Prioritizing Coverage-Oriented Testing Process – An Adaptive-Learning-Based Approach and Case Study

Fevzi Belli
University of Paderborn, Germany
e-mail: belli@upb.de

Mubariz Eminov
Mugla University, Turkey.
e-mail: *meminov@mu.edu.tr*

Nida Gökçe
Mugla University, Turkey
e-mail: nidagokce@yahoo.com

Abstract

This paper proposes a graph-model-based approach to prioritizing the test process. Tests are ranked according to their preference degrees which are determined indirectly, i.e., through classifying the events. To construct the groups of events, unsupervised neural network is trained by adaptive competitive learning algorithm. A case study demonstrates and validates the approach.

1. Introduction and Related Work

For a productive generation of tests, model-based techniques focus on particular, relevant aspects of the requirements of the system under test (SUT) and its environment. Real-life SUTs have, however, numerous features that are to simultaneously be considered, often leading to a large number of tests. In such cases, because of time and cost constraints, the entire set of system features cannot be considered. It is then essential to model the relevant ones. The modeled features are either functional behavior or structural issues of the SUT, leading to *specification-oriented* testing or *implementation-oriented* testing, respectively. Once the model is established, it ´guides´ the test process to generate and select test cases, which form sets of test cases (also called *test suites*). The test selection is ruled by an *adequacy criterion*, which provides a measure of how effective a given set of test cases is in terms of its potential to reveal faults [1, 16]. Some of the existing adequacy criteria are coverage-oriented. They use the ratio of the portion of the specification or code that is covered by the given test set in relation to the uncovered portion in order to determine the point in time at which to stop testing (*test termination problem*).

This paper is on model-based, specification- and coverage-oriented testing. The underlying model graphically represents the system behavior interacting with the user's actions. In this context, *event sequence graphs* (*ESG*, [5-7]) are favored. ESG approach view the system's behavior and user's actions as events, more precisely, as *desirable events,* if they are in accordance with the user expectations, otherwise they are *undesirable events*. Mathematically speaking, a complementary view of the behavioral model is generated from the model given. Thus, the model will be exploited twice, i.e., once to validate the system behavior under regular conditions and a second time to test its robustness under irregular, unexpected conditions.

The costs of testing often tend to run out the limits of the test budget. In those cases, the tester may request a complete test suite and attempt to run as many tests as affordable, without running out the budget. Therefore, it is important to test the most important items first. This leads to the *Test Case Prioritization Problem (TCPP).*

Existing approaches to solving TCCP usually suggest constructing a density covering array in which all pair-wise interactions are covered [2, 3]. Generally speaking, every n-tuple is then qualified by a number $n \in \mathbb{N}$ ($\mathbb{N}$: set of natural numbers) of values to each of which a degree of importance is assigned. In order to capture significant interactions among pairs of choices the importance of pairs is defined as the ´benefit´ of the tests. Every pair covered by the test contributes to the total benefit of a test suite by its individual benefit. Therefore, the tests given by a test suite are to be ordered according to the importance of corresponding pairs. However, such interaction-based, prioritized algorithms are computationally complex and thus usually less effective [18, 19].

The ESG approach favored in this paper generates test suites through a finite sequence of discrete events. The underlying optimization problem is a generalization of the *Chinese Postman Problem* (*CPP*) [8] and algorithms given in [5-7] differ from the well-known ones in that they satisfy not only the constraint that a minimum total length of test sequences is required, but also fulfill the coverage criterion with respect to converging of all event pairs represented graphically. This is substantial to solve the test termination problem and makes out a significant difference of this present paper from existing approaches. To overcome the problem that an exhaustive testing might be infeasible, the present paper develops a *prioritized* version of the mentioned test generation and optimization algorithms, in sense of "divide and conquer" principle. This is the primary objective and the kernel of this paper which is novel and thus, to our knowledge, has not yet been worked out in previous work.

The required prioritization has to meet the needs and preferences of test management how to spend the test budget. However, SUT and software objects, i.e., components, architecture, etc., usually have a great variety of features. Therefore, test prioritization entails the determination of order relation(s) for these features. Generally speaking, we have *n* objects, whereby each object has a number (*p*) of features that we call *dimension*. TCPP then represents the comparison of test objects of different, multiple dimensions. To our knowledge, none of the existing approaches take the fact into account that SUT usually has a set of attributes and not a single one when prioritizing the test process. Being of enormous practical relevance, this is a tough, *np*-complete problem.

Our approach assigns to each of the tests generated a degree of its preference. This degree is indirectly determined through estimation of the events qualified by several attributes. We suggest representing those events as an unstructured multidimensional data set and dividing them into groups which correspond to their importance. Beforehand, the optimal number of those groups is determined by using V_{sv} *index-based clustering validity algorithm* [13, 14]. To derive the groups of events we use a clustering approach based on unsupervised neural network (NN) that will be trained by an adaptive competitive learning (CL) algorithm [12]. Different from the existing approaches, e.g., as described in [9, 10, 13], input and weight vectors are normalized, i.e., they have length one. This enables less sensitivity to initialization and a good classification performance. The effectiveness of the proposed testing approach is demonstrated and validated by a case study a non-trivial commercial system.

The paper is organized as follows. Section 2 explains the background of the approach, presenting also the definition of neural network-based clustering. Section 3 describes the proposed prioritized graph-based testing approach. Section 4 includes the case study. Section 5 summarizes the results, gives hints to further research and concludes the paper.

2. Background

2.1. Event Sequence Graphs

Because the construction of ESG, test generation from ESG and test process optimization are sufficiently explained in the literature ([5-7, 15]), the present paper summarizes ESG concept, as far as it is necessary and sufficient to understand the test prioritization approach represented in this paper.

Basically, an *event* is an externally observable phenomenon, such as an environmental or a user stimulus, or a system response, punctuating different stages of the system activity. A simple example of an ESG is given in Figure 1. Mathematically, an ESG is a directed, labeled graph and may be thought of as an ordered pair *ESG=(α, E)*, where α is a finite set of nodes (vertices) uniquely labeled by some input symbols of the alphabet Σ, denoting events, and *E*: $\alpha \rightarrow \alpha$, a precedence relation, possibly empty, on α. The elements of *E* represent directed arcs (edges) between the nodes in α. Given two nodes *a* and *b* in α, a directed arc *ab* from *a* to *b* signifies that event *b* can follow event *a*, defining an *event pair* (*EP*) *ab* (Figure 1). The remaining pairs given by the alphabet Σ, but not in the ESG, form the set of *faulty event pairs* (*FEP*), e.g., *ba*. As a convention, a dedicated, start vertex, e.g., *[*, is the *entry* of the ESG whereas a final vertex e.g., *]* represents the *exit*. Note that *[* and *]* are not included in Σ; therefore, the arcs from and to them form neither EP nor FEP. The set of FEPs constitutes the *complement* of the given ESG ($\overline{ESG}$). Superposition of ESG and $\overline{ESG}$ leads to completed ESG ($\widehat{ESG}$) (Figure 1).

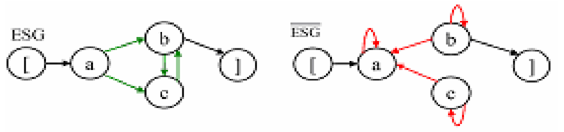


Figure 1. An event sequence graph ESG, its complement $\overline{ESG}$

A sequence of *n+1* consecutive events that represents the sequence of *n* arcs is called a *event sequence (ES) of the length n+1*, e.g., an *EP* (*event pair*) is an ES of length 2. An ES is *complete* if it starts at the initial state of the ESG and ends at the final event; in this case it is called a *complete ES* (*CES*). Occasionally, we call CES also *walks* (or *paths*) through the ESG given. A *faulty event sequence* (*FES*) *of the length n* consists of *n-1* subsequent events that form an ES of length *n-2* plus a concluding, subsequent FEP. An FES is *complete* if it starts at the initial state of the ESG; in this case it is called *faulty complete ES*, abbreviated as *FCES*. A FCES must not necessarily end at the final event.

2.2. Neural Network-Based Clustering

Clustering is a technique to generate an optimal partition of a given, supposedly unstructured, data set into a predefined number of *clusters* (or *groups*). Homogeneity within the groups and heterogeneity between them can be settled by means of unsupervised neural network-based *clustering algorithms* [12, 14]. To provide *training* of NN, a number of *learning algorithms* are used. We deal with the family of *competitive learning* (*CL*) algorithms that are a type of self-organizing networking models. According to CL algorithms, ´winning´ weight vector (weights of connections between input and output nodes) or *cluster center* can be adjusted by applying "winner-takes-all" strategy in training phase of the NN under consideration. In clustering a data set

$$X=\{x_1,...,x_i,...,x_n\}\subset\mathbb{R}^p,\ x_i=(x_{i1},...,x_{ij},...x_{ip})\in\mathbb{R}^p \quad (1)$$

where $\mathbb{R}$ is the set of real numbers and p is the number of dimension, which is to be partitioned into c number of clusters each of which contains a data subset S_k defined as follows:

$$X=\bigcup_{k=1}^{c} S_k \text{ with } S_k \cap S_j = 0 \quad \forall k \neq j \quad (2)$$

Each cluster is represented by a cluster center (*prototype*) that corresponds to a weight vector $w=(w_{k1},...,w_{kj},...,w_{kp})\in\mathbb{R}^p$ and after finding a trained value of all weight vectors $W=\{w_1,...,w_k,...,w_c\}\subset\mathbb{R}^p$ a data set $w\in\mathbb{R}^p$ is divided into k^{th} cluster by the condition:

$$S_k=\left\{x\in\mathbb{R}^p \,\middle|\, \|x-w_k\|\le\|x-w_j\| \quad \forall k\neq j\right\} \; k=1,...,c\in\mathbb{N} \quad (3)$$

To determine the optimal number c of groups, the V_{sv} *index-based cluster validity algorithm* [14] has been used. Furthermore, in order to divide the given data set into c groups the NN for training of which the adaptive CL algorithm [12] is applied.

Training: In the *training* phase the weight vectors of NN are updated usually according to Standard CL algorithm as follows: Firstly, for a data point $x_i\in\mathbb{R}^p$ selected from X the winner weight vector w_w *i*s determined by:

$$\mathrm{w}_W=\arg\min_k\left\{\|\mathrm{x}_i-\mathrm{w}_k\|\right\} \quad i=1,...,n\in\mathbb{N} \quad k=1,...,c\in\mathbb{N} \quad (4)$$

where $\|.\|$ is *Euclidean distance measure*. Then this vector is adjusted at step t by

$$\Delta\mathrm{w}_W(t)=\eta(t)(\mathrm{x}_i-\mathrm{w}_W) \quad (5)$$

where $\eta(t)$ is a *learning rate*. Secondly, the adjusted winner vector is calculated by

$$w_W(t)=w_W(t-1)+\Delta w_W(t) \quad (6)$$

Training process is iteratively preceded until the convergence condition for the all weight vectors is satisfied. Clearly, the CL algorithm actually seeks for a local minimum (with respect to the predetermined number of clusters) for squared error criterion by applying gradient descent optimization.

If the probability distribution function of a data set is given in advance, then the simple standard CL algorithm described above may get good quality of a clustering minimizing *mean squared error* (*MSE*) defined as [9-14]

$$E=\sum_{k=1}^{c} D_k \quad (7)$$

where D_k be *intra-cluster error* for k^th cluster S_k that is determined by

$$D_k=\frac{1}{p}\left(\sum_{x\in S_k}\|x-w_k\|^2\right) \quad (8)$$

However, in general, the distribution is not given in advance; hence the initial values of the weight vectors are randomly allotted. It negatively influences the clustering performance of considered CL algorithm.

In order to get a better clustering performance, in this paper we use the adaptive CL algorithm with deleting of weight vectors [9, 10, 12]. The main properties of this algorithm are:

a) Both data points $\tilde{x}_i$ and weight vectors $\tilde{w}_k$ are normalized to a unit length, i.e., they are presented as the unit vector the length of which is 1 (one).

b) The winner vector is determined by a dot product of data point $\tilde{x}_i$ and weight vector $\tilde{w}_k$ instead of (4)

$$\tilde{\mathrm{w}}_W=\arg\max_k\left\{\sum_{j=1}^{p}\tilde{x}_{ij}\tilde{w}_{kj}\right\} \quad i=1,...,n \quad k=1,...,c \quad (9)$$

or through the angle between these vectors as

$$\tilde{\mathrm{w}}_W{}^{\theta}=\arg\min_k\{\theta_k\} \quad k=1,...,c\in\mathbb{N} \quad (10)$$

which corresponds to cosine θ that is maximum, i.e.,1

c) The updating rule of a winner weight vector instead of (5) is based on the adjusting equation [11] expressed as follows

$$\Delta\tilde{w}_W(t)=\eta(t)\left(\frac{\tilde{x}_i}{p}-\tilde{w}_W\right) \quad (11)$$

d) There is a deletion mechanism that (starting with a greater number than the prepared, required number of weight vectors) sequentially eliminates one vector, w_s that has a minimum intra-cluster partition error, i.e., $D_k\ge D_s$, for all k, determined as

$$D_k=\frac{1}{p}\left(\sum_{\tilde{x}\in S_k}\tilde{x}\tilde{\mathrm{w}}_W\right) \quad k=1,...,c \quad (12)$$

and it proceeds until the number of weight vectors is equal to the predetermined one. i.e., the optimal number of clusters by using cluster validity algorithm [14], as mentioned above.

Adaptive Competitive Learning Algorithm:
Step 1. Initialization:

Initial number of output neurons l_0, final number of neurons l, maximum iteration $T_{\max}$, initial iteration of deletion $t_0 = T_{\max}/3$ and partial iteration $u = T_{\max}/3(l_0 - l + 1)$, Set $t \leftarrow 0$ and $m \leftarrow l_0$

Step 2. Standard Competitive Learning:

2.1. Choose an input vector $\tilde{x}_i$ at random among X

2.2. Select a winner $\tilde{w}_k$ according to (9)

2.3. Update the winner $\tilde{w}_w$ vector according to (11)

2.4. Set $t \leftarrow t + 1$

2.5. If $m > l$ and $t = t_0 + u \times q$ than go to Step 3, otherwise go to 2.1.

Step 3. Deletion Mechanism:

3.1. Delete $\tilde{w}_s$ calculating D_s according to (12) and checking $D_k \geq D_s$

3.2. Set $m \leftarrow m - 1$

Step 4. Termination Condition:

If $t = T_{\max}$ then terminate, otherwise go to Step 2.

Classification: After finding a value of weight vectors $\{w_1, ..., w_c\}$ that correspond to cluster centers, respectively, a data set is divided into c groups as follows:

$$S_k = \left\{ x \in \mathbb{R}^p \,\middle|\, \sum_{j=1}^{p} \tilde{x}_{ij} \tilde{w}_{kj} \geq \sum_{j=1}^{p} \tilde{x}_{ij} \tilde{w}_{mj} \;\; \forall k \neq m \right\} \quad (13)$$

$$i = 1, ..., n \quad j = 1, ..., p \quad k = 1, ..., c \quad m = 1, ..., c \in \mathbb{N}$$

Classification performance of the considered clustering algorithm was estimated by the MSE calculated using (7) and (8). Effectiveness of this algorithm was verified for different types of data sets in [12]. Computational time for classification depends on the number *n* of the events and the number *p* of the attributes.

3. Prioritized ESG-Based Testing

We consider the testing process based on the generation of a test suite from ESG that is a discrete model of a SUT. To generate tests, firstly a set of ESGs are derived which are input to the generation algorithm to be applied. We deal with the test generation algorithms [5,7] that generates tests for a given ESG and satisfies the following coverage criteria.

a) Cover all event pairs in the ESG.

b) Cover all faulty event pairs derived by the $\overline{ESG}$.

Note that a test suite that satisfies the first criterion consists of CESs while a test suite that satisfies the second consists of FCESs. These algorithms are able to provide the following constraints:

a) The sum of the lengths of the generated CESs should be minimal.

b) The sum of the lengths of the generated FCESs should be minimal.

The constraints on total lengths of the tests generated enable a considerable reduction in the cost of the test execution and thus the algorithms mentioned above can be referred to as the relatively efficient ones. However, as stated in Section 1, an entire test suite generated may not be executed due to limited project budget. Such circumstances entail ordering all tests to be checked and exercised as far as they do not exceed the test budget. To solve the test prioritizing problem, several algorithms have been introduced [1, 2]. Usually, during the test process for each *n-tuple* (in particular pair-wise) interaction a degree of importance is computationally determined and assigned to the corresponding test case. However, this kind of prioritized testing is computationally complex and hence restricted to deal with short test cases only.

Our prioritized testing approach is based on the ESG-based testing algorithms mentioned above. Note that our test suite consists of CESs which start at the entry of the ESG and end of its exit, representing *walks* (*paths*) through the ESG under consideration. This assumption enables to order the generated tests, i.e., CESs.

The ordering of the CESs is in accordance with their preference degree which is defined indirectly, i.e., by estimation of events that are the nodes of ESG and represent objects (modules, components) of SUT. For this aim, firstly events are presented as a *multidimensional event vector* $x_i = (x_1, ..., x_p)$ where p is the *number of attributes*.

Definition of the Attributes of Events: To qualify an event corresponding to a node in ESG, as a arbitrarily chosen example, we propose to use following 9 attributes, i.e., *p*=9, that determine the *dimension* of a data point represented in a data set. These attributes are given below:

x_1 : The number of sub-windows to reach an event from the entry [(gives its distance to the beginning).

x_2 : The number of incoming and outgoing edges (invokes usage density of a node, i.e., an event).

x_3 : The number of nodes (events) which are directly and indirectly reachable from an event except entry and exit (indicates its “traffic” significance).

x_4 : The maximum number of nodes to the entry [(its maximum distance in terms of events to the entry).

x_5: The number of nodes (events) of a sub-node as sub-menus that can be reached from this node (maximum number of sub-functions that can be invoked further).

x_6 : The total number of occurrences of an event (a node) within all CESs, i.e., walks (significance of an event).

x_7 : The *balancing degree* determines balancing a node as the sum of all incoming edges (as plus (+)) and outgoing edges (as minus (-)) for a given node.

x_8: The averaged frequencies of the usage of events within the CESs (determines the averaged occurrence of each event within all CESs).

x_9 : The number of FEPs connected to the node under consideration (takes the number of all potential faulty events entailed by the event given into account).

Indirect Determination of the Preference:

Step1. Construction of a set of events $X=\{x_{ij}\}$ where $i=1,...,n\in\mathbb{N}$ is an event index, and $j=1,...,p\in\mathbb{N}$ is an attribute index.

Step2. Training the NN using adaptive CL algorithm (see Section 2.2).

Step3. Classification of the events into c groups (see(13), Section 2.2).

Step4. Determination of importance degrees of groups according to length (ℓ) of weight vectors,

Step5. Determination of importance degrees of event groups (see (14), this present section)

Step6. An ordering of the CESs for prioritizing the test process.

Definition of importance degree and preference: The CESs are manually ordered, scaling their preference degrees based on the events which incorporate the importance group(s). *Importance* (*Imp(e)*) of e^{th} event is defined as follows:

$$\mathrm{Imp}(e) = c - \mathrm{ImpD}(S_k) + 1 \qquad (14)$$

where c is the optimal number of the groups; $ImpD(S_k)$ is defined by means of the importance degree of the group S_k to which the e^{th} event belongs.

Finally, choosing the events from the ordered groups, a ranking of CESs is formed according to their descending preference degrees. The assignment of preference degrees to CESs is based on the following rule:

a) The CES under consideration has the highest degree if it contains the events which belong to the "top" group(s) with utmost importance degrees, i.e., that is placed within the highest part of group ordering.

b) The CES under consideration has the lowest degree if it contains the events which belong to the group(s) that are within the lowest part of the "bottom" group(s) with least importance degree i.e., that is placed within the lowest part of group ordering.

Therefore, the preference degree of CES can be defined by taking into account both the importance of events (see 14) and the frequency of occurrence of event(s) within them that is formulated as follows:

$$\mathrm{Pref}(\mathrm{CES}_q) = \sum_{e=1}^{n} \mathrm{Imp}(e)\, f_q(e) \qquad q = 1,...,m \in \mathbb{N} \qquad (15)$$

where m is the number of CESs, n is the number of events, *Imp(e)* is importance degree of the e^{th} event (see (14)) and $f_q(e)$ is frequency of occurrence of event e within CES_q. This order determines the *preference degree* (*Pref(CES$_q$)*) of CESs as test cases (see (15).

4. A Case Study

Based on the web-based system ISELTA (*Isik's System for Enterprise-Level Web-Centric Tourist Applications*), we now present a case study to validate the testing approach presented in the previous sections [15]. Both the construction of ESGs, and generation of test cases from those ESGs, have been explained in the previous papers of the first author [5-7]. Therefore, the case study concentrates on test prioritizing problem.

ISELTA has been developed by our group in cooperation with a commercial enterprise to market various tourist services for traveling, recreation and vacation. It can be used by hotel owners, travel agents, etc., but also by end consumers. A screenshot in Figure 2 demonstrates how to define and reserve rooms of different types.

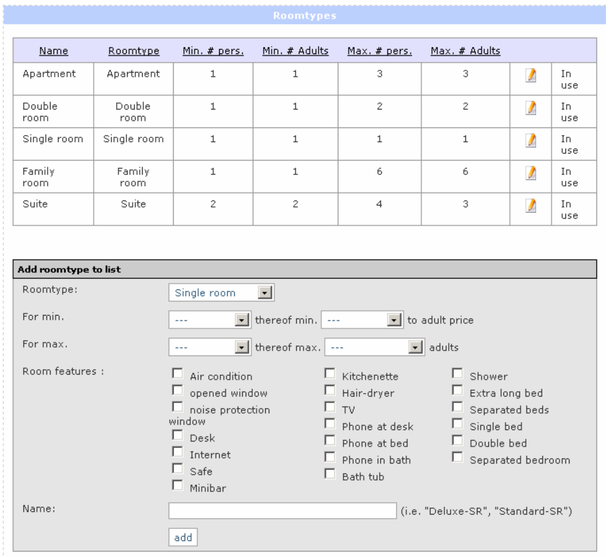
Derivation of the Test Cases: Figure 3 depicts the completed ESG of the scenario described above and in Figure 2. Test cases can now be generated using the algorithms mentioned in Section 3 and described in [6, 7] in detail. For the lack of space, reference is made to these papers and the CESs generated are listed below:

CES_1 = [4 5 4 5 9 1 4 5 10 1 4 5 11 1 4 5 9 2 3 4 5 10 2 3 2 3 1 4 5 11 2 3 4 6 4 6 7 8 1 2 3]
CES_2 = [1 4 5 9]; CES_3 = [2 3 4 6 7 8 2 3 4 5 10]
CES_4 = [4 5 11]; CES_5= [4 6 7 8]

Determination of Attributes of Events: As a follow-on step, each event, i.e., the corresponding node in the ESG, is represented as a multidimensional data point using the values of all nine attributes as defined in the previous section. Estimating by means of the ESG and $\overline{ESG}$, the values of attributes for all events are determined and the data set is constructed (Table 1).

For the data set gained from the case study (Figure 2, 3), the optimal number c of the groups is determined to be 5 which leads to the groups $S_k, k=1,...,5$. Importance degrees (*ImpD(S$_k$)*) of obtained groups are determined by comparing the length of their weight vectors (ℓ) and all *ImpD(S$_k$)* values that are presented in Table 2.

Indirect Determination of Preference Degrees: As mentioned in the previous section, the preference degree of the CESs is determined indirectly by (15) that depends on the importance of events (see (14)) and frequency of event(s) within CES. The ranking of the CESs is represented in Table 3.



Roomtypes

Name	Roomtype	Min. # pers.	Min. # Adults	Max. # pers.	Max. # Adults		
Apartment	Apartment	1	1	3	3		In use
Double room	Double room	1	1	2	2		In use
Single room	Single room	1	1	1	1		In use
Family room	Family room	1	1	6	6		In use
Suite	Suite	2	2	4	3		In use

Add roomtype to list

Roomtype: Single room

For min. --- thereof min. --- to adult price

For max. --- thereof max. --- adults

Room features :

☐ Air condition
☐ opened window
☐ noise protection window
☐ Desk
☐ Internet
☐ Safe
☐ Minibar
☐ Kitchenette
☐ Hair-dryer
☐ TV
☐ Phone at desk
☐ Phone at bed
☐ Phone in bath
☐ Bath tub
☐ Shower
☐ Extra long bed
☐ Separated beds
☐ Single bed
☐ Double bed
☐ Separated bedroom

Name: (i.e. "Deluxe-SR", "Standard-SR")

add

Figure 2. Room definition/reservation process in ISELTA

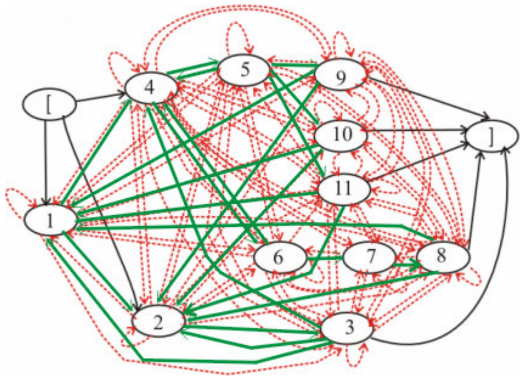


Figure 3. Completed ESG for room definition/selection (solid arcs: event pairs (EP); dashed arcs: faulty EP (FEP)

Legend of the Figure 3:

1: Click on "Starting"
2: Click on "Registering"
3: Registering carried out
4: Click on "log in"
5: Logged in
6: Click on "Password forgotten"
7: Password forgotten
8: Click on "Request"
9: Indicate service(s) offered
10: Indicate administrator
11: Indicate agent

Exercising the test cases (CESs, or walks) in this order ensure that the most important tests will be carried out first. Moreover, the achieved ranking of CESs complies with the tester's view. Thus, an ordering of the complete set of CESs (walks) is determined using the test suite generated by the test process, i.e., we now have a ranking of test cases to make the decision which test cases are to be primarily tested. Undesirable events can be handled in a similar way; therefore, we skip the construction of ranking of the FCES.

Table 1. Data set of events

Event no	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9
1	1	8	10	39	0	6	4	0,1860	8
2	1	8	10	40	0	7	6	0,1519	9
3	2	5	10	41	6	7	-3	0,1519	11
4	1	7	10	35	0	14	3	0,2469	7
5	2	5	10	29	0	10	-3	0,2112	9
6	2	3	10	36	0	4	-1	0,2199	7
7	3	2	10	37	0	3	0	0,1218	9
8	4	4	10	38	0	3	-2	0,1218	10
9	3	4	10	17	30	3	-2	0,1494	9
10	3	4	10	22	0	3	-2	0,0698	9
11	3	4	10	30	0	3	-2	0,1911	9

Construction of the Groups of Events

Table 2. Obtained groups of events

Groups ((3)and (13)	Events that belonging this group	Length of weight vectors (ℓ)	Importance Degree ImpD(S_k)
S_1	9	2,07	2
S_2	3,6,7,8,11	2,04	3
S_3	1,2,4	1,97	4
S_4	5	2,25	1
S_5	10	1,73	5

Table 3. Ranking of CESs (walks)

Pref-Deg.	**Pref (CES_q)(15)**	CESs	CESs (walks)
1	**126**	CES_1	[4 5 4 5 9 1 4 5 10 1 4 5 11 1 4 5 9 2 3 4 5 10 2 3 2 3 1 4 5 11 2 3 4 6 4 6 7 8 1 2 3]
2	**29**	CES_3	[2 3 4 6 7 8 2 3 4 5 10]
3	**13**	CES_2	[1 4 5 9]
4	**11**	CES_5	[4 6 7 8]
5	**10**	CES_4	[4 5 11]

5. Conclusions and Future Work

The model-based, coverage-and specification-oriented approach described in the previous sections provides a novel and effective algorithm for ordering the test cases according to their degree of preference. Such degrees are determined indirectly through the use of the events specified by several attributes, and not a single one. This is an important issue and consequently, the approach introduced radically differs from the existing ones.

The relevant attributes are visualized by means of a graphical representation (here, given as a set of ESGs). The events (nodes of ESG) are classified using unsupervised neural network clustering. The approach is useful when an ordering of the tests due to restricted budget and time is required. Run-time complexity of this approach is of $o(n^2)$, assuming that the number of events (n) greater than the number of attributes (p), otherwise it is $o(p^2)$.

We plan to apply our prioritization approach to a more general class of testing problems, e.g., to multiple-metrics-based testing where a family of software measures is used to generate tests [17]. Generally speaking, the approach can be applied to prioritize the testing process if the SUT is modeled by a graph the nodes of which represent events or sub-systems of various granularities (modules and functions, or objects, methods, classes, etc.).

References

[1] Binder, R.V.: *Testing Object-Oriented Systems*. Addison-Wesley, 2000.

[2] Bryce, R.C., Colbourn, Ch.C. Prioritized Interaction Testing for Pair-wise Coverage with Seeding and Constraints, *Information and Software Technolog* y, 48, 2006, pp. 960–970.

[3] Elbaum, S., Malishevsky, A., Rothermel, G.: Test Case Prioritization: A Family of Empirical Studies. *IEEE Transactions on Software Engineering*, 28(2), 2002, pp. 182-191.

[4] Cohen, D.M., Dalal, S.R., Freedman, M.L., and Patton, G.C.: The AETG System: An Approach to Testing Based on Combinatorial Design. *IEEE Trans. Software Engineering*, 23(7), 1997, pp. 437-44.

[5] Belli, F.: Finite-State Testing and analysis of Graphical User Interfaces, *Proc. 12th Int'l. Symp. Softw. Reliability Eng. (ISSRE'01)*, 2001, pp. 43-43.

[6] Belli, F., Budnik, Ch. J., White, L.: Event-Based Modeling, Analysis and Testing of User Interactions – Approach and Case Study. *J. Software Testing, Verification & Reliability,* John Wiley & Sons, 16(1), 2006, pp. 3-32.

[7] F. Belli, F., C. J Budnik: Test Minimization for Human-Computer Interaction, *J. Applied Intelligence* 7(2), Springer, 2007, pp. 161-174.

[8] Edmonds, J., Johnson, E.L.: Matching: Euler Tours and the Chinese Postman, *Math. Programming*, 1973, pp. 88-124.

[9] Maeda, M., Miyajima, H., Marashima, S.: An adaptive Learning and Self-Deleting Neural Network for Vector Quantization. *IEICE Trans. Fundamentals,*1996, pp.1886-1893.

[10] Maeda, M., Miyajim, H.: Competitive Learning Algorithm Founded on Adaptivity and Sensitivity Deletion Method. *IEICE Trans. Fundamentals*, 2000, pp. 2770-2774.

[11] Rummelhart, D.E.: Zipser, D.: Competitive Learning. *J. Cognitive Science*, 1985, pp. 75-112.

[12] Eminov, M., Gökçe, N.: Neural Network Clustering Using Competitive Learning Algorithm, *Proc. TAINN 2005*, 2005, pp. 161-168.

[13] Eminov, M.E.: Fuzzy c-Means Based Adaptive Neural Network Clustering. *Proc. TAINN-2003, Int. J. Computational Intelligence*, 2003, pp. 338-343.

[14] Kim, D.J., Park, Y.W. and Park, D.J.: A Novel Validity Index for Clusters, I*EICE Trans. Inf & System*, 2001, pp. 282-285.

[15] Belli, F., Budnik, Ch.J., Linschulte, M., and Schieferdecker, I.: Testing Web-Based Systems with Structured, Graphic Models – Comparison through a Case Study (in German), to appear in *Proc. Annual German National Conf. for Informatics (GI-Jahrestagung)*, 2006.

[16] Gerhart, S., Goodenough, J.B.: Toward a Theory of Test Data Selection, *IEEE Trans. On Softw. Eng.* , 1975, pp.156-173.

[17] Neate, B., Warwick, I., Churcher, N.: CodeRank: A New Family of Software Metrics, *Proc. Australian Software Engineering Conference – ASWEC 2006*, IEEE Comp. Press, 2006, pp. 369-377.

[18] Jeffrey, D., Gu, N.: Test Case Prioritization Using Relevant Slices, ICSE 2002.

[19] Kim, J.-M., Porter, A.: A History-Based Test Prioritization Technique for Regression Testing in Resource Constrained Environments, COMPSAC 2006