ISSN:0975-9646

GENERATION OF TEST CASES USING UML SEQUENCE DIAGRAM IN A SYSTEM WITH COMMUNICATION DEADLOCK

DebashreePatnaik[1],Arup Abhinna Acharya[2], Durga Prasad Mohapatra[3]
[1,2]*School of Computer Engineering,KIIT University,Bhubaneswar, India*
[3]*Department of Computer Science Engineering National Institute of Technology,Rourkela, India*

Abstract— **An environment in which different processors communicate with each other speeding up computation and improving the data availability is called as distributed environment. Communication and concurrency are the major issues of the distributed environment. Different processors try to communicate with each other avoiding deadlock. The resource contention which is introduced by the concurrent process in distributed computational environment gives rise to deadlock problem. In this paper an effort has been made by the authors to represent deadlock situations with the help of graph. An algorithm is devised to reveal deadlock has occurred. A real time banking system is depicted setting out to be an example. Test cases for the communication deadlock and the UML Sequence Diagram are laid out from the real time example**.

***Keywords*— Distributed System, deadlock, Sequence Diagram**

1.INTRODUCTION

A Distributed system is a collection of processors that do not share the memory or clock. Each processor has its own local memory and the processors communicate with each other through communications lines such as local area or wide area network. The processors in distributed system vary in size and function. The distributed system includes giving users access to the resources maintained by the system and there by speeding up computation and improving data availability and reliability. If the system is distributed it provides the mechanism s for process communication for dealing with the deadlock problem.

The distributed environment consists of n number of processes, each of which resides at different processor. The processes are numbered uniquely from 1 to n and one to one mapping exists between the processes and the processors. Synchronization in the distributed environment takes place, when a process p_i wants to enter its critical section, it generates a timestamp(t_i) and sends a message request (p_i,t_i) to all processes in the system. On receiving a request message a process may reply immediately (sends an acknowledgement to p_i) or it may defer sending a message. A process that has received a reply message from all other process in the system can enter its critical section, queuing incoming requests and deferring than existing its critical section, the process sends a reply message to its entire deferred request. In the distributed environment processes are concurrent. For concurrent processes synchronization and dead lock are two common issues.

The resource contention which is introduced by the concurrent process in distributed computational environment gives rise to deadlock problem. A deadlock is a persistent and circular wait condition where each process involved in the deadlock waits indefinitely for resources held by other processes while holding resources needed by others. Thus none of the otherprocess waiting for needed resources can continue computation without obtaining the waited for resources. Deadlock has adverse performances effects that offset the advantages of resource sharing and processing concurrency.

This paper proposes a method to investigate deadlock through link list and generate the test cases in the distributed environment for conformance of deadlock. A sequence diagram is drawn from the real life deadlock example, where communication deadlock has occurred. From the sequence diagram the wait – for – graph for the system is generated, and from this intermediate form the test cases are generated.

The rest of the paper is organized as follows: Section II describes the basic concepts, Section III focuses on the proposed approach, SectionIV highlights the test case generation , Section Vrepresents conclusion and future work.

II.BASIC CONCEPTS:

System consists of finite number of resources to be distributed among various computing process. Resources are usually partitioned into different types consisting number of identical instances. Memory space,CPU cycles, files messages are all resource types. If a process requests an instance of resource type, the allocation of any instance of the type will satisfy the request. If it will not then the instances are not identical and the resource type classes have not been defined properly. A process must request a resource before using it and must release the resource after using it. A process can request as many resources as it requires tocarry out the designated task. A process cannot request for three printers if there are two printers. A resource is utilized in following sequence:

a) Request: Process request the resource. If request is not granted immediately then the requesting process must wait until it can acquire the resource.

b) Use: The process can operate on the resource.

c) Release: The process can release the resource

The resources can be physical resource or logical resource. The necessary conditions for deadlock are

i).*Mutual Exclusion:* In non sharable mode at least one resource must be held, i.e only one process at a time can use the resource.

ii).*Hold and Wait:* A process must be holding at least one resource and waiting to acquire another resource.

iii).*No Preemption*: Resources cannot be preempted,that is a resource can be released only voluntarily by the process holding it.
iv).*Circular Wait:* A set of p_0,p_1,p_2 ……….p_n of the waiting process must exit such that p_0 is waiting for a resource held by p_2,p_3…………..pn-1 is waiting for a resource held by p_0.

III.PROPOSED APPROACH

Single instance of resource type means, the system consisting of only one resource for one type. Deadlock is detected with the help of wait for graph. A wait for graph is a graph derived from the resource allocation graph. It consists of process as vertices.
An edge from p_i to p_j in a wait for graph implies that a process p_i is waiting for process p_j to release a resource that p_i needs. An edge p_i to p_j exits in a wait for graph if and only if the corresponding resource allocation graph contains two edges p_i to r_q and r_q to p_j for some resource R_q. A system is in deadlock state if and only if the wait for graph contains cycle(assuming single instance of resource).

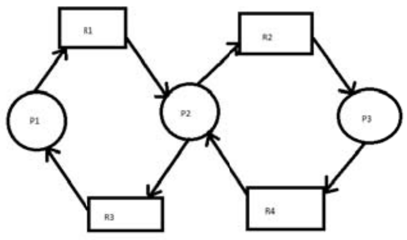


FIGURE.1.RESOURCE ALLOCATION GRAPH

In figure1. There are two cycles one is p_1 to p_2 to p_1, secondis p_2 to p_3 to p_2.

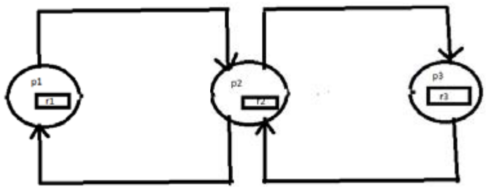


FIGURE2.WAIT FOR GRAPH

In the proposed method, deadlock is detected through the use of link list. The reference address of resource r_1 is 210, reference address of r_2,r_3 is 212, 214 respectively. The various information of a process running at different sites can be stored inform of link list. In figure3 the process p_i,p_j,p_k are distributed and executing concurrently. P_i holds a resource r_1 but is waiting to acquire the resource r_2 which is hold up by the process p_2. P_2 is in waiting state for resource r_3 which is already acquired by p_3. P_3 is waiting for resource r_1 which is there with the process p_1. Here all the processes are in hold and wait condition giving up a circular loop causing deadlock. The process information at each site is depicted in each node and deadlock is viewed through the link list even.

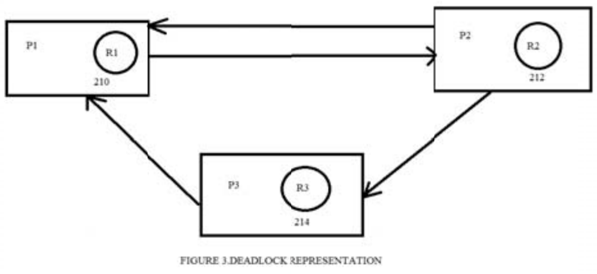
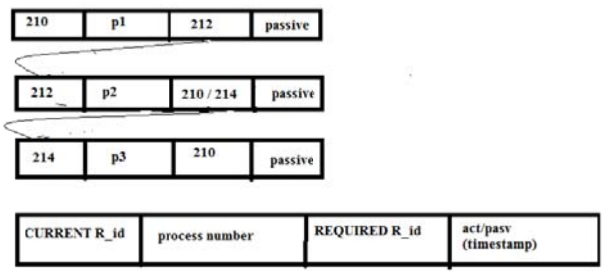


FIGURE 3. DEADLOCK REPRESENTATION



210	p1	212	passive
212	p2	210 / 214	passive
214	p3	210	passive
CURRENT R_id	process number	REQUIRED R_id	act/pasv (timestamp)

FIGURE.4.LINKED LIST REPRESENTATION.

CURRENTR_id: Represents the current id of the resource that it is already holding
Process number: Represents the process id and the site in which the process is being executed.
REQUIRED R_id: Represents the required id of the resource that it wants to acquire.
Active/Passive: Represents whether the communication is in active form or passive form.
Timestamp: Represents for the amount of time the processes has been in waiting state to acquire the resource.

ALGORITHM FOR CONFORMANCE OF DEADLOCK

DEADLOCK_LL:
Input: Information about each process and resource at all site
Output: Detection of deadlock

STEP1: Consider three arrays of $p_i[4]$,$p_j[4]$,$p_k[4]$ of structure type.p_i,p_j,$p_k$are the three processes executing at different sites and $p_i[4]$,$p_j[4]$,$p_k[4]$ are the process information stored over the list.

STEP2: For each process, all the index of respective arrays have to be filled. For example for process $p_i[4]$.

```
For( i=0;i<=3;i++)
{
X=curr_id;
Y=process_id;
Z=req_id;
If(pi== active)
Acive=1
Else
Passive=0
Timestamp(value);
}
```

STEP3: Similarly for process pj and pk the process index information are
X_1,Y_1,Z_1 for P_j and X_2,Y_2,Z_2 for p_k.

```
STEP4: if (Z==X1) && (Z1==X2)
{
If(X==Z2)
Conformance of DEADLOCK
else
SAFESTATE
}
```

A CASE STUDY: BANKING APPLICATION:
An example of the banking system is depicted below. Here different applications are considered as different processes as p_1,p_2,p_3,p_4,p_5 with their resources as r_1,r_2,r_3,r_4,r_5.

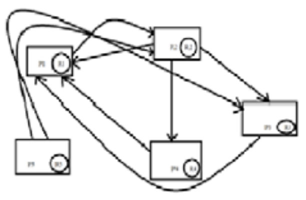


FIGURE.5.CASE STUDY

TABLE.1. PROCESSES AND RESOURCES

Processes	Resources
P1 – Deposits application	R1 – Account details table
P2 – Customer application	R2 – Customer details table
P3 – Credit/Debit Card Application	R3 – Credit/Debit card details table
P4 – Loan application	R4 – Loan details table
P5 – ATM application	R5 – ATM Information table

In a Bank we have thousands of different processes/applications executing using the thousands of available resources. We take the example of 5 such processes and resources. The description of the processes and resources:
P1- *Deposits application*: This deals with the all types of account details. Its resource R1 will maintain information such as: Account type, balance, account holder name, other attached accounts, to name a few.
P2- *Customer application:* This deals with details of the entire bank's customers. Its resource R2 will maintain information such as: Customer name and personal details, addresses, different types of accounts, loans or cards he is holding, other mandatory information of the customers.
P3- *Credit/debit card application*: This deals with details of all the Credit and debit cards issued by the bank. Its resource R3 will have information such as: Card holder's name, card type, different reward options available with the cards, available balance, interest details and payment due date, to name a few.
P4- *Loan Application:* This deals with the details of all the loans that have been sanctioned by the bank. Its resource R4 will be having information such as: Type of loans, Loans recipients name and other mandatory details, loan amount, instalment details, due amount and interest details.
P5- *ATM application:* This deals with the details of the all the ATM's installed by the bank in a particular region. Its resource R5 will be having information such as: ATM codes, each ATM transaction details and history.
On a given day there will be many processes or transactions with the aid of different applications and resources available in the Banks IT Operating System. To name a few of the transactions, applications for opening of new accounts, application for new loans and credit/debit cards, Deposit of salary in customers salary account, deduction of loan instalments or credit card dues, thousands of ATM transaction from different ATM machines, etc.

As there are many processes and resources distributed at different sites in the network, the processes need to communicate with each other for computational work. The communication can be depicted in terms of sequence diagram. The sequence diagram is the best way to represent communication between different processes. Figure .6 represents sequence diagram.
Let us assume a customer Mr.X, who has 2 accounts in our bank, account#1 & account #2. Today Mr.X has come to bank with 2 applications one for closing the account#2 and 2nd one for transferring the available balance in that account to be transferred to account account#1, which is a joint account with his wife.

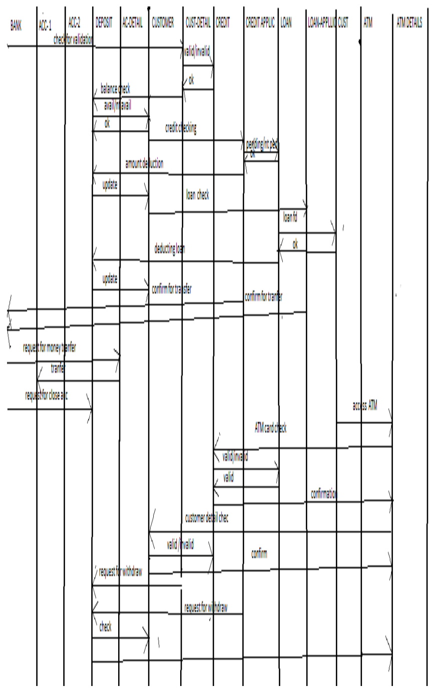


FIGURE.6. SEQUENCE DIAGRAM OF BANK TRANSACTION

IV.TEST CASE GENERATION:

The following are the test cases generated from the real time banking system. The real time banking system works as follows:

1. First theBank needs to check if Mr.X is a valid customer, for this the process P2 is initiated using the resource R2 to check details for Mr.X. Validation successfully completed.
2. Then P2 triggers P1 to refer the resource R1 to check the available balance in account#2. A balance is available for transfer.
3. Now the process P2 triggers P3 (using resource R3) and P4 (using resource R4) to find out if any Credit cards, debit cards or loans are attached to the account#2. A Credit card, a debit card and a personal loan are found attached to the account#2.
4. Process P3 checks for any payment due on the credit card. Results show a due amount of Rs.5000/-.
5. Now P3 triggers process P1 for deducting the due amount from account#2. Amount is available and Rs5000/- is deducted and the same is updated in resource R1.
6. Process P4 checks for any instalments pending of the attached personal loan. Results show one instalment of Rs.10000/- is pending on the loan.
7. Now P4 triggers process P1 for deducting the last due instalment from account#2. Amount is available and Rs.10000/- is deducted and the same is updated in resource R1.
8. Processes P3 & P4 returns back confirmation for closure of the account. P1 process is initiated to transfer the remaining balance from account#2 to account#1.
9. P1 triggers P2 to check the validity of the account and the account holder information. Account#1 is a valid and open account and joint account holders are Mr.X and Mrs.X.
10. Once the resource R1 confirms that the entire amount from account#2 has been transferred to account#1, P1 starts the transfer of balance from account#2 to account#1 and finally closes the account#2 and updates the same in R1 and R2.
11. At the exact same momentMr.Xtries to perform an ATM transaction using the debit card attached to account#1 initiating process P5.
12. Now P5 triggers P2 (using the resource R2)& P3 (using resource R3) to authenticate the customer& card details respectively. Both P2 & P3 successfully validate the authentication.
13. .Now P2 & P3 together trigger process P1 for withdrawal of money. This needs the use of the resource R1.
14. But simultaneously as a part of the account#2 closer procedure its remaining amount is getting transferred to account#1 and is updating the resource R1, resulting in the ATM transaction initiated by P5 to wait for resource R1 and may result in timed-out.

Deadlock Situations:

Now we have two situations, one in which the deadlock is avoided by reaching a Safe state and another in which the transaction failed because of deadlock.

1.The 1st deadlock situation could have come at step 3, when the P2 triggered P3 and P4, and later both P3 and P4 triggered P1 simultaneously to deduct the due amounts. If the available balance in the resource R1 was not enough then either one of the P3 or P4 transactions could have failed. But as the balance was available, a SAFESTATE was reached to avoid any deadlock.

2.The 2nd deadlock situation arrived at step 11, when Mr. tried to perform an ATM transaction using the debit card of account#1. This needs resource R1 for withdrawal. But simultaneously as a part of the account#2 closer process the entire amount from account#2 is getting transferred to account#1 and is updating the resource R1. As both debit and credit information cannot be updated simultaneously (as this will cause out-of-balance in the bank records), the ATM process P5 has to wait for R1 and may get timed out.

V. CONCLUSION AND FUTURE WORK

Problems that arise from concurrency and deadlock are discussed in context to distributed environment. The paper emphasizes on testing the behavioral aspects of the distributed objects using communication to stimulate the object's behavior. A method of deadlock analysis is presented through link list and UML Sequence Diagrams. The Sequence diagram is of a bank system. From the real life example of the bank system we have generated the test cases. The generated test cases can be optimized with the help of Genetic algorithm. The authors have plan to to optimise the test cases with Genetic algorithm The results of the applied genetic algorithm can be viewed by implementation in Matlab.

REFERENCES

[1] *A. Hessel, K.G. Larsen, M.Mikucionis, B.Nielsen, P. Pettersson, and A. Skou,"Testing real-time systems using UPPAAL.", In Proceedings of Formal Methods and Testing, volume 4949 of LNCS,*pages 77–117. Springer, 2008.
[2] *A. Bader, A. S. M. Sajeev, S. Ramakrishna,"Testing concurrency and communication in distributed object".*
[3] *H. R. Asaadi, R. Khosravi, M. Mousavi3, N. Noroozi2," Towards Model-Based Testing of Electronic Funds Transfer Systems".*
[4] *R. M. Hierons, Kil. Bogdanov, J.P. Bowen, R.Cleaveland, J.Derrick, J.Dick, M.Gheorghe, M. Harman, K. Kapoor, p. Krause, G.L¨uttgen, A. J. H. Simons, S. A. Vilkomir, M.R. Woodward, and H. Zedan,- " Using formal specifications to support testing. ACM Computing Surveys, 41(2), 2009".*
[5] *P. W. M. Koopman and R. Plasmeijer,-" Testing reactive systems with GASTIn Post-Proceedings of TFP". 2003, pages 111–129, Intellect, 2004.*
[6] *M.Mikucionis, B.Nielsen, and K. G. Larsen.,"Real-time system testing on-the-fly." In Proceedings of NWPT'2003, pages 36–38, 2003.*
[7] *D. Harrell," Executable object modeling with state charts". IEEE Computer, pages 31–42, July 1997.*