Vol 8. No. 1 – March, 2015

African Journal of Computing & ICT

Behavior-based Retrieval of Software

Moataz Ahmed
Information and Computer Science Department
King Fahd University of Petroleum and Minerals
Dhahran 31261, Saudi Arabia

Hamza Onoruoiza Salami
Department of Computer Science
Federal University of Technology
Minna, Nigeria
moataz@kfupm.edu, ho.salami@futminna.edu.ng

ABSTRACT

Reduced software development cost and time can be achieved by reusing existing software. One of the most important activities during reuse is retrieval. In the early stages of software development, UML state machine diagrams are used to model the behavior of different system objects. This work describes the retrieval of software from a repository by comparing the state machine diagrams of new and existing software systems. State machine diagrams are converted to directed graphs, which are compared using a Genetic Algorithm-based graph matching technique. Experimental results show that the proposed approach is effective in retrieving similar software from a repository.

Keywords- UML; state machine diagram, genetic algorithm; software retrieval; software reuse

African Journal of Computing & ICT Reference Format:
Moataz Ahmeda & Hamza Onoruoiza Salamib (2015). Behavior-based Retrieval of Software.
Afr J. of Comp & ICTs. Vol 8, No. 1. Pp95-.102.

I. INTRODUCTION

Among the benefits of software reuse are reduced risk, development time and overall cost [1]. Even though many software artifacts such as domain models, requirement specifications, designs, documentation, test data and source code can be reused, the benefits of reuse are maximized if it occurs at the early stages of software development. The reason is that when early-stage artifacts are reused, their corresponding later-stage artifacts can be reused as well [2]. Software reuse can be partitioned into four distinct tasks. It begins with the presentation of a query to the reuse system, followed by the retrieval of the software component that is most similar to the query, modification of the retrieved component to meet the needs of the new software, and incorporation of the modified component into the repository to facilitate future reuse [3].

The Unified Modeling Language (UML) is the *de facto* language for modeling systems in the early stages of software development such as during requirement analysis and design. UML diagrams are broadly divided into structure diagrams, which show the static nature of objects in a system irrespective of time, and behavior diagrams which show the dynamic behavior of the system over time [4]. Many of the existing work on UML-based software reuse have concentrated on class diagrams, sequence diagrams and use case diagrams. Consequently, this work focuses on the retrieval of state machine diagrams (SMDs). UML SMDs model the behavior of individual system entities such as objects (i.e., instances of classes) [4]. They show how an object responds to events according to its current state, and how it enters new states [5].

In order to compare SMDs during retrieval, they are converted to graphs, then a graph matching/similarity technique is used to determine the degree of similarity of the graph representations. Genetic algorithm (GA) is used alongside a similarity measure to perform the graph matching/similarity assessment. Experimental results show that this approach is effective in retrieving similar software from a repository

The remainder of this paper is organized as follows: Section II discusses related work. In Section III, we propose a method of representing SMDs as graphs. The similarity measure for SMDs is presented in Section IV. We describe how similarity between states in two UML SMDs is computed in Section V. Matching using GA is the subject of Section VI. We present experimental results in Section VII and conclude the paper in Section VIII.

2. RELATED WORKS

Significant research has been carried out on UML-based software reuse. For example, class diagram retrieval has been described in [6-8], while sequence diagram and use case diagram retrieval is discussed in [9-12]. In some of the existing works, graph representations of UML diagrams have been compared during retrieval: sequence diagrams are converted to graphs in [11, 12]; whereas class diagrams are converted to graphs in [6, 13]. To the best of the authors' knowledge, only Ali and Du [14] have considered SMDs during retrieval. In [14], design models consisting of class, sequence, activity, collaboration and state machine diagrams were described from six perspectives using pre-defined terms.

Similarity between query and repository model was computed in either of two ways: based on the distance of the shortest path between the descriptive terms in a conceptual graph; or from the degree of overlap between the descriptive terms. However, relying on only textual descriptions to compare models results in the loss of structural information contained in the UML diagrams [15].

Lately, some authors have used heuristic search techniques for matching while retrieving UML diagrams. Similarity measures have been combined with GA [13] and particle swarm optimization [6] in order to retrieve class diagrams. Furthermore, GA-based similarity assessment was used for retrieving sequence diagrams in [16]. This paper follows a similar approach to that used in our previous works (i.e., [13, 16]), by first converting SMDs to graphs, then using a similarity measure and GA to determine the degree of similarity of the graph representations.

2.1 Graph Representation of State Machine Diagrams

SMDs can be converted to labeled directed graphs in which each state other than a final state is represented by a node, and all final states are represented by a single node. Four types of edges can connect nodes of the graph: hierarchical edges labelled *H*, which connect composite states to their immediate sub states; transition edges labelled *xT*, which represent transitions between states, where *x* is the number of transitions from one state to another; beginning edges labelled *B*, which denote transitions from the start state; and ending edges labelled *xE*, which represent transitions to the end state, where *x* is the number of transitions from one state to any of the final states. Fig. 1 shows two SMDs *s* and *t*. The graph and adjacency matrix representations of *s* are shown in Fig. 2 and Table I, respectively.

3. SIMILARITY MEASURE

The degree of similarity of SMDs is computed by comparing their adjacency matrix representations. A difference matrix *DiffE* acts as a lookup table that indicates the degree of similarity between the four different types of edges described in Section III. Table II shows *DiffE*. The non-diagonal entries of *DiffE* are ones, indicating maximum dissimilarity. The diagonal entries for beginning edges and hierarchical edges are zero, signifying that identical types of edges have no difference between them. In the case of transition edges and ending edges, their labels indicate the number of transitions from one state to another, hence the diagonal entries take these numbers into account. For example, the difference between a 2*T* edge and a 3*T* edge is 1/2 – 1/3 = 0.17, whereas the difference between a 2*T* edge and a 4*T* edge is 1/2 – 1/4 = 0.25.

Let *adjS* and *adjT* be the adjacency matrices of *s* and *t*, respectively. *adjS* has *ns* rows while *adjT* has *nt* rows ($ns \le nt$). Let *K* be a permutation vector that maps all *ns* nodes of *adjS* to *ns* nodes of *adjT*. In essence, *K* is a one-to-one mapping from all the nodes of *adjS* to some (or all) of the nodes of *adjT*. Furthermore, let adj_T_K be a *ns* X *ns* adjacency matrix containing only the edges between nodes of *adjT* listed in *K*. The degree of similarity between *s* and *t* is given in (1).

$$sim(s,t) = \frac{\sum_{i=1}^{ns}\sum_{j=1}^{ns} DiffE(adjS(i,j), adjT_K(i,j))}{nr} + \lambda\frac{nt-ns}{ns}$$

Eqn ……………… (1)

where
nr is the number of times there is at least one edge at corresponding entry positions in *adjS* or $adjT_K$. $\lambda \in [0, 1]$ is a weight that determines how the unmapped nodes in *adjT* affect the degree of similarity. For example, choosing $\lambda = 0$ causes the similarity score between *s* and *t* to be zero (indicating maximum similarity) whenever *t* subsumes *s*. On the other hand, a large value of λ causes the value of *sim*(*s, t*) to increase when $nt > ns$.

In the remainder of this section, we attempt to theoretically validate the formula given in Eq. (1) by determining if it a similarity metric. Similarity measures which satisfy four metric axioms (self-similarity, minimality, symmetry and triangle inequality) are referred to as similarity metrics [17].

3.1 Self-similarity

Since corresponding edges of identical state machine diagrams are the same, and the diagonal entries of *DiffE* are either zero or reflect the differences in number of edges, the numerator of the first fraction in Eq. (1) is zero. Furthermore, graph representations of identical state machine diagrams have the same number of nodes so the numerator of the second fraction in Eq. (1) is zero. Therefore, $sim(s, s) = sim(t, t) = 0$.

3.2 Minimality

There are two cases to consider:

Case 1: if $s = t$, it follows that $sim(s, t) = sim(s, s) = 0$ from the first axiom.

Case 2: if $s \ne t$, either or both of the following conditions is true: (i) there is at least one pair of nodes whose corresponding edges in *adj_s* and *adj_tK* are of different types or have different multiplicities. Thus, the numerator of the first fraction in Eq. (1) is greater than zero (ii) *s* and *t* have different number of nodes, thus the numerator of the second fraction in Eq. (1) is greater than zero. If condition (i) and/or (ii) is satisfied, $sim(s, t) \in (0, 1]$.
Thus, $sim(s, t) \ge sim(s, s)$

3.3 Symmetry

Clearly, $sim(s, t) = sim(t, s)$ since *DiffE* is symmetric.

3.4 Triangular Inequality

We have not been able to prove that Eq. (1) satisfies triangular inequality, thus, the formula in Eq. (1) shall be referred to as a similarity measure rather than a metric.

3.5 Similarity Matrix of States

This section describes a method of computing pairwise similarities between states of two SMDs. The similarity values are kept in a states' similarity matrix *SS*, which will be used during matching. Each state other than a final state (all final states are listed as one state) is represented by a 10-dimensional vector indicating 10 properties of the state.

These properties are listed in Table III, while their values are given in Table IV for *s*. The similarity between nodes is the Euclidean distance of their feature vectors. Table V shows *SS* containing the pairwise similarity values between states in *s* and *t*.

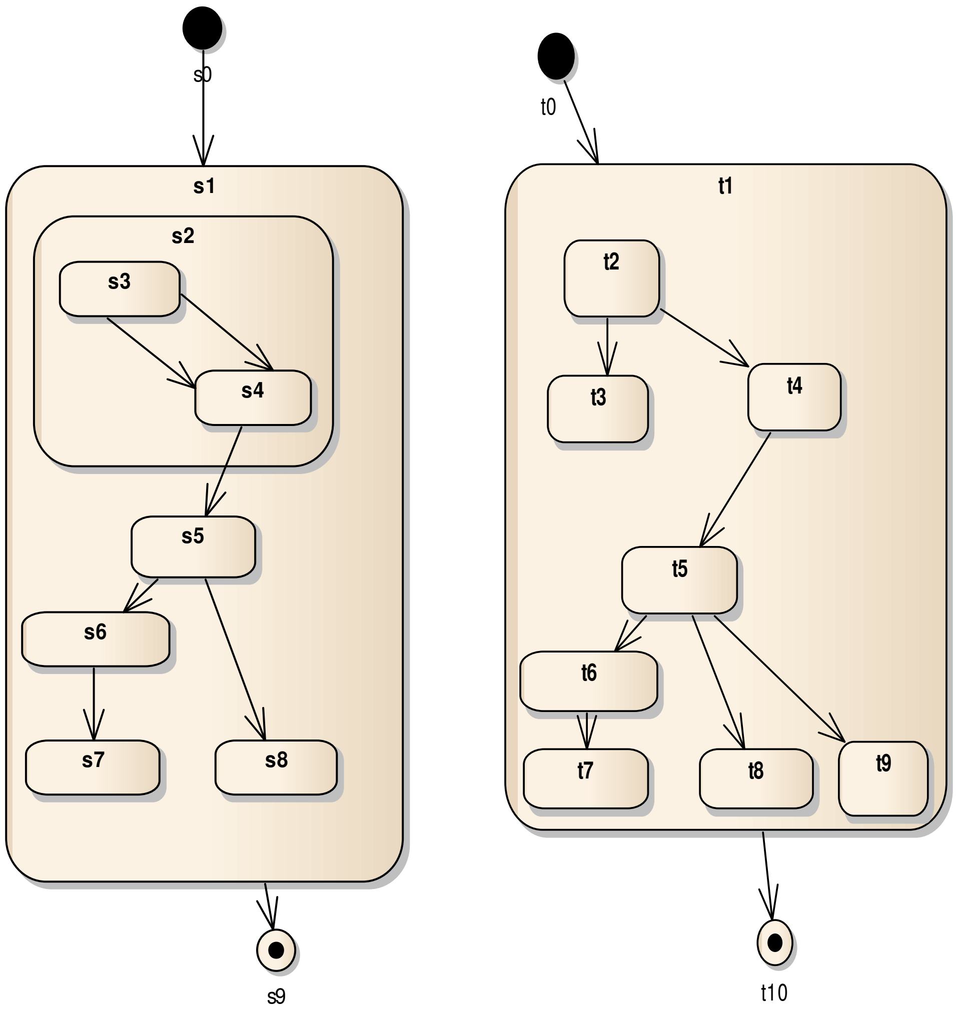

Figure 1. Two state machine diagrams *s* and *t*.

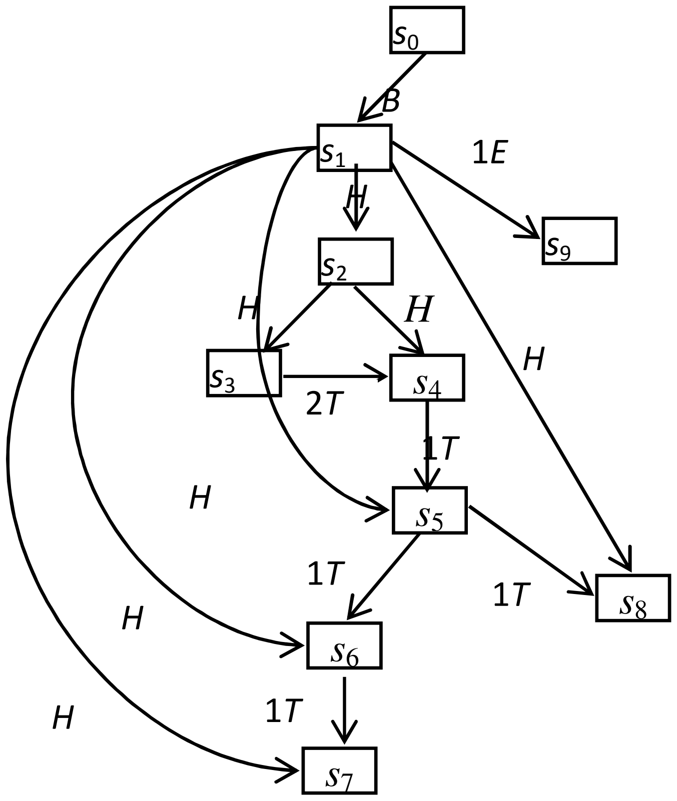

Figure 2. Graph representation of *s*

TABLE I. ADJACENCY MATRIX REPRESENTATION OF *S*

	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9
s_0	-	*B*	-	-	-	-	-	-	-	-
s_1	-	-	H	-	-	*H*	*H*	*H*	*H*	1*E*
s_2	-	-	-	*H*	*H*	-	-	-	-	-
s_3	-	-	-	-	2*T*	-	-	-	-	-
s_4	-	-	-	-	-	1*T*	-	-	-	-
s_5	-	-	-	-	-	-	1*T*	-	1*T*	-
s_6	-	-	-	-	-	-	-	1*T*	-	-
s_7	-	-	-	-	-	-	-	-	-	-
s_8	-	-	-	-	-	-	-	-	-	-
s_9	-	-	-	-	-	-	-	-	-	-

a. *B* = beginning edge, *H* = hierarchical edge, *xT* = *x* transitions, *yE* = *y* ending edges, - = no edge

TABLE II. DIFFE

	B	***H***	***yT***	***yE***
B	0	1	1	1
H	1	0	1	1
xT	1	1	|1/x – 1/y|	1
yE	1	1	1	|1/x – 1/y|

b. *B* = beginning edge, *H* = hierarchical edge, *xT* or *yT* = *x* or *y* transitions, *xE* or *yE* = *x* or *y* ending edges, - = no edge, |…| = absolute value

© 2015 Afr J Comp & ICT – All Rights Reserved - ISSN 2006-178
www.ajocict.net

TABLE III. FEATURES OF EACH STATE

Feature	Description
f_1	No. of transitions coming from the start state
f_2	No. of transitions coming in (except from the start state)
f_3	No. of transitions to a finish state
f_4	No. of transitions going out (except to finish states)
f_5	No. of states whose next state is this state
f_6	No. of next states
f_7	No. of ancestors
f_8	No. of descendants
f_9	No. of child states
f_{10}	Length of longest path from this state to its descendants

TABLE IV. FEATURES OF S

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
s_0	0	0	0	1	0	1	0	0	0	0
s_1	1	0	1	0	1	1	0	7	5	2
s_2	0	0	0	0	0	0	1	2	2	1
s_3	0	0	0	2	0	0	2	0	0	0
s_4	0	2	0	1	0	1	2	0	0	0
s_5	0	1	0	2	1	2	1	0	0	0
s_6	0	1	0	1	1	1	1	0	0	0
s_7	0	1	0	0	1	0	1	0	0	0
s_8	0	1	0	0	1	0	1	0	0	0
s_9	0	1	0	0	1	0	0	0	0	0

TABLE V. STATES' SIMILARITY MATRIX SS

	t0	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10
s0	0.00	11.53	1.73	2.24	1.73	3.32	1.73	2.24	2.24	2.24	2.00
s1	9.06	3.32	9.33	9.11	9.11	9.75	9.11	9.11	9.11	9.11	9.06
s2	3.46	8.77	4.12	3.32	3.61	5.39	3.61	3.32	3.32	3.32	3.46
s3	2.45	11.87	2.24	2.65	2.24	3.61	2.24	2.65	2.65	2.65	3.16
s4	2.83	11.87	2.65	2.24	1.73	3.32	1.73	2.24	2.24	2.24	2.83
s5	2.24	11.75	1.41	2.83	1.41	1.41	1.41	2.83	2.83	2.83	3.00
s6	1.73	11.58	2.00	1.41	0.00	2.83	0.00	1.41	1.41	1.41	1.73
s7	2.24	11.58	3.16	0.00	1.41	4.24	1.41	0.00	0.00	0.00	1.00
s8	2.24	11.58	3.16	0.00	1.41	4.24	1.41	0.00	0.00	0.00	1.00
s9	2.00	11.53	3.32	1.00	1.73	4.36	1.73	1.00	1.00	1.00	0.00

3.6 Matching Using Genetic Algorithm

Determining the value of K that results in an optimal (i.e., smallest) similarity value between s and t is a combinatorial optimization problem which may involve a huge search space. This section describes the use of GA to find a suitable value of K in order to compute $sim(s, t)$. GA is a powerful heuristic search algorithm that can used to solve combinatorial optimization problems. The GA used in this paper is similar to that used for graph matching in [18].

3.7 Chromosome Encoding and Population Initialization.

The number in the i^{th} gene indicates which node in t is mapped to the i^{th} node of s. In other words, each chromosome is of the same form as K. Figure 3 shows the how a chromosome encodes the mapping of states in two SMDs.

The initial population is constructed in three steps: (i) the first individual is formed by applying Munkres' allocation algorithm [19] on SS. (ii) A few additional individuals are generated by mutating the first individual. (iii) All other individuals are generated by randomly assigning values to their genes.

3.8Fitness Values

The fitness of a gene is read from SS. For example, if the i^{th} gene of a chromosome contains j, its fitness is $SS(i, j)$. The fitness of a chromosome is computed using Eq. (1).

3.9 Selection and Crossover

The selection and crossover operations are the same as those described in [16].

3.10 Mutation

Mutation involves swapping two randomly selected genes, or replacing a gene with a value that is not currently in the chromosome.

3.11 Uniqueness of individuals

When the population contains identical individuals, one of them is mutated until it becomes distinct from all other individuals in the population.

3.12 Termination Conditions

The GA terminates when any of the following three conditions is satisfied: the optimal similarity value of zero is obtained; the maximum number of generations is reached; or the population's fitness value does not improve within a fixed number of generations.

4. EXPERIMENTAL RESULTS AND DISCUSSION

In the experiments to evaluate the proposed method of retrieving software based on a comparison of their SMDs, we created a repository of 16 SMDs belonging to three domains: 7 diagrams are from the banking/business domain; 6 diagrams are from the education domain; while the other 3 diagrams are related to personal organization tasks such as managing diaries and appointments. Table VI summarizes the characteristics of the repository diagrams. 16 queries were formed by taking each of the repository diagrams in turn. A repository diagram is relevant to a query only if they belong to the same domain.

Retrieval quality was assessed using the Mean Average Precision (MAP), which is widely used for evaluating information retrieval systems. The average precision (AP) for a query is obtained using precision values calculated at each point when a relevant document is found. MAP for a set of queries is the mean of the AP scores for each query [20]. MAP can be computed using Eq. (2):

$$MAP = \frac{1}{N}\sum_{j=1}^{N}\frac{1}{Q_j}\sum_{i=1}^{Q_j}P(rel = i) \quad \text{.........(2)}$$

Where

N is the number of queries, Q_j is the number of relevant documents for query j and $P(rel = i)$ is the precision at the i^{th} relevant document.

The following parameters were used: size of population = 50; maximum number of generations = 100; number of generations to terminate GA if fitness value does not improve = 20; probability of mutation of genes = 0.10; and number of individuals from initial generation produced using Munkres' algorithm = 3. λ was set to 0.05 in order to compute fitness values using Eq. (1). The experiment was repeated 30 times. Table VII shows the mean MAP over 30 runs for the 16 queries. The standard deviation of MAP is shown in brackets. The time to search the repository is also presented in the table. The experiment was carried out using Matlab® computing language, on a personal computer having the following configuration: 2.67 GHz Intel Core 2 Quad processor; 4 GB RAM; and 32-bit Windows 7 operating system.

From the results presented in Table VII, our technique was capable of retrieving the most similar software from the repository. The standard deviation of MAP from 30 runs is very low, suggesting that our matching technique consistently produces good results.

5. CONCLUSION

This paper described an effective method of retrieving software for reuse by comparing the behavior of the software. The behaviors of software are manifested in the SMDs that show how events lead to change in state of system objects. A graph matching/similarity technique was used to determine the similarity of graph representations of SMDs. Experimental results show that the proposed method is promising.

The SMD similarity assessment technique described in this paper did not take into account the events, guard conditions and actions of transitions, as well as the names of states in SMDs. As a future work, these other pieces of information can be incorporated into the similarity assessment technique to determine if it leads to improved retrieval quality.

ACKNOWLEDGMENT

The authors would like to acknowledge the support provided by the Deanship of Research at King Fahd University of Petroleum and Minerals, Saudi Arabia under Research Grant 11-INF1633-04.

REFERENCES

[1] I. Sommerville, *Software Engineering*, 9th ed.: Pearson Addison Wesley, 2010.

[2] J. L. Cybulski, R. D. B. Neal, A. Kram, and J. C. Allen, "Reuse of early life-cycle artifacts: workproducts, methods and tools," *Ann. Softw. Eng.,* vol. 5, pp. 227-251, 1998.

[3] A. Prasad and E. K. Park, "Reuse system: An artificial intelligence - based approach," *Journal of Systems and Software,* vol. 27, pp. 207-221, 1994.

[4] OMG, "Unified Modeling Language Superstructure Specification V2.4.1," 2011.

[5] P. Roques, *UML in Practice: The Art of Modeling Software Systems Demonstrated through Worked Examples and Solutions*: Wiley, 2004.

[6] W. K. G. Assuncao and S. R. Vergilio, "Class Diagram Retrieval with Particle Swarm Optimization," in *The 25th International Conference on Software Engineering and Knowledge Engineering (SEKE 2013)*, 2013, pp. 632 - 637.

[7] P. Gomes, F. C. Pereira, P. Paiva, N. Seco, P. Carreiro, J. L. Ferreira, and C. Bento, "Case Retrieval of Software Designs using WordNet," in *European Conference on Artificial Intelligence (ECAI 02)*, 2002, pp. 245-249.

[8] K. Robles, A. Fraga, J. Morato, and J. Llorens, "Towards an ontology-based retrieval of UML Class Diagrams," *Information and Software Technology,* vol. 54, pp. 72-86, 2012.

[9] M. C. Blok and J. L. Cybulski, "Reusing UML Specifications in a Constrained Application Domain," in *Proceedings of the Fifth Asia Pacific Software Engineering Conference*: IEEE Computer Society, 1998.

[10] Y. Kotb, "Applying the Textual Entailment Approach to Automatic Reusable Software," in *The 7th International Conference on Informatics and Systems (INFOS)*, 2010, pp. 1-6.

[11] W.-J. Park and D.-H. Bae, "A two-stage framework for UML specification matching," *Inf. Softw. Technol.,* vol. 53, pp. 230-244, 2010.

[12] W. N. Robinson and H. G. Woo, "Finding Reusable UML Sequence Diagrams Automatically," *IEEE Softw.,* vol. 21, pp. 60-67, 2004.

[13] H. O. Salami and M. Ahmed, "Class Diagram Retrieval Using Genetic Algorithm," in *12th International Conference on Machine Learning and Applications* Miami, Florida, 2013, pp. 96-101.

[14] F. M. Ali and W. Du, "Toward reuse of object-oriented software design models," *Information and Software Technology,* vol. 46, pp. 499 - 517, 2004.

[15] K. Wolter, T. Kreb, and L. Hotz, "Determining Similarity of Model-based and Descriptive Requirements by Combining Different Similarity Measures," in *Proceedings of 2nd International Workshop on Model Reuse Strategies*, 2008.

[16] H. O. Salami and M. A. Ahmed, "Retrieving sequence diagrams using genetic algorithm," in *Proceedings of 11th International Joint Conference on Computer Sciences and Software Engineering*, Chonburi, Thailand, 2014, pp. 324-330

[17] S. Santini and R. Jain, "Similarity measures," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 21, pp. 871-883, 1999.

[18] Y. Wang and N. Ishii, "A genetic algorithm and its parallelization for graph matching with similarity measures," *Artificial Life and Robotics,* vol. 2, pp. 68-73, 1998.

[19] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the Society for Industrial and Applied Mathematics,* vol. 5, pp. 32-38, 1957.

[20] S. Teufel, "An overview of evaluation methods in TREC ad hoc information retrieval and TREC question answering," *Evaluation of Text and Speech Systems,* pp. 163-186, 2007.

© 2015 Afr J Comp & ICT – All Rights Reserved - ISSN 2006-178
www.ajocict.net

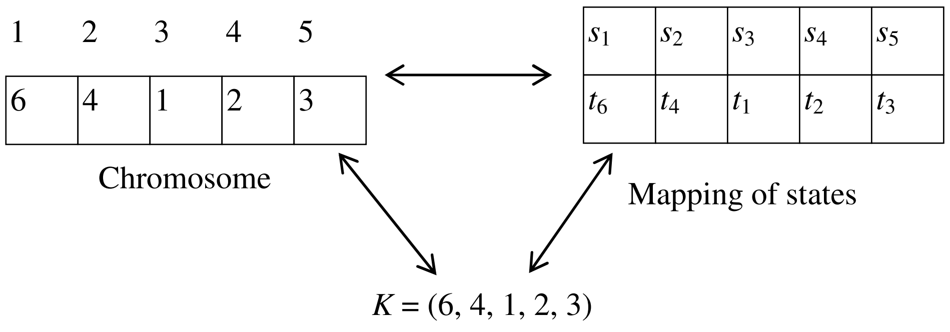

Figure 3. Chromosome encoding for comparing two state machine diagrams

TABLE VI. SUMMARY DETAILS OF REPOSITORY DIAGRAMS

	Banking/Business							**Education**						**Personal Organization**		
No. of states	10	4	4	6	5	8	5	10	7	8	3	6	6	5	5	5
No. of transitions	14	4	4	5	8	15	5	14	12	9	2	7	6	8	8	4

TABLE VII. RESULTS OF EXPERIMENTS

MAP (%)	**time to search repository (seconds)**
73.27 (0.24)	1.32