International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 5 Issue: 5 206 – 211

A Review on Software Architecture Optimization Methods

Akshay Jadhav
PG Scholar, DCEA
NITTTR
Bhopal, India
akshay15jadhav93@gmail.com

Dr. Sanjay Agrawal
Professor, DCEA
NITTTR
Bhopal, India
sagrawal@nitttrbpl.ac.in

Abstract— Due to the remarkable mechanical request for programming frameworks, the expansion of the uncertainty, the quality requirements and quality of testing, the programming engineering configuration has been transformed into essential progression movement and the examination site is developing rapidly. In the recent decades, programming engineering involves improved technologies, which means to organize a scan for design outline for an arrangement of value attributes, have multiplied. In any case, the results shown are divided into different research groups, many framework areas and different quality features. Coming about the inclusion of current research, we have played a well-structured writing survey and have broken the result of various check-sheets of different research groups. Considering this study, a scientific classification has been done which is used for current research. Apart from this, the effective investigation of the examination writing given in this audit is expected to help in exploration and merging the current research endeavors and inferring an examination plan for future advancements.

Keywords*- Software Architecture Optimization, Systematic Literature Review, Optimization Methods, Problem Overview*

*****

I. INTRODUCTION

Engineering details and models [6] are used to structure complex programming systems and to give an arrangement that is the foundation for later programming engineering works out. Because of outline judgments, programming experts are better strengthened in adapting to the extending diverse nature of today's item structures. Therefore, the building setup stage is seen as a proponent among the most basic activities in an item planning [7]. The decisions made in the midst of designing arrangement have imperative consequences for money related and quality targets. Instances of configuration level decisions consolidate the assurance of programming and hardware components, their replication, the mapping of programming components to open gear centre points, and the general structure topology. Due to the in-wrinkling framework many sided quality, programming models need to look over a combinatorial developing number of design choices when hunting down an ideal architecture plan concerning a characterized set of value attribute(s) and requirements. The outcomes of these plans are explored in the area of a plan that are regularly behind human capabilities and provide a test function to the structural framework.

The requirement for automatic configuration space investigation that enhances a current engineering detail has been perceived and a plenty of design improvement approaches in light of formal design determinations have been created. To deal with the complexity of the assignment, the enhancement approaches limit the changeability of engineering choices, upgrading the design by altering one of its particular viewpoints such as area, replication, choice of building components and so forth. Henceforth the examination exercises are scattered over many research groups, framework spaces, (for example, installed frameworks or data frameworks), and quality traits. Comparable methodologies are proposed in numerous spaces without monitoring each other.

A. Research Approach and Contribution

The deliberate writing survey of the current design advancement approaches provides an interface to the learning and gives a far reaching diagram of the present best in class. Thus, an entryway to new methodologies of engineering advancement can be opened, joining diverse sorts of compositional choices during the streamlining or utilizing offbeat optimization procedures. New exchange of examination techniques can be produced because of various enhancement areas. This can convey significant advantages to the general routine with regards to design advancement. As a rule, with the study we expect to accomplish the accompanying targets:

- To give a fundamental order structure in type of a scientific categorization to characterize existing engineering optimization approaches.
- To give an outline of the present best in class in the design enhancement area.
- To bring up flow patterns, holes, and headings for future research.

We inspected various papers from different research sub-regions, distributed in programming building diaries and gatherings. At first, we inferred a scientific classification by pre-framing a formal substance investigation. All the more particularly, in view of the underlying arrangement of catchphrases and characterized inclusion and avoidance criteria, we gathered an arrangement of papers, which we iteratively examined to recognize the scientific categorization ideas.

B. Related studies

Design streamlining can be memorized into the general research which teaches of Search Based Software Engineering (SBSE) [11] as it applies proficient inquiry techniques to distinguish an ideal or close ideal design detail. SBSE is connected in all periods of the product building process, including necessities, building, extend administration, plan, upkeep, figuring out, and programming testing. A far reaching review of various streamlining techniques connected to programming, building assignments is given by Harman et al. [10]. The overview shows that in the previous years, a specific increment in SBSE movement has been seen, with numerous new applications being tended to. The paper distinguishes inquire about patterns and connections between the pursuit strategies and the applications to which they have been connected. The concentration of Harman et al's. overview is on the wide field of SBSE, particularly on methodologies in the product testing stage which are likewise canvassed in definite reviews. Nonetheless, the region of engineering streamlining has not been investigated in detail. The SBSE study records a few ways to deal with streamlining the programming configuration, yet does not examine the properties of these methodologies aside from naming the utilized advancement methodology.

Adjacent to this general SBSE study, different reviews de-copyist sub-territories of engineering advancement and configuration space investigation that are just worried with a particular framework areas, or a particular streamlining technique. For example, the review of Grunske et al. [14] is worried with the space of wellbeing basic installed frameworks and thinks about 15 engineering enhancement strategies. Another illustration is the review of Villegas et al. [15], which assesses 16 approaches that objective run-time engineering enhancements with an attention on self-versatile frameworks. In the exploration sub-region of frameworks with high dependability requests, Kuo and Wan [16] have distributed an overview in 2007 looking at changed repetition designation approaches. At last, a few overviews are worried by the use of a particular advancement system, commonly identified with Genetic Algorithms [18] or Meta heuristics.

II. RESEARCH METHODS

Our literature review follows the guidelines proposed by Kitchenham [20], which structure the stages involved in a systematic literature review into three phases: planning, conducting, and reporting the review. Based on the guidelines, this section details the research questions, the performed research steps, and the protocol of the literature review. First, Section A describes the research questions underlying our survey. Then, Section B derives the research tasks conducted, and thus describe the procedure. Section C then details the literature search step and highlight the inclusion and exclusion criteria. Finally, Section D discusses threats to the validity of our study.

A. *Research Questions*

Based on the objectives described in the introduction, the following research questions have been received, which form the basis for the literature review:

RQ1. How can the current research on software architecture optimization be classified?

RQ2. What is the current state of software architecture optimization research with respect to this classification?

RQ3. What can be learned from the current research results that will lead to topics for further investigation?

B. *Research Tasks*

To answer the three research questions RQ1-3, four research tasks have been conducted: one task to set up the literature review, and three research tasks dedicated to the identified research questions. The tasks have been conducted sequentially and interconnected through a number of artefacts generated by their sub-tasks. The overall research method is outlined in Figure 1 and detailed in the following text.

The set-up task includes the definition of the review protocol, the selection of search engines, and the definition of a keyword list, a keyword-based collection of published architecture optimization papers, and a review filtering the papers according to a categorized set of inclusion and avoidance criteria. The search step and the inclusion/exclusion review step are explained in more detail in Section C of research methods.

Based on the set of selected papers, content analysis of the papers has been performed in the first research task (RQ1). The goal was to derive taxonomy to classify the current architecture optimization approaches. An iterative coding process was used to identify the main categories of the taxonomy. The coding process was based on the grounded theory [23] qualitative research method. First, we analyzed each paper with a goal to identify new concepts for the taxonomy. Second, after all papers have been reviewed and the taxonomy got updated with newly identified concepts, we consolidated the taxonomy terms, mainly by merging the synonyms and unifying the concepts on different levels of abstraction. Section 3 presents the findings.

In the second research task (RQ2), each paper collected in the set-up task was classified and cross-checked to remove inconsistencies within the classification. Extracted data were stored in a database, which enabled a descriptive quantitative analysis. The aim of the data extraction and the resulting classification was to provide a significant overview of the current research efforts and the archived results in this domain. Third section presents the findings.

In the third research task (RQ3), we cross-analyzed the survey results and integrated conceivable headings for further research. Consequently, the survey enables the transfer of knowledge from one research sub-area to another and thus aims at improving the overall research area. Fourth section provides conclusions and recommendations for future research.

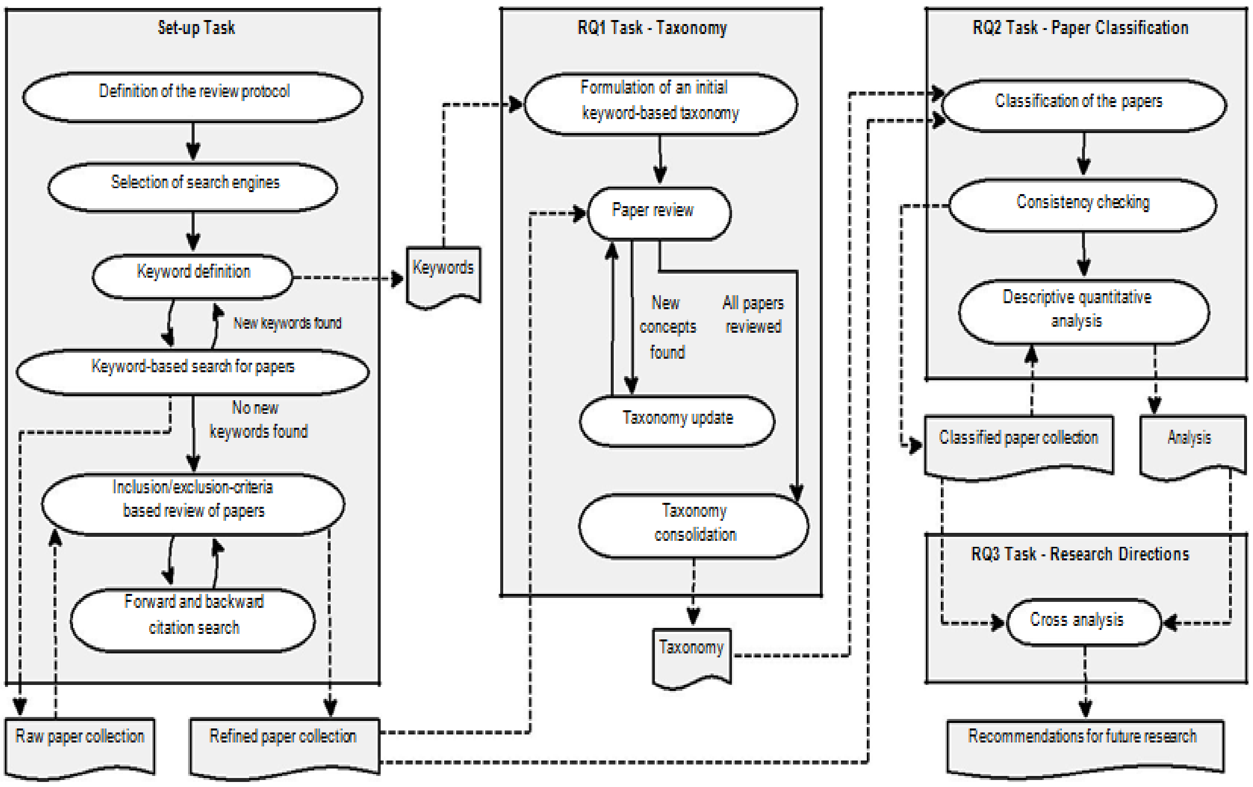
Set-up Task
Definition of the review protocol
Selection of search engines
Keyword definition
Keywords
New keywords found
Keyword-based search for papers
No new keywords found
Inclusion/exclusion-criteria based review of papers
Forward and backward citation search
Raw paper collection
Refined paper collection
RQ1 Task - Taxonomy
Formulation of an initial keyword-based taxonomy
Paper review
New concepts found
All papers reviewed
Taxonomy update
Taxonomy consolidation
Taxonomy
RQ2 Task - Paper Classification
Classification of the papers
Consistency checking
Descriptive quantitative analysis
Classified paper collection
Analysis
RQ3 Task - Research Directions
Cross analysis
Recommendations for future research

Fig. 1 Process model for Literature Review

C. Literature Search Process

The search strategy for the review was towards finding, published papers in journals and conference proceedings via the widely accepted literature search engines and databases such as Google Scholar, IEEE Explore, ACM Digital Library, Springer Digital Library, and Elsevier ScienceDirect.

In the search we focused on selected keywords, based on the aimed scope of the literature review. Examples of the keywords are: automated selection of software components, deployment optimization, energy consumption optimization, component selection optimization, automated component selection, reliability optimization, software safety optimization, redundancy allocation, optimal scheduling, hardware-software co-synthesis, and search based software engineering, run-time and design-time architecture optimization, software engineering optimization, self-adaptive software systems. The keywords were refined and extended during the search process.

Although the selection process was primarily based on the review of paper abstracts and keywords, in the cases where these two were insufficient, we also considered part of the introduction, contribution and conclusion sections.

1) Inclusion Criteria : The concentration of this literature review is on programming architecture streamlining. We comprehend the architecture of a software system to be "the key association of a framework encapsulated in its parts, their connections to each other, and to the environment, and the principles managing its design and evolution" [6]. Software architecture optimization is understood as an automated method aiming to reach an optimal architecture design with respect to a set of quality attributes. The main criteria for inclusion were based on the automation of software architecture optimization, both at run time and at design time. To enable automated optimization of software architectures, three basic prerequisites need to be fulfilled:

- A machine process-able representation of the software architecture must be available as an input for automated searches.
- A function or procedure which automatically evaluates aspect of quality for a given software architecture is required, called quality evaluation function/procedure in this work. Cost is also considered as optimization objective. Papers which solve constrained problems are included.
- A meaning of the considered outline space is required that portrays how a given programming architecture portrayal can be changed or improved by the streamlining.

2) Exclusion Criteria: We evicted papers that:

- Upgrade a solitary component without integrating context and interactions with other architectural elements.
- Concentrate on an engineering immaterial issue (e.g. Necessities prioritization, compiler enhancement, or assignment distribution to specialists that collaborate in executing and completing the undertakings)
- Optimize hardware with no relation to software.
- Solely optimize cost without considering whatever other quality trait.

We di**d** not bar papers for quality reasons, because the quality of the papers was generally acceptable.

D. Threats to Validuty

One of the major threats to the validity of this literature review is the incompleteness. The risk of this threat highly depends on the selected list of catchphrases and the limitations of the employed search engines. To decrease the risk of an inadequate keyword list, we have used an iterative approach to keyword list construction. A well known set of papers was used to build the initial taxonomy which evolved over time. New keywords were added when the keyword list was not able to find the state-of-the-art in the respective area of study. We used multiple search engines in order to omit the limitations implied by employing a particular search engine.

Another vital issue is whether a scientific classification is robust enough for the analysis and classification of the papers. To avoid taxonomy with insufficient capability, we need to classify the selected papers; therefore we used an iterative content analysis method to continuously evolve the taxonomy for every new concept encountered in the papers. New concepts were introduced into the taxonomy and taxonomy is updated.

Furthermore, in order to make the scientific classification a better foundation for analyzing the selected papers, we allowed multiple abstraction levels for selected taxonomy concepts. As a result, one of the concepts (namely the used optimization strategy) has different levels of detail, where the highest level is abstract with fewer classes, whereas lower levels have more details with more classes used to classify the papers. The appropriate level was chosen while displaying the outcomes. In order to reduce the classification tilt, paper classification results are checked.

III. TAXONOMY

Literature review depends on the quality of the project. The selected classification scheme, which affects depth of knowledge about each study approach was recorded In this paper, an iterative coding process has been employed to identify classification categories (see section 2 For details) and to provide an answer for the research questions.

A. *The Problem Category*

The primary class is identified with the issue the approaches plan to understand in this present reality. Generally speaking, the methodologies attempt to accomplish a specific enhancement objective in a particular setting. For instance, an improvement objective is to limit the reaction time of an engineering given costs limitations. A case setting is to consider implanted frameworks at configuration time. While the con-content of the issue is controlled by the sub-classifications area (i.e. The kind of focused frameworks) and stage (i.e. put in the advancement procedure) of the issue, the sub-classifications identified with the enhancement objective incorporate quality properties, limitations, and the dimensionality of the improvement issue, which is administered by the question if the arrangement of upgraded quality characteristics is accumulated into a solitary scientific capacity or decoupled into clashing targets (single/multi-target streamlining).

Specifically, the area has three conceivable qualities: Information Systems (IS) are business related frameworks worked on a broadly useful PC that incorporate for example e-business applications, enterprise and government data frameworks. Embedded Systems (ES) interestingly are acknowledged on a committed equipment to play out a particular capacity in a specialized framework. They scale from little compact gadgets like cell phones to extensive production lines and power plants. On the off chance that an approach is intended for both spaces, the third conceivable quality "general" is utilized. The stage class determines whether the issue is happening at configuration time (DT) or run-time (RT). The principle distinction between the two is that while the setting of a design time issue is known earlier, the setting of a run-time issue changes progressively (e.g. new task can arrive during run-time planning). Once more, the esteem "General" can be utilized here to mean methodologies that address both DT and RT.

The objective of the streamlining undertaking is normally the expansion of the product design quality under given requirements. Since the nature of a product framework as an idea is hard to characterize, because of its subjective nature, programming specialists don't characterize the quality directly however relate it to various framework characteristics, called quality properties [19]. In this work, we just consider quantifiable quality characteristics (cf. section C Segment 2). Cases are execution, dependability, cost, accessibility, and other settled quality characteristics (locate the full rundown in at [24]). While classifying quality traits, we took after generally acknowledged definitions and quality characteristic scientific classifications [25], [7], [26], [27]. In our scientific categorization, we recognize quality attributes to be advanced (classification quality traits) from extra limitations on quality characteristics or other framework properties (class imperatives). For instance, diminishing the reaction time and the expenses of a framework however much as could be expected is a setting with two quality credits to be streamlined. Expanding the accessibility while keeping the reaction time lower than 5 seconds and adhering to auxiliary requirements is a setting with one quality at-tribute to be optimized (accessibility) and two imperatives (for execution and basic).

Ultimately, the dimensionality class reflects if the approach addresses a solitary target streamlining (SOO) or multi-target streamlining (MOO) issue. The SOO upgrades a solitary quality trait only. The MOO streamlines different quality characteristics instantly, so that the nature of each engineering model is a vector of qualities. As quality traits regularly struggle, for the most part there is no single ideal outcome however a set of outcomes non-commanded by the others from the perspective of the improved qualities – i.e. arrangements that are Pareto-ideal [28]. Since in MOO a choice must be gone up against the last engineering outline chosen from the arrangement of coming about competitors, one can likewise utilize the multi-target changed to single-target streamlining (MTS) approaches, which encode the determination criteria taking after MOO into a solitary mathematical function, which is then upgraded as a solitary goal.

For an organized view on everyone of the estimations of the talked about sub-classifications variety [30]. The conceivable qualities are those found in the evaluated papers, gathered by equivalent words, since no current order, (for example, for quality traits) is accessible to utilize, thus, we clarify them in more detail in the following passages. The choice degrees of flexibility are worried with choosing substances in the design. These substances can be programming elements, (for example, modules) or equipment elements, (for example, servers or gadgets), bringing about "delicate product choice" and "equipment choice" qualities. We unambiguously recognize "component selection", since a few areas have a specific idea of a segment. For instance, in embedded system plan, segment determination could mean choosing a part understanding usefulness in equipment and a segment with broadly useful equipment acknowledging usefulness in programming. Besides, we unequivocally distinguish "service selection", on the grounds that by choosing the product to execute, choosing an administration likewise incorporates choosing the service provider (in this way including equipment angles too).

Replication degrees of opportunity are worried with changing the variety of a structural component. Under the expression "equipment replication", we subsume all degrees of opportunity that worry the quantity of an equipment element's duplicates, while perhaps at the same time changing the multiplicity of programming components (e.g. programming parts conveyed to the repeated servers). The prevalent term repetition designation is along these lines incorporated into "equipment replication". Under the expression "programming replication", we subsume degrees of opportunity that change the quantity of duplicates of programming elements as it were. For quickness, we incorporate both indistinguishable duplicates of the product and distinctive implementations of a similar usefulness (e.g. n-version programming) in the expression "programming replication" in this paper.

Parameter degrees of opportunity allude to different parameters of design components. We recognize "programming parameters" (e.g. number of strings of an application server) and "equipment parameters" (e.g. parameters for the hard disk drive). Equipment parameters may cover with equipment choice, in light of the fact that the decision (e.g. of a CPU with various speed) can be demonstrated both as equipment choice and as a parameter of the facilitating server.

B. *The Solution Category*

The solution category classifies the approaches according to how they achieve the optimization goal and thus describes the main step of the optimization process, which is depicted in Fig. 2. First, the subcategory architecture representation is the process input that describes the architecture to optimize. Second, the subcategory degree of freedom describes what changes of the architecture are considered as variables in the optimization. Third, the subcategory quality evaluation describes the used quality evaluation procedures which make up the objective function(s) of the optimization process. In addition, this category contains the techniques used to solve the formulated optimization problem: Subcategories are the overall optimization strategy and constraint handling.

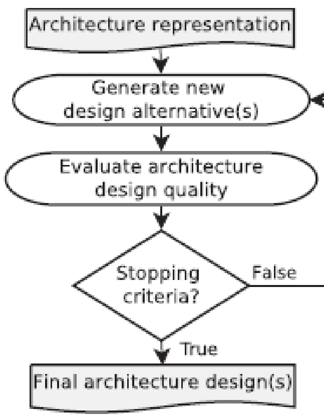
Architecture representation
Generate new design alternative(s)
Evaluate architecture design quality
Stopping criteria?
False
True
Final architecture design(s)

Fig. 2 Optimization process

C. *The Validation Category*

For the validation classification, two subcategories are considered, they are:

The approach validation describes techniques used to evaluate the practicality and correctness of the approach. This includes specifically the effort used up on the modeling of quality prediction functions and evaluating their accuracy. Possible validation types found in the reviewed approaches include revelation with a simple example, validation with dedicated standard problems or experiments with randomly generated problems, and validation with an academic or industrial case study. Besides these, the possible validation type also includes mathematical proofs with accurateness of the results and evaluation with related literature.

In contrast to the approach validation, the optimization validation purposely validates the used optimization strategy. Such a validation may evaluate 1) how well an approach approximates the global optimum and/or 2) the performance of an approach compared to other approaches. A possible type of an optimization validation for an approach that uses a heuristic is a comparison with a random search strategy, an exact algorithm, or a baseline heuristic algorithm.

IV. Conclusion

In this article, we have presented the results of a systematic literature review on architecture optimization which includes different approaches. Based on this review, we derived a scientific classification that aims to help researchers to classify existing and future approaches in this research area. Using this scientific classification, we have analyzed the present approaches and how it will help researchers, to relate their work with the current scenario and to identify the future approaches in a direction.

During the systematic literature review process, we learned about the different research areas and how they provide recommendation for future research in that sub-area. We acquired knowledge that, although there is some research communities that are interrelated through cross-citation of their research work. Such as community of reliability and performance architecture optimization, because of similarities in their research models. But still there are some communities remaining, which are isolated from others, e.g. scheduling community, optimization community, i.e. whose priority is to only focus on optimization strategies. The information presented in this survey aims to bridge the gap among the communities and allow for easier knowledge transfer.

In summary, we believe that the results of our systematic review will help to advance the architecture-optimization research area, and since we expect this research area to grow in the future. We hope that scientific classification in this paper will be useful in enhancing and judging new methods.

Acknowledgment

I would like to thank Dr. Sanjay Agarwal, who is involved in the review of this research paper, without his passionate participation and input; review could not have been productively conducted.

References

[1]. T. Back, ¨ Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms. New York: Oxford University Press, 1996.

[2]. D. Ardagna and R. Mirandola, “Per-flow optimal service selec-tion for Web services based processes,” The Journal of Systems and Software, vol. 83, no. 8, pp. 1512–1523, Aug. 2010.

[3]. D. Ardagna, G. Giunta, N. Ingraffia, R. Mirandola, and B. Pernici, “QoS-driven web services selection in autonomic grid environ-ments,” in On the Move to Meaningful Internet Systems 2006, ser.

Lecture Notes in Computer Science, R. Meersman and Z. Tari, Eds., vol. 4276. Springer, 2006, pp. 1273–1289.

[4]. A. Arcuri and L. C. Briand, "A practical guide for using statistical tests to assess randomized algorithms in software engineering," in Proceedings of the 33rd International Conference on Software Engineering, ICSE 2011, R. N. Taylor, H. Gall, and N. Medvidovic, Eds. ACM, 2011, pp. 1–10.

[5]. B. R. Arafeh, K. Day, and A. Touzene, "A multilevel parti-tioning approach for efficient tasks allocation in heterogeneous distributed systems," Journal of Systems Architecture - Embedded Systems Design, vol. 54, no. 5, pp. 530–548, 2008.

[6]. ISO/IEC Standard for Systems and Software Engineering—Recommended Practice for Architectural Description of Software-Intensive Systems, Int'l Standards Organization, ISO/IEC 42010 IEEE Std 1471-2000, first ed. 2007-07-15, p. c1-24, 2007.

[7]. L. Bass, P. Clements, and R. Kazman, Software Architecture in Practice, second ed. AddisonWesley, 2003.

[8]. Y. P. Aneja, R. Chandrasekaran, and K. P. K. Nair, "Minimal-cost system reliability with discrete-choice sets for components,"

[9]. J. D. Andrews and L. M. Bartlett, "A branching search approach to safety system design optimisation," Reliability Engineering & System Safety, vol. 87, no. 1, pp. 23–30, 2005.

[10]. M. Harman, "The Current State and Future of Search Based Software Engineering," Proc. Int'l Conf. Software Eng., L.C. Briand and A.L. Wolf, eds., pp. 342-357, 2007.

[11]. M. Harman, S.A. Mansouri, and Y. Zhang, "Search Based Software Engineering: A Comprehensive Analysis and Review of Trends Techniques and Applications," Technical Report TR- 09-03, Dept. of Computer Science, King's College London, Apr. 2009.

[12]. A. Aleti, S. Bjornander,¨ L. Grunske, and I. Meedeniya, "Archeopterix: An extendable tool for architecture optimization of AADL models," in ICSE 2009 Workshop on Model-Based Method-ologies for Pervasive and Embedded Software, MOMPES 2009. IEEE Computer Society, 2009, pp. 61–71.

[13]. J. T. Alander, "An indexed bibliography of genetic algorithms in testing," Univ. of Vaasa, Finland, Tech. Rep. 94-1-TEST, 2008.

[14]. L. Grunske, P.A. Lindsay, E. Bondarev, Y. Papadopoulos, and D. Parker, "An Outline of an Architecture-Based Method for Optimizing Dependability Attributes of Software-Intensive Systems," Architecting Dependable Systems IV, R. de Lemos, C. Gacek, and A.B. Romanovsky, eds., pp. 188-209, Springer, 2006.

[15]. N.M. Villegas, H.A. Mu¨ ller, G. Tamura, L. Duchien, and R. Casallas, "A Framework for Evaluating Quality-Driven Self-Adaptive Software Systems," Proc. Int'l Symp. Software Eng. Adaptive and Self-Managing Systems, pp. 80-89, 2011.

[16]. W. Kuo and R. Wan, "Recent Advances in Optimal Reliability Allocation," Computational Intelligence in Reliability Eng.g, Evolutionary Techniques in Reliability Analysis and Optimization, G. Levitin, ed., pp. 1-36, Springer, 2007.

[17]. M. Agarwal, S. Aggarwal, and V. K. Sharma, "Optimal re-dundancy allocation in complex systems," Journal of Quality in Maintenance Engineering, vol. 16, pp. 413–424, 2010.

[18]. H. Jiang, C. Chang, D. Zhu, and S. Cheng, "A Foundational Study on the Applicability of Genetic Algorithm to Software Engineering Problems," Proc. IEEE Congress Evolutionary Computation, pp. 2210-2219, 2007.

[19]. Neha Sharma, Amit Sinhal, Bhupendra Verma, "Software assessment parameter optimization using GA", International Journals of Computer Applications (0975-8887), vol. 72- no. 7, May 2013

[20]. Barbara Kitchenham, O. Pearl Brereto , David Budgen , Mark Turner, John Bailey, Stephen Linkman, "Systematic literature reviews in software engineering – A systematic literature review", Information and Software Technology 51 (2009) 7–15

[21]. Brajesh kumar Singh, A.K. Mishra, "Software Effort Estimation by GA tuned Parameters of Modified Constructive Cost model for NASA Software Projects", International Journals of Computer Applications (0975-8887), vol. 59- no. 9, Dec 2012

[22]. A. Azaron, C. Perkgoz, H. Katagiri, K. Kato, and M. Sakawa, "Multi-objective reliability optimization for dissimilar-unit cold-standby systems using a genetic algorithm," Computers & OR, vol. 36, no. 5, pp. 1562–1571, 2009.

[23]. B. Glaser and A. Strauss, Grounded Theory: The Discovery of Grounded Theory. de Gruyter, 1967.

[24]. A. Aleti, B. Bu¨hnova´, L. Grunske, A. Koziolek, and I. Meedeniya, "Optimization Survey," https://sdqweb.ipd.kit.edu/wiki/ OptimizationSurvey, 2012.

[25]. A. Avizienis, J.-C. Laprie, B. Randell, and C.E. Landwehr, "Basic Concepts and Taxonomy of Dependable and Secure Computing," IEEE Trans. Dependable and Secure Computing, vol. 1, no. 1, pp. 11- 33, Jan.-Mar. 2004.

[26]. B.R. Arafeh, K. Day, and A. Touzene, "A Multilevel Partitioning Approach for Efficient Tasks Allocation in Heterogeneous Distributed Systems," J. Systems Architecture—Embedded Systems Design, vol. 54, no. 5, pp. 530-548, 2008.

[27]. J. Balasubramanian, A.S. Gokhale, A. Dubey, F. Wolf, C. Lu, C.D. Gill, and D.C. Schmidt, "Middleware for Resource-Aware Deployment and Configuration of Fault-Tolerant Real-Time Systems," Proc. IEEE Real-Time & Embedded Technology and Applications Symp., pp. 69-78, 2010.

[28]. P.G. Busacca, M. Marseguerra, and E. Zio, "Multiobjective Optimization by Genetic Algorithms: Application to Safety Systems," Reliability Eng. & System Safety, vol. 72, no. 1, pp. 59-74, Apr. 2001.

[29]. Anne Koziolek, Lars Grunske, Barbora Buhnova, Aldeida Aleti, Indika Meedeniya, "Software Architecture Optimization Methods: A Systematic Literature Review", *IEEE Transactions on Software Engineering*, vol. 39, no. , pp. 658-683, May 2013, doi:10.1109/TSE.2012.64.

[30]. S. Burmester, H. Giese, E. Mu¨ nch, O. Oberschelp, F. Klein, and P. Scheideler, "Tool Support for the Design of Self- Optimizing Mechatronic Multi-Agent Systems," Int'l J. Software Tools for Technology Transfer, vol. 10, no. 3, pp. 207-222, June 2008.