**Emerging Topics in Software Engineering**

**UFCFCD-15-M**

**Student Number: 15027313**

**A Critical Review of using Search-Based Software Engineering Techniques to Refactor Code**

**Word Count:**

**1 INTRODUCTION**

Search-based software engineering (SBSE) is the name given to the search-based optimisation subfield of software engineering. It involves using metaheuristic search techniques to find a near-optimal solution, and will result in a decent solution but cannot guarantee that it is the very best solution. Henceforth, search-based refactoring will be referred to as SBR.

There are several ways to apply this to a variety of issues, but this review will focus on search-based or sear-supported refactoring. Refactoring is the activity of improving the internal structure of code, while leaving the external structure unchanged. Search-based refactoring is about automatically discovering and performing useful refactorings, which has previously been a predominantly manual activity due to the difficulties of successfully locating and accomplishing this automatically.

“Software systems are subject to continual change and as they evolve to reflect new requirements, their internal structure tends to degrade. The cumulative effect of such changes can lead to systems that are unreliable, difficult to reason about, and unreceptive to further change.” (Harman and Tratt, 2007).

The scope of this topic will include the use of SBSE to locate possible areas of, as well as performing, refactoring of existing code bases.

“The process of writing code can be compared to that of writing a paper or any other document – it starts with some ideas, a few drafts, and then a lot of time spent on refining and rewording, and is not something that should be rushed. Quality code takes a lot of time to write, and just like the writing of draft after draft will increase the author’s understanding of the topic area, code is written, changed, and changed again times and times over as the developers’ understanding of the system and the requirements expands. It is in the human nature to start small and build upward; in this case starting with code that works with limited functionality, expanding it to implement more functionality, and then refining it to optimise performance, readability, error handling, and so on.” (Birkehaug, 2016)

<place here some background and brief history. What is the scope and context of your chosen emerging topic?>

**2 FOCUS OF CRITICAL REVIEW**

Harmon and Langdon from UCL are two widely published and cited researchers in the area.

< What are the main areas of your chosen emerging topic? Who are the key researchers in the community of your emerging topic? In your critical review, what is/are the research question/s posed? >

**3 SURVEY METHODOLOGY**

The principal method behind locating papers was through the use of Google Scholar. The reason for choosing this particular search engine is that they display articles from a variety of different sources, and is a quick way to get an overview of papers as well as an indication of their recognition in the field, as they include a count of citations for each publication. While the citation count may indicate popularity and high-quality research, it should also be noted that these may be very low for newer publications, and it is thus difficult to judge how well a paper is received by these alone.

The first search done was “search based refactoring”, and did not include patents or citations. A brief skim of the abstract for the first 50 hits deemed the majority generally relevant, so as this essay aims to cover state of art, the search was narrowed down to only include articles published since 2012, resulting in a publication range of approximately 4.5 years at the time of writing. The software engineering field moves very quickly and this range appeared to provide enough relevant publications to fulfil this work’s requirements.

The following list shows search hit number, titles and publication years of the first 20 hits, sorted by publication dates in descending order. The ones marked with an asterix are the ones that had full-text available and the titles in bold are those that were ultimately included in the review, with justifications to follow.

|  |  |  |
| --- | --- | --- |
| # ft | Title | Year |
| 11 \* | **An experimental search-based approach to cohesion metric evaluation** | **2016** |
| 14 | [Book] Search-Based Software Engineering: 7th International Symposium, SSBSE 2015, Bergamo, Italy, September 5-7, 2015, Proceedings | 2015 |
| 20 | AutoRefactoring | 2015 |
| 10 \* | **On the use of many quality attributes for software refactoring: a many-objective search-based software engineering approach** | **2015** |
| 8 \* | **Search-based refactoring: Metrics are not enough** | **2015** |
| 1 \* | **Automated migration of build scripts using dynamic analysis and search-based refactoring** | **2014** |
| 3 \* | **High dimensional search-based software engineering: finding tradeoffs among 15 objectives for automating software refactoring using NSGA-III** | **2014** |
| 12 | On the use of machine learning and search-based software engineering for Ill-defined fitness function: a case study on software refactoring | 2014 |
| 13 \* | Search based software engineering for software product line engineering: a survey and directions for future work | 2014 |
| 18 \* | [Keynote] Dynamic adaptive Search Based Software Engineering needs fast approximate metrics | 2013 |
| 17 \* | Pareto-optimal search-based software engineering (POSBSE): A literature survey | 2013 |
| 7 \* | **Search-based refactoring detection** | **2013** |
| 9 | Search-based refactoring detection using software metrics variation | 2013 |
| 2 \* | **Search-based refactoring using recorded code changes** | **2013** |
| 16 \* | Dynamic adaptive search based software engineering | 2012 |
| 15 | Improving software security using search-based refactoring | 2012 |
| 4 | Search based software engineering: Techniques, taxonomy, tutorial | 2012 |
| 19 \* | Search-based model transformation by example | 2012 |
| 6 \* | **Search-based refactoring: Towards semantics preservation** | **2012** |
| 5 \* | **Search-based software engineering: Trends, techniques and applications** | **2012** |

Based on the titles as well as the abstracts provided by Google Scholar, most of these 20 articles were seemingly relevant in the sense that they refer to search-based refactoring, and include one of the SBSE techniques in some way. A closer look did however reveal that this was not the case for all of them. Several excluded papers turned out to reference other potentially relevant papers, some of which were found in the search, but those that were not have not been included as a sufficient amount of papers passed the selection criteria.

* #4 (was excluded, as it is a tutorial in the shape of a book chapter and not a research paper.
* #13 was also deemed irrelevant as it focuses on SBSE for software product lines, and only mentions refactoring in a generic listing of the possible applications of SBSE.
* #14 is also a reference to book and not an individual paper, and has been excluded.
* #16 and #18 are in fact the same paper, and as they are predominantly about dynamically adaptive software and not refactoring in the sense of only changing internal structure (as well as only mentioning refactoring a single time), they have been excluded too.
* #17 is a literature survey of papers using multi-objective search to find solutions, and only mentions refactoring in the titles of one of the surveyed works, which being published in 2007 also falls out of scope of this review.
* #19 does mention a plan of adapting their approach to other transformation problems including refactoring, but does not currently do so and has been excluded.

Following is a short summary of the nine papers that passed the selection and were available in full-text through a university account, as well as one (#8) that could fortunately be obtained directly from one of the authors.

1. “Automated migration of build scripts using dynamic analysis and search-based refactoring” (Gligoric *et al*., 2014), is using SBR to raise the abstraction level of the code, in order to assist the process of migrating build scripts.
2. “Search-based Refactoring Using Recorded Code Changes” (Ouni, Kessentini and Sahraoui, 2013) is using code changes recorded over time together with structural and semantic information in order to come up with more precise and efficient refactoring suggestions.
3. “High dimensional search-based software engineering: finding tradeoffs among 15 objectives for automating software refactoring using NSGA-III” (Mkaouer *et al*., 2014) proposes a scalable SBSE approach based on an evolutionary optimization method where the refactoring solutions are evaluated using 15 different quality metrics.
4. “Search-based software engineering: Trends, techniques and applications” (Harman *et al.,* 2012) provides a review and classification of SBSE literature, highlighting areas in need of more research.
5. “Search-based refactoring: Towards semantics preservation” (Ouni *et al*., 2012) focuses on finding an optimal refactoring sequence in order to minimise semantic errors while maximising quality improvements.
6. “Search-based refactoring detection” (Mkaouer, Kessentini and Cinnéide, 2013b) uses global and local heuristic search algorithms together with the code’s structural information to automate the detection of source code refactorings, using a manually revised version as a benchmark for the automatic refactoring, aiming to keep their metrics similar.
7. “Search-based refactoring: Metrics are not enough” (Simons *et al.,* 2015) recommends that future SBSE refactoring research should keep the human-in-the-loop in order to refactor code in a way that is helpful to the software engineers.
8. “On the use of many quality attributes for software refactoring: a many-objective search-based software engineering approach” (Boukdhir *et al.,* 2014) tackles the problem of optimising conflicting objectives by introducing a many-objective refactoring technique, evaluating refactoring solutions with a set of 8 distinct objectives.
9. “An experimental search-based approach to cohesion metric evaluation” (Cinnéide *et al.*, 2016) propose a search-based refactoring technique used to ‘animate’ metrics and observe their behaviour in a practical setting, in order to compare and discover how metrics relate to each other.

**4 Search-based refactoring**

<place your critical review of theme 1 here>

**5 <place the title of AREA 2 here>**

<place your critical review of theme 1 here>

***N* <place the title of AREA *N* here>**

<place your critical review of theme *N* here>

***N*+1 CONCLUSIONS AN FUTURE DIRECTIONS**

<place your conclusions and suggestions for future directions here>

**REFERENCES**

Friedberg, R. (1958) A learning machine: Part 1. IBM J Res. Dev. 2(1), 2–13

Friedberg, R., Dunham, B. and North, J. (1959) A learning machine: Part 2. IBM J. Res. Dev. 282–287

Harman, M. and Tratt, L. (2007) Pareto optimal search based refactoring at the design level. In: *Proceedings of the 9th annual conference on Genetic and evolutionary computation.* New York: ACM, pp. 1106-1113

//refworks

Amal, B., Kessentini, M., Bechikh, S., Dea, J. and Said, L.B. (2014) On the use of machine learning and search-based software engineering for Ill-defined fitness function: a case study on software refactoring. In: *Search-Based Software Engineering* [online]. Springer, pp.31-45.

Cinnéide, M.Ó., Moghadam, I.H., Harman, M., Counsell, S. and Tratt, L. (2016) An experimental search-based approach to cohesion metric evaluation. In: *Empirical Software Engineering* [online]. pp.1-38.

de Oliveira Barros, M. and Labiche, Y. (2015) *Search-Based Software Engineering: 7th International Symposium, SSBSE 2015, Bergamo, Italy, September 5-7, 2015, Proceedings* [online]. Springer.

Ghaith, S. and Cinnéide, M.O. (2012) Improving software security using search-based refactoring. In: *Search Based Software Engineering* [online]. Springer, pp.121-135.

Gligoric, M., Schulte, W., Prasad, C., Van Velzen, D., Narasamdya, I. and Livshits, B. (2014) Automated migration of build scripts using dynamic analysis and search-based refactoring, In: *ACM SIGPLAN Notices* 2014, ACM, pp. 599-616.

Harman, M., Burke, E., Clark, J.A. and Yao, X. (2012). Dynamic adaptive search based software engineering, In: *Empirical Software Engineering and Measurement (ESEM), 2012 ACM-IEEE International Symposium on 2012*, IEEE, pp. 1-8.

Harman, M., Clark, J. and Cinneidez, M.O. (2013) Dynamic adaptive Search Based Software Engineering needs fast approximate metrics (keynote), In: *Emerging Trends in Software Metrics (WETSoM), 2013 4th International Workshop on 2013*, IEEE, pp. 1-6.

Harman, M., Jia, Y., Krinke, J., Langdon, W.B., Petke, J. and Zhang, Y. (2014) Search based software engineering for software product line engineering: a survey and directions for future work, In: *Proceedings of the 18th International Software Product Line Conference-Volume 1* 2014, ACM, pp. 5-18.

Harman, M., Mansouri, S.A. and Zhang, Y. (2012) Search-based software engineering: Trends, techniques and applications. *ACM Computing Surveys (CSUR)* [online]. 45 (1), pp.11.

Harman, M., McMinn, P., De Souza, J.T. and Yoo, S. (2012) Search based software engineering: Techniques, taxonomy, tutorial. In:*Empirical Software Engineering and Verification* [online]. Springer, pp.1-59.

Kessentini, M., Sahraoui, H., Boukadoum, M. and Omar, O.B. (2012) Search-based model transformation by example. In: *Software & Systems Modeling* [online]. 11 (2), pp.209-226.

Mahouachi, R., Kessentini, M. and Cinnéide, M.Ó. (2013a) Search-based refactoring detection using software metrics variation. In: *Search Based Software Engineering* [online]. Springer, pp.126-140.

Mahouachi, R., Kessentini, M. and Cinnéide, M.Ó. (2013b) Search-based refactoring detection, In: *Proceedings of the 15th annual conference companion on Genetic and evolutionary computation 2013*, ACM, pp. 205-206.

Mkaouer, M.W., Kessentini, M., Bechikh, S., Cinnéide, M.Ó. and Deb, K. (2015) On the use of many quality attributes for software refactoring: a many-objective search-based software engineering approach. In: *Empirical Software Engineering* [online]. pp.1-43.

Mkaouer, M.W., Kessentini, M., Bechikh, S., Deb, K. and Ó Cinnéide, M. (2014) High dimensional search-based software engineering: finding tradeoffs among 15 objectives for automating software refactoring using NSGA-III, In: *Proceedings of the 2014 conference on Genetic and evolutionary computation* 2014, ACM, pp. 1263-1270.

Ouni, A., Kessentini, M. and Sahraoui, H. (2013) Search-based refactoring using recorded code changes, Software Maintenance and Reengineering (CSMR), In: *2013 17th European Conference on 2013,* IEEE, pp. 221-230.

Ouni, A., Kessentini, M., Sahraoui, H. and Hamdi, M.S. (2012) Search-based refactoring: Towards semantics preservation, In: *Software Maintenance (ICSM), 2012 28th IEEE International Conference on* 2012, IEEE, pp. 347-356.

Santos Neto, Baldoino Fonseca dos, Ribeiro, M., Silva, V.T.d., Braga, C., Lucena, Carlos José Pereira de and Costa, E.d.B. (2015) AutoRefactoring. In: *Expert Systems with Applications: An International Journal* [online]. 42 (3), pp.1652-1664.

Sayyad, A.S. and Ammar, H. (2013) Pareto-optimal search-based software engineering (POSBSE): A literature survey, In: *Realizing Artificial Intelligence Synergies in Software Engineering (RAISE), 2013 2nd International Workshop on* 2013, IEEE, pp. 21-27.

Simons, C., Singer, J. and White, D.R. (2015) Search-based refactoring: Metrics are not enough. In: *Search-Based Software Engineering* [online]. Springer, pp.47-61.