 Selection of primary studies

We applied the following criteria to the found sources for inclusion in this review.

- The article must be published in a journal or conference proceedings. This excludes drafts of articles and technical reports found on web sites of research groups.
- The article should present evidence for a causal relation between duplication and a quality attribute of the system, or between intermediate variables, e.g. between duplication and co-change. We judged this by reading the entire papers, not just the titles and abstracts, because sometimes evidence is stated in a case study which is used as an illustration of, for example, a clone detector, on which the paper focuses.
- The article should not be published before 1990. This boundary is chosen arbitrarily to limit the search for sources.

We have not applied quality criteria to the primary sources, because so few papers passed the selection criteria that no additional selection was needed. The resulting set contains 18 papers, which are discussed in section 3.

2.4 Aggregation of evidence

We analyzed the evidence in the included papers. When a claim was made, we analyzed the external validity, that is, for which situations the claim would hold. For example, if a paper draws conclusions from an experiment with one system, then those conclusions may not be valid in another system because of any kind of difference between those systems. However if conclusions are based on five different open source Java systems, and another paper draws the same conclusions from two other

open source Java systems, we may generalize the conclusions to the class of open source Java systems. An overview of the conclusions is in section 4. Since we are interested in the circumstances under which duplication is harmful, we also list what is known about the context factors under which these conclusions hold.

3 Article identification and selection

This section presents the results from steps 1, identification of research, and 2, selection of primary studies. We used the criteria specified in paragraph 2.3 to select and rate articles for inclusion in our review. The results of the searches are the combined papers listed below. Those papers that passed our inclusion criteria are listed under ‘Included’, the others are under ‘Not included’. The discarded papers are those that seemed to fit our criteria from looking at the title only, but were discarded after reading the paper itself. We have included the titles to be able to distinguish between papers from the same authors in the same year. Full details are given in the bibliography.

Included

1. Al-Ekram, Kapser et al. 2005 - Cloning by accident: An empirical study of source code cloning across software systems
2. Aversano, Cerulo et al. 2007 - How clones are maintained: An empirical study
3. Baker 1992 - A Program for Identifying Duplicated Code
4. Balazinska, Merlo et al. 1999 - Partial redesign of Java software systems based on clone analysis
5. Balint, Gîrba et al. 2006 - How developers copy
6. Geiger, Fluri et al. 2006 - Relation of Code Clones and Change Couplings
7. Jiang, Su et al. 2007 - Context-based detection of clone-related bugs
8. Jürgens, Hummel et al. 2008 - Static Bug Detection Through Analysis of Inconsistent Clones
9. Kamiya, Kusumoto et al. 2002 - CCFinder: A multilinguistic token-based code clone detection system for large scale source code
10. Kim, Sazawal et al. 2005 - An empirical study of code clone genealogies
11. Krinke 2007 - A Study of Consistent and Inconsistent Changes to Code Clones
12. Krinke 2008 - Is Cloned Code More Stable than Non-cloned Code?
13. LaToza, Venolia et al. 2006 - Maintaining mental models: A study of developer work habits
14. Li, Lu et al. 2004 - CP-Miner: a tool for finding copy-paste and related bugs in operating system code
15. Li, Lu et al. 2006 - CP-Miner: Finding copy-paste and related bugs in large-scale software code
16. Lozano, Wermelinger et al. 2007 - Evaluating the Harmfulness of Cloning: A Change Based Experiment
17. Monden, Nakae et al. 2002 - Software quality analysis by code clones in industrial legacy software
18. Toomim, Begel et al. 2004 - Managing Duplicated Code with Linked Editing

Not included

1. Adar and Kim 2007 - SoftGUESS: Visualization and exploration of code clones in context
2. Al-Ekram and Kontogiannis 2004 - Source code modularization using lattice of concept slices
3. Antoniol, Casazza et al. 2001 - Modeling clones evolution through time series
4. Antoniol, Villano et al. 2002 - Analyzing cloning evolution in the Linux kernel
5. Baker 1995 - On finding duplication and near-duplication in large software systems
6. Baker 1996 - Parameterized Pattern Matching: Algorithms and Applications
7. Baker 1997 - Parameterized duplication in strings: Algorithms and an application to software maintenance
8. Baker 2007 - Finding clones with dup: Analysis of an experiment
9. Baker and Manber 1998 - Deducing Similarities in Java Sources from Bytecodes
10. Bakota, Ferenc et al. 2007 - Clone Smells in Software Evolution
11. Balazinska, Merlo et al. 1999 - Measuring clone based reengineering opportunities
12. Balazinska, Merlo et al. 2000 - Advanced clone-analysis to support object-oriented system refactoring
13. Basit and Jarzabek 2005 - Detecting higher-level similarity patterns in programs
14. Basit and Jarzabek 2007 - Efficient token based clone detection with flexible tokenization
15. Basit, Rajapakse et al. 2005 - Beyond templates: A study of clones in the STL and some general implications
16. Basit, Rajapakse et al. 2005 - An Empirical Study on Limits of Clone Unification Using Generics
17. Baxter, Yahin et al. 1998 - Clone detection using abstract syntax trees
18. Bellon, Koschke et al. 2007 - Comparison and evaluation of clone detection tools

19. Beyer, Noack et al. 2003 - Simple and Efficient Relational Querying of Software Structures
20. Black, Schärli et al. 2003 - Applying traits to the smalltalk collection classes
21. Bodík, Gupta et al. 2004 - Complete removal of redundant expressions
22. Bouktif, Aritoniol et al. 2006 - A novel approach to optimize clone refactoring activity
23. Bruntink, Van Deursen et al. 2004 - An evaluation of clone detection techniques for identifying crosscutting concerns
24. Bruntink, van Deursen et al. 2005 - On the use of clone detection for identifying crosscutting concern code
25. Burd and Bailey 2002 - Evaluating Clone Detection Tools for Use during Preventative Maintenance
26. Burd and Munro 1997 - Investigating the Maintenance Implications of the Replication of Code
27. Canfora, Cerulo et al. 2006 - On the use of line co-change for identifying crosscutting concern code
28. Canfora, Cerulo et al. 2007 - Identifying changed source code lines from version repositories
29. Casazza, Antoniol et al. 2001 - Identifying Clones in the Linux Kernel
30. Church and Helfman 1993 - Dotplot: A Program for Exploring Self-Similarity in Millions of Lines for Text and Code
31. Clements, Kazman et al. 2002 - Evaluating Software Architectures: Methods and Case Studies
32. Cordy, Dean et al. 2004 - Practical Language-Independent Detection of Near-Miss Clones
33. Dagpinar and Jahnke 2003 - Predicting Maintainability with Object-Oriented Metrics - An Empirical Comparison
34. Davey, Barson et al. 1995 - The development of a software clone detector
35. De Lucia, Francese et al. 2004 - Reengineering web applications based on cloned pattern analysis
36. Deissenboeck, Hummel et al. 2008 - Clone detection in automotive model-based development
37. Di Lucca, Di Penta et al. 2002 - An approach to identify duplicated web pages
38. Di Penta 2005 - Evolution doctor: A framework to control software system evolution
39. Di Penta, Neteler et al. 2005 - A language-independent software renovation framework
40. Duala-Ekoko and Robillard 2007 - Tracking code clones in evolving software
41. Duala-Ekoko and Robillard 2008 - Clonetracker: tool support for code clone management
42. Ducasse, Nierstrasz et al. 2006 - On the effectiveness of clone detection by string matching
43. Ducasse, Rieger et al. 1999 - Language independent approach for detecting duplicated code
44. Ducasse, Rieger et al. 1999 - Tool Support for Refactoring Duplicated OO Code
45. Evans, Fraser et al. 2007 - Clone Detection via Structural Abstraction
46. Falke, Frenzel et al. 2008 - Empirical evaluation of clone detection using syntax suffix trees
47. Fanta and Rajlich 1999 - Removing Clones from the Code
48. Fioravanti, Migliarese et al. 2001 - Reengineering analysis of object-oriented systems via duplication analysis
49. Flores and Polo 2005 - Dynamic component assessment on PvC environments
50. Gabel, Jiang et al. 2008 - Scalable detection of semantic clones
51. Gallagher and Layman 2003 - Are Decomposition Slices Clones?
52. Giesecke 2006 - Generic modelling of code clones
53. Gitchell and Tran 1999 - Sim: a utility for detecting similarity in computer programs
54. Godfrey, Dong et al. 2004 - Four Interesting Ways in Which History Can Teach Us About Software
55. Godfrey and Zou 2005 - Using Origin Analysis to Detect Merging and Splitting of Source Code Entities
56. Guo and Zou 2008 - Detecting Clones in Business Applications
57. Hayase, Lee et al. 2008 - A criterion for filtering code clone related bugs
58. Higo 2006 - Code Clone Analysis Methods for Efficient Software Maintenance
59. Higo, Kamiya et al. 2004 - Aries: Refactoring support environment based on code clone analysis
60. Higo, Kamiya et al. 2005 - ARIES: refactoring support tool for code clone
61. Higo, Kamiya et al. 2007 - Method and implementation for investigating code clones in a software system
62. Higo, Ueda et al. 2002 - On software maintenance process improvement based on code clone analysis
63. Higo, Ueda et al. 2007 - Simultaneous Modification Support based on Code Clone Analysis
64. Hill and Rideout 2004 - Automatic method completion

65. Hordijk, Ponisio et al. 2008 - Structured Review of Code Clone Literature
66. Imai, Kataoka et al. 2002 - Evaluating software maintenance cost using functional redundancy metrics
67. Jablonski and Hou 2007 - CReN: a tool for tracking copy-and-paste code clones and renaming identifiers consistently in the IDE
68. Jarzabek and Li 2006 - Unifying clones with a generative programming technique: A case study
69. Jiang, Misherghi et al. 2007 - DECKARD: Scalable and accurate tree-based detection of code clones
70. Jiang and Hassan 2007 - A Framework for Studying Clones In Large Software Systems
71. Johnson 1993 - Identifying redundancy in source code using fingerprints
72. Johnson 1994 - Substring Matching for Clone Detection and Change Tracking
73. Johnson 1994 - Visualizing Textual Redundancy in Legacy Source
74. Juillerat and Hirsbrunner 2006 - An Algorithm for Detecting and Removing Clones in Java Code
75. Kamiya 2008 - Variation analysis of context-sharing identifiers with code clones
76. Kamiya, Ohata et al. 2001 - Maintenance support tools for JAVA programs: CCFinder and JAAT
77. Kapser and Godfrey 2003 - A taxonomy of clones in source code: The re-engineers most wanted list
78. Kapser and Godfrey 2003 - Toward a Taxonomy of Clones in Source Code: A Case Study
79. Kapser and Godfrey 2004 - Aiding comprehension of cloning through categorization
80. Kapser and Godfrey 2005 - Improved tool support for the investigation of duplication in software
81. Kapser and Godfrey 2006 - "Cloning Considered Harmful" Considered Harmful
82. Kapser and Godfrey 2008 - "Cloning considered harmful" considered harmful: patterns of cloning in software
83. Kapser and Godfrey 2006 - Supporting the analysis of clones in software systems: A case study
84. Kataoka, Ernst et al. 2001 - Automated Support for Program Refactoring Using Invariants
85. Kim 2007 - Understanding and Aiding Code Evolution by Inferring Change Patterns
86. Kim, Bergman et al. 2004 - An ethnographic study of copy and paste programming practices in OOPL
87. Kim and Notkin 2005 - Using a clone genealogy extractor for understanding and supporting evolution of code clones
88. Kim and Notkin 2006 - Program element matching for multi-version program analyses
89. Kim, Notkin et al. 2007 - Automatic Inference of Structural Changes for Matching across Program Versions
90. Komondoor and Horwitz 2001 - Tool Demonstration: Finding Duplicated Code Using Program Dependences
91. Komondoor and Horwitz 2001 - Using Slicing to Identify Duplication in Source Code
92. Komondoor 2003 - Automated duplicated code detection and procedure extraction
93. Kontogiannis 1997 - Evaluation Experiments on the Detection of Programming Patterns Using Software Metrics
94. Kontogiannis, Demori et al. 1996 - Pattern matching for clone and concept detection
95. Koschke 2006 - Survey of Research on Software Clones
96. Koschke, Falke et al. 2006 - Clone detection using abstract syntax suffix trees
97. Krinke 2001 - Identifying Similar Code with Program Dependence Graphs
98. Lague, Proulx et al. 1997 - Assessing the benefits of incorporating function clone detection in a development process
99. Lanubile and Mallardo 2003 - Finding function clones in Web applications
100. Lee and Jeong 2005 - SDD: high performance code clone detection system for large scale source code
101. Liu, Ma et al. 2006 - Detecting Duplications in Sequence Diagrams Based on Suffix Trees
102. Livieri, Higo et al. 2007 - Analysis of the Linux kernel evolution using code clone coverage
103. Livieri, Higo et al. 2007 - Very-large scale code clone analysis and visualization of open source programs using distributed CCFinder: D-CCFinder
104. Lozano 2008 - A methodology to assess the impact of source code flaws in changeability, and its application to clones

105. Ma and Woo 2007 - Applying a Code Clone Detection Method to Domain Analysis of Device Drivers
106. Marcus and Maletic 2001 - Identification of High-Level Concept Clones in Source Code
107. Mayrand, Leblanc et al. 1996 - Experiment on the automatic detection of function clones in a software system using metrics
108. Mende, Beckwermert et al. 2008 - Supporting the Grow-and-Prune Model in Software Product Lines Evolution Using Clone Detection
109. Mens, Tourwe et al. 2003 - Beyond the refactoring browser: Advanced tool support for software refactoring
110. Merlo, Dagenais et al. 2002 - Investigating large software system evolution: The Linux kernel
111. Rajapakse and Jarzabek 2005 - An Investigation of Cloning in Web Applications
112. Rajapakse and Jarzabek 2007 - Using server pages to unify clones in web applications: A trade-off analysis
113. Rieger, Ducasse et al. 2004 - Insights into system-wide code duplication
114. Roy and Cordy 2007 - A Survey on Software Clone Detection Research
115. Roy and Cordy 2008 - An Empirical Study of Function Clones in Open Source Software
116. Roy and Cordy 2008 - NICAD: Accurate Detection of Near-Miss Intentional Clones Using Flexible Pretty-Printing and Code Normalization
117. Roy and Cordy 2008 - Scenario-Based Comparison of Clone Detection Techniques
118. Roy and Cordy 2008 - Towards a mutation-based automatic framework for evaluating code clone detection tools
119. Rysselberghe and Demeyer 2003 - Evaluating Clone Detection Techniques
120. Shepherd, Pollock et al. 2007 - Case study: supplementing program analysis with natural language analysis to improve a reverse engineering task
121. Sutton, Kagdi et al. 2005 - Hybridizing evolutionary algorithms and clustering algorithms to find source-code clones
122. Tairas 2006 - Clone detection and refactoring
123. Tairas and Gray 2006 - Phoenix-based clone detection using suffix trees
124. Tairas and Gray 2008 - An information retrieval process to aid in the analysis of code clones
125. Tairas, Gray et al. 2006 - Visualization of clone detection results
126. Tonella, Antoniol et al. 2000 - Reverse engineering 4.7 million lines of code
127. Uchida, Monden et al. 2005 - Software analysis by code clones in open source software
128. Ueda, Ueda et al. 2002 - On detection of gapped code clones using gap locations
129. Wahler, Wahler et al. 2004 - Clone detection in source code by frequent itemset techniques
130. Walenstein 2006 - Code Clones: Reconsidering Terminology
131. Walenstein, Jyoti et al. 2003 - Problems creating task-relevant clone detection reference data
132. Wettel and Marinescu 2005 - Archeology of code duplication: Recovering duplication chains from small duplication fragments
133. Yamamoto, Matsushita et al. 2007 - Similarity of software system and its measurement tool SMMT
134. Yoshida, Higo et al. 2005 - On Refactoring Support Based on Code Clone Dependency Relation
135. Yu and Ramaswamy 2008 - Improving modularity by refactoring code clones: a feasibility study on Linux

Bibliography

This extended bibliography contains all papers included in and discarded from the review. The papers that were included are annotated with short descriptions after the dash. Discarded papers have a short explanation why they were discarded. Some references Wieringa 1996 - Requirements Engineering: Frameworks for Understanding; Fowler 1999 - Refactoring - Improving the Design of Existing Code; Kitchenham 2007 - Procedures for Performing Systematic Reviews are not papers included in the review, but methodological background about how we performed the review; those have not been annotated.

Adar, E. and M. Kim (2007). SoftGUESS: Visualization and exploration of code clones in context. International Conference on Software Engineering, Minneapolis, MN.

Al-Ekram, R., C. Kapser, et al. (2005). Cloning by accident: An empirical study of source code cloning across software systems. International Symposium on Empirical Software Engineering.

Al-Ekram, R. and K. Kontogiannis (2004). Source code modularization using lattice of concept slices. European Conference on Software Maintenance and Reengineering, CSMR, Tampere.

Antoniol, G., G. Casazza, et al. (2001). Modeling clones evolution through time series. Conference on Software Maintenance, Florence.

Antoniol, G., U. Villano, et al. (2002). "Analyzing cloning evolution in the Linux kernel." Information and Software Technology **44**(13): 755-765.

Aversano, L., L. Cerulo, et al. (2007). How clones are maintained: An empirical study. European Conference on Software Maintenance and Reengineering, Amsterdam.

Baker, B. S. (1992). "A Program for Identifying Duplicated Code." Computing Science and Statistics **24**: 49-57.

Baker, B. S. (1995). On finding duplication and near-duplication in large software systems. 2nd Working Conference on Reverse Engineering, Toronto, Ont, Can, IEEE.

Baker, B. S. (1996). "Parameterized Pattern Matching: Algorithms and Applications." Journal Computer System Science **52**(1): 28-42.

Baker, B. S. (1997). "Parameterized duplication in strings: Algorithms and an application to software maintenance." SIAM Journal on Computing **26**(5): 1343-1362.

Baker, B. S. (2007). "Finding clones with dup: Analysis of an experiment." IEEE Transactions on Software Engineering **33**(9): 608-621.

Baker, B. S. and U. Manber (1998). "Deducing Similarities in Java Sources from Bytecodes." Proc. of Usenix Annual Technical Conf.: 179-190.

Bakota, T., R. Ferenc, et al. (2007). "Clone Smells in Software Evolution." ICSM: 24-33.

Balazinska, M., E. Merlo, et al. (1999). Measuring clone based reengineering opportunities. International Software Metrics Symposium, Boca Raton, FL, USA, IEEE.

Balazinska, M., E. Merlo, et al. (1999). Partial redesign of Java software systems based on clone analysis. 6th Working Conference on Reverse Engineering, Atlanta, GA, USA, IEEE.

Balazinska, M., E. Merlo, et al. (2000). Advanced clone-analysis to support object-oriented system refactoring. Working Conference on Reverse Engineering.

Balint, M., T. Gîrba, et al. (2006). How developers copy. 14th IEEE International Conference on Program Comprehension, Athens.

Basit, H. A. and S. Jarzabek (2005). Detecting higher-level similarity patterns in programs. European Software Engineering Conference.

Basit, H. A. and S. Jarzabek (2007). "Efficient token based clone detection with flexible tokenization." ESEC/SIGSOFT FSE: 513-516.

Basit, H. A., D. C. Rajapakse, et al. (2005). Beyond templates: A study of clones in the STL and some general implications. 27th International Conference on Software Engineering, St. Louis, MO.

Basit, H. A., D. C. Rajapakse, et al. (2005). "An Empirical Study on Limits of Clone Unification Using Generics." SEKE: 109-114.

Baxter, I. D., A. Yahin, et al. (1998). Clone detection using abstract syntax trees. Conference on Software Maintenance, Bethesda, MD, USA, IEEE.

Bellon, S., R. Koschke, et al. (2007). "Comparison and evaluation of clone detection tools." IEEE Transactions on Software Engineering **33**(9): 577-591.

Beyer, D., A. Noack, et al. (2003). Simple and Efficient Relational Querying of Software Structures. 10th Working Conference on Reverse Engineering, Victoria, BC.

Black, A. P., N. Schärli, et al. (2003). "Applying traits to the smalltalk collection classes." ACM SIGPLAN Notices **38**(11): 47-64.

Bodík, R., R. Gupta, et al. (2004). "Complete removal of redundant expressions." ACM SIGPLAN Notices **39**(4): 596-597.

Bouktif, S., G. Aritoniol, et al. (2006). A novel approach to optimize clone refactoring activity. GECCO 2006 - Genetic and Evolutionary Computation Conference, Seattle, WA.

Bruntink, M., A. Van Deursen, et al. (2004). An evaluation of clone detection techniques for identifying crosscutting concerns. IEEE International Conference on Software Maintenance, ICSM, Chicago, IL.
Bruntink, M., A. van Deursen, et al. (2005). "On the use of clone detection for identifying crosscutting concern code." IEEE Transactions on Software Engineering **31**(10): 804-818.
Burd, E. and J. Bailey (2002). Evaluating Clone Detection Tools for Use during Preventative Maintenance. 2nd IEEE International Workshop on Source Code Analysis and Manipulation (SCAM'02), IEEE Computer Society.
Burd, E. and M. Munro (1997). Investigating the Maintenance Implications of the Replication of Code. International Conference on Software Maintenance.
Canfora, G., L. Cerulo, et al. (2006). On the use of line co-change for identifying crosscutting concern code. IEEE International Conference on Software Maintenance, ICSM, Philadelphia, PA.
Canfora, G., L. Cerulo, et al. (2007). Identifying changed source code lines from version repositories. Fourth International Workshop on Mining Software Repositories, MSR Minneapolis, MN.
Casazza, G., G. Antoniol, et al. (2001). Identifying Clones in the Linux Kernel. International Workshop on Source Code Analysis and Manipulation.
Church, K. W. and J. I. Helfman (1993). "Dotplot: A Program for Exploring Self-Similarity in Millions of Lines for Text and Code." American Statistical Association, Institue for Mathematical Statistics and Interface Foundations of North America **2**(2): 153-174.
Clements, P., R. Kazman, et al. (2002). Evaluating Software Architectures: Methods and Case Studies, Addison-Wesley Professional.
Cordy, J. R., T. R. Dean, et al. (2004). Practical Language-Independent Detection of Near-Miss Clones. Conference of the Centre for Advanced Studies on Collaborative research, Ontario, Canada.
Dagpinar, M. and J. H. Jahnke (2003). Predicting Maintainability with Object-Oriented Metrics - An Empirical Comparison. Working Conference on Reverse Engineering, Victoria, BC.
Davey, N., P. C. Barson, et al. (1995). "The development of a software clone detector." International Journal of Applied Software Technology **1**(3-4): 219-36.
De Lucia, A., R. Francese, et al. (2004). Reengineering web applications based on cloned pattern analysis. 12th IEEE International Workshops on Program Comprehension, Bari.
Deissenboeck, F., B. Hummel, et al. (2008). "Clone detection in automotive model-based development." ICSE: 603-612.
Di Lucca, G. A., M. Di Penta, et al. (2002). An approach to identify duplicated web pages. IEEE Computer Society's International Computer Software and Applications Conference, Oxford.
Di Penta, M. (2005). Evolution doctor: A framework to control software system evolution. European Conference on Software Maintenance and Reengineering, CSMR, Manchester.
Di Penta, M., M. Neteler, et al. (2005). "A language-independent software renovation framework." Journal of Systems and Software **77**(3): 225-240.
Duala-Ekoko, E. and M. P. Robillard (2007). Tracking code clones in evolving software. International Conference on Software Engineering, Minneapolis, MN.
Duala-Ekoko, E. and M. P. Robillard (2008). "Clonetracker: tool support for code clone management." ICSE: 843-846.
Ducasse, S., O. Nierstrasz, et al. (2006). "On the effectiveness of clone detection by string matching." Journal of Software Maintenance and Evolution **18**(1): 37-58.
Ducasse, S., M. Rieger, et al. (1999). Language independent approach for detecting duplicated code. Conference on Software Maintenance, Oxford, UK, IEEE.
Ducasse, S. e., M. Rieger, et al. (1999). Tool Support for Refactoring Duplicated OO Code. ECOOP'99 Workshop on Experiences in Object-Oriented Re-Engineering, Forschungszentrum Informatik, Karlsruhe.
Evans, W. S., C. W. Fraser, et al. (2007). "Clone Detection via Structural Abstraction." WCRE: 150-159.
Falke, R., P. Frenzel, et al. (2008). "Empirical evaluation of clone detection using syntax suffix trees." Empirical Software Engineering **13**(6): 601-643.
Fanta, R. and V. Rajlich (1999). "Removing Clones from the Code." Journal of Software Maintenance and Evolution **11**(4): 223-243.
Fioravanti, F., G. Migliarese, et al. (2001). Reengineering analysis of object-oriented systems via duplication analysis. International Conference on Software Engineering, Toronto, Ont.
Flores, A. and M. Polo (2005). Dynamic component assessment on PvC environments. IEEE Symposium on Computers and Communications, Murcia.
Fowler, M. (1999). Refactoring - Improving the Design of Existing Code, Addison-Wesley.
Gabel, M., L. Jiang, et al. (2008). "Scalable detection of semantic clones." ICSE: 321-330.
Gallagher, K. and L. Layman (2003). "Are Decomposition Slices Clones?" IWPC: 251-.
Geiger, R., B. Fluri, et al. (2006). Relation of Code Clones and Change Couplings. Fundamental Approaches to Software Engineering.
Giesecke, S. (2006). "Generic modelling of code clones." Duplication, Redundancy, and Similarity in Software.
Gitchell, D. and N. Tran (1999). "Sim: a utility for detecting similarity in computer programs." SIGCSE Bull. **31**(1): 266-270.
Godfrey, M., X. Dong, et al. (2004). Four Interesting Ways in Which History Can Teach Us About Software. International Workshop on Mining Software Repositories (MSR-04), Edinburgh, Scotland.
Godfrey, M. W. and L. Zou (2005). "Using Origin Analysis to Detect Merging and Splitting of Source Code Entities." IEEE Transactions on Software Engineering **31**(2): 166-181.

Guo, J. and Y. Zou (2008). "Detecting Clones in Business Applications." WCRE: 91-100.
Hayase, Y., Y. Y. Lee, et al. (2008). "A criterion for filtering code clone related bugs." DEFECTS: 37-38.
Higo, Y. (2006). Code Clone Analysis Methods for Efficient Software Maintenance. Graduate School of Information Science and Technology, Osaka University.
Higo, Y., T. Kamiya, et al. (2004). Aries: Refactoring support environment based on code clone analysis. 8th IASTED International Conference on Software Engineering and Applications, Cambridge, MA.
Higo, Y., T. Kamiya, et al. (2005). ARIES: refactoring support tool for code clone. 3rd workshop on Software quality, ICSE, St. Louis, Missouri, ACM Press.
Higo, Y., T. Kamiya, et al. (2007). "Method and implementation for investigating code clones in a software system." Information and Software Technology **49**(9-10): 985-998.
Higo, Y., Y. Ueda, et al. (2002). On software maintenance process improvement based on code clone analysis. 4th International Conference on Product Focused Software Process Improvement, Springer-Verlag.
Higo, Y., Y. Ueda, et al. (2007). "Simultaneous Modification Support based on Code Clone Analysis." APSEC: 262-269.
Hill, R. and J. Rideout (2004). Automatic method completion. 19th International Conference on Automated Software Engineering, ASE, Linz.
Hordijk, W., M. L. Ponisio, et al. (2008). Structured Review of Code Clone Literature, University of Twente, The Netherlands.
Hordijk, W., M. L. Ponisio, et al. (2009). Harmfulness of Code Duplication - A Structured Review of the Evidence. 13th International Conference on Evaluation and Assessment in Software Engineering, Durham, UK.
Imai, T., Y. Kataoka, et al. (2002). Evaluating software maintenance cost using functional redundancy metrics. IEEE Computer Society's International Computer Software and Applications Conference, Oxford.
Jablonski, P. and D. Hou (2007). "CReN: a tool for tracking copy-and-paste code clones and renaming identifiers consistently in the IDE." ETX: 16-20.
Jarzabek, S. and S. Li (2006). "Unifying clones with a generative programming technique: A case study." Journal of Software Maintenance and Evolution **18**(4): 267-292.
Jiang, L., G. Misherghi, et al. (2007). DECKARD: Scalable and accurate tree-based detection of code clones. International Conference on Software Engineering, Minneapolis, MN.
Jiang, L., Z. Su, et al. (2007). "Context-based detection of clone-related bugs." ESEC-FSE '07: Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering: 55-64.
Jiang, Z. M. and A. E. Hassan (2007). "A Framework for Studying Clones In Large Software Systems." SCAM: 203-212.
Johnson, J. H. (1993). Identifying redundancy in source code using fingerprints. Conference of the Centre for Advanced Studies on Collaborative research: software engineering, Toronto, Ontario, Canada, IBM Press.
Johnson, J. H. (1994). Substring Matching for Clone Detection and Change Tracking. International Conference on Software Maintenance (ICSM1 '94).
Johnson, J. H. (1994). Visualizing Textual Redundancy in Legacy Source. Conference of the Centre for Advanced Studies on Collaborative research.
Juillerat, N. and B. Hirsbrunner (2006). An Algorithm for Detecting and Removing Clones in Java Code. 3rd Workshop on Software Evolution through Transformations.
Jürgens, E., B. Hummel, et al. (2008). "Static Bug Detection Through Analysis of Inconsistent Clones." Software Engineering (Workshops): 443-446.
Kamiya, T. (2008). "Variation analysis of context-sharing identifiers with code clones." ICSM: 464-465.
Kamiya, T., S. Kusumoto, et al. (2002). "CCFinder: A multilinguistic token-based code clone detection system for large scale source code." IEEE Transactions on Software Engineering **28**(7): 654-670.
Kamiya, T., F. Ohata, et al. (2001). Maintenance support tools for JAVA programs: CCFinder and JAAT. International Conference on Software Engineering, Toronto, Ont.
Kapser, C. and M. Godfrey (2003). A taxonomy of clones in source code: The re-engineers most wanted list. 2nd International Workshop on Detection of Software Clones.
Kapser, C. and M. Godfrey (2003). Toward a Taxonomy of Clones in Source Code: A Case Study. Evolution of Large-scale Industrial Software Applications (ELISA), Amsterdam.
Kapser, C. and M. W. Godfrey (2004). Aiding comprehension of cloning through categorization. International Workshop on Principles of Software Evolution (IWPSE), Kyoto.
Kapser, C. and M. W. Godfrey (2005). Improved tool support for the investigation of duplication in software. IEEE International Conference on Software Maintenance, ICSM, Budapest.
Kapser, C. and M. W. Godfrey (2006). "Cloning Considered Harmful" Considered Harmful. 13th Working Conference on Reverse Engineering, IEEE Computer Society.
Kapser, C. and M. W. Godfrey (2008). ""Cloning considered harmful" considered harmful: patterns of cloning in software." Empirical Software Engineering **13**(6): 645-692.
Kapser, C. J. and M. W. Godfrey (2006). "Supporting the analysis of clones in software systems: A case study." Journal of Software Maintenance and Evolution **18**(2): 61-82.
Kataoka, Y., M. D. Ernst, et al. (2001). Automated Support for Program Refactoring Using Invariants. International Conference on Software Maintenance.
Kim, M. (2007). Understanding and Aiding Code Evolution by Inferring Change Patterns. 29th International Conference on Software Engineering, IEEE Computer Society.

Kim, M., L. Bergman, et al. (2004). An ethnographic study of copy and paste programming practices in OOPL. International Symposium on Empirical Software Engineering.
Kim, M. and D. Notkin (2005). "Using a clone genealogy extractor for understanding and supporting evolution of code clones." ACM SIGSOFT Software Engineering Notes **30**(4): 1-5.
Kim, M. and D. Notkin (2006). Program element matching for multi-version program analyses. International workshop on Mining software repositories, Shanghai, China, ACM Press.
Kim, M., D. Notkin, et al. (2007). Automatic Inference of Structural Changes for Matching across Program Versions. International Conference on Software Engineering, IEEE Computer Society.
Kim, M., V. Sazawal, et al. (2005). An empirical study of code clone genealogies. 10th European Software Engineering Conference.
Kitchenham, B. (2007). Procedures for Performing Systematic Reviews, University of Durham, UK.
Komondoor, R. and S. Horwitz (2001). "Tool Demonstration: Finding Duplicated Code Using Program Dependences." Lecture Notes in Computer Science **2028**: 383-??
Komondoor, R. and S. Horwitz (2001). Using Slicing to Identify Duplication in Source Code. 8th International Symposium on Static Analysis.
Komondoor, R. V. (2003). Automated duplicated code detection and procedure extraction, The University of Wisconsin - Madison.
Kontogiannis, K. (1997). Evaluation Experiments on the Detection of Programming Patterns Using Software Metrics. 4th Working Conference on Reverse Engineering, IEEE Computer Society.
Kontogiannis, K. A., R. Demori, et al. (1996). Pattern matching for clone and concept detection. Reverse engineering, Kluwer Academic Publishers. **3:** 77-108.
Koschke, R. (2006). "Survey of Research on Software Clones." Duplication, Redundancy, and Similarity in Software.
Koschke, R., R. Falke, et al. (2006). Clone detection using abstract syntax suffix trees. Working Conference on Reverse Engineering.
Krinke, J. (2001). "Identifying Similar Code with Program Dependence Graphs." Proc. Eigth Working Conference on Reverse Engineering: 301-309.
Krinke, J. (2007). "A Study of Consistent and Inconsistent Changes to Code Clones." WCRE '07: Proceedings of the 14th Working Conference on Reverse Engineering: 170-178.
Krinke, J. (2008). "Is Cloned Code More Stable than Non-cloned Code?" Source Code Analysis and Manipulation, 2008 Eighth IEEE International Working Conference on: 57-66.
Lague, B., D. Proulx, et al. (1997). Assessing the benefits of incorporating function clone detection in a development process. Conference on Software Maintenance, Bari, Italy, IEEE.
Lanubile, F. and T. Mallardo (2003). Finding function clones in Web applications. Seventh European Conference on Software Maintenance and Reengineering.
LaToza, T. D., G. Venolia, et al. (2006). Maintaining mental models: A study of developer work habits. International Conference on Software Engineering, Shanghai.
Lee, S. and I. Jeong (2005). "SDD: high performance code clone detection system for large scale source code." OOPSLA Companion: 140-141.
Li, Z., S. Lu, et al. (2004). "CP-Miner: a tool for finding copy-paste and related bugs in operating system code." OSDI'04: Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation: 20-20.
Li, Z., S. Lu, et al. (2006). "CP-Miner: Finding copy-paste and related bugs in large-scale software code." IEEE Transactions on Software Engineering **32**(3): 176-192.
Liu, H., Z. Ma, et al. (2006). "Detecting Duplications in Sequence Diagrams Based on Suffix Trees." APSEC: 269-276.
Livieri, S., Y. Higo, et al. (2007). Analysis of the Linux kernel evolution using code clone coverage. Fourth International Workshop on Mining Software Repositories, MSR, Minneapolis, MN.
Livieri, S., Y. Higo, et al. (2007). Very-large scale code clone analysis and visualization of open source programs using distributed CCFinder: D-CCFinder. International Conference on Software Engineering, Minneapolis, MN.
Lozano, A. (2008). "A methodology to assess the impact of source code flaws in changeability, and its application to clones." ICSM: 424-427.
Lozano, A., M. Wermelinger, et al. (2007). Evaluating the Harmfulness of Cloning: A Change Based Experiment. Fourth International Workshop on Mining Software Repositories, IEEE Computer Society.
Ma, Y.-S. and D.-K. Woo (2007). "Applying a Code Clone Detection Method to Domain Analysis of Device Drivers." APSEC: 254-261.
Marcus, A. and J. Maletic (2001). Identification of High-Level Concept Clones in Source Code. 16th IEEE international conference on Automated software engineering.
Mayrand, J., C. Leblanc, et al. (1996). Experiment on the automatic detection of function clones in a software system using metrics. Conference on Software Maintenance, Monterey, CA, USA, IEEE.
Mende, T., F. Beckwermert, et al. (2008). "Supporting the Grow-and-Prune Model in Software Product Lines Evolution Using Clone Detection." CSMR: 163-172.
Mens, T., T. Tourwe, et al. (2003). Beyond the refactoring browser: Advanced tool support for software refactoring. International Workshop on Principles of Software Evolution IWPSE.
Merlo, E., M. Dagenais, et al. (2002). Investigating large software system evolution: The Linux kernel. IEEE Computer Society's International Computer Software and Applications Conference, Oxford.

Monden, A., D. Nakae, et al. (2002). Software quality analysis by code clones in industrial legacy software. Eighth IEEE Symposium on Software Metrics.
Rajapakse, D. C. and S. Jarzabek (2005). "An Investigation of Cloning in Web Applications." ICWE: 252-262.
Rajapakse, D. C. and S. Jarzabek (2007). Using server pages to unify clones in web applications: A trade-off analysis. International Conference on Software Engineering, Minneapolis, MN.
Rieger, M., S. Ducasse, et al. (2004). Insights into system-wide code duplication. Working Conference on Reverse Engineering, WCRE, Delft.
Roy, C. K. and J. R. Cordy (2007). A Survey on Software Clone Detection Research, Queen's University at Kingston, Ontario, Canada.
Roy, C. K. and J. R. Cordy (2008). "An Empirical Study of Function Clones in Open Source Software." WCRE: 81-90.
Roy, C. K. and J. R. Cordy (2008). "NICAD: Accurate Detection of Near-Miss Intentional Clones Using Flexible Pretty-Printing and Code Normalization." ICPC '08: Proceedings of the 2008 The 16th IEEE International Conference on Program Comprehension: 172-181.
Roy, C. K. and J. R. Cordy (2008). "Scenario-Based Comparison of Clone Detection Techniques." ICPC: 153-162.
Roy, C. K. and J. R. Cordy (2008). "Towards a mutation-based automatic framework for evaluating code clone detection tools." C3S2E: 137-140.
Rysselberghe, F. V. and S. Demeyer (2003). Evaluating Clone Detection Techniques. International Workshop on Evolution of Large Scale Industrial Software Applications.
Shepherd, D., L. Pollock, et al. (2007). Case study: supplementing program analysis with natural language analysis to improve a reverse engineering task. 7th ACM SIGPLAN-SIGSOFT workshop on Program analysis for software tools and engineering, San Diego, California, USA, ACM Press.
Sutton, A., H. H. Kagdi, et al. (2005). "Hybridizing evolutionary algorithms and clustering algorithms to find source-code clones." GECCO: 1079-1080.
Tairas, R. (2006). Clone detection and refactoring. Conference on Object-Oriented Programming Systems, Languages, and Applications, OOPSLA, Portland, OR.
Tairas, R. and J. Gray (2006). Phoenix-based clone detection using suffix trees. 44th annual Southeast regional conference, Melbourne, Florida, ACM.
Tairas, R. and J. Gray (2008). "An information retrieval process to aid in the analysis of code clones." Empirical Software Engineering **14**(1): 33-56.
Tairas, R., J. Gray, et al. (2006). Visualization of clone detection results. OOPSLA Workshop on Eclipse Technology eXchange, ETX, Portland, OR.
Tonella, P., G. Antoniol, et al. (2000). "Reverse engineering 4.7 million lines of code." Software - Practice and Experience **30**(2): 129-150.
Toomim, M., A. Begel, et al. (2004). Managing Duplicated Code with Linked Editing. Symposium on Visual Languages - Human Centric Computing, VLHCC, IEEE Computer Society.
Uchida, S., A. Monden, et al. (2005). "Software analysis by code clones in open source software." Journal of Computer Information Systems **45**(3): 1-11.
Ueda, Y., Y. Ueda, et al. (2002). On detection of gapped code clones using gap locations. Software Engineering Conference, 2002. Ninth Asia-Pacific.
Wahler, V., V. Wahler, et al. (2004). Clone detection in source code by frequent itemset techniques. Fourth IEEE International Workshop on Source Code Analysis and Manipulation.
Walenstein, A. (2006). "Code Clones: Reconsidering Terminology." Duplication, Redundancy, and Similarity in Software.
Walenstein, A., N. Jyoti, et al. (2003). Problems creating task-relevant clone detection reference data. 10th Working Conference on Reverse Engineering.
Wettel, R. and R. Marinescu (2005). Archeology of code duplication: Recovering duplication chains from small duplication fragments. Seventh International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC, Timisoara.
Wieringa, R. J. (1996). Requirements Engineering: Frameworks for Understanding, Wiley.
Yamamoto, T., M. Matsushita, et al. (2007). "Similarity of software system and its measurement tool SMMT." Systems and Computers in Japan **38**(6): 91-99.
Yoshida, N., Y. Higo, et al. (2005). "On Refactoring Support Based on Code Clone Dependency Relation." IEEE METRICS: 16.
Yu, L. and S. Ramaswamy (2008). "Improving modularity by refactoring code clones: a feasibility study on Linux." ACM SIGSOFT Software Engineering Notes **33**(2).