Croatian Power Utility distribution level's UML model

LINDA VIĐAK, SVEN GOTOVAC
Department of Electronics
University of Split, FESB
R. Boskovica b.b., 21 000 Split
CROATIA

Abstract: - Large distributed system consists of large number of spatially dislocated elements. To develop efficient and optimal management of such a system it is necessary to collect information about system, i.e. to develop information system. This paper presents the development of UML model of information system for power distribution system of Croatian power utility, as an example of large distributed system.

Key-Words: - UML, information system, database, distributed systems, distribution level

1 Introduction

According to the definition, large distributed systems are systems that comprise large number of spatially dislocated, interacting elements. Mainly, those are different production systems like automated factories, water supply systems, waste water systems, electrical energy production, transmission and distribution systems, or traffic light supervision and control systems.

Efficient management (control, maintenance and development) of those large distributed systems consists of information collection about system on time, as well as information processing. That what creates a need for appropriate information system development [1].

Every distributed system is typical, so it is impossible to develop a unique information system for all of them, and in this paper, in the cooperation with Croatian Power Utility, we developed information system model for Croatian Power Utility distribution level.

Numerous methods can be used to design and manufacture modern information systems but no matter which method is chosen it is necessary to define a model of such a system. Model of the whole system, as well as the model of its parts, defines system information collection and processing, and activities that follows as a result of data processing [2] [3]. Today, there is a large number of approaches to system modeling, but in the moment one of the most popular and the most used is the Unified Modeling Language (UML) [4] [5] [6] [7] [8].

2 Croatian Power Utility distribution level organization

Complete Croatian Power Utility system is a very complex distributed system which function is to take care of electrical energy production, transport and distribution in Croatia. To achieve maximum system efficiency, to re-establish a balance between electrical energy production and consumption and in the same time to provide reliable and stable system work with minimal costs, Remote Control, Guidance and Supervision system (RCGS) is developed. At the distribution level it is organized at three levels. The first and the highest one is dispatching center. Its main purpose is to accomplish all dispatching functions, to control 35kV network and to make network states analyses for its own part of distribution network. Under the management of one dispatching center there are several control centers, that are concerned with underling 10kV and 0.4kV network. Their function is real time supervision, control and collected data analysis. Finally, on the lowest level there are remote stations and local guidance systems in the transformer stations. Their role is to collect, process and transmit the locally measured data, as well as plant control according to the directions of superior control center.

For example, in the dispatching center "Elektrodalmacija - Split" control area there is only one plant at the 110/35kV voltage level, about 40 35/10 kV plants and about 2000 10/0.4 kV plants. To integrate all of those components in the RCGS system would be technically very complex task, and the price would be unacceptable high. However, as the importance of all plants and their relationships is not the same for the system functionality, it is possible to include only more important ones to the RCGS system, to achieve satisfactory technical and functional level.

But, to realize even more efficient system the additional distribution level control and guidance improvements could be based on the plant state information. For example, information about plant's past sates, could be used to estimate its present states. That is especially important for ones out of RCGS system. To make this possible it is necessary to develop appropriate information system.

Collected data are also used for the maintenance and development purposes. That is another important function of developed information system.

2.1 Distribution level's information system

The main role (purpose) of any information system is to collect, store, process, and deliver necessary information important for an organization. The whole information system should comprise all business aspects of the organization: economical, technical and organizational. But in our work, for Croatian Power Utility, we took into account only the technical aspect of information system. Even with this restriction the development process wasn't easy at all. The system itself contains a large number of differently equipped plants, spatial system distribution enforces distributed data insertion, storage and usage, while measurements data collection is done by different performers. Moreover, information system is build for longer period of time, so it should be upgraded constantly and simple to maintain. Because of that, it is necessary to do qualitative analysis of the system development process and to use modern development tools in the information system development.

As the information system basis is the database to cover technical aspect of distribution level's information system two different databases are required. The first one contains information about electro distribution network elements' technical data, like information about plants, feeders, transformers, switches and disconnectors. The second one contains information about measurements that could be done in the plants. The first one already exists and it is Technical information system (TIS), and it is a bases for the other.

Development process of the second one, called MEASUREMENTS, is the main subject of this work. This information system (database) should comprise different kind of data like ones about electrical measurements and meteorological data.

3 The Unified Modeling Language introduction

The Unified Modeling Language (UML) is a general-purpose visual modeling language used to specify, visualize, construct, and document the artifacts of a software system. It is used to understand, design, browse, configure, maintain, and control information about such a system.

Like other languages it has its own syntax (graphical notation, set of diagram rules) and semantic, but it is not a programming language. There is a number of tools that can provide code generators from UML into a variety of programming languages (C, C++, Java), as well as a construct reverse-engineered models from existing programs. Moreover, UML is not intended to be a complete development method. It is just a tool that supports all, or at least most, of the existing development processes.

The fact that UML is not a development method but a development process becomes one of its benefits, because different organizations, different kind of projects, and different problem areas require different development processes, while the modeling language for development process could be unique. As UML supports the most of today’s development processes, and in the same time helps to solve the greatest modern problems of software development (large scale of distribution, concurrent work, and team development of system) it becomes a basis for the most development processes.

Various concepts and constructs in UML at the top level are divided into three areas: structural classification, dynamic behavior, and model management. Structural classification describes the things in the system and their relationships to the other things, and it includes: static view (class diagram), use case view (use case diagram), and implementation view (deployment and component diagrams). Dynamic behavior describes the behavior of a system over time, and it includes state machine view (state chart diagram), activity view (activity diagram), and interaction view (sequence and collaboration diagram). Model management describes the organization of the models themselves into hierarchical units called package.

4 The application MEASUREMENTS' UML model

In the application MEASUREMENTS it's required to collect and record different measurement data needed for plant's states estimation, their control and development. The main part of the application is a database, that should comprise two different kind of data: electrical and meteorological. Electrical measurement data comprise all electrical values that could be measured in the plant, like current, voltage, active and reactive power, active and reactive energy and resistance. Those values are measured at different plant bays (feeder, transformer, coupler, house transformer, …) at the 35/10kV and 10/0.4kV plants. Resistance measurements are related with plant's earth electrode resistance value or dead short current circuit of low-level voltage circuit resistance at the specific fields in the plant. Finally, the meteorological data represent daily temperatures and other general meteorological conditions information like overcast, wind strength and rainfall. They are very important to analyze and estimate the consumption according to the weather conditions.

Measurements could be performed in three different ways, so the application should provide different interfaces for data insertion. The simplest case is a single measurement by hand held measurements (voltmeter, ampere meter, wattmeter, etc.) where only one measurement value is measured, and the application would need an interface for that kind of measurements. A bit complex insertion interface is required for measurements with data logger, an instrument that can measure several values for longer period of time. Those data should be processed by special software and stored in database. Finally the most complex results are achieved from RCGS system, which stores the results into Paradox database. As we choose Oracle database management system (DBMS) for our database an interface for establishing connection between Oracle and Paradox was needed.

In the application MEASUREMENTS' development we used multitier architecture and application is realized with tools from Java Enterprise package.

With multitier architecture the system is divided at three levels: presentation, business and data level. Presentation level cares only about user oriented application interface, at user side. Data level is concerned with database, and it is represented by DBMS, while the main part of business level is to establish the connection between data and presentation level.

As we can see with multitier architecture client takes care only about presentation level, so it is called "thin client", while business level and database contact are spread over different components and can be developed on one ore even more servers.

'Thin' client applications are much easier to maintain, on the client there is only small chunk of code, while larger part of software logic is at server side, so all necessary updates, changes have to be done only on server side.

The problem with multitier application is its complexity. They require careful development process, they communicate with different services and server components.

4.1 Use case diagram

The first step in creation of UML model of the software system MEASUREMENTS is definition of the use case model, i.e. definition of use cases and users. From the point of future software system there are three groups of users. The first group enters data, the second uses those data, and the third one provides undisturbed work for the first and the second one. Every one of these users have an adequate use case, so the user INSERTER is connected to use case INSERT DATA, that defines data insertion. User READER is associated to adequate use case SEARCH AND PRINT, that defines different kinds of existing data searching. The third user is ADMINISTRATOR, who is responsible for maintaining the existing, functional state of database, so it is associated with use case MAINTAIN.

However, use case model is not so simple. Neither user INSERTER, nor use case INSERT DATA doesn't express clearly all the types of users, who will insert data, or all different ways of data insertion, respectively. So, it is necessary to define three different users KEYBOARD INSERTER, DL INSERTER and SDV INSERTER. First of them represents a user who inserts data collected through a single measurement, so it is necessary to define an adequate use case INSERT FROM KEYBOARD. The second one, DL INSERTER represents user who inserts data collected through data logger measurement. As those measurements required additional treatment in Excel, for insertion is defined use case INSERT FROM EXCEL. The last user in this group is SDV INSERTER, i.e. Remote Control System that stores data in Paradox database, so it is necessary to define use case that automatically transfer data from Paradox, and it is named INSERT FROM PARADOX. As we can see the user INSERTER is the generalization of users KEYBOARD INSERTER, SDV INSERTER, DL INSERTER, and use case INSERT DATA is the generalization of use cases INSERT FROM KEYBOARD, INSERT FROM EXCEL, and INSERT FROM PARADOX (fig. 1).

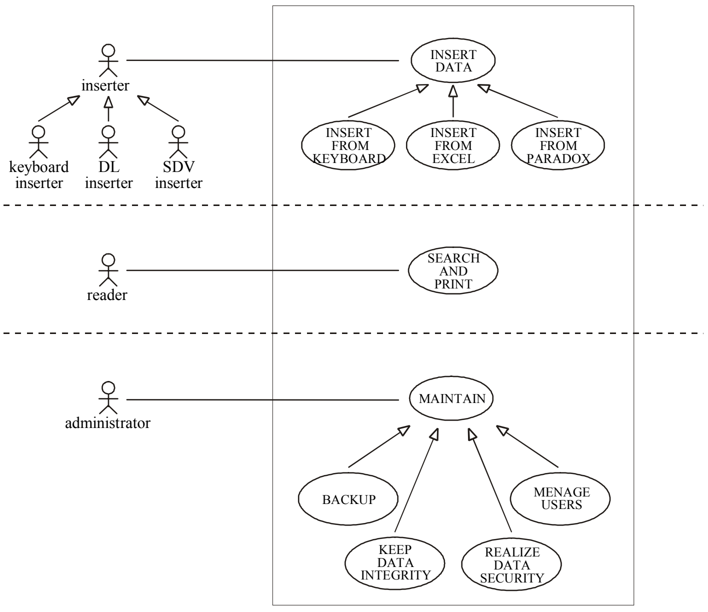

Fig. 1 – Use case diagram of application MEASUREMENTS

There is one more generalization in this part as we can see on fig. 1. Use case 'MAINTAIN' includes series of use cases that explain the administrator role:

- ➢ BACKUP – creates data backup copies;
- ➢ KEEP DATA INTEGRITY – takes care of data integrity;

- MENAGE USERS – adds new users, assigns and deletes privileges of existing users;
- REALIZE DATA SECURITY – realizes adequate data security level.

4.2 Class diagram

In general, according to multitier architecture all application classes can be divided into three groups. First group includes boundary classes, which represent graphical user interface (GUI) or presentation level of multitier architecture. Group of GUI classes in this application is collected into package called GUI CLASSES. Then there is a package of control classes called CONTROL CLASSES that represents business level. Finally, the third one is data level and it describes database tables through entity classes, so all classes from that level are collected into package called ENTITY CLASSES (fig. 2).

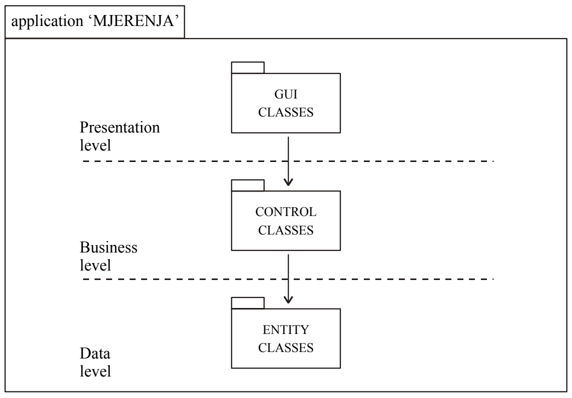

Fig. 2 – Package diagram

All classes in GUI CLASSES package could be divided into two main groups, one concerned with insertion of data, and the other with presentation. Class INSERT is a generalization of group of classes that describes different actions and interfaces required during the insertion of data. So, class WEB INSERTION describes new data insertion, WEB UPDATE describes existing data update, and WEB DELITION describes existing data deleting (fig. 3).

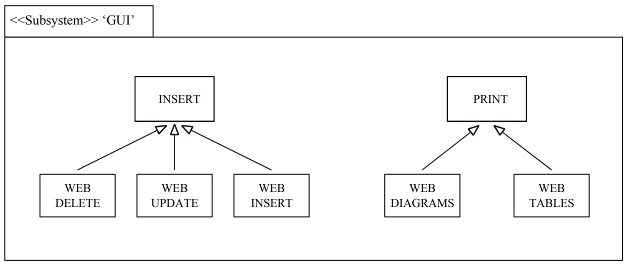

Fig. 3 – GUI package

Class PRINT is a generalization of two classes WEB DIAGRAMS and WEB TABLES that defines different ways of data presentation (fig. 3).

Control class package contains two sub-packages. ENTERPRISE JAVA package that defines application program interfaces (API-s) enabling running of Java web applications and Oracle runtime package represents Oracle database management system (DBMS). These two packages are interconnected through JDBC interface (Java DataBase Connectivity), that defines communication between Java application, and Oracle DBMS-a (fig. 4).

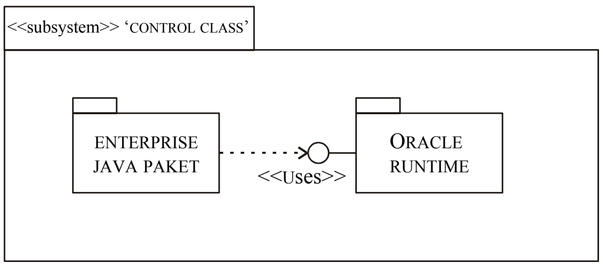

Fig. 4 – Control class package

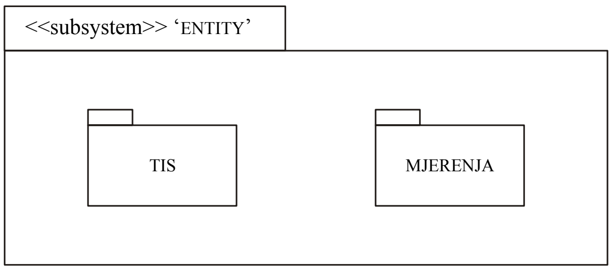

Fig. 5 – Entity package

As final product is relation database, entity package holds entity-relationship diagrams. As we can see on fig. 5, package ENTITY comprises two sub packages. One describes entity-relationship (ER) diagram of MEASUREMENTS' database, while the other does the same for database TIS. It is a database of Technical Information System that comprises information about all plants owned by HEP.

When all packages and associated classes are defined, it is necessary to define their interconnections. As we can see in fig. 6, Java Script (script language for dynamic web pages) and classes from ENTERPRISE JAVA package are used by web classes to establish connection with Oracle database management system (DBMS). The connection between ORACLE RUNTIME package and ENTERPRISE JAVA package is established through JDBC (Java Database Connectivity) driver. Depending on requirements Oracle DBMS then access to the adequate database.

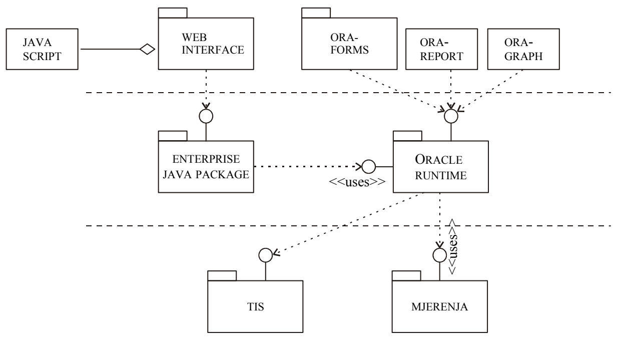

Fig. 6 – Interconnections between packages

4.3 Sequence diagram

Once when use case model and class diagram are finished we could start with realization of other ULM diagrams. First one is sequence diagram that shows time sequence of messages between objects. As we can see on fig. 7 this diagram shows three important objects to access the data (web client, web server, database server), each one corresponding to one level of multitier architecture. User initiates the interaction and messages are exchanged between them.

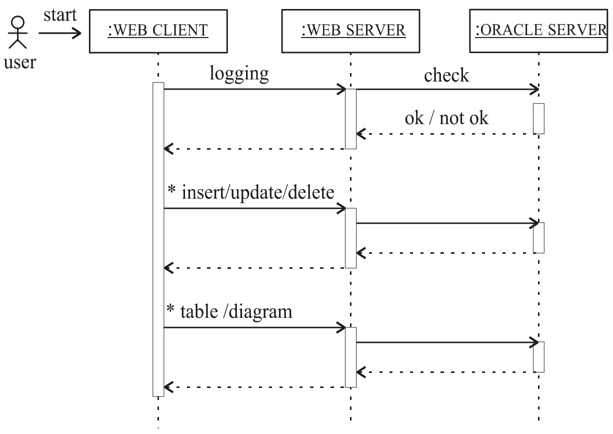

Fig. 7 – Sequence diagram

4.4 Activity diagram

When sequence diagram is defined, the next step is activity diagram. This diagram shows workflow of business activities. Because of that it is divided according to the activities that take place in database, i.e. insertion or presentation of data from database.

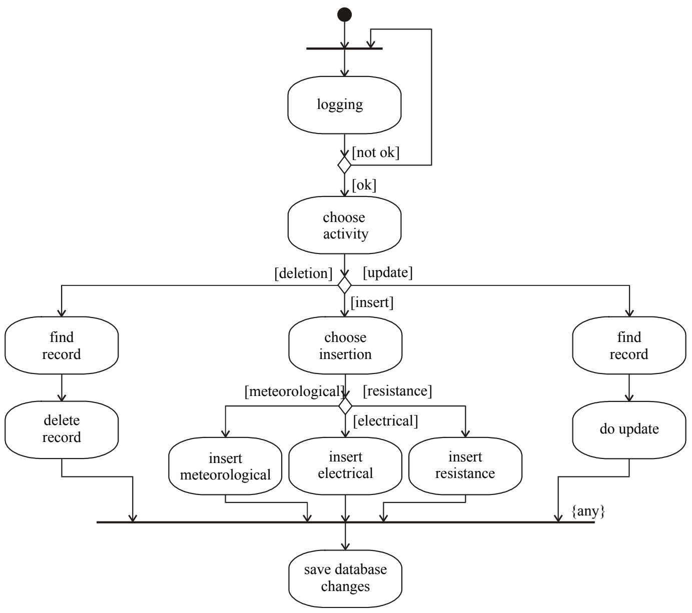

Fig. 8 – Activity diagram for insertion, updating, and deleting data

So, on fig. 8 we can see activity diagram for insertion, updating and deleting data from database, while a similar diagram for data reading is shown on fig. 9.

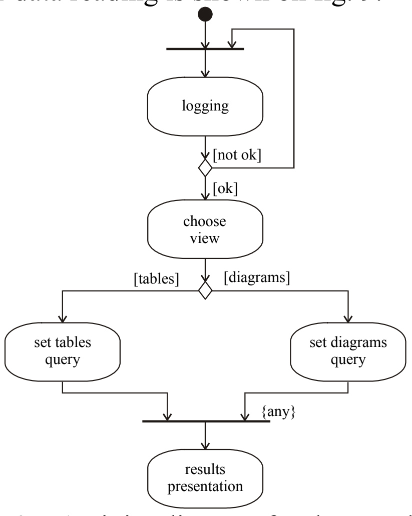

Fig. 9 – Activity diagram for data reading

4.5 Component and deployment diagram

These two diagrams are concerned about physical structure of system. So on fig. 10 is shown component diagram that describes software components of system, their interfaces, and dependencies. This diagram shows which user, through which interface accesses the database, and in addition it shows flow and direction of data and commands.

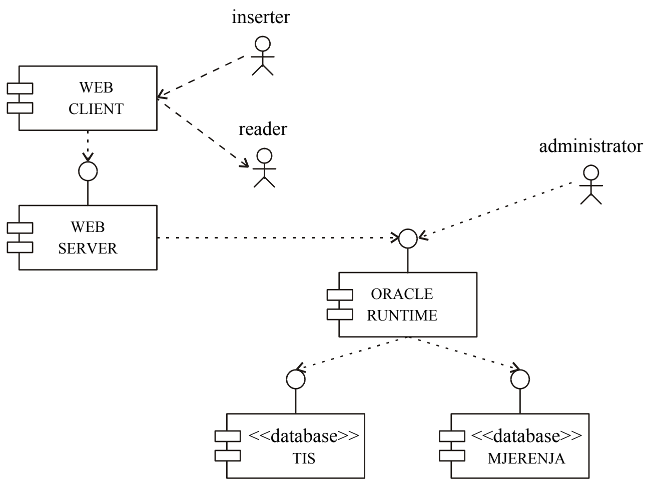

Fig. 10 – Component diagram

Deployment diagram shows arrangement of run-time components instances on node (a run-time resource, such as computer, device or memory) instances and their interconnections. In this example of distribution network the nature of problem suggests distributed approach. Optimal solution should be if every plant has its own database server with local data, and that data could be exchanged between plants. Such an approach is shown on fig. 11.

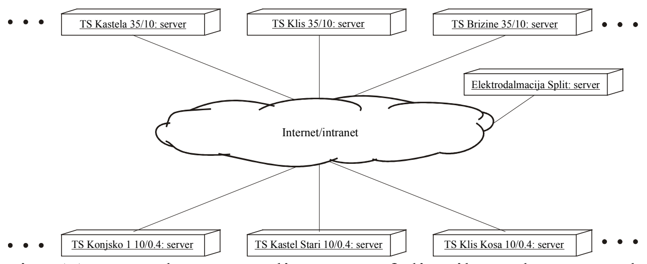
TS Kastela 35/10: server
TS Klis 35/10: server
TS Brizine 35/10: server
Elektrodalmacija Split: server
Internet/intranet
TS Konjsko 1 10/0.4: server
TS Kastel Stari 10/0.4: server
TS Klis Kosa 10/0.4: server

Fig. 11 – Deployment diagram of distributed approach

But, although this solution would be optimal, the first step in solving of the problem is a centralized approach shown on fig. 12.

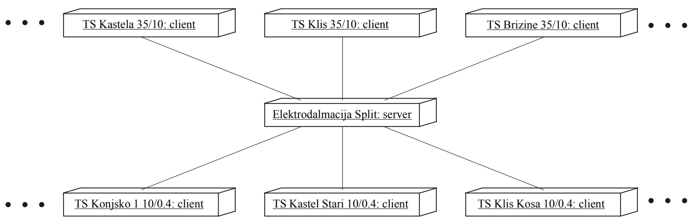
TS Kastela 35/10: client
TS Klis 35/10: client
TS Brizine 35/10: client
Elektrodalmacija Split: server
TS Konjsko 1 10/0.4: client
TS Kastel Stari 10/0.4: client
TS Klis Kosa 10/0.4: client

Fig. 12– Deployment diagram of centralized approach

5 Conclusion

Distribution system of Croatian Power Utility, like other large distributed systems, consists of a large number of spatially dislocated elements. To develop efficient and optimal management of such a system it is necessary to collect information about system, i.e. to develop information system.

In this case it was necessary to develop two different database systems. First one contains information about plants and their technical descriptions, and the second one that comprises data measured in the plants. The first one already exists and it is widely in use. Design of the second one is elaborated in this paper, and it is based on the UML model.

There is a number of benefits for use of UML model. The main one is the fact that several developers can work each on separate chunks of code, which are then deployed as specific components to comprise a complete application. In addition, it is industry wide accepted and defines seamless mapping from analysis to design and implementation. Once when a product is finished it is easy to maintain and upgrade system, or add new components.

References:

[1] X1. Sven Gotovac, Julije Ožegović, Lada Pravdica, Linda Viđak, *Remote control and supervision system research*, SoftCOM '98, 1998

[2] X2. Bruce Boes, *Development Information Systems, A New Paradigm in Software Development*, http://www.upspringsoftware.com/whitepapers/discover/dis.html

[3] X4. Bruce Boes, *Development information systems begin a software revolution*, http://www.serverworldmagazine.com/hpchronicle/2000/10/revolution.shtml, 2000

[4] X5. Lawrence Pfleeger, *Software Engineering: Theory and Practice, 2/e*, PrenticeHall, 2001

[5] X6. Ivar Jacobson, Grady Booch, James Rumbaugh, *The Unified Software Development Process*, Addison-Wesely, 1999.

[6] X7. James Rumbaugh, Ivar Jacobson, Grady Booch *The Unified Modeling Language Reference Manual*, Addison-Wesely, 1999.

[7] X8. *UML Tutorial –part 1*, http://www.sparxsystems.com.au/UML_Tutorial.htm

[8] X9. "*UML in CS320*", http://www.csci.csusb.edu/rootproj/cs320/uml/cs320wuml.html

[9] X10. M. Tamer Ozsu, Patrick Valduriez, *Principles of distributed database systems*, Prentice Hall, 1999