LAPPEENRANNAN TEKNILLINEN YLIOPISTO

LAPPEENRANTA UNIVERSITY OF TECHNOLOGY

Department of Information Technology

VLADIMIR LISS

Thick XML client and technology of its Construction

Master of Science Thesis

The topic of the Master of Science Thesis has been confirmed by the Department Council of the Department of Information Technology on the 9th of April 2003

Supervisor & 1st examiner: Prof. Jan Voracek

2nd examiner: Vladimir Botchko

Lappeenranta, 02.05.2003

Aviacionnaya street house 11, flat 12
196066, Saint-Petersburg, Russia
Phone: +7 911 212 69 69

TIIVISTELMÄ

LAPPEENRANNAN TEKNILLINEN YLIOPISTO
Tietotekniikan Osasto

Liss Vladimir
Thick XML client and technology of its construction
Diplomityö
2003

66 sivua, 15 kuvaa
Ohjaaja ja 1 tarkastaja: Prof. Jan Voracek
2 tarkastaja: Dr. Vladimir Botchko

Hakusanat: client application, XML-document, Rational Unified Process

Keywords: client application, XML-document, Rational Unified Process

Nykyään kolmeen kerrokseen perustuvat client-server –sovellukset ovat suuri kinnostuskohde sekä niiden kehittäjille etta käyttäjille. Tietotekniikan nopean kehityksen ansiosta näillä sovelluksilla on monipuolinen käyttö teollisuuden eri alueilla. Tällä hetkellä on olemassa paljon työkaluja client-server –sovellusten kehittämiseen, jotka myös tyydyttävät asiakkaiden asettamia vaatimuksia. Nämä työkalut eivät kuitenkaan mahdollista joustavaa toimintaa graafisen käyttöliittyman kanssa.

Tämä diplomityö käsittelee client-server –sovellusten kehittamistä XML –kielen avulla. Tämä lähestymistapa mahdollistaa client-server –sovellusten rakentamista niin, että niiden graafinen käyttöliittymä ja ulkonäkö olisivat helposti muokattavissa ilman ohjelman ytimen uudelleenkääntämistä.

Diplomityö koostuu kahdesta ostasta: teoreettisesta ja käytännöllisestä. Teoreettinen osa antaa yleisen tiedon client-server –arkkitehtuurista ja kuvailee ohjelmistotekniikan pääkohdat. Käytannöllinen osa esittää tulokset, client-server –sovellusten kehittämisteknologian kehittämislähestymistavan XML: ää käyttäen ja tuloksiin johtavat usecase– ja sekvenssidiagrammit. Käytännöllinen osa myos sisältää esimerkit toteutetuista XML-struktuureista, jotka kuvaavat client –sovellusten kuvaruutukaavakkeiden esintymisen ja serverikyselykaaviot.

ABSTRACT

LAPPEENRANTA UNIVERSITY OF TECHNOLOGY
Department of Information Technology

Liss Vladimir
Thick XML client and technology of its construction
Master of Science Thesis
March 2003

66 pages, 15 figures
Supervisor and 1st examiner: Prof. Jan Voracek
2nd examiner: Dr. Vladimir Botchko

Keywords: client application, XML-document, Rational Unified Process

Nowadays, development of the three-layers client-server applications causes big interest, as on the part of developers, and customers. Due to prompt development of computer technologies such applications find wide application in various areas of the industry. At present, there are a lot of tools for development of client-server applications, which adequately satisfy the requirements, which customer showed to the software product. However, for example, such resources do not respond possibilities of flexible operations with a graphic user interface.

This master thesis is devoted to the technology of development of the client-server application on the basis of a markup language - XML. Such approach enables to construct the client-server application with easily customized user interface and change of its appearance, without recompilation of the kernel of the program.

Master thesis consists of theoretical and practical parts. In the theoretical part, the overview of the client - server architecture is given, main phases of software engineering process are described. In the part, which contains practical results, the approach to development of the technology of development of a client-server-based application on basis XML is stated, use cases diagrams and the sequence diagrams describing the most important processes are resulted. The experimental part also, contains examples of the developed XML-structures describing appearance of screen forms of the client application and templates of requests to the server.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

AC	Application Client
AS	Application Server
DBMS	Data Base Management System
DBS	Data Base Server
DTD	Document Type Definitions
HTTP	Hypertext Transfer Protocol
JDBC	Java Data Base Connectivity
MSC	Message Sequence Chart
ODBC	Open Database Connectivity
RDA	Remote Data Access
RUP	Rational Unified Process
SQL	Structured Query Language
UML	Unified Modeling Language
XML	eXtensible Markup Language

FOREWORD

This thesis is a result of big experimental work I have participated in the period from June 2002 to September 2002 in Russian software developing company, during my summer holidays in Lappeenranta University of Technology. I would like to thank all people who helps and inspired me during my work.

The idea of this master thesis was born in the head of the director of the company – Gennady Razumovsky, who devoted and interested me in the problem. Thanks to Sergey Romanenko for advises in technical details of the work.

Especially thanks to Prof. Jan Voracek, who has agreed to supervise this topic and gave useful comments on structure of the paper. Also I wish express my gratitude to all people who made IMPIT program, which gave me possibility of graduation in Finland, especially, to Jan Voracek, the head of the program and Nina Kontro-Vesivalo, the IMPIT coordinator, who helped me a lot, during my study in Lappeenranta.

I would like to thank all my friends in Russia and in Finland who helped me with advices and deals. In particular, my special hugs to my fiancée Tatyana Galkina, who shared all my pleasures and problems with me, during my work.

Of course, I owe to my parents and sister for their wisdom, experience, constant support and help during my studying in Finland.

Lappeenranta, April 2003

Vladimir Liss

1. INTRODUCTION

Construction of an intelligence system is really a problem that should be solved on the majority of modern firms irrespective of the fact which sorts they are engaged in business. The term "intelligence system" concerns to the class of the software products facilitating, or "automizing" business dealing. [9]

Depending on a concrete usage, intelligence systems can differ very strongly on the functions, the architecture, and implementation. However it is possible to allocate, at least, two properties, which are common for all intelligence systems.

The intelligence system is intended for collection, storage and information processing. Therefore in a basis of any intelligence system the environment of storage and data access lays. The environment should provide a level of reliability of storage and efficiency of access that correspond to the usage of an intelligence system.

Intelligence systems are oriented on an end user. Such users can be very far from the world of computers. For them the terminal, a personal computer or a workstation represent only the instrument of their own professional work. Therefore the intelligence system is obliged to have the simple, convenient, easily mastered user interface which should give an end user all functions necessary for his work, but at the same time not enable him to fulfill any unnecessary operations. [9]

All approaches to organization of intelligence systems are based on the common architecture "client - server". Distinction will consist only that clients and servers do.

Nowadays there is very extensive kit of tools for development the client - server of applications. Conditionally, it is possible to divide them on some classes, main classes of development tools of applications are:

- Traditional programming systems and their visual successors;
- Modern desktop DBMS as components for construction of applications the client – server;
- Specialized resources for development of applications the client - server;

- Resources of automation of office activity;
- Resources of complex automation of development of applications;

Traditional programming languages, such as C, Pascal and Cobol historically were the first resources used for development of applications. Thus for each database server there was a library for the defined client platforms, providing a set of functions for interaction with the server. On a measure of development of the concept of visual programming in all environments of development sets of the specialized components have appeared, permitting to build applications the client - server. As a rule, these components, on the one hand, allow create rather easily the simple application, on the other hand, at development of really big complexes they are not too effective from the point of view of productivity, usage of specific possibilities of a concrete database server, and also implementation of complex operations above the database. The plus of such development tools it is necessary to relate their wide prevalence, the big flexibility from the point of view of the programmer, possibility of optimization of requirements to the equipment at the expense of careful writing the code. Besides for the majority of classical development tools specialized libraries for operation with the concrete server platform recently are created. [6]

Other big class of the development tools used at creation of the architecture "client - server", local DBMS are. Originally in a package from any local DBMS the specialized programming language was used. These programming languages were used by the big number of programmers, on them huge size of the software has been developed. Therefore when necessity of carry of the developed software from local tasks and in a client-server architecture has appeared, the solutions have been naturally offered, permitting to use these specialized development tools with database servers. Thus usage of the accumulated knowledge and available developments that accelerated was achieved and facilitated creation of applications the client - server. On the other hand, this approach has also the minuses. Local DBMS initially intended for solution of other tasks. Therefore the programs developed with their help frequently turn out bulky and ineffective. Further, the DBMS as appreciably concedes the programming language in expressiveness to classic languages of programming. And, at last, the important

factor is that similar applications, as a rule, are rather exacting to resources of a client place. [6]

One more widely widespread class of tools is the specialized development tools initially oriented to creation the client - server of applications. They, in turn, can be broken at the systems created by developers of the concrete server and oriented, first of all, on operation with the given server, and the systems of wide assignment providing operation with a plenty of servers. The given variant of creation of sentences allows in the greatest measure and most simply realizing functions of operation about the client - server the database. Thus the closest integration of the server and the client is provided. Disadvantages of this approach are a natural continuation of its advantages. Orientation to a client-server architecture results in usage of specialized programming languages which are less universal, the applications written on them are worse adapted to solution of common tasks, and also, as a rule, demand powerful client places.

For access to database servers can be used as well desktop office packages, such as an Excel. This variant has appeared in connection with very wide circulation of these products. The given variant is applied, as a rule, in the complex with other development tools to processing those or other output forms, or received data sets. Advantage of these variants is habitual to the user the interface and possibility of usage of rich possibilities of these office products at information processing. The main disadvantage - extreme needs in the resources for the created applications. Besides the programming languages accessible in office products are not simply poor possibilities for programming, and very poor.

The integrated development tools of applications too are used at creation of client-server systems. These resources provide complete cycle of creation of an intelligence system, from documenting business of the process on firm and designing of structure and components of the application before obtaining ready program units. The given resources demand very high qualification of developers and a high degree of formalizing of the automized task. At execution of these conditions they allow to receive prototypes of the big systems for short enough period. Main disadvantages of this variant are very high cost of products and their studying, presence of unique programming languages and, again, high requirements to the hardware.

Generally speaking, unambiguously to specify the development tools most suitable in a concrete case it is practically impossible. In practice, choice of this or that software product for creation of the system is carried out on the basis of the big number technical, organizational and even political factors, in view of all previous history of development of intelligence systems for concrete firm.

Modern line of development practically all developments is the increasing and greater rapprochement of tools of various classes among themselves. On the one hand, traditional programming languages become more and more visual, and their components for development of applications the client - server - more and more specialized. On the other hand, desktop DBMS and specialized development tools everyone are more closely integrated with classical resources. Recently already there are the products joining in and specialized toolkit for construction of the application, oriented on operation with a database server and the compiler from a classic language of programming. [6]

Now at designing and development of information technologies such tools and system resources, as are widely used:

- Industrial DBMS (ORACLE, MS SQL Server, Sybase Adaptive Server);
- Designing tools (Power Designer, BPWin, Rational Rose);
- Systems and languages of development (Power Builder, JBuilder, PowerSoft +, Optima ++, Visual Basic, Visual C ++, MapBasic, Borland C ++ Builder);

The considered tools appreciably meet the main requirements showed to them at creation of intelligence systems. At the same time, there is a lot of requirements at creation of modern intelligence systems which are not supported by the considered tools. To such requirements concern:

- Possibility of dynamic change of external representation of the user interface of the client-server application;
- The universal format for information interchange between separate components of the big application;
- Possibility of usage of XML-documents as intermediate data format in systems with three tier architecture;

- Representation of requests results to a database in some universal format in such a manner that the part of a DBMS, becomes "transparent" for the application;

To such kind of requirements satisfies XML. XML language is a new SGML-derivative markup language of the documents, permitting to structure the information of different type, using for this purpose an arbitrary set of instructions.[15]

The present master thesis is devoted to development of the technology and implementation of client-server applications based on XML.

Usage of XML-documents for the description of external representation and filling of screen forms provides possibility of fast development and easy customization of the client-server applications. The suggested technology allows changing external representation of the user interface forms of client applications without change and recompilations of the initial code of the program. Except for it the suggested technology is universal for implementation of client applications and does not depend on the language of implementation of the functional components. Properties of expandability of the XML-document allows to enter the new units describing the components specific to everyone application of user interface and does not demand essential processing of the existing initial code.

2. OVERVIEW OF THE “CLIENT-SERVER” ARCHITECTURE

The architecture " the client - the server " is known already enough long time, but earlier was used in large networks of a scale of firm more often. Today, with development of Internet, this technology even more often involves looks of software developers. It can be presented so:

- The client forms and dispatches request to the database of the server is truer - to the program handling requests;
- This program makes manipulations from a DB stored on the server, according to request, forms result and transfers it to the client;
- The client receives result, displays it on the display and waits for the further operations of the user. Cycle repeats, while the user will not complete operation with the server;

The server is a special program which accepts request from the user's application, handles it on local for the server returns to the client application of the user some result of processing.

The program - client can be as on the same computer, as the program - server, and on another, exchanging the data with the program - server through the computer network. It is a classical (two-tier) client-server architecture. At development of applications in a classical client-server architecture (at usage of the computer network) the software of data storage allocates on the physical computer - server (for example, a database server), the interface with the user is placed on the computer - client, and is necessary to arrange data processing between client and server computers. Splitting of algorithm of data processing on two physical systems can frequently result in inconsistency of various units of the architecture. That it to avoid, all data processing can be placed either on the computer - server, or on the computer - client. These both approaches have the disadvantages. [8]

If all on the client (the "thick" client):

- Administration becomes complicated;
- Upgrade of the software becomes complicated;
- There are complexities at distribution of powers since access is differentiated not on operations, and on the data;
- The computer network is overloaded;
- Data protection is weakened since difficultly correctly to arrange powers.

If all on the server (the "thin" client):

- Implementation since becomes complicated. DBMS-LANGUAGES are badly adapted to development of such software and have no good resources of debugging;
- Productivity of the programs written in DBMS -LANGUAGES, a little bit below, than created in other languages that is of great importance for complex systems;
- Insufficient reliability of the programs written in DBMS -LANGUAGES, can result in failure of all database server;
- Programs in DBMS -LANGUAGES are intolerable on other database management systems. [9]

For solution of the given problems it is possible to install one more (or more) the physical server and to place on it handlers of the data. Such architecture is named as a three-level (multilevel) client-server architecture, and the intermediate server name as the server of applications. More in detail the given architecture will be considered below. [8]

One of main principles of the technology "client - server" consists in sharing functions of the standard application on three groups having the various nature. The first group is a function of input and mapping of the data. The second group unites only applied functions, characteristic for the given data domain. At last, fundamental functions of storage and data management concern to the third group (databases, file systems, etc.)

According to it in any application the following logical components are allocated:

- **Components of representation** (presentation), realizing functions of the first group;
- **Applied components** (business application), supporting functions of the second groups;
- **Components of access to information resources** (resource access) or the manager of resources (Resource manager), supporting functions of the third group.

Distinctions in implementation of applications within the framework of the technology "client - server" are defined by three factors. First, that, what mechanisms are used for implementation of functions of all three groups. Secondly, as logical components are arranged between computers in the network. Three approaches are allocated, each of which is realized in appropriate model:

- Model of access to remote data (Remote Date Access - RDA);
- Model of a database server (DataBase Server - DBS);
- Model of the server of applications (Application Server - AS).

In RDA-model codes the component of representation and applied the component are combined and will be fulfilled on the computer - client, last supports both functions of input and representation of the data, and only applied functions. Access to information resources is provided, as a rule, with operators of the special language (SQL language, for example, if the question is databases) or calls of functions of the special library (if is present appropriate API). Requests to information resources are routed on the network to the remote computer (for example, to a database server). Last handles and fulfils requests and returns to the client bulks (Figure 1). Speaking about the architecture "client - server", in most cases mean this model. [8]

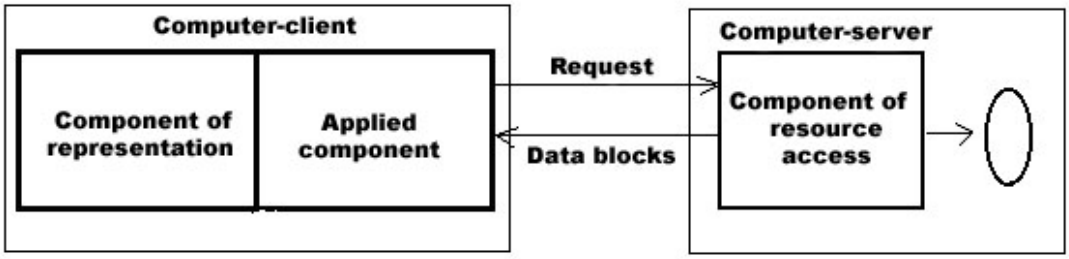

Figure 1. Model of access to remote data.

The DSB-model (Figure 2) is created in the supposition, that the process fulfilled on the computer - client, is limited to functions of representation while purely applied functions are realized in stored procedures which also name as compiled resident procedures, or they are stored by procedures of the database directly in the database and fulfilled on a computer - database server (where functions also the components, controlling data access, that is a kernel of a DBMS). The concept of an information resource is narrowed down up to databases as the mechanism of stored procedures - the distinctive characteristic of DBS-model - is present while only in a DBMS, and that not in all.

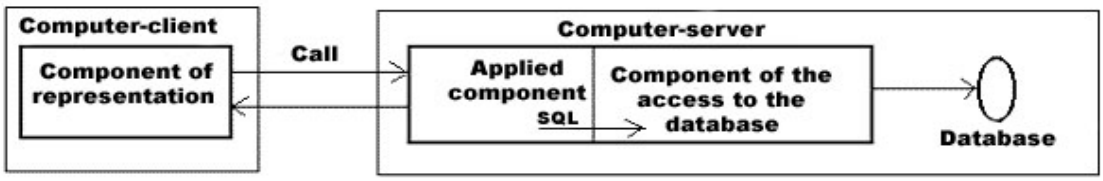

Figure 2. Model of a database server.

In practice the mixed models when support of integrity of the database and some elementary applied functions are supported by stored procedures (DBS-model) are frequently used, and more complex functions are realized directly in an application which is fulfilled on the computer - client (RDA-model). However such solutions including units at once of two models cannot change in essence our representations about their relation. [8]

In AS-model (Figure 3) the process fulfilled on the computer - client responds, as is usual, for input and representation of the data (that is realizes functions of the first group). Applied functions are fulfilled by group of processes (servers of applications), functioning on the remote computer (or several computers). Access to the information resources, necessary for solution of applied tasks, is provided with exactly same way, as in RDA-model. From applied components resources of various types - the databases, the indexed files, queues are accessible, etc. Servers of applications are fulfilled, as a rule, on the same computer where the manager of resources functions, however can be fulfilled and on other computers.

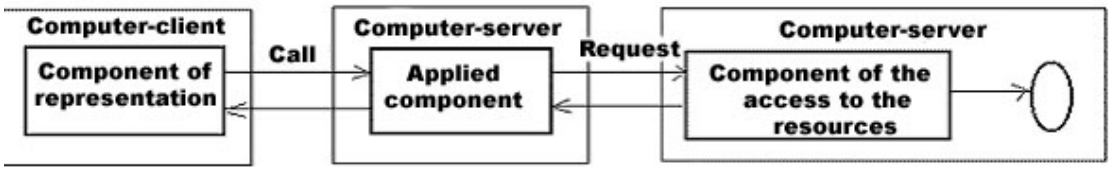

Figure 3. Model of the server of applications.

In what fundamental distinction between these models consists? RDA-and DBS-models base on the two-tier scheme of sharing of functions. In RDA-model applied functions are handed to the program - client, in DBS-model the responsibility for their execution incurs a kernel of a DBMS. In the first case applied components merges with a component of representation, in the second - it is integrated in components of access to information resources. On the contrary, in AS-model the scheme of sharing of functions where applied components it is selected as the major unit of the application is realized classical three-tier scheme, for its definition universal mechanisms of the multitask operating system are used, and interfaces with two other components are standardized. Purely, its advantages, which have the major value for only practical activities, also follow from this feature of AS-model. [8]

2.1. RDA-model

Main advantage of RDA-model lies in a practical plane. Today there is a set of the tools providing fast creation of desktop-applications, working with SQL-oriented a DBMS. The majority of them supports a graphic interface of the user in a MS Windows, the

standard of ODBC interface, contain resources of automatic code generation. Differently, main advantage of RDA-model consists in unification and wide choice of development tools of applications. Overwhelming majority of these development tools in fourth-generation languages (switching and resources of automation of programming) just also creates codes in which applied functions and functions of representation are mixed.

The RDA-model has a number of limitations.

First, interaction of the client and the server by means of SQL-searches essentially loads the network. As the application is unallotted and all its logic is localized on the computer - client so far as the application requires transmission on the network of the data of great volume, probably, redundant. As soon as the number of clients increases, the network turns to " a neck of a bottle ", braking speed of all intelligence system.

Second, satisfactory administration of applications in RDA-model is practically impossible. It is obvious, that if various functions by the nature (to the function of representation and only applied functions) are mixed in the same program written in 4GL language if necessary changes of applied functions it is necessary to copy all program entirely. By collective operation above the project, as a rule, each developer is entrusted with the implementation of separate applied functions that makes impossible a control behind their mutual consistency. Each of developers should program the interface with the user that puts uniform style of the interface and its integrity under a question. [8]

2.2. DBS-model

Despite of a wide circulation, RDA-model gives up the place to more technological DBS-model. Last it is realized in some relational DBMS (Ingres, Sybase, Oracle). Its basis the mechanism of stored procedures - makes a resource of programming of a kernel of a DBMS. Procedures are stored in the data directory, divided between several clients and fulfilled on the same computer where the nucleus of a DBMS functions. The language, on which stored procedures are developed, represents the procedural extension of query language SQL and is unique for each concrete DBMS. The attempts

of regularizing SQL concerning stored procedures yet have not resulted in appreciable success. Besides in many implementations of the procedure are interpreted, that makes their execution by slower, rather than execution of the programs written in languages of the third generation. The mechanism of stored procedures - one of composite components of an active database server. [8]

In DBS-model the application is distributed. Components of representation it is fulfilled on the computer - client while applied components (realizing business - functions) it is made out as a set of stored procedures and functions on the computer - server of a DB. Advantages of DBS-model before RDA-model are obvious: it and possibility of the centralized administration of business - functions, and lowering of traffic of the network, and possibility of sharing of the procedure between several applications, and saving of resources of a computer at the expense of usage of once created schedule of execution of the procedure. However there are also disadvantages.

First, the resources used for writing of stored procedures, strictly speaking, are not programming languages true. More likely, it - the various procedural SQL extensions which are not maintaining matchings on graphic resources and functionalities with languages of the third generation, such as C or Pascal. They are built - in concrete DBMS, and, naturally, frameworks of their usage are limited. Therefore, the system, in which applied components it is realized by means of stored procedures, is not mobile concerning a DBMS. Besides in the majority of a DBMS there are no possibilities of debugging and testing of stored procedures that transforms last into rather dangerous mechanism. Really, a little complex not organized combination of operation of triggers and start of procedures can, on neat expression of one of developers, " completely to carry all database ".

Second, the DBS-model does not provide required efficiency of usage of computing resources. Objective limitations in a nucleus of a DBMS do not allow while to organize its frameworks effective balance of loading, migration of procedures on other computers - servers of a DB and to realize other useful functions. Attempts of developers of a DBMS to provide in the systems these possibilities (the distributed stored procedures, searches with priorities, etc.) yet do not allow to achieve desirable effect.

Thirdly, decentralizing of applications (one of key factors of modern information technologies) demands essential variety of variants of interaction of the client and the

server. At implementation of the application system such mechanisms of interaction, as stored queues, asynchronous calls, etc. which in DBS-model are not supported can be necessary.

Today hardly it is possible to speak that stored procedures in their present state represent the adequate mechanism for the description of business - functions and implementations applied the component. To transform them into really powerful resource, developers of a DBMS should play back in them the following possibilities:

- The extension graphic resources of languages of procedures;
- Resources of debugging and testing of stored procedures;
- Preventing conflicts of procedures with other programs;
- Support priority request processing.

Meanwhile these possibilities are already realized in AS-model which to the greatest degree mirrors strengths of the technology "client - server".

2.3. AS-model

Basic element accepted in AS-model tree-tier architecture is the server of the application. In its frameworks applied functions are realized some, each of which is made out as service and gives some services to all programs which wish and can take advantage of them. Servers of applications can be a little, and everyone them gives the defined set of services. Any program, which uses them, is considered as the client of the application (Application Client - AC). Details of implementation of applied functions in the server of applications are completely hidden from the client of the application. AC accesses with search to the concrete service, but not to AS, that is servers of applications are depersonalized and are only some kind of "framework" for design of services that allows controlling balance of loading effectively. The requests acting from the AC are drawn up in queue to the AS-process, which extracts and transfers them for processing to the service according to priorities.

The AC is treated more widely, than components of representation. It can support the interface with an end user (then it is a component of representation), can provide arrival of the data from some devices (for example, sensors), and can be in itself, at last, AS.

The latter allows realizing the application system containing AS of several levels. The architecture of such system can look as a kernel surrounded by concentric rings. The kernel will consist of servers of applications in which base applied functions are realized. Rings symbolize sets AS being clients in relation to servers of a lower layer. The number of levels of servers in AS-model, generally speaking, is not limited.

It is uneasy to see, that the AS-model has universal character. Precise differentiation of logical components and rational choice of software for their implementation provide models such level of flexibility and an openness which while is inaccessible in RDA- and DBS-models.[8]

3. PROBLEM DEFINITION OF DEVELOPMENT OF THE TECHNOLOGY OF CONSTRUCTION AND IMPLEMENTATION OF THE "THICK" CLIENT

The purpose of the master thesis consists in development of the technology of designing and implementation of client applications on the basis of the description of external representation of the interface forms in the XML-format. Suggested technology should provide creation of client applications irrespective of the selected architecture, the technology and protocols of interaction client and server-based applications. Besides possibility of change of user interface, without recompilation of a kernel of the client application should be realized, and also the template of request to the server, not demanding modification of all system is developed, in case of modify structure of the database.

For reaching an object in view during execution of operation it is necessary to solve the following tasks:

- Development of the scheme of the XML-document for the description of the interface forms;
- Designing and implementation of a main kernel ("engine") systems of visualization which carries out conversion of the XML-description directly in the screen form during operation of the client application;
- Implementation of the pilot project showing to possibility of the suggested technology of development of client applications.

For the description of scheme XML - the document DTD standard (Document Type Definition), developed by corporation W3C [17] is used. DTD is a formal description of requirements to XML document. [17] In the projected XML-document possibilities for the description of the following units of the interface forms and functionalities of the client application should be stipulated:

- Sizes of forms (length, width, height);
- Sizes of separate components (the table, the text boxes, dropping out lists, radiobuttons, checkboxes, buttons);
- Sizes of fonts;

- The type of fonts;
- Color of fonts;
- Layout of components on the screen form (alignment on edges);
- Searches of the data about the server;
- Rules of check of a control of a syntactical and semantic correctness of the entered data;
- Rules of event processing on the form;
- Rules of printing of contents of the screen form;
- The help on using the form.

For designing a main system kernel programming language Java in which convenient and powerful resources of operation with XML-documents now are realized is used. The system kernel of visualization is built into the client application and fulfils the following operations:

- Loading the description of the form and its representation;
- Interaction with the user;
- Interaction with the server.

4. SOFTWARE DEVELOPMENT PROCESS SELECTION

Primary factor of success of the project on development of an intelligence system is presence of well defined and well controlled process. Engineering process is the sum of various sorts of the activity necessary for conversion of requirements of the user in an intelligence system. Now the leader among the engineering processes supporting the object-oriented methodology, is the Unified engineering process of the intelligence systems selected as engineering process in the given degree project. It is stipulated by that, that Unified the process is the generalized frame of the process which can be specialized for a wide range of intelligence systems, various usages, levels of the competence and sizes of the project. [2]

The unified process *is component - oriented*. It means, that the created intelligence system is created on the basis of the program components connected by formally defined interfaces. For developments of graphics models of an intelligence system in the Unified process the Unified Modeling Language - UML is used.

Main aspects of the Unified process consist that it is controlled by requirements (variants of usage), architecturally - oriented, iterative and incremental.

The intelligence system forms for service of its users. The concept "user" of the Unified engineering process concerns not only to people working with the system, but also to other (external) systems. Thus, the concept "user" concerns to somebody or something that cooperates with a developed intelligence system. Interaction of users with the system is described by variants of usage. The variant of usage is a part of functionality of the system, necessary for obtaining by the user significant for him appreciable and measurable result. The sum of all variants of usage makes model of variants of usage, which describes complete functionality of the system. The model of variants of usage replies: that the system presumably makes for each user. The model of variants of usage allows presenting functionality of the system in concepts of result, significant for the user, and not just in concepts of functions, which the system should realize. Variants of usage are not only a resource of the description of requirements to the system. They also route its designing, implementation and testing, that is they route

engineering process. Variants of usage are defined, projected, realized and are input data on which testing the developed intelligence system is carried out.

The unified process - *controlled by variants of usage.* It means, that engineering process passes series of the working processes generated by variants of usage.

As variants of usage control engineering process they form in pair with the architecture of the system. Thus, variants of usage control the architecture of the system, and the architecture of the system influences choice of variants of usage.

The concept of the architecture of an intelligence system includes the most important statistical and dynamic aspects of the system. The architecture grows from requirements to the system. These requirements are mirrored in variants of usage. However they also depend on set of other factors, such, as choice of the platform for operation of an intelligence system (that is the computer architecture, the operating system, a DBMS, network protocols), usages of ready program components, requirements to allocation, usage of the inherited systems and nonfunctional requirements (for example, productivity and reliability). The unified engineering process helps the architect of the system to concentrate on the correct purposes, such, as clearness, ease of modification and possibility of a reuse. The architecture should be developed so that to allow the system to develop not only at the moment of initial development, but also in the future generations. To construct such architecture it is necessary to understand completely key functions that are key variants of usage of the system. These key variants of usage make from 5 up to 10 % of all variants of usage and are extremely important, as contain functions of a system kernel.

The unified process - *architecturally - oriented.* It means, that engineering process necessarily includes the following stages:

- Creation of a rough sketch of the architecture, since that part which is not connected to variants of usage (the so-called platform). Though this part of the architecture does not depend on variants of usage for its development it is necessary to understand variants of usage in general.
- Operation with a subset of the allocated variants of usage, each of which corresponds to one of key functions of the developed system. Each of selected variants of usage in details is described and realized in concepts **of subsystems**, **classes** and **components.**

- After variants of usage are described and completely developed, the most part of the architecture is researched. The created architecture, in turn, will be base for complete development of other variants of usage.

Peak efficiency of engineering process of an intelligence system is achieved at usage of the iterative approach. Iterations concern to steps of working processes. Tasks which should be solved on each iteration, are selected under effect of two factors. First, during iteration it is necessary to work with group of variants of usage, which raises applicability of the system during the further development. Second, during iteration it is necessary to be engaged in the most serious risks. Sequential iterations use results of development in that sort in which they have remained at the termination of the previous iteration. On each iteration, developers determine and describe pertinent variants of usage, create the project using the selected architecture as directing, realize the project and check correspondence of components to variants of usage. If iteration has achieved the purpose - engineering process passes to the following iteration.

The unified engineering process assumes splitting life cycle of an intelligence system into four phases (Figure 4, taken form [18]): diagnostic inspection, designing, construction and implantation. On each of these phases all main working processes stipulated by the object-oriented methodology are fulfilled. Difference of one phase from another consists in on what processes emphasizes.

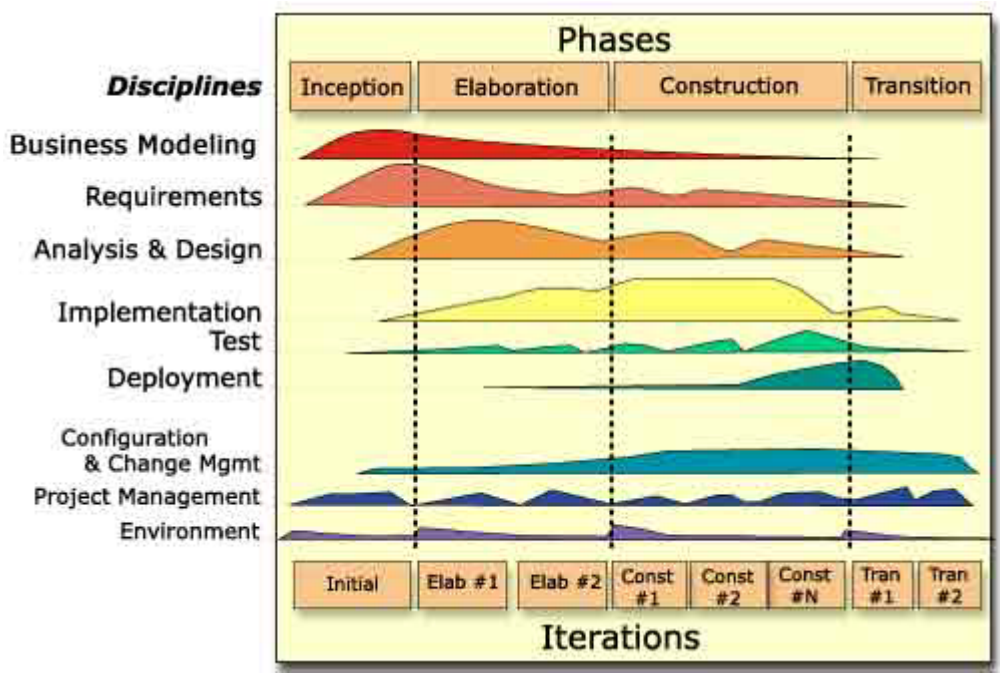

Figure 4. Phases of the Rational Unified Process

During a phase " Business modeling " is developed the concept of the future intelligence system and there is a schedule of its development. In particular, on this phase answers to questions should be obtained:

- What the system should do first of all for its main users?
- How the architecture of the system should look?
- What schedule and in what system development will cost?

The simplified model of variants of usage, containing the most critical variants of usage, gives the answer to the first question. At the same stage there is an initial variant of the architecture which contains the most important subsystems, come to light and placed on приоритетности the most important risks, the phase of designing is in details planned and the project is roughly estimated.

Results of operation at stages of the Unified process are artefacts. **The artefact** is a common name for any kinds of the information created, an intelligence system changed or used at creation. There are two main such as artefacts - models and text documents. The model in the Unified process is the abstraction describing the modelled system from the defined point of view and at the defined level of abstraction. The model unambiguously shows, that will take place in the system at an occurrence of an event,

described in it. The model also can describe interaction between the system and its environment. Main models in the Unified process, connected to development of the architecture, are the model of variants of usage, conceptual model, model of designing and model of allocation. The model of variants of usage defines application of the system. The conceptual model represents main concepts of data domain and link between them. The model of designing and model of allocation describe the architecture of an intelligence system. [2]

Text documents are necessary for the description of additional specifications, which cannot be displayed on models. To such specifications concern: a requirement specification, the schedule of development, the description of main technical solutions, etc. For each working stage in the Unified process the set of templates of text documents, which should be developed, is defined. Templates set structure of the document and define its contents. Examples of such templates are:

- The requirement specification – defines main requirements to the system, stages and periods of development;
- The schedule of carrying out of development - contains the list of operations and periods of their execution
- The dictionary of data domain - represents any information, which can be demanded for understanding of models and text documents and is used for definition of the terminology specific to problem area;
- Descriptions of variants of usage - are contained with the detailed description of technological processes in which the intelligence system is used and is the additional document to model of variants of usage;
- The architectural document - provides thorough survey of the architecture of the system, contains set of architectural representations for the description of various aspects of the system.

The unified engineering process is supported several tool CASE by resources, leaders among which are Rational Rose and Together.

Definition of requirements to an intelligence system

Requirements are descriptions of necessary or desirable properties of an intelligence system. Definition of requirements pursues two purposes: to define

essential requirements and to present them in the form convenient for users, customers and developers. " Essential requirements " are understood as those requirements which implementation will bring to users appreciable and significant result for them. Under " representation in the form convenient for users, customers and developers " that users and customers should understand the total description of requirements is understood mainly. It is one of main results of the working process of definition of requirements. Very important it is correct to formulate requirements to identify possible risks and to avoid misunderstanding between developers, customers and users after delivery of the ready product.

During the working process of definition of the requirement of users as requirements are identified. The functional requirements are described by the variants of usage, which are included in model of variants of usage. Other requirements can join those variants of usage to which they concern, to be saved in the separate list or to be fixed any otherwise. [1]

The main purpose of a stage of definition of requirements is a formal description of requirements to a developed intelligence system. The main tasks - definition of users of the system, model building of variants of usage and definition of additional (nonfunctional) requirements to the system.

The intelligence system has set of categories of users. Each category of users is represented by a separate *actant.* Actants are not necessarily people. Actants can be other systems or the external equipment cooperating with the system. Actants during execution of variants of usage communicate with the system, dispatching it of the message and receiving answer messages. Having determined, that actants make, and that - variants of usage, precise differentiation of duties between actants and an intelligence system is possible to receive. This differentiation helps to define sphere of application of the system better. [2]

5. OBJECT-ORIENTED ANALYSIS AND FORMALIZATION OF THE REQUIREMENTS TO THE DEVELOPED SYSTEM

The first stage of execution of the master thesis was the stage of the object-oriented analysis and formalizing of the requirement to the developed system. The purpose of the given stage was the formal description of the functional requirements to ***a system kernel of visualization*** of the interface forms of client applications.

5.1 Theoretical background

At a stage of definition of requirements of diagnostic inspection it is necessary to determine all actants, considering as its users and what external systems will work with an intelligence system should cooperate with it. Each category of users or external systems is represented by one actant. [1]

Variants of usage form that users, working with the system, could satisfy the requirements. The variant of usage defines sequence of operations (including its various variants), which the system should fulfill, bringing appreciable and measurable result, significant for some actant. The initial information for selection of variants of usage is the list of its functions and sentences of users as they want to use the system for execution of the operation. Each function or a method of usage of the system, appreciable result bringing to the user, - candidates in variants of usage. These variants of usage during inspection can be specified and divided into more individual variants of usage or switch on in more common variants of usage.

At definition of variants of usage of the function of the system can be broken on three groups:

- Obvious - their execution is obvious to the user;
- Hidden - should be fulfilled imperceptibly for the user; it concerns, for example, many base technical functions, such as saving, transmission or information processing;
- Additional - optional functions, which addition will not result in essential rise in price of development and will not affect execution of other functions.

Obvious functions are considered first of all and will derivate variants of usage of a top level. The variants of usage obtained thus will be detailed, that results in appearance of the hidden functions and variants of usage appropriate to them. Additional functions are considered in the last queue. [2]

Standard error at construction of model of variants of usage is representation as variants of usage of separate technological steps, operations or transactions. The variant of usage is the description concerning the big completed process. The process from the beginning up to the end describes sequence of events, operations and the transactions required for reaching any result or representation some given organizations or performers. [3]

There are some practical methods of selection of variants of usage. One of them consists in the analysis of actants: external users of the system are defined and for each user processes, which it initiates or in which participates, are determined. Other method is based on the analysis of events: external events to which the system and each event should react contacts the user are allocated.

At the analysis of complex variants of usage it can be demanded to develop the additional models defining a context of the system. There are, at least, two approaches to the description of a context of the system: simulation of data domain and business - simulation. The model of data domain describes the important concepts of a context as objects of data domain. Identification and the name of these objects help to develop the dictionary of terms. Subsequently allocated objects can become a basis for construction of conceptual model of the system. The purpose of business - simulation will consist in the description of technological processes to understand them. The business - model defines, what business processes the system should support. The architect of the system and the head of development during carrying out of inspection accept solution on necessity of model building of data domain or business - simulation at a stage of definition of requirements. At choice of a way of the description of a context of the system it is necessary to take into account, that business - simulation more complex process, demands the big temporary expenses and manpower resources. [2]

The model of variants of usage describes the functional requirements to the system. In addition to variants of usage at carrying out of inspection common principles of construction of user interface for implementation of this or that variant of usage should be defined. One of ways to develop the specification of user interface is to make

some versions of representations of the information, which it is necessary to handle, discuss thumbnails with users, to make prototypes and to present their users for approbation.

Except for the functional requirements described by model of variants of usage, and requirements to user interface there are nonfunctional (additional) requirements to an intelligence system. Such properties of the system concern to nonfunctional requirements, as limitations of the environment of operation and implementation, productivity, dependence on the platform, ease of support, expandability and reliability. Reliability are understood as such characteristics, as fitness, accuracy, an average time between failures, number of errors, etc. Requirements on productivity impose specific conditions on execution of functions: speed, capacity, the response time, and used resources. The majority of the requirements connected to productivity, concern only to some variants of usage and should be assigned to them. Practically it means, that these requirements should be described in a correct context that is inside variants of usage.
So that effectively to define requirements to an intelligence system the set of methods and artefacts is used. All artefacts make as a whole a set of requirements. Traditional requirement specifications are represented by a set of artefacts: model of variants of usage, additional requirements and text specifications.

As requirements can vary, the way of their upgrade, permitting to inspect brought changes is necessary. It is made by means of iterations. Each iteration mirrors some change of a set of requirements. The number of brought changes with each iteration step-by-step should decrease, and requirements to be stabilized. [2]

5.2 Practical results

The task of the given stage was definition of functions, which should be fulfilled by the system of visualization. Result of execution of a stage is the model of variants of usage of the nucleus, resulted on fig.

As a result of the fulfilled analysis of data domain the following main functional tasks have been detected:

- Loading the description of the form and its mapping;
- Interaction with the user;
- Interaction with the server;

Detailed implementation of the primary goals will be considered in the chapter devoted to an object-oriented design of the client application. In this chapter we shall be limited to consideration of subtasks, which include main variants of usage.

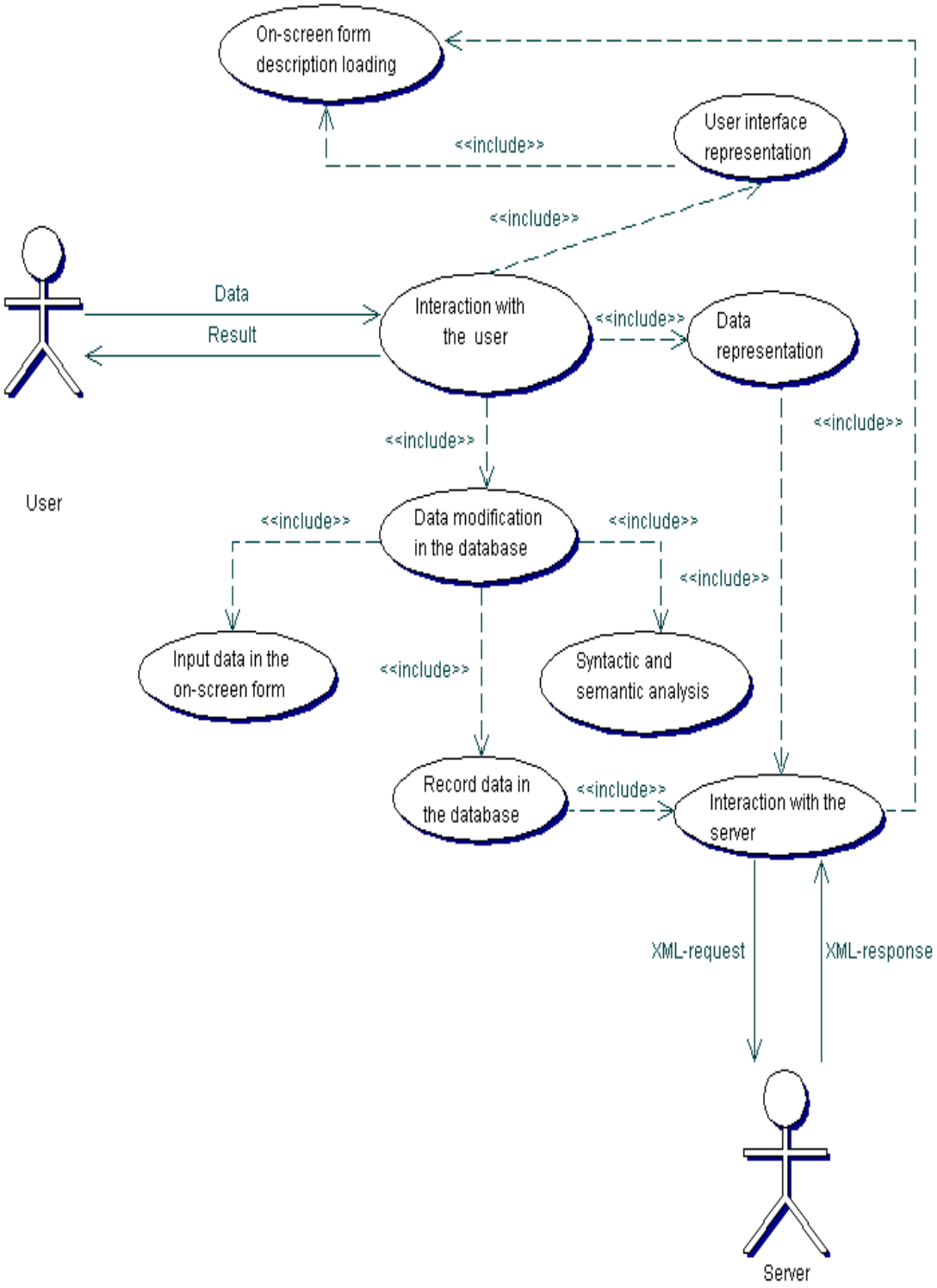

Figure 5. The diagram of the use cases of the client application

Interaction with the user it is carried out through the user interface realized on the basis of markup language XML and programming language Java, possibility of choice of user interface also is given the user depending on type of the selected application. The given variant of usage includes:

- **Representation of the user interface** (the menu of the client application, screen forms and its components) which includes **loading the description of the form;**
- **Representation of the data** stored in a DB and requested by the user. The given variant of usage also includes **interaction with the server** which we shall consider below;
- **Change of the data in a DB** enables to remotely add new recordings, to edit already existing recordings, to delete the recordings stored in a DB.

Change of the data in a DB, in turn, will consist from:

- **Data entry in the screen form**, realizes input or change of the data through user interface;
- **The syntactical and semantic control** of the entered data realized on the side of "client" with the help of specially developed XML-structure, resulted in the application to degree operation. Such approach allows to balance load between a server and client part and, thus, to receive the defined benefits in productivity of the system;
- **Data record in a DB** which is carried out after pressing by the user of the appropriate button the screen form realizing the mechanism of interaction with the server. The command on data record is carried out with the help of the SQL-searches dispatched on the server as specially developed XML-structure, the given task includes the task **of interaction with the server.**

Interaction with the server is one of the most important tasks at development of the application. Interaction is carried out on the protocol selected at the discretion of the developer, and includes **loading the description of the form.**

6. OBJECT-ORIENTED DESIGN: developing of the XML-structure of the document and the structure of the "core"

The purpose of the given stage was designing the scheme of the XML-document for the description of screen forms and construction of a program frame of a kernel.

6.1 Theoretical background

At stages of definition of requirements and the analysis the main attention is given learning of requirements, concepts and the operations connected to the system. After creation of the artefacts describing the requirements and conceptual model of the system, it is possible to pass to a design stage. At this stage logical solution on an object-oriented paradigm is developed. Main artefacts of a design stage are the model of designing and model of allocation. [5]

The model of designing forms with usage of conceptual model as input data. It adapts for the selected environment of execution, libraries for construction of user interface and-or a DBMS. It can be adapted also for a reuse before the created hardware-software components.

Similarly to the model created at an analysis stage, the model of designing also defines concepts (classes, subsystems and interfaces), ratios between them and interactions which realize variants of usage. However the units defined in model of designing, are " design clones " more conceptual units defined in model of the analysis in the sense that new units (project) are adapted for the environment of execution, and the previous units (analysis) - are not present. In other words, model of the project more "physical" by the nature, and model of the analysis more "conceptual".

Overall objective of a design stage - development of the architecture of the future intelligence system. Thus it is necessary to construct such architecture which will allow to realize variants of usage as now, and in the future and to make it is cost effective.

The architecture depends on what variants of usage the system should support. On early iterations some variants of usage, which will allow developing the architecture in the best way, are selected. These important for creation of the architecture variants of usage will include what are necessary for users at the moment of system development and, probably, in its subsequent releases.[2]

However the architecture will be influenced not only important for the architecture with variants of usage, but also other factors:

- The system software, which is used for construction of the system (for example, operational system or a DBMS);
- The software of an average level;
- The inherited products, which will enter a new intelligence system;
- Standards and corporate rules, which are necessary for observing;
- Common nonfunctional requirements (that is usages not adhered to concrete variant);
- Requirements to the allocation, defining as the system is arranged on computers, for example, according to the architecture the client / server.

In engineering process of the architecture there are design models, the documents describing the architecture, and models of allocation. For the big intelligence system consisting of hundreds or thousand of classes of the analysis, it is impossible to realize variants of usage, using only classes of designing. Such system will be too great that in it was possible to do without classification of the maximum order. Classes are grouped in subsystems. The subsystem is an intelligent classification of classes or other subsystems. The subsystem has a set of interfaces, which provide its existence and usage. These interfaces define a context of a subsystem (actants both other subsystems and classes).

The model of designing is an object model, which describes physical implementation of use cases. In this model the main attention is given how the functional and nonfunctional requirements together with other limitations concerning the environment of implementation, make an intelligence system. Besides the model of designing represents abstraction of implementation of the system and is used as input data for implementation. [2]

Subsystems of designing and the classes of designing making model of designing, represent abstraction of subsystems and components of implementation of the system. These abstractions are trivial and represent simple mapping the project on implementation.

In model of designing variants of usage are realized as classes of designing and their objects. This implementation is represented as cooperation and is named as the project of implementation of variant of usage. The class of designing is the most

approximate to a reality abstraction of the class or similar implementation of the system. The language used for the description of classes of designing, the same, as the programming language for implementation. Ratioes with other classes in which the given class of designing participates, frequently receive obvious expression at implementation of this class. Methods (that is implementations of operations) the classes of designing are directly displayed on appropriate methods of classes of implementation (that is the code). The class of designing can shift processing of some requirements of the subsequent implementation, having transferred it requirements to implementation of the class. In result begins possible to postpone decision making, which cannot be made on the basis of model of designing, for example, concerning questions of coding of an intelligence system. The class of designing is frequently asked by a stereotype, which directly is displayed in a construction of the appropriate programming language. The class of designing can be active - objects of the class will use an own string of handle, working in a parallel way with other active objects.

The project of implementation of variants of usage is the cooperation, which is included in model of designing which defines implementation of concrete variant of usage in terms of classes of designing and their objects. The project of implementation of variants of usage contains the text description of stream of events, the diagrams of classes displaying classes participating in it of designing, and the diagram of interactions displaying stream of events of variant of usage in terms of interactions between objects of designing. If in it there is a necessity, diagrams also can display subsystems and the interfaces, which are included in implementation of variant of usage.

It is frequently difficult to read diagrams of model of designing without auxiliary resources. As such resource the project of events were the text description, explaining and complementary diagrams and denotations on them can appear. The text should be composed in terms of the objects cooperating for realization of variant of usage, or in terms of subsystems participating in this interaction. [1]

Other text documents of model of designing are requirements to implementation. In them various requirements, for example nonfunctional, to implementation of variant of usage gather. Requirements included in these descriptions are found out on a design stage, some from them could be defined during the previous working processes, but implementations of variants of usage have changed them.

The model of designing is used for representation of the architecture of the system and contains such document, as " the Description of architecturally significant artefacts ". Architecturally significant the following artefacts of model of designing usually are considered:

- Decomposition of model of designing on subsystems, interfaces and dependences between them. This decomposition is very important for the architecture in general as subsystems and their interfaces will derivate common structure of the system.
- Key classes of designing, what are developed from significant classes of the analysis, mirror the common concept of designing and have set of links with other classes of designing.
- Projects of implementation of variant usages describing a certain important and critical functionality which are required to be developed in the beginning of life cycle of a new intelligence system. [2]

Except for model of designing for the description of the architecture of intelligence system the model of allocation is used. The model of allocation is an object model, which defines physical allocation of the system that is how functionality is distributed on computing sites. Basic elements of model of allocation are sites and links between them. Each site represents a computing resource, usually the processor, the server, an automated workplace or other device. Sites have the links representing channels of information interchange, for example, the Internet, Intranet, the internal bus, etc. The Model of allocation describes some various network configurations, including the configurations for testing and simulations. Functionality (or the process), fulfilled on a site, is defined by the components loaded on a site. The model of allocation describes mapping the architecture of programs on the architecture of an intelligence system (equipment). [2]

Thus, main result of a design stage is the model of designing, which saving the structure of the system defined at an analysis stage, is the drawing for the subsequent implementation. The model of designing includes the following units:

- Subsystems of designing, service subsystems, and also their dependences, interfaces and contents;
- Classes of designing, including active classes and their operations, attributes, ratioes and requirements in implementation;

- Projects of implementation of variants usages describing designing of variants of usage in terms of cooperations inside model of designing;
- Architectural representation of model of designing, including architecturally significant units.

Result of designing also is the model of allocation defining all configurations of networks on which the system can be fulfilled.
Models of designing and allocation are the basic data for the subsequent activity on implementation and testing

6.2 Practical results

At the given stage the following tasks were solved:

- Development DTD of the scheme of the XML-document;
- Detailing of the use cases detected at an analysis stage and construction of model of interaction of components of the kernel;
- Designing structure of the kernel and relations between its components.

Results of operation at the given stage are:

- XML - schemes of the document realized in DTD format;
- Model of interaction as diagrams of sequences for the primary goals of the uses cases detected at an analysis stage;
- Preliminary model of main classes of the kernel;

6.2.1. Developing of the XML-document structure

In this chapter the result of operation of a development cycle of the XML-structures realizing graphics components and event handlers of screen forms represented.

In sort of that all developed structures, are described in DTD format, main denotations and agreements, accepted for this format, described below.
In XML-documents, DTD defines a set of the elements, identifies elements, which can be in other elements, and defines the attributes for each of them. [17]
The element is defined with the help of a descriptor **!ELEMENT** in which the name of a element and structure of its contents is pointed. Lists of attributes of the element are defined with the help of the keyword **!ATTLIST**. Inside it names of attributes, types of their values and additional parameters are set. [13]

In total there are six possible types of attribute values: [19]

- CDATA - contents of the document can be any character data;
- ID - defines the unique identifier of a unit in the document;
- IDREF (IDREFS) - specifies, that value of attribute the name (or some such names divided by blanks in the second case) of the unique identifier of a unit defined in this document should appear;
- ENTITY (ENTITIES) - value of attribute should be the name (or the list of names if it is used ENTITIES) the component (macro definitions), defined in the document;
- NMTOKEN (NMTOKENS) - contents of a unit can be only one separate word (i.e. this parameter is limited variant CDATA);
- The list of valid values - is defined the list of values which the given attribute can have;

Also in definition of attribute it is possible to use the following parameters:

- #REQUIRED – defines mandatory attribute which should be set in all units of the given type;
- #IMPLIED - the attribute is not mandatory;

- #FIXED "value" - specifies, that the attribute should have only an indicated value, however definition of attribute is not mandatory, but during analysis its value in any case will be transferred to the program - analyzer;
- Value - sets value of attribute by default;

XML-structures represented graphical components of the user interface.

JLabel (display area for a short text string or an image, or both. A label does not react to input events) [10]:

```
<!ELEMENT jlabel (#PCDATA | border)>
<!ATTLIST   jlabel
            id          CDATA                   #REQUIRED
            bounds      CDATA                   #REQUIRED
            icon        CDATA                   #IMPLIED
            alignment   (left | center | right)  “left”>
```

Attribute bounds='x1;y1;x2;y2', represents as coordinates of rectangle.

JPanel (display area for a short text string or an image, or both. A label does not react to input events) [10]:

```
<!ELEMENT jpanel (#PCDATA | border)>
<!ATTLIST   jpanel
            id          CDATA                   #REQUIRED
            bounds      CDATA                   #REQUIRED>
```

JTable (is a user-interface component that presents data in a two-dimensional table format) [10]:

```
<!ELEMENT jtable  (columns)>
<!ATTLIST   jtable
            id          CDATA                       #REQUIRED
```

```
                model       CDATA                          #REQUIRED>


<!ELEMENT columns  (column+)>
<!ATTLIST   columns>

<!ELEMENT column (#PCDATA | border) >
<!ATTLIST   column
                type        (string | image | lock_unlock | plus_minus )  “string”
                width       CDATA                          # IMPLIED
                font        CDATA                          # IMPLIED >
```

JTextField (is a lightweight component that allows the editing of a single line of text) [10]:

```
<!ELEMENT jtextfield (#PCDATA | border)>
<!ATTLIST   jtextfield
                id          CDATA                  #REQUIRED
                bounds      CDATA                  #REQUIRED
                model       CDATA                  #IMPLIED>
```

JButton (An implementation of a "push" button) [10]:

```
<!ELEMENT jbutton (#PCDATA | border)>
<!ATTLIST   jbutton
                id          CDATA                  #REQUIRED
                bounds      CDATA                  #REQUIRED
                model       CDATA                  #IMPLIED
                font        CDATA                  #IMPLIED
                forecolor   CDATA                  #IMPLIED
                icon        CDATA                  #IMPLIED>
```

The attribute forecolor is represented in the RGB-format, for example,
red color: (255; 0; 0).
The attribute font is represented as (the name of the font; type style; a font size), for example (verdana; bold; 1)
The attribute icon is intended for association of the component with a graphics image.

JCheckBox (An implementation of a check box -- an item that can be selected or deselected, and which displays its state to the user) [10]:

```
<!ELEMENT jcheckbox (#PCDATA | border)>
<!ATTLIST   jcheckbox
            id          CDATA           #REQUIRED
            bounds      CDATA           #REQUIRED
            model       CDATA           #IMPLIED
            icon        CDATA           #IMPLIED
            selected    (true | false)  “false”>
```

JRadioButton (An implementation of a radio button -- an item that can be selected or deselected, and which displays its state to the user) [10]:

```
<!ELEMENT jradiobutton (#PCDATA | border)>
<!ATTLIST   jradiobutton
            id          CDATA           #REQUIRED
            bounds      CDATA           #REQUIRED
            model       CDATA           #IMPLIED
            icon        CDATA           #IMPLIED
            selected    (true | false)  “false”>
```

JMenuItem (An implementation of an item in a menu. A menu item is essentially a button sitting in a list. When the user selects the "button", the action associated with the menu item is performed) [10]:

```
<!ELEMENT jmenuitem (#PCDATA | border)>
<!ATTLIST   jmenuitem
```

```
                id          CDATA                   #REQUIRED
                bounds      CDATA                   #REQUIRED
                model       CDATA                   #IMPLIED
                icon        CDATA                   #IMPLIED>
```

JCheckBoxMenuItem:

```
<!ELEMENT jcheckboxmenuitem (#PCDATA | border)>
<!ATTLIST   jcheckboxmenuitem
                id          CDATA                   #REQUIRED
                bounds      CDATA                   #REQUIRED
                model       CDATA                   #IMPLIED
                icon        CDATA                   #IMPLIED
                checked     (true | false)          “false”>
```

JRadioButtonMenuItem:

```
<!ELEMENT jradiobuttonmenuitem (#PCDATA | border)>
<!ATTLIST   jradiobuttonmenuitem
                id          CDATA                   #REQUIRED
                bounds      CDATA                   #REQUIRED
                model       CDATA                   #IMPLIED
                icon        CDATA                   #IMPLIED
                checked     (true | false)          “false”>
```

JComboBox (A component that combines a button or text field and a drop-down list. The user can select a value from the drop-down list, which appears at the user's request) [10]:

```
<!ELEMENT jcombobox (#PCDATA | border)>
<!ATTLIST   jcombobox
```

```
                id              CDATA           #REQUIRED
                bounds          CDATA           #REQUIRED
                model           CDATA           #IMPLIED
                selectedindex   CDATA           #IMPLIED
                editable        (true | false)  “false”>
```

JList (A component that allows the user to select one or more objects from a list) [10]:

```
<!ELEMENT jlist (#PCDATA | border)>
<!ATTLIST   jlist
                id              CDATA           #REQUIRED
                bounds          CDATA           #REQUIRED
                model           CDATA           #IMPLIED
                selectedindex   CDATA           #IMPLIED
                editable        (true | false)          “false”>
```

XML-structures represented event handlers.

MouseEvent (An event which indicates that a mouse action occurred in a component) [14]:

```
<!ELEMENT mouseevent #PCDATA >
<!ATTLIST   mouseevent
                id          CDATA                   #REQUIRED
                object      CDATA                   #REQUIRED
                type    (mouse_clicked|mouse_double_clicked|
                        mouse_entered|mouse_exited|
                        mouse_pressed|mouse_released) mouse_clicked>
```

ComponentEvent (A low-level event which indicates that a component moved, changed size, or changed visibility) [14]:

```
<!ELEMENT componentevent #PCDATA >
<!ATTLIST   componentevent
            id          CDATA                   #REQUIRED
            object      CDATA                   #REQUIRED
            type    (component_hidden | component_moved |
                     component_resized | component_shown)
                                                component_hidden>
```

ContainerEvent (A low-level event which indicates that a container's contents changed because a component was added or removed) [14]:

```
<!ELEMENT containerevent #PCDATA >
<!ATTLIST   containerevent
            id          CDATA                   #REQUIRED
            object      CDATA                   #REQUIRED
            type    (component_added | component_moved)
                                                component_added >
```

FocusEvent (A low-level event which indicates that a Component has gained or lost the input focus) [14]:

```
<!ELEMENT focusevent #PCDATA >
<!ATTLIST   focusevent
            id          CDATA                   #REQUIRED
            object      CDATA                   #REQUIRED
            type    (focus_gained| focus_lost)  focus_gained>
```

WindowEvent (A low-level event that indicates that a window has changed its status) [14]:

```
<!ELEMENT windowevent #PCDATA >
<!ATTLIST   windowevent
```

```
id          CDATA                          #REQUIRED
object      CDATA                          #REQUIRED
type    (window_activated | window_closed |
         window_closing | window_deactivated |
          window_deiconified | window_iconified |
          window_opened)                    window_opened>
```

ItemEvent (A semantic event which indicates that an item was selected or deselected) [14]:

```
<!ELEMENT itemevent #PCDATA >
<!ATTLIST   itemevent
            id          CDATA                          #REQUIRED
            object      CDATA                          #REQUIRED
            type        (item_statechanged | selected | deselected)
                                                  item_statechanged>
```

6.2.2. Detailed elaboration

Interaction with the server is one of main mechanisms with which help effective and correct operation "client - server" of the application is realized. We shall consider it by the example of any operation (loading, deleting, addition, change) made by the user above the data.

After making by the user of the some defined action (for example, attempts to load the data in the table of the screen form from a DB), control is transferred to an event handler, which, in turn, requests the template of XML-request from the Form Manager. After obtaining required result, there is a substitution of the data in the template and the request to the Server Manager on execution is formed. The Server Manager makes some operations in order to pass the request to the server. There is an opening connection, preparation of HTTP-header, conversion of XML-request in a string (since in such form the request will be handled by the server), and sending the request to the server. After obtaining the response from the server, the Server Manager makes reconversion of the

string in the XML-format and parses the turned out structure. In case the analysis has given incorrect result, the error code sent to the handler, which parses it and as the final of the operation, returns to the user error message. Otherwise, the handler receives response in the XML format, and then generated the data and displays it to the user as result. MSC the diagram describing the given algorithm is shown in a Figure 8.

Data transfer between the client and the server is carried out on HTTP protocol (Hypertext Transfer Protocol), which is the protocol of an application layer and intended for allocation and handle of the intelligence systems realizing the mechanism of links [14]. It is the main object-oriented protocol, which can solve tasks of handle of exchange between servers and objects of the distributed systems, using their methods of searches. In modern conditions a basis the most popular method the structured query language (SQL) - the language which allows to state complex searches to databases in rather brief form is. The request referred on the server, as the XML-document, is handled by the special interpreter on the side of the server, the result of operation is handled on the side of the client, and displayed in the required format in screen forms. The XML-structure realizing requests on the server, is resulted below.
Access of the server - application to the database, is carried out with help JDBC of the interface. JDBC (Java Data Base Connectivity) is applied program Java interface for execution of SQL-requests [7]. The server-based application is realized on the basis of J2EE technology.

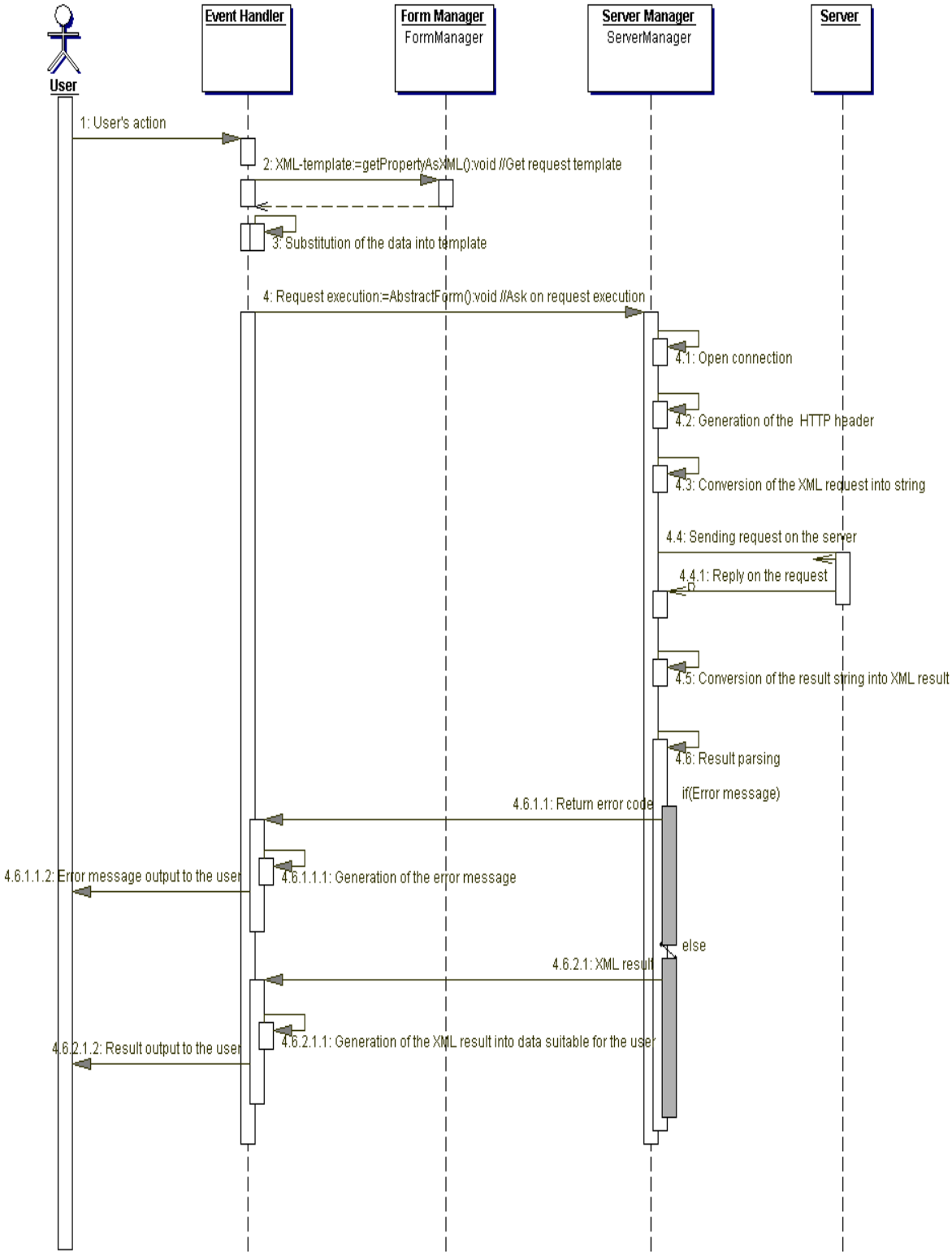

Figure 6. Sequence diagram of the interaction with the server

XML-structure of the realization of the SQL-request to the database, looks in the following way:

Request

Any client request should be made in a tag "request". Its attributes:

- type – the type of the client;
- user – the identifier of the client;
- password – the password;
- object – name of business process; possible values:
  - supply;
  - replication.
- operation – the operation in the given business process; possible values:
  - insert – addition of the object;
  - delete – deleting of the object;
  - list – select of the object (возвращаемое значение – список неструктурированных объектов);
  - update – modification of values;

```
<!ELEMENT request (item)>
<!ATTLIST   request
            type        CDATA       #REQUIRED
            user        CDATA       #REQUIRED
            password    CDATA       #REQUIRED
            object      (supply | replication) 'supply'
            operation   (insert | delete | list | select | update)
            from        CDATA       # IMPLIED
            to          CDATA       # IMPLIED>
```

Item

Each object, which participates in the operation, is represented logical or physical entities. They are represented by the nested tags "item" with attributes:

- name – entity name;
- relation – field name on which the derived unit is connected with parent (it is meaningful only for derived units;

```
<!ELEMENT item (item*, property*, parameter+)>
<!ATTLIST  item
           name              CDATA     #REQUIRED
           relation          CDATA     #IMPLIED
           parentrelation    CDATA     #IMPLIED
           currentrelation   CDATA     #IMPLIED>
```

Property

The list of resulting properties of request is underlined with the help of tags "property", where attribute:

- name – name of property;
- alias – alias;
- order – sets sorting on the given field.

```
<!ELEMENT property     EMPTY>
<!ATTLIST  property
           name        CDATA     #REQUIRED
           alias       CDATA     # IMPLIED
           order       CDATA     # IMPLIED>
```

Parameter

Conditions of selecting of the data are enumerated in a tag "parameter" with attributes:

- name – name of the request parameter;
- e (equal);
- l (larger);
- s (smaller);
- le (larger or equal);
- se (smaller or equal);
- ne (not equal);
- in – in an interval (value of the given attribute – values listed through a comma);
- nin – excepting an interval (value of the given attribute – values listed through a comma);
- like - similar .

```
<!ELEMENT parameter     EMPTY>
<!ATTLIST  parameter
              name          CDATA      #REQUIRED
              e             CDATA      #IMPLIED
              te            CDATA      #IMPLIED
              l             CDATA      #IMPLIED
              s             CDATA      #IMPLIED
              le            CDATA      #IMPLIED
              se            CDATA      #IMPLIED
              ne            CDATA      #IMPLIED
              in            CDATA      #IMPLIED
              nin           CDATA      #IMPLIED
              like          CDATA      #IMPLIED>
```

The result of operation of the server on request of the client is made in a tag <response> (attributes and their values of the given tag same as in a tag <request>) inside which are made in the same tags logical the essence, required in search. The

enclosure of tags сущностей illustrates ratioes between them. Attributes сущностей are represented as attributes of tags and their values.

Other, not less important mechanism, **loadings of the description of the form** we shall consider on a similar example - operation of the user (search about opening of the screen form).
The user presses the button of call of the form then there is a search to the manager of forms on show of the form. The manager of forms, in turn, makes the following operations:

- Requests the initial Java-code for the appropriate form of the handler of classes;
- Requests the XML-description of the form of the object - loader XML;

After obtaining a java of the code, the class appropriate to the given form is formed and the object which, in a consequence and will handle events for the new form if its loading will pass correctly is generated. After obtaining the XML-description of the form, the result is handled and in case outcome is correct, the form becomes accessible to the user, otherwise, the form is not initialized. The diagram of sequences of the given use cases, is represented in a figure 7.

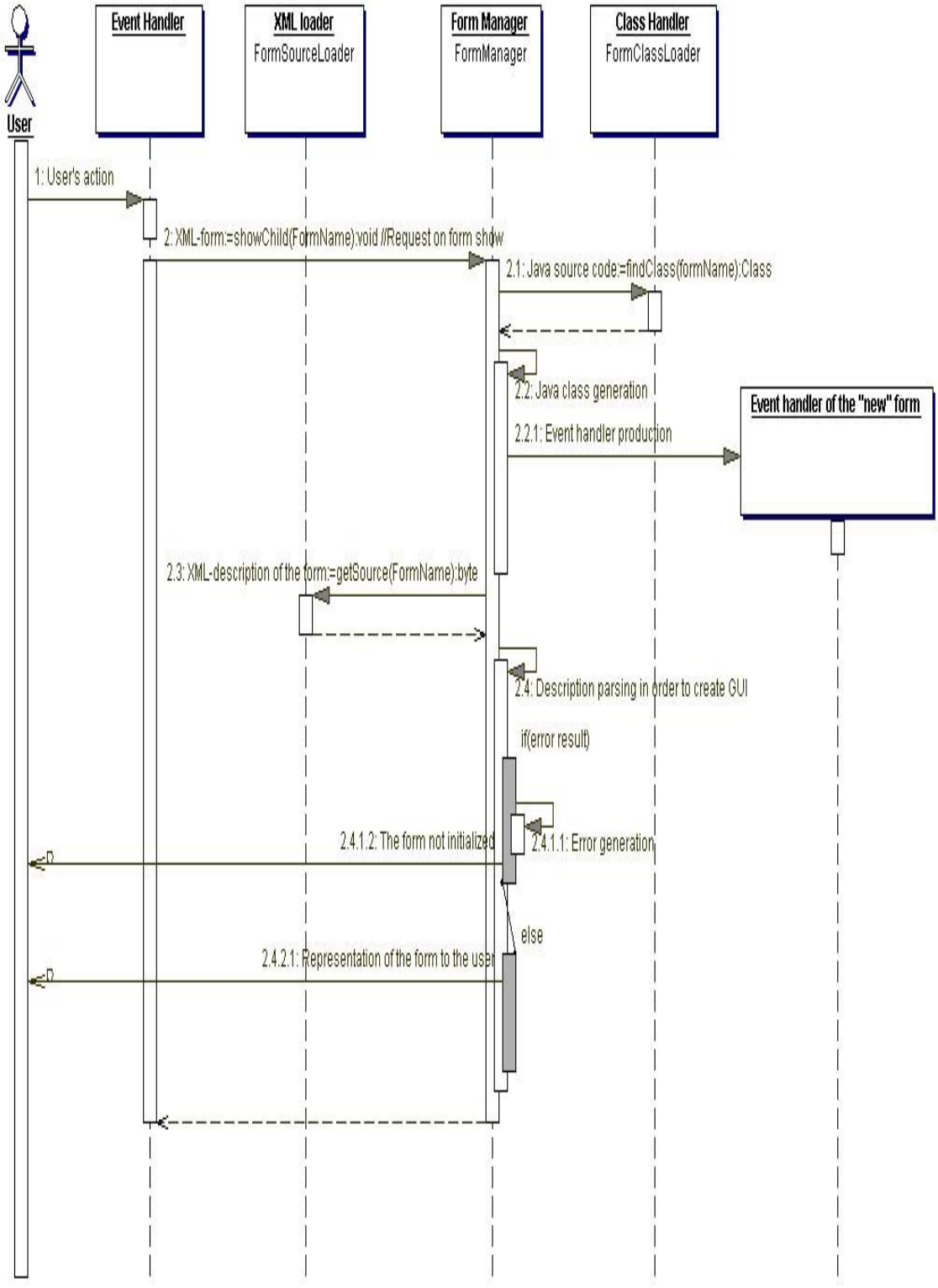

Figure 7. Sequence diagram of the loading of the screen form

6.2.3. Class diagram

The suggested variant of construction of the kernel, will consist of four main packages divided by a principle of functionalities contained in them.
The diagram of classes is resulted in a figure 8:

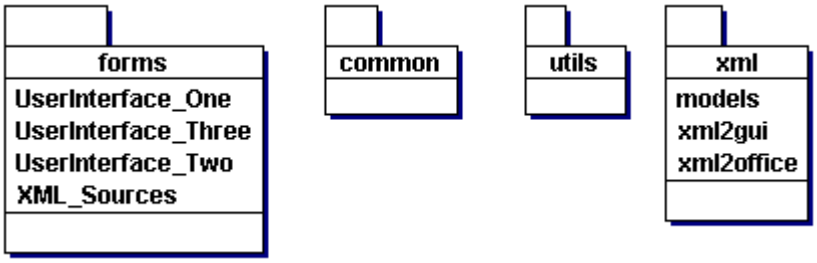

Figure 8. The diagram of the classes realizing a system kernel of visualization

Forms package comprises classes, which are responsible for functionality of each separate screen form. They include implementation of such operations as, addition, deleting, editing of the data into a DB, event processing on pressing of buttons forms. Besides the given package comprises the source files containing XML-documents, describing the screen form.

Common package, is the main package and comprises the classes responsible for loading of forms and the main application, operation with a server-based application, and also the class handling inquiry answer, coming with the server.

Utils package, contains classes permitting to realize special operations of operation with strings, the tool of processing and output of messages to the screen to the user, resources of processing of XML-documents, the classes realizing event processing, described by XML-structures, and also the class realizing operation with files (opening, saving, choice of the file on a mask).

Xml package, contains classes, convert XML-descriptions of forms in the graphics format, classes of operation with data models, and also the classes supporting operation

from MS of an Excel and MS of Word and intended for implementation of such possibilities of the client application, as printing of the data in the file.

For display of the information received as a response from a server, are developed xml models which allow to display its client, in convenient for viewing and work a kind. First we shall enter concepts, xmlmodel, and then we shall paint with the developed structures for each available model.

xmlmodel – the model being XML the document, containing in itself the data received as a result of request from the server.

Examples of the XML-structures realizing are below resulted
XmlTableModel:

```
<!ELEMENT xmltablemodel  (column+, row, id)>
<!ATTLIST   xmltablemodel
            id              CDATA           #REQUIRED
            parent          CDATA           #REQUIRED>

<!ELEMENT column (#PCDATA | border)>
<!ATTLIST   column
            path            CDATA           #REQUIRED
            type            CDATA           #REQUIRED
            width           CDATA           #IMPLIED
            editable        (true | false)  “false”>

<!ELEMENT row (#PCDATA | border)>
<!ATTLIST   row
            path            CDATA           #REQUIRED>

<!ELEMENT id (#PCDATA | border)>
<!ATTLIST   id
```

```
              name              CDATA          #REQUIRED>
```

XmlTextFieldModel:

```
<!ELEMENT xmltextfieldmodel (node)>
<!ATTLIST   xmltextfieldmodel
              id                CDATA          #REQUIRED
              parent            CDATA          #REQUIRED>

<!ELEMENT node (#PCDATA | border)>
<!ATTLIST   path
              path              CDATA          #REQUIRED>
```

The XML-structure realizing validation of the entered data

```
<!ELEMENT xml     (validator) >
<!ATTLIST   xml
              id          CDATA                    #REQUIRED>

<!ELEMENT validator  (item) >

<!ELEMENT item   (check)>
<!ATTLIST   item
              object      CDATA                    #REQUIRED
              description CDATA                    #REQUIRED>

<!ELEMENT check   EMPTY>
<!ATTLIST   check
```

```
type (string | float | integer | date)   CDATA   # IMPLIED
length                                   CDATA   # IMPLIED
equals                                   CDATA   # IMPLIED
notnull (true|false)                     CDATA   # IMPLIED
compare                                  CDATA   # IMPLIED
value                                    CDATA   # IMPLIED>
```

The XML-structure realizing a printing of the documents

```
<!ELEMENT bookmarks (document) >
<!ATTLIST   bookmarks
            id          CDATA           #REQUIRED>

<!ELEMENT document (text+, table+)>
<!ATTLIST   document
            id          CDATA           #REQUIRED
            template    CDATA           #REQUIRED>

<!ELEMENT text  #PCDATA>
<!ATTLIST   text
            id          CDATA           #REQUIRED
            model       CDATA           #REQUIRED
            path        CDATA           #REQUIRED>

<!ELEMENT table #PCDATA>
<!ATTLIST   table
            id          CDATA           #REQUIRED
            model       CDATA           #REQUIRED>
```

7. REALIZATION OF THE CLIENT APPLICATION

The purpose of the given stage was definition of structure of the code with the subsequent implementation of classes and objects of the client application.

The description of structure of the code, conditional its division into packages, have been resulted in the previous chapter. In the given part of degree operation, we shall result diagrams of main and most important classes of the application, and also we shall make the brief browse of main methods of these classes.

In a figure 9 the diagram of classes **of common package** is resulted. At a sight of the developer which main objects are FormManager class and ServerManger which methods are resulted in figures 10 and 11.

The primary goal of FormManager class, consists in initialization and loading of screen forms of the client application, the method showForm () is main.

ServerManager class organizes interaction between the client and the server, the method post() is main. Its task is opening connection, creation of HTTP-header, conversion of request in string and also reconversion.

In a figure 14 the diagram of classes **of xml package** intended for conversion of XML-structures in graphics sort is resulted. SwingComponentProducer classes, intended for creation of graphics components under the XML-description of forms and EventProducer class forming event handlers on each components are main. Other classes realize auxiliary methods and functions which are necessary for correct operation with graphics objects.

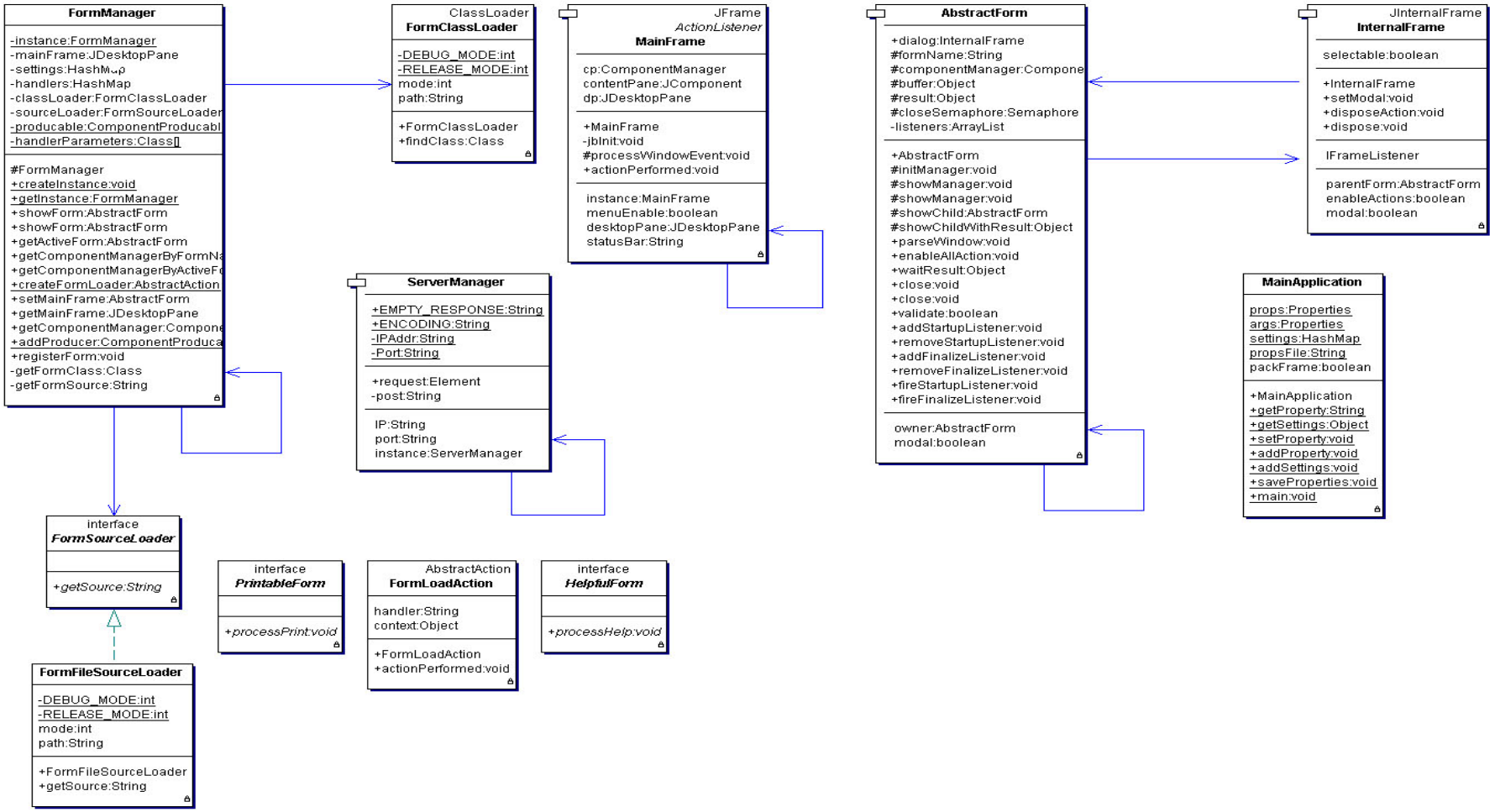

Figure 9. The diagram **of common classes** of the kernel of the client application

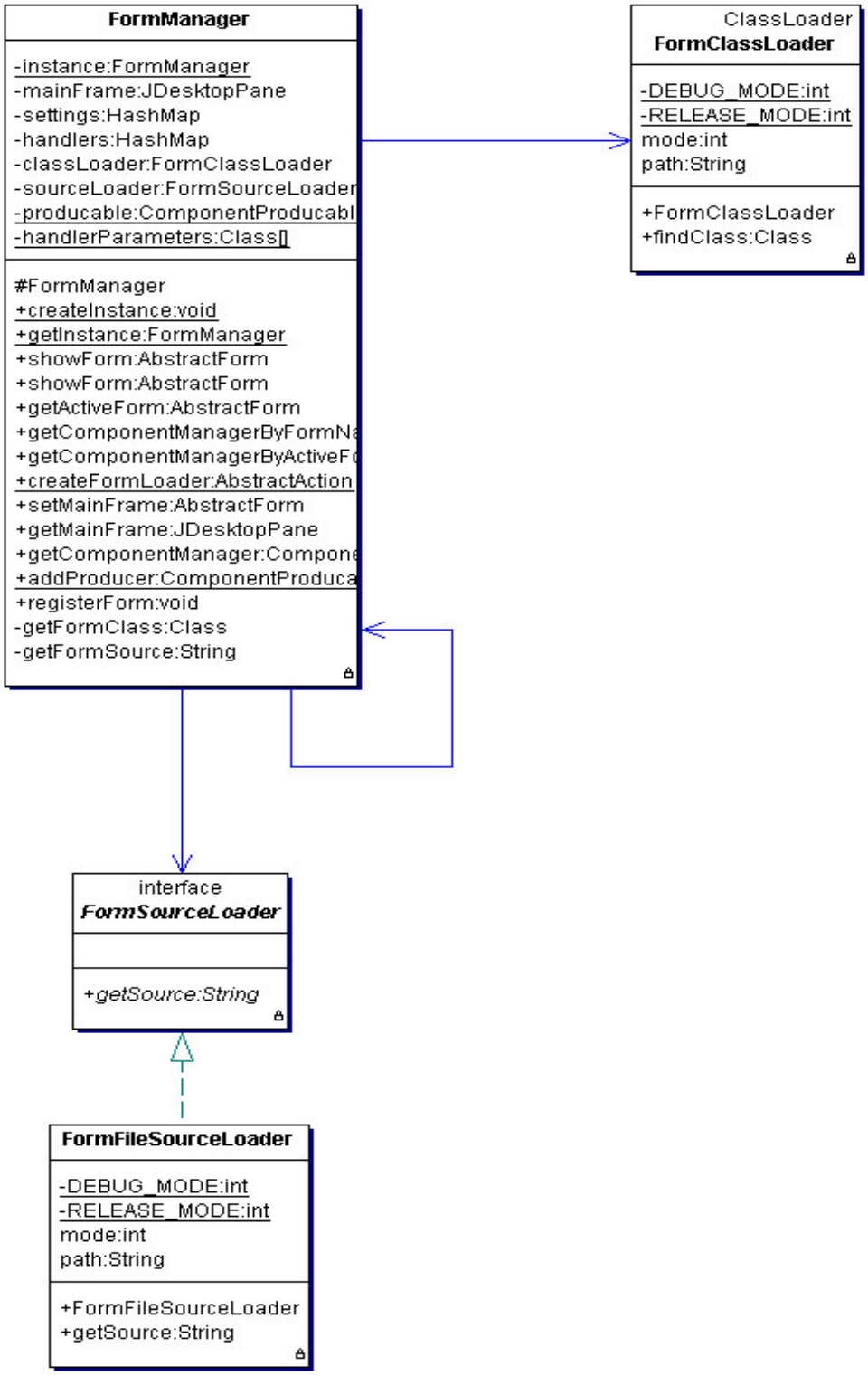

Figure 10. Detailed diagram of the class FormManger

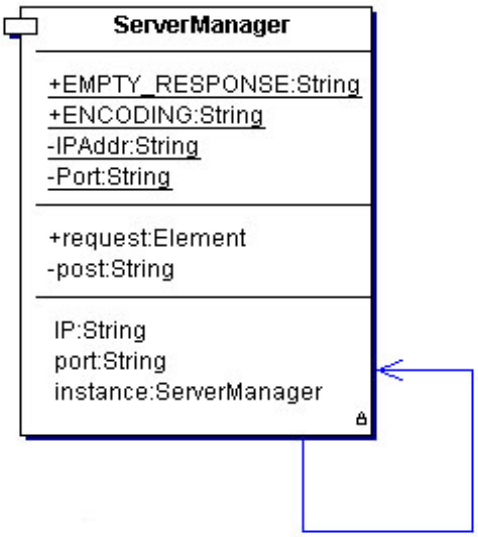

Figure 11. Detailed diagram of the class ServerManager

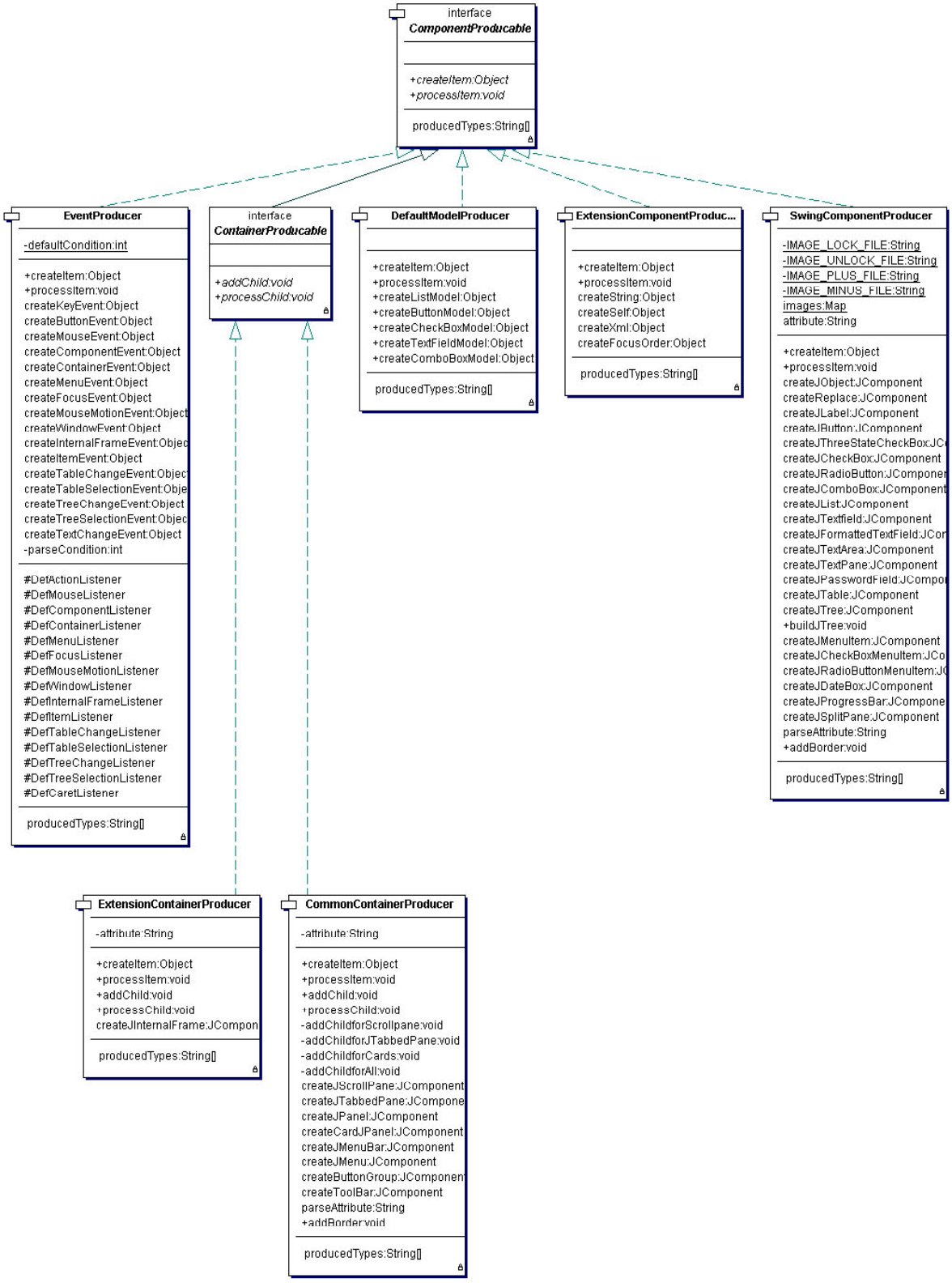

Figure 12. The diagram of the classes realizing conversion XML – structures in the graphical interface

8. TESTING

Approbation of the suggested technology of development of the client application has passed "Elections" by the example of the application "Service" of the system. System "Elections" has been created for automation of the information processes which are carried out during preparation and elections and referenda.

The general architecture of the developed system is shown in a figure:

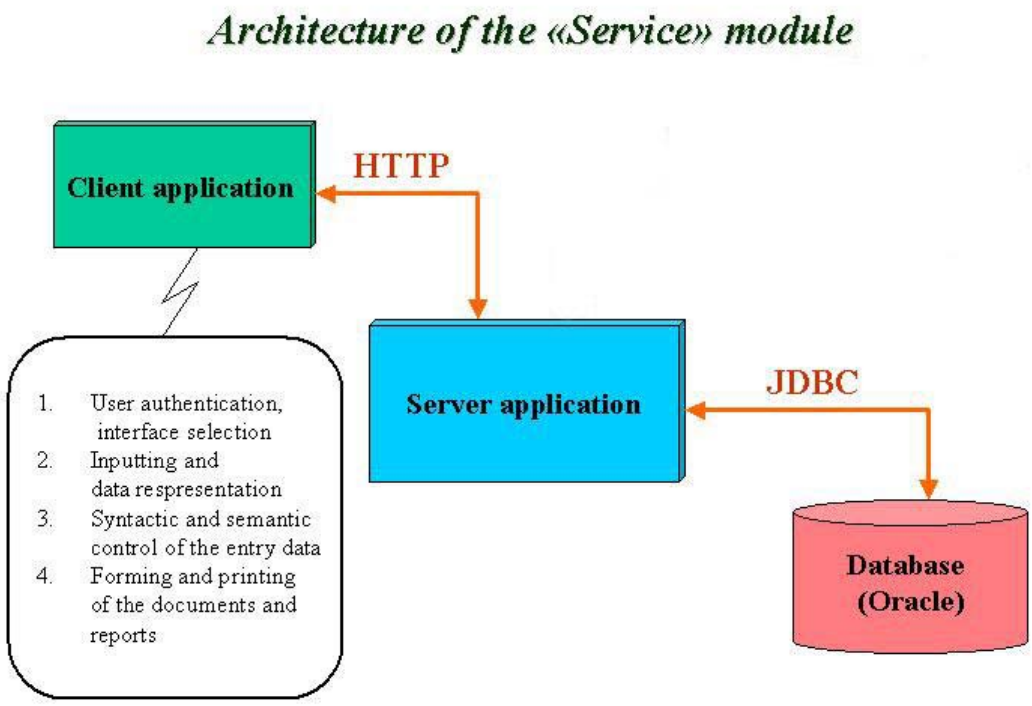

Figure 13. The architecture of the system "Service"

The client application is intended for the remote operation with the database containing about 100000 recordings about a state of program - means, accessories and other information. In a figure 14, the general view the main menu of the client application is shown. At choice of this or that choice the screen form containing in the list of the data opens. Depending on the given rights, the user can add, delete the data, open the form in detail or to carry out search of recording through a filter. In a figure 15, the form as the list and the form representing details of selected recording are shown. Time of initialization of the list of the data in the screen form, depends on amount of the data

contained in base according to search sent on the server, and also from characteristics of the computers fulfilling roles of the client and the server.

Loading of graphics components of screen forms occurs instantly, and response time of the system on search of the user does not exceed 3-5 seconds, as the computer - client the computer with the processor of an Intel of a Pentium 4, 1500 MHz, 512 Mb RAM was used, as the computer - server a computer with the processor of an Intel of a Pentium 4, 1500 MHz, 1 Gb RAM.

As the primary goals of the suggested system coincide with tasks of the unit "Service" with the help of the offered technology of designing it was possible to save time of development.

The developed client application has been successfully inserted in territory of the Russian Federation and on the basis of the suggested technology the new intelligence system is developed.

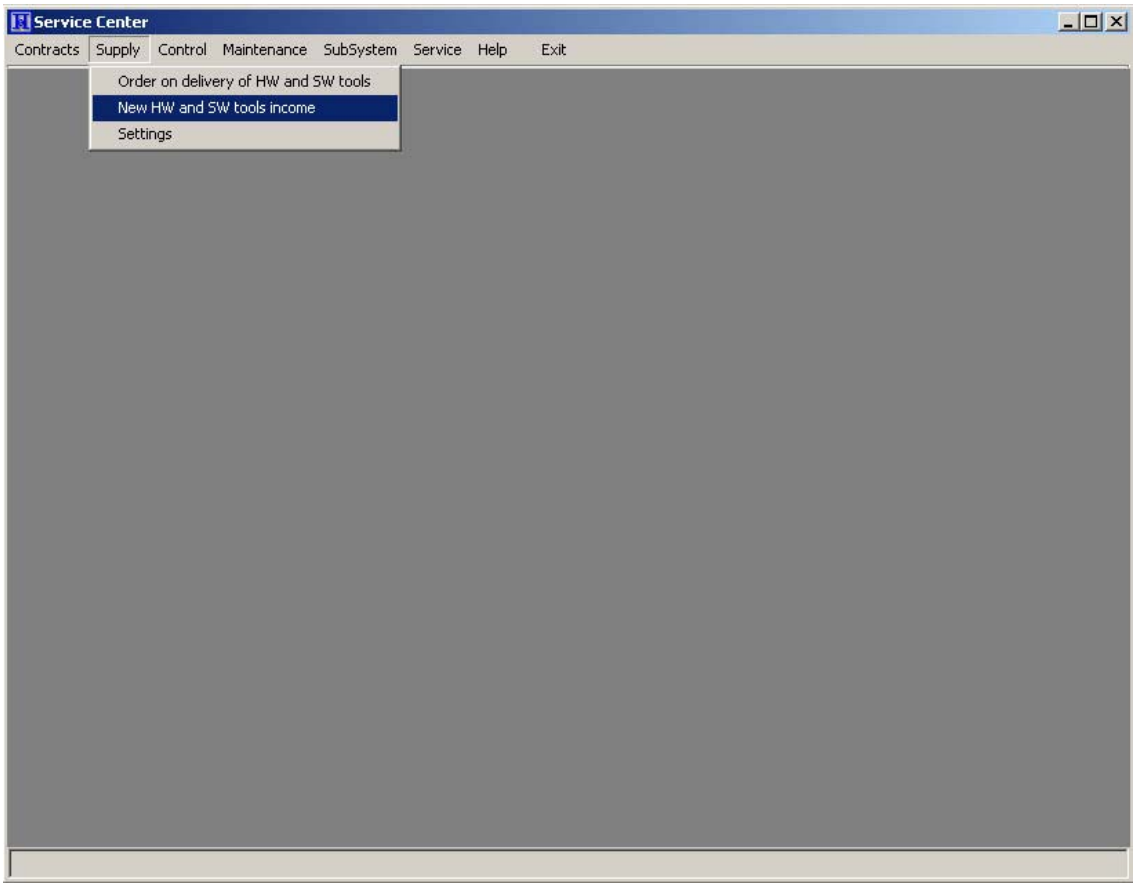

Figure 14. The main menu of the client application

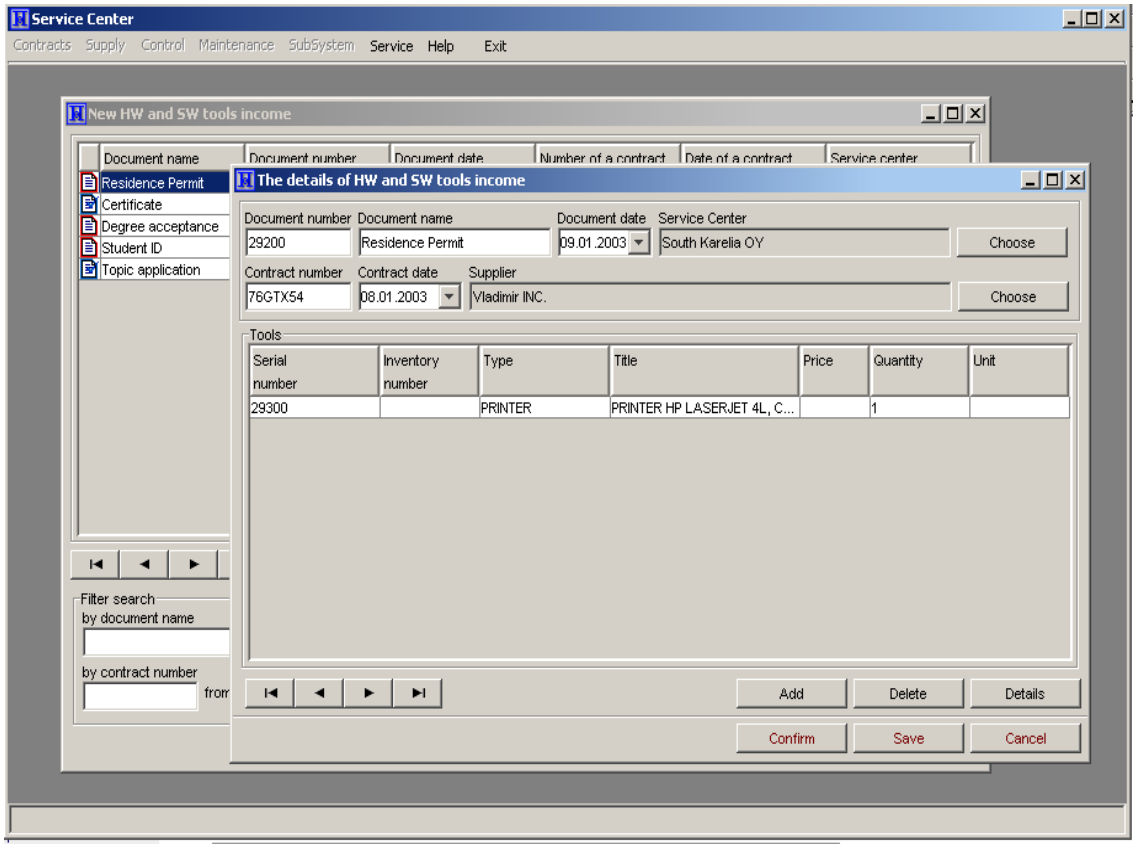

Figure 15. Screen forms of the client application

9. CONCLUSIONS

Nowadays, there are a lot of tools for development of client-server applications, which adequately satisfy the requirements, which customer showed to the software product. However, for example, such resources do not respond possibilities of flexible operations with a graphic user interface. The main idea of the master thesis was to offer such technology, which met requirements of flexible work with customization of the client-server application. In this work the approach of using XML-technology is offered.

The technology of usage of XML-documents suggested in the master thesis for the description of external representation and filling of screen forms provides possibility of fast development and easy customization of the client-server applications. The offered technology allows changing external representation of the interface forms of client applications without change and recompilations of the initial code of the program. Except for it the suggested technology is universal for implementation of client-server applications and does not depend on the language of implementation of the functional components. Properties of expandability of the XML-document allows to enter the new units describing the components specific to everyone application of user interface and does not demand essential processing of the initial code of a kernel of the client application. Using this technology we can consider XML-documents as the universal format for information interchange between separate components of the big application.

The experimental part of the thesis contains universal XML-structures, which describe screen forms of the user interface and also templates of the XML-requests to the database. Usage of these templates gave us representation of requests results to the database in some universal format in such a manner that the part of a DBMS becomes "transparent" for the application.

Results of this master thesis have been approved in the real client-server application and can be used in development of similar systems; except for it usage of the obtained results will allow to reduce considerably time for such creation of systems.

The analysis of results of operations shows that the tasks set before the author, are fulfilled and can be used in the future.

REFERENCES

[1]. Kruchen P., *The Rational Unified Process. An Introduction*, Williams Publishing, 2002. ISBN 5-8459-0239-8

[2]. Jacobson I., Booch G., Rumbaugh J., *The Unified Software Development Process*, Piter, 2002

[3]. Booch G., Rumbaugh J., Jacobson I., *The unified modeling language user guide,* DMK Press, 2001. ISBN 5-94074-144-4.

[4]. Pressman R., *Software Engineering: A Practitioner's Approach,* Fifth edition,

[5]. Larman C., *Applying UML and Patterns. An Introduction to Object-Oriented Analysis and Design.* Williams Publishing, 2001. ISBN 5-8459-0125-1

[6]. Tools for developing of the client-server applications URL: http://www.osp.ru/dbms/1996/05/52.htm (Accessed 28.04.2003)

[7]. Introduction to JDBC. URL: http://dmivic.chat.ru/JDBC/intro.doc.html#1005262 (Accessed 17.04.2003)

[8]. Client-server description. URL: http://www.osp.ru/os/1994/03/1.htm

[9]. Designing and development of the information systems. URL: http://www.citforum.ru/cfin/prcorpsys/index.shtml (Accessed 20.04.2003)

[10]. Swing components tutorial. URL: http:// java.sun.com (Accessed 29.04.2003)

[11]. Leonenkov A., *UML Tutorial,* BHV, 2002. ISBN 5-94157-008-2

[12]. Boggs U., Boggs M. *UML and Rational Rose,* LORI, 2000

[13]. Erik T. Ray, *Learning XML,* SYMBOL, 2001

[14]. Naughton P., Schildt H., *Java 2: The Complete Reference,* BHV, 2000

[15]. XML news and information, URL: http://www.xml.com (Accessed 9.04.2003)

[16]. McLaughlin B., *Java and XML,* SYMBOL, 2001

[17]. World Wide Web Consortium, URL: http://www.w3.org (Accessed 30.04.2003)

[18]. Rational Rose Products, URL: http://www.rational .com (Accessed 11.03.2003)

[19]. Introduction to XML, URL: http://www.sql.ru/articles (Accessed 18.03.2003)

[20]. Gamma E., Helm R., Johnson R., Vlissides J., *Design Patterns,* Piter, 2001

[21]. The basics of the client-server architecture, URL: http://pcmag.ru/?ID=35099

[22]. Harold E.R., Means S., *XML in a Nutshell,* SYMBOL, 2001

[23]. XML resources, URL: http://www.xml.org (Accessed 30.04.2003)