ISSN 2278-7763

A REVIEW ON OBJECT MODELLING METHODOLOGIES

ROHIT SHARMA[1], Dr. VINODINI KATIYAR[2]

[1]Research Scholar, C.M.J. University, Shillong, India. [2]Professor and Head, S.R.M.C.E.M. Lucknow, India.
Email – rohitsharma2412@gmail.com

ABSTRACT
Object plays an important role in database for obtaining the data and is at the core of database architecture. This paper is concerned with exploring the techniques of object modelling which is necessary to understand the dynamics of object oriented database and consequently the evolutionary technique of multi-view database. Thus this paper is concerned with exploring various object modelling methodologies.

KEYWORDS- Object, Object Modelling , Multi -view database.

1. INTRODUCTION

Models contain progressively more detail. Subsequent models are made to contain more detail than their predecessor models by focusing on a smaller portion of the system and extending the analysis to produce additional depth in the portion. The process of model building, therefore, parallels the system developers' understanding of the system [1]. As the system analysts begin their study of the target system, their understanding is very general. For any software development project, the initial conceptualization represents a starting point, and further analysis may eliminate one or more of these objects or add other objects. In addition, this initial object configuration is probably similar to the configuration with which most people would start because the elements making up the conceptualization are familiar and common. During the model-building process, the models become increasingly detailed, and the system developers' understanding also increases. Modelling software systems also creates concise and unambiguous representations, which facilitate communication between collaborating system developers. Before significant time is invested in programming a poorly conceptualized system these precise representations can also be evaluated, verified, and corrected.

2. OBJECT MODELING

2.1 OBJECT MODELLING TECHNIQUE (OMT)

The Object Modelling Technique (OMT) software engineering methodology is another well known example of a software engineering methodology. Understanding OMT is the requirement of present study as it deals with object-oriented development in the analysis and design phases.

The analysis phase starts with a problem statement which includes a list of goals and a definitive enumeration of key concepts within a domain. The following problem statement is expanded into three models: an object model, a dynamic model, and a functional model. The object model represents the artifacts of the system. The dynamic model shows the interaction between these artifacts represented as events, states, and transitions. Also the functional model shows the methods of the system from the perspective of data flow. The object-model diagram is started by analysis phase state diagrams, event-flow diagrams, and data-flow diagrams. The analysis phase is then complete.

Both the system design phase and the analysis phase follow each other. However the overall architecture is established. Firstly the system is designed into subsystems which are then allocated to processes and tasks, taking into account concurrency and collaboration [2]. Along with a strategy the persistent data storage is established to manage shared global information. Then, boundary situations are examined to help guide trade off priorities. There exists implementation of plan. Object classes are established along with their algorithms with special attention to the optimization of the path to persistent data. Examination of issues of inheritance, aggregation, and default values.

The engineering methodology of OMT software is sequential in the sense that first comes analysis, followed by design. Every phase,

IJOART

Copyright © 2013 SciResPub.

ISSN 2278-7763

a cyclical approach is taken among the smaller steps. Also it is very much like the Booch methodology where emphasis is placed on the analysis and design phases for initial product delivery [3]. In both cases either OMT or Booch they do not emphasizes implementation process, testing, or other life cycle stages.

2.2 RATIONAL OBJECTORY METHODOLOGY

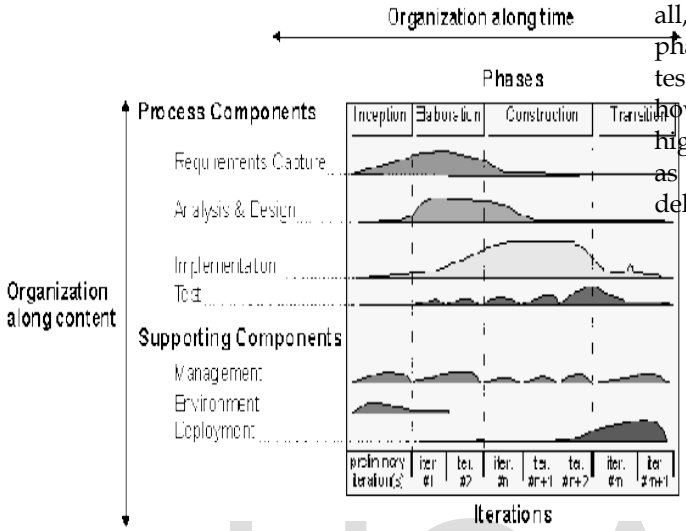

Figure 1: The Rational Objectory Methodology

A full life cycle software engineering methodology is known as Rational Objectory Methodology. It is an iterative process governed by requirements management. Rational Objectory activities create arid maintains models to aid the developer in supporting the methodology.

The Rational objectory software engineering methodology [4] can be explained in two dimensions: components of time and process. The dimension of time: represents the dynamic aspect of the process and is expressed in terms of cycles, phases, iterations and milestones. The component of process dimension is described in terms of process components, activities, artifacts, and workers.

2.3 ITERATIONS

Each phase in the Rational Objectory software engineering methodology can be further broken down in various iterations. It is a complete development loop resulting in a internal or external system release. Each iteration goes through all aspects of software development: analysis arid design, requirement capture, implementation and testing.

- The Requirements Capture
- Analysis and Design
- Implementation
- Testing
- Comparison to Water Sluice

2.4 SOFTWARE ENGINEERING PHASES

Before we descend into the definition of software engineering methodologies, we need to define the meanings of some of the fundamental phases. There are four fundamental phases in most, if not all, software engineering methodologies. These phases are analysis, design, implementation, and testing. These phases address what is to be built, how it will be built, building it, and making it high quality. These phases will now be defined as they apply to the life cycle stage of product delivery emphasized in this paper.

a. The Analysis Phase
   - Things
   - Actions
   - States
   - Typical Scenarios
   - Incomplete and Non-Monotonic Requirements

b. The Design Phase

2.5 ARCHITECTURE

The architecture defines all components terraces, and behaviours of the system. The building blocks for the system are the components. They built from scratch or re-used from an existing component library. They refine, capture the meaning of details from the requirement document. The components are composed with other components using their interfaces. An interface forms a common boundary of two components. The interfaces architectural surface where independent components meet and communicate with each other. All over the interface they affect each other. The interface defines behaviour where one component responds to the stimuli of another component's actions.

2.6 IMPLEMENTATION PLAN

The implementation plan establishes the schedule and needed resources. It explains implementation details including programming languages, environments of programming and many more.

The implementation plan is considered as of the design, which the position is taken here, also it could consider as the first accomplishment in the implementation phase [5]. One of the goals of the design phase is to establish a plan to complete the system. Thus it is very natural to include the implementation plan. Also, the trade-offs

IJOART

Copyright © 2013 SciResPub.

ISSN 2278-7763

between alternative architectures can be influenced by differences in their implementation plans.

2.7 CRITICAL PRIORITY ANALYSIS

The critical priority analysis generates a list of critical tasks. It is necessary to successfully accomplish a critical task. The project may succeed or fail based on the outcome of these tasks. Various projects may have more than one critical task.

There are two major categories of critical tasks. A category of tasks are associated with the building of the system. They are the critical tasks that the teams must accomplish well. The other category of critical tasks is associated with the system itself.

2.8 PERFORMANCE ANALYSIS

Once given the typical scenarios from the document required, the system can be designed to meet performance objectives. Different system architectures will yield different predicted performance characteristics for each typical scenario. Depending on the usage frequency of the scenarios in the system, each architecture [6] will have benefits and drawbacks with advantages and disadvantages. The trade-offs are then weighted to establish the system architecture. Frequently a system is designed to give fast response to an action initiated by a human customer at the expense of having to do more complex systems work such as including indexes, cache management, and predictive pre-calculations.

- Test Plan
- The Implementation- In the implementation phase, the team builds the components either from scratch or by composition.
- Critical Error Removal- There are three kinds of errors in a system, namely critical errors, non-critical errors, and unknown errors. A critical error prevents the system from fully satisfying the usage scenarios. These errors have to be corrected before the system can be given to a customer or even before future development can progress.
- Regression Test- Quality is usually appraised by a collection of regression tests forming a suite of programs that test one or more features of the system. A regression test is written and the results are generated. If the results are in error, then the offending bug is corrected. A valid regression test generates verified results. These verified results are called the gold standard. This term is borrowed from financial markets where paper money issued by governments was backed by real gold.
- Internal Testing- Internal testing deals with low-level implementation. Here each function or component is tested. This testing is accomplished by the implementation teams. This focus is also called clear-box testing, or sometimes white-box testing, because all details are visible to the test. Internal limits are tested here.
- Application Testing- Application testing deals with tests for the entire application. This is driven by the scenarios from the analysis team. Application limits and features are tested here. The application must successfully execute all scenarios before it is ready for general customer availability. Te scenarios are a part of the requirement document and measure success. Application testing represents the bulk of the testing done by industry. Unlike the internal and unit testing, which are programmed, these test are however the script in the system has a collection of parameters and collect results. Before scripts may have been written by hand but in many modern systems this process can be automated. Most current applications have graphical user interfaces (GUI). Testing a GUI to assure quality becomes a bit of a problem. Mostly there is, if not all, GUI systems have event loops. The loop of GUI event contains signals for keyboard, window mouse, and other related events. Associated with each and every event are the coordinates on the screen of the event. The coordination of the screen can be related back to the GUI object and then the event can be serviced. Whereas if some GUI object is positioned at a different location on the screen. Basically the events at the new coordinates should be associated with the same GUI object. This association which is logical can be accomplished by giving unique names to all of the GUI objects and providing the unique names as additional information in the events in the event loop. The application of GUI reads the next event off of the event loop, GUI object location, and services the event.

IJOART

Copyright © 2013 SciResPub.

ISSN 2278-7763

- Stress Testing- Stress testing deals with the quality of the application in the environment. The idea is to create a more demanding environment of the application than the application would experience under normal workloads. This is the hardest and most complex type of testing to accomplish and it requires a joint effort from all teams. A test environment is established with many testing stations. At each station, the system is exercised by the script. These scripts are usually based on the regression suite. Many stations are added, all one by one hammering on the system, till the system breaks. The system is repaired and the stress test is repeated until a level of stress is reached that is higher than expected to be present at a customer site. Race conditions and memory leaks are often found under, testing of stress. A race condition is a conflict between at least two tests. Each and every test works correctly when done in isolation. When the both tests are run in parallel, both of tests fail. This is basically due to an incorrectly managed lock.

2.9 EVOLUTIONARY TECHNIQUES OF OBJECT MODELLING

This section is dedicated to explain the evolutionary or advanced approaches of the techniques of data modelling[7]. In principle, object oriented database system need to include the features of object oriented programming and relational database systems.

In figure 2, green box represents the features of object oriented programming, blue box shows the significant features of relational database while violet color reflects unique feature of multi-view database i.e. accession by different users as per their requirements. However, the data can be shared by different users. The figure also shows the evolutionary makeup of an object oriented database with multi-view paradigm. The concept is superior to traditional approach of object oriented database. This new concept is extremely vital in the present era of globalization.

Following figure reinforce the point in better way:

IJOART

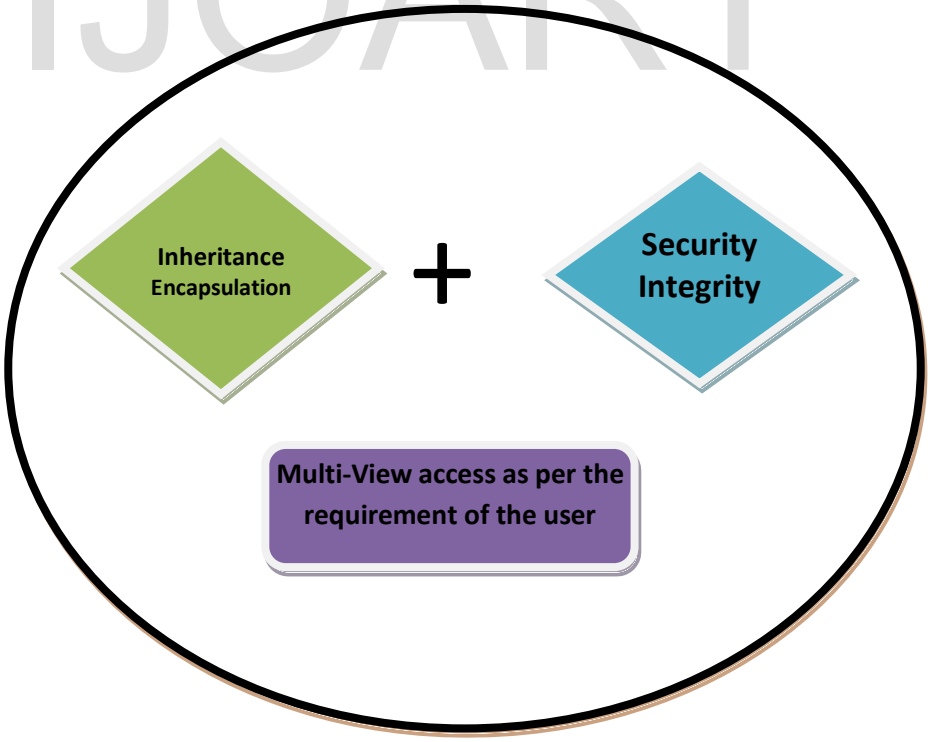

Figure 2: Showing the evolutionary makeup of an Object Oriented Database with Multi-view paradigm

Copyright © 2013 SciResPub.

ISSN 2278-7763

3. OTHER METHODOLOGIES

Apart from methodologies described in the previous section, three major categories of methodologies are presented:

- Sequential
- Cyclical
- Water Sluice

The sequential and cyclical methodologies, informally called as the waterfall and spiral methodologies, are generic in design and have been simplified to emphasize a key aspect.

- In a sequential methodology, the four phase of analysis, design, implementation, and testing follow each other sequentially.

- In a cyclical methodology, the four phase of analysis, design, implementation, and testing are cycled with each cycle generating an incremental contribution to the final system.

- The Water Sluice is a hybrid borrowing the steady progress of the sequential methodology along with the iterative increments of the cyclical methodology and adds priority and governors to control change.

4. CONCLUSION

Here in this paper various data modelling approaches have been explored. Approaches which are necessary to understand the dynamics of object oriented database and consequently the evolutionary technique of multi-view database is viewed. In our review it shows that the full life cycle software engineering methodology is known as Rational Objectory Methodology. It is an iterative process governed by requirements management. Apart from various methodologies described in section 2 section 3 shows three major categories of methodologies known as Sequential, Cyclical and Water Sluice. Thus the whole paper shows a clear view of object, its role in data base and various object modelling methodologies.

ACKNOWLEDGMENT

I express my deep gratitude to my supervisor Dr. Vinodani Katiyar without her able and valuable guidance, persistent encouragement and critical comments the study would have not been completed.

REFERENCES

[1]. HobermanSteve (2009) "Data Modeling Made Simple", 2nd Edition, Technics Publications, LLC.
[2]. Codd, E.F., "A Relational Model of Data for Large Shared Data Banks", IBM Research Laboratory, Vol. 13, No.6, June 1970.
[3]. P.-J. Charrel, D. Galarreta, C. Hanachi, B. Rothenburger, "Multiple Viewpoints for the Development of Complex Software", in Proceedings of the IEEE Systems, Man and Cybernetics Conference, 17-20 October , 1993, Le Touquet, France, p. 556-561.
[4]. M. Gergatsoulis, Y. Stavrakas, D. karteris, A. Mouzaki ,D. Sterpis, A Web –Based System for Handling Multidimentional Infromation through MXML. Lecture Notes in Computer Science (LNCS 2151), (2001), PP.352-365, Springer –Verlag.
[5]. S. Abiteboul, A. Bonner, Objets and views, Proceedings of the Int'l Conference on Management of Data,ACM SIGMOD, (1991), pp. 238- 247. Denver, Colorado.
[6]. Bertino, E. (1992) A View Mechanism for Object-Oriented Databases. Proc. The 3rd International conference on EDTB'92 (pp. 136-151). Vienna, Australia.
[7]. Sheth, Amit.P. and Larson A. James (1990), Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases, ACM Computer Surveys, Vol. 22, No. 3, September 1990.

IJOART

Copyright © 2013 SciResPub.