DESIGN OF DYNAMIC MODEL FOR TEMPORAL PROTOCOL

Arpita Mathur[1], Ajay Mathur[2]
[1]Department of Computer Science, Lachoo Memorial College of Sc. & Tech., A-Sector, Shastri Nagar, Jodhpur (India)
[2]Department of Computer Science, Govt. Polytechnic College, Jodhpur (India)

Abstract: The need of protocols in today's environment increases as the networks explores in order to fix livelocks and deadlocks in protocol. In this paper I will explain how model based testing can be used to test the protocols that shows sequential (time based) temporal relationship between entities based on UML sequence diagram. Here dynamic model for protocol is validated and then it detects the errors which were seeded in temporal relationship.

I. EXPERIMENTAL SETUP

Validation of non-functional and functional properties of the temporal protocols during the early stages of design and development is important to reduce cost resulting from protocol anomalies, design errors like deadlock or livelock situations and/or violations of time constraints. The MBT approach is well suited to test these protocols. UML with its several diagrams supports techniques to overcome the aforementioned complexities. In this work a protocol specification mechanism is formulated for temporal (time based) protocols based on sequence diagram of UML to show how different entities interact with each other. We have considered a registration protocol here for testing the dynamic model for protocols and collected data for analysis. Figure 6.1 shows the sequence diagram for the registration protocol. Here remote client, remote server and email sender are the three entities. The client creates connection to the server. The server responds with a message to the client. During registration process email address is sent from client to the server. The server generates the password, creates a user record and then email the password to the email address. The server then sends a response to the user that the password will be emailed to him in an email.

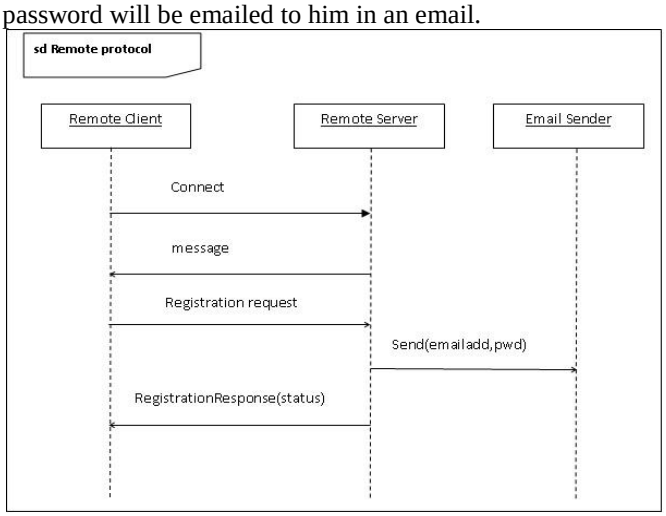

Figure 1: Sequence diagram for registration protocol

The temporal relationship between entities in registration is shown in figure 6.2. At time t_0 client sends message number 1 which is received by server at time t_1. The server then sends message number 2 to client which is received at time t_2. In a similar fashion other messages are sent by one entity and received by another shown in diagram.

Table 1 gives relational operators and the code used in table to show the temporal relationship. Temporal relationship is shown in Table 2 and table 3 is the message sequence matrix. In this work these three tables together are used for protocol specification mechanism formulated for temporal protocols. Temporal relation table and message sequence matrix are made for the registration protocol taken as example here.

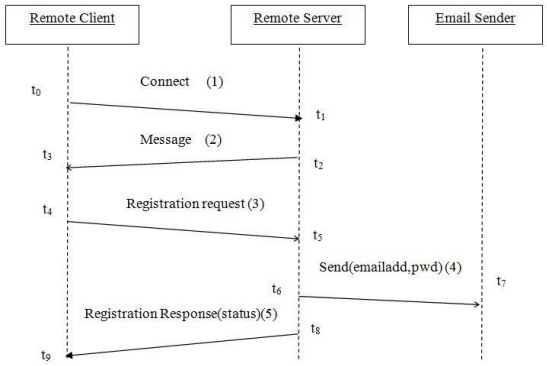

Figure 2: Temporal relation between entities

Table 1: Operator Code table

Operator	Code
X	0
= =	1
!=	2
<	3
<=	4
>	5
>=	6

Table 2: Temporal Relationship (TR)

t	0	1	2	3	4	5	6	7	8	9
0										
1	5									
2	0	5								
3	0	0	5							
4	0	0	0	5						
5	0	0	0	0	5					
6	0	0	0	0	0	5				
7	0	0	0	0	0	0	5			
8	0	0	0	0	0	0	0	5		
9	0	0	0	0	0	0	0	0	5	

Only lower triangular matrix is used in table 6.2 as other values can be found as the complements. For example, here t[1] > t[0] so t[0] <= t[1] and t[6] >t[7] then t[7] >= t[6] i.e. TR[0][1] is complement of TR[1][0] and TR[7][6] is complement of TR[6][7]. TR[2][0] depicts t[2] is independent of t[0]. X stands for don't care or independent of.

Similarly TR[4][2] and TR[8][7] depicts t[4] and t[8] are independent of t[2] and t[7] respectively.

Table 3 gives the detail of what messages are sent or received (0/1) by which entity and at what time interval. For example, the first row shows message number 1 is sent by entity 0 at time interval t[0] and second row says message number 1 is received by entity 1 at time t[1]. Similarly, seventh row shows message number 4 is sent by entity 1 at time t[6] and row six says this message is received by entity 2 at time t[7].

Table 3 : Message Sequence Matrix

Time	Entity	Msg No	Send'/Received
0	0	1	0
1	1	1	1
2	1	2	0
3	0	2	1
4	0	3	0
5	1	3	1
6	1	4	0
7	2	4	1
8	1	5	0
9	0	5	1

This message sequence matrix is input to the dynamic model for temporal protocols to send and receive messages using object threads. These threads are synchronized. We have used Java to implement this dynamic model as it supports multithreading and thread synchronization which are used to represent concurrent entities.

II. BLOCK DIAGRAM FOR THE EXPERIMENT

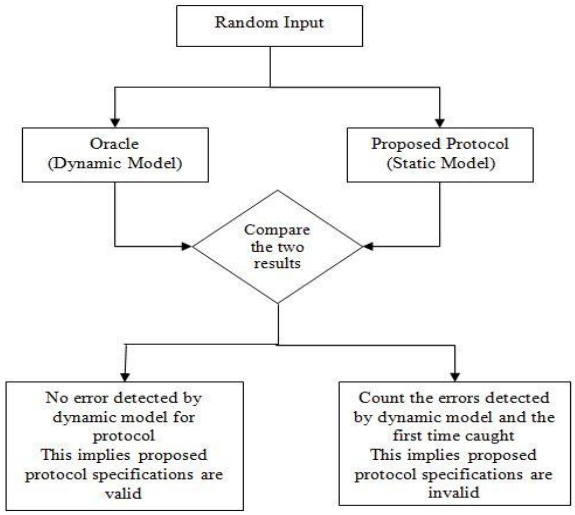

Figure 3 Block diagram for testing the proposed protocol specification model

As shown in figure 5 in this experiment we are giving random input to our proposed protocol specification and to the dynamic model for protocol (oracle). The outputs of proposed model and dynamic model are then compared. If the outputs of oracle and proposed model match it is implied that proposed protocol specifications are valid else it implies proposed protocol specifications are invalid.

III. CONCLUSION

A protocol specification mechanism was developed which gives the temporal relationships and the message sequence. The proposed dynamic model and temporal relationship may be validated through model based testing. The experiment may be conducted on protocol specification by giving random inputs to the static model/temporal relationship and dynamic model. Hence, the ability of dynamic model to validate a protocol may be proven.

REFERENCES

[1] Glenford J. Myers, The Art of Software Testing, second edition, John Wiley & Sons, Inc., 2004, ISBN 0-471-46912-2.

[2] Practical Software Testing, A Process-Oriented Approach, Ilene Burnstein, Springer-Verlag New York, Inc.,2003, ISBN 0-387-95131-8.

[3] Software Testing and Continuous Quality Improvement, Second Edition, William E. Lewis, Auerbach publications, ISBN 0-8493-2524-2.

[4] IEEE Standard Glossary of Software Engineering Terminology (Std610.12-1990), Copyright 1990 by IEEE.

[5] UML based Test Specification for Communication Systems, A Methodology for the use of MSC and IDL in Testing, Dissertation, 2004

[6] H. Balzert, Software Management, volume 2, first edition, 1998, ISBN 3827400651.

[7] Kaner, C., J. Falk, & H. Nguyen, 1999, Testing Computer Software, Wiley Computer Publishing, second edition, ISBN 0471358460.

[8] E.J. Weyuker, "On Testing Non-testable Programs", The Computer Journal, vol. 25, no.4, pp. 465—470, 1982.

[9] H. Robinson,"Graph Theory Techniques in Model-Based Testing", 16th International Conference and Exposition on Testing Computer Software,

[10] I. K. El-Far, & J. A. Whittaker, "Model-based Software Testing", Encyclopedia on Software Engineering, Volume 1. New York, USA:, John Wiley & Sons Inc, 2001. pp. 825-837, ISBN 0-471-21008-0.

[11] Mikko Aleksi Makinen, "Model Based Approach to Software Testing", Master's Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Technology May 22, 2007

