Modeling and Validation of Object-Oriented Test Case through Finite State Machine

Taskeen Zaidi
Department of Computer Science
Babasaheb Bhimrao Ambedkar University
Vidya Vihar, Rae Bareli Road, Lucknow

Vipin Saxena
Professor, Deptt. of Computer Science
Babasaheb Bhimrao Ambedkar University
Vidya Vihar, Rae Bareli Road, Lucknow

ABSTRACT
Distributed computing approach is preferred over centralized approach due to low cost involvement and for providing reliability and expandability to network. An object-oriented language Unified Modeling Language is proposed by the authors to model the dynamic behavior for execution of tasks for the digital watch under distributed environment. A UML state diagram is designed and then converted into the transition diagram for computation of valid test cases by the use of Finite State Machine (FSM). Test cases are validated for the validation of the UML state diagram. The approach for generating the test cases through FSM is very reliable and efficient and does not support for the invalid test cases.

Keywords
Distributed system, UML, State diagram, FSM, Valid Test cases

1. INTRODUCTION
Distributed computing approach is very popular now days as it offers execution of tasks without using global clock, resources sharing, data sharing audio and video sharing, reliability and scalability with low cost involvement.Various types of algorithms are available for executions of tasks under distributed environment are well explained by [1-2]. Distributed computing systems are connected statically and dynamically in network by using various kinds of topologies described by Tanenbaum[3]. Hwang[4] has described network properties, processors architecture, memory hierarchy, message passing mechanisms as well as various models of computers. Under mutual exclusion condition, the tasks are executed inside the critical section first and for this purpose necessary conditions for mutual exclusion are well explained by Milenkovic[5]. Lamport's [6] has proposed an algorithm for execution of tasks under distributed environment using time, clock and ordering of events. Authors [7-8] have used UML for analyzing the performance of parallel and distributed architecture applications. The various aspects, versions, structure of UML are well explained in [9-10]. The UML has been adopted by OMG and later on various versions appear time to time available on [11]. The executions of tasks in reflexive, symmetric and transitive manners under distributed environment explained by Zaidi and Saxena [12]. Saxena and Arora [13] have analyzed the performance of object-oriented systems. The execution of tasks under distributed environment as well as validation of UML model has been analyzed by Zaidi and Saxena [14].

In the present work authors have used a well known object-oriented modeling language i.e. Unified Modeling Language to construct a state model for execution of processes under distributed environment for the digital watch system. UML state diagram is converted into FSM and various valid test cases are also designed for the validation of the proposed UML state model.

2. BACKGROUND
2.1 A Distributed system
A distributed system is an autonomous collection of devices which provides resources sharing, data sharing, audio-video sharing, scalability and reliability to a network with low cost involvement. Processes communicated with each other by message passing technique and these systems do not share global clock. In the following figure PP_1, PP_2,...,PP_N and PQ_1, PQ_2,...,PQ_M are the different processors that have different processing speed while on the other hand MP_1,MP_2,...,MP_N and MQ_1,MQ_2,...,MQ_M are different memories attached with the corresponding processors.

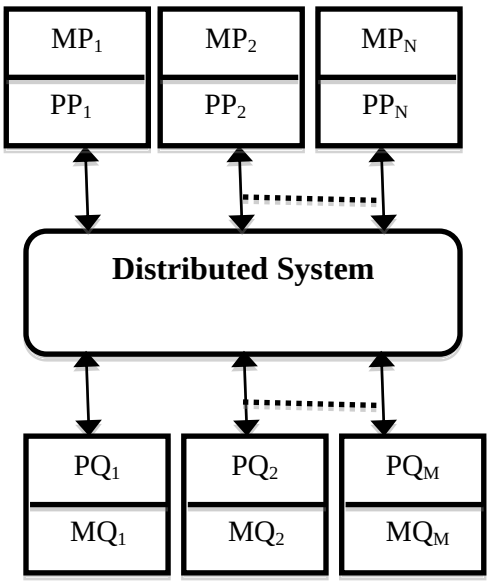

Fig. 1: A distributed computing system

The following conditions are applicable for the distributed computing system:

1. Computer systems attached under distributed environment share their own local memory, cache and processor;
2. These systems do not share global clock;
3. Every task is executed in distributed environment according to clock condition, having timestamp and ordering of events.

2.2 PROCESS MIGRATION

In a distributed computing approach process migration is related to transmission of process from one processor to another for sharing of resources according to availability of processors. Process migrates from processor 1 to another processor 2 using write back cache and in the middle process migrates from processor 2 to processor 1 using write through cache. Both the cases cause inconsistency; a coherent protocol must be established before one process migrates to another. The inconsistency can be controlled by write invalidate and write update policy.

3. A UML STATE DIAGRAM

State diagram represents the dynamic behavior of a system. It consists of the state and the events are grouped together to form a state. A UML state diagram for the digital watch system is given below and in this diagram, users press the buttons to set hours, minutes and seconds as well as increments hours/minutes/seconds in digital clock. The current time in hours, minutes and seconds are displayed as output and represented through Display_time.

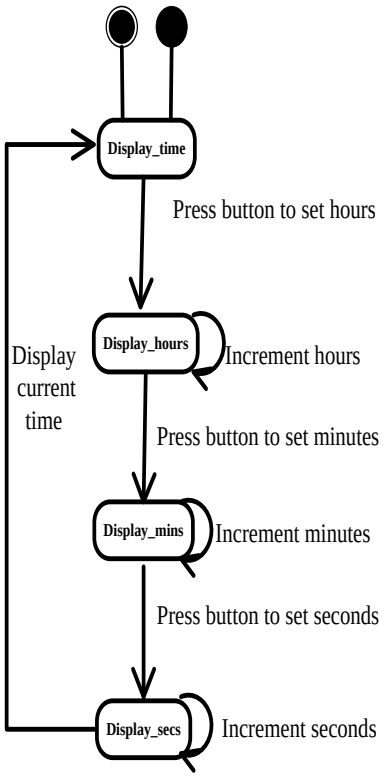

Fig. 2: A UML state diagram

Table 1. List of Events for Transition Table

a	Press button to set hours
b	Increment hours
c	Press button to set minutes
d	Increment minutes
e	Press button to set seconds
f	Increment seconds
g	Display current time

All the states are represented as q_0,q_1,q_2 and q_3 and seven events are represented in the above table. A UML state diagram is converted into the state transition diagram and represented below:

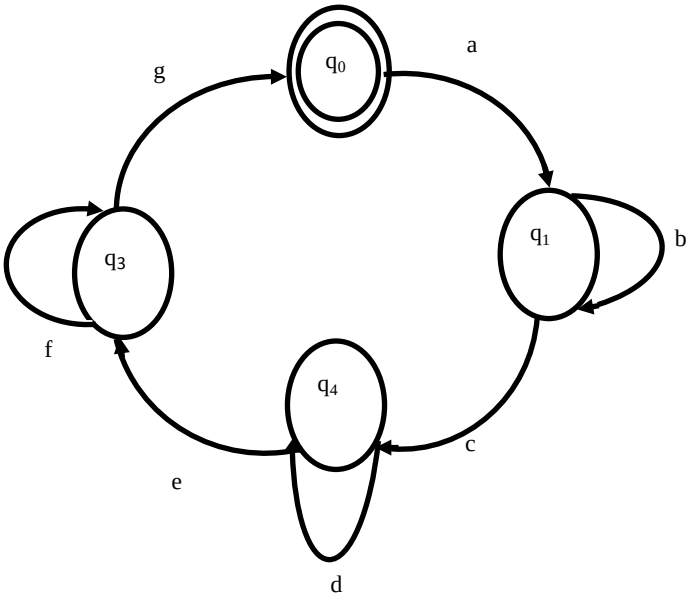

Fig. 3: A state transition diagram

From the above, an equivalent grammar is generated below:

$\delta(q_0,a) = q_1 \quad \Rightarrow \quad q_0 \rightarrow aq_1$
$\delta(q_1,b) = q_1 \quad \Rightarrow \quad q_1 \rightarrow bq_1$
$\delta(q_1,c) = q_2 \quad \Rightarrow \quad q_1 \rightarrow cq_2$
$\delta(q_2,d) = q_2 \quad \Rightarrow \quad q_2 \rightarrow dq_2$
$\delta(q_2,e) = q_3 \quad \Rightarrow \quad q_2 \rightarrow eq_3$
$\delta(q_3,f) = q_3 \quad \Rightarrow \quad q_3 \rightarrow fq_3$
$\delta(q_3,g) = q_0 \quad \Rightarrow \quad q_3 \rightarrow g$

From the above grammar, the events are organized into the transition table, ϕ is considered as null and table consists of four states and seven events.

Table 2. List of Events for Transition Table

δ/Σ	a	b	c	d	e	f	g
q_0	q_1	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ
q_1	ϕ	q_2	q_2	ϕ	ϕ	ϕ	ϕ
q_2	ϕ	ϕ	ϕ	q_2	q_3	ϕ	ϕ
q_3	ϕ	ϕ	ϕ	ϕ	ϕ	q_3	q_0

From the above table the following test cases are derived

Valid Test Case 1

If desired hours are incremented

For this, the equivalent grammar is given below:

$q_0 \rightarrow aq_1$

$q_1 \rightarrow bq_1$

By replacing the non-terminals on RHS, the following is concluded which shows that the desired hours are incremented.

$q_0 \rightarrow abq_1$

Valid Test Case 2

If desired minutes are incremented

For this, the equivalent grammar is given below:

$q_1 \rightarrow cq_2$

$q_2 \rightarrow dq_1$

By replacing the non- terminals on RHS the following is concluded which shows the desired minutes are incremented.

$q_1 \rightarrow cdq_2$

Valid Test Case 3

If desired seconds are incremented

For this, the equivalent grammar is given below:

$q_2 \rightarrow eq_3$

$q_3 \rightarrow fq_3$

By replacing the non- terminals on RHS the following is concluded which shows the desired seconds are incremented.

$q_2 \rightarrow efq_3$

Valid Test Case 4

If current time is displayed on digital watch

For this, the equivalent grammar is given below:

$q_0 \rightarrow aq_1$

$q_1 \rightarrow bq_1$

$q_1 \rightarrow cq_2$

$q_2 \rightarrow dq_2$

$q_2 \rightarrow eq_3$

$q_3 \rightarrow fq_3$

$q_3 \rightarrow g$

By replacing the non-terminals on RHS the following is concluded which shows that the desired time is displayed on digital watch.

$q_0 \rightarrow abcdefg$

4. CONCLUDING REMARKS

From the above work, it is concluded that the software professionals are using the object-oriented modeling language for designing purpose and developing the various kinds of the software designs. The presented approach is very useful for the generation of the valid test cases from the object-oriented software designs, from the test cases one can find the errors during the early stages of software development or before writing the coding of the software design.

5. ACKNOWLEDGMENTS

Our thanks are due to the University Grants Commission, New Delhi, India.

6. REFERENCES

[1] Siberschatz, A., and.Galvin, P.B, 2000, Operating Systems Concepts, 5th Edition, John Wiley and Sons, Inc. D.D.L.L.D.

[2] Siberschatz, A., and Peterson, J. L, 1988, Operating System Concepts, Addison–Wesley, Alternate Edition.

[3] Andrew S. Tanenbaum, M., 1995, Distributed Operating Systems, Prentice Hall.

[4] Hwang, K., 1993, Advanced Computer Architecture, McGraw-Hill Series in Computer Engineering, Inc Publishing.

[5] Milenkovic, M., 1997, Operating Systems: Concepts and Design, TataMcgraw-Hill.

[6] Lamport, L., 1978, Time, Clocks and Ordering of Events in a Distributed System, Communications of ACM, Vol. 21,No.7, pp.558-565.

[7] Pllana, S., and Fahringer, T., 2002, On Customizing the UML for Modeling Performance Oriented Applications. In <<UML>>, Model Engineering Concepts and Tools, Springer-Verlag, Dresden, Germany.

[8] Pllana, S., and Fahringer, T., 2002, UML Based Modeling of Performance Oriented Parallel and Distributed Applications, Winter Simulation Conference, Retrieved on June 19, 2012.

[9] Booch, G., 1994: Object-Oriented analysis and Design with applications, Second Edition Addison Wesley.

[10] Booch, G., RambaughJ., and JacobsonI., 1999: The Unified Modeling Language, User Guide Addison Wesley, Raeding MA.

[11] OMG 2001: Unified Modeling Language Specification, Available Online via http://www.omg.org (Accessed on 12 Nov. 2013).

[12] Saxena, V.,and Zaidi, T., 2012, "Modifications in Lamport Algorithm for Distributed Computing System", International Journal of Computer Applications", Vol. 53(6).

[13] Saxena, V., and Arora, D., 2009, Performance Evaluation for Object-oriented Software Systems, SIGSOFT Software Engineering Notes, Vol. 34, No. 2, 1-5, doi:=http://doi.acm.org/10.1145/1507195.15072-13.

[14] Zaidi, T., and Saxena, V., 2013, Modeling and Validation of Execution of Tasks on National Knowledge Network under Distributed Environment, SIGSOFT Software Engineering Notes, to be appeared in May, 2013.