SYSTEM OPERATION CONTRACT FOR CONCURRENCY CONTROL IN A DISTRIBUTED DATABASE MANAGEMENT SYSTEM USING DATABASE MANAGEMENT SYSTEM MODEL

C.C OGBONNA[1] , P.O ODO[2], S.I EZICHI[3], S.A UGWU[4]
* DEPARTMENT OF ELECTRONIC ENGINEERING, UNIVERSITY OF NIGERIA NSUKKA

E- Addresses :1ogbonnachidube32@gmail.com, 2 polycarp.odo@unn.edu.ng
3 samuel.ezichi@outlook.com 4 tophersammy@yahoo.com

ABSTRACT: The first task of a database designer is to produce a conceptual model that reflects the structure of the information to be held in the database. A common approach to this is to develop an entity-relational model, often with the aid of drawing tools. Another popular approach is the unified modeling language. This work presents the deriving system operation contract for concurrency control in a distributed database management system from database management system model. The aim of this paper is to provide a roadmap on how a complex and software system can be analyzed and designed from the system model to system operation contract which is the gap between the requirement and the design in the software development life cycle.

Keywords: Database, Database management system, Distributed database management system model, concurrency control, operation contract

INTRODUCTION

The operation contracts are requirement analysis artifacts. They aim to bridge the gap between the requirement and the design in the development life cycle [1].Precisely, they show a relation between the actors' input and the system reaction in the level of the domain objects. The operation contracts' focus is to point out the required modifications in the state of the domain objects without getting into details of how things happened. Operation contracts help to identify the responsibilities of domain objects based on system operations, which are deduced from the use-cases text. System operation contract has been used in several approaches: mainly as an input to automate the production of design artifacts such as sequence diagrams [2, 3, 4, and 5]. Operation contracts attract numerous researchers because they tackle the possibility of designing object interactions based on systematic approaches. Larman incorporates some UML diagrams in addition to the operation contracts in a lightweight-agile version of UP to leverage the understanding of the software system requirement aspects of the system operations in a declarative way, and they avoid implementation-specific details. The main issue of declarative specifications is the under specification, which is the possibility of obtaining several ways of implementation [6]. For this matter and in the design phase, the operation contracts can serve to construct interaction diagrams by the help of GRASP

patterns. The GRASP patterns include nine design patterns: Creator, Controller, Pure Fabrication, Information Expert, High Cohesion, Indirection, Low Coupling, Polymorphism, and Protected Variations. The GRASP patterns require three software requirement artifacts: domain model, use-cases, description and operation contract.

Korth [7] in his work explained database as a collection of interrelated data. Typically users access a database through application programs that run on top of the database management. He mentioned that examples of such applications are airline reservation and automatic teller system.

Subharthi [8] defined database management system as a collection of software programs that allow multiple users to access, create, update and retrieve data from and to the database. He added that the database functionality is optimal storage and retrieval of data, maintain correctness of the data, and maintaining consistency of the system at all-time

Distributed computing systems have increasingly become, for the past two decades or so, an active area of research and development. This however, was an inevitable consequence as the cost of processing elements has continually fallen which was a contributing factor in attracting and encouraging many of such systems to be built, cost effectively, and be experimented with [9].

Tamew[10] in his work defined a distributed database as a collection of multiple, logically interrelated database distributed over a computer Network. He further demonstrated that, a distributed database [DDBS] is not a "collection of files" that can be individually stored at each node of computer network. To form a DDBS, files should not only be logically related, but there should be structures among the files and access should be via a common interface. He noted that physical distribution does not necessarily imply that the computer system be geographically far apart; they can be actually be in the same room. It implies that communication between them is not done over network instead of through shared memory or shared disk. He suggested that multiprocessor system should not be considered as a DDBS because the mode of operation is through shared memory.

A wonderful contribution made by[11] described concurrency control as the activity of coordinating concurrent accesses to a data- base in a multiuser database management system(DBMS).Concurrency control permits users to access a database in a multi- programmed fashion while preserving the illusion that each user is executing alone on a dedicated system. The main technical difficulty in attaining this goal is to prevent database updates performed by one user from interfering with database retrievals and updates performed by another. The concurrency control problem is exacerbated in a distributed database management system (DDBMS) because (1) users may access data stored in many different computers in a distributed system, and (2) a concurrency control mechanism at one computer cannot instantaneously know about interactions at other computers.

THE GOAL OF THE RESEARCH WORK

The major goal of a system operation contract for concurrency control in a distributed database management system is for a detailed description of system state changes when an operation happens and to bridge the gap between the requirement and the design in the development life cycle The diagram in figure 1 summarizes the goals of this research work and serves as a frame work to be used to achieve this goal.

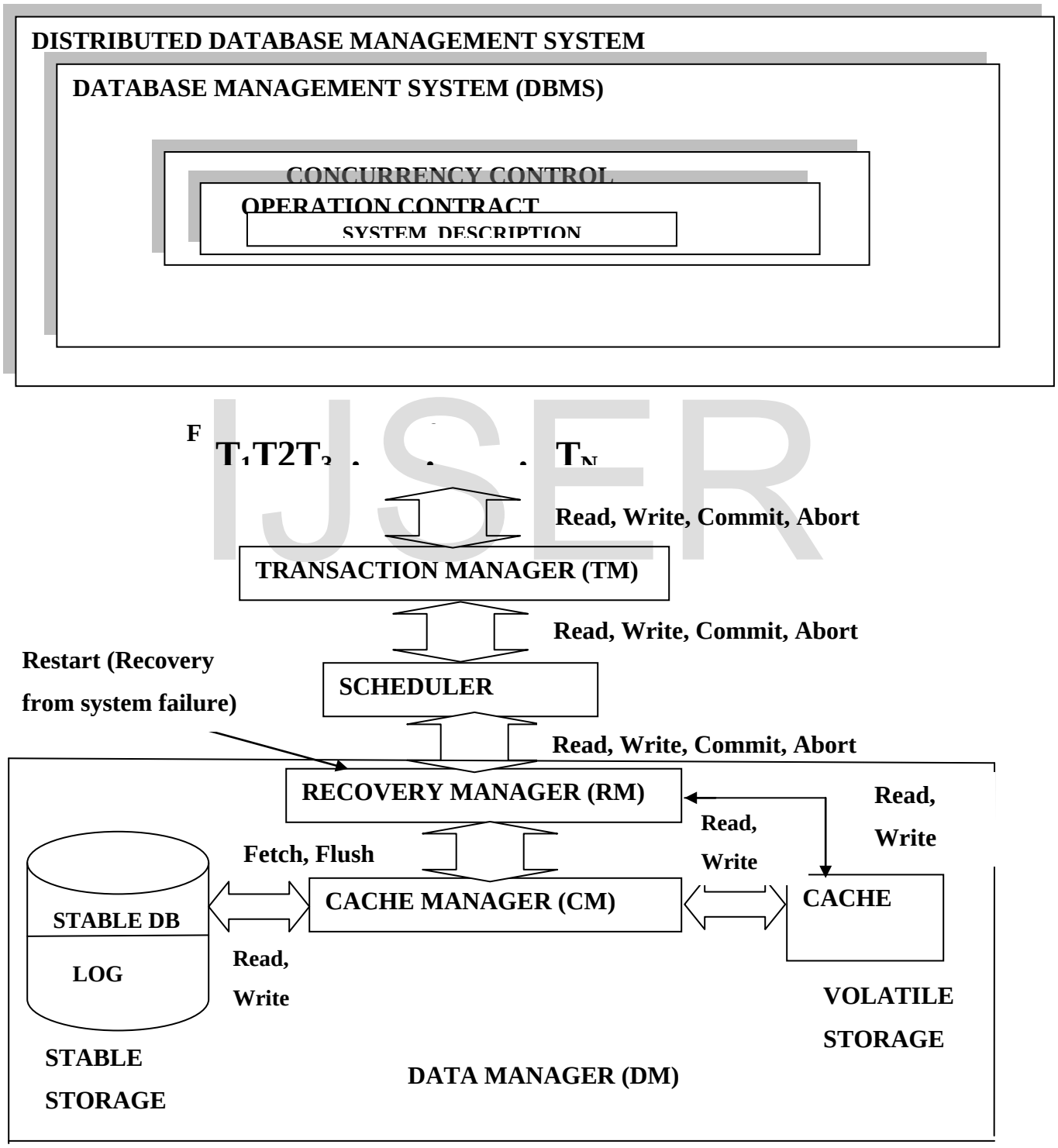

Figure 2 :Model of a Database management system [12]

2.1 MODEL OF A DATABASE MANAGEMENT SYSTEM

Based on the diagram shown in figure 2, the database management system works as follows: Transactions submit their operations to the transaction manager which passes them on to the scheduler. The scheduler receives Read, Write, Commit and Abort operations from the transaction manager. The scheduler can also send abort to the data manager immediately. For Read, write or commit operations, the scheduler must decide, possibly after some delay whether to reject or abort the operation. If it rejects the operation, the scheduler sends negative acknowledgement to the transaction manager, which sends an Abort back to the scheduler, which in turn promptly passes the Abort to the data manager. If the scheduler accepts the operation, it sends to the data manager, which processes it by manipulating storage. When the data manager has finished processing the operation, it acknowledges to the scheduler, which passes the acknowledgement to the transaction manager. For a read, the acknowledgment includes the value read. In addition to Read, Write, Commit and Abort. The data manager may also receive a Restart operation. This is sent by an external module, such as the operating system, upon recovery from system failure. The task of Restart is to bring the database to a consistent state, removing effects of and applying missing effects of committed ones

2.2 AN IMPROVED ARCHITECTURE FOR CONCURRENCY CONTROL IN A DISTRIBUTED DATABASE

The diagram in Figure 3 is an improved architecture for a distributed database. It is a star-wired topology with Intelligent Ethernet managed switched Network that is resided in the Coordinator. The architecture consists of N nodes Participants and Coordinator (Intelligent Ethernet managed switched network (IEMNS)) . Each node has a unique number from 1, 2, 3, 4, N. In star configuration, data passes through the coordinator to reach other participants on the Network. The intelligent Ethernet Switches has the ability to manage multiple traffic. Instead of flooding that traffic to all users, they use Internet Group management protocol (IGMP) to direct the traffic only to the desired recipients [14].This reduces the chances of data collisions that occur during data transaction [13]

3 DERIVATION OF SYSTEM OPERATION CONTRACT

Based on the Figure 2 and 3 the distributed database management system as shown has a “home site” the site where it originated and which process information, also a communication links, which transmit information from site to site. Transaction submits its operation to the transaction manager at its home site, and the transaction manager subsequently forwards the operation to the appropriate sites. A Read(x) or Write(x) operation is forwarded to the site where x is stored and is processed by the scheduler and data manager of that site as if it were operation submitted by a local transaction. The result of the operation is then returned to the transaction manager of Transaction’s home site. A commit operation concerns all sites involved in the processing of transaction. Consequently, the transaction manager of

transaction's home site should pass the commit operation of transaction to all sites where Transaction accesse data items. The same is true for Abort. Thus, the processing of a logically simple operation (commit or Abort) must take place in multiple places in Data base system

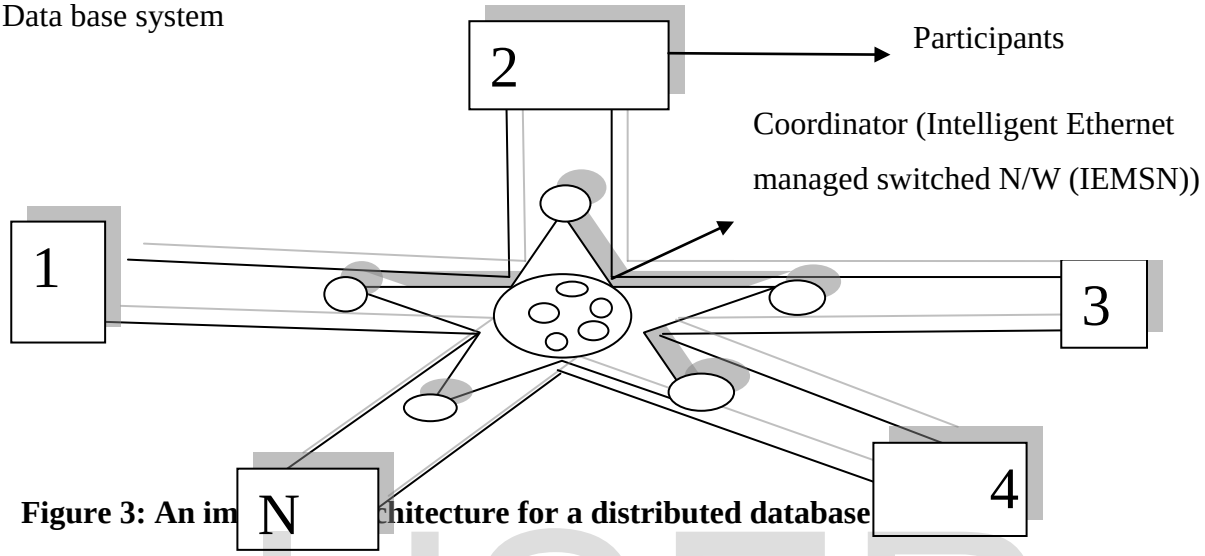

Figure 3: An im chitecture for a distributed database

3.1 Use Case Analyses for Concurrency Control in Distributed Database

The use case for concurrency control in distributed database is shown in Figure 4. A use case is a series of related interactions between a user (or more generally an "action") and a system that enables their user to achieve a goal [15]

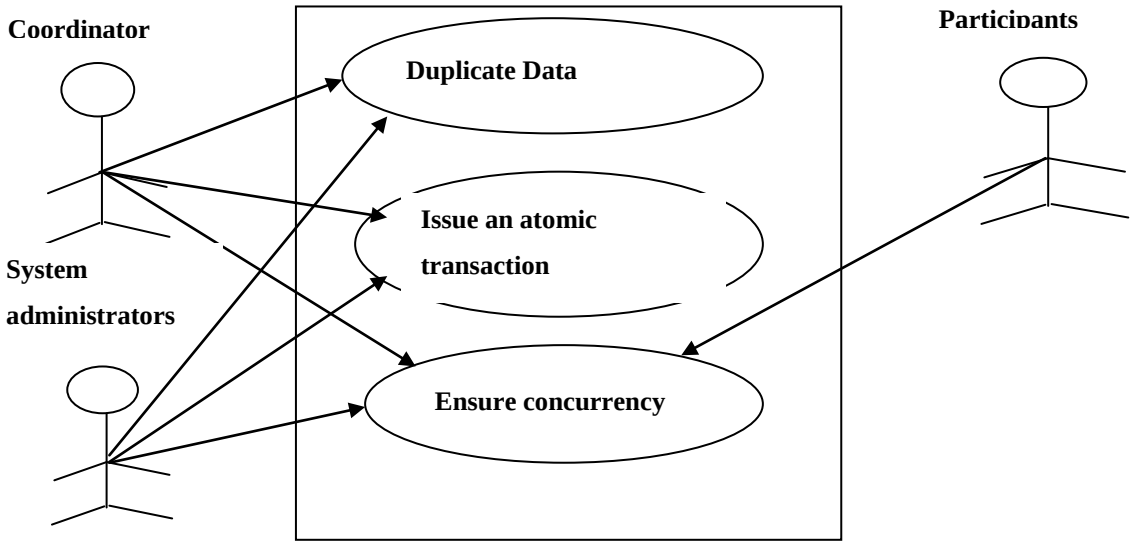

Fig.4. USE CASE FOR CONCURRENCY CONTROL

3.2 Sequence Diagram for Concurrency Control

The sequence diagram is used primarily to show the interactions between objects in the sequential order. It is used in the transition from requirements expressed as use cases to the next and more formal level of refinement. It can be used to document how object in an existing system currently interact [17]. The diagram in figure 5 represents a sequence diagram for concurrency control that is being derived from the diagram in figure 4.

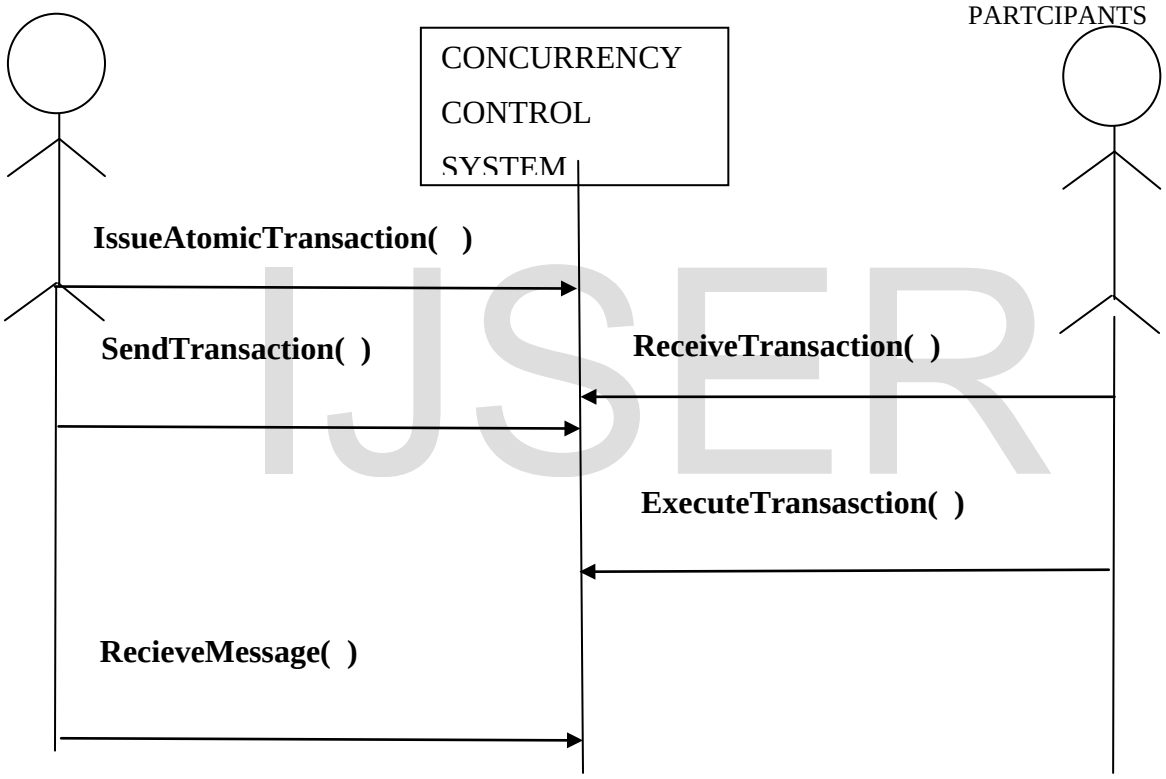

FIGURE5. SEQUENCE DIAGRAM FOR CONCURRENCY CONTROL

3.3 CONTRACT FOR THE CONCURRENCY CONTROL IN A DISTRIBUTED DATABASE

The operation contracts are requirement analysis artifacts. They aim to bridge the gap between the requirement and the design in the development life cycle [1]. Precisely, they show a relation between the actors' input and the system reaction in the level of the domain objects. The contract used in this study is derived directly from the Figure 5 i.e. the sequence diagram for concurrency control and indirectly from Figure 2 i.e.model of a database management system.

***NAME: IssueAtomicTransaction( )**

RESONBILITY: 1) Checks the participants status
2) Writes a mark in the Log if a participants is not available

TYPE: Distributed database management System

CROSS REFERRENCE:

NOTES:

EXCEPTION: 1) Unavailable participants
2) Unsaved transaction

OUTPUT: Receive a message about the availability of cooperative participants.

PRECONDITIONS: 1) there must be an internet service
2) The transaction must be atomic

POSTCONDITIONS: 1) Atomic transaction was issued
2) Distributed database is associated with a transaction manager

***NAME: SendTransaction ( )**

RESONBILITY: 1) Send an atomic transaction.

TYPE: Distributed database management System
CROSS REFERRENCE:

NOTES:

EXCEPTION: 1) inconsistent transaction

OUTPUT: Receive a message about the committed operation or aborted transaction.

PRECONDITIONS: 1) Participants must be available
2) Transaction must be consistent

POSTCONDITIONS: 1) Transaction was sent
2) Distributed database was associated with concurrent result

***NAME: ReceiveTransaction ( )**

RESONBILITY: 1) Receive an atomic transaction
2) Write in Log

TYPE: Distributed database management System

CROSS REFERRENCE:

NOTES:

EXCEPTION: 1) Inadequate data storage

OUTPUT: display message for a received message transaction.

PRECONDITIONS: 1) the capacity of database must be sufficient

POSTCONDITIONS: 1) Data was received.
2) Distributed database is associated with a data manager.

***NAME: ExecuteTransactionTransaction( )**

RESONBILITY: 1) Execute received transactions
2) Write committed or aborted in log
3) Sends Ready to the coordinator
TYPE: Distributed database management System

CROSS REFERRENCE:

NOTES:

EXCEPTION: 1) Uncommitted transaction

OUTPUT: Ready message is received by the coordinator

PRECONDITIONS: 1) Transaction must be committed

POSTCONDITIONS: 1) Transaction is executed
2) Transaction is committed
3) Distributed database is associated with a scheduler
***NAME: ReceivedMessage( )**

RESONBILITY: 1) Received a Ready message
2) Completes the transaction phase
TYPE: Distributed database management System

CROSS REFERRENCE:

NOTES:

EXCEPTION: 1) Unavailable message

OUTPUT: Received Ready message is got by the coordinator.

PRECONDITIONS: 1) There must be committed transaction

POSTCONDITIONS: 1) Received Ready was achieved

CONCLUSION

A computer can do everything if the instruction is under o's and 1's and depending on the type of program you have loaded; also how its data is structured and maintained. The detailed description of the effect of the execution of the system operation involved is one of the most important tasks in the analysis and design stages of a software system. System operation contract has been used in several approaches: mainly as an input to automate the production of design artifacts such as sequence diagrams [2, 3, 4, and 5]. Operation contracts attract numerous researchers because they tackle the possibility

of designing object interactions based on systematic approaches. We believe our method is useful when software developers need to analyze and design a system from the system model which is considered as the most appropriate way of understanding how a system operates and is maintained

REFERENCES

[1] Abdelzad, V.,A visual Notation and an improvement for the synyax of Lerner's operation contract, Ottawa, Canada, 2016,.

[2]Bousetta, B., Omar, E., &Gadi, T. (Automating software development process: Analysis-PIMs to Design-PIM model transformation. International Journal of Software Engineering and Its Application, 7, 167–196. 2013.

[3]Shakya, B &Nantajeewarawat, E.. A design pattern knowledge base and its application to sequence diagram design. 2013 International Computer Science and Engineering Conference ICSEC (pp. 179-184). NakornPathom, Bangkok, Thailanddoi: 10.1109/ICSEC.2013.6694775. 2013.

[4]Laosen, N., &Nantajeewarawat, E. A knowledge-based approach for generating UML sequence diagrams from operation contracts (pp. 388–399). Presented at the Tenth International Conference on Knowledge, Information and Creativity Support Systems, Phuket, Thailand.2015.

[5]Lohmann, M., Sauer, S., & Engels, G. Executable visual contracts. In 2005 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'05) (pp. 63–70). IEEE.2005.

[6]Larman, C. Applying UML and patterns: an introduction to object-oriented analysis and design and iterative development (3rd ed.). Upper Saddle River, N.J: Prentice Hall PTR.2005.

[7]KorthH.F and Siberschatz.A. , Database system concept, McGraw-Hill, New York, 1986.

[8] Subharthi .P, System performance Evaluation Techniques.[online].Available: http://www.cse.wusti.edu/ujan/cse567-08/index.html. 2008
.

[9] ZedanH.S.M, Distributed computer system: theory and practice. Butterworth and co, 1980.

[10] Tamer O.M and Patrick V. Principles of distributed database system: Springer New York Dordrecht Heidelberg London. 2010.

[11] Philip A.b and Nathan G., Concurrency control in distributed database systems: Computing surveys, vol 13, No 2, 1981

[12] Bernstein, P.A; Hadzilacos, V, and Godman, N. Concurrency control and Recovery in Database System. Addison Wesley, 1987

[13]OGBONNA C.C,and P.O.ODO; Improvement on concurrency control in a distributed database; International journal of science and Engineering Research, volume8, issue3,ISSN 2229-5518 March 2017

[14]Cisco system .Inc. "Ethernet, Hubs, switches, and the Evolving Factory Network",www.cisco.com/go/offices, pp.1 – 2. 1992-2002

[15] Michael .S ., Use cases definition (Requirement management bases), 2009

[16]Sidharth T. Sample of an event chain diagram ,2010.

[17] Donmald .B. The sequence diagram, 2004.