www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 2 Issue 8 August, 2013 Page No. 2403-2408

Empirical Study on Migrating Data From Relational Databases To Object-Oriented Technology.

Dr Leelavathi Rajamanicakam
drleelavathir@gmail.com

Abstract: *Tremendous changes have been taking place in information technology for a few decades. Due to the rapid evolution in this area, the demand for innovation in this area is much higher than elsewhere. This requires large effort of companies to respond quickly to market conditions in order to organize work and conduct business more efficiently. In particular, companies have to reengineer existing form of new key technologies like WWW or E-Commerce. A typical scenario in many companies when applying a reengineer process is That on the one hand, a large body of data is captured in relational or even hierarchal or network databases, and on the other hand object oriented applications have to be developed. Thereby, a new object model is constructed which represents the current state of the company's business processes. However, the new object model and the existing relational database's model usually do not go well together. In other words, a large semantic gap between both models must be bridged. The approach proposed in this paper is database migration. Basically, this approach comprises two tasks. In the first task, the relational database schema is reengineered. The schema is transformed into a well designed and intuitively understandable object oriented schema, which the new applications can adapt. Afterwards, the data are (automatically) migrated into an object oriented DBMS.*

An algebra is proposed for a formal definition of data migration process. The schema transformation process is subdivided into three sequential phases. In the first phase, the relational schema is transformed (automatically) into an SOT schema. This initial SOT schema is then redesigned resulting in the adequate object oriented schema. Finally, in the third phase t resulting SOT schema is (again automatically) transformed into an object oriented schema according to the ODMG standard. The data migration process is generated automatically for each schema transformation phase. In order to implement schema transformation, the concept of transformation rule is proposed. The transformation rules define elementary restructuring operations within the SOT model. A basic set of transformation rules has been proposed which can be extended. Finally, a prototype has been implemented as a proof of concept.

Keywords: Ecommerce, relational schema, life cycle, object technology.

1. Introduction

In general process covers all areas of information systems. In this thesis, in particular, three areas are of major interest.

Relational Databases Systems: Relational database system represents the current standard in technology for implementing database applications. The concept was proposed in the early seventies, and now commercial DBMS (database management systems) like DB2, oracle, Informix Sybase or dominate the market of data persistence. A large body of electronic data is to be read in relational database now a days. One big bonus of RDBMS is the maturity they have gained in extensive research efforts in the last decades. This allows RDMS products to be used for high performance and mission-critical database applications.

Object-Orientation: Now a day's object oriented paradigm prevails in modern software development. It has emerged as an important technology to ensure software quality, reusability, portability, maintainability and extensibility. Almost all the components of the new information systems are developed within an object-oriented software engineering life cycle. In particular, these components include database systems, user interfaces, operating systems and applications. The phase of object –oriented software engineering comprises, amongst others, analysis, design and implementation. More and more object-oriented programming languages like small talk, C++ and java are replacing procedural languages.

Reengineering: The rapid changes in information technology and society force companies to quickly respond to changing conditions on a global market. This raises the problem of reengineering information system take advantage of these technologies. In general, two cases motivate in green engineering can be distinguished. Firstly, changes in the internal organization of companies must be reflected in information systems. This is also referred to as business process reengineering (BPR). Secondly, due to the emerging key technologies like E-commerce, the WWW or data-ware housing, companies have to adapt, modify or even rewrite parts of their information systems.

A typical scenario in many companies when apply in reengineering process is that the one hand, a large body of data is captured in relational (or even hierarchical or network) databases, and on the other hand new object-oriented applications have to be developed. Thereby, a new object model is constructed which represents the current state of the Companies Business Processes. However, the new object model and the existing relational database's model usually do not go well together. In other words, a large semantic gap between both models must be bridged.

The evolutions of database technology on one side and software engineering on the other side have barely influenced

each other in the last decades. As a consequence, RDBMS and object-orientation incorporate principally different paradigms. Existing relational databases and object-oriented applications cannot be integrated in a seamless way, a problem which is known as impedance mismatch. More precisely, the transition of data stored in the relational database to object-oriented applications and vice-versa is non-trivial.

Meanwhile, object-oriented DBMS (OODBMS) have been proposed to support seamless integration of object technology and data persistence, and various commercial products are available. However, many organizations principally refrain from using OODBMS because existing products cannot compete with RDBMS with respect to maturity and reliability. Other organizations are willing to give OODBMS a try, but then require the existing relational data to be available in object-oriented database stems.

As top gap solutions, mainly hybrid approaches such as object-oriented views over related schemas have been proposed. There by, the data-

base remains untouched and a mapping between objects and relations is defined. However, these approaches do not resolve the data model mismatch. Moreover, the implementation of a mapping between both models is an expensive and error prone activity, and the required data conversions at runtime lead to performance degradations.

The approach proposed in this thesis is database migration. Basically, this approach comprises two tasks. In the first task, the relational database schema is reengineered. The schema is transformed into a well-designed and intuitively understandable object-oriented schema, which the new applications can adapt. Afterwards, the data are (automatically) migrated to OODBMS. There are several reasons why database is migration is worth further investigation. First, database migration promises better results than other approaches like object views for the following reasons. Since the data are converted to objects only once, database migration principally allows more flexibility with respect to the reengineering the relational schema into a suitable object –oriented schema. In addition, performances higher because data does not have to be converted at runtime. Secondly, existing approaches for database migration do not exploit the full potential of the object-oriented paradigm, so that the resulting object oriented schema still "looks rather relational" and retains existing drawbacks and weaknesses of the relational schema. Finally, efforts are taken to remedy the current immaturity of commercial OODBMS for some mission-critical database applications. Nevertheless, OODBMS have the advantage of being more efficient when modeling complex data structures. They are well suited for storing data of complex (non-standard applications) such as CAD systems and office automation systems. Recently, object-relational DBMS (ORDBMS) have started to offer some object –oriented features, and further features are likely to be addresses in the future. ORDBMS can be expected to extend current commercial RDBMS products. Hence, they offer the same reliability as RDBMS. ORDBMS have the same problem, as discussed before, for OODBMS, namely to convert an existing relational schema in to new exploiting object-oriented features, and to adapt the database. Thus, the results present in this thesis for OODBMS will also be important for the (future) object-relational database systems. Besides reengineering of the complete information system of a company, database migration may also be of interest for other tasks. For Example, it may happen that new applications do not have to operate on the original databases.

Instead from time-to –time the database pr part of it can be downloaded into an object-oriented database. For example, could represent a company's product catalogue on which web-based java applications then operate.

2. RELATIONAL DTABASE CONCEPTS.

Relational database systems represent the standard technique for implementing database applications. The main foundations of relational databases were laid in The early seventies, mainly in the relational data model[Cod70]. Later, the Entity-Relationship model[Che76] was proposed for simple yet powerful modeling of relational database schemas. This chapter aims to reflect the current state of the art in relational database application design. The focus is on three main aspects. Firstly, data models used for constructing relational database systems discussed. These are, on the concept level, the entity-relationship model and, on the logical level, the relational data model. Secondly, the object -relation ethical foundations are discussed. This is based on relational algebra which represents the main formalism for implementing RDBMS. Most available DBMS provide access via the declarative query language SQL (Structured Query language), which is based on relational algebra. Finally, the last part of this chapter deals with design strategies for constructing relational database applications. Database application design is not only the task of encoding relational schemas. Usually it is a sequential and iterative process which covers several design phases like database design and functional analysis.

3. DATA MODELS

The process of designing database schema is usually decomposed into several phases. In each phase, a different data model is used. Typically three levels of designs are distinguished, as shown in fig 2-1: conceptual design, logical design and physical design. The task conceptual design is to formalize the results obtained from requirements engineering by means of a certain method. The resulting conceptual schema has a higher level of abstraction than the subsequent logical schema and does not include any implementation details. Logical design consists of mapping the conceptual schema into a logical schema which can be processed by the DBMS. The logical schema is thus expressed by means of data definition language. Finally, the physical schema describes the internal storage structure of the database.

The Entity-Relationship model.

The Entity relationship(ER) model was introduced by Chen[Che76], and describes data as entities, relationships and attributes. An entity is a "thing" in the real world with an independent existence, For example, a department, a projector an employee. Each entity has attributes- the particular properties that describe it. For example, an employee entity may be described by the employee's name, salary and address. An entity type defines a collection of entities that have the same attributes. The relationship type is the concept to define associations between two or more entity types, and is formally defined as a subset of the Cartesian product of the participating entity types.

The relational data model is one of the traditional data models like the network model and hierarchical models. These have been quite successful in developing the database technology for many traditional data base applications. The relational data

model and algebra have a clear foundation in the proposal of Codd. Relational database design follows a waterfall –oriented approach. Firstly, a Conceptual schema is created, usually by means of entity-relationship model. Such a conceptual schema describes the universe of disclosure as a set of entities which are characterized by a number of attributes. In addition, relationship between entities can also be defined. In the second stage of relational database design, logical design, a relational schema is created in terms of a data definition language which can be processed by a DBMS. The transformation of ER schema into a relational schema is a semiautomatic process. Sometimes, specific constricts in the ER schema such as generalization relationships, may raise several alternative constructs in the relational schema. Although parts of the specified semantics in the ER schema may get lost in the relational schema, the latter can be enhanced by adding constraints or views.

Object –oriented concepts are slightly older than relational concepts. However, it took quite a long time for object-orientation to enter the main stream. In the late seventies, the first popular language Small- talk-80 was introduced by Xerox[GR89]. The following languages C++ and Java became widely used in industry. Parallel to the evolution of the object-oriented

programming languages, conventional languages were extended with concepts for implementing abstract data types and information hiding, in order to satisfy new requirements of software engineering. Examples of such languages are Modula-2 and Ada. However pure object oriented languages were considered the best choice for integrating new software engineering requirements, supporting reuse, maintaining software, sustaining the object oriented software engineering life cycle, and modeling the universe of discourse.

Later, persistency of data captured in objects was desired for seamless integration of applications and databases. Efforts in this field resulted in the object-oriented database system manifest[ABD+89], where the requirements for object databases were delineated, more than 20 years after the first concept was introduced.

The phase of software engineering typically comprise, amongst others, analysis, design and implementation. The main description of this chapter is organized as follows. The basic object-oriented modeling. Concerning the construction of the database schemas, the Object Database Management Group (ODBMG) has introduced a common schema definition language and query language.

4-SPECIFICATION OF OBJECT BEHAVIOUR

The example present in this category deals with shifting functionality from applications to the DBMS. In order to discover how far the inclusion of (object specific) behavioral information in an object oriented design process influences the resulting database schema. A UML class diagram representing the same geometric information before object specific behavior can be specified, objects have to be explicitly designed. When considering the relational schema, several objects are not explicitly modeled. For example, this is the case for composite objects, polylines and polygons. With regard to composite objects, they are identified in the relational schemas by those ID values in the general GeoObject table which appear in the CompID attribute of any tuple. They can be retrieved by the following query: Select distinct CompID from GeoObject.

From the relational point of view, there is no reason to explicitly model composite objects in the relational schema. It is sufficient that composite objects can be extracted from the database, where a view can be defined containing composite objects using the query mentioned above. In a object-oriented design, various object specific operations can be specified, for composite objects. Examples of such operations are rotate, move or scale, Or an operation boundary() which returns the smallest possible rectangle with angle0, enclosing all of its components but not intersecting any of them. Another example of objects not being modeled explicitly are points, which are used to define all kinds of basic objects. Points are not stored in an extra table. Instead, pairs of two attributes for the x-coordinates and y-coordinates are shared over most existing tables. Again, in the relational schema, there is no reason for an additional table point for two reasons. Firstly, a single point has no meaning without the basic object in which it is defined. Hence, every point is involved in exactly one composition relationship. Secondly, the union of all points presents in the database has no meaning. In other words, there is no need to define queries over all points.

In an object-oriented design, however, object specific operations for points can ideally be implemented as methods of a class point. In addition operations common to all objects like rotation or scaling can be easily implemented if a certain origin is specified for all objects. For rectangles, rounded rectangles and ellipses, the x and y values represent the origin. The origin of a line is the point specified by the values x1 and y1. Finally, for polylines and polygons, the coordinate taking the index value 1 is defined as the origin.

The main contribution of this chapter has been to demonstrate that relational and object-oriented database design follow different design strategies and consequently result in structurally different database schemas. There is no universally mapping strategy between relational constructs on the one side and object-oriented constructs on the other. In particular, this concerns the primary construct relations, tuples, classes and objects.

Relation VS Classes: The examples demonstrated that not every relation in a relational schema corresponds to a class in a corresponding object- oriented schema. Conversely, not every class present in a object-oriented schema is derived from a corresponding relation in a relational schema. The two main properties of object-orientation highlight this fact are object-specific behavior and encapsulation.

Tuples VS Objects: Not every tuple in a relational database is represented as an object-oriented database. Conversely, not every object is derived from one specific tuple. Reasons for this are the lack of conceptual modeling constructs in relational database design, and the possible specification of the object life cycles.

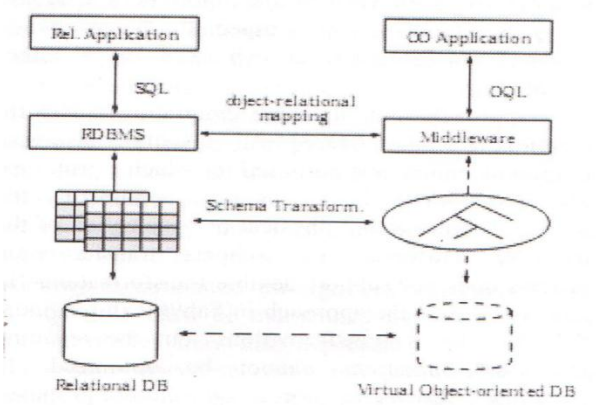
The contributions of this chapter from the requirements for the migration approach. The examples presented in this chapter demonstrate the consequence of these different design strategies, which is that converting a relational schema to a corresponding object-oriented schema is a non-trivial task. Consequently, the migration algorithms must be powerful enough to support such conversations.

5. STATE OF THE ART IN MIGRATION TO OBJECT TECHNOLOGY.

The wide acceptance of object technology in software engineering has motivated the combination object technology

and data persistence. Currently, the effort to smoothly integrate relational data and object technology is especially high when new object-oriented applications have to access existing relational databases. Relevant approaches are discussed and the advantages and disadvantages of each of them are compared. Database migration, the subject of this thesis, is one of the approaches. In general, two different approaches can be distinguished. Firstly, the data is retained in relational databases. The simplest solution, a gateway between a relational and an object-oriented database.

Database migration is the second approach, in which the RDBMs is completely replaced by an object-oriented one, and both schema and database must be migrated. The decision to apply a particular migration strategy for a certain database system depends on several practical constraints. For example, when old database applications still need to access existing databases, data migration cannot be chosen, unless the applications need to be reengineered for some reason.



Architecture of object view of relational databases

The transformation from the enriched relational schema to the object-oriented schema is usually processed automatically. In principle, every table of the relational schema is mapped into a class and foreign key constraints are mapped into reference attributes. Consequently, every tuple of the relational database is represented as an object in the object view. Some approaches allow lightly more flexible transformations and can, for example, support inheritance hierarchies on the object level. There as on why only rather trivial schema transformations are supported is the need for data conversion at run time. When transforming a schema to construct an object view, the functionality for data conversion, in one or both directions must also be specified. This deficiency is illustrated in next section.

6. OBJECT RELATIONAL MAPPING.

The transformation of object-oriented schema into a relational alone, when using objects for programs and relations for persistence. In some cases new object-oriented applications still tend to use relational DBMSs due to its advantages of maturity and widespread use. These approaches offer an automatic schema transformation process from an object-oriented schema to a relational schema. Since both object-relational mapping and relational–object mapping share the task of specifying mappings between relational constructs and object-oriented constructs, several object-relational mapping product also support the inverse direction. Research projects in this area are Penguin[KH93] and object Driver[Leb93].

The Construction of object views is not merely the inverse task of the object-relational mapping process, which can be performed automatically. In the case of object- relational mapping, the object-oriented schema has been created by forward engineering. When creating object views the relational schema must undergo a reverse engineering process.

In addition, schema transformation, object views also have to support data manipulation operations. However, the information on how to convert elements of the relational schema to elements of the object-oriented schema is usually not sufficient for automatically generating data manipulation expressions. Therefore, not all approaches support seamless data manipulation operations. List (key, Index, Value1, Value2, Value3, Value4).

In this relation, a list value is composed of tuples having a common value min the attribute key. The two ordering criteria of a list value are the index value and the attribute name. Two instances of the list values. Whereas on the object-oriented side typical list operations, like insertion of a value at some position, are provided, these must be implemented manually.

The algorithm for implementing the insertion operation is non-trivial. First, the index value of the tuple and the insertion point where the new element is inserted must be computed. Then, all the elements behind the insertion point must be shifted one position to the right. In some cases like the first list value, a new tuple must be inserted.

For combinational relational databases and object technology have been discussed. In principle, two different strategies exist: object views over relational database and database migration. Although both strategies seem to very different, often practical constraints determine which migration strategy to use. As regards, object views, various approaches have been proposed and several commercial tools already exist. Although this strategy exhibits serious problems, At least for performance reasons, it represents the current trend of migration to object technology.

In contrast, database migration has not received comparable attention. The reasons for this phenomenon are manifold. First, ser refrain from introducing OODBMS technology due to their low maturity and limited capabilities, that is, the market has not provided them with a robust, integrated object-oriented development and deployment tool set. Second, the few existing tools are not mature enough for commercial usage and are rather inflexible. This means that they show the same inadequacies as object views, except for performance gradation.

Other reasons for choosing object views lie in diverse practical constraints. Especially in old and barely maintained databases, it may be the case that a reverse engineering process does not extract the intended semantics of the database. Database migration requires a deeper insight into the semantics of the database. On the other hand, if legacy applications cannot be totally replaced by new ones, the data must reside as is in the relational database.

7. SEMANTIC ENRICHMENT.

Semantic enrichment is the task of gathering additional semantic information which is not explicitly available from the relational database system. This task is also known as reverse engineering, which emphasizes that the result of the semantic enrichment process can be represented through a conceptual schema. Whereas the process of database design is called forward engineering, reverse engineering can be considered the inverse process, that is, there construction of conceptual schema out of an existing database. Reverse engineering is not only essential in the field of database migration, but of high importance for information systems reengineering in general.

However, in contrast to forward engineering, reverse engineering has not received comparable attention in the literature.

Semantic approaches and CASE tools hardly exist. Reverse engineering of database systems is a topic of its own. There are various reasons for applying reverse engineering to databases, besides a subsequent database migration, as considered in this context. During the design and maintenance of the logical database schema, some domain semantics might not be captured anymore. Thus without possessing a conceptual view of the database it is difficult for users to understand the semantics of the database and retrieve data correctly. The need for conceptual schemas becomes more essential for redesigning existing databases when new application requirements are considered. Another area of application is database integration, which is best performed at the conceptual level[BLN86]. Finally, the semantics of the database must be extracted when switching to another data model. This is required for both object views and data migration.

8. SCHEMA TRANSFORMATION AND DATA MIGRATION.

Several approaches have been proposed for schema transformation from relational schema to object oriented ones. Most of these approaches have one aspect in common, that is, they provide a one step mapping, where every element of the target object schema is directly derived from an element of the relational source schema. These approaches are viable at best for small and well designed schemas which, for example are derived from an entity-relationship schema and are in third normal form(3NF). In addition, not all subsequent definition of all database emigration process. This is especially the case when schema transformation is expressed on a rather informal level.

Two approaches for database emigration are worth mentioned. In [AYCD98] an algebraic database migration approach is proposed for which a prototype also exists[AY98]. The focus is on optimizing the migration process and physical or organization of the database. However, the schema transformation process does not support flexible transformations. In contrast to this, the approach in [Fah96] exhibits more flexible schema transformations, but the resulting migration operations cannot be optimized. In particular, various transformation rules are proposed for both the relational and object-oriented model. Furthermore, the instance mapping must be applied after each individual schema transformation operation separately, and is expressed on a rather information level. As a consequence, the database migration process cannot be optimized. Some other approaches propose useful schema transformations in both, the relational and object-oriented[BP96] context. In relational database design, schema transformations have been used for reverse engineering[HTJC93b] or quality improvement[BCN92], in order to reduced efficiencies such as de normalization or optimizations. In object-oriented design on other hand, schema transformations have been used for schema refinement[BP96].

The purpose of traditional data models, such as relational or the object-oriented model is to allow a formal representation of the universe of discourse.

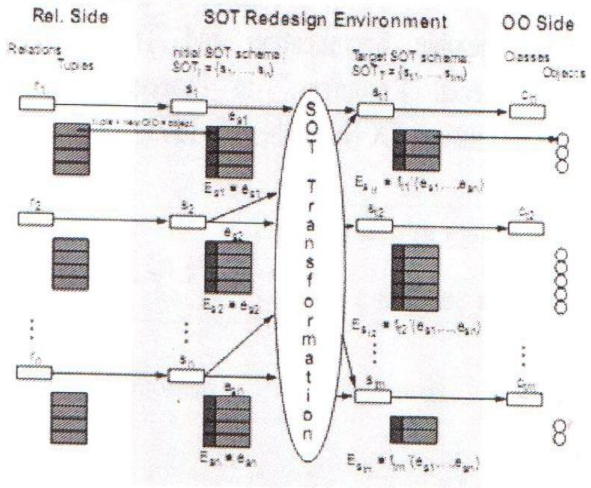
The support of schema evolution is not a major requirement of these data models. Existing algebras, like relational or object-oriented algebra are one way to describe a formal foundation. The algebras way may be used in different ways: as a formal semantics of the data and a query language itself. Besides optimization, the requirements for the SOT data model and algebra differ from existing approaches. Therefore there are not suitable for schema transformation and data migration, and a new approach is presented in this chapter.

The main purpose of SOT data model is the support of schema restructuring and data mapping. Schema restructuring is the act of modifying the SOT schema. Data mapping changes the database state, such that the resulting database is consistent with the modified SOT schema. Schema restructuring and data mapping are implemented by algebra. For simplicity, schema and data share the same algebra. Several reasons exists why neither the relational nor the neither object-oriented nor any other existing data model fulfill the requirements of database migration. The main reason is that the purposes of these data models are different from ours. Most existing algebras serve as theoretical foundation to describe the formal semantics of data and of a query language. However, the object-oriented data model does not support (formal) schema updates.

More precisely, the relational model lacks the identity aspect and the support of complex structures. Identity is simulated through key attributes whose uniqueness must be maintained by the user. Since key attributes may also represent contents of the universe of discourse, restructuring rules relaxing the uniqueness requirements of these attributes may cause problems. As regard complex structures, aggregates and sets may be simulated by additional relationships, but list or array structures cannot be expressed directly. On the other hand, the object-oriented model is too restrictive in object identity and inheritance handling. Schema restructuring cannot easily be propagated to the data level. The interface of an object or its class membership (usually) cannot be changed once it is created. As regards object identity, there can be no external influence on creating new object identifiers as will be demonstrated later. Concerning cardinality, the “not null” and “candidate key” restrictions, known from the relational data model, do not exist.

Conceptual model like the entity-relationship model, which are considered. The migration frame work, the main originality of the transformation process lies in the ability for flexible schema redesign. Input of the transformation process is a relational schema and a set of tables. On the object-oriented side, a schema expressed in the ODMG interface notation and data expressed in the ODMG object interchange format (OIF) is targeted. This way the approach is not dependent on the concrete target DBMS.

The data migration process is generated after completing these three steps of schema transformation.



9. SOT ALGEBRA, DEFINED OVER THE INITIAL EXTENSIONS.

Finally, in the third step SOT model is mapped to a class structure and the object-oriented database is created from the SOT extensions. The target class structure is expressed in the ODMG object definition language (ODL) [CB00], and the database is created either by creating a migration application or by sorting the SOT extensions in a dump file in the ODMG object t interchange format (OIF). Like the first step, this step can be performed automatically as well.

10. CONCLUSIONS.

Existing approaches for migration do not exploit the full potential of the object-oriented paradigm so that the resulting object-oriented schema still looks rather relational and retains the drawbacks and weaknesses of the relational schema. Therefore, one of the goals of this approach is to support schema transformation into an adequate object-oriented schema s obtained by forward engineering, rigorously using an object-oriented design method. In the first part of the paper, the fundamental differences between relational and object-oriented database are discussed. For the implementation of the database migration process an intermediate data model is proposed which allows defining both, schema transformation and data migration. This data model contains all object- oriented modeling constructs and supports flexible schema transformations. Furthermore, algebra is proposed for formal definition of the data migration process.

References

[1] Steven E. Minze, "A signaling protocol for Complex Multimedia services." IEEE Journal on Selected Areas in communications, vol. 9. No. 9. December 1991, pp.1383-1394.

[2] Stanley L. Moyer, Howard E. Bussey, Paul E. Brierley. "A Connection Management Interface for Fabric Control Application," in Proc. IEEE International Conference on Communications ICC'93, Geneva, Switzerland, May 23-26, 1993.

[3] Steven E. Minzer, Petros N. Mouchtaris, An Object-Oriented Solution for Signaling in Heterogeneous Multimedia networks," in Proc. Bell core Object-oriented Systems and Technology Symposium, 1993..

[4] Margret A. Ellis, Bjarne Stroustrup. The Annotated C++ Reference Manual, Addison-Wesley, Reading MA, 1990.

[5] S.-C (Tom) Soon, P.E. Brierley, et. Al. "Using State machines for Object-oriented Modeling," in Proc, Bellcore Object-Oriented Systems and Technology Symposium, 1991.

Author Profile

Dr Leelavathi Rajamanickam received PH.D degree in Computer Science and Engineering from University of Allahabad in 2011 and Master's degrees in Computer Applications from Kakatiya University in 2000, respectively. During 2001-2007, she worked as Head of Department in Lal Bahadur College, Warangal, India and from 2007-2012 worked in Malaysia.