Metaheuristic Search Approach Based on In-house/Out-sourced Strategy to Solve Redundancy Allocation Problem in Component-Based Software Systems

Adil A.A. Ssaed, Wan M. N. Wan Kadir and Siti Zaiton Mohd Hashim

Department of software Engineering
Faculty of Computer Science & Information Systems, Universiti Teknologi Malaysia
81310 Johor, Malaysia
adil_sudan@hotmaill.com, wnasir@utm.my, sitizaiton@utm.my

Abstract

The redundant components' nodes are widely used as design tactic to improve the reliability of complex embedded systems. The challenge is that; the obtained solution might not be sufficient because it demands high cost, and the system might be failed if the predicted cost does not estimated reasonably. Many publications have addressed this area to compose the systems by assembling COTS components only. However, combination of outsourced components and in-house developed components is essential to produce cost effective systems. This paper proposes an architecture optimization approach based on Metaheuristic Swarm Intelligent algorithm in order to answer the question: What type of components to be selected and how many components are appropriate for each subsystem in order to obtain optimal software architecture. The objective is to minimize cost under reliability constraints by combining components from different sources in order to compose system that satisfies the requirements. Sensitivity analysis applied on a numerical simple case study showed the potentiality of the approach.

Keywords: *Optimization, Reliability Redundancy, Particle Swarm Optimization (PSO), quality, Cost, Design Decision, Redundancy Allocation Problem (RAP)*

1. Introduction

Recently, Component-based system has received an increased interest to develop software systems. However, reliability prediction, cost estimation as well as architecture optimization became more important for composing such systems. Early evaluation of architecture helps architects to: detect the probable contradictions between solutions; for obtaining higher reliability, aids to identify the appropriate number of the components within each subsystem. Architects are taking right design decision by selecting optimal design that satisfies the requirements. Optimizing system architecture that satisfies the functional requirements while keeping the quality attributes at high levels is still an open challenge [1].

The redundant components' nodes are widely used as design tactic to improve the reliability of complex embedded systems. Architects often deal with systems having elements configured as a combination of parallel and series system model. To evaluate system reliability, architect breaks the system down into subsystems; each sub system consists of one or several components. Then, reliability of each subsystem could be calculated. Finally, put these single units back into a system and get its reliability according to serial or parallel considerations. In the composite model, the impact of an individual component should be analyzed on the overall behavior of the software. For such analysis, the combined model should be reconstructed and resolved again and again to obtain the revised predictions. The first challenge is that how optimal design can be selected among thousands or millions of possible architecture's candidates. In general, there are two scenarios in the case of choosing alternatives subject to cost/reliability tradeoffs. The first scenario is to minimize the software cost for a given reliability target. The second scenario is to maximize the software reliability while staying within the pre–specified cost limits. Minimizing software cost for a given reliability threshold is important in the context of systems where the cost target is not negotiable [2]. However, in the case of many commercial software systems, cost is the most important driving factor, and the problem then is to achieve minimum cost while reliability within the predefined threshold. The second challenge is that combination of outsourced components and in-house developed components might be used to compose a system. In this case, the complexity of representing the new variables such as the time-to-deliver components and the required number of test for in-house developed components must be considered because their parameter's impact on the quality of the composite system. Therefore, these parameters should be balanced to meet the predefined requirements.

In this paper, we present an architecture optimization approach based on a Swarm Intelligent - Particle Swarm Optimization (PSO). The algorithm used to minimize the cost under delivery-time and reliability constraints. Combination of components as a source for the components that the system composed of is considered. A simple case study is demonstrated and sensitivity analysis in performed. The proposed approach has three main stages: First, we transfer the architecture model to an optimization problem; secondly, we run an intelligent optimization technique to search and evaluate alternative's design architecture. The optimization technique is used to automate the selection of the best design. This can assist architects to take right design decisions on the early stage of development; finally, an architect can perform tradeoff on solutions resulted from the previous step to decide on the architecture.

2. Related Work

This work is related to researches presented in [3] and [4]. The study in [3] has proposed two optimization models for selecting the best COTS products for the modules in a software system development. The approach used AHP to assign weights for the modules. The disadvantage of the AHP is that the number of pair-wise comparisons to be made, may become very large (n (n−1)/2), and thus become a lengthy task. The study has employed a contingent decision strategy for constraints. The contingent identified as follow: "There is no best way to make decisions". The effective decision making style that effect some situations, may be not successful in other situations. That means an optimal decision making style depends upon various internal and external constraints. This strategy allows flexibility to select the next step based on conditional situation. Based on this strategy, Parametric decision variables to represent the amount of testing on each component has been developed in [1] to represent the amount of testing on each component. The authors proposed an optimization approach that combined in-house developed component and out-source component (COTS). LINGO tools, which are a non-linear model solver, have been used to calculate the results. Others disadvantage of this approach is are: It does not search all design space; and no solutions found for complex and large case; complexity of and limitations to represent complex cases; scenarios only was applied; and the redundancy component was not supported.

The redundancy allocation in automotive industry has attracted many researchers. In [4] the redundancy allocation problem is handled to improve reliability. The author has applied the approach to Anti-Lock Brake System (ABS). He suggested in his approach COTS as a source for components to compose the systems. Moreover, the approach involves response time as constraint and uses Ant Colony Algorithm (ACO) algorithm as metaheuristic technique in order to search and optimize the architecture.

Fewer approach focus on optimizing architecture using combination of in-house component and out-source component (COTS). The complexity of such case is that; there are several factors may impact cost and reliability of in-house components. For example, numbers of test on component impact on its reliability and developers' skills have an impact on the time to deliver the component. Our approach is closed to approach which introduced in [1]. The novelty of our approach is that: Our approach considered the redundancy level on each sub system; we explore all design space rather than selected scenarios; and tradeoff on multiple options can be obtained since we employ an intelligent metaheuristic, PSO algorithm, search technique. The proposed approach aids architects not only to decide on level of redundancy but also to decide on which component should be developed in-house and which one should be outsourced. Therefore, the proposed approach contributes to reduce the time to develop systems, save development cost, and aid to control the way the systems are composed to fulfill the quality requirements.

3. Particle Swarm Optimization (PSO)

Since, meta-heuristics evolutionary techniques such as Genetic Algorithms (GAs) methods have proven its usefulness to solve the problem of spanned design space; we elect the Particle Swarm Optimization (PSO) to be used in the proposed approach. In recent researches Swarm Intelligent (SI) techniques such as PSO [5, 6, 7] an alternative search technique, often performed better than many evolutionary techniques such as GA when applied to various problems [8, 9]. In this paper, we introduce the implementation of PSO to search design space. Our objectives from using PSO are to search the design space according reliability/cost preferences, in addition to automatically generate and evaluate the alternative's design. Then architects can reason on the provided options and choose the optimal solution that satisfies the specified quality requirements.

Particle Swarm Optimization (PSO) is one of the swarm intelligent (SI) optimization methods, it's an adaptive optimization method introduced in 1995 by Kennedy and Eberhart [10, 11]. The method inspired by social behavior of swarms such as bird flocking or fish schooling. In PSO, particles never die, and particles are considered as simple agents that move and interact through the search space and record the best solution that they have visited. Each particle represents a candidate solution in the solution search space and each particle has a position vector and velocity. The behavior of the particles is subjected to their capability to train from their past personal experience and from the success of their neighbor to adapt the flying speed and direction to the target. However, particles manage the current position, velocity and personal best position. Beside the personal best solution, the swarm is targeting the global best solution

There are two key steps when applying PSO to optimization problems: the representation of the solution and the fitness function. One of the advantages of PSO is that PSO take real numbers as particles. It is not like GA, which needs to change to binary encoding, or special genetic operators have to be used.

Let us assume an optimization problem with a single objective function Y denoted as f (x). f is an objective function to be minimized or maximized, the minimization problem is taken as general, where

$Y = \min f(x),\ x \in X$

Where Y is an objective space which contains an objective vector y. X is parameter space that has x decision vector. An n is a number of decision vectors.

The function has domain constraint. In addition, strategy of how to deal with the velocity is needed, (example of strategy: delete, edit, modify, and penalty). The domain constraint is a set of constraints, which is denoted as:
$Dc = \{dc_1, \ldots, dc_n\}$
Thus, the minimization problem expressed as follows:
$Y= f(x),\ x \in \{x_1, x_2, \ldots, x_n\}$

The above minimization problem is subjected to the following constraints
$dc_i(x) > 0$
$dc_i(x) = 0$

Example:
$\min [(x_1-3)^2 + (x_2-1)^2]$ *(1)*
Subject to
$x_1^2 - x_2^2 < 0$ *(2)*
$x_1 + x_2 = 1$ *(3)*

If N defined as number of particles that flying in design space of D-dimensional. Each particle i, re-positions itself xi units in a direction expressed by pi = (pi1, pi2. . . piD). And the best position achieved by the whole swarm (gBest), for a given subset of the swarm which denoted as pg = (pg1, pg2. . . pgD). The difference between the current position of the particle i and the best position of its neighborhood expressed by (pg − xi). For simplification the two equations are:

v [i] =

w*v[i] + c1 * rand() * (pbest [i] – present [i]) + c2 * rand() * (gbest [i] – present [i]) (4)

present [i] = persent [i] + v [i] (5)

Where w is inertia weight, c1 and c2 are learning factors (weights)

The parameters used in algorithm are:

c2: acceleration factor related to lbest =1, c1: acceleration factor related to gbest =3, w=0.4
Finally, the processed to search design space using PSO explained in the following algorithm;

```
For each particle
Initialize particle with feasible random number
END
Do
    For each particle
        Calculate the fitness value
        If the fitness value is better than the best fitness value (pBest) in history
            Set current value as the new pBest
    End
    Choose the particle with the best fitness value of all the particles as the gBest
    For each particle
        Calculate particle velocity according to velocity update equation
    Update particle position according to position update equation
End
    Until (maximum number of iterations) or (condition be satisfied).
```

4. The Proposed Approach

The following example demonstrates a general case study which performed to minimize the cost under delivery-time and reliability constraints. Systems usually are composed of a predefined number of components. There are two different sources for these components: In-house components and out-source components (COTS). There is a level of redundancy within each component. The cost, delivery time and reliability attributes of outsource components are provided by the vender. Architect can use either COTS or an in-house component aiming to minimize the software cost under delivery time and reliability constraints. However, identifying the required amount of testing on each in-house instance to achieve a reliability level that satisfies the reliability of the whole system is essential. The selection of components for each component (subsystem) should be optimized considering the source of component and the redundancy level. We use the PSO to automatically search design space for optimal solution.

We are aiming to optimize the selection of instance for each component. Either one type of the available COTS or in-house component can be selected to minimize the software cost under delivery time and reliability constraints. The redundant components for each subsystem are similar to the selected ones. Furthermore, the amount of testing required for each in-house instance is required to achieve a reliability level that satisfies the reliability of the whole system.

Objective Function Formulation

Assume S is Software Architecture to be optimized and it composed of n components, and the maximum number of available components for each component is m. The objective function formulated to minimize cost denoted as follow:

$$MinCost = \sum_{i=1}^{n} \left(C_i(t_i + \tau_i \quad Ntest_i)y_i * (\mathrm{a}(\mathrm{C_i}) + 1) + (\textstyle\sum_{j=1}^{m} C_{ij}\, x_{ij}\cdot) * (\mathrm{a}(\mathrm{C_i}) + 1)\right) \tag{1}$$

Where C_{ij} is the jth instance of ith component, t is the estimated development time, the average time need to perform a test case is τ, and $Ntest_i$ is number of test to be executed, $\mathrm{a}(C_i)$ is the redundancy level of component and indicates the number of redundant components $(\mathrm{a}(\mathrm{C_i}) = 0,1,..,\mathrm{h})$. Equation (2) defines y_i which represents the type of selected component, y_i has two values; 0 means outsourcing, 1 means in-house component. The order of selected component is represented by x_{ij}, x_{ij} has to values; 0 indicates that no element selected from COTS, while 1 indicates that the j th COTS instance of the i th component is selected, see equation (3)

$$y_i = \begin{cases} 1 & \text{if the ith component is in} - \text{house developed} \\ 0 & \text{other wise} \end{cases} \tag{2-a}$$

$$x_{ij} = \begin{cases} 1 & \textit{if the ith COTS instant of the ith component is selected} \\ 0 & \textit{other wise} \end{cases} \quad (2\_b)$$

From above formulas we conclude that, if the ith component has only mi < m COTS components then the x_{ij}'s are defined for $a(C_i) + 1 \leq j \leq m_i$ for each component i, if an instance is bought (i.e. some $x_{ij} = 1$), in this case there is no in-house development (i.e. $y_i = 0$) and if an instance is developed in-house (i.e. $y_i = 1$) and there is no component is bought (i.e. some $x_{ij} = 0$). The following equation represents this constraint as well as the constraint that at most one COTS instance is bought for each component:

$$y_i + \sum_{j=1}^{m} x_{ij} = 1 \ \forall i = 1 \ldots\ldots n \text{ and } a(C_i) = 0, \ldots\ldots, h \quad (3)$$

Constraints:

There are two constraints; delivery time constraint and reliability. For in-house component the delivery time constraint must be less than or equals be the expected or required delivery time. The second constraint is the calculated reliability which must be less than or equals to the predefined reliability,

$$Ti \leq T \quad (4)$$

$$\prod_{i=1}^{n} R_i^{SUB} \geq R \quad (5)$$

For the first constraint: From equation (4): Ti is delivery time, and T is expected or required delivery time. The delivery time is the summation of estimated development time t_i, and the test cases time τ_i which is depend on the executed number of test $Ntest_i$. That can be expressed as follow: $(t_i + \tau_i\, Ntest_i) * (a(C_i) + 1)$. Thus,

$$Ti = y_i(t_i + \tau_i\, Ntest_i) * (a(C_i) + 1) + (a(C_i) + 1) * \sum_{j=1}^{m} d_{ij} x_{ij} \quad (6)$$

Where d_{ij} is average delivery time, y_i and x_{ij} as previously expressed in equations (2a) and (2b), thus the equation can be stated as below:

$\max(Ti) \leq T$ Or

$$T1 \leq T, \ldots, Tn \leq T \quad (7)$$

We can use the parameter (Number of test to be performed) as variable to evaluate option include in-house developed components.

For second constraint (5): The calculated reliability represented in left part, while another part represents the required reliability level. The calculated reliability must not exceed the required reliability. The probability of failure on demand for in-house developed component is $1 - \rho_i$. Thus the average number of failures θ_i of component i is calculated as follows;

$$R_i = e^{-\theta i} \quad (8)$$

$R_i = e^{-\theta i}$ Represents the probability of no failures occurring during the execution of the ith component and it is given in a Poisson distribution with parameter θ_i. Therefore reliability for single component is θ_i. And reliability for sub system is denoted by θ_{i}^{SUB} and expressed as:

$R_{i}^{SUB} = (1-(1-\theta_i))^{s_i}$, where s_i average number of invocations. (9)

The reliability for whole system θ^S is summation of R_{i}^{SUB}

$$R^S = \prod_{i=1}^{n} R_{i}^{SUB} \tag{10}$$

θ_i is calculated based on the total number of performed for in-house developed component N_i^{SUC}. Testability defines the total number of test N_i^{SUC}on one component and denoted by as follows;

$$N_i^{SUC} = (1-\pi_i)\ Ntest_i \qquad \forall i = 1 \dots n \tag{11}$$

Using Bayes's theorem, the probability ρ_i "that the in-house developed instance is failure free during a single run given that N_i^{SUC}test cases have been successfully performed" can be calculated. Assume event A indicates failure-free test casesN_i^{SUC}, while event B denotes the component is failure free during a single run, and π_i is testability. So we have

$$\rho_i = P(A/B) = \frac{P(A/B)P(B)}{P(A/B)P(B) + P(A/\bar{B})P(\bar{B})} \tag{12}$$

$$P(A/B) = 1, \tag{13}$$

$$P(B) = 1 - \pi_i\ , \tag{14}$$

$$P(A/\bar{B}) = (1-\pi_i)^{N_i^{SUC}}, \tag{15}$$

$$P(\bar{B}) = \pi_i\ ,$$

Therefore

$$\rho_i = \frac{1-\pi_i}{(1-\pi_i) + \pi_i (1-\pi_i)^{N_i^{SUC}}} \tag{16}$$

Thus reliability constraint is thus can be given by the equation stated below

$$\theta_i = (1-\rho_i) s_i y_i + \textstyle\sum_{j=1}^{m} \mu_{ij} s_i x_{ij}\ . \tag{17}$$

Where, μ_{ij} the probability of failure on demand, and s_i the average number of invocations, and if we consider the redundant component on each subsystem the equation in (14) becomes;

$$\theta_i = (1-\rho_i) s_i y_i + \textstyle\sum_{j=1}^{m} \mu_{ij} s_i x_{ij} \tag{18}$$

5. System Model

The system composed from predefined number of subsystems. Each subsystem consists of at least one component and additional components might be included. The maximum number of components is $a(C_i) = h$. Either one of the available COTS or in-house component can be selected to minimize the software cost according to the equation (1) under delivery time constraint as stated in (4) and reliability constraints expressed in (5). The required amount of testing on each in-house instance has its impact to the achievement reliability level that satisfies the reliability of the whole system. The expected optimal solution minimize the system's cost while reliability over the threshold R, and system delivery under threshold T. A simple example is provided in this paper to investigate the feasibility of the approach. Next section presents a sensitivity analysis results which performed using a simple example.

6. Simple Example and Sensitivity Analysis

The system we used to demonstrate the approach is consisting of three components. Two sources of components are available, these sources are: three outsourced COTS components are available from vender for each component; and an in-house instance of each component can be built. The particle swarm optimization technique is used to optimize the architecture that satisfies the requirements such as least expensive, lowest probabilities of failure on demand, or sufficient delivery times. We have considered two sets of parameters values for the COTS components and the in-house developed components. Table 1 and Table 2 show these values respectively [1]. All possible scenarios can be explored to study the effect of varying characteristics of components and the threshold of the constraints.

Parameters for COTS and in-house developed components are shown in Table 1 and Table 2 respectively. The second column in Table 1 presents the set of COTS instances available, for each component. Architect can select one of these instances to compose the system. The third column lists the cost for each of them, $\mathbf{C_{ij}}$ units. Then the table lists the average delivery time $\mathbf{d_{ij}}$, average number of invocations, $\mathbf{S_j}$, and probability of failure on demand, $\boldsymbol{\mu_{ij}}$, for each instance in the fourth, fifth, and sixth columns respectively.

Table 1. Preliminary Parameters for COTS Components

component	COTS alternatives	Cost C_{ij}	Average delivery time d_{ij}	Average no. of invocations S_i	Probability of Fail on demand μ_{ij}
C_1	C_{11}	1	4	180	0.0002
	C_{11}	2.5	4	180	0.0002
	C_{11}	2	4	180	0.0004
C_2	C_{21}	2	4	20	0.0002
	C_{22}	3	4	20	0.0002
	C_{23}	6	15	20	0.0004
C_3	C_{31}	10	4	60	0.0002
	C_{32}	14	10	60	0.0002
	C_{33}	10	10	60	0.0004

Table 2 presents the parameters values collected for the in-house components. The average delivery time, testing time $\mathbf{t_i}$, in days, and unitary cost for each component, $\mathbf{C_i}$, unit per day, are depicted in columns 2, 3, and 4 respectively. The fifth column shows the average number of invocations s_j, and the testability π_i for each component presented in the last column.

Table 2. Preliminary Parameters for in-house Developed Components

component	Development time t_i	Testing time τ_i	Unit development cost c_i	Average no. of invocations S_i	Testability π_i
C_1	1	0.05	1	180	0.002
C_2	2	0.05	1	20	0.002
C_3	3	0.05	1	60	0.002

7. Results and Analysis

As it has been shown in [1], we applied the sensitivity analysis and then evaluated its results to validate the proposed approach. Sensitivity analysis is commonly used to predict the impact of difference parameter's values in the result. Using sensitivity analysis helps architects to identify the key architectural parameters values. Less effort is required to identify parameter's values that satisfy the targeted accuracy. In addition to that, sensitivity analysis would help to find out which parameter the system is most sensitive to. To perform a sensitivity analysis in this study, we controlled the values relevant to a threshold on delivery time, reliability, and probability of failure on demand. Furthermore, we changed some figures of COTS components to study the effects on the solutions. The variations of these parameters might produce different solutions. The produced solution could have lower failure on demand, least cost, and higher reliability. On the other hand, the level in which all components can be selected from in-house developed components can be identified. Below are some of the results and analysis, which were obtained as a result from performing a special case of our approach. This particular case is performed setting the maximum number of components in each sub system to one component, either in-house or COTS.

There is nothing to do with probability of failure on demand of a COTS instance since it is a fixed value as it has given from vendor or estimated from the customer. However, probability of failure on demand (π_i) of an in-house component could be optimized. Considering the initial values of testability for in-house developed component, it can be improved by an increase, which is denoted as a decision variable that represents the successful tests. From Table 3, the first record shows the case when we limited time-to-deliver component between (0 and 4), reliability threshold is 0.7 as it can be seen all selected components are COTS components. After that the scope of time-to-deliver has been increased (from 4 to 7), as a result one component from in-house developed component included in the system, this is shown in second record which it shows also the cost has become eight units instead of 13 in the previous case, because the cost of in-house component lessor than COTS. Moreover, the reliability of the system decreased because the in-house component has lower reliability, but the reliability still beyond the threshold.

When the delivery time increased to a specific level, then the developed in-house components could be involved in the composite system instead of COTS, and thus cost could be decreased. In this case the reduction in reliability was unsurprised and fortunately it kept on above the threshold. Consequently, the optimization could be derived towards a solution in which all components are selected from in-house developed components (see Table 3 rows 3 and 4). This is done by first increasing probabilities of failure on demand of all the COTS components. Then the delivery time is increased by one unit each iteration, the new solutions are generated in each iteration until a solution that composed from all in-house developed components is reached and then iteration stopped. For real cases, only the second step can be applied.

Since the values of testability were too low optimal solution with in-house components, no amounts of testing were suggested. The in-house developed components could be involved when the there is enough time to develop them is available. And once there is an in-house developed component is included, testing time is required for this component especially if the required reliability level is high. Sensitivity analysis clearly showed; the increase of probabilities of failure on demand of all the COTS components will produce a solution made of only in-house components. And if reliability and time to development threshold could increase, *as in Table 3, row 4 where reliability increased from 0.8 to 0.9*, and probabilities of failure on demand of the entire COTS instances are also increased, then testing time activities would have shown.

Record 4 and record 5 in Table show that, if system reliability increased (*from 0.8 to 0.9*), then the need for test time increased by more than three time in average and also the cost become double (from 65 to 112 $).

Table 3. Results Sample

No	**T** (time-to-deliver component)	**R** (probability- of-failure-on- test)	**System components**			**Number of test** $Ntest_i$			**System reliability**	**cost**
			Comp1	**Comp2**	**Comp3**	**Comp1**	**Comp2**	**Comp3**		
1	0 to 4 (4)	0.70	COTS11	COTS21	COTS31	0	0	0	0.7	13
2	0 to 7 (7)	0.70	In-house	COTS22	COTS31	0	0	0	0.7	8
3	0 to 15 (15)	0.70	In-house	COTS22	COTS31	278	0	0	0.7	29.9
4	0 to 25 (24)	0.90	In-house	In-house	In-house	264	312	287	0.8	64.75
5	0 to 50 (45)	0.95	In-house	In-house	In-house	880	973	789	0.9	111.8

The cost could be increased once one or more in-house components involved, record 1 and record 2 from the same table show that; the cost decreased by 3 units to obtain similar reliability by substituting an in-house component.

8. Conclusion

This paper has proposed the implementation of optimization technique to solve the reliability redundancy allocation problem (RRAP) in Component-Based Embedded Systems. In addition, the optimal level of redundancy for each subsystem, the appropriate combination of the system from COTS and in-house developed components are considered. PSO is used to search design space for optimal solutions. We applied the approach on a simple numerical case study. The initial result shows the potentiality of the approach. The scenarios for sensitivity analysis were stated and observed. These experiments demonstrated that we can optimize architecture predicting solutions that meet the predefined requirements. Consequently, the architect can perform a tradeoff on the resulted Pareto optimal solutions to take right design demission. We plan to study how well the approach will perform to optimize real system architectures which consist of components from different sources as well as redundancy components. The future works include more experiments on complex and large case studies from embedded system domain and information system. Anti-Lock Brake System (ABS) and Retrieval Information System (RIS) example will be introduced to evaluate the applicability and efficiency of the proposed approach.

References

[1] V. Cortellessa, F. Marinelli and P. Potena, "An optimization framework for "build-or-buy" decisions in software architecture", Comput. Oper. Res., vol. 35, no. 10, **(2008)**, pp. 3090-3106.
[2] S. S. Gokhale, "Cost constrained reliability maximization of software systems", in Reliability and Maintainability, 2004 Annual Symposium - RAMS, **(2004)**.
[3] H.-W. Jung and B. Choi, "Optimization models for quality and cost of modular software systems", European Journal of Operational Research, vol. 112, no. 3, **(1999)**, pp. 613-619.
[4] I. Meedeniya, A. Aleti and B. Zimmerova, "Redundancy Allocation in Automotive Systems using Multi-objective Optimisation", in Symposium on Automotive/Avionics Systems Engineering SAASE, http://www.jacobsschool.ucsd.edu/GordonCenter/g_leadership/l_summer/docs/saase/papers/MeedeniyaAleti Buhnova.pdf, **(2009)**.
[5] V. G. Gudise and G. K. Venayagamoorthy, "Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks", in Proceedings of the Swarm Intelligence Symposium. SIS '03. IEEE, **(2003)**.
[6] D. W. Boeringer and D. H. Werner, "Particle swarm optimization versus genetic algorithms for phased array synthesis", Antennas and Propagation, IEEE Transactions on, vol. 52, no. 3, **(2004)**, pp. 771-779.
[7] R. Eberhart and Y. Shi, "Comparison between genetic algorithms and particle swarm optimization", Evolutionary Programming VII, V. Porto, et al., Editors, **(1998)**, Springer Berlin / Heidelberg, pp. 611-616.
[8] J. J. Liang, et al., "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions", IEEE Transactions on Evolutionary Computation, vol. 10, no. 3, **(2006)**, pp. 281-295.
[9] S. N. Qasem and S. M. Shamsuddin, "Radial basis function network based on time variant multi-objective particle swarm optimization for medical diseases diagnosis", Applied Soft Computing, vol. 11, no. 1, **(2011)**, pp. 1427-1438.
[10] J. Kennedy and R. Eberhart, "Particle swarm optimization", in Proceedings, IEEE International Conference on Neural Networks, **(1995)**.
[11] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm", in Systems, Man, and Cybernetics, IEEE International Conference on Computational Cybernetics and Simulation, **(1997)**.

Authors

Adil Ali Abdelaziz has graduated from University of Khartoum. Currently he is PhD student in Software Engineering Department – Universtiti technologi Malaysia. His research interests are component-Base system, model driven development, quality attributes of CBS, Optimization technique and in specific, Particle Swarm optimization - PSO. He is a member of Sudanese Students Research Group (SSRG) and the Ssoftware Engineering Research Group-UTM

Mr. Wan M.N. Wan Kadir is an Associate Professor in Software Engineering Department – Universtiti technologi Malaysia. He has a PhD. in the field of Software Engineering from the University of Manchester. His research interest covers various Software Engineering knowledge areas based on the motivation to reduce the cost of development and maintenance as well as to improve the quality of large and complex software systems.