International Journal of Engineering &Technology, 7 (1.7) (2018) 79-83

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Combinatorics based problem specific software architecture formulation using multi-objective genetic algorithm

V. Nivethitha [1] *, P. M Abhinaya [1]

[1] *Assistant Professor, Department of Computer Science and Engineering, School of Computing, Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Avadi, Chennai-62, TamilNadu, India*
**Corresponding author E-mail: nive.nivethitha@gmail.com*

Abstract

In Software Development Process, the design of complex systems is an important phase where software architects have to deal with abstract artefacts, procedures and ideas to discover the most suitable underlying architecture. Due to uncontrolled modifications of the design and frequent change of requirements, many of the working systems do not have a proper architecture. Most of the approaches recover the architectural blocks at the end of the development process which are not appropriate to the system considered. In order to structure these systems software components compositions and interactions should be properly adjusted which is a tedious work. Search-based Software Engineering (SBSE) is an emerging area which can support the decision making process of formulating the software architecture from initial analysis models. Thus component-based architectures is articulated as a multiple optimisation problem using evolutionary algorithms. Totally different metrics is applied looking on the design needs and also the specific domain. Thus during this analysis work, an effort has been created to propose a multi objective evolutionary approach for the invention of the underlying software system architectures beside a versatile encoding structure, correct style metrics for the fitness operate to enhance the standard and accuracy of the software system design.

Keywords: *Search Based Software Engineering; Multi Objective Evolutionary Algorithms.*

1. Introduction

Software Architecture of a system defines the relationship between major structural elements of the software, the styles and design patterns that can be used to accomplish the requirements defined for the system, and the constraints that affect the way in which architecture can be implemented. The architecture design provides a blueprint and guidelines for developing a software system based on its requirements analysis specification. The architectural design must cover the functional and non-functional requirements [1].

Throughout the development cycle, engineers have to be compelled to decide concerning the suitable structures and styles for any code. The automated abstract thought and analysis of style alternatives could be a difficult method once solely restricted information concerning the system is accessible. In this context, architectural analysis is an important phase in the system development because it provides careful information concerning the look specifications of the code in earlier stages [2].

Frequently software architectures tackle the discipline analysis from a working system to increase the practicality. this might be a troublesome method as a result of for many of the code systems there's no correct documentation attributable to the continual demand changes and therefore the architectures that are recovered at the end of the development cycle by reverse engineering method from the source code leads to inappropriate immateriality and don't represent the initial conception of the system. Thus the discovery method may be done with the data that are offered within the earlier stages of development like elaborated class diagrams. These models offer a more robust intermediate read of the code between the immateriality of the specification and therefore the specificity of the code.

Recently, Evolutionary Computation finds its application in various fields and also in software engineering. Since the appearance of SBSE Evolutionary Computation has grown to huge extents and considerable efforts has been made to propose new techniques, methods and operators to solve complex applications [3].

This work studies and deliberates the strategies to face the architectural analysis as a multi-objective improvement problem. More accurately, the matter of discovering component based software package architectures is self-addressed, specializing in the choice and modification of many quality metrics by using the Non-Dominated Sorting Genetic Algorithm-II.

In this framework, the identification of software system architectures is taken into account throughout the initial stages of software system formation, once software system architects still wish to {change} their current software system structure as necessities change or check the accuracy of the ensuing style. Software system architects, need alternative sources of knowledge so as to get the supposed design once source code articles don't seem to be accessible. Initial class diagrams, usually the most used representations in the analysis phase, constitute an interesting factor for architecture discovery. These diagrams deal more precise analysis information than source code [4].

2. Proposed framework

The problem is framed as the search of the most effective obtainable analysis components, i.e. classes and their relationships, to adapt those extremely abstract artefacts of the software system

Copyright © 2018 V. Nivethitha, P. M Abhinaya. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

design, i.e. components, interfaces and connectors. More exactly, a gaggle of classes is outlined as a component and also the component behavior is obtained by the union of their individual behavior. The relationships known between classes belonging to completely different components is thought as candidate interface. A provided or needed interface is sculptured by analyzing the direction of every relationship. Finally, every connecter is outlined by a combination of provided and needed interfaces.

3. Architecture diagram

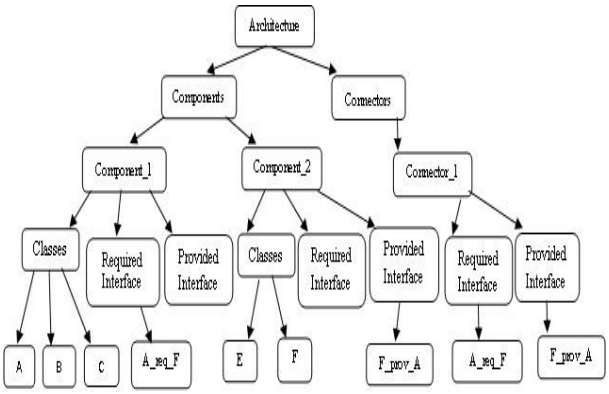


Fig. 1: Genotype.

Figure 1 shows that entire architecture design of Genotype

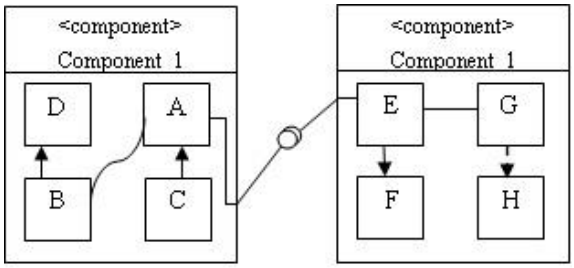


Fig. 2: Phenotype.

Figure 2 shows thecomponents of phenotype

4. Encoding solutions

The software design problems require easy mapping process that can improve the efficiency. Tree structures have been used, as they have been successful in both computational and human domains [5]. Components, connectors, interfaces and other elements represent a graded composition. Classes and its associated relationships comprises a component, whose entire description requires the definition of its provided and required interfaces. Connectors are identified from relationships established between classes belonging to different components. Then, mapping a component diagram into a tree structure is shown in Fig. 1, Thenodes represent the elements that can be different from one solution to another, i.e. a number of component and connectors as well as the distribution of classes and interfaces among them.

The initialization method begins with a random distribution of classes among variety of elements that are haphazardly elected between a least and a most worth. Then, the design interfaces and connectors are mined from the design by observing the new relationships established between classes that belongs to totally different components [6].

Each and every solution is forced by the subsequent assumptions (a)components don't seem to be allowed to be empty; (b) every class is assigned to only 1 component; (c) elements, that don't specify any interface, don't seem to be permitted;

And (d) the reciprocally dependent components, i.e. 2 components wherever one component needs from and provides to the opposite, ought to be penalised. Once the population is initialized and when genetic operator is applied.

Constraints (a) And (b) is enforced. Constraints(c) and (d) are enforced in conjunction with the creation of the population that aims to support a bigger diversity of solutions and to cut back the formatting time. These constraints facilitate in removing the invalid individual from the population by appropriate use of operators and also a selective pressure on the evolutionary process.

Genetic operator

A mutation operator is used to discover design options. It can provide a variety of solution by applying diverse architectural transformations. A weighted roulette is used to select a precise transformation type and also decides the different mutation procedures that can be applied to the selected parents. There are five different mutation procedures and they are listed as follows: components can be added and removed, two components can be merged, split a component and transfer a class.

5. Evaluation objectives

Different metrics associated to architectural ideas are elite and custom-made for the planned problem, providing a large vary of arrangements. The creator selects the foremost suited metrics for the precise state of affairs.

1) Intra-modular Coupling Density (ICD) [6] forms a trade-off between cohesion and coupling measures, as given in Equation 1, and will be aimed to supply high values for every element i, the quantitative relation between the total number of interior and exterior relations is calculated.CI_i^{in}represents the count of connections between classes within the component i and $CI_i^{out}$is the amount of external relations that refers the interface. The value ofCI_i^{out} I ranges between [0,1], the value of ICD will vary to a most value of n, i.e the overall range of components.

$$ICD_i = \left(\frac{classes_{total} - \#classes_i}{classes_{total}}\right).\left(\frac{CI_i^{in}}{CI_i^{in}+CI_i^{out}}\right) \quad (1)$$

2) External Relations Penalty (ERP) [6]. It computes the total number of relationships present among the classes between the components, i and j. These associations are acquired from the preliminary analysis model in terms of generalizations (ge), associations (as), aggregations (ag) and compositions (co) between classes. The relationships that are defined as interfaces are not considered relevant to the metric. Further, the strengths of the relationships among the classes are evaluated based on a weighted sum (wx). The best value for this metric is 0, which indicates that all the associations among components occur through well-developed interfaces.

$$ERP=\sum_{i=1}^{n}\sum_{j=i+1}^{n}(W_{as}.n_{as_{ij}}+W_{ag}.n_{ag_{ij}}+W_{co}.n_{co_{ij}}+w_{ge}.n_{ge_{ij}}) \quad (2)$$

3) (3)The Groups/Components quantitative relation (GCR) [6] measure is projected for the reduction of the total number of associated teams of classes (cgroups) within every component. Software system architects typically hunted for |a collection of components wherever every component is contained by one group of powerfully connected classes, that determines the practicality level per component. Therefore, the optimum value of this metric is one, which means that there's an equivalent variety of groups than components.

$$GCR = \frac{\#cgroups}{\#Components} \quad (3)$$

The Encapsulation (Enc) proposed in [7] is given as equation. The encapsulation of every component $iEnc_i$ is given by the quantitative relation of the hidden classes, i.e. those who don't participate in exterior interactions with classes that belong outside of the component. The entire encapsulation, ranges between [0,1] and

will be maximized. It is the measure of the individual encapsulation of every component.

$$\mathrm{Enc_i} = \frac{\#\mathrm{inner_{classes}}}{\#\mathrm{totol_{classes}}} \mathrm{Enc} = \frac{1}{n}\sum_{i=1}^{n} \mathrm{Enc_i} \quad (4)$$

The Instability (Ins) [8] measures the extent of instability of the component and it aims to cut back the dependencies among the components. A component is instable if the dependency level is over than dependencies. The instability measure of the design is calculated as the average of the instability measure of every component (Insi) (5), and varies in between [0,1]. ACi is referred as afferent coupling of component i, i.e. the total number of components that use its services and ECi is efferent coupling, i.e. the number of external components that give their facilities to component i.

$$\mathrm{Ins_i} = \frac{\mathrm{EC_i}}{\mathrm{EC_i}+\mathrm{AC_i}} \mathrm{Ins} = \frac{1}{n}\sum_{i=1}^{n} \mathrm{Ins_i} \quad (5)$$

Design Size (DS) [8], counts the total sum of components that outstrips the threshold size. The component size is decided by the utmost proportion of well-defined classes contained within every component in regard to the entire range of classes within the analysis model. This measure, varies within the vary [0, 1], n where n represents the total components.

$$\mathrm{CC_i} = \begin{cases} 1 \text{ if} \mathrm{size(i)} > \mathrm{threshold} \\ 0 \text{ otherwise} \end{cases}$$

$$\mathrm{DS} = \sum_{i=1}^{n} \mathrm{CC_i} \quad (6)$$

Abstractness (Ab) This measure is delineated as the quantitative relation of abstract classes within a component abi. The measure of abstractness of a component are often computed by total range of abstract classes to the total range of classes in a very component.

$$\mathrm{Ab_i} = \frac{\mathrm{Na_i}}{\mathrm{Nc_i}} \mathrm{Ab} = \frac{1}{n}\sum_{i}^{n} \mathrm{ab_i} \quad (7)$$

Optimization of Multi-Objective using Non Dominated Sorting Genetic Algorithm-II`

Multi-objective optimization problem (MOP) looks for solutions categorized by a set of decision variables that can discover the most optimal trade off solution amongst a set of objective functions, that usually conflicts with one another other [9].

The Non Dominated Sorting Genetic Algorithm-II, addresses the Multi-Objective problems by dealing the objectives self-sufficiently and returns a group of optimum solutions. Every answer is obtained by a distinct trade-off between all the objectives, during which one objective is improved that may imply the loss of others. This idea is formally articulated as pareto dominance [10].It is outlined as follows: a solution b is meant to dominated by another solution a if a has equal or higher dominating values for all the objectives and a superior value for at least one objective than b. If these conditions can't be satisfied by either a or b, then each solutions are referred as non-dominated, that means that they can't be compared or equivalent. The set of non-dominated solutions defines the Pareto set (PS), and also the mapping of those solutions to the target space is named Pareto front (PF). Getting a most fitting estimated resolution to the PF may be a twofold task. On one hand a multi-objective approach is requested to search out non-dominated solutions that are near the front, On the opposite hand, the ensuing PF ought to be properly distributed among the target space, creating an honest unfold of non-dominant solutions important.

Step 1: Parent population, pt of size Z has to be produced

Step 2: Sort the random parent population supported non-domination.

Step 3: for every non-dominated solution, assign a fitness (rank) up to its non-domination level (1 is that the best level, a pair of is that the next best level, and so on).

Step 4: Qt of size Z offspring population has to be produced using binary tournament selection, recombination, and mutation operators.

Step 5: From the primary generation, formation of recent generation includes the subsequent steps:

a) Produce the pairing pool Rt, of size 2Z by combining the parent population, Pt and also the offspring population, Qt.
b) Rt, The pooled population is sorted, supported by the non-dominated sorting technique to classify all non-dominated fronts (Fr1, Fr2, . . . ,Frl).
c) Generate the new parent population, Pt+ one of size Z by adding non-dominated solutions ranging from the primary hierarchic non-dominated front, Fr1. once the whole non-dominated solutions exceed the population size Z, reject a number of the lower hierarchic non-dominated solutions. This is often achieved through a sorting procedure that is completed in keeping with the thronged comparison operator supported the situation distance.
d) Qt+1 of size Z, offspring population is made by applying the choice crossover and mutation operations on the fresh obtained parent population, Pt+1Step 6: Repeat Step

6. Experimental analysis

In this paper, we propose an effective trait based encryption and mark plot, which is a one-to-numerous encryption technique. At the end of the day, the message is intended to be perused by a gathering of clients that fulfill certain entrance control administers in a BAN. Then, we plan a convention to secure the information interchanges between embedded/wearable sensors and the information sink/information purchasers. Our future research lies in the accompanying ways: outline a more productive encryption approaches with less calculation and capacity necessity (CP ABE with consistent figure content length), which could be better reasonable for pragmatic circumstances (the multiauthority CP ABE conspire) in BAN.

The algorithm and the experimental framework have been implemented in Java. In addition to this, some available Java libraries have been used. Thus, the proposed approach directly collects data from the analysis models. The algorithmic program has been executed many times and executions are dispensed for attainable combos of two, four and six objectives to visualize the measurability of the algorithmic rule.

The performance of the algorithm is evaluated by the two quality indicators, hyper volume (HV) and Spacing (S) [10].The hyper-area engulfed by the front within the search area is termed as (HV), and S measures the unfold of front. Since HV needs a probe area between [0, 1], the standardization approach projected in [11] has been applied.

a) Parameter set-up

Different values are thought of for the population size and therefore the maximum number of evaluations, reflecting a range of configurations for addressing the 2, 4 and 6-objective problems. The table one depicts all the parameters established for the algorithm.

Table 1: Parameter Setup

Population Size	100, 120, 126
Minimum-Maximum Components	2.0-8.0
Mutator weight	wadd= 0.2, wremove= 0.3, wmerge= 0.2 wsplit= 0.1, wmove= 0.2
ERP metrics	was = 1, wag = 3, wco= 1, wge= 5
DS threshold	0.3

Two system styles are thought of for the experimentation. As shown in Table 2,presents completely different complexity in

terms of the number of classes (#Class) and candidate interfaces (#Intf), i.e. straight relationships. The overall range of relationships is split into associations (As), dependences (De), aggregations (Ag), compositions (Co) and generalizations (Ge).

Table 2: Problem Instances

Problem Instances	Classes	Relationships As	De	Ag	Co	Ge	Intf
Datapro4j	58	3	4	3	2	50	12
Marvin	31	5	11	22	2	8	28

b) Analysis of 2-objective problems

All the possible combinations of the objectives are considered. The discovery problem is analyzed as two objective problem and the results are obtained. Table three depicts the most effective rankings once applying the fried man test over every combination of 2 objectives for the two problem instances thought-about.

Table 3: Evaluation Results of Two Objective Problem

Objectives	NSGA-II HV	S
ICD-ERP	2.63	3.52
ICD-GCR	2.72	2.59
ICD-Ins	3.28	2.87
ICD-Enc	2.53	3.96
ICD-DS	3.27	3.28
ICD-Ab	1.3	5.4
ERP-Ins	3.26	3.1
ERP-GCR	2.85	5.73
ERP-Enc	1.63	2.71
ERP-Ab	1.50	3.35
ERP-DS	3.21	3.0
GCR-Ins	3.33	3.00
GCR-DS	3.27	3.12
GCR-Enc	1.62	3.71
GCR- ICD	1.7	3.25
GCR-Ab	2.40	3.22
Ins-DS	3.28	3.4
Ins-Enc	1.9	3.55
Ins-Ab	2.71	3.44
DS-Enc	1.75	3.17
DS-Ab	1.8	3.20

The possible combinations of the objectives are considered and the results are given in terms of Hyper Volume and Search Space. There are certain observations that can be inferred from the combinations of the objectives. Obtaining good optimized solutions in both hyper volume and search space is difficult. NSGA-II algorithm performs efficiently for the many objective problem.

c) Analysis of 4 Objective Problem

The problem is analyzed as a four objective problem and therefore the potential objectives are combined. With the rise within the objectives the algorithm provides an economical solution. The table shows the results of the four objective problem.

Table 4: Evaluation Results of Four Objective Problem

Objectives	NSGA-II HV	S
ICD-ERP-GCR-Ins	1.82	4.65
ICD-ERP-GCR-DS	2.24	3.47
ICD-ERP-GCR-Enc	1.83	4.05
ICD-ERP-GCR-Ab	1.32	5.1
ICD-ERP-Ins-DS	2.41	4.46
ICD-ERP-Ins-Enc	2.04	4.03
ICD-ERP-Ins-Ab	1.30	3.34
ICD-ERP-DS-Enc	2.04	3.92
ICD-ERP-Enc-Ab	1.42	4.22
ICD-GCR-DS-Enc	2.4	4.36
ICD-GCR-Ins-Enc	2.36	4.95
ICD-GCR-Ins-Ab	1.67	3.59
ICD-GCR-Enc-Ab	1.24	4.93
ICD-DS-Ins-Ab	1.61	4.80
ICD-DS-Ins-Enc	2.21	5.20
ICD-Ins-Enc-Ab	2.58	4.00
GCR-Ins-DS-Enc	1.48	3.65
ERP-GCR-Ins-DS	3.24	2.93
ERP-GCR-Ins-Enc	1.63	4.52
ERP-GCR-Ins-Ab	1.28	3.92
ERP-GCR-DS-Enc	1.85	4.62
ERP-GCR-DS-Ab	1.20	5.00
ERP-GCR-Enc-Ab	1.00	4.60
ERP-Ins-DS-Enc	1.21	4.56

d) Analysis of 6 Objective Problem

The problem is formulated as six objective problem and the results are obtained. The NSGA-II provides the best results in the following combinations which is shown in table 5.

Table 5: Evaluation Results of Six Objective Problem

Objectives	NSGA-II HV	S
ICD-ERP-GCR-Ins-DS-Enc	2.32	3.58
ICD-ERP-GCR-DS-Ins-Ab	2.20	2.20
ICD-ERP-GCR-DS-Ins-Enc	1.81	3.00
ICD-ERP-GCR-DS-Enc-Ab	2.82	2.34
ICD-ERP-GCR-Ins-Enc-Ab	2.62	2.21
ICD-ERP-DS-Inc-Enc-Ab	2.54	2.62
ERP-GCR-DS-Inc-Enc-Ab	2.1	3.00

7. Conclusion

In this proposed framework, multi objective evolutionary nsga-ii algorithm is used which simultaneously deals with different metrics at the architectural level, and provide a trade-off solution for all the possible combinations of the objectives. Two case studies have been considered the datapro4j and the marvin dataset. The problem is analyzed as multi objective problem and completely different values are thought-about for the population size, reflecting a range of formations for managing the 2-objective, 4-objective and 6-objective problems. Many design metrics have been considered to improve the efficiency. The results obtained have shown that the nsga-ii algorithm accomplishes an interesting trade off and competitive results for the considered objective combinations. These solutions are best suited to provide a support to the architects for the discovery process than the existing single objective evolutionary approach. The proposed work thus provides many design alternatives for the system considered and is more efficient than the previous working models

References

[1] S. Ducasse and D. Pollet, “Software Architecture Reconstruction: A Process-Oriented Taxonomy,”IEEE Trans. Softw. Eng., vol. 35, no. 4, pp. 573–591, 2009. https://doi.org/10.1109/TSE.2009.19.

[2] L. Dobrica and E. Niemela, “A survey on software architecture analysis methods,” IEEE Trans. Softw.Eng., vol. 28, no. 7, pp. 638–653, 2003. https://doi.org/10.1109/TSE.2002.1019479.

[3] D. Whitley, An overview of evolutionary algorithms: practical issues and common pitfalls. Inf. Softw. Technol. 43 (14) (2001) 817–831. https://doi.org/10.1016/S0950-5849(01)00188-4.

[4] C.L. Simons, I.C. Parmee, R. Gwynllyw, Interactive, evolutionary search in upstream object-oriented class design, IEEE Trans. Softw. Eng. 36 (6) (2010)798–816. https://doi.org/10.1109/TSE.2010.34.

[5] S. Kebir, A.-D. Seriai, A. Chaoui, S. Chardigny, Comparing and combining genetic and clustering algorithms for software component identification from object-oriented code, in: Proc. 5th Int. C* Conference on Computer Science and Software Engineering, 2012, pp. 1–8. https://doi.org/10.1145/2347583.2347584.

[6] Aurora Ramírez, José Raúl Romero, SebastiánVentura”An approach for the evolutionary discovery of software Architectures” Journal Information Sciences: an International Journal archive Volume 305 Issue C, June 2015 Pages 234-255.

[7] Bansiya and C. G. Davis, “A Hierarchical Model for ObjectOriented Design Quality Assessment,” IEEE Trans. Soft. Eng., vol. 28, no. 1, pp. 4–17, 2002. https://doi.org/10.1109/32.979986.

[8] V. L. Narasimhan and B. Hendradjaya, “Some theoretical considerations for a suite of metrics for the integration of software components,” Information Sciences, vol. 177, no. 3, pp. 844–864, 2007. https://doi.org/10.1016/j.ins.2006.07.010.

[9] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm NSGA-II,” IEEE Transactions

on Evolutionary Computation, vol. 6, no. 2, pp. 182–197, 2002. https://doi.org/10.1109/4235.996017.

[10] C. A. CoelloCoello, G. B. Lamont, and D. A.VanVeldhuizen, Evolutionary Algorithms for Solving Multi-Objective Problems. Springer, 2nd ed., 2007.

[11] F. Luna, D. L. Gonz´alez- ´Alvarez, F. Chicano, and M. A. Vega-Rodr´ıguez, "The software project scheduling problem: A scalability analysis ofmulti-objective metaheuristics," Applied Soft Computing, vol. 15, pp. 136–148, 2014. https://doi.org/10.1016/j.asoc.2013.10.015.