Presentation and Visualization of Redundant Code: Working Session Summary

Andrew Walenstein[1], Jim Cordy[2], Will Evans[3], Ahmed Hassan[4], Toshihiro Kamiya[5], Cory Kapser[6], and Ettore Merlo[7]

[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] Queen's University, School of Computing, Kingston, ON, Canada, K7L 3N6
cordy@queensu.ca
[3] University of British Columbia, Department of Computer Science, 201-2366 Main Mall, Vancouver, BC, Canada V6T 1Z4
will@cs.ubc.ca
[4] University of Victoria, Electrical and Computer Engineering, Engineering Office Wing 315, P.O. Box 3055, Victoria, BC, Canada V8W 3P6
ahmed@ece.uvic.ca
[5] National Institute of Advanced Industrial Science and Technology (AIST), Information Technology Research Institute, Ubiquitous Software Group
[6] University of Waterloo, David R. Cheriton School of Computer Science, Waterloo, Ontario, N2L 3G1, Canada
cjkapser@uwaterloo.ca
[7] École Polytechnique de Montréal, Départment de Génie Électrique (DGEGI), C.P. 6079, Succ. Centre Ville, Montréal, Québec, Canada H3C 3A7
merlo@rgl.polymtl.ca

Abstract. 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

1 Introduction

This report summarizes the proceedings of a working session from the DRASIS Workshop [1]. The topic of the working session was the presentation and visualization of aspects relating to duplicated, copied, or cloned code. Several members of the medium-sized group (around 15) had extensive experiences in different visualization types, and were able to relay some of their experiences. The session

Dagstuhl Seminar Proceedings 06301
Duplication, Redundancy, and Similarity in Software
http://drops.dagstuhl.de/opus/volltexte/2007/966

had three distinct phases of activity: (1) survey and recount of presentation or visualization techniques, (2) discussions of tasks supported by visualizations, and (3) a less structured session of asking questions and brainstorming about filtering of clones, and classifying them. This report is structured into sections along those lines.

2 Review of Presentation Techniques

The session started out with a recounting of well-known and some less-well-known clone visualization techniques. The implicit purpose of the review appeared to be to prime the discussion: to ensure all discussants present were aware of well-known techniques, and to bring the different approaches together in hopes that the juxtaposition would spark insight. Initial overview and discussion was lead by Toshihiro Kamiya. The recount covered the following (inexhaustive) list of different visualization techniques:

1. 2-D scatter plots in the style of "dotplot" [2] and "duploc" [3].
2. Phylogenetic trees [4] or genealogies [5].
3. Visual presentation of alignment or multiple alignment (diffing).
4. Treemaps and "duplication webs" [6].
5. "SeeSoft" [7]-style clone location views.
6. "Hasse" diagrams; used with clustered clone relationships.

In addition to this review, two live demonstrations were made on laptops, introducing techniques not as widely known. One visualization was demonstrated by Cory Kapser. The system he demonstrated is known as "CLICS" [8]. The visualization uses an ordinary "box and arrow" diagram of a system's high level component interconnection graph, in which the edges convey clone relationships between the components, including the classification of the relationships. A second visualization was demonstrated by Ahmed Hassan. It showed a graph using labeled edges. Rectangles depicted subsystems or modules. Edges depicted aggregated clone relationships between subsystems. Diamonds were shown on edges which indicated the size of the clones. The volume of and shape of the diamonds conveyed additional information about the clone relationships: their width, for example, showed the number of clones between the subsystems. Colour was used for redundant coding of the clone information. The effect was to quickly convey the location of possible maintenance issues.

One of the most intense discussions regarded an exemplar-based method of viewing and navigating entire clusters of clones. Interest was generated, in part, by the realization that two participants—Jim Cordy and Will Evans—had independently generated similar visualizations. Some of the details are availble in published form [9,10], but from the discussions the key common features of the visualization process are:

- algorithms are used to collect related clone pairs into clusters,
- scoring criteria are used to rank the closeness of the clone instances,

– entire clusters are abstracted to a *schema* level by using identifiers in the places where variations occur,
– users are able to drill-down from clone instances to their locations and context.

2.1 Discussion of Tasks Supported by Visualization

The group was sensitive to the fact that the value of a visualization is related to the tasks for which they are used. The group chose to ask which types of tasks are supported by the reviewed visualizations. It was agreed that the visualizations presented by Ahmed Hassan seem especially appropriate for system assessment and estimation of effort during software evolution. An interesting point was made by three of the members: that the task of producing correct tools was itself helped greatly by visualizations. All of Jim Cordy, Ettore Merlo, and Will Evans were able to recount experiences where the visualizations helped the tool-builders themselves rather than the tools' end users.

3 Filtration and Classification Brainstorming

The remainder of the session consisted of divergent discussion; it took form by asking questions and then discussing possible answers.

Question: can we visualize cloning to understand *intent*?

Clones may exist because the developer had a particular intention for the clones. An extended discussion was started regarding whether one may differentiate between *divergent* and *convergent* clones. The terms "divergent" and "convergent" refer the to evolutionary directions taken cloned pairs. In divergent evolution, the pairs evolve independently and differences will tend to increase over time; equivalently, the clones are loosely coupled. In many cases, the developer may intend for pairs to be allowed to diverge.

In discussing these properties of clone relationships, participants agreed that all clone management tools will need to have functionality to allow users to drop clone relationships. Some simply are not of interest to the user, even though they might well be considered as clones by someone. The underlying idea is that the user's context alters "relevance" [11]. Two contexts, in particular, were discussed:

– When clones cross subsystem ownership boundaries. In such cases, these are "don't care clones" because the subsystems will evolve separately. Moreover, there is no desire to refactor the code to remove the clones because it would create an artificial coupling where none is desired (Conway's law).
– When there is a business case to keep or ignore the clone. This was noted by Ettore Merlo to occur in genuine industrial contexts.

Question: What is the Relation between Original Author and Clones?

This question was precipitated by Cory Kapser recounting a case study on log files that showed that a single maintainer took responsibility of a full clone set. The maintainer performed "parallel" updates to all instances to make them consistent.

Question: Is there a Correlation between Proximity and "Interestingness" of Clones?

The question was more-or-less directly prompted by the previous two questions. In particular, a chain of reasoning was performed, as follows:

- One important aspect of "interestingness" is whether the clones in a clone pair are maintained by different organizational or business units or concerns,
- Conway's law will tend to ensure the code structure reflects organizational structure,
- Modularity ensure concerns are localized,
- ... so does clone proximity indicate commonality of concern, and therefore indicate degree of interest?

The group also asked a corollary question: is there a (negative) correlation between proximity and clone divergence rates?

Question: Can we Combine Some Metrics to Use As an Estimator for Deciding Whether Clone Pairs are Convergent or Divergent?

This, once more, was a follow-on question that was precipitated by prior questions. The idea was that management structure in place could either force convergence, or allow for divergence. The thought occurred that a measure of code locality might then be used as a heuristic for the likelihood that a given clone pair would be convergent or divergent. That is, if one measures the distance of clone pairs (in terms of distance within the system structure graph, presumably), can one reasonably use the distance as a heuristic for the likelihood that they are divergent?

References

1. Walenstein, A., Koschke, R., Merlo, E.: Duplication, redundancy, and similarity in software: Summary of Dagstuhl seminar 06301, Dagstuhl, Germany, Dagstuhl (2006) ISSN 1682–4405.
2. Church, K.W., Helfman, J.I.: Dotplot: A program for exploring self-similarity in millions of lines of text and code. Journal of Computational and Graphical Statistics **2** (1993) 153–174

3. Rieger, M., Ducasse, S.: Visual detection of duplicated code. In: ECOOP '98: Workshop ion on Object-Oriented Technology, London, UK, Springer-Verlag (1998) 75–76
4. Yamamoto, T., Matsushita, M., Kamiya, T., Inoue, K.: Measuring similarity of large software systems based on source code correspondence. In: Product Focused Software Process Improvement. Volume 3547 of Lecture Notes in Computer Studies. Springer Berlin / Heidelberg (2005) 530–544 http://www.springerlink.com/content/h6m2vg5c3ejk3814.
5. Kim, M., Sazawal, V., Notkin, D., Murphy, G.: An empirical study of code clone genealogies. In: ESEC/FSE-13: Proceedings of the 10th European software engineering conference held jointly with 13th ACM SIGSOFT international symposium on Foundations of software engineering, New York, NY, USA, ACM Press (2005) 187–196 http://doi.acm.org/10.1145/1081706.1081737.
6. Rieger, M., Ducasse, S., Lanza, M.: Insights into system-wide code duplication. In: WCRE '04: Proceedings of the 11th Working Conference on Reverse Engineering (WCRE'04), Washington, DC, USA, IEEE Computer Society (2004) 100–109 http://doi.ieeecomputersociety.org/10.1109/WCRE.2004.25.
7. Eick, S.G., Steffen, J.L., Eric E. Sumner, J.: Seesoft-a tool for visualizing line oriented software statistics. IEEE Transacctions on Software Engineering **18** (1992) 957–968 http://dx.doi.org/10.1109/32.177365.
8. Kapser, C., Godfrey, M.W.: Improved tool support for the investigation of duplication in software. In: ICSM '05: Proceedings of the 21st IEEE International Conference on Software Maintenance (ICSM'05), Washington, DC, USA, IEEE Computer Society (2005) 305–314 http://dx.doi.org/10.1109/ICSM.2005.52.
9. Cordy, J.R., Dean, T.R., Synytskyy, N.: Practical language-independent detection of near-miss clones. In: Proceedings of the 15h IBM Center for Advanced Studies Conference (CASCON'04), Toronto, ON, Canada (2004) 29–40
10. Evans, W.S., Fraser, C.W.: Clone detection via structural abstraction. Technical Report MSR-TR-2005-104, Microsoft Research (2005)
11. Walenstein, A., Jyoti, N., Li, J., Yang, Y., Lakhotia, A.: Problems creating task-relevant clone detection reference data. In van Deursen, A., Stroulia, E., Storey, M.A.D., eds.: WCRE, IEEE Computer Society (2003) 285–295