

## Comprehensive Study of the Testing Coverage of Testing Methods for Communications Protocols and Software

Dr. Hazem El-Gendy \*, Dr. Magdi Amer \*\*

\* Faculty of CS & IT, Ahram Canadian University, Egypt, [h\\_elgendy@masrawy.com](mailto:h_elgendy@masrawy.com)

\*\* Faculty of Eng., OumKorea Uni., Makkua, Kingdom of Saudi Arabia, [Magdi.amer@gmail.com](mailto:Magdi.amer@gmail.com)

**Abstract:** The importance of certification/conformance testing the implementations of communications protocols and software systems are fast increasing. Formal methods for the derivation of testing sequences have been developed. In this paper, we present the results of a comprehensive study of the coverage of these testing methods using a simulator. We also present analytical remarks on the methods and analysis of their scope of applicability.

[Hazem El-Gendy, Magdi Amer. **Comprehensive Study of the Testing Coverage of Testing Methods for Communications Protocols and Software.** *J Am Sci* 2013;9(1):79-84]. (ISSN: 1545-1003). <http://www.jofamericanscience.org>. 15

**Keywords:** Communications Protocols, Conformance/Certification Testing, Formal Methods, Test Derivation Methods, Implementation Under Test (IUT), Finite State Machine (FSM).

### I. Introduction

Telecommunications has turned the world into a global village. Furthermore, the applications of telecommunications and software applications have penetrated almost all other fields and areas of development. Consequently, the demand on telecommunications applications, protocols, and systems as well as software applications and systems are fast increasing [1-16].

This increase in demand is conditional to and require correctness of the developed systems and their compliance/conformance to given specifications. This in turn increased the importance of testing the implementations of these systems for conformance/compliance to given standards or specifications.

Now, the cost of testing constitutes a large percentage of the cost of these systems. To make testing more effective in detecting errors when present and to make it cost effective, formal methods for the derivation of test sequences have been developed. These methods rely on applying sequences of inputs to the Implementation Under Test (IUT) and observing the resulting output sequences. If the resulting output sequences all coincide with the output sequences corresponding to the applied input sequences, according to the given specifications, the IUT is considered a faithful implementation according to the testing method in consideration. Otherwise, the IUT is considered faulty. It is not the role of the testing

method to identify and locate the fault; even though, it provides valuable information for this aspect.

These formal methods, for Finite State Machine – based systems, include: Transition Tour Method [3], Distinguishing Sequence Method [5], W Method [6], Unique Input/Output Sequence (UIO) [2], and Unique Input/Output Sequences Set [7].

The use of formal methods for the derivation of the testing sequences has the following pluses:

1. Facilitates study of the coverage of the method as in this paper.
2. Facilitates extension/enhancements of the method.
3. Facilitates significant reduction in the cost of testing.

In this paper, we study the coverage of the different testing methods indicated above. We use simulation to apply the methods to a large number of faulty FSMs and record the statistics for which method detects the largest number of errors. We also look at the scope of applicability of the methods.

The rest of this paper is organized as follows. Section II presents assumptions of conformance testing. In Section III, we present the 5 testing methods with analytical remarks on each of them. Then, in Section IV, we present the study of the coverage of the testing methods. We conclude the paper in Section V.

## II. Generation of Testing Sequences from Formal Protocol Specifications

In this section, we present important issues in conformance testing the implementations of communications protocols and/or software.

### II.1. Assumptions of Conformance Testing

The following are the assumptions of the work on conformance testing of the implementations of a communications protocol, and interoperability analysis [1]:

1. The given protocol standard is correct<sup>1</sup>. Correct here means, for example, that it achieves its intended purpose (Verification of the designs of Protocol Standards is outside the scope of Conformance Testing), and that it possesses certain correctness properties such as deadlock free (i.e., the design does not put conforming implementations in situations where deadlock is eventual. Validation of the designs of protocols is also outside the scope of conformance testing). This is often accommodated, in the protocols design, by including an Error state.
2. The given protocol standard is complete. By complete, we mean that all interactions required to faithfully achieve every capability, particularly mandatory capabilities, are specified in the standard<sup>2</sup>. However, some standards are incomplete (intentionally or unintentionally) leaving some areas of the standard for bi-lateral or multi-lateral agreements among vendors. In these cases, interoperability analysis cannot be applied to these incomplete parts (capabilities); but, can be applied to the standard augmented with such agreements as far as the implementations of the parties of the agreement are considered.

In the conformance testing of communication protocols, the implementation under test (IUT), in general, is treated as a black box and tested over a communication network. The selection of cost-effective test sequences is critical. A reasonable test suite should broadly cover both the control flow and the data flow of the given protocol.

### II.2. Testing the Control Flow of Interactions

<sup>1</sup>This is a typical assumption in the area of Conformance Testing of the Implementation of a Protocol Standard where the standard, the testing is conducted with respect to it, is assumed error free. Yet, sometimes Conformance Testing unveils some errors in the standard.

<sup>2</sup>Definition of completeness of standard is significantly different from definition of completely specified FSM.

Testing the control flow means testing the control aspects of different possible sequences of interactions. A conforming IUT should accept those and only those sequences of interactions that are allowed by the specification [2].

### II.3. Testing the Data Flow in the Interactions

Testing the data flow means testing the interdependence between parameters of the interactions (in legal sequences) with respect to initialization and reference operations on the parameters.

The sequences that a conforming IUT accepts should satisfy a set of data flow constraints, as given by the specification, on the parameters of the interactions of those sequences [1].

### II.4. Properties for an Effective Test Method

A 'good' test method satisfies the following properties in descending order of importance:

1. The resulting test suite is able to detect the presence of as many types of errors in the IUT as possible, if there are any. This is required for a high degree of confidence that different conforming IUTs will interwork.
2. The derived test sequences are as short in total length as possible. This is required to reduce both the time and the cost of testing.
3. The method is feasible, there exists an algorithm which to derive test sequences.
4. The corresponding algorithm facilitates automation and is of minimal complexity and cost.

The test methods we will survey fall into two categories; the first includes EFSM-based methods (appropriate for Estelle and SDL), the second category includes logic-based methods (appropriate for Lotos).

## III. Formal Derivation of Test Sequences from EFSM-Based Specifications

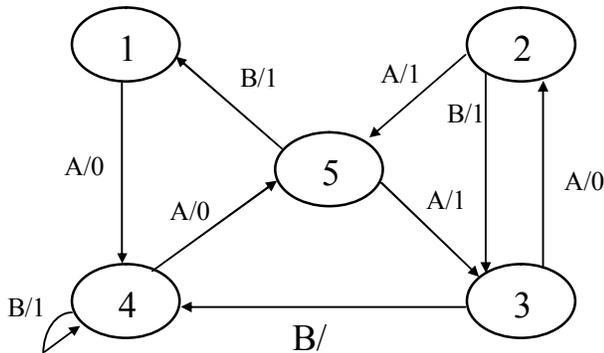
In this subsection, we review methods for deriving test sequences from a protocol specification modeled by an Extended Finite State Machine (EFSM) as in Estelle and/or SDL specifications.

A number of FSM-based-test sequence derivation methods for testing control flow have been proposed. These are: transition tour, characterization set, distinguishing sequence, and unique input/output methods.

### III.1. Transition tour methods by Naito

A transition tour [3] of a FSM is obtained by traversing the corresponding transition graph of the FSM so that every transition is covered at least once. This method has a wide range of applicability as it can be applied to any protocol specified by an FSM. So, for the machine M indicated in Figure 1, a

transition tour testing sequence can be: A/0 B/1 A/0  
A/1 A/0 B/1 B/0 A/0 A/1 A/0 A/1 B/1.  
a/n stands for on receiving Input a send output n



**Figure 1: A Transition Diagram for a Machine M**

**Remarks:** The test sequences derived by applying this method can detect errors in transition but cannot detect errors in states. Thus, its power of detecting errors is limited [4,5].

For example: starting from state 5, if the implementation on receiving B outputs 1 and goes to state 3 rather than state 1, the testing sequence “A/0 B/1 A/0 A/1 A/0 B/1 B/0 A/0 A/1 A/0 A/1 B/1” won’t detect such an error. Such an error cannot be detected unless terminating state of the transition is verified. Verification of the states can be by applying a sequence or a set of sequences that represent a unique signature of the state (the implementation is in the state iff it outputs the corresponding output to the signature) as in the following methods.

### III.2. The distinguishing sequence method by Gonenc

The method is proposed in [5] for testing switching circuits defined by FSM’s. It is based on identifying a distinguishing sequence (DS) for the graph of the switching circuit. A sequence of inputs is a distinguishing sequence iff it can be applied at any state in the FSM and the application of this sequences at any two different states produce two different output sequences.

Formally,

An input sequence  $DS = I_1; I_2; \dots; I_n$  is a distinguishing sequence iff for any two states  $s_k$  and  $s_j$  such that  $k \neq j$ , there should be  $O(s_k) \neq O(s_j)$ .

where  $O(s)$ : output sequence resulting from applying DS to the IUT at state  $s$ .

So, we can recognize the initial state of the machine by applying this input sequence to a conforming implementation and observing the

output sequence. Then, the test sequences consist of a state tour and a transition tour. In the state tour, we try to recognize the existence of every state  $s$  in the FSM by applying a sequence that leads the machine to state  $s$  (preamble sequence of state  $s$ ) then we apply the DS. The implementation passes this test case iff it outputs the output sequence corresponding to the preamble sequence followed by the output sequence corresponding to DS for state  $s$ . This latter part is unique for the state  $s$  as when we apply DS at any two other states, the corresponding outputs are different. In other-words, applying DS and observing the outputs, the output sequence is sufficient to completely determine the state the implementation was at on applying DS.

The transition tour aims at recognizing the existence of every transition between two states  $s_i$  and  $s_j$  by applying a sequence that leads the machine to state  $s_i$  then applying the input sequence of the transition followed by the DS (post-amble sequence). So, for the M machine of Figure 2, the Input sequence “B B” is a DS [5]; if it is applied at any two different states, the corresponding output sequences are different. To validate state 4, we apply an Input sequence that leads to state 4 then, we apply the DS (BB); so, we get A B B. If the corresponding output sequence is 011, the implementation is considered to correctly support state 4. To validate the transition from state 3 to state 4, we apply a sequence that leads from initial state (state 1) to state 3 then the Input of the transition (B) then DS (BB, to validate the state at which the implementation arrives on executing the transition); so, we get: A A A B B B. If corresponding output sequence is 001011, the implementation is considered to correctly support the transition.

**Remark:** This method has a good error-detection capability [5] but it has a narrow range of applicability. This is because it requires almost fully specified FSM and the DS does not exist for every FSM.

### III.3. The W-method by Chow

Chow’s method [6] can test the implementation against many kinds of errors. This method is based on generating  $P$ , the set of all partial paths in the testing tree (set of preamble sequences as in the DS method), and a characterized set  $W$ . A characterization set  $W$  is a set of inputs specified for every state in the FSM such that the sets of outputs resulting from applying the set of elements of  $W$  at any two different states should be different.

Then, the test sequences are the sequences in  $P \times W$ . For example,  $\{A, AA, B\}$  is a  $W$  set for the machine  $M$  of Figure 2.

So, the only difference between this method and the DS Method is the use of a  $W$  set as the signature rather than a DS. This makes the method has wider applicability.

**Remarks:** The length of  $P$  ( $|P|$ ) increases roughly at the order of the number of states in the FSM multiplied by the number of transitions in the FSM. Also, the length of  $W$  increases roughly at the order of the number of states in the FSM. So, the test sequences derived by using this method tend to be very long for any real protocols. Also, there is no characterization set for some protocols; i.e., there is no single set of input interactions that satisfy the requirements of a characterization set.

#### III.4. The UIO method by Sabnani, et al

This method is somewhat similar to the DS and  $W$  methods but has a new signature called Unique Input/Output Sequence (UIO Sequence). The main idea of Sabnani and Dhubra's method [2] is to find an input/output sequence (UIO) for each state in the specification such that this sequence is a unique characteristic of only that state. A unique characteristic of a state means that according to the specification, applying that input sequence starting from that state should give an output sequence exactly the same as the expected one, and applying that input sequence starting from any other state should give a different output sequence. For example,  $A/1 A/1$  is a UIO sequence for state 2 of the machine  $M$  in Figure 2.

**Remarks:** This method is a generalization of the distinguishing sequence with the difference that instead of having only one sequence (DS) per FSM, we have one sequence (UIO) per state in the FSM. This increases the applicability of the method. However, for some protocols one or more state will not have a UIO.

For testing the data flow aspects of the IUTs, relatively fewer techniques have been proposed [66, 71, 72].

#### III.5. The Unique Input/Output Sequences Set by El-Gendy

This method [7] utilizes a unique signature of every state. The unique signature is a set of sequences (rather than a single sequence as in UIO method) of Input/Outputs such that the set of input sequences for state  $s$  are all acceptable at state  $s$  and produce the corresponding set of output sequences. Also, at any other state  $s'$  either at least one of the input sequences is not fully acceptable at state  $s'$  or

the corresponding sequence of output sequences is different from that specified for  $s$ . Then, the testing sequences consist of 2 over-lapping tours:

1. State tour that consists of one of the following for every state  $s$ :  
Preamble sequence of state  $s$ .  $UIOSeqSet(s)$ <sup>3</sup>
2. Transition tour that verifies a state for every transition  $i_j/o_j$  that goes from state  $s$  to state  $s'$ :  
(Preamble of state  $s$ ;  $i_j$ ).  $UIOSeqSet(s')$ .

**Remarks:** This method has the widest scope of applicability of all methods. This is because the presence of the  $UIOSeqSet$  is guaranteed by the fact that the FSM is minimal. Also, an algorithm that facilitates full automation of the derivation of the  $UIOSeqSet$  is given. Nevertheless, as the method utilizes unique signature of every state plus a transition tour and a state tour, the coverage of the test method is expected to be as good as any of the other methods.

#### IV. Study of the coverage of the various testing methods

In this Section, we present the results of comprehensive study of the testing coverage (the strength to detect errors) of the 5 testing methods. We conducted simulation experiments using Mat Lap simulator. We supplied many FSMs then the simulator introduced various types of errors in the supplied FSM (simple state error, simple transition error, combined errors, multiple errors). The simulator applies various testing methods to the generated erroneous FSM and records which method detects the errors. Figures 2-6 present the resulting statistics for the 5 testing methods. From this study, we conclude the following:

1. The coverage of the transition tour method is far less than the coverage of the other 4 methods.
2. The coverage of the  $UIOSeqSet$  is slightly higher than the coverage of the other 3 methods.
3. The applicability of the  $UIOSeqSet$  is the highest followed by the transition tour method followed by the UIO sequence method followed by the other 2 methods that require fully specified FSM. By fully specified, it is meant that every possible input, according to the specification, is a valid input at every state. So, if there are  $n$  states and  $m$  different inputs, the machine has to have at least  $n$  times  $m$  transitions for any of these methods to be

<sup>3</sup> This is to verify the state by applying its unique signature,

applicable! For a communications protocol, for example, every input PDU of the protocol has to be a valid input at EVERY state!

Naturally, these constraints are not practical for a communications protocol; our focus. For a communications protocol, a PDU is valid only at few states and almost never at every state. Furthermore, communications protocols are seldom-deterministic FSMs but rather EFSMs. They are EFSM because there can be a transition that produces an output PDU without receiving an input. Also, there can be a transition on receiving an input without producing an output PDU, there can be a silent transition (with neither input nor output), and there can be more than one transition on receiving the same input. In communications protocols, transitions don't depend only on input but also checking predicates.

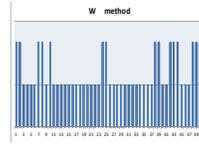


Figure 4: Coverage of the W Method

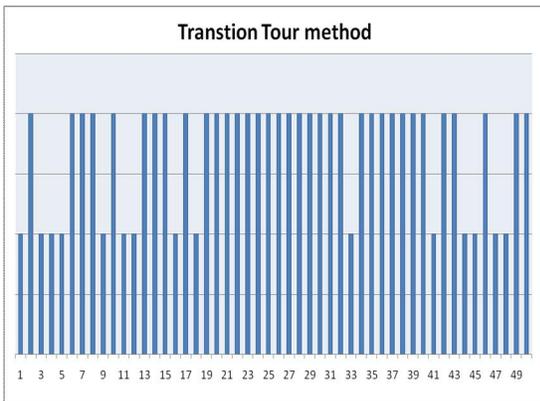


Figure 2: Coverage of the Transition Tour Method

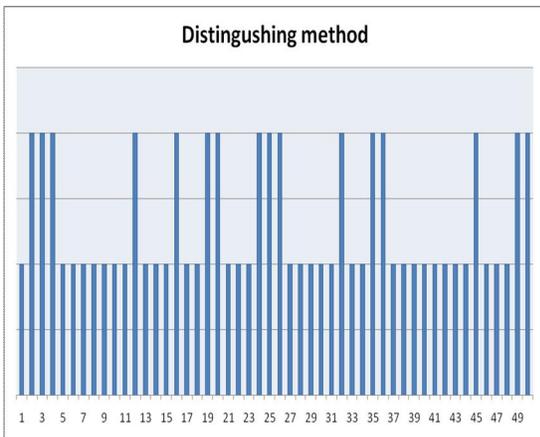


Figure 3: Coverage of the Distinguishing Sequence Method

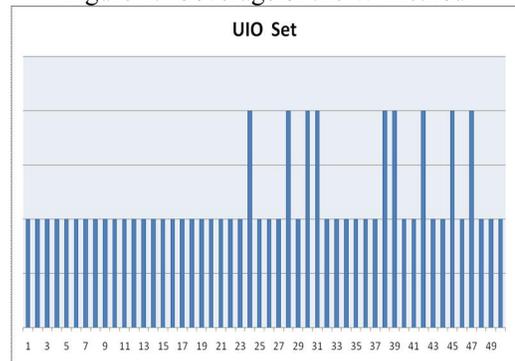


Figure 5: Coverage of the UIO Method

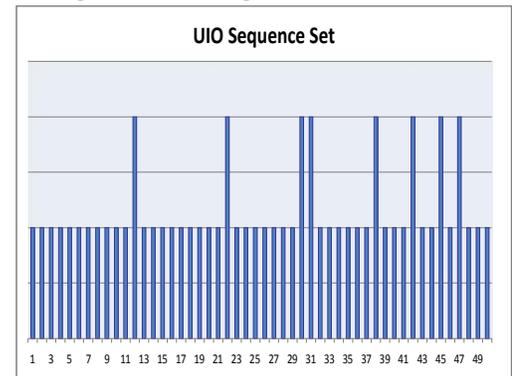


Figure 6: Test Coverage of the UIOSeqSet Method

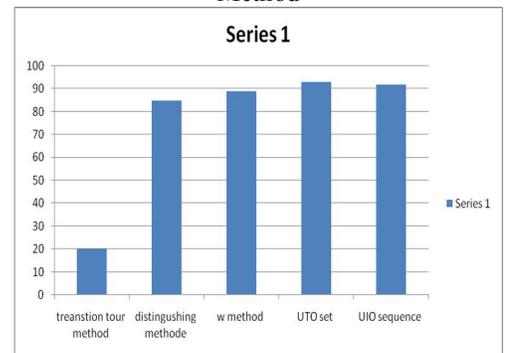


Figure 7: Statistics of the Coverage of the 5 Methods

## V. Conclusions

In this paper, we presented a simulation-based comprehensive study of the coverage of test sequence derivation methods in the literature. This is vital for the development of sound testing sequences and for the extension/generalization of existing testing methods. It is also important for the development of cost-effective testing.

Analytical remarks and investigations of the scope of applicability of these methods were also presented in this paper.

## References:

1. Hasan Ural, "Test Sequence Selection Based on Static Flow Analysis", *Computer Communication Journal*, Vol. 10, No. 5, Oct. 1987, pp. 234-242.
2. Krishan Sabnani and Anton Dahbura, "A Procedure for Generating Protocol Tests", *Proceedings of the 9th Data Communications Symposium*, Sept. 1985, Whistler, British Columbia, pp. 36-43. Also, appeared in *Computer Communication Review*, Vol. 15, No. 4, Dec. 1985.
3. S. Naito and M. Tsunoyama, "Fault Detection for Sequential Machines by Transition Tours", *Proceedings of the 11th IEEE Fault Tolerant Computing Symposium*, 1981.
4. Hazem El-Gendy, "A New Method for Deriving Test Sequences For Protocols Specified in LOTOS", *Proceedings of the International Conference on Networksp sponsored by the International Aassociation of Science and Technology for Development (IASTED)*, Orlando, Florida, USA, January 8-10, 1996.
5. D. Sidhu and T. Leung, "Formal Methods for Protocol Testing: A Detailed Study", *IEEE Transactions on Software Engineering*, Vol. 15, No. 4, April 1989, pp. 413-426.
6. Tsun S. Chow, "Testing Software Design Modeled by Finite-State Machines", *IEEE Trans. on Software Eng.*, Vol. SE-4, No. 3, May 1987, pp. 178-187.
7. Hazem El-Gendy, and Nabil El Kadhi, "Unique Input/Output Set: Universal Formal Method for Automated Testing FSM-Based Systems", *Proceedings of the 8<sup>th</sup> World Multi-conference on Systemics, Cybernetics, and Informatics sponsored by the International Institute of Informatics and Systemics (IIS)*, Orlando – Florida, USA, July 18-21, 2004.
8. Hazem El-Gendy and Nabil ElKadhi, "New Method for Testing FSM-Based Systems", *Proceedings of the 11<sup>th</sup> IEEE International Conference on Software, Telecommunications, and Computer Networks sponsored by IEEE Communications Society, Vinece-Italy, and Split-Croatia*, October 7-10, 2003.
9. Hazem El-Gendy and Nabil ElKadhi, "Towards Standardized Conformance Test Suite for ISO Transport Layer Protocol", *Proceedings of the 11<sup>th</sup> IEEE International Conference on Software, Telecommunications, and Computer Networks sponsored by IEEE Communications Society, Vinece-Italy, and Split-Croatia*, October 8-10, 2003.
10. Hazem El-Gendy and Nabil El Kadhi, "Testing Data Flow Aspects of Communications Protocols, Software, and Systems Specified in Lotos", *International Journal on Computing Methods in Science and Engineering*, Published in Greece, 2005.
11. Hazem El-Gendy and Nabil El Kadhi, "Using Formal Methods: Importance, Experience, and Comparative Analysis", *International Journal on Computing Methods in Science and Engineering*, Published in Greece, 2005.
12. Hazem El-Gendy, Mohamed Gabriel, Ahmed Samir, Narrayan Debnath, and Nabil El-Kadhi, "Towards Mosques Management Information System", Accepted for Publications in the 7<sup>th</sup> ACS/IEEE International Conference on Computer Systems and Applications sponsored by ACS, IEEE, and IEEE Computer Society, Rabat, Morocco, May 10-13, 2009.
13. Hazem El-Gendy, Nabil ElKadhi, and Narrayan Debnath, "Formal Automated Transformation of SDL Specifications to Lotos Specifications", *Proceedings of the 12<sup>th</sup> IEEE International Symposium on Computers & Communications sponsored by both IEEE Computer Society and IEEE Communications Society*, Morocco, July 6-9, 2008.
14. Hazem El-Gendy, Nabil ElKadhi, and Narayan Debnath, "Formal Automated Transformation of SDL Specifications to Estelle Specifications", *Proceedings of the 23<sup>rd</sup> International Conference on Computers And Their Applications (CATA'08) sponsored by the International Society for Computers & their Applications (ISCA)*, April 8-10, 2008, Cancun, Mexico.
15. Nabil El-Kadhi and Hazem El-Gendy, "Advanced Method for Cryptographic Protocol Verification", *Journal of Computational Methods in Sciences and Engineering*, Volume 6, Numbers 5, 6, 2006, pp. 109-119.
16. Hazem El-Gendy, "Formal Method for Automated Transformation of Lotos Specifications to Estelle Specifications", *International Journal of Software Engineering & Knowledge Engineering*, USA, Vol. 15, No. 5, Oct. 2005, pp. 1-19.

11/22/2012