06301 Abstracts Collection
Duplication, Redundancy, and Similarity in Software
— Dagstuhl Seminar —

Rainer Koschke[1], Ettore Merlo[2] and Andrew Walenstein[3]

[1] Universität Bremen, DE
koschke@informatik.uni-bremen.de
[2] Ècole Polytechnique de Montréal, CA
ettore.merlo@polymtl.ca
[3] Univ. of Louisiana - Lafayette, US
walenste@ieee.org

Abstract. From 23.07.06 to 26.07.06, the Dagstuhl Seminar 06301 "Duplication, Redundancy, and Similarity in Software" was held in the International Conference and Research Center (IBFI), Schloss Dagstuhl. During the seminar, several participants presented their current research, and ongoing work and open problems were discussed. Abstracts of the presentations given during the seminar as well as abstracts of seminar results and ideas are put together in this paper. The first section describes the seminar topics and goals in general. Links to extended abstracts or full papers are provided, if available.

Keywords. Software clones, code redundancy, clone detection, redundancy removal, software refactoring, software

06301 Summary – Duplication, Redundancy, and Similarity in Software

This paper summarizes the proceedings and outcomes of the Dagstuhl Seminar 06301. The purpose of the seminar was to bring together a broad selection of experts on duplication, redundancy, and similarity in software in order to: synthesize a comprehensive understanding of the topic area, appreciate the diversity in the topic, and to critically evaluate current knowledge. The structure of the seminar was specifically formulated to evoke such a synthesis and evaluation. We report here the success of this seminar and summarize its results, much of which is a record of working groups charged with discussing the topics of interest.

Keywords: Duplication, redundancy, similarity, code clone, clone detector, refactor, code smells, software evolution, program development, visualization, software visualization

Joint work of: Walenstein, Andrew; Koschke, Rainer; Merlo, Ettore

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2007/971

06301 Working Session Summary: Presentation and Visualization of Redundant Code

This report summarizes the proceedings of a workshop discussion session presentation and visualization of aspects relating to duplicated, copied, or cloned code. The main outcomes of the working session were: (a) a realization that two researchers had independently generated very similar methods for browsing and visualization clone "clusters," and (b) a list of questions for visualization, particularly in relation to how the "proximity" of clones may relate to interest in the clone.

Keywords: Code clone, clone visualization, presentation, software visualization

Joint work of: Walenstein, Andrew; Cordy, James R.; Evans, William S.; Hassan, Ahmed; Kamiya, Toshihiro; Kapser, Cory; Merlo, Ettore

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2007/966

Similarity in Programs

An overview of the concept of program similarity is presented. It divides similarity into two types - syntactic and semantic - and provides a review of eight categories of methods that may be used to measure program similarity. A summary of some applications of these methods is included.

The paper is intended to be a starting point for a more comprehensive analysis of the subject of similarity in programs, which is critical to understand if progress is to be made in fields such as clone detection.

Keywords: Computer programs, similarity, code clone, software comparison, program metrics, Levenshtein distance, parameterized difference, feature space, shared information, plagiarism, compression

Joint work of: Walenstein, Andrew; El-Ramly, Mohammad; Cordy, James R.; Evans, William S.; Mahdavi, Kiarash; Pizka, Markus; Ramalingam, Ganesan; von Gudenberg, Jürgen Wolff

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2007/968

Subjectivity in Clone Judgment: Can We Ever Agree?

An objective definition of what a code clone is currently eludes the field.

A small study was performed at an international workshop to elicit judgments and discussions from world experts regarding what characteristics define a code clone. Less than half of the clone candidates judged had 80their criteria for judgment rather than their interpretation of the clone candidates. In subsequent open discussion the judges provided several reasons for their judgments. The study casts additional doubt on the reliability of experimental results in the field when the full criterion for clone judgment is not spelled out.

Keywords: Code clone, study, inter-rater agreement, ill-defined problem

Joint work of: Kapser, Cory; Anderson, Paul; Godfrey, Michael; Koschke, Rainer; Rieger, Matthias; van Rysselberghe, Filip; Weißgerber, Peter

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2007/970

Info

Magiel Bruntink (CWI - Amsterdam, NL)

Hi, I'm with the CWI in Amsterdam. My work in the area of clone detection has focussed on the relation between clones and crosscutting concerns (CCCs) from the AOP world. I think that CCCs are a common source for clones, but it is not always clear what kind of clones you can really observe in the code. My goal for the workshop would be to further look at this and discuss with the people working in the clone detection field.

I've written a paper on this topic, which was published in TSE and is downloadable here: http://www.computer.org/portal/cms_docs_transactions/transactions/tse/featured_article/e0804.pdf

Who the heck is Jim Cordy, and what does he want to discuss?

James R. Cordy (Queens University - Kingston, CA)

Short Bio

Jim Cordy is Professor and Director of the School of Computing at Queen's University, Kingston, Canada. He is the lead author of the TXL source transformation language and has worked for many years in the software analysis and transformation research domain. From 1995-2000 he was chief scientific officer at Legasys Corporation, a medium sized software company specializing in automated legacy software analysis and renovation. Through IBM Global Services, Legasys was responsible for the Year 2000 analysis and reprogramming system used by the majority of the large Canadian banks. Based on this industrial experience, his keynote paper at IWPC 2003 emphasized a number of errors of understanding in the academic research community, including the myth that cloned code was accidental, harmful or undesirable in legacy software systems. He is also the co-author of one of the most simple and elegant clone identification techniques, which won the Best Paper award at CASCON 2004, and has worked on the related problem of clone resolution in dynamic web pages.

Questions for the Seminar

I'm hoping that our discussions at the seminar can include three particular topics:

(1) Structure sensitivity in approximate clone detection - are non-structural methods useful at all? - and how can structural methods handle approximation?

(2) Parameterized clone resolution methods - how to extract and implement variance in approximate clones?

(3) Analysis of abstraction vs. risk - when is the extra risk introduced by clone resolution worth it, and when not?

References:
[1] J.R. Cordy, "The TXL Source Transformation Language", Science of Computer Programming 61,3 (August 2006), pp. 190-210.
[2] J.R. Cordy, "Comprehending Reality: Practical Challenges to Software Maintenance Automation", Proc. IWPC 2003, IEEE 11th International Workshop on Program Comprehension, Portland, Oregon, May 2003, pp. 196-206 (Keynote paper).
[3] J.R. Cordy, T.R. Dean and N. Synytskyy, "Practical Language-Independent Detection of Near-Miss Clones", Proc. CASCON'04, 14th IBM Centre for Advanced Studies Conference, Toronto, October 2004, pp. 29-40 (Best paper award).
[4] N. Synytskyy, J.R. Cordy and T.R.Dean, "Resolution of Static Clones in Dynamic Web Pages", Proc. WSE 2003, IEEE 5th International Workshop on Web Site Evolution, Amsterdam, September 2003, pp. 49-58.
[5] S. Grant and J.R. Cordy, "An Interactive Interface for Refactoring Using Source Transformation", Proc. REFACE'03, 1st International Workshop on Refactoring: Achievements, Challenges, Effects, Victoria, November 2003, pp. 30-33.
[6] N. Synytskyy, J.R. Cordy and T.R.Dean, "Robust Multilingual Parsing Using Island Grammars", Proc. CASCON 2003, 13th IBM Centres for Advanced Studies Conference, Toronto, October 2003, pp. 149-161.

Keywords: Clone detection, clone resolution, myths and magic

Full Paper:
http://www.cs.queensu.ca/~cordy/papers.html

Clone Detector Use Questions: A List of Desirable Empirical Studies

Thomas R. Dean (Queens University - Kingston, CA)

Code "clones" are similar segments of code that are frequently introduced by "scavenging" existing code, that is, reusing code by copying it and adapting it for a new use. In order to scavenge the code, the developer must be aware of it already, or must find it. Little is known about how tools - particularly search tools - impact the clone construction process, nor how developers use them for this purpose. This paper lists five outstanding research questions in this area and proposes sketches of designs for five empirical studies that might be conducted to help shed light on those questions.

Keywords: Code clone, clone detector, code search, reuse, code scavenging, empirical study

Joint work of: Dean, Thomas R.; Di Penta, Massamiliano; Kontogiannis, Kostas; Walenstein, Andrew

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2007/969

Breakout Session: Similarity by Defintion

Mohammad El-Ramly (University of Leicester, GB)

This session aimed to answer two questions: What does it mean to say that two pieces of code are similar? And how can this similarity be measured? In summary, there are five types of code similarity: Representation similarity (textual, syntactic and structural), Semantic or Behavioural similarity, Execution similarity, Metrics similarity and Feature-based similarity. Similarity measures are straightforward for some types and not so easy to find for other types, e.g., semantic similarity. The application of clone detection determines the type(s) of code similarity to use.

Keywords: Code similarity, clone detection, duplicate code

Joint work of: Cordy , James R.; El-Ramly , Mohammad; Evans, William; Toshihiro, Kamiya; Komondoor , Raghavan; Mahdavi , Kiarash; Pizka , Markus; Ramalingam , Ganesan; Walenstein , Andrew; von Gudenberg , Jürgen Wolff

Program Compression

William S. Evans (University of British Columbia - Vancouver, CA)

The talk focused on a grammar-based technique for identifying redundancy in program code and taking advantage of that redundancy to reduce the memory required to store and execute the program.

The idea is to start with a simple context-free grammar that represents all valid basic blocks of any program. We represent a program by the parse trees (i.e. derivations) of its basic blocks using the grammar. We then modify the grammar, by considering sample programs, so that idioms of the language have shorter derivations in the modified grammar. Since each derivation represents a basic block, we can interpret the resulting set of derivations much as we would interpret the original program.

We need only expand the grammar rules indicated by the derivation to produce a sequence of original program instructions to execute.

The result is a program representation that is approximately 40% of the original program size and is interpretable by a very modest-sized interpreter.

Keywords: Program compression, clone detection, bytecode interpretation, variable-to-fixed length codes, context-free grammars

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2007/963

Generic modelling of code clones

Simon Giesecke (Universität Oldenburg, D)

Code clones, i.e. instances of duplicated code, can be found in many software systems. They adversely affect the software systems' quality, in particular their maintainability and comprehensibility. Thus, this aspect is particularly important to consider in software maintenance and reengineering. Many different algorithms detecting code clones have been developed. For various reasons, it is difficult to compare the results of different algorithms. Most notable among these reasons is that there is no conceptual model allowing description of code clones determined by different algorithms. Much more, each algorithm uses its specific concept of code clones, which is rarely made explicit.

To overcome these problems, we have developed a generic model for describing clones. The model is generic in that it is independent of the programming language examined and of the clone detection algorithm used. It is flexible enough to facilitate various granularities of artifacts employed for selection and comparison, including inexact clones. The model allows separation of concerns between clone detection, description and management, which reduces the effort for the implementation of tools supporting these activities. On the basis of the model, we have implemented a prototype tool supporting these activities, tightly integrated into the Eclipse environment.

Keywords: Code clones, clone detection, reference model

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2007/960

A brief bio of Mike Godfrey

Michael Godfrey (University of Waterloo, CA)

Mike Godfrey is an assistant professor at the University of Waterloo in Ontario, Canada. His broad research goal is to understand what, exactly, software is, and how it evolves over time (nice, narrow little topic, eh).

With respect to code duplication, he and his grad students (notably Cory Kapser and Lijie Zou) have tried to focus on three fronts: case studies of cloning within large open source systems over time (eg Linux, Apache, PostgreSQL), origin analysis, and clone analysis.

When a software system evolves, often there are several program entities (eg methods, functions, classes, files, etc) in the new version that were not present in the previous one. Origin analysis (the term is ours) attempts to answer the question: "Where did these apparently new entities *really* come from?" That is, were they renamed, moved, clones, merged, split from entities in the previous version, or are they actually new? A paper in the Feb 2005 issues of IEEE Transaction in Sw Eng is the best summary of our work to date on this topic.

Clone analysis starts where clone detection ends. That is, there are many techniques for performing clone detection, but until recently there has not been much interest in analysing and processing the voluminous data that clone detection tools typically create. Our work to date has included a classification scheme for clones based on their relative position within the system architecture, a supporting tool to aid navigation and investigation of cloning activity, and a set of case studies. The best summary of our work to date on the topic is a paper in the March/April 2006 issue of the Journal of Software Maintenance and Evolution.

Keywords: Software cloning, clone analysis, origin analysis, case studies, Mike Godfrey bio

Managing Known Clones: Issues and Open Questions

Kostas Kontogiannis (University of Waterloo, CA)

Many software systems contained cloned code, i.e., segments of code that are highly similar to each other, typically because one has been copied from the other, and then possibly modified. In some contexts, clones are of interest because they are targets for refactoring. This paper summarizes the results of a working session in which the problems of merely managing clones that are already known to exist. Six key issues in the space are briefly reviewed, and open questions raised in the working session are listed.

Keywords: Code clone, software evolution, change management, code visualization, redundancy, metamodels, software management environments

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2007/967

Duplication, Redundancy, and Similarity in Software

Kostas Kontogiannis (University of Waterloo, CA)

THEME AND RELEVANT ISSUES TO DISCUSS

The problem of code cloning detection and code cloning analysis are important areas for researchers in the Software Engineering community to consider and to investigate. These relate to program comprehension (identification of idioms), system refactoring (clustering and amalgamation of clones to macros and parameterized procedures), migration to new object oriented platforms (use of clones to identify inheritance in the migrant object model), and quality assessment, to name a few. Some issues and problems in the area of code cloning and code duplication that warrant still the attention of researchers and can be interesting topics for our Dagstuhl meeting include:

– Tools and techniques for incremental clone detection so that identification complexity can be reduced. The incrementality may relate to the use of progressively more and more accurate methods in smaller and smaller sets of candidates.

– The combination of different methods (static analysis, data flow analysis, dynamic analysis, text based analysis) to improve the accuracy, recall, and precision of the current state of the art methods.

– More systematic classification / categorization of clone types.

– Experimental studies and how to design such studies in order to validate the relationship of code cloning with overall system quality and in particular maintainability.

– Experimental studies to justify the use of cloning, and under which conditions this will be acceptable or even desirable in large systems (e.g. to increase performance).

– The role of code cloning to program understanding, componentization, and design recovery, and in particular to the identification of programmatic idioms and clichés.

SHORT BIO / RELATED EXPETIENCE

Kostas Kontogiannis is an Associate Professor at the Department of Electrical & Computer Engineering at the University of Waterloo, Canada. Kostas received a B.Sc. degree in Mathematics from the University of Patras, Greece, a M.Sc. degree in Computer Science from Katholieke Universiteit Leuven, Belgium, and a Ph.D. degree in Computer Science from McGill University, Canada. Kostas is working in the areas of software reverse engineering, software reengineering, and software systems integration. Some of his most cited work includes the clone detection in large industrial systems using software metrics and approximate pattern matching techniques, and the use of code cloning to facilitate the migration of procedural systems to object oriented platforms.

Kostas has been the recipient of three IBM University Partnership Awards and a Canada Foundation for Innovation (CFI) Award. Kostas is a visiting scientist at the IBM Center for Advanced Studies in IBM Toronto Laboratory, and member of the IEEE Distinguished Visitors Program (2003-2006). Kostas can be reached by e-mail at kostas@swen.uwaterloo.ca

References:

[1] "Incremental Transformation of Procedural Systems to Object Oriented Platforms", Y. Zou, K. Kontogiannis. In proceedings of the IEEE International Computer Software and Applications Conference (COMPSACŠ03), November 2003, Dallas TX. pp.290-295.

[2] "Quality Driven Transformation Compositions for Object Oriented Migration", Y. Zou, K. Kontogiannis. In Proceedings of the IEEE Asia Pacific Software Engineering Conference (APSECŠ02) December 2002, Brisbane, Australia pp.346-355.

[3] "Pattern Matching Techniques for Clone Detection", K. Kontogiannis, R. DeMori, E. Merlo, M. Galler, and M.Bernstein. In Journal of Automated Software Engineering, Kluwer Academic Publishers, Vol. 3. pp.77-108, 1996.

[4] "Pattern Matching for Design Concept Localization", Kontogiannis, K., DeMori, R., Merlo, E., Bernstein, M., Galler, M. In Proceedings of the IEEE Working Conference on Reverse Engineering (WCRE'95) July 1995, Toronto, ON. pp. 96-103.

[5] "Towards an Integrated Toolset for Program Understanding", Mylopoulos, J., Stanley, M., Wong, K., Bernstein M., DeMori, R., Ewart G., Kontogiannis, K., Merlo, E., Muller, H., Tilley, S., Tomic, M. In Proceedings of CASCON'94, November 1994, Toronto, ON. pp. 19-31.

[6] "Localization of Design Concepts in Legacy Systems", Kontogiannis, K., DeMori R., Bernstein, M. In Proceedings of the IEEE International Conference on Software Maintenance (ICSMŠ94) September 1994 Victoria.BC, pp. 414-423.

Survey of Research on Software Clones

Rainer Koschke (Universität Bremen, D)

This report summarizes my overview talk on software clone detection research. It first discusses the notion of software redundancy, cloning, duplication, and similarity. Then, it describes various categorizations of clone types, empirical studies on the root causes for cloning, current opinions and wisdom of consequences of cloning, empirical studies on the evolution of clones, ways to remove, to avoid, and to detect them, empirical evaluations of existing automatic clone detector performance (such as recall, precision, time and space consumption) and their fitness for a particular purpose, benchmarks for clone detector evaluations, presentation issues, and last but not least application of clone detection in other related fields.

After each summary of a subarea, I am listing open research questions.

Keywords: Software redundancy, code clone, software evolution, clone detector, empirical evaluation

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2007/962

Working Group Report "Empirical Studies in Duplication Detection" I

Rainer Koschke (Universität Bremen, D)

We describe the outcome of the first Working Group "Empirical Studies in Duplication Detection" I

Keywords: Software redundancy, duplication, similarity, empirical studies

Allowing Overlapping Boundaries in Source Code using a Search Based Approach to Concept Binding

Kiarash Mahdavi (King's College - London, GB)

One approach to supporting program comprehension involves binding concepts to source code. Previously proposed approaches to concept binding have enforced nonoverlapping boundaries. However, real-world programs may contain overlapping concepts. This paper presents techniques to allow boundary overlap in the binding of concepts to source code. In order to allow boundaries to overlap, the concept binding problem is reformulated as a search problem. It is shown that the search space of overlapping concept bindings is exponentially large, indicating the suitability of sampling-based search algorithms. Hill climbing and genetic algorithms are introduced for sampling the space. The paper reports on experiments that apply these algorithms to 21 COBOL II programs taken from the commercial financial services sector. The results show that the genetic algorithm produces significantly better solutions than both the hill climber and random search.

Joint work of: Mahdavi, Kiarash; Gold, Nicolas; Harman, Mark; Li, Zheng

Keywords: Concept Assignment, Slicing, Clustering, Heuristic Algorithms

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2007/961

Code Normal Forms

Markus Pizka (TU München, D)

Because of their strong economic impact, complexity and maintainability are among the most widely used terms in software engineering. But, they are also among the most weakly understood! A multitude of software metrics attempts to analyze complexity and a proliferation of different definitions of maintainability can be found in text books and corporate quality guide lines. The trouble is that none of these approaches provides a reliable basis for objectively assessing the ability of a software system to absorb future changes. In contrast to this, relational database theory has successfully solved very similar difficulties through normal forms. This talk transfers the idea of normal forms to code. Semantic dependencies form the basis for code normal form criteria which in turn allow to elminiate redundancy and inconsistency systematically.

The Guy Behind the Name

Filip van Rysselberghe (University of Antwerp, B)

I'm a Phd student at Lab On Re-Engineering (LORE) under the supervision of Prof. Serge Demeyer. For my PhD research, I study how successful software systems evolved over time with the intention to deduce how systems can be maintained/evolved best. To meet this goal, I study which change operations (and sequences of them) are applied and how code problems evolve over time. The answer to these questions is used to understand when and how a certain maintenance task is performed (best). As data source, change data stored in versioning systems is used.

Now how does code duplication fit into this story. Well, for my master's thesis, I compared clone detection by means of exact string matching with parameterized matching using metric fingerprints. This bit of research fed my interest in the problem of clones, their detection and their removal. Due to this interest I for example decided to use clone detection techniques to detect move operations in a systems versioning system. I also carried out a priliminary study to verify how the clones in a system evolve in order to gather more information to decide on a ranking for the removal of clones. This latter subject is the subject I'm most interested in for te moment, deciding based on the past which clones have been removed, why?, ... in order to better rank the detection results in the future.

I also wrote out a number of master thesis studies in this context. This year a student is trying to apply Balazinska's automatic classification on clones other than function clones to verify whether the conclusions hold. Next year a student will study the evolution of clones within a system, with an emphasis on the relation between the presence of code smells and the presence of clones.

Looking at the list of participants I expect the seminar to result in a number of very concrete studies or experiments. Maybe, but that may already be quite difficult given the short amount of time we have, some initial "laptop" tests/experiments/implementations. Since my Phd topic is the study of historical information (i.e. changes) I would primarily contribute on how we could use history to get more insight into the cloning and clone removal process. Hence I'm primarily interested in the clone ranking subject.

I just want to mention one last thing. I always carry around an open-source, Java implementation of a point-of-sale system (the kind of system used in shops & restaurants to sell products & services) which is in need of re-engineering. Might be a fun case (see what we can detect, which clones WE would remove, reason for a clone, etc.)

The Software Similarity Problem in Malware Analysis

Andrew Walenstein (Univ. of Louisiana - Lafayette, USA)

In software engineering contexts software may be compared for similarity in order to detect duplicate code that indicates poor design, and to reconstruct evolution history. Malicious software, being nothing other than a particular type of software, can also be compared for similarity in order to detect commonalities and evolution history. This paper provides a brief introduction to the issue of measuring similarity between malicious programs, and how evolution is known to occur in the area. It then uses this review to try to draw lines that connect research in software engineering (e.g., on "clone detection") to problems in anti-malware research.

Keywords: Software, software evolution, commonality, program similarity, code clones, code smells, malicious software, malware, worms, Trojans, viruses, spyware, botnets, software visualization

Joint work of: Walenstein, Andrew; Lakhotia, Arun

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2007/964

Code Clones: Reconsidering Terminology

Andrew Walenstein (Univ. of Louisiana - Lafayette, USA)

This report discusses terminology choices and considerations relating to copied or redundant code within software systems, i.e., relating to "code clones." Inadequacies of existing terminology are raised and alternative terms are discussed.

Keywords: Code clone, exact clone, near clone, clone types, accidental clone, duplicate, copy, redundant

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2007/965