

Robustness Testing of Real-Time Automotive Systems Using Sequence Covering Arrays

Graciela Becci and Gunwant Dhadyalla
University of Warwick

Alexandros Mouzakitis
Jaguar Land Rover

James Marco
Cranfield University

Andrew David Moore
QM Systems, Ltd.

ABSTRACT

Testing real-time vehicular systems challenges the tester to design test cases for concurrent and sequential input events, emulating unexpected user and usage profiles. The vehicle response should be robust to unexpected user actions.

Sequence Covering Arrays (SCA) offer an approach which can emulate such unexpected user actions by generating an optimized set of test vectors which cover all possible t-way sequences of events.

The objective of this research was to find an efficient nonfunctional sequence testing (NFST) strategy for testing the robustness of real-time automotive embedded systems measured by their ability to recover (prove-out test) after applying sequences of user and usage patterns generated by combinatorial test algorithms, considered as “noisy” inputs.

The method was validated with a case study of an automotive embedded system tested at Hardware-In-the-Loop (HIL) level. The random sequences were able to alter the system functionality observed at the prove-out test. The outputs were affected in different ways such as unexpected events, lead-lag phase shift, and shifted parameter values.

The method is effective to find concerns and efficient due to the reduction in the number of test cases compared to the full factorial alternative. The method is also efficient because it only requires the observation of the recoverability of the system at the prove-out test, without the need of determining the system response to the noisy combinatorial inputs. The NFST method has been recommended as a testing standard for our partner vehicle manufacturer for testing electronic and software systems applicable to all vehicle variants.

CITATION: Becci, G., Dhadyalla, G., Mouzakitis, A., Marco, J. et al., "Robustness Testing of Real-Time Automotive Systems Using Sequence Covering Arrays," *SAE Int. J. Passeng. Cars – Electron. Electr. Syst.* 6(1):2013, doi: 10.4271/2013-01-1228.

INTRODUCTION

Research Aims

The aim of this project was to find an efficient and effective non-functional strategy for testing the robustness of the system to sequence of events. The strategy is applicable at different levels in the product development lifecycle, and validated through a case study of an existing automotive system.

Non-functional Sequence Testing Strategy

The NFST strategy is intended to replicate extreme input interactions through a set of optimized test cases based on Combinatorial Sequence Testing approaches, observing the ability of the system to recover as a measure of system robustness.

Non-functional Sequence Testing Method

A process was developed to support achieving the goal of an efficient set of sequence test vectors. This involved:



Documents

Document Sharing Hub

Determining system boundaries based on features and functions. 2) Creating Parameter (P) diagram for the system under test (SUT). 3) Generating combinatorial sequence test cases. 4) Executing tests and analysing results.

NON FUNCTIONAL SEQUENCE TESTING (NFST)

Rationale for using Combinatorial and Sequence Combinatorial Testing

Research conducted in the area of combinatorial testing [1] provides evidence to support the claim that most software failures result from the combination of simple events rather than complex ones. Even where many hundreds of variables existed, most failures were caused by interactions between just two variables and nearly all failures by interactions between four and six variables. The Covering Array (CA) is a proven method for generating combinations of inputs that, in this case, are intended to emulate unpredictable user-usage profiles.

The CA approach is effective to uncover faults produced by the interaction of two or more parameters however in this case study the gains made by analysing greater than 3-wise interactions were considered less efficient for the number of test cases generated [1].

A particular approach of combinatorial testing for an event based system is sequence testing, based on the fact that some faults are triggered by particular sequence in which events occur.

There are commercial tools available for black-box sequence testing that allow for the definition of the sequences and the transition steps using sinusoidal, square or saw-tooth timing profiles. However, the sequence generation has to be conducted outside these tools with no automatic approach to generate test cases [2].

Sequence Testing

As described by Kuhn *et. al*, in testing event-driven software, the critical condition for triggering failures often is whether or not a particular event has occurred prior to a second one, not necessarily if they are back to back. This situation reflects the fact that in many cases, a particular state must be reached before a particular failure can be triggered [3].

The time between events, as shown in Figure 1, is dependent on many factors. For example the clock rate of the different microprocessors, timer values employed in software functions for sampling rates of sensors and signals or network bus read and write intervals. Therefore the time between events has to be a variable which is selected to represent aspects of real world behaviour.

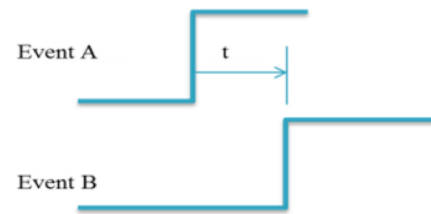


Figure 1. Example Sequence test for variable timing

The sequences of events are generated to ensure that any t events are tested in every possible t -way order. The standard example mentioned in literature [3], is redefined here to include the time vector. The example considers four events, a , b , c , and d as described in Table 1. All 3-way sequences can be tested with only six tests even though there are $4! = 24$ possible permutations of these four events.

The time vectors are generated according to an exponential distribution and included in the last three columns of Table 1. The concept of time generation is explored in later sections.

Table 1. Example of the NFST approach: sequences of events and the related time interval

	1	2	3	4	t_{1-2}	t_{2-3}	t_{3-4}
1	a	d	b	c	0.082	0.047	0.080
2	b	a	c	d	0.015	0.033	0.001
3	b	d	c	a	0.158	0.122	0.002
4	c	a	b	d	0.061	0.022	0.055
5	c	d	b	a	0.064	0.052	0.006
6	d	a	c	b	0.028	0.010	0.246

Testing real-time embedded systems with high number of inputs can lead to large test sets. In this case 16 input events relating to a specific function are selected out of a possible 57 for the whole feature. A full factorial test case generation for the 16 inputs events ($16!$) results in a total of $2.09E+13$ sequences.

Sequence Covering Arrays (SCA)

SCA is an approach intended to optimize the number of test cases generated while giving full input coverage for the number of grouped events [3].

Definition. We define a sequence covering array, $SCA(N, S, t)$ as an $N \times S$ matrix where entries are from a finite set S of s symbols, such that every t -length permutation of symbols from S occurs in at least one row; the t symbols in the permutation are not required to be adjacent [3].

Considering the permutation of the input parameters grouped by t elements, the number of sequences is calculated according to the expression

$$n!/(n-t)!$$

where n is the total number of events and t is the number of events considered in any given sequence.

In our case study, the permutation of the 16 parameters grouped by 3 events, results in 3360 sequences.

SCAs therefore significantly reduce the number of test vectors, giving in this case 20 test vectors for testing the 16 input parameters in a 3-way sequence interaction.

Rationale for HIL Sequence Testing

At Model-in-the-loop (MIL) level, the source of race conditions is related mainly to multitasking and multithreaded access to the system and is not representative of real-time race conditions as seen in a distributed vehicle networked system.

At Hardware-in-the-loop (HIL) level the vulnerability of the system is higher as race conditions are seen in real-time and may arise at any point in the communication between ECUs through the communication networks, affecting signals and network messages.

The interaction of different users and the environment with the vehicular system increases the vulnerability of the system to race conditions due to the variability in the usage for different users.

Testing at HIL level is selected for this project due to the higher level of disturbances, simulating complex user-usage variation that the NFST testing approach may replicate in a real-time vehicular system.

NFST Modelling Process: Time between Events

The time between events requires a special consideration due to the fact that smaller intervals are associated with occurrence of some type of failure and the related hazard rate.

Research findings demonstrated a high vulnerability of real-time embedded systems to low voltage transients in the range of smaller intervals, evaluated by the higher probability of occurrence of Diagnostic Trouble Codes DTC in the range of 100 ms [4].

Examples of these types of failure are race conditions typically at cranking stage where the power supply has a high demand, and the failure can be related to signal propagation through the network, handshaking protocols with the microprocessor, clocks and timers' synchronization, etc. [4].

According to reliability theory, an exponential distribution is used to generate the time between failure events in the lower range of the time scale, where the probability of failure is higher [5].

The exponential distribution allows for a temporal skewing towards smallest intervals to increase the probability of uncorrelated fast transients disturbance [6]. It also ensures that there is an efficient use of available testing time by

having more tests conducted where the time between events is small, whilst not neglecting the possibility of a possible longer delay.

The time between events has an exponential distribution with probability density function

$$F_{Exponential} = \lambda e^{-\lambda x}$$

Where x is the continuous time parameter, and λ is the rate parameter. The rate parameter is determined by the properties of the system(s) under test. For example if testing a system that is part of a high speed CAN network where the transmission rate is 1Mbps, the value of λ would be larger than for a system on a low speed CAN network where the transmission rate is 125kbps. In our example λ was taken at 0.022 to ensure the majority of times between events were within 200ms of each other but not neglecting the possibility of generating time values greater than 200ms, as shown in [Figure 2](#)

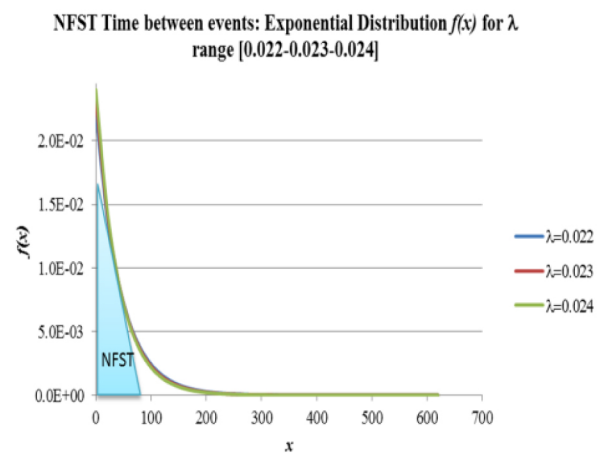


Figure 2. Exponential distribution $f(x)$ for generating the NFST time between events.

NFST Events and Time Generation

For NFST testing, the events are generated according to the combinatorial approach of Sequence Covering Array (SCA); and the time between events are randomly selected from a pool of exponentially distributed samples within the range of 1 and 500 ms.

NFST TEST PROCEDURE

Case Study

In order to evaluate the impact of the NFST method, a highly distributed feature, made up of many functions, is selected as the Hardware-in-the-loop (HIL) SUT in [Figure 3](#).

The NFST test procedure is described in [Figure 4](#), and starts by generating the sequences of event using the Sequence Covering Array (SCA) approach. The SCA algorithm generates an optimised set of sequences, consisting