International Journal of Database Theory and Application
Vol. 11, No. 4 (2018), pp.41-58
http://dx.doi.org/10.14257/ijdta.2018.11.4.04

ADF: Adaptive Dragonfly Optimization Algorithm Enabled with the TDD Properties for Model Transformation

Pramod P. Jadhav[1] and Shashank D. Joshi[2]

[1]Assistant Professor, Dr. D.Y. Patil School of Engineering, Logegaon, Pune
jadhavpp64@gmail.com
[2]Professor, Bharati Vidyapeeth (Deemed to be University)
College of Engineering, Pune
Pune, Maharashtra, 411043, India.

Abstract

Model Transformation (MT) is one of the leading research problems in software engineering. MT automates the design problem by creating a target model for any source model and thereby, makes the design simpler. For formulating the target model, MT defines a set of rules for the transformation. In this work, the model transformation is achieved through an optimization framework enabled with Test Driven Development (TDD) properties. This work intends to transform the UML class diagram (CLD) to the Relational Schema (RS) model, and the suitable blocks for the transformation are chosen through the proposed Adaptive DragonFly (ADF) algorithm. The proposed ADF algorithm uses an optimal fitness function by including the TDD test cases. Then, optimal blocks obtained from the proposed ADF algorithm is used for achieving the transformation from CLD to RS model. Finally, the experimentation of the proposed scheme is done by constructing an example base with 3 CLD models. Simulation is carried out by choosing different optimization algorithms for analysis and evaluated based on metrics such as AC and fitness measure. The ADF algorithm yielded improved performance than existing models with high values for 0.7962 and 0.8798 for AC and fitness respectively.

***Keywords**: Model transformation, UML class diagram, relational schema, test driven development, dragonfly algorithm*

1. Introduction

The primary task in Model-Driven Development (MDD) [1, 2] is the generation of Models and Model Transformations (MTs), which need the fully-developed environment depending on the leading practices of software engineering concepts. The generated models must be evaluated, synthesized, preserved, and are forced to undergo configuration management to secure their quality. This process makes the MT as the primary task in MDD. MT is used in forwarding engineering, and it focuses on the maintenance of models and produces the code by the transformation mechanism. Consequently, several transformation languages are rising in the literature [3]. Their model transformation has different stages suitable for various purposes. For example, the models are used to detect the missing parts or disputes in the requirements stage. At the design stage, models are used to find the properties of the architectural choices, and at the running stage, they are used to monitor the system using the desired model. In software engineering, both the models and the final artifacts are software, which is the salient feature divergent to conventional engineering. Model transformation helps in the

Received (September 2, 2018), Review Result (November 2, 2018), Accepted (November 8, 2018)

transition from the model to the system through the automation. In some cases, model transformations can be described as software manipulation actions [4, 5].

The model transformation [6, 7, 8] performed under certain conditions may yield different results, and hence, finding the suitable model transformation technique is a challenging task. Thus, the recent works in the field study of the MT introduce various verification techniques for finding the effective standard [4, 5]. Two commonly used methodologies for the MT are 1) TDD [9-12] and 2) Optimization based models [1, 24]. Various literature works [9] have utilized the TDD for the MT through the application of the test cases and the reusable patterns. The literature works [10-12] have utilized the TDD criteria for the MT, but have achieved desired results. Other literature works [1, 24] have used the optimization based approaches for the transformation of the source models to the target model. The target model yielded through the optimization algorithms have less compliance with the Metamodel. The optimization based approaches view the solution as the partial target model obtained through the predefined examples. The MT driven through the examples has better convergence to the Metamodel. One of the prime challenges for the transformation with the example is the creation of the example database suitable for the source model. The manual creation of the example is a tedious process, and also, the availability of the examples for the source is very scarce. Automating the database creation process reduces the manual intervention. The literature work [16] introduced the model transformation as an optimization by examples (MOTOE) approach for the automation of the MT.

In [3], MOTOE has been utilized for transforming the CLD model to RS model. Using the optimization based approaches for transforming the models represented as examples reduced the complexity of the MT process. In MOTOE, an optimization technique is used along with various example bases for identifying optimal constructs and thereby, achieves MT. While using a large number of example base source codes, construct involved in transformation also increases. Hence, it is difficult to identify the suitable block sets for the transformation. This challenge can be avoided by using the heuristic based strategy for the search process. The heuristic based algorithms develop a transformation solution for individual constructs in the source code. Since MOTOE model performed well on transforming a large number of source models, it can be employed in validating the industrial data. Transforming the source model to target model can be done through two strategies, 1) Parallel exploration of different transformation, and 2) initial transformation possibility improvement. The first strategy of model transformation can be done using the global search algorithm, like Particle Swarm Optimization (PSO) [17], while the other strategy requires hybrid search model, like PSO integrated with Simulated Annealing (SA) [18] [3].

The primary intention of this research is to design and develop an optimization aided MT by integrating TDD properties with the ADF algorithm. Here, the problem of model transformation is represented as a search problem [3], which means the transformation of the original model to the target model by defining the constructs randomly. Accordingly, the solution is represented as mapping blocks selected for each construct. Thus, the initial particle population with different possibilities (solutions) transforms the source model by combining blocks from the transformation examples. The finding of the best mapping blocks of every construct is done using the proposed ADF algorithm, which is designed by adapting the weights of the Dragonfly Algorithm (DA) [19]. Then, this work proposes new fitness function with the factors, such as Adequacy, Internal coherence, External coherence, Association of the transformations and TDD property. The proposed fitness derives the properties of the TDD for satisfying the Syntactical Correctness and Semantical Correctness [12] of the target model. Finally, the transformation process depends on the suitable block sets returned by the proposed ADF algorithm.

The contribution of this work towards the field of the model transformation is enlisted as follows:

- The primary contribution is the design of the ADF algorithm by changing the weights of the DA adaptively.
- The secondary contribution is the design of the TDD based fitness function for incorporating the properties of the test cases for the MT.

The rest of this paper is organized as follows: Section 1 introduces the MT and the ways of incorporating the TDD into MT. Section 2 briefs various literature works contributed towards the model transformation, and it also discusses the research gaps in the works. Section 3 provides the basics and the standard definition involved in the MT. Section 4 provides a brief explanation of the proposed ADF algorithm driven with the properties of the TDD for the MT. Section 5 provides the simulation results of the proposed ADF algorithm while applied to the various standard datasets. Section 6 concludes the paperwork.

2. Motivation

2.1. Literature Review

Literature presents the different methods employed for MT. Here, various related works in the field of model transformation are described, and the advantages and disadvantages of each work are presented. From the literature, the model transformation is classified as, a) MT by optimization, b) MT by TDD, and c) MT by other methods.

a) MT by optimization: Marouane Kessentini *et al.* [3] have presented the MT based on PSO algorithm. The main advantage of this method is that it produces the transformation of a source model, even if the rule induction is not possible or is arduous to perform. For large models, it takes more time for performing the transformation, which is the drawback of this method. Marouane Kessentini *et al.* [13] have developed the model transformation based on heuristic search. This method provides various colors for traceability links, which help the tester to identify the error source. The drawback of this method is that it is very tricky to fetch the transformation examples and the performance of the system is based on these transformation examples. J.W. Ko *et al.* [24] have presented the model transformation based on the PSO algorithm and the graph transformation. The advantage of this method is that the establishment of the algorithm for testing the model transformation is very simple. The graph transformation and the optimization algorithm take more time for performing the computation.

b) MT by TDD: J. Ko and Y. Song [9] have introduced the model transformation by the test driven development by means of reusable patterns. The test creation process of this method is very fast, and it was based on the interaction among the users and the software systems. The test scripts generated by this method were not admissible by the test automation tool. P. Giner and V. Pelechano [10] have suggested the model-to-model transformation by the test-driven method. In this method, the effects of changes occurred in the Metamodel affect the tests only, so that the maintainability was enhanced. This method did not consider the test case quality, which is the drawback of this approach. J. Steel and M. Lawley [11] have proposed the Model-based test-driven approach. The advantages of this method are that it gives the detailed definition and development of test suites. This method had limitations, like version skew. Timo Kehrer and Sven Wenzel [12] have suggested the MT by the TDD method. The benefit of this method is that it utilizes the models, which were taken from real scenarios. The method used models from real scenarios. This method was troubled by the dull design of reference models and the complex target/actual-comparison.

c) MT by Other methods: Andrea Ciancone *et al.* [5] have introduced the model transformation by testing operational transformations. The unit testing performed by this method was simple and fast. The drawback is it needs a wide validation of usability. Zoltán Balogh and Dániel Varró [14] have suggested the MT based on inductive logic programming. This method uses the Aleph for deriving the transformation rules of the object-relational mapping. This method had the practical and conceptual limitations. Esther Guerra and Mathias Soeken [15] have introduced the Specification-driven model transformation testing. This method has various advantages like the models are valuable even if there is a minimum exhaustive coverage, non-deliberate errors are identified, and so on. The disadvantage of this method is that it cannot detect the extra classes in the input models.

2.2 Research Gaps

The various research gaps presented in the literature works for the successful model transformation are briefed in this section. Many works tend to create a manual database for the transformation process [15]. Creation of the manual database for the transformation process requires more time and besides it concentrates on the single issue. Otherwise, the manual database created for the training is suitable for the specified applications, and thus, it ignores maximum test cases. Some works tend to create a more manual database to avoid the above mentioned problems. The real challenge arises during the design of the evaluation function for selecting the suitable models for the transformation. Thus, an increase in the problem size makes the design of the evaluation/objective function to be a tedious process.

The literature works [20, 21] used the heuristic search strategy to find the suitable block sets for the transformation. Finding the correctness of the target model from the optimization problem requires a standard reference model [23]. Many methodologies have not defined a standard reference model to check the compliance of the target model. The optimization problem applied to the complex environment may suffer from the candidate transformation errors. Rectification of these errors requires more resource and time [22]. Some works [13, 22] have included the target Metamodel for finding the correctness of the transformation. The testing of the transformation process should also find the error causes in the model. The works [1, 3, 24] have used the PSO algorithm for the MT. The PSO based MT suffers from local convergence. Hence improved search strategy need to be utilized for the MT.

3. Model Transformation

This section introduces the basics of the MT and the various terms involved in the MT. MT defines the creation or transformation of one type of database to another database. The software developers require a different database based on their application. The transformation of one model to another reduces the time for the software development. Transformation of one database to another should not alter the basic information in the database. Some of the basic definitions involving in the model transformation are defined as follows,

Model to transform: The model to transform defines the source model subjected to the transformation. In this work, the transformation of the Unified modeling language (UML) database to the RS database is performed. Hence, the UML database is considered as the model to transform, and the RS database defines the target model. The model contains various constructs, with various classes, association, and aggregation. The constructs of the models have predefined syntax.

Model constructs: Model constructs define each element present in the source and the target model. The various constructs in the UML model are classes, associations, and generalizations. For the CLD database, the class acts as a construct. Many databases contain complexly structured constructs, in which each construct has its subconstructs. The constructs in the model define the properties of the class.

Block: The block in the MT defines the required transformation trace to change one model to another. To perform the transformation, the block must be defined for each construct involving in the transformation. The block is formed by combining the constructs that can be transformed together. The construct C_1 and C_2 can be mapped together to form a block if it posses the similar properties. The blocks in the MGT are represented as a concrete model.

4. Proposed Methodology: Integration of the TDD with the Proposed ADF Algorithm for the Model Transformation

Figure 1 shows the block diagram of the proposed TDD based model transformation model. This paper aims to transform the source model represented as UML class diagrams (CLD) into RS model. The proposed MT technique achieves the transformation from base model to target model by using MOTOE technique as implemented in [3]. For transforming the CLD model to RS model by optimization based approaches, this model additionally includes the TDD properties for block selection. As shown in figure 1, the proposed MT technique has two phases, 1) training phase to select the optimal blocks, and 2) testing phase to perform the transformation based on the selected optimal blocks.

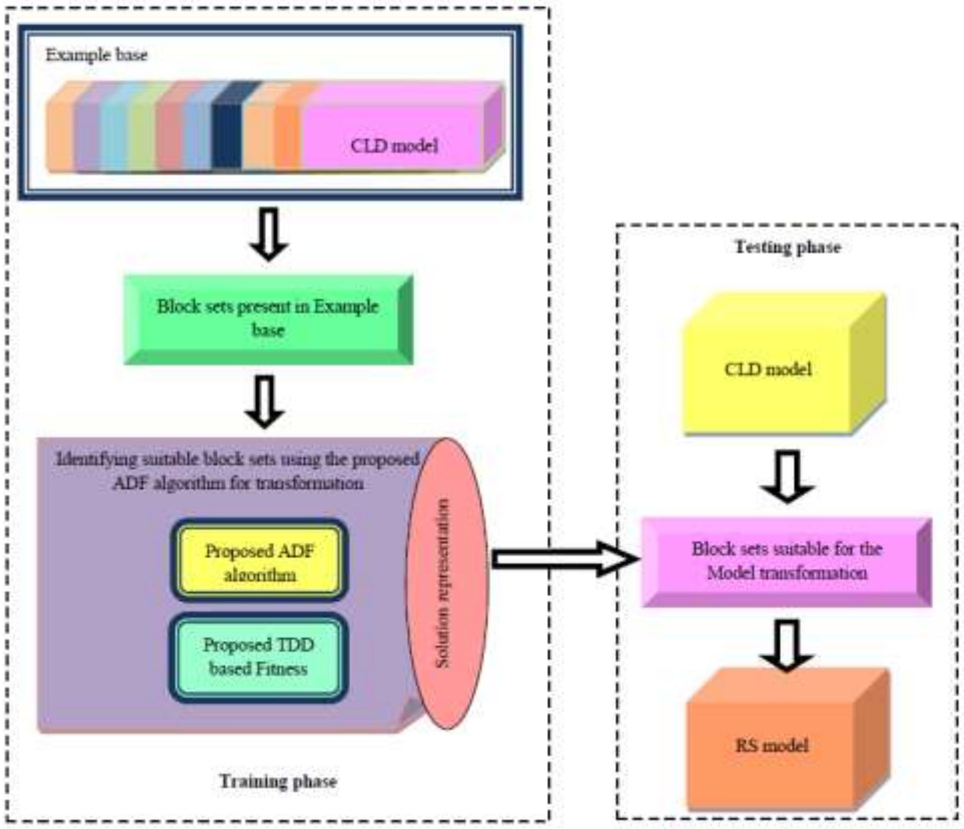

Example base
CLD model
Block sets present in Example base
Identifying suitable block sets using the proposed ADF algorithm for transformation
Proposed ADF algorithm
Proposed TDD based Fitness
Solution representation
Training phase
Testing phase
CLD model
Block sets suitable for the Model transformation
RS model

Figure 1. The Architecture of the Proposed TDD based Model Transformation

In the training phase, the proposed MT model uses 'A' CLD models collected to form an example base. The CLD model constitutes different constructs under a different class, aggregation, association, and generalization. Then, the constructs present in CLD is manually split into different blocks. Each block set is given as the data to the proposed ADF algorithm. For achieving transformation, it is necessary to select the optimal blocks from a set of blocks obtained from the example base. The proposed ADF algorithm chooses the optimal blocks among 'X' blocks available in example base. For evaluating the fitness of the proposed ADF algorithm, this paper uses novel TDD based fitness measure. After the training, a CLD model is provided to the proposed MT model.

In the testing phase, based on the optimal block sets obtained from the proposed ADF algorithm, the transformation of CLD to the RS model is done. The blocks sets are chosen based on class information of CLD.

4.1. Building the Block sets for the model transformation

This work achieves MT based on MOTOE technique as represented in [3]. Here, the CLD model is transformed into a RS model by defining a new optimization scheme along with TDD test cases. For the transformation process, an example base and source model is provided as input to the MOTOE framework. Transformation of CLD model to RS model can be represented as $\{EB, CLD_z, RS_z\}$. While transforming the source model to target model, it is necessary to provide the source model and the example base to the MOTOE system. The output from MOTOE system is the target model.

The initial step in the proposed model transformation scheme is building the necessary block sets from example base. This work aims to transform the CLD models present in example base to its equivalent RS. The primary step in the transformation process is building the block sets from the example base. The example base contains the CLD models with different constructs. The CLD in example base is represented as $\{CLD_1, CLD_2, \ldots, CLD_z, \ldots, CLD_A\}$. The CLD has different cases and thus, has varying constructs. The constructs in CLD_z are represented as $\{C_1, C_2, \ldots, C_c, \ldots, C_P\}$. Block sets can be formed by combining the related constructs, which can be transformed in together. The block sets computed from example base EB can be referred as follows, $\{B_1, B_2, \ldots, B_b, \ldots, B_X\}$. From the example base, a total of X blocks sets are found.

Figure 2 presents the example of CLD model in the example base. The CLD model constitutes four classes, two associations, and one aggregation, i.e., a total of 7 constructs. To transform the CLD model to its equivalent RS model as depicted in figure 3, the proposed MT technique need to identify seven optimal blocks among X block sets from example base. Here, the selection of optimal blocks is done as an optimization problem and is solved with the proposed ADF algorithm.

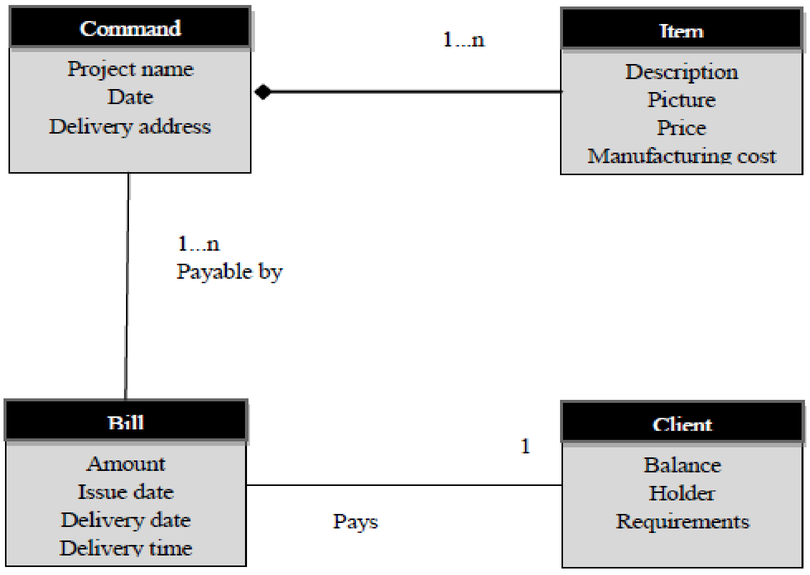


Figure 2. Example of the CLD

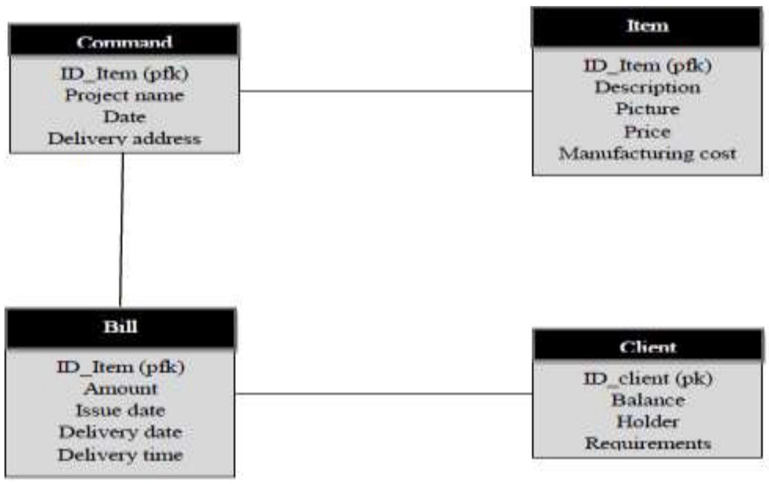


Figure 3. Equivalent RS Model for the CLD Model Presented in Figure 2

4.2. ADF Agorithm: Finding the Suitable Block Set for the Model Transformation

The block sets produced in the previous step from example base vary between 1 and X. Only certain blocks from the block sets are suitable for the MT. This paper proposes a novel heuristic-based search strategy, ADF algorithm, along with the test cases to find the optimal blocks suitable for the transformation. The proposed ADF algorithm finds its basics from the existing DA [19]. DA provides better convergence in the larger search space. The proposed ADF algorithm adaptively changes the weight of the parameters used in the DA.

4.2.1. Solution Encoding

Figure 4 provides the solution encoding of the proposed ADF algorithm. The proposed ADF algorithm aims at finding the suitable blocks from the block sets obtained from the previous section. The CLD model considered for the transformation, represented as CLD_z, contains seven constructs. Also, the example base contains a total of A CLD models with X number of blocks. The proposed ADF algorithm can choose equivalent optimal blocks for the seven constructs of CLD. The block sets from example base contain a total of X blocks. The proposed ADF algorithm tries to find the seven optimal blocks from the X blocks. The individual blocks in the block sets are represented as B_b. The solution encoding for the proposed ADF algorithm for CLD_z with seven constructs has the size of 1×7. Each solution takes the block number between 1 and X.

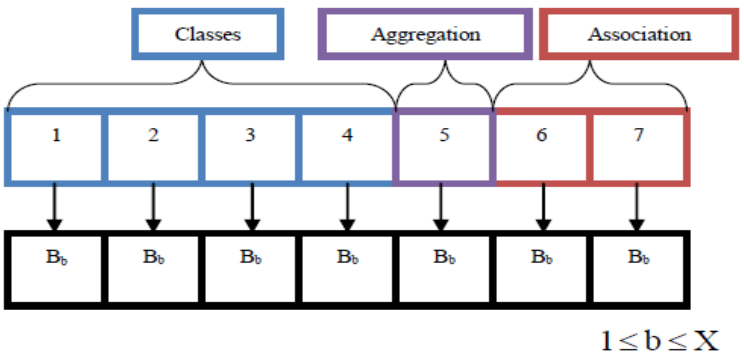


Figure 4. Solution Encoding for the ADF Algorithm

As shown in the above figure, the CLD model considered for transformation CLD_z has seven constructs, with four classes, one aggregation class, and two association class. The proposed ADF algorithm tries to find the seven optimal matching blocks for each construct of CLD among X block sets.

4.2.2 Proposed TDD Property based Fitness Function

This work proposes the TDD based fitness function to find the optimal block value for the transformation. The fitness utilized in work [3] depends on the factors, such as Adequacy, Internal coherence, External coherence, and Association of the transformations. The proposed fitness function includes the TDD property along with these parameters for defining the fitness of the selected optimal block by the ADF algorithm. The test cases required for the transformation of the UML database to its equivalent model is utilized from work [12]. The proposed fitness function based on the TDD property is expressed in the following expression,

$$Fitness = F_1 + F_2 \tag{1}$$

where, the fitness terms F_1 and F_2 are represented by the following expression,

$$F_1 = \sum_{c=1}^{7} a_c * (I_c + E_c) \tag{2}$$

where, a_c represents the adequacy factor, I_c represents the internal coherence, and E_c represents the external coherence factor for the c^{th} construct. The fitness regarding the TDD is represented as follows,

$$F_2 = \frac{\text{Number of satisfied test cases}}{\text{Total number of test cases}} \tag{3}$$

The proposed fitness function uses the test cases from work [12] to satisfy the following factors,

- Syntactical correctness
- Semantical correctness

The syntactical correctness verifies the fact whether the target RS model conforms to the Metamodel. Semantical correctness verifies the fact whether the target RS model satisfies the user's requirement.

4.2.3 Algorithmic Steps of the Proposed ADF Algorithm

For finding the optimal blocks suitable for transformation, this work introduces the ADF algorithm by modifying the DA. DA finds the optimal solution based on heuristic properties of the dragonfly. Accordingly, to DA [19], the dragonfly alters its position from one place to another with varying velocity based on five factors. The factors altering the position of the dragonfly are 1) separation, 2) alignment, 3) Cohesion, 4) attraction towards the food source, and 5) distraction from the enemy. These factors influence the survival of the dragonfly. Existing DA uses weight parameters between [0, 1] for each factor and thereby, improve the convergence. Since DA is inspired by natural behavior; the movement of dragonfly need not be the same during iteration. Hence, to capture the adaptive behavior of a dragonfly at varying iteration, it is necessary to adjust the weights regarding the iteration count. In the proposed ADF algorithm, adaptive weights are employed for calculating the velocity and position update. The adaptive weights are calculated based on the current iteration count and total iteration.

The algorithmic steps involved in the proposed ADF algorithm are briefed below:

i) Initialization

The initial step in the proposed ADF algorithm is random assignment of blocks to the solution space. Each element in solution can be referred with the dragonfly, and the characteristics of dragonfly influence the solution update. The proposed ADF algorithm is set to identify the optimal blocks for the transformation, and thus, the solution space is represented as follows,

$$Z = \{Z_1, Z_2, \ldots, Z_i, \ldots, Z_7\} \tag{4}$$

where, Z_i indicates the position of i^{th} dragonfly in the solution.

ii) Calculate the fitness of each solution in the population

The randomly initialized blocks in the initial step are subjected to the fitness evaluation. The quality of transformation from CLD to RS model can be made significant if the blocks selected among X blocks represent each construct of CLD_z. For this process, the fitness of the solution is evaluated based on equation (1). The solutions satisfying the fitness derived in (1) is further taken for evaluation.

iii) Update the velocity of the dragonfly with the adaptive weight

Movement of dragonflies from one place to another depends on the velocity of the dragonfly. The velocity factor acts as the step vector for a position update. Thus, the velocity of the dragonfly depends on factors, such as separation, alignment, Cohesion, attraction towards the food source, and distraction from the enemy. Now, the velocity update can be expressed as,

$$\Delta Z(t+1) = (k.K_i + l.L_i + m.M_i + n.N_i + o.O_i) + \alpha_2.\Delta Z(t) \tag{5}$$

where, k, l, m, n and o refer to the weight for adjusting separation, alignment, cohesion, attraction, and distraction, respectively. Also, K_i, L_i, M_i, N_i and O_i indicate the separation, alignment, cohesion, attraction, and distraction of the i^{th} dragonfly influencing the velocity update. The term α_2 indicates the weight for adjusting the step vector at the current iteration. In the proposed ADF algorithm, the weights influencing the velocity update are adaptively chosen for each iteration, and thus, initially an equal value α_1 is chosen, i.e.) $k =$ $l =$ $m =$ $n =$ $o =$ α_1. For each iteration, the value of α_1 is adaptively chosen, and thus, the velocity update equation of proposed ADF algorithm is modified as follows,

$$\Delta Z(t+1) = \alpha_1(K_i + L_i + M_i + N_i + O_i) + \alpha_2.\Delta Z(t) \tag{6}$$

where, α_1 and α_2 refers to the adaptive weight parameters, which depend on iteration count. The expressions for the adaptive weights are formulated as follows,

$$\alpha_2 = 1 - \alpha_1 \tag{7}$$

$$\alpha_1 = \frac{t}{T_{max}} \tag{8}$$

where, t indicates the current iteration, and T_{max} refers to the maximum iteration count. The expression of factors influencing the velocity of the dragonfly is expressed in the following equations,

$$\text{Seperation,} \quad K_i = -\sum_{j=1}^{7} Z - Z_j \tag{9}$$

where, Z_j refers to the position of j^{th} neighbour dragonfly.

$$\text{Alignment,} \quad L_i = \frac{\sum_{j=1}^{7} v_j}{7} \tag{10}$$

where, v_j indicates the velocity of j^{th} neighbour dragonfly in the solution.

$$\text{Cohesion,} \quad M_i = \frac{\sum_{j=1}^{7} Z_j}{7} - Z \tag{11}$$

$$\text{Attraction,} \quad N_i = Z^+ - Z \tag{12}$$

$$\text{Distraction,} \quad O_i = Z^- - Z \tag{13}$$

where, Z^+ and Z^- indicate the attraction, and distraction caused due to the presence of food source and enemy.

iv) Update the position of the dragonfly

Now, the velocity update achieved by the proposed ADF algorithm influences the position of dragonfly and thus, the position update of the ADF algorithm is given by below expression.

$$Z(t+1) = Z(t) + \Delta Z(t+1) \tag{14}$$

Substituting the velocity update of proposed ADF algorithm in above equation yields the final position update equation for proposed ADF algorithm, and it is expressed as,

$$Z(t+1) = Z(t) + \alpha_1 (K_i + L_i + M_i + N_i + O_i) + \alpha_2 . \Delta Z(t) \tag{15}$$

v) Update the position of the dragonfly based on levy flight movement

Besides several factors, the dragonfly also moves from one place to another based on Levy flight movement, and it is expressed as,

$$Z(t+1) = Z(t) + \mathrm{Levy}(r) \times Z(t) \tag{16}$$

where, $\mathrm{Levy}(r)$ indicates the levy flight with dimension r.

vi) Termination

Finally, at the end of the iteration, T_{max}, the best solution (satisfying the fitness criteria) is retained. The final solution provides the suitable seven block sets for transforming the CLD to its equivalent RS model, i.e. CLD_z to RS_z.

Algorithm 1 indicates the pseudo code of the proposed ADF algorithm for identifying the optimal blocks for transforming CLD to RS model.

Algorithm 1. The Pseudo Code of the Proposed ADF Algorithm for MT

Sl. no	Proposed ADF algorithm
1	**Inputs:** Block sets, Population size
2	**Outputs:** Suitable block sets for the model transformation
3	**Begin**
4	**Initialize** the block sets as the population of the ADF algorithm
5	**Initialize** the step vectors of the Z
6	**If** ($t < T_{max}$)
7	**For** ($t=1$; $t < T_{max}$)
8	Find the TDD based fitness value of the block sets defined in the population
9	Find the adaptive weight for each iteration
10	**If** (solution i has neighbor)
11	Update the velocity of the solution using the (6)
12	Update the position of the solution using the (15)
13	**Else**
14	Update the position of the solution using the (16)
15	**End if**
16	**End for**
17	**End if**
18	**If** ($t = T_{max}$)
19	**Return** suitable block sets
20	**End if**
21	**End**

5. Results and Discussion

This section presents the simulation results achieved by TDD property and optimization driven MT model. The performance of the proposed ADF algorithm is evaluated by considering different UML class diagrams, and evaluated based on fitness and automatic correctness metrics.

5.1 Experimental setup

The proposed ADF algorithm with TDD properties for model transformation is implemented in the JAVA platform, and PC used for the simulation purpose has the configuration of Windows 10 OS, 4 GB RAM, and Intel I3 processor.

5.1.1 Database description

For the experimentation purpose, an example base with 3 CLD model is constructed. The simulation criterion is to train the example database and transform the individual CLD respectively. The example base for simulation is manually constructed, and it contains different CLD with a different number of classes, aggregation, association, attributes, and generalization. Table 1 presents the information about the three CLD models used for the experimentation.

Table 1. Class Information about CLD Models

Name of CLD model	No. of classes	No. of generalization	No. of association
CLD 1	5	2	2
CLD 2	7	2	4
CLD 3	6	2	2

After training the example base, a total of $X = 6$ blocks are obtained. Thus, for transforming the CLD model to the RS model, it is necessary to identify the suitable blocks among X blocks from example base. Since the class variable for CLD differs from one other, it is necessary to set the population size of the ADF algorithm differently during transformation.

Parameter setting for transforming CLD 1: The CLD 1 model has five classes, two generalizations, and two association, and size of the population Q of ADF for CLD 1 is set as 9.

Parameter setting for transforming CLD 2: The CLD 2 model has a total of 13 constructs, and hence, the size of the population of ADF while transforming CLD is chosen to be 13.

Parameter setting for transforming CLD 3: The CLD 3 model has a total of 10 constructs, and thus, the population size of ADF for CLD 3 is set as 10.

5.1.2 Evaluation Metrics

The performance of the proposed ADF algorithm for finding the optimal blocks suitable in MT is verified using metrics, such as fitness measure, and Automatic Correctness (AC). In this work, the expression for the fitness measure is defined as the maximization function and thus, the model with high fitness is considered to be a better model. The definition of automatic correctness is defined below:

Automatic Correctness: AC measure is obtained by comparing the class variables present in the transformed RS model to the already known RS model of the source. For measuring the RS value, the comparison is made by evaluating each construct. The value of AC varies between 0 and 1, and a high value indicates improved performance.

5.1.3 Comparative techniques

For evaluating the performance of the proposed ADF algorithm for MT, several optimization algorithms have been utilized for the comparative analysis. The existing works used for the comparative analysis are briefed below:

Simulated annealing [3]: In the work [3], Simulated Annealing (SA) has been utilized along with the PSO algorithm for achieving the model transformation.

PSO [3]: In [3], PSO has been used along with SA to build a new optimization namely PSO-SA for model transformation. It has not used TDD properties for model transformation and thereby, achieved reduced performance.

DA [19]: DA in [19] is applied in the proposed method instead of the ADF algorithm for the optimal selection of blocks and thereby, achieved model transformation.

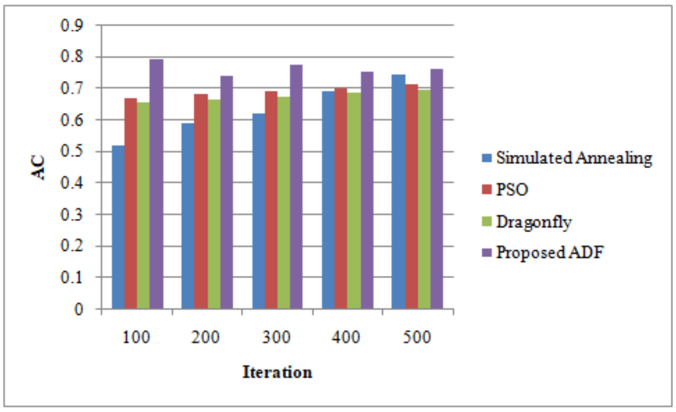
5.2 Comparative Analysis

Here, the comparative analysis of the proposed ADF algorithm against various existing models is analyzed while transforming the CLD model to its equivalent RS model. Evaluation is done by varying the total iteration count T_{max} and evaluated based on metrics, such as fitness and AC. For improved performance, it is necessary to achieve high values for AC and fitness measure, respectively.

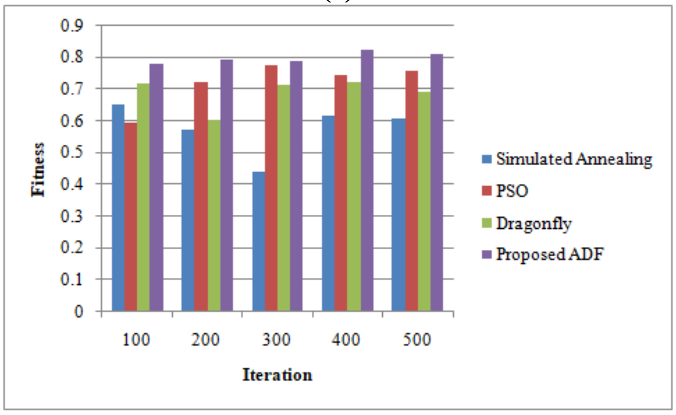
5.2.1 Comparative Analysis for CLD 1

Figure 5 presents the comparative analysis of techniques while transforming the CLD 1 to its equivalent RS 1. Figure 5.a evaluates the performance of comparative models based on AC metric, while CLD 1 is provided as test input. For maximum iteration T_{max}=100, the existing SA, PSO, and DF algorithm achieved AC value of 0.519155, 0.672651, and 0.657059, respectively. The performance of the proposed ADF algorithm is better than the existing algorithms, by achieving a high AC value of 0.795121. Similarly, while T_{max} =500, the existing SA, PSO, and DA (represented as Dragonfly in the graph) achieved AC value of 0.74476, 0.712651, and 0.697059, respectively, but the proposed ADF algorithm has high AC value of 0.763989.

Figure 5.b presents the performance of comparative models based on fitness measure while transforming the CLD 1 model. For maximum iteration T_{max}=100, the existing SA, PSO, and Dragonfly have achieved low fitness measure of 0.697059, 0.596229, and 0.720346 respectively, while the proposed ADF algorithm has a high fitness value of 0.779866. Increasing the iteration limit to 400 increased the performance of comparative models. At this iteration, the proposed ADF algorithm has an overall high fitness measure of 0.827051 overcoming other existing models.



(a)



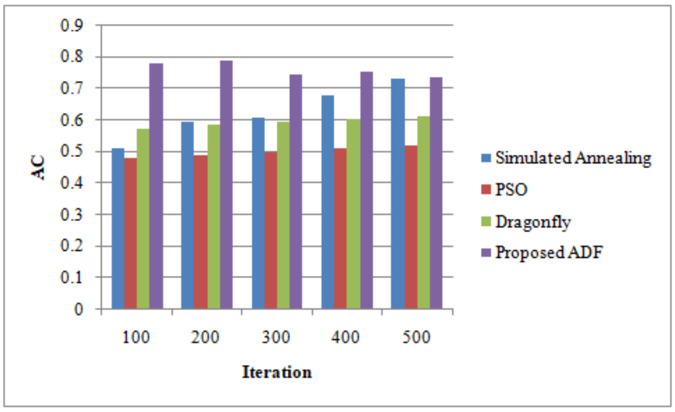
(b)

Figure 5. Comparative Analysis of Models while Transforming CLD 1 based on (a) AC, and (b) Fitness Measure

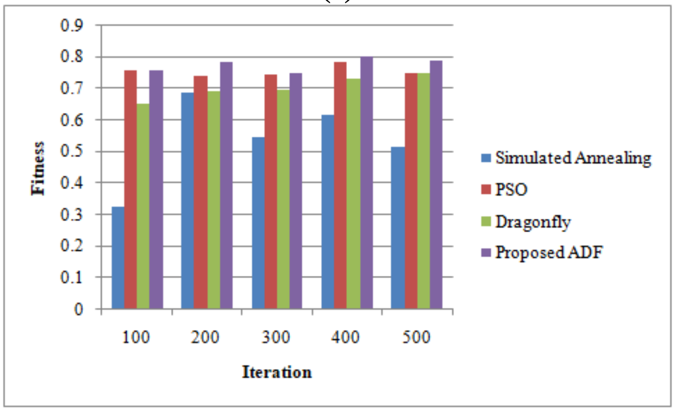
5.2.2 Comparative Analysis for CLD 2

Figure 6 presents the comparative analysis of models while transforming the CLD 2 to its equivalent RS 2 model. Figure 6.a evaluates the performance of comparative models based on AC metric while CLD 2 is provided as test input. For maximum iteration T_{max} =100, the existing SA, PSO, and Dragonfly achieved AC value of 0.509678, 0.47988, 0.575217 respectively. The performance of the proposed ADF algorithm is better than existing algorithms, and it has achieved high AC value of 0.780298. Similarly, while T_{max}=200, the existing SA, PSO, and Dragonfly achieved AC value of 0.595203, 0.48988, and 0.585217 respectively, but the proposed ADF algorithm has high AC value of 0.789479.

Figure 6.b presents the performance of comparative models based on fitness measure while transforming the CLD 2 model. At iteration T_{max}=400, the existing SA, PSO, and Dragonfly have achieved low fitness measure of 0.616834, 0.786885, and 0.733107 respectively, while the proposed ADF algorithm has a high fitness value of 0.803559. At the iteration T_{max} of 300, the performance of comparative models is poor. Here, the existing SA, PSO, and Dragonfly achieved fitness value of 0.545825, 0.746384, and 0.698969 respectively, while the proposed ADF algorithm has a value of 0.7481.



(a)



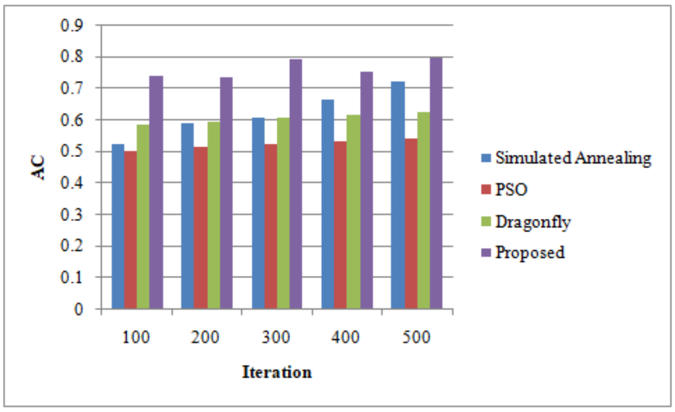
(b)

Figure 6. Comparative Analysis of Models while Transforming CLD 2 based on (a) AC, and (b) Fitness Measure

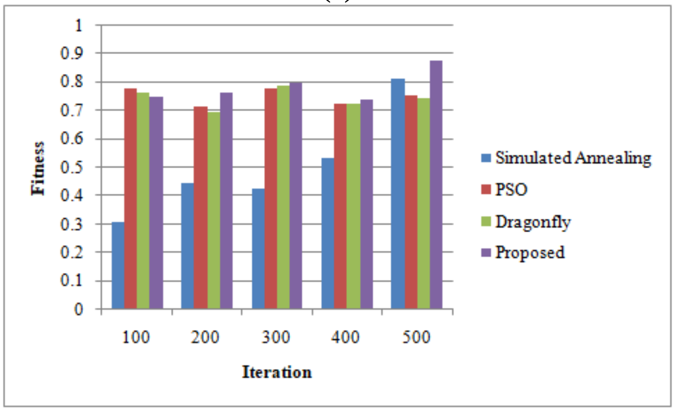
5.2.3 Comparative Analysis for CLD 3

Figure 7 presents the comparative analysis of models while transforming the CLD 3 to its equivalent RS 3 model. Figure 7.a evaluates the performance of comparative models based on AC metric while CLD 3 is provided as test input. For maximum iteration T_{max} =100, the existing SA, PSO, and Dragonfly achieved AC value of 0.523753, 0.503976, and 0.58746479 respectively. The performance of the proposed ADF algorithm is better than existing algorithms, and it has achieved high AC value of 0.741164. Similarly, while T_{max}=500, the existing SA, PSO, and Dragonfly achieved AC value of 0.721834, 0.543976, and 0.62746479 respectively, but the proposed ADF algorithm has high AC value of 0.799995.

Figure 7.b presents the performance of comparative models based on fitness measure while transforming the CLD 3 model. For maximum iteration T_{max}=100, the existing SA, PSO, and Dragonfly have achieved low fitness measure of 0.308858, 0.780605, and 0.765173 respectively, while the proposed ADF algorithm has a high fitness value of 0.749576. Increasing the iteration limit to 500 increased the performance of comparative models. At this iteration, the proposed ADF algorithm has an overall high fitness measure of 0.879874 overcoming other existing SA, PSO, and Dragonfly, which have the low fitness of 0.812074, 0.756204, and 0.746557, respectively.



(a)



(b)

Figure 7. Comparative Analysis of Models while Transforming CLD 3 based on (a) AC, and (b) Fitness Measure

5.3 Comparative Discussion

Table 2 presents the comparative discussion of comparative models against the existing models. For the experimentation, 3 CLD models were utilized and transformed into their equivalent RS model. After training the example base with 3 CLD, the CLD is individually tested. Comparative discussion table depicted below evaluates the transformation quality regarding AC and fitness measure while transforming individual CLDs. As the simulation is done by iteration count of comparative algorithms, table 2 constitutes the best values while transforming individual CLDs. While performing MT for CLD 1, the proposed ADF algorithm achieved AC and fitness values of 0.7951 and 0.8270, respectively. For CLD 2 model, the proposed ADF algorithm achieved AC and fitness measure of 0.7894, and 0.8035. While transforming the CLD 3 model to its equivalent RS, the performance of ADF was better with the values of 0.7962 and 0.8798 for AC and fitness, respectively.

Table 2. Comparative Discussion

Input	Evaluation metrics	Comparative techniques			
		SA	PSO	DF	Proposed ADF
CLD 1	AC	0.5191	0.6726	0.6570	**0.7951**
	Fitness	0.616	0.7447	0.7227	**0.8270**
CLD 2	AC	0.5952	0.4898	0.5852	**0.7894**
	Fitness	0.6168	0.7868	0.7331	**0.8035**
CLD 3	AC	0.6092	0.5239	0.6074	**0.7962**
	Fitness	0.81207	0.7562	0.7465	**0.8798**

6. Conclusion

This paper introduces the optimization driven MT framework for transforming the CLD model to its equivalent RS model. The proposed MT framework introduces an optimization algorithm, namely ADF, by modifying existing DF algorithm. Initially, a group of CLD models is grouped for training and forms an example base. Since the proposed model transformation scheme is driven by example, and optimization, the block sets in each CLD models, are found. Then, based on the proposed ADF algorithm finds optimal block sets from example base. The fitness of ADF solution includes TDD test cases. Finally, the CLD model gets transformed into a RS model based on optimal blocks selected by the proposed ADF algorithm. Experimentation of MT scheme with ADF algorithm is evaluated by considering example base with 3 CLD models. Several existing heuristic models, such as SA, PSO, and DA are used for comparative analysis, and the performance of models is evaluated based on AC and fitness measure. From the analysis, it is evident that the proposed MT model with the ADF algorithm achieved improved performance with the values of 0.7962 and 0.8798 for AC and fitness respectively.

References

[1] R. France, B. Rumpe, "Model-driven development of complex software: a research roadmap," in Proceedings of International Conference on Software Engineering (ICSE), **(2007)**, pp.37-54.

[2] A. Kleppe, J. Warmer, W. Bast, "MDA Explained: The Model Driven Architecture: Practice and Promise," Addison-Wesley, **(2003)**.

[3] Marouane Kessentini, Houari Sahraoui, Mounir Boukadoum, and Omar Ben Omar, "Search-based model transformation by example," Software & Systems Modeling, vol. 11, no. 2, **(2012)**, pp 209–226,.

[4] M.J. Harrold, "Testing: a roadmap," Proceedings of the Conference on the Future of Software Engineering, **(2000)**, pp. 61-72.

[5] Andrea Ciancone, Antonio Filieri, and Raffaela Mirandola, "Testing operational transformations in model-driven engineering," Innovations in Systems and Software Engineering, vol. 10, no. 1, **(2014)**, pp 19–32.

[6] Tomasz Straszak and Michał Smiałek, "Model-Driven Acceptance Test Automation Based on Use Cases," Computer Science and Information Systems, vol. 12, no. 2, **(2015)**, pp.707–728.

[7] Gürkan Alpaslan and Oya Kalıpsız, "Model Driven Web Application Development with Agile Practices," International Journal of Software Engineering & Applications (IJSEA), vol.7, no.5, **(2016)**, pp. 1-11,

[8] Mina Boström Nakićenović, "An Agile Driven Architecture Modernization to a Model Driven Development Solution," International Journal on Advances in Software, vol. 5, no. 3 & 4, **(2012)**, pp. 308-322.

[9] J. Ko and Y. Song, "Test Driven Development of Model Transformation with Reusable Patterns," Lecture Notes in Electrical Engineering, vol. 114, **(2012)**.

[10] P. Giner and V. Pelechano, "Test-Driven Development of Model Transformations," Lecture Notes in Computer Science, vol. 5795, **(2009)**.

[11] J. Steel and M. Lawley, "Model-based test driven development of the Tefkat model-transformation engine," Proceedings of 15th International Symposium on Software Reliability Engineering, **(2004)**, pp. 151-160.
[12] T. Kehrer and S. Wenzel, "Test-Driven Development of Model Transformations," In Proceedings of the 7th Nordic Workshop on Model-Driven Engineering (NW-MODE'09). **(2009)**.
[13] M. Kessentini, H.Sahraoui, and M. Boukadoum, "Example-based model-transformation testing," Automated Software Engineering, vol. 18, no. 2, **(2011)**, pp. 199–224.
[14] Z. Balogh and D. Varró, "Model transformation by example using inductive logic programming," Software & Systems Modeling, vol. 8, no. 3, **(2009)**, pp. 347–364.
[15] Esther Guerra and Mathias Soeken, "Specification-driven model transformation testing," Software & Systems Modeling, vol. 14, no. 2, **(2015)**, pp. 623–644.
[16] M. Kessentini, H. Sahraoui, M. Boukadoum, "Model transformation as an optimization problem," in Proceedings of International Conference on Model Driven Engineering Languages and Systems, vol. 5301, **(2008)**, pp 159-173.
[17] J. Kennedy, R.C. Eberhart, "Particle swarm optimization," in Proceedings of IEEE International Conference on Neural Networks, **(1995)**, pp. 1942–1948.
[18] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, "Optimization by simulated annealing," Sciences, vol. 220, no. 4598, **(1983)**, pp. 671–680.
[19] S. Mirjalili, "Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems," Neural Computing and Applications, vol. 27, no. 4, **(2016)**, pp. 1053–1073.
[20] J.M. Mottu, B. Baudry, Y.L. Traon, "Model transformation testing: Oracle issue," in Proceedings of IEEE International Conference on Software Testing Verification and Validation Workshop, **(2008)**.
[21] Y. Lin, J.Zhang, J.Gray, "A testing framework for model transformations," Model-Driven Software Development, **(2005)**, pp. 219-236.
[22] B. Baudry, T. Dinh-Trong, J.M. Mottu, D. Simmonds, R.France, S. Ghosh, F. Fleurey, Y.L. Traon, "Model transformation testing challenges," In Proceedings of the IMDDMDT workshop at ECMDA, **(2006)**.
[23] J. Kuster and M. Abd-El-Razik, "Validation of model transformations—first experiences using a white box approach," Proceedings of 3rd International Workshop on Model Development, Validation and Verification, **(2006)**.
[24] J.W. Ko, S.J.Beak, and Y.J. Song, "Model Optimization with Graph Transformation and PSO Algorithm for Model Transformation Verification," Proceedings of the International Conference on IT Convergence and Security, vol. 120, **(2012)**.