Markov-Based Anomaly Correction in Embedded Systems

Roghayeh Mojarad and Hamid R. Zarandi

***Abstract*—In this paper, an anomaly correction method is proposed which is based on Markov anomaly detection method. The proposed method employs the probability of transitions between events to evaluate the behavior of a system. This method consists of three steps: 1) Construction of transition matrix by probability of transitions between events and list of known events are generated in training phase; 2) Detection of anomaly based on Markov detection method will be done. In test data when the probability of transition previous event to current event does not reach a predefined threshold, an anomaly is detected. Threshold is determined based on constructed transition matrix in step 1; 3) Check the defined constraints for each anomalous event to find source of anomaly and the suitable way to correct the anomalous event. Next, an event with the highest compliance with the constraints is selected. Evaluation of the proposed method is done using a total of 7000 data sets. The operational scope of corrector and the number of injected anomalies varied between 3 and 5, 1 and 7, respectively. The simulation experiments have been done to measure the correction coverage rate which is between 53.5% and 97.2% with average of 77.66%. For evaluation of hardware consumptions of the proposed method, this method is implemented by VHDL. Power, area and time consumptions are on average 87.43μw, 415.48μm^2, and 4.12ns, respectively.**

***Index Terms*—Anomaly, anomaly detection, anomaly correction, correction coverage, embedded systems, fault, operational scope.**

I. Introduction

Digital systems are inseparable parts of modern life style, one category of them are embedded systems which are embodied in other ones for controlling and management purposes, such as traffic controllers [1], [2], high speed network switches, airplanes and spacecraft controllers and medical devices [3]. The characteristic of safety is vital in embedded systems, because any anomaly or fault can cause severe financial and physical damages. Therefore, it is essential to improve fault-tolerance in them.

Sensors are one of important component in embedded systems [4]. The data produced by sensors are known as sensor data or sensor data set. The faults in the output of the sensors are known as anomaly. Anomaly detection and correction are one of the methods to tolerate faults [5].

Anomaly detection methods are a kind of fault detection methods such that fault detection can be either explicit or implicit. Explicit fault detection is usually based on pattern recognition such as a sign is detected which is directly linked to a specific fault [6], [7]. On the other hand, in the implicit fault detection; there are some indirect indicators such as an anomaly. Some faults do not have any explicit sign and are detected by unusual behavior. As an example, an Ethernet broadcast storm that is marked indirectly by high packet traffic on a network abnormally [8]. Packet traffic is measured by sensors. There are many sensors that can measure the state of a network, process or system.

Anomaly detection and correction methods respectively try to distinguish and amend abnormal events in sensor data. As importance of data increases, the significance of these methods enhances. For example, anomaly in a credit card transaction could be a sign of thievery [9], or an anomaly in MRI (Magnetic Resonance Imaging) could falsely show a tumor which result in a wrong decision and put one's life in jeopardy [10], again there is the same problem in controlling hazardous chemicals [11].

Sensor data can be either numerical or categorical [8]. Numerical data are continuous, scalable and have a unique zero; and mathematical operation can be performed on them. On the other hand, categorical data are discrete and there is no order and mathematical operation among them, as an apple is not twice of an orange [8]. Moreover, as the computational power increases, more sensors deliver categorical data [8]. Anomaly detection and correction are more challenging within categorical data than numerical data, as it does not have ability to perform statistical analysis on data. Correcting the anomalies are important to hold integrity, reliability, safety, security and in general dependability of the system.

To the best of our knowledge, there is not any anomaly correction method for categorical data in the scope of embedded systems. Categorical data are in form of sequence of symbols. Anomaly in categorical data can be either an unknown symbol or an unknown or unexpected sequence of symbols [8].

Although some works had been done on anomaly detection methods, now ever not much effort was done in the anomaly correction field. In this paper, an anomaly correction method is proposed to improve fault tolerance in embedded systems. This method is based on Markov model and consists of three steps: 1) Training in design time, 2) Detecting of anomaly in run time, 3) Correcting of anomaly in run time.

Training in design time: Generate transition matrix and list of known sequences.

Detecting of anomaly in run time: Anomaly is detected based on Markov detection method by using the transition matrix.

Correcting of anomaly in run time: Checking the defined constraints for finding the source of the anomaly and considering three states of *unique substitution*, *multiple substitutions* and *deletion* which are explained in the following, for correcting of the anomaly.

There are some constraints for each state that they must be checked and if all constraints for a state are met, that state will be selected as a candidate for corrected state.

Manuscript received November 22, 2014; revised May 5, 2015.

Roghayeh Mojarad and Hamid Reza Zarandi are with the CE & IT Department, Amirkabir University of Technology, Tehran, Iran (e-mail: {roghaye_mojarrad, h_zarand}@aut.ac.ir).

The corrector can see and operate within a window which is known as corrector window. The corrector uses of this length for checking constraints because it can only see the data are in this window. Number of constraints for each state is related to the length of corrector window. The correction is done by analyzing the constraints and using a similarity function.

Threshold is defined as the maximum distance from the normal data without being detected as anomaly. This value is set regarding to the training data sets.

Major metrics for evaluating the proposed methods are correction coverage, power, area and time consumptions. The evaluation was done using a total of 7000 test data sets for different number of injected anomalies. The range of injected anomaly varied from one to seven. The average coverage of results that is 77.66% shows the effectiveness of the proposed method on improving fault-tolerance metrics of the embedded systems.

The structure of the rest of paper is as follows: the Section II presents a literature review followed by background information in Section III. Section IV is depicted to elaborate the proposed method and the Section V discusses the evaluation method with its results and the paper is wrapped up with conclusion in Section VI.

II. Related Work

Although many researches were done on anomaly detection [12]-[14]; however they lack the ability to detect anomaly in categorical data. DWC (Duplication with Comparison) and TMR (Triple Modular Redundancy) are of these types; also if they have the ability to recognize anomaly in categorical data they are applicable for special sensors like eyes-detection sensors [15]. Among effective methods on this field, Markov [8], Stide [8], Probability-based [16] and buffer-based [16] methods are noticeable. The proposed method is a method for anomaly correction in categorical data. The details of the mentioned works come as follow:

Markov detection method includes three phases: 1) Training 2) Threshold setting 3) Testing. The probabilities of transitions between states are calculated and transition matrix is generated. Next, a suitable threshold is defined based on the results of training phase. Finally, probability of each transition which exists in the testing data is calculated by transition matrix and it compares with threshold [8]. If complementary of probability of the transition is more than threshold value, the Markov detection method detects anomaly.

Stide works using three above steps as well: Training data sets break into overlapping sequences with the length of *N*. The value of *N* is calculated within practice [17]. Duplications are removed from these sequences; afterwards they are stored in a database.

Testing data set breaks down into overlapping sequences with the similar length of *N*, too. These sequences are analyzed sequentially and are individually searched in the database, if the sequence exists in the database the score is set to zero and one if otherwise. The frame with maximum number of ones shows the placements of the anomaly [17].

In probability-based method, the relative distances of symbols are used. This method consists of above three steps as well: Probability matrix is formed by the information of the data sets. Rows of the matrix are all possible permutations of pair of symbols in the training data sets and its columns are possible distances of two symbols in one pair. The entry in i^{th} column and j^{th} row shows the probability of occurrence of the j^{th} pair with the distance of *i* between each symbol.

All information about distances between symbols is extracted and with an operation called probability multiplication the normality of the sequence is computed; if normality were less than the threshold, the sequence will be marked as anomaly [16].

Buffer-based method is similar to Stide method, however with this difference the rare sequences in training phase are dropped off the database. The testing phase is the same, as each dataset breaks into overlapping sequences with length of *N*; and if they are not in the database, the score would be set to one and zero if otherwise. If the number of unsuccessful searches is higher than a given threshold, it will be marked as anomaly [16].

III. Background Knowledge

The definitions of several keywords of the context are as follows [8]:

- Surprise factor: determines how likely the event does not happen.
- Categorical data: some kind of data that is not in mathematical bonds and is classified.
- Training data set: a data set used for training phases and teaching the system about normal data.
- Testing data set: a data set used for testing the system and analyzing its performance.
- Normal data set: a data set which includes normal data which are obtained from training data set.

A. Anomalous Event

Anomalous event in the output of the sensor is either these two types:

- Event consists of those symbols which do not exist in normal data
- Event consists of symbols or permutation of symbol which is different from one in normal data

An anomaly can be divided in three groups of unknown symbol, unknown sequence and rare sequence [8]:

Unknown symbol: These anomalies occur when a symbol comes in testing data set but does not exist in normal data set. For example, if in training phase, the symbols *A*, *B*, *C* and *D* are learned, the symbol *G* in the testing data set would be considered as anomaly.

Unknown sequence: A sequence of symbols different from sequences in the training phase. *Unknown sequence* is a more general type of *unknown symbols*, a sequence that consists of known symbols but still be unknown, and that is when the permutation of the symbols is different from them in normal data. For example, if normal data set consist *ABC*, *BCD*, *CDE* the testing sequence *DCB* is unknown; it includes known symbols but these symbols did not occur in normal data in that specific order.

Rare sequences: *Rare sequences* are those that might occur in the training phase but they are not common; that is their occurrence is low. So, they do not exist in the normal data.

Their occurrence is compared with defined threshold to be included in the list of known sequences. For example, if training dataset includes sequences *ABC*, *BCD*, *CDF* and *DFH*; and the percentage of occurrence of the first and last two are 45% and 5%; if the threshold is 9% then *CDF* and *DFH* would are two rare sequences and they although occurring in the training data set would be marked as anomaly.

B. Anomaly Detection

Anomaly detection has a definite operational scope that is equal to the length of its window. Window size of anomaly detection determines the length which it can see from data stream. Anomaly detection is able to analyze only within this scope. One of important way for evaluating detection and correction method is to inject anomalies into normal data.

Different coverage of anomaly correction and detection is related to window size and number of injected anomalies. Because window size is the length that anomaly detector and corrector view, this size affects the scope of operation of the anomaly detector and corrector.

C. Operational Scope of Anomaly Detection

In an anomaly detection system which operates as sliding window, the operational scope is equal to the size of the window. As the number of anomalies is not always the same as the size of window, additionally this difference can affect its performance; therefore the size of window is important to be wisely adjusted. Length of window is set in respect of constraints of surroundings and its application. The operational scope of a detection or correction system can be one of the following types [8]:

Whole Scope: The operational scope is equal to total number of anomalies such that the detector or corrector has only the ability to view all of the anomalies.

Internal Scope: The number of anomalies is greater than the operational scope.

Encompassing Scope: The operational scope is greater than the number of anomalies.

Boundary Scope: The detector sees a part of anomaly as well as parts of background data.

Background Scope: The detector sees only background data without any anomalous data.

Fig. 1 shows operational scope of an anomaly detector or corrector in different modes.

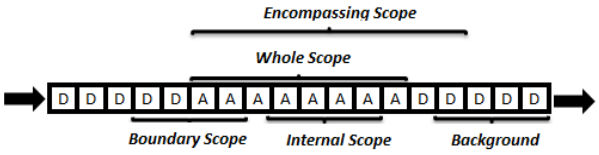

Fig. 1. Operational scope of anomaly detector [8].

D. Markov-Based Anomaly Detection [8]

Markov based anomaly detector distinguishes normal or anomaly in a flow of data. In this method, probability of transition between two states is modeled by using of transition matrix. Key attribute of Markov model is that next state is only depending on the current state. For example, aerology is the same and next state only depends on current state [8], [18]. This attribute can be shown formally as Equation (1), in which X_{t+1} and X_t are the next and current X state while and X_0 are old state and initial state, respectively [8]:

$$P(X_{t+1} = x_{(t+1)} \mid X_t = x_t, X_{t-1} = x_{t-1}, ..., X_0 = x_0) = P(X_{t+1} = x_{t+1} \mid X_t = x_t) \quad (1)$$

Markov anomaly detector operates in three phases:

1) Training phase: In which transition matrix is generated.
2) Setting threshold value.
3) Testing phase: Evaluating the system by the testing data sets.

Transition matrix is filled by the probability of transition between each two states which require two counters for each transition, counting the specific transition and total number of transition from its source. The required computation for finding the probability of sample transition $A \rightarrow B$ is shown in (2), in which $P(A, B)$ is the probability of transition, $F(A, B)$ is the number of occurrence of this transition in training data sets, and $F(A)$ is the number of transitions from the state A.

$$P(A,B) = \frac{F(A,B)}{F(A)} \quad (2)$$

Fig. 2 shows a Markov model with four states. Each state is depicted with circles with its name on it and arcs illustrate the transitions with their occurrences on them. Fig. 3 shows the corresponding transition matrix of the previous model.

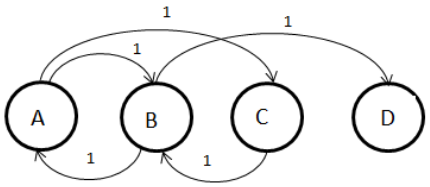

Fig. 2. Markov model with four states.

	A	B	C	D
A	0	0.5	0.5	0
B	0.5	0	0	0.5
C	0	1	0	0
D	0	0	0	0

Fig. 3. Transition matrix of Markov model presented in Fig. 1.

The next phase of anomaly detector is setting the threshold value which is done regarding training data set.

The third phase is evaluating using testing data sets. For this purpose, probability of each transition is extracted from the transition matrix; its complementary value would be the surprise factor of that transition. Equation (3) shows a trend for providing of surprise factor of each transition.

$$Suprise\ Factor = 1 - P(State_{Current}, State_{Next}) \quad (3)$$

If the surprise factor exceeds the threshold value, the transition is detected as anomaly and normal if otherwise. The next section provides details of the proposed method.

IV. The Proposed Markov Correction Method

The proposed method consists of three phases: 1) Training

in design time, 2) Detection of anomaly in run time and 3) Correction of anomaly in run time; the first two phases are similar to ones in Markov detection system and had been discussed in previous section; the third phase is the part which is used to correct anomaly.

During analyzing a sequence, when it is detected as anomaly in detection phase; it is to be corrected in the third phase, the first of all, corrector must determine entities which cause anomaly. In order to achieve this goal, each entity of the anomalous sequence is individually analyzed and different constraints are made. The number of these constraints is related to the size of corrector window. Smaller windows show the effectiveness of the system because it can obtain the desired goal with the less hardware consumptions such as power, area and time rather than larger ones which include more information and need high hardware consumptions. In this proposed corrector, the size of the window varied from 3 to 5. Number of constraints for window size of 3 and 4 would be 12 and 20, respectively. In order to clarify this note, one example is provided, twelve condition of a system with window size of three are formed in the following manner. If training sequence is *ABCDEFGHI* and testing sequence is *ABCDEZGHI*; then the order of analyzing the testing data would be as follows:

TABLE I: THE ORDER OF ANALYZING THE TESTING DATA

Order	Sequence
1	ABC
2	BCD
3	CDE
4	DEZ
5	EZG
6	ZGH
7	GHI

	ABC	BCD	CDE	DEF	EFG	FGH	GHI
ABC	0	1	0	0	0	0	0
BCD	0	0	1	0	0	0	0
CDE	0	0	0	1	0	0	0
DEF	0	0	0	0	1	0	0
EFG	0	0	0	0	0	1	0
FGH	0	0	0	0	0	0	1
GHI	0	0	0	0	0	0	0

Fig. 4. Transition matrix for mentioned example.

In the training phase, the transition matrix is constructed. Fig. 4 presents transition matrix of this example. If threshold value is equal to 0.9, the detector performs following trend.

$$P(ABC, BCD) = 1,$$
$$Suprise\ Factor = 1\text{-}P(ABC, BCD) = 0 < 0.9 \checkmark$$
$$P(BCD, CDE) = 1,$$
$$Suprise\ Factor = 1\text{-}P(BCD, CDE) = 0 < 0.9 \checkmark$$
$$P(CDE, DEZ) = 0,$$
$$Suprise\ Factor = 1\text{-}P(CDE, DEZ) = 1 > 0.9 \times$$

Therefore, the first two transitions are normal, subsequently the third transition is not normal and the detector marks it as anomaly.

The twelve constraints for correcting the anomalous sequence DEZ is grouped as in three categories. The first, the anomalous symbol must be detected. For each symbol of sequence, four constraints should be met as shown in Table II.

The question mark depicts unknown place holder and arrow shows the transition. That is, *ABC|BC*? means the corrector searches from possible transitions for a transition from *ABC* to a state with B and C as its two first entities.

Each symbol of the sequence can only be corrected if all of its corresponding constraints are met; there is a symbol that can be placed as the question mark such that all the transitions are normal.

Correction mechanism is either substitution or deletion of the anomaly. To operate it, all options are analyzed and the best option is selected. Regarding to the options, followings might occur:

- *Unique substitution*: If only one option meets the constraints. This kind of the substitution is done rather straightforward.
- *Multiple substitutions*: If many options meet the constraints; for each option the occurrence of corresponding transitions are summed; subsequently the option with the nearer occurrence to random number is selected. Therefore, this selection is based on weight of occurrence of events. This strategy is selected because the method does not always select one state with high frequency.
- *Deletion*: If no substitution is possible, the system checks whether removing symbols would solve the problem. In other words, this state checks whether removing a number of symbols makes the sequence or transition normal.

TABLE II: TWELVE POSSIBLE CONSTRAINTS FOR STATE OF SUBSTATION OF THE EXAMPLE

Order	Sequence
1st symbol	*ABC →BC?* *BC?→C?E* *C?E→?EZ* *?EZ→EZG*
2nd symbol	*BCD→CD?* *CD?→D?Z* *D?Z→?ZG* *?ZG→ZGH*
3rd symbol	*CDE→DE?* *DE?→E?G* *E?G→?GH* *?GH→GHI*

[?] The question mark depicts unknown place holder

For the correction of the mentioned example, the first symbol cannot be selected as anomalous symbol because all constraints which need for selection of this symbol are not met. In fact, there is no symbol to be replaced with the question mark or to be deleted to have all the transitions as normal. Table III shows which constraints do not meet the required constraints for selecting of first symbol as anomalous symbol. Based on Table III, two of the transitions cannot be found in transition matrix. Therefore, the first

symbol is not an anomalous symbol and does not need to correct.

TABLE III: STATES OF CONSTRAINTS MEETING FOR FIRST SYMBOL OF ANOMALOUS SEQUENCE

Meet of First Symbol's Constraints	
ABC→BC?	
BC?→C?E	
C?E→?EZ	✗
?EZ→EZG	✗

The second symbol cannot be anomaly; as there is no option for changing, and three of the transition does not exist in the transition matrix. Table IV shows those transitions outside the transition matrix.

TABLE IV: STATES OF CONSTRAINTS MEETING FOR SECOND SYMBOL OF ANOMALOUS SEQUENCE

Meet of Second Symbol's Constraints	
BCD→CD?	
CD?→D?Z	✗
D?Z→?ZG	✗
?ZG→ZGH	✗

The third symbol of the anomalous sequence can be corrected because all of constraints for this symbol are met. So, the corrector looks for a symbol by substituting the anomalous symbol. In this easy example, the corrector substitutes the question mark with symbol *F* because all the corresponding transitions exist with this symbol in transition matrix and there is not another symbol with these constraints.

To assure and prevent false correction, after the correction; the detection starts over for the corrected sequence and the window does not slide forward for the next sequence.

If the system could not find a suitable substitution, each symbol of anomalous sequence is checked whether it can be removed. To clarify this note, Table V presents the constraints of this case for mentioned example. This is obvious that this anomaly is not created because of spontaneous data reporting of sensor which the deletion case is useful for it. For this reason, any constraints of this case are not met.

TABLE V: NINE POSSIBLE CONSTRAINTS FOR STATE OF DELETION OF THE EXAMPLE

Order	Sequence
1st symbol	*ABC → BCE* *BCE→CEZ* *CEZ→EZG*
2nd symbol	*BCD→CDZ* *CDZ→DZG* *DZG→ZGH*
3rd symbol	*CDE→DEG* *DEG→EGH* *EGH→GHI*

The correction phase is not done fully yet; as these constraints are suitable for one anomaly in the corrector's window size. In other words, anomalies with length of one are in the length of window of corrector that only one question mark is used in it. Therefore, if none of them can be removed then a similarity function is used. Similarity function is designed for correcting multiple anomalies. The most similar known sequence to anomalous sequence is extracted; such that the transitions from or to that sequence are detected as normal. In the case of existence of a unique option with highest similarity, that sequence with be substituted with the anomalous sequence; otherwise, the system tries to find the most similar sequence regarding the next and previous sequences. Afterwards the correction phase is completed and correction is performed.

This method uses a basic similarity function in itself which is shown in the (4). It is assumed that X , Y are two sequences such that $X=\{X_0,X_1,\ldots,X_N\}$, $Y=\{Y_0,Y_1,\ldots,Y_N\}$ and this equation measures the similarity between these two sequences.

$$Sim(X,Y)=\sum_{i=0}^{i=N}S(X_i,Y_i)$$
$$S(X_i,Y_i)=\begin{cases}0 & If\ X_i=Y_i\\ 1 & Otherwise\end{cases} \quad (4)$$

Correction with the highest similarity value is the most suitable correction because it matches to the normal event in higher level rather than others. In other words, it meets the most constraints.

Fig. 5 illustrates the pseudo code algorithm of the correction procedure:

In the stage of training, a transition matrix is generated which consists of probability of transition between known sequences. Then, a threshold value based on train data is set.

In the testing stage, the first, the surprise factor is measured; if this number exceeds the threshold value, an anomaly is detected; while, in otherwise case, no anomaly is detected and window of corrector will moves.

Notation:

W: window size for corrector

Training-Stage-Algorithm (*training data*, *W*)

1. *Construct a transition matrix by probability of transition between events.*
2. *Set the threshold value to a given value.*

Testing-Stage-Algorithm(*testing data*, *transition matrix*, *threshold*, *W*)

1. **IF** (1-(Probability of transition between two events) > threshold) **THEN**
2. *Detect-Anomaly=1;*
3. *Replacement=1;*
4. **ELSE**
5. *Detect-Anomaly=0;*
6. *Move DW forward;*
7. **END IF**
8. **IF** (Detect-Anomaly=1) ***THEN***
9. *Find the anomaly symbol(input anomalous event, input transition matrix, output anomaly symbol, output TorF)*
10. **IF** (TorF = 1) ***THEN***
11. *Candidates-replacement=results of search transitions which are related with anomalous sequence with unknown symbol in the transition matrix*
12. **IF** (Candidates-replacement = 1) ***THEN***
13. *Do replacement*
14. **ELSE IF** (Candidates-replacement > 1) ***THEN***

15. *Select one of them based on weighted probability.*
16. **ELSE**
17. *Replacement=0;*
18. **END IF**
19. **ELSE IF (TorF=0 or Replacement=0)** ***THEN***
20. *YorN = Existence of the transition with new sequence in the transition matrix after deleting each symbol of anomalous sequence*
21. **IF** (*YorN*=1) ***THEN***
22. *Candidates-deletion = results of search transition of anomaly sequence with unknown symbol in the transition matrix*
23. **IF** (Candidates-deletion=1) ***THEN***
24. *Deletion is done*
25. **ELSE IF** (Candidates-deletion>1) ***THEN***
26. *Selects one of them based on weighted probability.*
27. **ELSE**
28. *Use similarity function*
29. **END IF**
30. **ELSE**
31. *Use similarity function*
32. **END IF**
33. **ELSE**
34. *Use similarity function*
35. **ELSE**
36. *Move DW forward*

Fig. 5. The proposed anomaly correction algorithm.

In the state which an anomaly is detected, in the line 7, the function of *Find the anomaly symbol* finds source of anomaly based on checking of constraints; anomalous symbol is source of anomaly in anomalous sequence and output of *TorF* is shown the number of alternatives of sources of anomaly in that sequence. If this number is equal to one, the unique substitution is occurred and if this number is more than one, the multiple substitutions are occurred and the method selects one of them based on weighted probability that is provided by transition matrix. In the case which the number of alternatives of source of anomaly is not equal or more than one, the deletion state should be checked. Also, in the case that output of *TorF* is zero which shows the method cannot find the source of anomaly, the deletion state should be considered. The value of *YorN* shows if by deleting of a symbol, the anomalous sequence can be corrected or not; if this value is one, the state of deletion does and otherwise the similarity function selects is used to correct the anomalous sequence.

V. Application

This method is designed for the embedded systems and it can be placed between sensor part and controller part in embedded systems. This place is shown in Fig. 6.

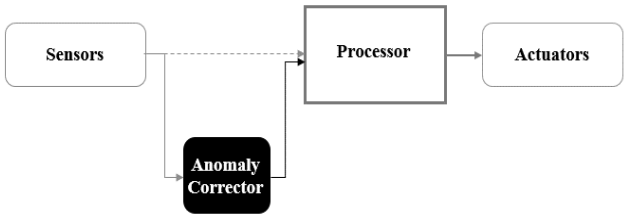

Fig. 6. Location of the proposed method in the embedded systems.

As mentioned previously, the proposed method in order to detect and correct the i^{th} event, it investigates all the events in the range of in which w illustrates the length of window. Therefore, this method has to wait for coming events and that causes some delay in the input of the controller part of embedded systems.

A real-time system is one that must process information and produce a response within a specified time. Subsequently, this method can be known as real-time method in correction methods.

To be clear, suppose a system of network traffic reporting which reports each 1ms; if an anomaly is occurred in its sensor, the proposed method corrects that anomaly before reporting false information. This action could prevent the loss of life and property.

VI. Experimental Study

The normal data are provided from the University of New Mexico's website [19]. They are synthetic data for *sendmail* were collected at UNM on Sun SPARC stations running unpatched SunOS 4.1.1 and 4.1.4 with the included *sendmail* [19].

The method is evaluated in three phases:

1) To construct of testing data, a program is implemented. The first of this program, it randomly selects a part of normal data as background data for injecting of anomaly to it. The background data saves as correct data. Then by considering [20], the program randomly selects place of anomaly injection and injects or replaces one or more than one abnormal symbol to the background data to make testing data. Number of the injected anomalies varied from one to seven and for each of number of anomalies, 1000 testing data sets are constructed which forms 7000 testing data sets.
2) Implementation is done by VHDL to measure the correction coverage which is key factor for enhancing the fault tolerance in embedded systems. In this phase, the proposed method runs for 7000 testing data set and create corrected data for each testing data set. In order to hardware analysis, including estimating power, area and time consumption, the system is synthesized by Synopsis Design Compiler with library of the 45nm Nangate opencell [21], [22].
3) The third phase of evaluation is checking the corrected data that are generated by the proposed method and comparing them with the original data that do not have any anomaly and were created in the second phase of evaluation. In order to calculate the correction coverage of the system, another program is implemented. It is used to compare of two data; one of them is output data of proposed correction method and another one is original data without any anomaly injection. If two data are same with each other, correction coverage increases unit because the number of produced files for each number of anomaly injection is 1000 and correction coverage is measured for each state.

A. Correction Coverage Analysis

The results of evaluation of the correction coverage are shown in Fig. 6.

There is a logical reason for values of correction coverage which are less than 100% that it is using of random functions in case of *multiple substitutions*. Because, if there are many

options to select, regarding the weights of states and random number; an option would be selected. As this selection is not deterministic, therefore the correction coverage is not ideal.

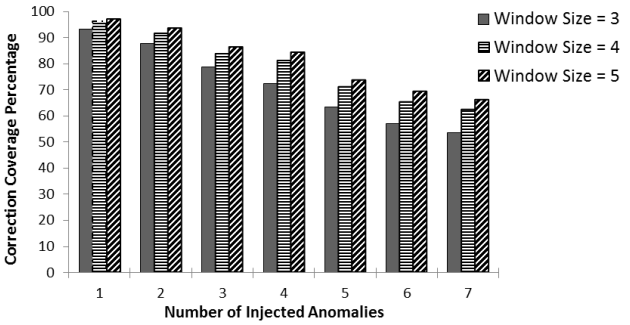

Fig. 6. Correction coverage with various window sizes.

Fig. 6 is shown as the number of anomalies increases, the correction coverage decreases; because in that case, finding of the suitable options are more difficult and similarity function operates with lower accuracy. The results of the correction coverage with window size of three, four and five shows the similar manner. The size of window directly affects the coverage as more information is analyzed and better decision can be made. It is a fact, as corrector's window size increases, the correction coverage enhances, too.

B. Power, Area and Time Consumption Analysis

The results of synthesis and the estimation of power, area and time consumption are shown in Table VI.

As the window size increases, the power, area and time consumption increase too; in that case finding suitable correction need more time and power than smaller window size because it needs to check more constraints. Furthermore, as larger array is needed to save the options in larger window size and the area consumption is higher.

In this approach, there is a trade-off between precision and consumption; that is, the larger window provides higher correction coverage but consumes more power, area and time.

TABLE VI: HARDWARE CONSUMPTION OF MARKOV-BASED ANOMALY CORRECTION WITH VARIOUS WINDOW SIZES

Window Size	Area (μm^2)	Dynamic Power (μw)	Leakage Power (μw)	Time (ns)
3	249.64	62.85	1.87	3.98
4	415.48	87.43	1.91	4.12
5	10787.73	814.89	96.21	18.80

VII. CONCLUSIONS

The goal of this paper is to decrease anomalies in embedded systems and consequently increasing the fault tolerance metrics in these systems. This paper proposes an anomaly correction method in categorical data.

All of previous work only detect anomaly, but proposed method in addition of detection of anomaly, can correct it. Table VII is shown the differences of mentioned different methods. The assign "+", "-" are shown to have the determined ability or not; as an example, the Markov method has ability to detect anomalies but it does not have the ability for anomaly correction. Also, this method has the high consumption. By analyzing of this Table, it can be concluded that the proposed method can provide anomaly detection and correction with modest consumption.

TABLE VII: ANALYZING OF MENTIONED DIFFERENT METHODS

Methods	Detection	Correction	Consumption
Markov [9]	+	-	++
Stide [9]	+	-	+
Probability-based [14]	+	-	+
Buffer-based [14]	+	-	+
The proposed method	+	+	++

Although Markov-based anomaly correction method have noticeable hardware consumption but its correction coverage is significant, too. Each of two other detection methods have the less amount of detection coverage than Markov detection method; there for, applying of correcting method to them is not well as applying it to Markov detection method.

This paper shows anomaly correction methods can be built based on anomaly detection ones. For that all possible solutions will be tested against detection criteria.

The results of the evaluation shows that this method can be used for many applications such as medical equipment for reporting vital sign, as it can prevent averagely 77.66% of catastrophic failures of the system with window size of three, four and five and different number of anomaly from one to seven. It is considerable note that this method was evaluated with small window size and the correction coverage was significantly high, and that shows the effectiveness of the proposed method.

Although, it is noticeable that the proposed method does not work in real time, because this method waits for coming events in order to correct current event. This latency is directly related to the length of window and consequently the precision. Therefore, when a system does not require high precision, the method can eliminate some constraints that related to coming events and works with some constraints that depend on the previous events.

REFERENCES

[1] A. V. Lovato, E. Araujo, and J. D. S. da Silva, "Fuzzy decision in airplane speed control," in *Proc. IEEE International Conference on Fuzzy Systems*, 2006, pp. 1578–1583.

[2] A. Patcha and J. M. Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends," *Computer Networks*, vol. 51, no. 12, pp. 3448–3470, 2007.

[3] C. Spence, L. Parra, and P. Sajda, "Detection, synthesis and compression in mammographic image analysis with a hierarchical image probability model," in *Proc. IEEE Workshop on Mathematical Methods in Biomedical Image Analysis*, 2001, pp. 3–10.

[4] N. Imran, L. Jooheung, K. Youngju, L. Mingjie and F. D. Ronald, "Amorphous slack methodology for autonomous fault-handling in reconfigurable devices," *International Journal of Multimedia & Ubiquitous Engineering*, vol. 7, no. 4, 2012.

[5] L. Dilillo, B. Alberto, V. Miroslav, G. Patrick, P. Serge, and V. Arnaud, "Error resilient infrastructure for data transfer in a distributed neutron detector," in *Proc. IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, 2011, pp. 294-301.

[6] K. Kaur and E. N. Singh, "A survey of intrusion detection techniques," *International Journal of Advanced Research*, vol. 3, no. 6, pp. 402–405, 2013.

[7] M. Bicego, V. Murino, M. Pelillo, and A. Torsello, "Similarity-based pattern recognition," *Pattern Recognition*, vol. 39, no. 10, pp. 1813–1814, 2006.

[8] R. A. Maxion and K. M. C. Tan, "Anomaly detection in embedded systems," *IEEE Transactions on Computers*, vol. 51, no. 2, pp. 108–120, 2002.

[9] E. Aleskerov and B. Rao, "A neural network based database mining system for credit card fraud detection," *Computational Intelligence for Financial Engineering (CIFEr)*, pp. 220–226, 1997.
[10] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys*, vol. 41, no. 3, p. 15, 2009.
[11] B. Quanz, H. Fei, J. Huan, J. Evans, V. Frost, G. Minden, D. Deavours, L. Searl, D. DePardo, M. Kuehnhausen, D. Fokum, M. Zeets, and A. Oguna, "Anomaly Detection with Sensor Data for Distributed Security," in *Proc. 18th International Conference on Computer Communications and Networks*, Aug. 2009, pp. 1–6.
[12] J. W. Branch, C. Giannella, B. Szymanski, R. Wolff, and H. Kargupta, "In-network outlier detection in wireless sensor networks," *Knowledge and Information Systems*, vol. 34, no. 1, pp. 23–54, 2013.
[13] D. J. Hill and B. S. Minsker, "Real-time bayesian anomaly detection for environmental sensor data," in *Proc. International Association for Hydraulic Research Congress*, 2007, vol. 32, no. 2, p. 503.
[14] W. Du, L. Fang, and N. Peng, "Lad: Localization anomaly detection for wireless sensor networks," *Journal of Parallel and Distributed Computing*, vol. 66, no. 7, pp. 874–886, 2006.
[15] A. Amir, L. Zimet, A. Sangiovanni-Vincentelli, and S. Kao, "An embedded system for an eye-detection sensor," *Computer Vision and Image Understanding*, vol. 98, no. 1, pp. 104-123, 2005.
[16] M. Zandrahimi, H. R. Zarandi, and M. H. Mottaghi, "Two effective methods to detect anomalies in embedded systems," *Microelectronics Journal*, vol. 43, no. 1, pp. 77–87, 2012.
[17] S. A. Hofmeyr, S. Forrest, and A. Somayaji, "Intrusion detection using sequences of system calls," *Journal of Computer Security*, vol. 6, no. 3, pp. 151–180, 1998.
[18] J. D. Hamilton, *Time Series Analysis*, Princeton: Princeton University Press, 1994.
[19] Computer Immune System. (Sep. 2011). [Online]. Available: http://www.cs.unm.edu/~immsec/data/synth-sm.html
[20] K. Das and J. Schneider, "Detecting anomalous records in categorical datasets," in *Proc. the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007, pp. 220–229.
[21] H. Bhatnagar, *Advanced ASIC Chip Synthesis Using Synopsys Design Compiler, Physical Compiler and PrimeTime*, Springer, 2002.
[22] Date Visited. (Aug. 2013). [Online]. Available: http://www.tkt.cs.tut.fi/tools/public/tutorials/synopsys/design_compiler/gsdc.html

Roghayeh Mojarad was born in 1989 in Iran. She graduated as a bachelor with the major of computer hardware engineering from University of Tehran in 2011. She received her master degree of computer architecture from Amirkabir University of Technology in 2013. She was recognized as the top student in her graduate school. Her research interests include fault tolerant architecture and systems, reliable multiprocessors systems, digital system design, FPGA architecure design, HW/SW co-design.

Hamid Reza Zarandi has received his B.S., M.S., and Ph.D. degrees in computer engineering from the Sharif University of Technology, Tehran, Iran, in 2000, and 2002, and 2007, respectively. He has joined Amirkabir University of Technology as a faculty member since 2007.