2010 25th International Symposium on Defect and Fault Tolerance in VLSI Systems

# A Probabilistic Method to Detect Anomalies in Embedded Systems

Mahroo Zandrahimi, Alireza Zarei, Hamid R. Zarandi
Department of Computer Engineering and Information Technology
Amirkabir University of Technology
Tehran, Iran
E-mails: {mzand, alirezazarei, h_zarandi}@aut.ac.ir

***Abstract— Current-day embedded systems are very vulnerable to faults and defects. Anomaly detection is often the primary means of providing early indication of faults and defects. This paper presents a probabilistic method, which employs the probability of data events to evaluate the behavior of system. In order to measure the probability of events in the system, sampling of two events with distinct distance is done. Consequently, during test stage, the probability of events can be measured. An anomaly exists in test data provided that this probability does not reach a predefined threshold. The experiments on 112 standard benchmarks show that the proposed method can detect 100% of anomalies. Also, the area overhead of the proposed detector grows linearly, while the area overhead of other typical detectors grows exponentially by the increase in one of the detector's parameters.***

***Keywords-anomaly; anomaly detection; categorical data; dependability; embedded systems***

## I. INTRODUCTION

As computer systems become smaller and smaller, they will be embedded in every device, such as modern automotives [1], airplanes, air traffic controllers [2], military applications [3], hospital equipments [4], and nuclear equipments [5]. Many of these devices with embedded computers will be safety critical that any bugs can affect serious damage on human life, and therefore will require a higher level of dependability than usual. It is presumed that fault tolerance, which is one way of achieving high dependability, will be employed in such devices. In several fault tolerance methods, the first step requires that fault be detected and, then, bringing fault tolerance measures to tolerate it. Hence, fault detection is an essential first step in achieving dependability [6].

Faults can be detected either explicitly or implicitly. In the former, faults are typically detected through pattern recognition which aims to classify data (patterns) based either on a priori knowledge or on statistical information extracted from the patterns [7]. In the latter, fault is detected due to some indirect indicator, such as anomaly, that may have been caused by the fault. In such systems, many different measures are available for unusual behaviors. For example, there can be many sensors measuring the state of a system [8]. These sensors can be hardware or software. The data produced by such sensors are referred to as sensor-data that can be numeric or categorical [6]. Numeric data that are usually continuous are on a ratio scale and have a unique zero point; besides, they have mathematical ordering properties; for example, ten is twice five. Categorical data are discrete; they usually consist of a series of unique labels as categories and have no mathematical ordering properties; for example, blue cannot be added to red [9]. It seems likely that as computing power increases, more of the sensor data will be in the form of categorical data [10]; hence, anomaly detectors will be required to operate primarily on categorical data, presenting a real challenge to developers and users of such sensors because categorical data are much more difficult to handle statistically than numeric data are. This paper focuses on detecting anomalies in categorical data.

An anomaly is an event or object that differs from some standard or reference event, in excess of some threshold, in accordance with some similarity or distance metric on the event [6]. The reference event is what characterizes normal behavior. The similarity metric measures the distance between normal and abnormal. The threshold establishes the minimum distance that encompasses the variation of normalcy which means that any event exceeding that distance is considered anomalous. In categorical data, anomalous events are typically defined by the probabilities of encountering particular juxtapositions of symbols or subsequences in the data stream; i.e., symbols and sequences in an anomaly are juxtaposed in an unexpected way.

Categorical data sets are comprised of sequences of symbols [11]. The collection of unique symbols in a data set is called the alphabet. Training data are obtained from the process over a period of time during which the process is judged to be running normal. The juxtapositions of symbols and sequences within these data would be considered normal because no faults occurred during the collection period. Once the training data are characterized which is termed the training phase, characterizations of new data, monitored while the process is in an unknown state, are compared to expectations generated by the training data. Any sufficiently unexpected juxtaposition in the new data would be judged anomalous, the possible effect of a fault.

Two typical methods that have been presented in recent years for detecting anomaly in categorical data are Markov and Stide anomaly detectors [6 and 12]. The Markov anomaly detector determines whether the states in the sequential data stream, taken from a monitored process, are normal or anomalous. It calculates the probabilities to assess the transition between events in a testing set. These states and probabilities can be described by a Markov model. Although the coverage of this detector is

1550-5774/10 $26.00 © 2010 IEEE
DOI 10.1109/DFT.2010.25

relatively high, it imposes a great deal of area overhead which grows exponentially by the increase in the size of detector window. The other method, Stide is a sequence, time-delay, embedding anomaly detector inspired by natural immune systems that distinguish self (normal) from nonself (anomalous). The reference to "time" recognizes the time-series nature of categorical data on which the detector is typically deployed. Stide mimics natural immune systems by constructing templates of "self" and, then, matching them against instances of "nonself". This detector is merely able to detect anomalies of certain classes; besides, the area overhead imposed by this detector is relatively high.

This paper presents a probabilistic method that uses the relative distance between symbols in training data. This method employs the probability of events to evaluate the behavior of system in training stage, and if this probability does not reach a predefined threshold, an anomaly exists in test data. In addition, the experiments on standard benchmarks show that the proposed method is able to detect 100% of anomalies, while the area overhead grows linearly by the increase in the size of detector window.

The remainder of this paper is organized as follows: formulating the problem is presented in section 2; the proposed method is introduced in section 3. Section 4 is devoted to the procedure of generating the standard benchmark data sets, experimental results are given in section 5, and finally section 6 concludes the paper.

## II. FORMULATING THE PROBLEM

This section presents the terminology and the definitions related to the proposed method. The following definitions are inspired from [6].

Fault could manifest itself as an event injected into a normal stream of data, and that event could be regarded as normal or as anomalous. There are three phenomena that could make an event anomalous: a) Foreign symbols, a foreign symbol is a symbol not included in the training-set alphabet. For example, any symbol, such as a *Q*, not in the training-set alphabet comprising *A B C D E F* would be considered a foreign symbol. Events containing foreign symbols are called foreign symbol-sequence anomalies. b) Foreign *n*-grams/ sequences, an *n*-gram (a set of *n* ordered elements) not found in the training data set (and also not containing a foreign symbol) is considered a foreign *n*-gram or foreign sequence. A foreign *n*-gram event contains *n*-grams not present in the training data. For example, given an alphabet of *A B C D E F*, the set of all bigrams would contain *AA AB AC ... FF*, for a total $6^2$=36. If the training data contained all bigrams except *CC*, then *CC* would be a foreign *n*-gram. c) Rare *n*-grams/ sequences, a rare *n*-gram event, also called a rare sequence, contains *n*-grams that are infrequent in the training data. In the example above, if the bigram *AA* constituted 96% of the bigrams in the sequence, and the bigram *BB* and *CC* constituted 2% of each, then *BB* and *CC* would be rare bigrams.

An anomaly detector determines the similarity or distance between some standard event and the possibly anomalous events in its purview. The purview of a sliding window detector is the length of the window. The extent which the detector window overlaps the anomaly can be thought of as the detector's view of anomaly. Fig. 1 depicts different views of an anomaly injected into a sensor data stream from the perspective of a sliding-window anomaly detector. In this figure anomaly is depicted by *A* symbol, and sensor-data stream is depicted by *d* symbol. The right directed arrows indicate the direction of data flow. The case in which the window is the same size as the anomaly and the entire anomaly is captured exactly within the window is called the *whole view*. When the size of the detector window is less than the length of the anomaly, the detector has what is called an *internal view*. For the case in which the detector window is larger than the anomaly and both anomalous and normal background data are seen, this is the *encompassing view*. In a *boundary condition*, the detector sees part of the anomaly and part of the background data. Boundary conditions that arise at both ends of an injected sequence embedded in normal data occur independently of the relative sizes of the detector window and anomaly. The *background view* sees only background data and no anomalies. These conditions except boundary condition depend on the size of the injected event relative to the size of the detector window.

Different kinds of anomalies contain foreign symbols; foreign *n*-grams together with rare *n*-grams were explained in previous paragraphs. Another type of anomaly is a common *n*-gram that appears commonly in the normal data. That an anomalous sequence can be composed of several different kinds of subsequences, along with the concept of internal sequences and boundary sequences, gives rise to the idea of creating a map of the anomaly space for sliding-window detectors. By giving such a map, it should be possible to determine the extent to which that map is covered by a particular anomaly detector. An anomaly-space map is shown in Fig. 2. In this figure the window size of the detector, relative to the size of the anomaly, is shown in the three columns of the figure: detector window size less than anomaly size (DW<AS), detector window size equal to the anomaly size (DW=AS), and detector window size greater than the anomaly size (DW>AS). For each of these conditions, the figure addresses three kinds of anomalies: foreign-symbol sequence anomalies (sequences comprising only foreign symbols), foreign-sequence anomalies (sequences comprising only foreign sequences), and rare-sequence anomalies (sequences comprising only rare sequences). The first two letters in each cell identify the type of anomalous sequence; in other words, FS refers to foreign-sequence anomaly, RS refers to rare-sequence anomaly, and FF refers to foreign-symbol-sequence anomaly. The next two letters identify the type of conditions comprising internal view and encompassing view, each of which can be alien, rare, or common. The last two letters refer to the boundary conditions, which can also be alien, rare, or common. For the case that the detector window size is equal to the anomaly size, no internal or encompassing conditions occur. These impossible conditions are struck out by dashes replacing the middle two letters.

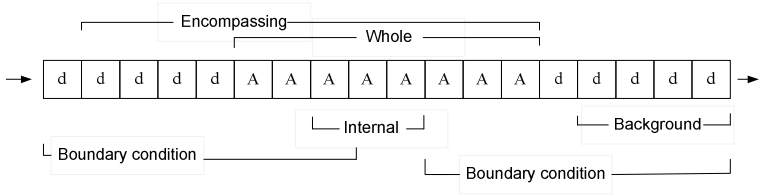


Figure 1. Different views of an anomaly detector

Foreign-Symbol-Sequence Anomalies

| DW < AS | DW=AS | DW>AS |
|---|---|---|
| FF AI AB | FF-AB | FF AE AB |

Foreign-Sequence Anomalies

| DW<AS | DW=AS | DW > AS |
|---|---|---|
| FS AI AB | FS-AB | FS AE AB |
| FS RI AB | | ~~FS RE AB~~ |
| FS CI AB | | ~~FS CE AB~~ |
| FS AI RB | FS-RB | FS AE RB |
| FS RI RB | | ~~FS RE RB~~ |
| FS CI RB | | ~~FS CE RB~~ |
| FS AI CB | FS-CB | FS AE CB |
| FS RI CB | | ~~FS RE CB~~ |
| FS CI CB | | ~~FS CE CB~~ |

Rare-Sequence Anomalies

| DW<AS | DW=AS | DW > AS |
|---|---|---|
| RS AI AB | RS-AB | RS AE AB |
| RS RI AB | | ~~RS RE AB~~ |
| RS CI AB | | ~~RS CE AB~~ |
| RS AI RB | RS-RB | RS AE RB |
| RS RI RB | | ~~RS RE RB~~ |
| RS CI RB | | ~~RS CE RB~~ |
| RS AI CB | RS-CB | RS AE CB |
| RS RI CB | | ~~RS RE CB~~ |
| RS CI CB | | ~~RS CE CB~~ |

Figure 2. Anomaly space map [6]

## III. PROPOSING A DETECTION METHOD FOR EMBEDDED SYSTEMS

In this method, by using the available information in the sliding window, it is attempted to somewhat evaluate the behavior of the system by means of employing the probability of events. One method to detect the anomaly is through identifying the probability of the subsequent anomalies. Here, sampling of two events with distinct distance is done in order to measure the probability of events in the system; this refers to the training stage. Consequently, during the test stage, the probability of events among a given window size can be measured. An anomaly exists in test data provided that this probability doesn't reach a predefined threshold. The following subsections include the explanations about the two stages of the proposed method.

### *A. The training stage*

In this method, due to the fact that the size of the detector window is restricted to the number of *n* elements, it is endeavored to extract all possible information from data and then to employ them in the test stage. In other words, the relative distance of symbols to each other is employed to record the behavior of the system in the training stage. As an example, a detector window is shown in Fig. 3, which has *n* elements.

It is obvious from the above figure that two elements are placed in the *n-1* distance, four elements in the *n-2* distance, 6 elements in the *n-3* distance, and therefore *n-1* elements are orderly placed in one distance of each other. In this stage, it is attempted to record the number as well as the distance of elements to each other in order to store their frequency function.

Eventually, with the obtained information in the form of a matrix, test data will be studied. Thereby in the statement $f_i(x,y)$, *f* is the frequency function of the number of elements *x* and *y* that are placed in the '*i*' distance of each other. Hence, during the training stage, the values $f_i(x,y)$ for each *x* and *y* in the alphabet symbols as well as each distance *n-1*, …2 and *i*=1 are measured. An example of the constructed matrix obtained from training data is shown in Fig. 4. In this figure, the rows of the matrix are all juxtapositions of two symbols of the alphabet containing eight symbols from *A* to *H*, and the columns are distance of symbols varying from 1 to 7 in the detector window of size eight (*DW*=8). The elements of the matrix are $f_i(x,y)$, which stands for the frequency function of symbols *x* and *y* with the distance of '*i*'. As an example, $f_4(A,B)$ is the frequency function related to the number of repetitions for symbols *A* and *B* that are placed in the '4' distance of each other. Consequently, the probability of events related to the two symbols of *x* and *y* that are placed in the '*i*' distance of each other can be measured if the value of each $f_i(x,y)$ is divided by the total number of the frequencies (Cumulative Distribution Function) in the obtained matrix.

### *B. The test stage*

In the test stage, through defining the probability function of the detector window, a value will be obtained based on the relative distance of the elements in the detector window. The closer this value is to 1, the more normal data will be determined, and the closer this value is to 0, the more anomalous data will be probable. Such function will be defined as there of:

$$P = \prod_{x,y \epsilon DW} \prod_{i=1}^{n-1} p_i(x,y) \qquad (1)$$

In the function above, *DW* is a set of elements in the detector window; besides, *n* is the number of elements in the detector window. Furthermore, In the statement $P_i(x,y)$, *P* is the probability that the elements *x* and *y* place in the '*i*' distance of each other. This probability is retrieved from the constructed matrix of the training stage. This function employs the product of probabilities to indicate the effect of the probabilities of each value thoroughly and properly. Suppose that *ADEF* are the symbols that a detector window of size 4 can see in its purview. The probability function of the detector window is measured by:

$$P_3(A,F) \times P_2(A,E) \times P_2(D,F) \times P_1(A,D) \times P_1(D,E) \times P_1(E,F) \quad (2)$$

If this sequence is common to the training data, the measured value will be close to "1"; also, this value will be "0" provided that the mentioned sequence is foreign to the training data, and finally the more infrequent this sequence is, the closer this value is to "0". The more this value exceeds a *threshold*, the more anomalous the event will seem. The *threshold* determines how dissimilar from normal an event can be, while remaining within the bounds of what is considered to be normal behavior. If the threshold of dissimilarity is exceeded, then the observed behavior is judged to be anomalous. The threshold is often situation-specific and depends on environmental characteristics. Fig. 5 depicts a pseudo code of the proposed detector where γ and β establish the minimum distance that encompasses the variation of normalcy; any event exceeding these thresholds is considered anomalous.

## IV. CONSTRUCTING THE BENCHMARK DATASETS

This section provides details of the benchmarking process. In general, three kinds of data need to be generated: training data (normal background), testing data (background plus anomalous events), and the anomalies themselves. In benchmarking parlance, the training and testing data constitute the anomaly detector workload.

### *A. Generating the training and test data*

The training data serve as the "normal" data into which anomalous events are injected. The requirements for the training data are that a large portion of the data should be comprised of common sequences, that they contain a small proportion of rare sequences, and that there has to be a relatively high predictability from one symbol to another.

The alphabet has eight symbols: *A, B, C, D, E, F, G,* and *H*. Since increasing the alphabet size would not change the outcome, maintaining a relatively small alphabet size facilitates a more manageable experiment, yet permits direct study of detector response. The training data were generated from an eight by eight state transition matrix with the probability of 0.9672 in one cell of each row, and 0.004686 in every other cell, resulting in a sequence of conditional entropy 0.1 [13]. One million data elements were generated so that there would be a sufficient variety of rare sequences for the test data. Ninety eight percent of the training data consisted of the repetitions of the sequence *ABCDEFGH*, seeding the data set with common sequences. This is the data set applied to train the detector used in this study, i.e. to establish a model of normalcy against which unknown data can be compared.

Test data, containing injected anomalies, are used to determine how well the detector can capture anomalous events and correctly reject events that are not anomalous. The test data consist of two components: a background, into which anomalies are injected, and the anomalies themselves. Each is generated separately, after which the anomalies are injected into the background under strict experimental control. The background is the same as the training data generated in the previous stage.

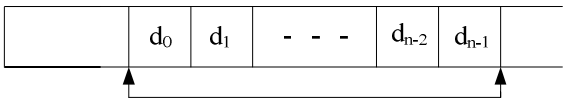


Figure 3. Detector window of an anomaly detector

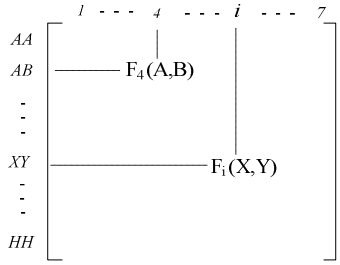


Figure 4. An example of the constructed matrix obtained from training data

- Training_stage_algorithm(training data, *n*)
  1) Construct a matrix with *(# of alphabets)*$^2$ rows and *(n-1)* columns.
  2) While (the end of training data)
     2.1) For all pairs of symbols in the *DW* that are placed in the *n-1* distance to one distance of each other, record the frequency function in the matrix.
     2.2) Move *DW* forward.
  3) $CDF = \sum_{x,y\ \epsilon\ rows\ of\ matrix} \sum_{i=1}^{n-1} f_i(x,y)$
  4) Divide each element of matrix by CDF.
- Test_stage_algorithm(test data, matrix, $\beta$, *n*)
  1) flag=0; counter=0;
  2) While (the end of test data)
     2.1) $P = \prod_{x,y\epsilon DW} \prod_{i=1}^{n-1} p_i(x,y)$
     2.2) if ($P<\beta$ and flag=0)
          counter++; set flag;
        else if ($P<\beta$ and flag=1)
          counter++; if (counter>$\gamma n$) alarm();
        else if ($P>=\beta$ and flag=1)
          reset counter; reset flag;
     2.3) Move *DW* forward.

Figure 5. Pseudo code of the proposed detector

### B. Constructing the anomalous injections

Once the background data are available, anomalous events must be injected into them to finalize the test data. The selected anomaly type is a foreign sequence of length AS for which all subsequences of length less than AS that make up the internal sequences and the boundary sequences are rare. Rare is defined to be any sequence of detector window length that occurs in the training data less than one percent of the time.

Once that anomaly type is determined, e.g. FS RI RB, as described in section 2, the next step is to inject a foreign sequence composed of rare sequences into the test data. A catalog of rare *n*-grams is obtained from the training data. Rare *n*-grams are drawn from the catalog and composed to form a foreign sequence of the appropriate size. Sufficient numbers of rare *n*-grams are added to the constructed anomaly to form the rare boundary conditions as the anomaly passes through the detector window. For example, the sequences of size three *BAF*, *AFH, FHE, HEC, ECC,* and *CCF,* each of which occurred less than 1% of the time in the training data; consequently, these are rare sequences. Combining these six sequences (*BAFHECCF*) produces one anomaly of size four (*FHEC*) whose internal sequences and boundary conditions are made up of rare sequences of size three. This anomaly was injected into the background data.

Eight injection sizes and fourteen detector window sizes have been tested. The procedure, outlined above for creating the anomalous events and for injecting them, is repeated for each combination of injection size and window size, resulting in 112 total data sets.

## V. EXPERIMENTAL RESULTS

This section describes the experiments performed to evaluate the detection coverage of each method; moreover, an accurate estimation of the number of memory elements which is required for each method was done. The required memory was estimated merely for storing the constructed model of normalcy during the training stage.

### B. Detection coverage

Each of the three detectors, Markov, Stide, and the proposed detector was implemented by C++ program and provided with the same set of training data. From the training data, the detectors learned their models of normal behavior. Then, each detector was tested by using each of the 112 test data sets described in section 4. The size of the detector window was varied from 2 to 15, and the size of the injected event was varied from 2 to 9.

Detection and blind regions for each method are depicted in Fig. 6. These decision maps illustrate the detection capability of each method with respect to an injected foreign sequence composed of rare sequences. The x-axis of each map marks the increasing size of the foreign sequence injected into the test data; the y-axis marks the size of the detector window required to detect a foreign sequence of a specified size. Each star mark indicates successful detection of the foreign sequence anomaly whose size is indicated on the x-axis through using a detector window whose size is marked on the y-axis. The areas that are bereft of stars indicate detection blindness which means that the detector was unable to detect the injected foreign sequence whose corresponding size is marked on the x-axis.

The results show that although all detectors use the concept of a sliding window, their different similarity metrics significantly impact their detection capabilities. In the case of Stide, even though there is a foreign sequence present in the data stream, it is visible only if the size of the detector- window is at least as large as the foreign sequence- a requirement that the Markov detector and the proposed detector do not have. This figure shows that the proposed detector, and the Markov detector have 100% detection capability, yet Stide detector has only 75% detection coverage; moreover, the following subsection shows that the number of memory elements required for the proposed detector is significantly less than the Markov and Stide detector.

### C. Memory estimation

In this subsection , an accurate estimation of the number of memory elements required for each detector is done; furthermore, this estimation was examined through several experiments when the window size of the detectors vary from 2 to 15.

Suppose that $\sum$ is the number of symbols in the alphabet and $n$ is the length of the sliding window. The Markov detector requires memory to store the transition matrix related to the normal behavior of the system. The number of rows and columns of this matrix is each equal to the number of states constructed during the training stage. The number of states in the worst-case is equal to $\sum^n$, which are all the possible juxtapositions of $\sum$ symbols in the detector window of size $n$. Therefore, the required memory for storing the matrix elements is obtained by:

$$\sum^n \times \sum^n = \sum^{2n} \quad (3)$$

The Stide detector requires a database consisting of all unique sequences extracted from the stream of training data. In the worst-

case, $\sum^n$ unique juxtapositions of symbols with the size of $n$ are possible, so the required memory is obtained by:

$$\sum^n \times n \qquad (4)$$

The proposed detector requires memory to store the constructed matrix obtained from training data. The rows of the matrix are all juxtapositions of two symbols of the alphabet, and the columns are distance of symbols varying from 1 to $n$-1. Therefore, the estimated memory is obtained by:

$$\sum^2 \times (n-1) \qquad (5)$$

It is interesting to note that the required memory of the Markov and Stide detectors grows exponentially, while the required memory of the proposed detector grows linearly by the increase in the size of $n$. Table 1 shows the number of memory elements required for each detector. The numbers are extracted by using training data generated in section 4. It is obvious from this table that the difference between the required memory elements of the proposed detector and the other detectors becomes significant as $n$ grows from 2 to 15. Furthermore, the required memory of the proposed detector remains constant, while the required memory of the Markov and Stide detectors approaches the worst-case by the increase in the size of training data.

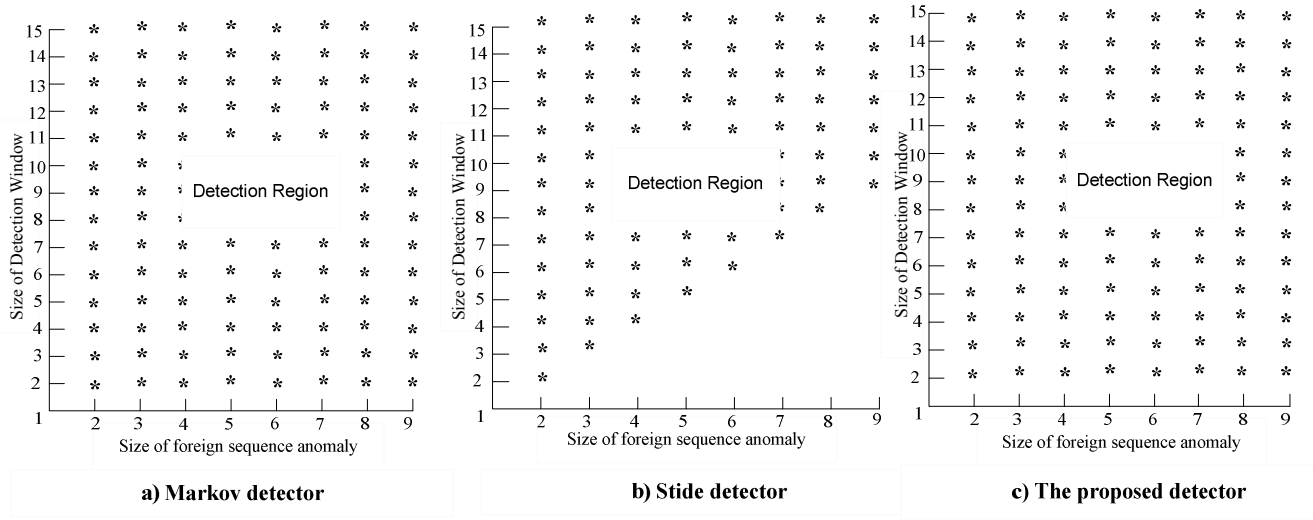


Figure 6. Detection coverage of the three detectors

TABLE I. NUMBER OF MEMORY ELEMENTS REQUIRED FOR EACH DETECTOR

| n | **Markov** [6] | | **Stide** [6] | | **The proposed detector** |
|---|---|---|---|---|---|
| | ***# of states*** | ***# of memory elements*** | ***# of sequences*** | ***# of memory elements*** | ***# of memory elements*** |
| 2 | 64 | 4096 | 64 | 128 | 64 |
| 3 | 491 | 241081 | 491 | 1473 | 128 |
| 4 | 1042 | 1085764 | 1042 | 4168 | 192 |
| 5 | 2545 | 6477025 | 2545 | 12725 | 256 |
| 6 | 4226 | 17859076 | 4226 | 25356 | 320 |
| 7 | 6371 | 40589641 | 6371 | 44597 | 384 |
| 8 | 8986 | 80748196 | 8986 | 71888 | 448 |
| 9 | 12085 | 146047225 | 12085 | 108765 | 512 |
| 10 | 15675 | 242705625 | 15675 | 156750 | 576 |
| 11 | 19749 | 390023001 | 19749 | 217239 | 640 |
| 12 | 24310 | 590976100 | 24310 | 291720 | 704 |
| 13 | 29385 | 863478225 | 29385 | 382005 | 768 |
| 14 | 34955 | 1221852025 | 34955 | 489370 | 832 |
| 15 | 40993 | 1680426049 | 40993 | 614895 | 896 |

## VI. CONCLUSIONS

This paper presented a probabilistic method, which employs the probability of events to evaluate the behavior of system. Sampling of two events with distinct distance is done in order to measure the probability of events in the system; this refers to the training stage. Consequently, during test stage, the probability of events among a given window size can be measured. An anomaly exists in test data provided that this probability doesn't reach a predefined threshold. The experiments on 112 standard benchmarks show that the proposed method can detect 100% of anomalies. Also, the area overhead of the proposed detector grows linearly, while the area overhead of other typical detectors grows exponentially by the increase in the size of the detector window. This point is very essential for embedded systems which have restrictions on required hardware.

## REFERENCES


[1] M. Short and M. J. Pont, "Assessment of high integrity embedded automotive control systems using hardware in the loop simulation," *Journal of Systems and Software*, vol. 81, no. 7, pp. 1163-1183, 2008.

[2] A. V. Lovato, E. Araujo, and J. D. S. da Silva, "Fuzzy decision in airplane speed control," *In the Proceedings of IEEE International Conference on Fuzzy Systems*, pp. 1578-1583, 2006.

[3] R. Bastide, D. Navarre, P. Palanque, A. Schyn, and P. Dragicevic, "A model-based approach for real-time embedded multimodal systems in military aircrafts," *In the Proceedings of ACM international conference on Multimodal Interfaces,* pp. 243-250, 2004.

[4] H. Al Nahas and J. S. Deogun, "Radio frequency identification applications in smart hospitals, " *In the Proceedings of IEEE International Symposium on Computer-Based Medical Systems*, pp. 337-342, 2007.

[5] A. Scholz, S. Sommer, C. Buckl, G. Kainz, A. Kemper, A. Knoll, J. Heuer, and A. Schmitt, "Towards and adaptive execution of applications in heterogeneous embedded networks," *In the Proceedings of ACM/IEEE International Workshop on Software Engineering for Sensor Network Applications*, 2010.

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

[7] M.Bicego, V. Murino, M. Pelillo, and A. Torsello, "Similarity-based pattern recognition," *Journal of pattern recognition*, vol. 39, no. 10, Pages 1813-1824, 2006.

[8] R. A. Maxion and F. E. Feather, "A case study of ethernet anomalies in a distributed computing environment," *Journal of IEEE Transactions on Reliability*, vol. 39, no. 04, pp. 433-443, 1990.

[9] S. Boriah, V. Chandola, and V. Kumar, "Similarity measures for categorical data: a comparative evaluation," *In the Proceedings of SIAM International Conference on Data Mining*, pp. 243-254, 2008.

[10] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: a survey," ACM *Computing Surveys,* vol. 41, no. 3, Article 15, 2009.

[11] S. Budalakoti, A. Srivastava, and M. Otey, "Anomaly detection and diagnosis algorithms for discrete symbol sequences with applications to airline safety," *In the Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*, pp. 101-113, 2007.

[12] S. Jha, M. C. Tan, and R. A. Maxion, "Markov chains, classifiers, and intrusion detection," *In the proceedings of IEEE International Workshop on Computer Security Foundations*, pp.206-219, 2001.

[13] R. A. Maxion and K. M. Tan, "Benchmarking anomaly-based detection systems," *In the Proceedings of IEEE International Conference on Dependable Systems and Networks*, pp. 623-630, 2001.