UML Modeling of Generic Agent Database Approach under Distributed Computing Environment

Vipin Saxena, PhD.
Department of Computer Science
B.B. Ambedkar University
(A Central University)
Rae Barely Road, Lucknow-25, U.P.
(India)

Nimesh Mishra
Department of Computer Science
B.B. Ambedkar University
(A Central University)
Rae Barely Road, Lucknow-25, U.P.
(India)

ABSTRACT

Due to presence of high speed bandwidth network called as National Knowledge Network (NKN) environment, the communication between the two computer systems by message passing and data transmission techniques have improved significantly. The existence of heterogeneous network environment has different configuration supporting network. In such type of networks, the process execution depends upon several factors related with the hardware like cpu speed, clock rate, memory size, etc. and software events like execution scenario of processes based on software programming and database approach results in a best optimum performance of the network. One of such kind of approach has been proposed by the authors and a model has been designed for the general process execution concept through distributed database agent supported in distributed computing environment to obtain optimum process execution. This approach has been designed by considering two phases; one phase based on the client node and other is based on controller system. The model has been created by the use of well known Unified Modeling Language (UML) and UML class, sequence and activity diagrams are designed.

General Terms

Generic System, Distributed Approach

Keywords

Performance, Distributed Database, Distributed Computing, Agent, Generic, UML Class, Sequence, Activity Diagrams.

1. INTRODUCTION

Emphasis is given on performance of process execution over a high speed distributed computing environment. Execution of a process needs several factors like cpu cycle, clock rate, processing speed, throughput required, scheduling scheme, process priority, process size, etc. These factors have individual importance towards the process execution. Lots of research work has already been done to improve the performance execution of the process. But these works are generally based on some specific criteria of the above mentioned factors. In the proposed work, authors have proposed a new fully generic approach which is based on all the aforesaid factors and proposed a model in which the general factors are responsible for the process execution in the form of agent program for a distributed database which has been created related to the several important factors as mentioned above. Several factors related with process execution are well explained by Hwang[1]. These factors are like cpu cycle, clock rate, throughput required, etc. Generic approach has also been applied on the designing of network architecture [2]. Distributed computing environment has been discussed and several models have been proposed for the process execution in the distributed computing environment [3]. Proper process scheduling is the most important task as it decides which process should be allowed to use the resources and for how much time in such a manner that there should not occur any type of conflict or deadlock [4-5]. Process is attached with a Process Control Box called as PCB [6]. The life cycle of the process is explained by Milenkovic. One of major aspects of the process execution in distributed computing environment is the performance issue. Load balancing has been described in order to perform the process execution in the distributed computing environment [7]. Generic approach has also been applied in the distributed database concept in order to check the integrity constraint [8] through the use of agent based approach. Transactions management in real time distributed computing system is also one of the issues in heterogeneous networked computing environment. A new approach to manage the transactions in dynamic ways has been elaborated and dynamic intelligent agent has been created which keeps tracks of timing of the transactions [9]. Performance Evaluation of well known object-oriented programming languages on dual core also on Pentium processors has been done and results have obtained that reports performance estimation for object-oriented software systems [10-11]. Observation of literature states that first time Unified Modeling Language (UML) is proposed in the year 2002 for parallel and distributed applications i.e. in the field of Advanced Computer Architecture by Pllana and Fahringer [12, 13]. Along with this object-management group, OMG [14, 15] group has released UML specification after the development of various kinds of UML diagrams by Booch et al. [16, 17]. The present work is the motivation from the literature and idea to develop a new kind of generic approach for the process execution based on distributed database concept. Authors have proposed an agent base concept which collects the hardware configuration of the interconnected heterogeneous systems and also there is a distributed database approach which records the data regarding the need of the user's processes. The whole database structure is kept under the distributed network environment. There is a controller which performs the task of manipulation of database fields and current configuration status of the process which is going to be executed.

2. BACKGROUND

2.1 Distributed Computing

The network architecture has been classified into two phased i.e. centralized and distributed approach of computing. Distributed approach is complex but it is most robust and has high reliability in comparison of the centralized system. Distributed approach is transparent to that of end user but in this, the process execution takes place in a fully distributed manner, i.e. request given by a program in one node collects information from the other node and results the final output is given on the first node and distributed approach is elaborated in the following figure:

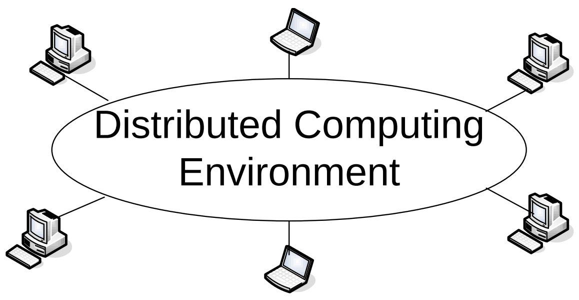

Fig 1: Distributed computing environment

2.2 Process

A process is explained in the form of subtask, subprogram, collection of lines of code, macro or subroutine, etc. The entire process is controlled by its id called as Process_ID. Detail information of process and its different status is stored in a data structure known as Process Control Block. Whenever a process gets activated its PCB is also created and helps in keeping track of process till the completion of job in form of process termination. The attributes and operations on process are grouped together and put in the form of UML class as represented in following figure 2:

PROCESS
Process_ID: Integer Process_Size: Integer Process_Priority: Integer Process_In_Time: String Process_Out_Time: String

Process_Create()
Process_Ready()
Process_Execute()
Process_Suspended()
Process_Resume()
Process_Terminate()

Fig 2: Class diagram of process

2.3 Distributed Database

Pleas Distributed database is considered as a database management in which whole of the database is stored on multiple computers which are interconnected with each other. This distributed database concept is implemented in two different manners one is by replication, here (copies of data) are created and placed on several nodes and other is in the form of fragments in which parts of a relation are kept at different locations. Fragments can be done in two different manners. Vertical Fragments i.e. subset of columns and horizontal fragment i.e. subset of tuples. All this implementation of distributed database concept is kept transparent to the users. Distributed database approach is one of the very important aspects concerned with the proposed generic approach model. A generic database is created and is designed here in such a way that it is distributed throughout the distributed computing environment.

3. GENERIC APPROACH

In the proposed approach, authors have studied the different aspects of the process execution which individually affect the execution of the process. These factors are classified into two classes one is related with the hardware configuration like speed of processors, CPU cycle, clock rate, memory status, etc. and other class factors may include, process priority, process size, cpu scheduling, scheme, etc. In the current approach authors have designed a generic database through an agent program which stores the different process affecting factors. According to this generic database status, the process is allocated to the servers for execution. In the proposed approach, each agent program keeps on updating their nodes database. A controller program is present which is periodically interrupted by the nodes and is informed about the node status. The controller also plays the role of getting information about the process status. Once the favorable node status and process status are obtained by the controller then the selective process is propagated to get executed on that favorable node. This proposed generic approach is thus considered to be a approach which results in a load balancing under the distributed environment and its generic nature can be applied and implemented in all the cases whatever manner the user wants to executes the process. In this approach it has been considered that we have to create a general database which may be in a shared manner among the entire network cluster node or this database may have replica copies to support the distributed computing concept. The status of each node can be categorized into the following manner like ready waiting executing state along with its hardware configuration, etc. By looking this information of the database of the nodes, the controller used to send the new coming process to get executed on the specific ready node which is ready to execute the process according to user's requirement.

4. PROPOSED MODEL

The proposed model as shown in the figure.3 illustrates the whole functioning of the generic approach model for process execution in distributed computing approach. In this model a cluster of interconnected nodes is taken. An agent based collector is present which collects the configuration status ofeach node and stores it into the database. Generated process from the nodes which are in need of resources to get executed are also traced through the controller system for their primary status like size, priority, memory, programming language behavior, etc and then are classified into appropriate groups according to the user's requirements. These classified processes are passed to the controller which further compares these processes with the created database which retains the configuration status of the nodes. The whole process results in the appropriate nodes selection for the appropriate process to get executed.

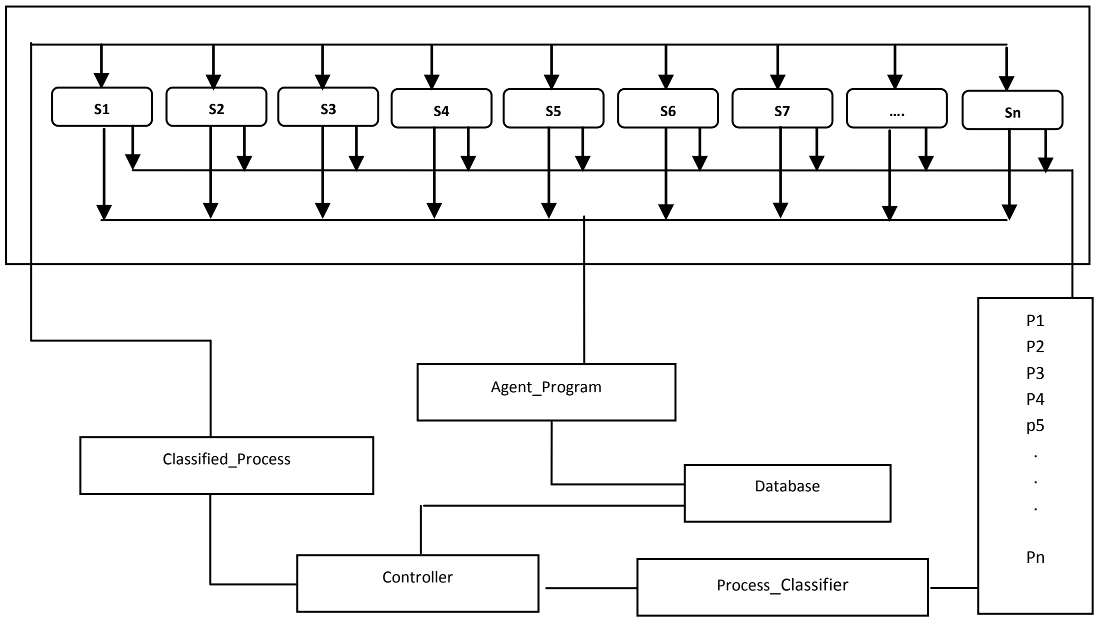

Fig 3: Process execution model for generic approach

5. UML MODELING FOR PROPOSED GENERIC DATABASE APPROACH

5.1 UML Class Diagram

Communication in distributed computing environment takes place by message passing mechanism in the form of signals. Proposed UML class model for the generic database concept for process execution is shown as in figure 4. In the proposed UML class diagram, Controller class is the main class and is responsible for controlling the whole functioning. Controller class interacts directly with Agent class, Fitness_ Comparator class and the Process_Scattered class along with User class and Process class. Process class interacts consecutively with the Environment class, Configuration class, Process_Detail class and with the Filteration_Block class. The Environment class, Configuration class and Process_Detail class performed the task of setting the criteria of process and then the processes are classified with the Filteration_Block class which interacts with the User class. Server class interacts with the Controller class and Agent class which collect the current status of the configuration of the nodes.

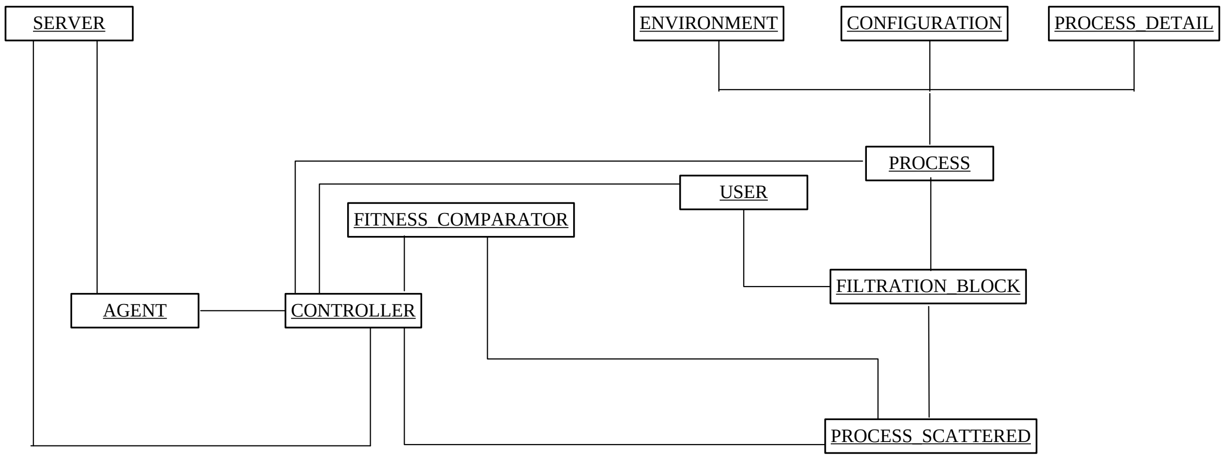

Fig 3: Process execution model for generic approach

5.2 UML Sequence Diagram

In the proposed generic approach, Process object interacts with the Controller object and informs its execution behavior. Controller object also collects the configurations detail of the node through Agent object which is responsible for getting the hardware configuration of the interconnected nodes. The Controller classifies the processes according to their specified criteria by interacting with the Filtration_Block object through User object. Then, the entire Controller compares the process current status with the record provided by the Agent and then the fittest process scenario is obtained by the Fitness_Block object.

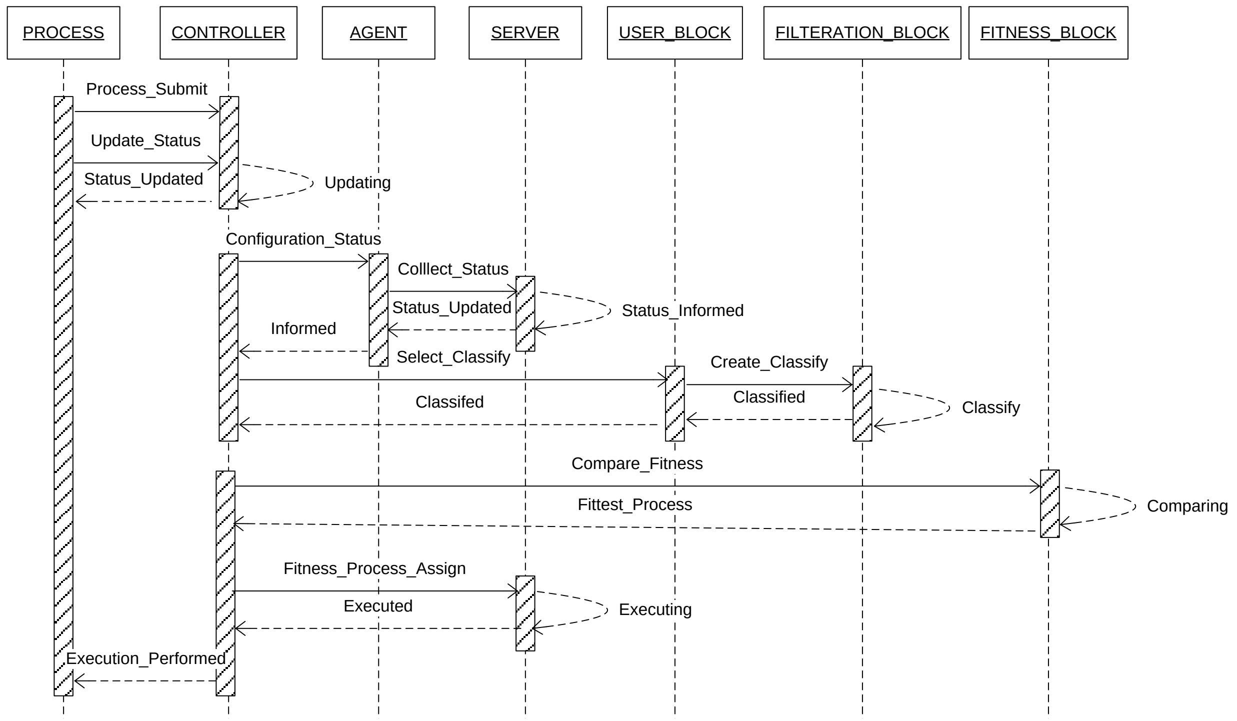

Fig 5: UML sequence diagram for generic approach

6. EXPERIMENTAL STUDY

Authors have used the generic approach for process execution in a very efficient manner. The whole mechanism is based on two aspects of the distributed computing environment. These are the creation of the general database which keep the record of hardware configuration of nodes and creation of the data structure using controller which collects the process status. In the experimental study of the proposed approach a cluster of 20 interconnected network systems, having different hardware configuration has been selected which form a distributed network environment. Let S1, S2, S3,….., Sn are the set of servers interconnected with each other. First of all an agent module program is loaded by default in the operating system of each server. A general database has been created by this agent program by recording the configuration status of each of these servers individually. Along with this a controller program also executes, that collects the information of the processes going to be executed. Then the controller evaluates the status of the generic distributed database record with that of the processes behavior and finally allocates the process to the appropriate node for its execution. In the proposed approach authors have tried to provide generic approach for the process execution according to generic criteria of program behavior as set by the programmer. Execution time i.e. CPU time is calculated by controller according to above discussed parameters for appropriate process allocation at node where it could get executed.

The CPU time i.e. execution time is given as

$$T=I_C*CPI*\gamma \qquad ...….. (I)$$

Where CPI is given as

$$CPI=(p+m*k) \qquad ……. (II)$$

T =CPU time
I_C = Instructions count of program code
γ =Processor cycle time
CPI=Cycles per instruction needed
p =Processor cycle needed for instruction decode and execute
m =Number of memory references needed
k =Ratio between memory cycle and processor cycle

Language	VC#			JAVA		
Lines of Code	10^2	10^3	10^4	10^2	10^3	10^4
Execution Time in Milliseconds	78	390	1500	96	406	3703
	78	359	1890	93	406	3718
	93	390	1500	94	390	3781
	93	375	1921	93	390	3703
	78	375	1468	92	407	3766
Avg. Execution Time	**84**	**377.8**	**1655.8**	**93.6**	**399.8**	**3734.2**

Table 1. Execution time evaluation of VC# and JAVA

7. GRAPHICAL ANALYSIS OF PROCESS EXECUTION

Authors have performed the experimental study for the two selected languages. A simple program written in both the languages has been designed and is tested in the system. The execution time of different lines of code of the program code in both the languages is calculated and shown above in table1. The graph is also represented in figures 6 and 7 which show the degradation of execution time of one of language as lines of code increases as compared with that of other. As a result of this analysis, it can be predicted easily that after comparison of execution behavior of programs designed with different programming platform and with different hardware configuration, one can select such a scenario of nodes which will result in best performance according to users requirements based on hardware configuration and the software program behavior. From this case study and the proposed approach it is clear that the execution time also depends on the language efficiency as well as on hardware efficiency.

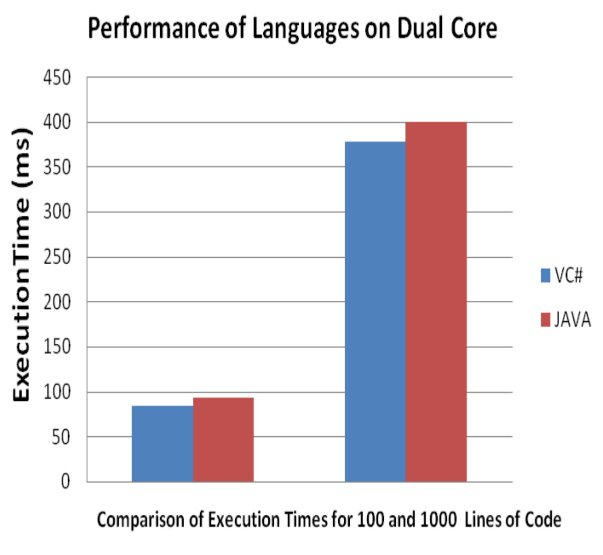

Fig6: Comparative performance of VC# and Java (10^2 and 10^3) lines of code.

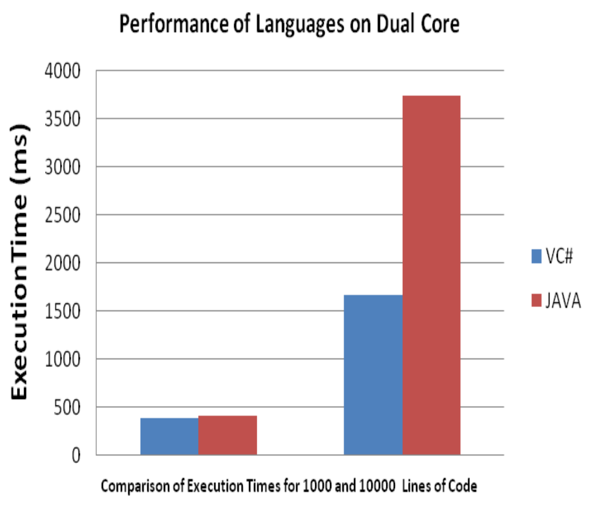

Fig7: Comparative performance of VC# and Java (10^3 and 10^4) lines of code.

The above proposed approach results in best performance of process execution based on user's requirements. This is only possible if one can classify the program behavior based on different aspects of users need when the program is generated and before it is really executed. One such factor which is responsible for the process execution time is its CPI calculation which if calculated at the time of program compilation could easily detect the program behavior. The proposed approach is based on such concepts that if have prediction of process execution time before its execution can be done then we can easily proceed the process to get executed on node which are more suitable for them taking into account of future throughput of the whole process execution.

8. CONCLUDING REMARKS

The proposed approach of the generic database system is very useful approach which results in a very commonly operated distributed database supportive approach for process execution. Generally the approaches that have been already proposed are specific based on some specific criteria but the present approach provides flexibility to the users that they just have to set this requirement and according to their need they could get best optimum process execution scenario to obtain the best optimum solution of their domain. This approach is also very efficient because although the network configuration goes on changing day by day and will keep on going changing but by this approach process execution can be efficiently done by providing generic features of this approach which will itself be a very stable concept of process execution based on the users need.

9. REFERENCES

[1] Hwang K. Advance Computer Architecture, 4th ed, Tata McGraw Hill, Reprint 2004, pp. 80-88.

[2] Takabatake T., Kaneko K. and Ito H., HCC Generalized Hierarchical Completely- Connected Networks, IEICE TRANS. INF. & SYST., Vol. E83-D, NO.6 June 2000.

[3] Liu M.L., Distributed Computing Principles and Applications, Pearson Education, 2003, pp. 25-26.

[4] Tanenbaum A.S., Distributed Operating Systems, Prentice Hall, 1995.

[5] Siberschatz A. and Galvin P. B., Operating Systems Concepts, 5th ed, John Wiley & Sons, Inc, 2000.

[6] Milenkovic M., Operating Systems Concepts and Design, Tata Mcgraw-Hill, 1997, pp. 46-58.

[7] Alakeel, A.M., A Guide to Dynamic Load Balancing in Distributed Computer Systems, IJCSNS International Journal of Computer Science and Network Security, Vol.10 No.6, pages 153-160, June 2010.

[8] Madiraju P. and Sunderraman R., A Mobile Agent Approach for Global Database Constraint Checking, ACM Symposium on Applied Computing, 2004.

[9] Singh, Y.J., Singh Y.S., Gaikwad A. and Mehrotra, S.C. , Dynamic management of transactions in distributed real-time processing system, International journal of database management systems (IJDMS), vol.2, No.2,May 2010,

[10] Saxena, V. and Shrivastava,M., UML Design for Performance Evaluation of Object Oriented Programs on Dual Core Processors, International Journal of Computer Theory and Engineering, Vol. 1, No. 4, 1793-8201. October2009

[11] Saxena, V and Arora, D,. "Performance Evaluation for Object Oriented Software Systems" , SIGSOFT Software Engineering Notes, March 2009, Volume 34, Number 2.

[12] Saxena, V. and Shrivastava, M., "Performance Evaluation of Non-Linear Pipeline through UML", International Journal of Computer and Electrical Engineering,Vol.2, No.5, pp.860-866, October, 2010.

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

[14] Pllana S. and Fahringer T., UML-based Modeling of Performance-oriented Parallel and Distributed Applications. Proceedings of the Winter Simulation Conference, Vol. 1, Issue. 8–11, Dec. 2002, pp 497–505.

[15] OMG, 2001, Unified Modeling Language Specification. Available: http://www.omg.org. (Accessed on 15th Jan. 2012).

[16] OMG, 2002, OMG XML Metadata Interchange (XMI) Specification. Available: http://www.omg.org. (Accessed on 15th Jan 2012).

[17] Booch G., Rumbaugh J. and Jacobson I., The Unified Modeling Language User Guide, Addison Wesley, Reading, MA, 1999.

[18] Booch G., Rumbaugh J. and Jacobson I., The Unified Modeling Language User Guide, Twelfth Indian Reprint, Pearson, 2004.