Asian Journal of Information Technology 15 (9): 1457-1467, 2016
ISSN: 1682-3915

Unsupervised Learning Technique Using Hybrid Optimization for Non-Functional Requirements Classification

[1]K. Mahalakshmi, [1]S. Manikandan, [1]S. Nithyanantham, [2]K.A. Sathiyaseelan and [2]P. Sudhakar
[1]Karpagam College of Engineering, Coimbatore, Tamil Nadu, India
[2]Department of CSE, Annamalai University, Chidhabaram,Tamil Nadu, India

Abstract: Problems in software engineering area can be solved mathematically. In this study, Support Vector Machine (SVM) are utilized to categorize Non-Functional Requirements (NFRs). The NFR-Classifiers are used to identify cross-cutting predominant framework toward disintegration found in necessities particular or early plan records are proposed. Optimization is used to acquire the best results under given circumstances. In order to improve the efficiency of SVM, Artificial Bee Colony (ABC) technique with Differential Evolution (DE) is used. The proposed technique improves the classification accuracy by 90.54% than existing techniques.

Key words: Support vector machine, non-functional requirements, radial basis function, artificial bee colony, differential evolution, hybrid optimization

INTRODUCTION

Software Engineering, as a designing order is a territory where there are issues that can be unraveled numerically. Such issues can be described by the quest for an answer in a space of conceivable arrangements. Such issues for the most part have high number of conceivable outcomes and high determination many-sided quality. In this way, the search process for these issues must be done through a computerized strategy and not through a thorough one, permitting those included in the process to attempt exercises where human limit is more advantageous (Freitas *et al*., 2010). "Search-based Software Engineering (SBSE)" from this moment, several researchers in collaboration with software engineers and system developers have modeled and solved various problems of software engineering using search techniques, especially meta-heuristics. Table 1 shows the spread of application of meta-heuristics to problems in this area.

Prerequisites Engineering (PE) is an exploration discipline perceived inside of Software Engineering as of now in the 1970's and spotlights on the deliberate level of programming advancement and the choice making in regards to what programming to construct. RE includes intertwined sub-procedures, for example, elicitation, detail, acceptance and prioritization and additionally advertise situated exercises identified with programming item administration including programming discharge arranging (Regnell and Kuchcinski, 2013).

Table 1: Phases and problems tackled in SBSE

Phase	Problem
Requirements	Requirements analysis
	Requirements selection
	Requirements planning
Testing	Test data generation
	Selection of test cases
	Prioritization of test cases
	non-functional testing
Estimation	Size estimation
	Cost estimation
Project planning	Resource allocation
	Allocation of Personnel
Code optimization	Parallelization
	Compilation optimization
Maintenance	Re-engineering software
	Automated maintenance
Compiler	Heap allocation
	Code size
Software project	Modularization

Combinatorial enhancement issues in PE happen in a few territories, including:

- Prioritization where an arrangement of prerequisites are evaluated in view of criteria, for example, advantage, cost and hazard, to discover the necessities of most astounding need, while adjusting the perspectives of chose partners
- Discharge arranging, where an arrangement of necessities are booked for resulting improvement in light of their needs and in addition asset imperatives and planning limitations, for example, priority and coupling, while attempting to upgrade perspectives, for example, partner advantage versus usage cost

Corresponding Author: K. Mahalakshmi, Karpagam College of Engineering, Coimbatore, Tamil Nadu, India

- Product offering displaying where an arrangement of variety focuses might incorporate distinctive blends of discretionary elements relying upon different limitations offering a group of items, while attempting to enhance viewpoints, for example, reuse SVM

NFRs portray the normal characteristics of a product framework, for example, convenience, security and look and feel of client interface. These qualities do sway on the building outline of the product framework and fulfillment of partners pertinent with the product framework. NFRs are advanced by partners with no ability in programming building and the quantity of NFRs is frequently vast. In addition, NFRs are scattered over the utilitarian prerequisites and the sorts of the NFRs are obscure on the grounds that they are composed in regular dialect. These attributes of NFRs make its characterization a work serious and tedious employment without the appropriation of programmed procedures. By and by, programming engineers can't overlook NFRs on the grounds that they are conclusive on the achievement of the product framework (Zhang *et al*., 2011).

The expanding programming unpredictability and rivalry in programming industry have highlighted the significance of NFR to be considered as essential part in programming advancement process. In the early period of programming advancement, programming necessity designing has ordered necessity as either practical prerequisites (FR) or NFR. All together for a product framework to be of a quality, it ought to meet both FR which characterize what the framework must have the capacity to perform and NFR which characterize how the framework will do it. Both prerequisites required appropriate thought all through the product improvement process (Too *et al*., 2013). Hence, NFR are constantly expressed casually in necessity archives, not considered enough and making them hard to uphold amid programming advancement. Therefore, the outline of a product framework may not mirror the NFR appropriately and difficult to accomplish clients requesting in quality programming. Writing has calling attention to blunders because of exclusion then again ineffectually managing NFR has prompted a progression of disappointments in programming advancement and are among the most troublesome, costly sort to adjust. There were a few genuine occurrences happened coming about because of ignoring NFR as a sample in London Ambulance catastrophe, the framework was deactivated soon after its sending because of some NFR were dismissed amid framework advancement.

NFRs are typically seen generally late being developed procedures. Stakeholder's quality limitations from necessities gathering procedure is archived through numerous antiques like reminders, meeting notes and meeting minutes with investigators for the most part neglecting to get an unmistakable picture on framework wide NFR. NFR-Classifiers identify and group early angles which are a sympathy toward cross-cutting prevailing framework decay found in prerequisites determination or early outline records. Numerous "early viewpoints" are like abnormal state NFRs like execution, security, movability and ease of use. Transitional level perspectives can be logging also, verification and lower-level concerns concentrate on automatic ideas such as buffering furthermore, reserving. The NFR grouping helps the designers to actualize changes and to cross check the nature of the framework (Mahalakshmi *et al*., 2014a).

The NFR Framework is a well-documented systematic method of handling decisions in regards to NFA during late requirements and early design. The NFR are presented as softgoals in a Softgoal Interdependency Graph that is progressively refined until the leaf-softgoals are clear and free of ambiguity. Leaf-softgoals are the lowermost nodes of the graph that is they have no children of their own. Methods of achieving these leaf-softgoals are then identified and the relationships defined by a qualitative relationship; these methods are called operationalizations and like softgoals can also be refined (Affleck *et al*., 2013).

Support Vector Machine (SVM) is a computer algorithm which assigns labels to objects learning by example. For example, a SVM can recognize fraudulent credit card activity through examination of thousands of fraudulent and non-fraudulent credit card activity reports. Alternatively, it can recognize handwritten digits through examination of a large collection of scanned images of handwritten zeroes, ones and so on. SVMs are also successfully applied to an increasingly variety of biological applications. SVM is a mathematical entity, an algorithm (or recipe) to maximize a particular mathematical function regarding a given data collection (Mahalakshmi *et al*., 2014b).

The Eq. 1 presents QP formulation for SVM classification. This is a simple representation:

$$\min_{f,\xi_i} \| f \|_K^2 + C\sum_{i=1}^{l}\xi_i \qquad (1)$$
$$y_i f(x_i) \geq 1-\xi_i, \text{ for all } i \; \xi_i \geq 0$$

SVM classification, Dual formulation in Eq. 2:

$$\min_{\alpha_i}\sum_{i=1}^{l}\alpha_i - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j y_i y_j K(x_i,x_j) \qquad (2)$$
$$\sum_{i=1}^{l}\alpha_i y_i = 0$$
$$0 \leq \alpha_i \leq C, \text{ for all } i$$

Variables ξ_i are called slack variables measuring the error at point (x_i,y_i). Training SVM is challenging when training points numbers are large. SVM, the best machine learning calculations and for the most part utilized for example acknowledgment is one of the. SVM strategy concocted the machine learning procedures of supervision. This grouping, exceedingly proficient and savvy in different applications are utilized. SVM machine learning is a mix-up. In the calculation that given an arrangement of preparing samples, each identified with one of the few sort as, a model that predicts that the new SVM preparing calculation fabricates a scope of illustration. SVM learning for the general issue which is going for more prominent factual limit. Design acknowledgment in view of learning information either needs or measurable data in various controls in the crude information, information detachment is an effective device that plans to characterize from the SVM (Joshi and Kale, 2014).

SVM with various pieces, for example, polykernel and RBF are utilized to group the NFR. The execution of the RBF part is indigent upon the parameters C and gamma (γ). The ideal estimations of these parameters are detailed as enhancement issue. "Swarm insight" and hereditary calculation are broadly utilized for advancement, to amplify or minimize the cost capacity. The most widely recognized swarm advancements depend on the aggregate conduct of the ants, fish educating or social conduct of winged creature running. ABC calculation is actualized for parameter determination of the RBF portion of SVM.

An enhancement model for the NFR Framework by first communicating the Softgoal Interdependency Graph as a coordinated diagram, then characterizing related constants and variables. Once the diagram is characterized the estimations for the given variables are laid out lastly the chart information is utilized to develop a goal capacity. The connections between softgoals, leaf softgoals and operationalizations are demonstrated utilizing the coordinated diagram G = (N, A). The Extended NFR Framework does not take into account this circumstance as need softgoals have a tendency to overwhelm the operationalization score and consequently its determination. The principle motivation behind the enhancement model is to enhance the fulfillment score for the predefined Softgoal Interdependency Graph.

Enhancement is the demonstration of getting the best results under given circumstances. Streamlining can be characterized as the procedure of finding the condition that gives the most extreme or least estimation of the capacity. On the off chance that x* relates the base estimation of capacity f(x), the same indicate additionally compares greatest estimation of the capacity -f(x). Hence, enhancement can be taken to mean minimization since the most extreme of the capacity can be found by looking for of the negative of the same number (Maleki *et al*., 2014).

Presently the meta-heuristic calculations are utilized tremendously as a part of cross breed advancement issues. One of the essential uses of these calculations is to contribute the enhancement and proficiency of the upgraded arrangements. Metaheuristic calculations are the calculations which seek the close advanced answers in the issues spaces falteringly. These calculations are extremely productive in tackling the hard and complex issues. A typical technique which can be utilized as a part of estimation of the product ventures is to utilize a favored capacity with algorithmic strategies for finding the estimations of the cost estimation which prompt the highest support. At the point when measurements of the issue go high by the expansion of the quantities of the components and the variables of the issue, the algorithmic techniques will be not able accomplish the genuine answers (Mausa *et al*., 2012).

Use of inquiry based advancement calculations in programming designing is a developing and separate territory of programming building called the hunt based programming designing (SBSE). It is available in a wide range of improvement or multi-target issues and each study demonstrates the benefits of taking care of an issue with such calculations. Numerous product designing issues have an excess of conceivable answers for an issue in their quest space for the thorough hunt to complete in sensible time. Infrequently it is adequate to discover one ideal arrangement or a "close ideal" answer for an issue. In both cases, seek based streamlining calculations, regularly alluded to as heuristic calculations, might discover its utilization in productively taking care of an issue or offering an understanding into the scope of conceivable arrangements (Storn and Price, 1997).

There are two key necessities for utilizing look based improvement calculations: seek space and wellness capacity. The pursuit space can be characterized as n-dimensional space where an object of enhancement can be found where n speaks to the quantity of variable parameters that characterize the item. The wellness capacity is a quantitative measure of the study's quality. The wellness capacity indicates incredible adaptability, since it is characterized by issue one is attempting to break down.

Advancement models for programming frameworks that are produced utilizing particular outline strategy. Four distinctive programming structures are considered: one program, no excess; one system with repetition; different projects, no repetition and various projects with

repetition. The enhancement issues are explained by utilizing our adaptation of set up streamlining strategies.

To encourage enhance the effectiveness of the SVM, Hybrid advancement technique in view of ABC and DE is proposed. The DE calculation is a parallel direct pursuit strategy, utilizing NP parameter vectors as populace for each era. DE is sorted into a class of coasting point encoded, transformative enhancement calculations. In no time, there are numerous DE variations; the particular variation utilized as a part of this examination is DE/rand/1/canister plan (Abraham *et al*., 2012).

Mixture plans are 2 sorts: organized pipelining sort cross breed and extra administrator sort half and half. In the principal, enhancement is connected to people in populace, trailed by more change utilizing DE look. In the second, DE is connected as standard hereditary administrator to relating likelihood where pipelining sort half breed strategy is utilized for its focal points where an era, after stochastic streamlining process (ABC) is connected to populace people. In populace, "n" best differential vectors from ebb and flow populace are picked in view of wellness qualities to create starting populace required for neighborhood look through DE. Seek proceeds till most extreme eras are come to or it fulfills a predefined measure (Rashwan *et al*., 2013).

Literature review: Rashwan *et al.* (2013) and Li and Huang (2009) proposed a Software Requirements Specification (SRS) contains all the requirements for a system-to-be. These are typically separated into FR and NFA. In order to be improved software development support, an automated analysis of SRS documents for different NFR types was required. Two significant contributions towards this goal: A new gold standard corpus containing annotations for different NFR types, based on a requirements ontology and a SVM classifier to be automatically categorized requirements sentences into different ontology classes. Results obtained from two different SRS corpora demonstrated that the effectiveness of the approach.

Li and Huang (2009) and proposed a hybrid approach that integrates the Self-Organizing Map and SVM for wafer bin map classification. The log odds ratio test was employed as a spatial clustering measurement preprocessor to distinguish between the systematic and random wafer bin map distribution. After the smoothing step was performed on the wafer bin map, features such as co-occurrence matrix and moment invariants are extracted. The proposed method can transform a large number of wafer bin maps into a small group of specific failure patterns and thus shorten the time and scope for troubleshooting to yield improvement. The experimental results showed that the approach can obtain over 90% classification accuracy and outperformed back-propagation neural network.

A 4 layered analysis approach to identify NFRs was suggested by Song *et al.* (2012) which had advantages over non-layered approach. Rules were suggested for use in every layer as part of the approach was successfully applied in 2 case studies. The identified NFRs were validated through a check list. A metric ensured the identified NFRs computation have been completed. Employing a software platform ensures higher degree software reuse by enabling multiple software products share platform-provided services. But, platform development involves stakeholders from various application domains.

Application situations vary and so NFRs for software platforms address wider needs than those of one product. Lessons learned in developing NFRs for large software platform was described by Xiping and Ameller *et al.* (2013) and the challenging issues and techniques which offset them was discussed. Such techniques are pragmatic helping with NFR reconciliation and management. Improved NFR specifications quality permitted automation of platform performance testing for 2 year.

Software architects work with incomplete/ill-specified NFRs using them to make decisions. Through this, existing NFRs are refined/modified and new ones emerge. Though research centered on how software architects treat NFRs, no empirical studies investigated this practice. A survey presented by Ameller *et al.* (2013) and Ramsin and Paige (2010) based on interviews with 13 software architects addressed 2 issues: how architects face NFRs from an engineering perspective and how NFRs influence decision-making? The survey exposed that architects elicit NFRs themselves through iteration. They don't document NFRs and validate them only partially.

An iterative method to elicit and specify SDM requirements using existing methodologies as supplementary resources was suggested by Ramsin and Paige (2010) and Wehrmeister *et al.* (2013). The method was the analysis phase of an engineering process methodology aimed at ultimate design and target methodology implementation. The finalized criteria highlighted qualities that target methodology had to have and was hence the basis to define a final requirements set. As an example, researchers demonstrated how the new elicitation process could identify requirements of a general object-oriented SDM. The new method helps methodology engineers produce a requirements set not only more complete in span but also concrete and rigorous.

Wehrmeister *et al.* (2013) and Solinas *et al.* (2013) presented a model-driven engineering approach combining Unified Modeling Language (UML) and Aspect Oriented Software Development (AOSD) to design real-time and embedded automation systems. The new approach allowed smooth transition from initial phases to implementation through use of software tools, comprising system specification and automatic source code generation. Experiments on using this approach to design real-world automation systems examples were presented. Results indicated a positive impact on automation systems design. NFRs encapsulation improved, increasing developed artifacts reuse. Generated source code statistics indicated the proposed approach generated fair amount of code per model element.

Incorporating security into every Software Development Life Cycle (SDLC) stage is a criterion that contributes leading to a conceptually secure SDLC. A special emphasis on including security from requirements elicitation was added by Solinas *et al.* (2013) and Chung and Leite (2009). In everything, security treatment must be approached with criteria to ensure that evidence of best practices is used for construction and best solutions to meet security requirements. The new work presented criteria to build secure software, incorporating Security Patterns, from requirements elicitation phase.The state of the art on treating NFRs, when providing prospects for future directions was reviewed by Chung and Leite (2009) and Slankas and Williams. Functional and non-functional characteristics determined software system utility. Flexibility, performance, usability, interoperability and security add to score. There is lop-sided emphasis on software functionality now, though not useful/usable sans non-functional characteristics.

Slankas and Williams (2013) and Lu *et al.* (2008) examined document types (installation manuals, data use agreements, regulations, requirements specifications, proposals requests and user manuals) having NFRs in 14 NFR categories (reliability, capacity and security) measuring how to identify/classify NFR statements in those documents effectively. NFRs were presented in evaluated documents. A support vector machine algorithm, using a NFR word vector representation performed twice as successfully compared to a multinomial Naïve Bayes classifier. k nearest neighbor classifier with unique distance metric with F1 measure of 0.54, outperformed optimal Naive Bayes classifier with F1 measure of 0.32 in experiments. It was found that stop word lists beyond common determiners lacked minimal performance effect.

A requirement tool to support model based re presenting a Model-driven Object-oriented Requirement editor (MOR Editor) and supported requirement document modelling and model-driven document editing was proposed by Lu *et al.* (2008) and Pandey *et al.* (2010). MOR Editor supported link related requirement artifacts and RE objectized requirement artifacts, ensuring reusable requirement template and creating a requirement document model integrating artifacts in analysis, design and implementation. Requirement documents traceability, consistency, completeness and maintainability was enhanced with MOR Editor's support. A case study using MOR Editor proved the merit of the new requirement editor. An effective RE process model for software development and requirements management was proposed by Pandey *et al.* (2010), Alizadegan and coauthors, Mahalakshmi and Prabhakar (2014a,b) who suggested an effective RE process model for software development QR production. Requirement management/planning phases were independently executed for good requirements management. Implementation of the new RE process impacted positively on software product production quality.

MATERIALS AND METHODS

In this study, a hybrid advancement procedure is proposed to enhance the productivity of the SVM. Generally populace based streamlining calculations, experience the ill effects of long computational times in view of their developmental/stochastic nature. Hybridization empowers getting the best favorable circumstances of nature enlivened heuristic strategies and wiping out weaknesses such as untimely joining and computational time. In this study another Hybrid ABC calculation which joins with DE is proposed to advance the parameters C and γ. Streamlining strategies are adjusted to execute the quest for ideal mix of (C, γ). The target capacity depends on Root Mean Squared Error (RMSE) accomplished by the SVM-RBF. In this way, the enhancement finds the mix of (C, γ) with the most minimal RMSE to enhance the execution.

Differential Evaluation (DE): DE is a population based inquiry system working ceaseless areas and connected effectively to various issues. DE hunt down worldwide optima utilizing contrasts between contemporary populace individuals which allows every individual pursuit conduct to self-tune. Till now, DE pulled in consideration and applications in different fields. At all DE calculation cycles, each populace of arrangements part

is considered as an objective thusly for substitution in resulting era. New hopeful arrangements are produced by including weighted distinction between two haphazardly picked populace individuals to a third, arbitrarily picked populace part, alluded as the base.

An improvement undertaking of D parameters is spoken to by a D-dimensional vector. In DE, a NP arrangement vectors populace is arbitrarily made at first and effectively enhanced over G eras through utilization of transformation, hybrid and determination administrators, to achieve ideal arrangement. The DE flowchart is appeared in Fig. 1. The techniques determined in it are clarified in taking after segments.

Initialization: Here, every decision parameter in each initial population vector is assigned a randomly chosen value from corresponding feasible bounds as in Eq. 3:

$$X_{j,i}^{(0)} = X_j^{min} + \mu_j(X_j^{max} - X_j^{min}), \quad i=1,....,N_P,\ j=1,.....,D \tag{3}$$

Where, μ_j is the uniformly distributed random number within the range (0, 1), generated a new for each value of j. The X^{max}_j and X^{min}_j are the upper and lower bounds of the jth decision parameter, respectively.

Mutation: This operator creates mutant vectors X'_i perturbing a randomly selected vector X_a with difference of two other randomly selected vectors X_b and X_c, based on following expression in Eq. 4:

$$X_i'^{(G)} = X_a^{(G)} + F(X_b^{(G)} - X_c^{(G)}), \quad i=1,....N_P \tag{4}$$

Where, a, b and c are randomly chosen indices, so that a, b, c $\in$ {1,...,N_P} and a≠b≠c≠i. Generation of new (random) values for a, b and c of I is to be noted. F is scaling factor where an algorithm control parameter in range (0, 2) adjusts mutation operator perturbation size and improves algorithm convergence.

Crossover: This increases diversity among mutant parameter vectors. A trial vector X''_i is created from

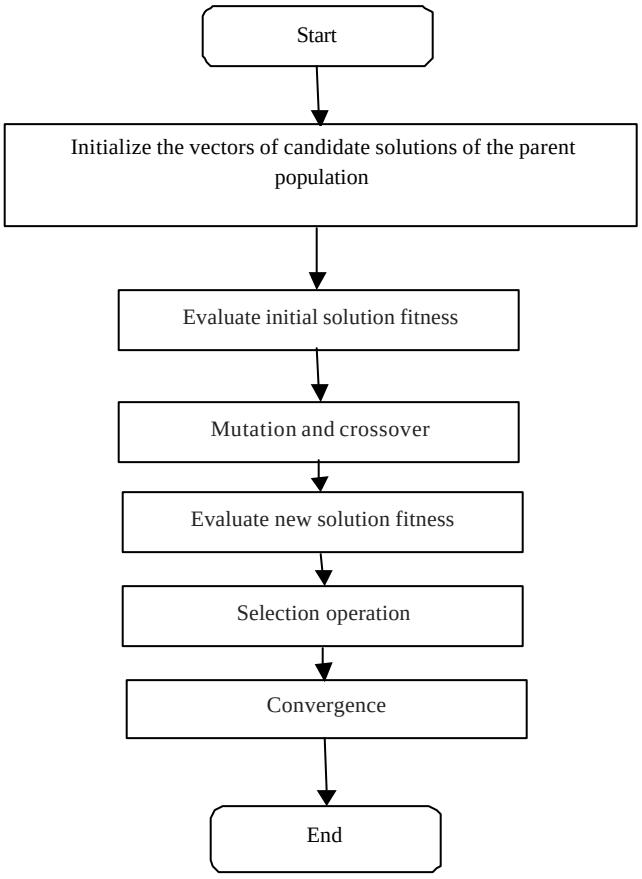

Fig. 1: Flow chart for differential evolution

components of every mutant vector X'_i and its corresponding target vector X_i, based on a series of D-1 binomial experiments as in Eq. 5:

$$X''^{(G)}_{j,i} = \begin{cases} X'^{(G)}_{j,i} & \text{if } \rho_j \le C_R \text{ or } j=q \\ X^{(G)}_{j,i} & \text{otherwise} \end{cases}, \; i=1,....N_P, \; j=1,....,D \quad (5)$$

Where, ρ_j denotes a uniformly distributed random number represented within range (0, 1), generates a new for every value of j. Cossover constant C_R is chosen from within range (0, 1), is an algorithm parameter controlling population diversity helping it to escape from local minima. q, a randomly chosen index $x \in \{1,...,D\}$, ensure that trial vector gets minimum one parameter from mutant vector.

Selection: This operator forms population choosing between trial vectors and predecessors (target vectors) those individuals which present a better fitness or are more optimal:

$$X^{(G+1)}_{j,i} = \begin{cases} X''^{(G)}_{j,i} & \text{if } f(X''^{(G)}_i) \le f(X^{(G)}_i) \\ X^{(G)}_i 7 & \text{otherwise} \end{cases}, i=1,....N_P \quad (6)$$

The optimization process is repeated for many generations which ensure an individual improve fitness as they explore solution space searching for optimal values.

Proposed hybrid Artificial Bee Colony-Differential Evolution (ABC-DE): In this method, a generation, after stochastic optimization process (ABC) is applied to all population individuals. In the population, n best differential vectors from current population are selected based on fitness values to generate initial population needed for local search via DE. Bee population of solutions, x1, 1 = 1,…,SN.

Fitness function lowest Root Mean Squared Error (RMSE) (ABC-DE is adapted to execute the search for optimal combination of (C, γ). The objective function is based on RMSE attained by the SVM-RBF). ABC uses following to produce a candidate food position from old one in memory:

$$v_{ij} = x_{ij} + \phi_{ij}\left(x_{ij} - x_{kj}\right) \quad (7)$$

Where, $k \in (1,...,SN)$ and $j \in (1,...,D)$ are randomly chosen indices D being the number of optimization parameters. Though k is determined randomly, it is different from i. Øij is a random number between (-1, 1). In DE, Population P of generation G contains NP (population size of DE) solution vectors called individuals population and each vector represents potential solution for optimization problem:

$$P^{(G)} = X^{(G)}_i = X^{(G)}_{j,i} \quad i = 1,..,NP, \quad j = 1,...,D, G = 1,...,G_{max} \quad (8)$$

The ABC-DE algorithm has three types of bees: employed, onlooker and scout bees. One half of the colony includes employed bees and the other half onlooker bees. For each food source (solution), there is one employed bee i.e. the employed bee's number equals the number of food sources. The employed bees produce new solutions from current solutions by Eq. 9.

$$V^{(G+1)}_{j,1} = \begin{Bmatrix} x^{(g)}_{j,r3} + F*\left(x^{(G)}_{j,r1} - x^{(G)}_{j,r2}\right), \text{ if } rand_j[0,1] < CR \text{ or } j = K, \\ x^{(g)}_{j,1} \qquad \text{otherwise} \end{Bmatrix} \quad (9)$$

Where, r1,r2, $r3 \in (1,NP)$ randomly chosen vectors of population also $r1 \neq r2 \neq r3 \neq i$, k = (int(randi(0,1)*D)+1) and $CR \in (0,1)$, $F \in (0,1)$. F is a real-valued factor in range (0.0, 1.0) that controls amplification of differential variations. CR is a real valued crossover factor in range (0.0, 1.0) that controls the probability.

Based on probability, onlooker bees choose solutions with better fitness producing new solutions by (de Freitas *et al*., 2010). Solution choosing probability is calculated by Eq. 10:

$$p_i = \frac{fit_i}{\sum_{n=1}^{sn} fit_n} \quad (10)$$

Where:
fit_i = The fitness function

The new solutions replace current solution, if new solution's fitness is equal or better than current one; this being done Eq. 11:

$$x^{(G+1)}_i = \begin{cases} V^{(G+1)}_i & \text{if } V^{(G+1)}_i \le \beth\left(X^{(G)}_i\right) \\ X^{(G)}_i & \text{otherwise} \end{cases} \quad (11)$$

Where:
$\beth(X^{(G)}_i)$ = The cost function

If a better solution cannot be produced by employed or onlooker bees, after some iterations determined by control parameter limit, employed bee converts to scout and initializes solution randomly. As in DE, control parameters F and CR are ABC-DE algorithm's important elements. These control parameters play a big role to produce new solutions. As observed, the ABC-DE algorithm's structure is similar to ABC; whereas the way of producing a new solution is similar to DE. In fact, ways to produce new solutions in DE are embedded in the ABC algorithm itself (Mahalakshmi and Prabhakar, 2014a,b).

The pseudo code of ABC-DE algorithm is shown below:

Initialize the population of solutions xi , i = 1, . . . ,SN
Evaluate the population
cycle = 1
Repeat
Step 1: Produce new solutions ti for the employed bees by using

$$V_{j,1}^{(G+1)} = \begin{Bmatrix} x_{j,r3}^{(g)} + F^{*}\left(x_{j,r1}^{(G)} - x_{j,r2}^{(G)}\right), \text{ if } rand_j[0,1] < CR \text{ or } j = K, \\ x_{j,1}^{(g)} \qquad \text{otherwise} \end{Bmatrix}$$

where
$i \in [1,NP]; j \in [1,D]; r_1, r_2, r_3 \in [1,NP]$,
randomly selected, except: $r_1 \neq r_2 \neq r_3 \neq i$, $k = (int(rand_i[0,1] \times D) + 1)$,
and $CR \in [0,1]$, $F \in [0,1]$.
r_1, r_2 and r_3 refers to three randomly
chosen vector population and evaluate new solution produced.
Step 2: Apply the greedy selection process for the employed bees by using

$$X_i^{(G+1)} = \begin{cases} V_i^{(G+1)} & \text{if } \gamma(V_i^{(G+1)}) \leq \gamma(X_i^{(G)}) \\ X_i^{(G)}, & \text{otherwise} \end{cases}$$

where $\gamma(X)$ refers to the value of cost function
that to be minimized will propagate the population of the next generation.
Step 3: Calculate the probability values Pi for the solutions xi by

$$p_i = \frac{fit_i}{\sum_{n=1}^{sn} fit_n}$$

where fit_i is the fitness value of solution i that is
proportional to the nectar amount of the food source in the position i
and SN is the number of food source that is equal to number of employed bees (BN).
Step 4: The new solutions replace current solution, if new solution's fitness is
equal or better than current one; this being done by

$$x_i^{(G+1)} = \begin{cases} V_i^{(G+1)} & \text{if } V_i^{(G+1)} \leq \beth\left(X_i^{(G)}\right) \\ X_i^{(G)} & \text{otherwise} \end{cases}$$

$\beth\left(X_i^{(G)}\right)$ is the cost function.
Step 5: Produce the new solutions ti for the onlookers from the solutions xi
selected depending on Pi (equation used in step 1) and evaluate them
Step 6: Apply the greedy selection process for the onlookers by using
(equation used in step 2)
Step 7: Determine the abandoned solution for the scout, if exists and replace
it with a new randomly produced solution xi by

$$x_i^j = x_{min}^j + rand(0,1)(x_{max}^j - x_{min}^j)$$

Step 8: Memorize the best solution achieved so far
cycle = cycle + 1
untilcycle = Maximum Cycle Number (MCN)

RESULTS AND DISCUSSION

NFR dataset available in the promise data repository is used to evaluate the proposed methodology. The experiments are conducted for proposed ABC feature selection and SVM with ABC DE optimization.

Table 1 tabulates the classification accuracy for SVM with ABC DE optimization and SVM with ABC optimization for without feature selection, IG based feature selection and ABC based feature selection are used. Figure 2 shows the percentage of features selected by the proposed ABC-DE optimization.

Table 2 and Fig. 3 depicts the classification accuracy for SVM with ABC DE optimization and SVM with ABC optimization for without feature selection, IG based feature selection and ABC based feature selection are used.

Figure 3 shows that the proposed hybrid ABC-DE with ABC based feature selection shows high classification accuracy of 90.54%. It is observed that the proposed ABC-DE improves the classification accuracy in the range of 1.96-2.78% when compared to SVM with ABC optimization. The proposed ABC-DE achieves higher classification accuracy for ABC based feature selection by 3.42% when features are selected and by 1.6% when IG feature selection is used.

The Precision, Recall and F measure for the techniques used such as performance, look and feel, usability, availability, security, functional, fault tolerance, scalability, operational, legal and maintainability for without feature selection, IG based feature selection and

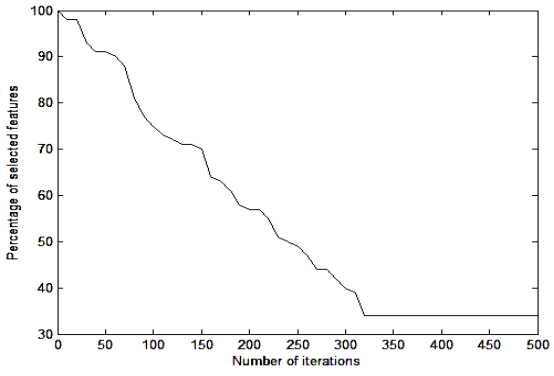

Fig. 2: Percentage of features selected by the proposed ABC

Table 2: Classification accuracy

Parameters	Without feature selection	IG based feature selection	ABC based feature selection
SVM with ABC DE optimization	87.5	89.1	90.54
SVM with ABC optimization	85.1	87.18	88.78

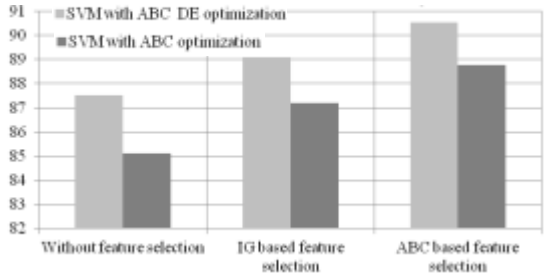

Fig. 3: Classification accuracy

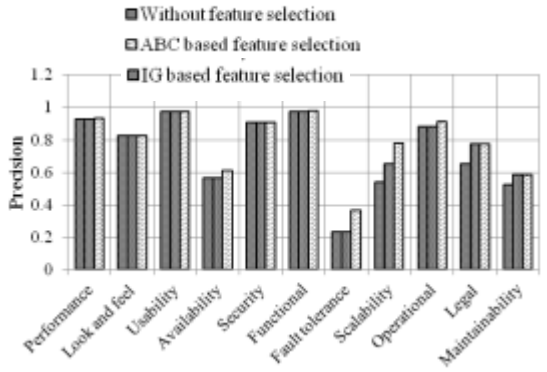

Fig. 4: Average precision

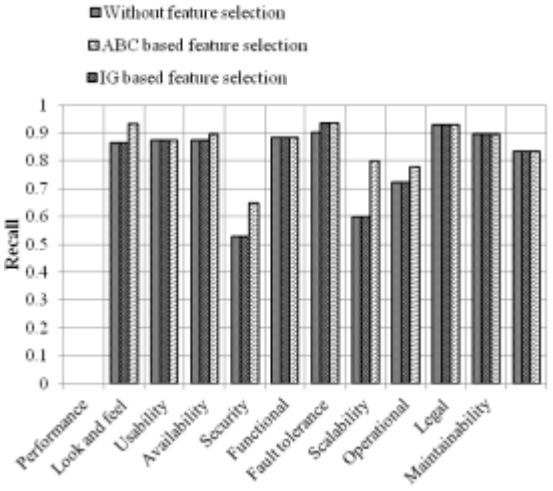

Fig. 5: Average recall

Table 3: The precision for the NFR Class

NFR parameters	Without feature selection	IG based feature selection	ABC based feature selection
Performance	0.864	0.864	0.932
Look and feel	0.872	0.872	0.872
Usability	0.872	0.872	0.894
Availability	0.529	0.529	0.647
Security	0.884	0.884	0.884
Functional	0.901	0.936	0.936
Fault tolerance	0.600	0.600	0.800
Scalability	0.722	0.722	0.778
Operational	0.930	0.930	0.930
Legal	0.895	0.895	0.895
Maintainability	0.833	0.833	0.833

Table 4: The recall for the NFR class

Parameters	Without feature selection	IG based feature selection	ABC based feature selection
Performance	0.927	0.927	0.932
Look and feel	0.829	0.829	0.829
Usability	0.976	0.976	0.977
Availability	0.563	0.563	0.611
Security	0.910	0.910	0.910
Functional	0.973	0.974	0.978
Fault tolerance	0.231	0.231	0.364
Scalability	0.542	0.650	0.778
Operational	0.883	0.883	0.914
Legal	0.654	0.773	0.773
Maintainability	0.526	0.588	0.588

ABC based feature selection are shown in Table 3 and 4. Figure 4 and 5 show the Precision, Recall and F-Measure for the various techniques respectively.

CONCLUSION

RE process model is iterative process that is continual throughout the project. RE exercises happen crosswise over numerous stages, making process models seem iterative. The important step in NFA is to identify and classify NFA. This study utilizes SVM for grouping, the RBF bit is upgraded utilizing Hybrid ABC made up of ABC took after by DE. The proposed hybrid and hybrid ABC-DE with ABC based component choice shows high accuracy of 90.54%. The proposed ABC-DE improves the classification accuracy between 1.96-2.78% when compared with SVM with ABC streamlining. The proposed ABC-DE accomplishes higher classification accuracy for ABC based element choice by 3.42% when elements are chosen and by 1.6% when IG highlight determination is utilized. It is watched that the proposed cross breed ABC-DE SVM accomplishes 9.45% change in order exactness when contrasted with Bagging with Random Forest. Thus it is observed from the results that the proposed crossover ABC-DE SVM enhances the precision by 1.96% contrasted and SVM with ABC improvement.

REFERENCES

Abraham, A., R.K. Jatoth and A. Rajasekhar, 2012. Hybrid differential artificial bee colony algorithm. J. Comput. Theor. Nanosci., 9: 249-257.

Affleck, A., A. Krishna and N.R. Achuthan, 2013. Optimal selection of operationalizations for non-functional requirements. Proceedings of the Ninth Asia-Pacific Conference on Conceptual Modelling-Volume 143, January 29-February 1, 2013, Australian Computer Society, Inc., Darlinghurst, Australia, ISBN: 978-1-921770-28-9, pp: 69-78.

Ameller, D., C. Ayala, J. Cabot and X. Franch, 2013. Non-functional requirements in architectural decision making. Software IEEE., 30: 61-67.

Chung, L. and J.C.S.D.P. Leite, 2009. On Non-Functional Requirements in Software Engineering. In: Conceptual modeling, Foundations and applications. Alexander, T.B., K.C. Vinay, G. Paolo and S.Y. Eric (Eds.). Springer Berlin Heidelberg, Berlin, Germany, ISBN: 978-3-642-02462-7, pp: 363-379.

Freitas, F.G.D., B.C.L. Maia,. G.A.L.D. Campos and J.T.D Souza, 2010. Optimization in software testing using metaheuristics. J. FSMA. Inf. Syst., 5: 3-13.

Joshi, S.S. and N.D. Kale, 2014. Survey: support vector machine and Its deviations in classification techniques. Intl. J. Adv. Res. Comput. Sci. Software Eng., Vol. 4,

Li, T.S. and C.L. Huang, 2009. Defect spatial pattern recognition using a hybrid SOM-SVM approach in semiconductor manufacturing. Expert Syst. Appl., 36: 374-385.

Lu, C.W., C.H. Chang, W.C. Chu, Y.W. Cheng and H.C. Chang, 2008. A requirement tool to support model-based requirement engineering. Proceedings of the 32nd Annual IEEE International Conference on Computer Software and Applications, July 28-August 1, 2008, IEEE, Turku, Finland, ISBN: 978-0-7695-3262-2, pp: 712-717.

Mahalakshmi, K., D.R. Prabhakar and D.V. Balakrishnan, 2014a. Kernel Optimization For Improved Nonfunctional Requirements Classification. J. Theor. Appl. Inf. Technol., 60: 64-72.

Mahalakshmi, K., R. Prabhakar and V. Balakrishnan, 2014b. Optimizing support vector machine for classifying non functional requirements. Res. J. Appl. Sci. Eng. Technol., 7: 3643-3648.

Maleki, I., A. Ghaffari and M. Masdari, 2014. A new approach for software cost estimation with hybrid genetic algorithm and ant colony optimization. Int. J. Innov. Applied Stud., 5: 72-81.

Pandey, D., U. Suman and A.K. Ramani, 2010. An effective requirement engineering process model for software development and requirements management. Proceedings of the International Conference on Advances in Recent Technologies in Communication and Computing, October 16-17, 2010, Kottayam, pp: 287-291.

Ramsin, R. and R.F. Paige, 2010. Iterative criteria-based approach to engineering the requirements of software development methodologies. Software IET., 4: 91-104.

Rashwan, A., O. Ormandjieva and R. Witte, 2013. Ontology-based classification of non-functional requirements in software specifications: A new corpus and SVM-based classifier. Proceedings of the 2013 IEEE 37th Annua Computer Software and Applications Conference (COMPSAC), July 22-26, 2013, IEEE, Kyoto, Japan, pp: 381-386.

Regnell, B. and K. Kuchcinski, 2013. A scala embedded DSL for combinatorial optimization in software requirements engineering. Proceedings of the First Workshop on Domain Specific Languages in Combinatorial Optimization, September 16-20, 2013, COSpeL, Uppsala, Sweden, pp: 19-34.

Slankas, J. and L. Williams, 2013. Automated extraction of non-functional requirements in available documentation. Proceedings of the 2013 1st International Workshop on Natural Language Analysis in Software Engineering (NaturaLiSE), May 25-25, 2013, IEEE, San Francisco, Californa, pp: 9-16.

Solinas, M., L. Antonelli and E. Fernandez, 2013. Software secure building aspects in computer engineering. Lat. Am. Trans. IEEE., 11: 353-358.

Song, X., B. Hwong and J. Ros, 2012. Lessons from developing nonfunctional requirements for a software platform. Software IEEE., 29: 74-80.

Storn, R. and K. Price, 1997. Differential evolution-A simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim., 11: 341-359.

Too, C.W., S.A. Hassan, J. Din, A. Ghani and A. Azim, 2013. Towards improving NFR elicitation in software development. Intl. J. Inf. Technol. Comput. Sci., 7: 33-44.

Wehrmeister, M.A., C.E. Pereira and F.J. Rammig, 2013. Aspect-oriented model-driven engineering for embedded systems applied to automation systems Ind. Inf. IEEE. Trans., 9: 2373-2386.

Zhang, W., Y. Yang, Q. Wang and F. Shu, 2011. An empirical study on classification of non-functional requirements. Proceedings of the Twenty-Third International Conference on Software Engineering and Knowledge Engineering (SEKE 2011), July 7-9, 2011, Eden Roc Renaissance Hotel Miami Beach, USA., pp: 190-195.