ISSN 1870-4069

Genetic Programming in Software Engineering

Leslie Loaiza-Meseguer, Angel J. Sánchez-García,
Jorge Octavio Ocharán-Hernández

Universidad Veracruzana,
Facultad de Estadística e Informática,
Mexico

leslielm63@gmail.com, {angesanchez, jocharan}@uv.mx

Abstract. Industry 4.0 has led to automatic optimization of process improvements. Software Engineering is present in all the phases of the Software Development Life Cycle, implying a systematic and disciplined process of development. Nowadays there are optimization problems within the phases and activities of Software Engineering, problems that can be solved with the application of Genetic Programming. The purpose of this Systematic Literature Review is to analyze the current state of the application of genetic programming in Software Engineering by collecting the phases and activities of software development where genetic programming has been used and summarizing the advantages of using this technique. Thanks to this research work we found the way in which genetic programming has been applied previously and the advantages that its application has both in functional and non-functional properties. In addition, the utility that it has for a software engineer to use this technique as an automation tool in the process of software development was found.

Keywords: Genetic programming, software engineering, systematic literature review, optimization.

1 Introduction

Industry 4.0 seeks to improve processes and products through the incorporation of new technologies, cloud computing, the Internet of Things and Artificial Intelligence, among others. Software Engineering, for other hand, seeks to develop computer systems through a systematic, disciplined and orderly process in order to obtain a quality product and reduce the number of defects.

This implies that Software Engineering is present in all the phases of the life cycle of a software project [1]. Nowadays there are optimization problems within the phases and activities of Software Engineering that need to be solved since, during the construction of a project, several factors can be found that negatively influence its performance, production time, and reliability, among other aspects [2].

Genetic programming is applicable and effective for a wide variety of problems that arise in a wide variety of fields, mainly for the development of computer programs that perform a user-defined task. In addition, this technique is able to take advantage of the exponential increase in available computational power and solve optimization problems

Table 1. Research questions.

Quesition	Motivation
RQ1.- In which phases of software development has genetic programming been used?	The purpose of this question is to know the phases of software development in which genetic programming has been used to identify promising areas of the use of this technique or its variants.
RQ2.- In which activities of the software development phases has genetic programming been used?	It is important to identify which are the specific activities of each of the Software Engineering phases where genetic programming has been applied to promote improvements in software engineers.
RQ3.- What are the advantages of using genetic programming?	It is important to know about the benefits of applying genetic programming and why to use it in the different Software Engineering activities.

Table 2. Keywords and synonyms identified.

Concept	Synonyms
Software Engineering	
Genetic programming	GP

[3]. In Software Engineering, genetic programming has been used to represent code structures. It is reported that there are problems in the construction phase that have been addressed with optimization algorithms [4], specifically with genetic programming for code refactoring [5].

However, genetic programming is not limited to the coding phase, as it has impacted activities such as Software reliability [5], code repair [6], defect prediction [7], among others. With the above, it can be seen that the application of genetic programming supports Software Engineers, providing them with tools that help them to increase the efficiency of their work.

This paper is organized as follows: Section 2 describes the background as well as related work. In Section 3, the method used to carry out this research work is detailed. Section 4 presents the results obtained. Finally, section 5 draws the conclusions and future work.

2 Background and Related Work

Software engineering activities employ economic and human resources and involve the investment of time. Whether a software development project is successful or not depends entirely on the human factor[2], which is found in different phases of software development, such as design, construction and maintenance.

As a consequence of these factors, artificial intelligence techniques have been used to help reduce time, costs and human errors. Recent works have shown that Artificial Intelligence can bring benefits in each of the phases of software development, for example, requirements analysis [8], design [9], coding and testing[10].

It is reported that there are problems in the construction phase that have been addressed with optimization algorithms [4] specifically with genetic programming for

Table 3. Data source.

Database	Website
IEEE Xplore	https://ieeexplore.ieee.org/Xplore/home.jsp
Science direct	https://www.sciencedirect.com/
ACM	https://dl.acm.org/
Springer Link	https://link.springer.com/

Table 4. Inclusion criteria.

ID	Description
IC1	Studies with full access.
IC2	Studies published between 2017 and 2022
IC3	Studies that in the title or abstract allude to any of the phases or activities of software engineering.
IC4	Studies that contain in the abstract indications of answering at least one research question.

code refactoring [5]. Genetic programming is an extension of the traditional genetic algorithm in which each individual in the population is represented as a program variant (patched program).

The program variant is generated using one of the operations of the genetic algorithm: mutation and crossover. The acceptability of each variant is calculated through a user-defined fitness function.

These program variants that obtain high fitness scores are selected for the next evolution. The evolution process will continue again and again until a valid patch is found [6].

In a manual search of related work, no Systematic Literature Review on the application of genetic programming in Software Engineering was identified. For this reason, the main objective of this research is to describe the current state of the use of genetic programming in each of the phases of the software development life cycle, emphasizing the benefits for (although not limited to) software engineers, software developers and testers.

3 Research Method

The method used to carry out this systematic review of the literature is based on the guidelines proposed by Kitchenham & Charters [11], which are described below.

3.1 Research Questions

The research questions that guided this systematic review are shown in Table 1.

Table 5. Exclusion Criteria.

ID	Description
EC1	Studies in a language other than English.
EC2	Studies that are book chapters, presentations, abstracts or technical reports.
EC3	Repeated or duplicated studies.

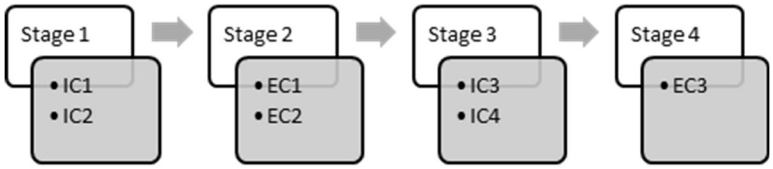

Fig. 1. Primary study selection procedure.

3.2 Search Strategy and Data Sources

The terms used to search for the primary studies are defined in Table 2. Being an exploratory study, each phase of software development (such as requirements, design, coding or testing) was not placed in the search string.

In addition, the term "Software Engineering" was added, which implies a development process, instead of putting only the term "software" since studies referring to the use of software to test genetic programming in different areas could be included. The search string used is based on the search terms defined above and is made up as follows:

"Software Engineering" AND "Genetic programming"

Table 3 shows the databases that were selected for the search of the primary studies, as well as their e-mail addresses.

3.3 Selection of Primary Studies

The inclusion and exclusion criteria described in Table 4 and 5 are intended to determine which studies will be included or excluded for the elaboration of this study.

3.4 Selection Procedure

The primary study selection procedure consists of four stages. Fig. 1 shows in detail the primary study selection criteria previously defined in section 3.3 that are applied in each of these stages. Table 6 shows in detail the results of each database during the 4 stages.

The list of references of the 41 primary studies selected can be found in [12]. The template used to extract data from each primary study can be found in [13]. The questions defined in order to evaluate the quality of primary studies can be found in [14].

Table 6. Application of inclusion and exclusion criteria by stage.

Stage	IEEE Xplore	ACM	SpringerLink	ScienceDirect	Total
Initial search	182	613	2,741	476	4,012
Stage 1	28	235	226	31	520
Stage 2	25	196	190	17	428
Stage 3	12	18	14	2	46
Stage 4	12	13	14	2	41

4 Results

As a result of the application of stage 4 of the primary study search selection process, of the 41 resulting studies were identified. The most came from SpringerLink (34%), followed by ACM (32%), IEEE Xplore (29%) and ScienceDirect (5%). Of these selected studies, 59% correspond to articles published in journals, while 41% are conference papers, as it is shown in Fig. 2.

The distribution of primary studies by year of publication was also identified, with the majority of studies coming from 2017 and 2021, as it is shown in Fig. 3.

4.1 RQ1.- In Which Phases of Software Development has Genetic Programming Been Used?

As can be seen in Fig. 4, the phase that has had the most applications of genetic programming, is the construction phase, occupying 68% of the total selected studies.

It was found that the tree structure used by genetic programming to represent its individuals (computer programs) is very useful for the construction of software, since it allows the creation of a new code from an existing code [15]; thus, removing branches from one tree to insert them into another [16], which promotes the improvement of both functional and non-functional properties [17], automatic code generation[18], as well as code reuse and restructuring[19].

Genetic programming was found to be a tool that facilitates pattern identification [20], this property allows us to identify code smells [21], locate faults [22] and patch generation [23]; the latter enables automatic program repair [24]. On the other hand, the Testing phase is mentioned in 20% of the articles.

Wei et al. [25] again points out the ability of genetic programming for pattern identification, which, according to their study, allows the identification of the worst-case scenario in the execution of a software, as well as the detection of vulnerabilities and performance errors.

Other authors propose that the application of genetic programming allows the automation of black box testing [26] and the evaluation of graphical interfaces [27]. Regarding the Design phase, only 7% of the articles were found to mention it. It was found that genetic programming helps the automation of both prototype generation [28] and modeling of software product lines [29].

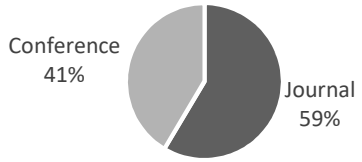

Fig. 2. Selected primary studies by publication type.

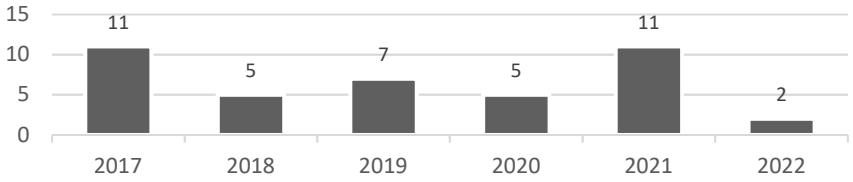

Fig. 3. Selected primary studies by year.

Finally, the Planning phase was found to be present in 3% of the articles and the Maintenance phase in 2%, indicating a lack of information on the areas of opportunity for genetic programming in these phases.

4.2 RQ2.- In Which Activities of the Software Development Phases Has Genetic Programming Been Used?

Twenty-seven papers that mention activities of software development phases involving the application of genetic programming were found. In the Planning phase, by applying genetic programming in the restructuring of plans, initializing the population of individuals with existing plans, we can reuse their information to carry out the generation of new plans [30].

In the Design phase it was found that 67% of the items correspond to prototype generation. This activity is achieved by automating the definition of basic elements and the way to combine them, to be subsequently composed and tested with real users and thus find the optimized compositions [31].

The 33% of the articles refer to the modeling of Software Product Lines (SPL). The automatic generation of the generic models used by this activity is possible by means of an initial population of these and the calculation of the set of valid characteristics for each one, to subsequently apply genetic programming to them [29].

In the Construction phase we were able to identify that 32% of the items correspond to the activity of automatic program repair. This activity aims to generate error repairs without human intervention, without the need for special instrumentation or annotations in the source code [32].

This application searches and generates modifications from an abstract syntax tree that can patch a bug in the underlying program and creates new program variants by mutation and crossover [15].

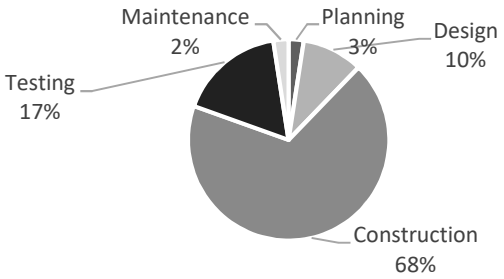

Fig. 4. Selected primary studies by phase.

The 32% of the articles mention automatic coding of programs, for this activity it is important to mention that one of the most relevant applications of genetic programming is the technique called program synthesis, which is able to automatize the coding of programs by automatically generating source code in a programming language, that maintains the constraints of a predefined specification [32].

It was determined that 31% of the articles correspond to the identification of bugs in the software, which is achieved through the identification of recurrences [24] and sequence-by-sequence learning [15].

These properties are useful in code smells identification [21] and fault localization [17]. Finally, 5% of the articles mention that genetic programming supports code restructuring by identifying patterns within the code, in order to subsequently optimize its fragments and reuse them to create new code [19].

In the testing phase, it was found that 67% of the articles talk about black box testing, to which the application of genetic programming is useful to discover local variables, actions performed on output variables, counting loops and while loops, due to the ability of genetic programming to discover a functional relationship between data features and to group them into categories [26].

Also, the application of genetic programming in black box testing is useful when identifying worst-case execution as well as vulnerabilities in programs by identifying patterns in data inputs [25].

The 33% of the articles allude to interface evaluation, where genetic programming allows the automatic generation of rules to evaluate their quality, providing previously defined quality metrics, context criteria and list of possible types of problems, taking advantage of the principle of genetic programming where individuals adapt to their environment through mutation and crossover [27].

4.3 RQ3.- What are the Advantages of Using Genetic Programming?

Based on the answers to questions RQ1 and RQ2, it was found that thanks to the mutation and crossover operators, the principles of biological evolution on which genetic programming is based and its ability to identify patterns, there are numerous advantages in the different phases and activities of software development.

The use of genetic programming in the Planning phase, serves for the restructuring of plans, reducing the costs of operating in complex environments of change and uncertainty, by adapting autonomously to change in the pursuit of its quality objectives

[30]. In the Design phase, prototyping [31] and modeling of software product lines [29] can be automated using genetic programming, greatly improving the performance of these activities [28].

In the Construction phase, the application of this technique helps in the automation of different activities such as program repair [32], bug identification [24] and code restructuring [19]. Obtaining the improvement of both functional and non-functional properties, such as code size, execution time or memory consumption [17].

Furthermore, genetic programming is a technique that has great flexibility since it offers the possibility of handling a large number of individuals and of reworking the solutions obtained by relaunching a new evolution from one or more previously obtained solutions, so that, with its application, the activity of evaluating graphical interfaces can be automated and thus optimize the process involved [17].

All this together helps a software engineer to do his job efficiently since it eliminates the manual part of his work and increases the quality of his results.

5 Conclusions and Future Work

This paper identified the phases and activities of software development in which genetic programming has been applied, as well as the advantages of using it. A systematic Literature Review was carried out where the selection process of primary studies was divided into 4 parts where, after applying the previously defined selection criteria, 41 primary studies were obtained as a result.

Subsequently, after carrying out a preliminary data synthesis, the primary studies were classified into 5 development phases: Planning, Design, Construction, Testing and Maintenance. The software development phase where genetic programming proved to have more applications is the Construction phase with 28 articles, followed by the Testing phase with 8 articles, the Design phase with 3 articles and the Planning and Maintenance phases with 1 article each.

With respect to the research questions, thanks to the data synthesis, it was found the way in which genetic programming has been previously applied, the advantages of its application in both functional and non-functional properties, in addition to the usefulness for a software engineer to use this technique as an automation tool in the different processes that exist at the time of software development. As a result, the objectives of the research work were achieved.

It was found that genetic programming has a great relationship with well-established areas, for example, Program synthesis, which has a strong impact on new fields such as Genetic improvement. This field of science uses genetic programming to correct bugs in software and improve both functional and non-functional software requirements [19]. Therefore, as future work, we will seek to identify the applications and advantages of these areas.

References

1. Boehm, B.: Software engineering. IEEE Transactions on Computers, pp. 1226–1241 (1976) doi: 10.1109/TC.1976.1674590

2. Yanyan, Z, Renzuo, X.: The basic research of human factor analysis based on knowledge in software engineering. In: 2008 International Conference on Computer Science and Software Engineering, pp. 1302–1305 (2008) doi: 10.1109/CSSE.2008.219
3. Koza, J. R.: Genetic programming: On the programming of computers by means of natural selection, MIT press (1992)
4. Robles-Aguilar, A, Ocharán-Hernández, J. O., Sánchez-Garcia, A. J., Limon, X.: Software design and artificial intelligence: A systematic mapping study. In: 2021 9th International Conference in Software Engineering Research and Innovation (CONISOFT), pp. 132–141 (2021) doi: 10.1109/CONISOFT52520.2021.00028
5. Chen, H, Zhang, Y, Zhao, J.: Improved genetic programming model for software reliability. In: 2009 International Asia Symposium on Intelligent Interaction and Affective Computing, pp. 164–167 (2009) doi: 10.1109/ASIA.2009.38
6. Qi, Y., Mao, X., Lei, Y., Dai, Z., Wang, C.: Does genetic programming work well on automated program repair? In: 2013 International Conference on Computational and Information Science, pp. 1875–1878 (2013) doi: 10.1109/ICCIS.2013.490
7. Rathore, S. S., Kuamr, S.: Comparative analysis of neural network and genetic programming for number of software faults prediction. In: 2015 National Conference on Recent Advances in Electronics and Computer Engineering (RAECE), pp. 328–332 (2015) doi: 10.1109/ RAECE.2015.7510216
8. Ernst, N. A., Gorton, I.: Using AI to model quality attribute tradeoffs. In: 2014 IEEE 1st International Workshop on Artificial Intelligence for Requirements Engineering (AIRE), pp. 51–52 (2014) doi: 10.1109/AIRE.2014.6894856
9. Wangoo, D. P.: Artificial intelligence techniques in software engineering for automated software reuse and design. In: 2018 4th International Conference on Computing Communication and Automation (ICCCA), pp. 1–4 (2018) doi: 10.1109/CCAA.20 18.8777584
10. Xie, T.: The synergy of human and artificial intelligence in software engineering. In: 2013 2nd International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE), pp. 4–6 (2013) doi: 10.1109/RAISE.2013.6615197
11. Kitchenham, B., Charters, S.: Guidelines for performing systematic literature reviews in software engineering. Keele University and Durham University Joint Report (2007)
12. Appendix A Primary studies references: https://docs.google.com/document/d/11uCIfSXL-hxb1xVBhpcbCOF1T0rUyC99TBMfdck9R5c/edit?usp=sharing
13. Appendix B Data extraction template: https://docs.google.com/document/d/1JQ4 x7up_ZlkXBol26z500ff-wK4DlH14IWh274JBJp8/edit?usp=sharing
14. Appendix C Quality assessment questions: https://docs.google.com/document/d/1RQuy Ztujhr0wHsWES8dR2nswFO25pIjLozUTfKcxZao/edit?usp=sharing
15. Li, D., Wong, W. E., Jian, M., Geng, Y., Chau, M.: Improving search-based automatic program repair with neural machine translation. IEEE Access, vol. 10, pp. 51167–51175 (2022) doi: 10.1109/ACCESS.2022.3164780
16. Langdon, W. B., Lam, B. Y., Modat M., Petke, J., Harman, M.: Genetic improvement of GPU software. Genetic Programming Evolvable Machines, vol. 18, pp. 5–44. (2017) doi: 10.1007/s10710-016-9273-9
17. Sohn, J., Yoo, S.: Empirical evaluation of fault localization using code and change metrics. IEEE Transactions on Software Engineering, vol. 47, no. 8, pp. 1605–1625 (2021) doi: 10.1109/TSE.2019.2930977
18. Miller, J. F.: Cartesian genetic programming: Its status and future. Genetic Programming Evolvable Machines, vol. 21, pp. 129–168 (2020) doi: 10.1007/s10710-019-09360-6
19. Krauss, O.: Genetic improvement in code interpreters and compilers. In: Proceedings Companion of the 2017 ACM SIGPLAN International Conference on Systems, Programming, Languages, and Applications: Software for Humanity, pp. 7–9 (2017) doi: 10.1145/3135932.3135934

20. Huppe, S., Saied, M. A., Sahraoui, H.: Mining complex temporal API usage patterns: An evolutionary approach. In: 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C), pp. 274–276 (2017) doi: 10.1109/ICSE-C.2017.147
21. Kessentini, M., Ouni, A.: Detecting android smells using multi-objective genetic programming. In: 2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft), pp. 122–132 (2017) doi: 10.1109/MOBILE Soft.2017.29
22. Kim, Y., Mun, S., Yoo, S., Kim, M.: Precise learn-to-rank fault localization using dynamic and static features of target programs. ACM Transactions on Software Engineering and Methodology, vol. 28, no. 4, pp. 1–34 (2019) doi: 10.1145/3345628
23. Cao, H., Liu, F., Shi, J., Chu, Y., Deng, M.: Automated repair of Java programs with random search via code similarity. In: 2021 IEEE 21st International Conference on Software Quality, Reliability and Security Companion (QRS-C), pp. 470–477 (2021) doi: 10.1109/Q RS-C55045.2021.00075
24. Yuan, Y., Banzhaf, W.: Toward better evolutionary program repair. ACM Transactions on Software Engineering and Methodology, vol. 29, pp. 1–53 (2020) doi: 10.1145/3360004
25. Wei, J., Chen, J., Feng, Y., Ferles, K., Dillig, I.: Singularity: Pattern fuzzing for worst case complexity. In: Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pp. 213–223 (2018) doi: 10.1145/3236024.3236039
26. Drusinsky, D.: Reverse engineering concurrent UML state machines using black box testing and genetic programming. Innovations in Systems Software Engineering, vol. 13, pp. 117–128 (2017) doi: 10.1007/s11334-017-0299-9
27. Ines, G., Makram, S., Mabrouka, C., Mourad, A.: Evaluation of mobile interfaces as an optimization problem. Procedia Computer Science, vol. 112, pp. 235–248 (2017) doi: 10.1016/j.procs.2017.08.234
28. Valencia-Ramírez, J. M., Graff, M., Escalante, H. J., Cerda-Jacobo, J.: An iterative genetic programming approach to prototype generation. Genetic Programming Evolvable Machines, vol. 18, pp. 123–147 (2017) doi: 10.1007/s10710-016-9279-3
29. Vescan, A., Pintea, A., Linsbauer, L., Egyed, A.: Genetic programming for feature model synthesis: A replication study. Empirical Software Engineering, vol. 26, no. 58 (2021) doi: 10.1007/s10664-021-09947-7
30. Kinneer, C., Coker, Z., Wang, J., Garlan, D., Le-Goues, C.: Managing uncertainty in self-adaptive systems with plan reuse and stochastic search. In: Proceedings of the 13th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), pp. 40–50 (2018) doi: 10.1145/3194133.3194145
31. Salem, P: User interface optimization using genetic programming with an application to landing pages. In: Proceedings of the ACM Human-Computer Interaction, vol. 1, pp. 1–17 (2017) doi: 10.1145/3099583
32. Sobania, D., Rothlauf F.: A generalizability measure for program synthesis with genetic programming. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 822–829 (2021) doi: 10.1145/3449639.3459305