# Duplication, Redundancy, and Similarity in Software: Summary of Dagstuhl Seminar 06301

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

[1] University of Louisiana at Lafayette, Center for Advanced Computer Studies,
P.O. Box 44330, Lafayette, LA 70504-4330, U.S.A.
`walenste@ieee.org`
[2] Universität Bremen, Fachbereich 3 - Mathematik und Informatik,
Postfach: 330 440, 28334 Bremen, Germany
`koschke@informatik.uni-bremen.de`
[3] École Polytechnique de Montréal, Départment de Génie Électrique (DGEGI),
C.P. 6079, Succ. Centre Ville, Montréal, Québec H3C 3A7, Canada
`merlo@rgl.polymtl.ca`

**Abstract.** 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

## 1 Introduction

On July 23–26, 2006, 30 participants gathered at Schloss Dagstuhl, Germany, to participate in a seminar titled "Duplication, Redundancy, and Similarity in Source Code." The key purpose of the seminar was to gather together interested experts from around the globe in order to discuss topics related to the title. The organizers were specifically interested in formulating a seminar that would generate a new synthetic understanding of the area. That is, we considered the structure of the seminar to be an instrument for generating or extracting new knowledge from the participants through an iterative process of brainstorming and intense and focused discussion. Using the motif of knowledge mining, one could say we had knowledge to mine, the participants encoded the knowledge implicitly, and the purpose of the seminar was to serve as the instrument that could extract it.

The main question was "what are the important issues and research relating to duplication, redundancy, and similarity in source code, and what is the states of knowledge in the field, broadly construed?" The following sections recount the motivation, summarize the structure and organization of the seminar, and provide an overview of the contents and outcomes of the individual discussion sessions. This paper concludes with our reflections on the seminar, and an overview of the remainder of the proceedings.

## 2   Motivation

> *"Entia non sunt multiplicanda sine necessitate."*
> —Latin version of so-called Occam's Razor: "Entities should not be multiplied without necessity."

A venerable and long-standing goal and ideal in software development is to avoid duplication and redundancy. Duplication and redundancy can increase the size of the code, make it hard to understand the many code variants, and cause maintenance headaches. Avoiding redundancy or duplication can be viewed as the main motivation underscoring several diverse and long-standing software engineering research areas. It is a foundational motivation in investigations on parameterization, subclassing, code generation, software reuse, software refactoring, and frameworks. These are core software engineering topics.

Despite the ethic of avoiding redundancy, software, in practice, frequently contains redundancies and duplications. Some of these may be desirable; others may not be. For instance the technique of "code scavenging" is frequently used, and works by copying and then pasting code fragments, thereby creating duplicated or highly similar code. Redundancies or duplications can also occur because of missed reuse opportunities, or through purposeful duplication because of concerns of efficiency or to decouple parallel or forked threads of development.

Because redundancies frequently do exist in code, methods for detecting and removing them from software are needed in many contexts. Over the past few decades, smatterings of research on these issues have contributed towards addressing the issue. Techniques for finding similar code and on removing duplication have been investigated in several specific areas such software reverse engineering, plagiarism in student programs, copyright infringement investigation, software evolution analysis, code compaction (e.g., for mobile devices), and design pattern discovery and extraction. Common to all these research areas is the problems involved in understanding the redundancies and finding similar code, either within a software system, between versions of a system, or between different systems.

### 2.1   A Fragmented Research Area

Although this research has progressed over decades, only recently has the pace of activity in this area picked up such that significant research momentum could

be established. Rainer Koschke organized the *First International Workshop on Detection of Software Clones (IWDSC)*, co-located with SCAM'2002. Although still modest in size (about 15 people), it drew clone detection researchers from Europe, North America, and Japan. Seven months later, Andrew Walenstein and Arun Lakhotia organized a working session on clone detection benchmarking at IWPC'2003.

In November of 2003, the *Second International Workshop on Detection of Software Clones (IWDSC'2003)* was held in conjunction with WCRE. This workshop drew about 40 participants, and resulted in a summary writeup which was published in ACM Software Engineering Notes [1]. This third workshop saw a broadening of scope and participant background. The workshop included a research team focused on refactoring of redundant code for within agile development methods, a team exploring code sharing within and between open-source software, several teams investigating duplication within WWW sites, and an author who has served as an expert witness in copyright litigation.

These smaller workshops have had little effect in bringing together the multiple threads of research, and the distributed collections of researchers. For instance, out of the 60-70 people present at the three clone-related conferences in the two years prior, to the best of our knowledge only one person (Andrew Walenstein) was present at all three. The smaller workshops were scattered geographically and temporally, and were only 2-6 hours in length. This ensured fragmentation of both the topic and the community, and made it impossible to dig deeply into the issues. Furthermore, because the workshops were co-located with certain selected conferences, it made it extremely difficult to draw together researchers working on similar problems from different areas (agile methods, reverse engineering, code compaction, plagiarism, and so on).

Clearly, finding and eliminating code copying and redundancy can be expected to be an ongoing concern in many different areas relating to software. But this topic has been fragmented: it has been split up into multiple distinct topics (plagiarism, etc.) and different communities.

## 2.2   Defragmenting the Research

The central purpose of the Seminar was to solidify and give shape to the emerging research area and community. In short, the key goals were *synthesis* and *brainstorming*. More specifically, we wished to bring together researchers from diverse fields in order to synthesize a more comprehensive understanding of the area, and to evaluate the current state of research, discuss common problems, discover opportunities for collaboration, exchange ideas, and envision new approaches.

To set the stage, the organizers specifically avoided using terms specific to any one thread of research. We eschewed terms such as "clone detection" and "code refactoring" for use in the title. Instead, we focused on essential issues in the properties and comparison of source code: duplicity, redundancy, and similarity comparisons. In this manner we sought to avoid predisposing the participants to thinking along pre-set lines; to refocus on essential issues with the hope that new commonalities may be found.

To try to enable the hoped-for synthesis, the organizers specifically sought to invite a broad range of researchers who could bring multiple different perspectives to the table. We sought to include those who have published in different areas and different conferences, and tried to ensure that all main related areas were covered. These included: traditional clone detectors and clone visualization, code refactoring, plagiarism detection, code compaction, malware analysis, and empirical investigations of duplication and redundancy in software. We also tried to ensure diversity in the group, striving to invite representatives from industry as well as young academics.

## 3      Seminar Structure and Organization

Since the seminar was conceived of as focusing on synthesis and brainstorming, canned presentation was minimized in favour of interactive discussion. There were four main activities: keynote presentations, working groups, working group reporting sessions, and table-of-contents (overview) brainstorming session. Each of these are summarized in separate sub-sections.

### 3.1   Keynote Presentations

To set the stage, four presentations were made on selected topics by invited presenters. These were:

| PRESENTER | TOPIC |
|---|---|
| Rainer Koschke | Clone Detection |
| Ettore Merlo | Plagiarism |
| Will Evans | Program Compression |
| Andrew Walenstein | Malware Analysis |

The intent of the presentations were to briefly summarize the topic areas and to identify possible commonalities and differences between the other topics. The overview was intended to help participants with different backgrounds be aware of work done in different areas, and to appreciate both common and distinct problems within the different areas. It was also hoped that the initial discussions would set the stage for topics of the breakout or "working" groups.

### 3.2   Working Groups

By design, no topics were constructed *a priori* for the working sessions. Indeed, the organizers took the position that the list of topics of interest was a critical unknown, and the list constructed during the workshop would be an important outcome in the sense that it would represent a prioritization of topic importance by virtue of whether they survived expressions of interest from the participants. Thus, a goal of the seminar was to allow the participants to collectively work to construct a selection of topics that were mutually of interest. And, indeed, the question of constructing appropriate working groups was an intensely debated topic. This section reports on the process and its outcomes.

## Process

The organizers had spent much time in deriving a process for establishing the working groups. The process was arrived at primarily after discussing the successes and drawbacks of workshops we had previously been involved in, including IWDSC'2003. We wished it to be simple. More importantly, we wanted the process to ensure that topics would be selected that participants truly were interested in, and which would generate great discussions and useful outcomes. The method is essentially a group brainstorming with open voting, utilizing multiple but limited votes as the method to reduce the number of topics.

The brainstorming phase generates a potential topic list, and works as follows. An organizer stands at the front of the room and solicits requests for topics. These are written down. Frequently a topic will be raised that is similar-but-not-exactly-like another topic. The organizer essentially makes his or her own judgment on whether to join topics together or not, although open feedback from the audience is encouraged. After some time, the number of topics grows.

After the topics list is generated, an open voting phase begins. Before voting is permitted, a "champion" must step forward for each topic. Should the topic be selected, the champion is charged with chairing the discussion, i.e., moderating it, and taking notes. They would later be called upon to also present the working group findings at the reporting sessions. If a topic could not attract a champion, it was discarded as not being important enough.

The idea of the voting method is to limit the topics list to only those that enough people are interested in. Each attendee is given exactly three votes, each of which he or she must use. Votes are cast for the topic they are willing to discuss in a working group. A minimum number of votes is required for any topic. Any topics not meeting the minimum after a given round are dropped and voting is re-done using the new topic. After topics are to a suitable amount, a final assignment round is performed, i.e., everyone casts a single "vote" to say which working session they wish to attend. In the case where a session is too popular some people are asked to move.

While the process we settled on was not without flaws, we found it worked well. We received much positive feedback from our participants. In our opinion, it was so successful that we expect to try it (or some variation) again.

## Outcomes

The entire nomination and voding process was performed three times, one for each working groups session during the seminar; no efforts were made to steer the participants to avoid discussing topics already discussed in previous sections.

At first, no significant structure was emerging from the topic proposition and discussion. After some time, though, the suggestions became more universally agreed upon, and a list of approximately 15 topics was settled on. Thereafter, voting reduced the numbers rapidly. Eventually, everyone seemed satisfied that they had a working session they could go to that was of interest to them. The

same process was performed on the second day, with similar success. The final list of working group topics were as follows:

- Monday, July 24 2006
  - Clone Detectors
  - Applications of Clone Detectors
  - Managing Known Clones
- Tuesday, July 25 2006 (morning)
  - Empirical Studies of Clones
  - Environments for Managing Clones
  - Redundancy - Taxonomy and Definition
  - Presentation and Visualization
- Tuesday, July 25 2006 (afternoon)
  - Empirical Studies of Clone Detectors
  - Similarity by Definition (Foundational Issues)
  - Judging Clones by Example

### 3.3   Working Groups

Working groups ranged in size from 3 or 4 to 19. Generally speaking, the larger working groups did not, in our opinion, lead to as much interaction in the discussion. The working groups were allowed to discuss what they wished so long as it was related to the topic. In our experience, as a general rule most sessions followed a similar course of action: (1) review of known material and status of the field, (2) raising of important issues and open questions, (3) discussion of possibilities in the future.

### 3.4   Working Group Reporting Sessions

After the working groups were separated, the entire seminar as a whole was reconvened. The champions for each working group were charged with making a short presentation to the group. After the presentation, an open discussion session ensued. It was typical for these discussion to be lively, and often the discussions centered on discussion of open issues. This sort of discussion satisfied the main motivating goals of the seminar.

### 3.5   Table of Contents Session

The final seminar session was devoted to arriving at a table of contents for a proposed DRASIS book. The practice of the topic brainstorming lead to a smooth process of constructing a table of contents. The feedback we received was that the table of contents emerged naturally and represented a clear organization of the topic area. Regardless of whether a book is eventually produced, organization of a topic area is an important knowledge construction task, and we feel the prototype table of contents provides an example of a consensus understanding that was communally arrived upon in an open process. An abbreviated table of contents appears in Figure 1.

1. Summary and Motivation
   - Most important lessons learned, Success stories
2. Definition, Terminology, Types of Clones/Taxonomy
   - Types of similarity (textual, syntactic, structure, semantic, execution), Level of abstraction (req. des. doc, code (prog/test)), Characteristics (intent, measures, relations to aspects), Occurrence (web, programs, across languages, architecture, specification), Similarity measures (attribute-based, metrics, shared info, edit distance), Purpose-specific measures (synatic/semantic clones, execution-curve), Relating clones to quality attributes, Similarity by definition (explicit, implicit), Miscellanea (size, dependence, freq, stability, meaningfulness)
3. Detectors
   (a) Technologies, taxonomy of detectors
      - Textual (text, identifiers), Lexical (tokens, parameterized), Syntatic (parse trees, ASTs), Semantic (flow graphs), Meta data (programmer, subsystem, history)
   (b) Normalization
   (c) Implementation techniques
      - metric, frequent item sets, suffix trees, hashing, edit distance, slicing, graph matching, tree matching, latent semantic indexing, dynamic programming, transformations, normalized compression distance
   (d) Quality attributes of detectors
      - language independence, implementation effort, time and memory efficiency, precision and recall, clone type, ranking or degree of evidence, granularity, distinction of clone types, difference analysis, user-defined filters, stability, tolerance, customizability
   (e) Detection/analysis Process
      - inventory, positioning and scoping, analysis, evaluation, subsequent actions
   (f) Task-/projects-oriented filters
   (g) Open issues
      - Selection, Process, Technology, Task-relevance
4. Presentation and Visualization
5. Development Environments
6. Refactoring Tools
7. Benchmarks and Evaluation
8. Empirical Data
   - clone evolution, prevalence, relations to features or activities
9. Empirical Studies
   (a) Types of empirical studies (survey, case studies, controlled expt.)
   (b) (null) Hypothesis
   (c) Evolutionary aspects
      - fault prediction, clone prediction, evolution of clones
   (d) Characteristics of good subject systems
      - small (study), large (realism), available, industrial, diversity, history
   (e) Candidate systems
   (f) Threats to validity
10. Managing Known Clones
   - cost/benefits, avoidance, refactoring, uneliminable, environments, awareness, engineering, clone operations, roles
11. Applications
   - refactoring opportunities, aspect mining, evolution, plagiarism, compression, malware analysis

**Fig. 1.** Simplified table of contents resulting from brainstorming sessions

## 4   Reflections, Conclusions, and Acknowledgments

The remaining entries in this proceedings consist of one of three types of entries. The first are summaries of the keynote presentations. The aim of these summaries is to establish broad-brush outlines of the breadth of topics in the area—to firmly assert that there is more to the area than simply "clone detection." Following this is a summary report on terminological discussions that permeated the seminar. Finally, reports on working sessions are included; these serve to document their outcomes, which primarily consist of open questions and issues. We are hopeful that they will be instrumental in the next wave of research in the area.

As organizers, we hoped the seminar would bring about a new understanding of the field and, in so doing, help lay the foundations for future research in the area. In reflecting back on the seminar, we have to conclude that it produced many successes. The discussions were lively and we know that many interesting ideas for future research were discussed in the working groups and the in the open discussions during the working group reporting sessions. We believe that the variety of interests of the participants served a key purpose: we think it helped broaden the scope and forced a critical reexamination of foundational assumptions, including terminology and concepts.

In closing, we wish to thank the participants for their cooperation, discussion, and efforts, and especially wish to thank the champions for their leadership, and thank every participant who spent time in writing up reports or summaries, or presenting the reports orally. We are particularly grateful to the Dagstuhl organization and the German government for making the seminar possible.

## References

1. Walenstein, A., Lakhotia, A., Koschke, R.: The second international workshop on detection of software clones: Workshop report. SIGSOFT Software Engineering Notes **29** (2004) 1–5. http://doi.acm.org/10.1145/979743.979752.