A Survey on Computational Intelligence Applications in Software Engineering and its Data

K. Narendar Reddy[1] and Kiran Kumar Jogu[2]
[1]CVR College of Engineering/ Dept. of CSE, Hyderabad, India
Email: knreddy@cvr.ac.in
[2]IBM India Software Lab, Hyderabad, India
Email: kirankumarj9@gmail.com

Abstract: **Ubiquitous software has become an indispensable technology for science, engineering, and business. Software is everywhere, as a standalone system, or part of a new technology, or as a service in the cloud. Hence, it has paramount importance. As size and complexity of software systems are increasing, software engineering problems such as software effort estimation, software testing, software defect prediction, software project scheduling, software reliability maximization, software module clustering, and software maintenance have become more difficult to handle. In order to reduce the high cost of performing software engineering activities and to increase software quality and reliability, computational intelligence techniques are being used for problem solving using research oriented approaches and for decision-support. Computational intelligence has been used in different fields for a long time. There has been a recent surge in interest in the application of computational intelligence techniques in software engineering. Search based software engineering and machine learning for software engineering are the areas of computational intelligence research which have been showing promising results in this context. Search based software engineering reformulates software engineering problems as optimisation problems, and then using optimisation algorithms problems are solved. Software engineering produces lot of data related to software, like effort estimates, source code, test cases, data on bugs and fixes, version data, and metrics data. As part of analytics on software data, machine learning techniques are used to solve some software engineering problems and for effective decision making. The objective of this survey paper is to identify software engineering problems and applications of computational intelligence techniques to solve those problems. In survey, computational intelligence applications for solving different software engineering problems are identified and presented. In this paper, some research questions which indicate research directions and some possible research topics are presented. New research issues and challenges posed by the hard problems in software engineering could stimulate further development of new theories and algorithms in computational intelligence.**

Index Terms – **Computational Intelligence, Software Engineering Problems, Search Based Software Engineering, Optimisation Techniques, Machine Learning, Software Data Analytics.**

I. INTRODUCTION

As size and complexity of software systems are increasing, software engineering problems have become more difficult to handle. In order to increase software reliability and to reduce the high cost of performing software engineering tasks, computational intelligence techniques and algorithms are being used as problem solving, decision-support, and research oriented approaches. As per the definition in [1], "Computational intelligence is the study of adaptive mechanisms to enable or facilitate intelligent behavior in complex and changing environments. As such, computational intelligence combines artificial neural networks, evolutionary computing, swarm intelligence and fuzzy systems". Software sizes are becoming bigger. The complexity of software is increasing with size non-linearly, and in addition, software complexity further increasing as it is changing very rapidly to keep pace with changing user business dynamics and needs. Increased software complexity poses many problems. Computational intelligence is the appropriate vehicle to address software problems. In this paper, two areas of computational intelligence which are showing promising results in software engineering are considered for review. They are: 1) "Search-Based Software Engineering" (SBSE). 2) Machine Learning for Software Engineering. "Search-Based Software Engineering (SBSE)" [5] reformulates software engineering problems as optimisation problems. These reformulated problems are solved using optimisation algorithms. Some of the software engineering problems which can be reformulated as optimization problems are; 1) software project scheduling with an aim to minimize cost and time of completion of different tasks, 2) test case design with an aim to maximize code coverage and bug detection, 3) test case design with an aim to minimize testing effort and maximize bug detection, 4) during starting of an iteration in WinWin spiral model identifying set of requirements maximizing user satisfaction and probability of completion in given time and cost. Similarly, many more problems with different optimization criteria can be reformulated as optimization problems. Some of the search based optimisation techniques which are being used are: 1) Genetic programming 2) Genetic algorithms 3) Ant colonies 4) Particle swarm optimization 5) Hill climbing 6) Simulated annealing.

As per the definition in [2], "machine learning deals with the issue of how to build programs that improve their performance at some task through experience". Software systems process data, but software is data too [3]. Software engineering activities and different stake holders produce lot of data related to software. Machine learning makes use of software data to create machine learning models to solve software engineering problems. Some of the software

doi:10.32377/cvrjst1114

engineering problems which can be addressed by machine learning models are; 1) predicting effort for the next project based on machine learning model, which is created using the data from previous completed projects, 2) predicting project duration for the next project based on machine learning model, which is created using the data from previous completed projects, 3) software defect prediction can be made using the defect prediction models created using previous versions of software.

Major types of machine learning include; decision trees, concept learning, artificial neural networks, reinforcement learning, Bayesian belief networks, genetic programming and genetic algorithms, instance-based learning, and analytical learning.

Some problems can be viewed either as optimization or machine learning problem. For example, test case design can be viewed as an optimization or machine learning problem. That is where research comes into picture to figure out which approach and technique is better.

The aim of this paper is to carry out basic survey (not exhaustive survey) to find research openings in the context of application of computational intelligence techniques in software engineering.

II. Software Engineering Problems And Software Data

An iteration in a software development model consists following activities (phases):

1. Software requirements collection, specification and planning phase, 2. Design phase
3. Implementation and testing phase

The maintenance activity takes place in parallel with development. While next version is under development previous deployed versions will be under maintenance. Some problems which are encountered frequently are identified and listed out in this paper. Some of the problems which are encountered during software engineering are identified by different authors in [3], [4], [5], [6]. [7], [8],[9],[10],[11]. They are categorized and listed below:

A. *Software Engineering Problems*

1. Software requirements collection, specification and planning phase:
   - Eliciting, recording of all functions and constraints
   - Ambiguity, completeness, conflicts in requirements, prototyping, requirements tracing
   - Cost and time estimation for the project
   - Tasks, dependencies, duration, resources for the project
2. Design phase:
   - High-level (architectural) - architectural design problems, modularity, coupling
   - Low-level (detailed) - algorithms selection, complexity of modules, cohesion
   - Design alternatives, inconsistencies in software design
3. Implementation and testing phase:
   - Software reuse
   - Source code searching
   - Integration method, defect prediction
   - Test case generation, test case prioritization
   - Prediction of test effectiveness
   - Bug management/triage, debugging
4 .Maintenance phase:
   - Software understanding and comprehension
   - Impact analysis, ripple effects during changes
   - Regression testing
   - Automatic software repair, quality enhancement
   - Reengineering legacy software, software module clustering
5. Problems related to umbrella activities:
   - Configuration management
   - Prediction of software quality and reliability
   - Classification of software components

B. *Software Data*

Lot of data are generated during software development and maintenance. Over the time from different projects the data populate databases in the range of terabytes and more. Data related to software are listed below.

- Source code, data on versions, code analysis data
- Test cases designed, test execution data
- Data on bugs and fixes
- Metrics data on size, design, code, testing, project, process, and maintenance
- Cost and schedule data
- Usage and run time data of deployed software
- User feed back

III. Computational Intelligence Techniques To Solve Software Engineering Problems

In this section computational intelligence techniques which are applied to software engineering problems are surveyed and presented. The survey is not exhaustive. This survey gives basis for further refinement in the subsequent work.

A. *Solving Software Engineering Problems with Optimization Techniques (SBSE)*

"Search based software engineering" reformulates software engineering problems as optimization problems. The problems and optimization techniques are discussed in detail in [4],[5],[8],[9],[12]. The optimization techniques are used by different researchers in solving different software engineering problems. Simulated annealing approach is used in solving many problems like improving software quality prediction[13], next release problem [14], program flaw finding[15]. Genetic programming applications can be found in software cost predictive modeling [16], in reliability modeling[17], and in model for software quality enhancement[18]. Software release planning [19] and software test data generation [20] are solved by genetic algorithms. Hill climbing technique is used to improve program structure by module clustering [21]. Hill climbing technique and genetic algorithm are

doi:10.32377/cvrjst1114

applied for regression test case prioritization [22]. Software modularization using hill climbing, simulated annealing, and genetic algorithm is presented in [23],[24]. "Search based software engineering techniques" use metrics as fitness functions in one form or the other [25]. Proposing new metrics for using in optimization techniques is also an important research area. There is little work on the combinations of search algorithms [26]. Research on exploring the potential of combinations of search algorithms has lot of scope. Bug detection using particle swarm optimization is given in [27]. Ant colony optimization techniques [28] and particle swarm optimization [29] techniques have potential to be used in solving software engineering problems and have not been used much in the literature. In addition to the papers listed out in this section, interested researcher on SBSE can look into the repository of papers on SBSE maintained by Y.Zhang [49].

B. Solving Software Engineering Problems with Machine Learning Techniques

Application of machine learning in the context of software engineering are discussed in [6], [7], [10], [11]. Solutions to different software engineering problems using machine learning techniques are attempted by many researchers. Instance based learning is used in component retrieval and reuse [6] and software project effort estimation [30]. Genetic programming is applied for understanding and validating user software requirements [31]. One of the supervised learning methods, "concept learning", is used to derive functional requirements from legacy software [6]. To predict effort required for software development, artificial neural networks & decision trees [32], bayesian analysis [33], artificial neural networks & naïve bayes classifier [34], and artificial neural networks [35] are applied and results are presented. For predicting software defects bayesian belief networks have found applications [36], [37], [6]. Detecting bad aspects of code and design are important. Because, it enables refactoring, to improve quality of code and design. Bayesian approach is used in detecting design and code smells [38]. Overview on techniques based on machine learning used for software engineering is given in [39]. Even though there is active research happening in machine learning applications in software engineering, more research is possible as machine learning has the potential [40].

IV. Discussion And Research Directions On Computational Intelligence Applications In Software Engineering

Standalone software, distributed software, or software as a service in the cloud need to be designed, coded, and tested before deployment. As part of evolution and maintenance (corrective, perfective, adaptive, and preventive) software is changed. During change software is not available for use. Change is applicable to all the above mentioned types of software. The computational intelligence techniques and research directions presented in this paper are applicable to standalone, distributed and cloud software as service.

The application of computational intelligence techniques in software engineering started more recently. Diversity, size and complexity of software systems are increasing. In addition, due to frequent changes to software systems to keep pace with changing user business requirements and personal needs, the software systems are getting deteriorated and becoming complex. Due to this scenario, the earlier easily solvable software engineering problems have now become more challenging. To address software engineering problems, researchers started exploring the application of computational intelligence techniques. Survey indicates results are encouraging and there is lot of potential in further research in this area. Computational intelligence techniques are grouped under two headings. They are; optimization techniques and machine learning techniques. Optimization techniques and machine learning techniques are listed in introduction section. Some techniques are used as optimization and machine learning techniques. Similarly, some problems can be viewed either as optimization or machine learning problem. For example, project estimates can be viewed as an optimization or machine learning problem. That is where research comes into picture to figure out which approach and technique is better. Some problems which are encountered frequently are identified and listed out in this paper.

A. Research Directions in Software Data Analytics

There are many research questions that need to be addressed related to software data analytics. These questions indicate the scope for research work in those lines. Some of them are listed below.

- What data and how to analyse the data to address a particular software engineering problem?
- How to integrate heterogeneous data? Much of the software data is unstructured, while some data is stored in structured format [3].
- Which algorithm is better for data analysis? Do we need to design a new algorithm for a particular situation?
- How to customize existing algorithms to suite to the data, problem and situation?

B. Some of the More Specific Possible Research Topics in the Category of Software Data Analytics

1) Project cost and time estimation. Design and use of hybrid models to predict estimates.
2) Building machine learning models using metrics data and predict the defects and design quality for the software that is going to be deployed. Prediction will help to know whether further testing is required or not.
3) Building machine learning models using source code and source code analysis data to design test cases and prioritizing test cases.
4) Building machine learning models using source code and source code analysis data for automatic bug repair.

5) Building machine learning models using coupling data to predict regression testing effort. Prediction will help to make decision whether change can be implemented or it has to be deferred.
6) Building machine learning models for software testing [40].
7) Building machine learning models using bug data for defect prediction.
8) Building machine learning models using version data to predict software maturity index.

Research on software data analytics requires software data. But, from where to get the data. There are some options available. They are:

1) Use software data from reliable public domain sites contributed by research groups.
   Example: http://promisedata.org/repository [41] and other repositories [42].
2) Lot of software data are available from the sites of standard open source tools.
   Example: Software data on different plug-ins to eclipse
3) Data published in research papers and text books.
   Example: COCOMO database containing 63 projects published in Boehm's text book [43].
4) Create small programs with problems and parameters related to your research interest. Research findings based on small size programs need to be justified that results are in fact applicable to bigger size programs (scalability).
5) Use synthetic data. Synthetic data are "any production data applicable to a given situation that are not obtained by direct measurement" according to the McGraw-Hill Dictionary of Scientific and Technical Terms. Synthetic data are generated to meet specific needs or certain conditions. This can be useful when designing any type of technique, algorithm, or a method, because the synthetic data are used as a simulation or as a theoretical value, situation, etc.

C. Research Directions in the Application of Optimization Techniques

There are many research questions that need to be addressed related to solving software problems using optimization techniques ("Search Based Software Engineering"). These questions indicate the scope for research work in those lines. Some of them are listed below.

- What are the benefits of solving a software engineering problem using optimization techniques? Can the benefits be quantified? For example: Test case design to maximize code coverage. It should be quantified to indicate how much extra code coverage is achieved by using optimization techniques over traditional techniques.
- For a particular optimization criterion or criteria which algorithms perform better? Why?
- Can we customize the algorithms for better results?
- Can we propose new metrics which can be input for existing optimization techniques for solving a problem? How these metrics are different?
- Can we create an hybrid algorithm (from existing algorithms)? But, why?
- Can we identify a new problem to which existing optimization techniques can be applied?
- In solving a problem, use of optimization techniques versus machine learning models (Software data analytics). Why the results are not same? Justify.

D. Some of the More Specific Possible Research Topics in the Category of Application of Optimization Techniques in Software Engineering

1) Modular design (clustering) with an aim to reduce change propagation due to ripple effects during maintenance and regression testing.
2) Modular design with an aim to maximize cohesion and minimize coupling. Application of search based optimization techniques in design can be found in [44].
3) Refactoring software design during an iteration in a sprint of scrum agile process model with an aim to reduce coupling and to increase cohesion. Refactoring using SBSE is presented in [45].
4) Test case design for scrum agile process with an aim to reduce testing time.
5) Test case design with an aim to maximize code coverage and bug detection.
6) Test case design with an aim to maximize bug detection and minimize testing effort. Some work on test case design is given in [46].
7) Prioritising the test cases with an aim to reduce testing effort with same bug coverage. Recent work on prioritizing test cases can be found in [47].
8) Test case design to detect critical defects [48].
9) Debugging with an aim to minimize the time to locate the cause for the bug. Code, cohesion and coupling metrics can be used for this purpose.
10) Reengineer the software with an aim to minimize coupling and maximize cohesion.
11) Version management with an aim to minimize redundancy and retrieval time of a required component.
12) During reuse, aiming to identify and retrieve most appropriate component for a given situation.
13) Model based testing with an aim to maximize the likelihood of early detection of defects and estimation of software testing effort.

For carrying out research using optimization techniques in software engineering requires software (program) as input. This program can be developed for research purpose or any free open source tool can be downloaded.

Computational Intelligence initial subjects of interest are; fuzzy systems, neural networks, evolutionary computation, and swarm intelligence. But, different authors of research papers treat computational intelligence as an umbrella, under which more and more algorithms, techniques, and methods are slowly added, as advancement is taking place as part of basic research and applied

doi:10.32377/cvrjst1114

research. That is how computational intelligence subjects of interest have grown.

V. CONCLUSIONS

The software engineering problems which can be addressed by computational intelligence are identified from different publications. Some of them are: software effort estimation, software testing, software defect prediction, software project scheduling, software reliability maximization, software module clustering, and software maintenance. Different computational intelligence techniques which can be used to solve some of the software engineering problems are surveyed and given. As size and complexity of software systems are increasing, software engineering problems have become more difficult to handle. In this context, based on the survey, it is found that computational intelligence techniques which include optimization techniques and software data analytics play a significant role in solving software engineering problems and developing high quality software products with low maintenance cost. Based on the survey, some research questions and research topics related to software data analytics and application of optimization techniques to software engineering problems are identified and presented. Research topics indicate computational intelligence in software engineering and its data has tremendous scope for aspiring researchers.

Detailed research directions for software in the cloud will be presented in future work. Some of the research directions in this context are; virtualization, multi-tenant modeling, testing as a service (TaaS), design of cloud services user interface, design of cloud computing metrics, performance testing of SaaS, security testing of SaaS, and architectures for dynamic scalability.

REFERENCES

[1] Andries P.Engelbrecht, Computational intelligence: An introduction, Wiley, 2002.

[2] T.Mitchel, Machine learning, McGraw-Hill, 1997.

[3] Andrian Marcus and Timothy Menzies, "Software is data too", FoSER, ACM, pp.229-231, 2010.

[4] Witold Pedrycz, "Computational intelligence as an emerging paradigm of software engineering', SEKE '02, ACM, pp. 7-14, 2002

[5] Mark Harman, "The current state and future of search based software engineering", Proceedings of Future of software engineering (FOSE '07), IEEE, pp.342-357, 2007

[6] Du Zhang, " Applying machine learning algorithms in software development" Proceedings of Monterey workshop on modeling software system structures in a fastly moving scenario, pp. 275-290, Italy, 2000

[7] Mark Harman, " The role of artificial intelligence in software engineering", RAISE '12 - Proceedings of the first international workshop on realizing AI synergies in software engineering, IEEE, pp. 1-6, 2012

[8] Mark Harman, S.A.Mansouri, and Y.Zhang, "Search-based software engineering: Trends, Techniques and applications", ACM Computing surveys, 45(1), Article no. 11, 2012

[9] Pedrycz, W and Peters J.F. (eds), Computational Intelligence in software engineering, World Scientific, 1998.

[10] Du Zhang and Jeffrey J.P.Tsai,"Machine Learning and Software Engineering",Proceedings of 14th IEEE International Conference on Tools with Intelligence, pp.22-29,2002

[11] Du Zhang, J.J.P.Tsai, Machine learning applications in software engineering, World Scientific, 2005

[12] Ilhem Boussaid, Julien Lepagnot, and Patrick Siarry,"A survey on optimization metaheuristics', International journal of Information Sciences, pp.82-117, March 2013

[13] S. Bouktif, H. Sahraoui, and G. Antoniol, "Simulated annealing for improving software quality prediction", In GECCO 2006: Proceedings of the 8th annual conference on Genetic and evolutionary computation, ACM, Volume 2, pp. 1893–1900, 2006.

[14] M. Harman, K. Steinhofel, and A. Skaliotis, "Search based approaches to component selection and prioritization for the next release problem", In 22nd International conference on software maintenance (ICSM 06), 2006.

[15] N. Tracey, J. Clark, and K. Mander, "Automated program flaw finding using simulated annealing", In International symposium on software testing and analysis (ISSTA 98),pp. 73–81, 1998.

[16] J. J. Dolado,"On the problem of the software cost function", Information and software technology, 43(1):61–72, 2001.

[17] Eduardo Oliveira Costa, Aurora Trinidad Ramirez Pozo, and Silvia Regina Vergilio," A genetic programming approach for software reliability modeling", IEEE Transactions on reliability, 59(1), pp.222-230, 2010

[18] Taghi M. Khoshgoftaar, Yi Liu, and Naeem Seliya," A multiobjective module-order model for software quality enhancement", IEEE Transactions on evolutionary computation, 8(6), pp. 593-608, 2004

[19] D. Greer and G. Ruhe,"Software release planning: an evolutionary and iterative approach", Information and software technology, 46(4), pp.243–253, 2004.

[20] Christoph C.Michael, Gary McGraw, and Michael A.Schatz,"Generating software test data by evolution", IEEE Transactions on software engineering, 27(12), pp.1085-1110, 2001

[21] Kata Praditwong,Mark Harman, and Xin Yao," Software module clustering as a multi-objective search problem", IEEE Transactions on software engineering, 37(2), pp.264-282, 2011

[22] Z.Li,M.Harman, and R.Hierons,"Search algorithms for regression test case prioritization", IEEE Transactions on software engineering, 33(4), pp.225-237, 2007

[23] S. Mancoridis, B. S. Mitchell, C. Rorres, Y.-F. Chen, and E. R. Gansner, "Using automatic clustering to produce high level system organizations of source code", In International workshop on program comprehension (IWPC'98), IEEE, pp. 45–53, 1998.

[24] S.Mancoridis, B. S.Mitchell, Y.-F. Chen, and E. R. Gansner, "Bunch: A clustering tool for the recovery and maintenance of software system structures", In IEEE International conference on software maintenance, pp. 50–59, 1999.

[25] M. Harman and J. Clark," Metrics are fitness functions too", In 10th International software metrics symposium (METRICS 2004),, IEEE, pp. 58–69, 2004.

[26] K.Mahdavi, M. Harman, and R.M. Hierons, "A multiple hill climbing approach to software module clustering", In IEEE International conference on software maintenance, pp.315–324, 2003.

[27] Arun Reungsinkonkarn and Paskorn Apirukvorapinit, "Bug detection using particle swarm optimization with search space reduction", ISMS '15 Proceedings of the 2015 6th International conference on Intelligent Systems, Modelling and Simulation, IEEE, pp.53-57, 2015

[28] M. Dorigo and C. Blum,"Ant colony optimization theory: A survey", Theoretical computer science,344(2-3),pp.243–278, 2005.

[29] X. Zhang, H. Meng, and L. Jiao," Intelligent particle swarm optimization in multiobjective optimization", In IEEE Congress on Evolutionary Computation, volume 1, pp. 714–719, 2005.

[30] M. Shepperd and C. Schofield, "Estimating software project effort using analogies", IEEE Transactions on software engineering, 23(12), pp. 736-743, 1997.

[31] M. Kramer, and D. Zhang, "Gaps: a genetic programming system," Proc. of IEEE International conference on computer software and applications (COMPSAC 2000).

[32] K. Srinivasan and D. Fisher, "Machine learning approaches to estimating software development effort," IEEE Transactions on software engineering, 21(2), pp. 126-137, 1995.

[33] S. Chulani, B. Boehm and B. Steece, "Bayesian analysis of empirical software engineering cost models," IEEE Transactions on software engineering, 25(4), pp. 573-583,1999.

[34] Jyoti Shivhare and Santanu Ku.Rath, "Software effort estimation using machine learning techniques", Proceedings of the 7th India Software Engineering Conference (ISEC'14), ACM, Article No. 19, 2014.

[35] C. Mair, G. Kadoda, M. Lefley, K. Phalp, C. Schofield, M. Shepperd, and S. Webster, "An investigation of machine learning based prediction systems", The Journal of Systems and Software, 53(1), pp.23–29, 2000.

[36] N. Fenton and M. Neil, "A critique of software defect prediction models," IEEE Transactions on software engineering, 25(5), pp. 675-689,1999.

[37] V. U. B. Challagulla, F. B. Bastani, I.-L. Yen, and R. A. Paul, "Empirical assessment of machine learning based software defect prediction techniques", International journal on artificial intelligence tools, 17(2),pp. 389–400, 2008.

[38] F.Khomh,S.Vaucher,Y.G,Gueheneuc, and H.A.Sahraoui," A Bayesian approach for the detection of code and design smells", in Proc. of Int. Conf. Quality Software, pp.3015-314, 2009 .

[39] T. Menzies, "Practical machine learning for software engineering and knowledge engineering," in Handbook of Software Engineering and Knowledge Engineering. World-Scientific, December 2001, available from http://menzies.us/pdf/00ml.pdf.

[40] Lionel C.Briand, "Novel applications of machine learning in software testing", Proccedings of the 8th International Conference on Quality Software, IEEE, pp.3-10, 2008.

[41] G. Boetticher, T. Menzies, and T. Ostrand, "PROMISE repository of empirical software engineering data," 2007, available at http://promisedata.org/ repository.

[42] D. Rodriguez, I. Herraiz, and R. Harrison, "On software engineering repositories and their open problems," in The International workshop on realizing AI synergies in software engineering (RAISE'12), 2012.

[43] B.W.Boehm,Software engineering economics, Prentice-Hall, 1994.

[44] Outi Raiha,"A survey of search-based software design", Computer science review, Elsevier, 4(4),pp.203-249,2010.

[45] M. Harman and L. Tratt, "Pareto Optimal Search-Based Refactoring at the Design Level," Proc. 9th Ann. Conf. Genetic and Evolutionary Computation (GECCO 07), ACM Press, pp. 1106-1113,2007.

[46] Shaukat Ali, L.C.Briand, Hadi Hemmati, and Rajwinder K.Panesar-Walawege, " A systematic review of the application and empirical investigation of search-based test case generation", IEEE Transactions on software engineering, 36(6), pp.742-762, 2010.

[47] Alessandro Marchetto, Md. Mahfuzul Islam, Waseem Asghar, Angelo Susi, and Giuseppe Scanniello, "A multi-objective to prioritise test cases", IEEE Transactions on software engineering, 42(10), pp.918-940, 2016.

[48] A. Baresel, H. Sthamer, and J. Wegener, "Applying Evolutionary Testing to Search for Critical Defects," *Proc. Conf. Genetic and Evolutionary Computation* (GECCO 04), LNCS 3103, Springer, pp. 1427-1428,2004.

[49] Y.Zhang, "Repository of SBSE papers", http://crestweb.cs.ucl.ac.uk/resources/sbse_repository/.

doi:10.32377/cvrjst1114