LATVIJAS UNIVERSITĀTE

ĢIRTS KARNĪTIS

INFORMĀCIJAS SISTĒMU INTEGRĀCIJAS PROBLĒMAS

SAISTĪTĀS PUBLIKĀCIJAS

(promocijas darba pielikums)

Rīga - 2004

Publikāciju saraksts

J.Bicevskis, G.Karnitis. **Problems in the Integration of Registers of State Significance in Latvia**, Baltic IT Review, No.1, 1998, page75

G. Arnicans, J. Bicevskis, G. Karnitis. **The Concept of Setting Up a Communications Server**, in Abstracts of Papers of 3rd International Conference "Information Technologies and Telecommunications in the Baltic States" , pages 48-57, Riga, 1999

G. Arnicans, J.Bicevskis, G. Karnitis. **The Unified Megasystem of Latvian Registers: Development of a Communication Server – the First Results and Conclusion**, in Abstracts of Papers of 4th International Conference "Information Technologies and Telecommunications in the Baltic States" , pages 163-168, Riga, 2000

G. Arnicans, G. Karnitis **Heterogeneous Database Browsing in WWW Based on Meta Model of Data Sources**, in Proceedings of the 4th IEEE International Baltic Workshop, edited by Albertas Čaplinskas, Vol. 1, pages 175-187Vilnius, Lithuania, 2000

G. Arnicans, G. Karnitis. **Heterogeneous Database Browsing in WWW Based on Meta Model of Data Sources**, in Selected Papers of 4th IEEE Baltic Workshop on Databases and Information Systems, pages 167-178, Kluwer Academic Publishers, 2001

G. Arnicans, J. Bicevskis, G. Karnitis, E. Karnitis. **Smart Integrated Mega-System as a Basis for E-governance**. Proceedings D of the 5th International Multi-Conference Information Society IS'2002, pages 197-202, Ljubljana, 2002

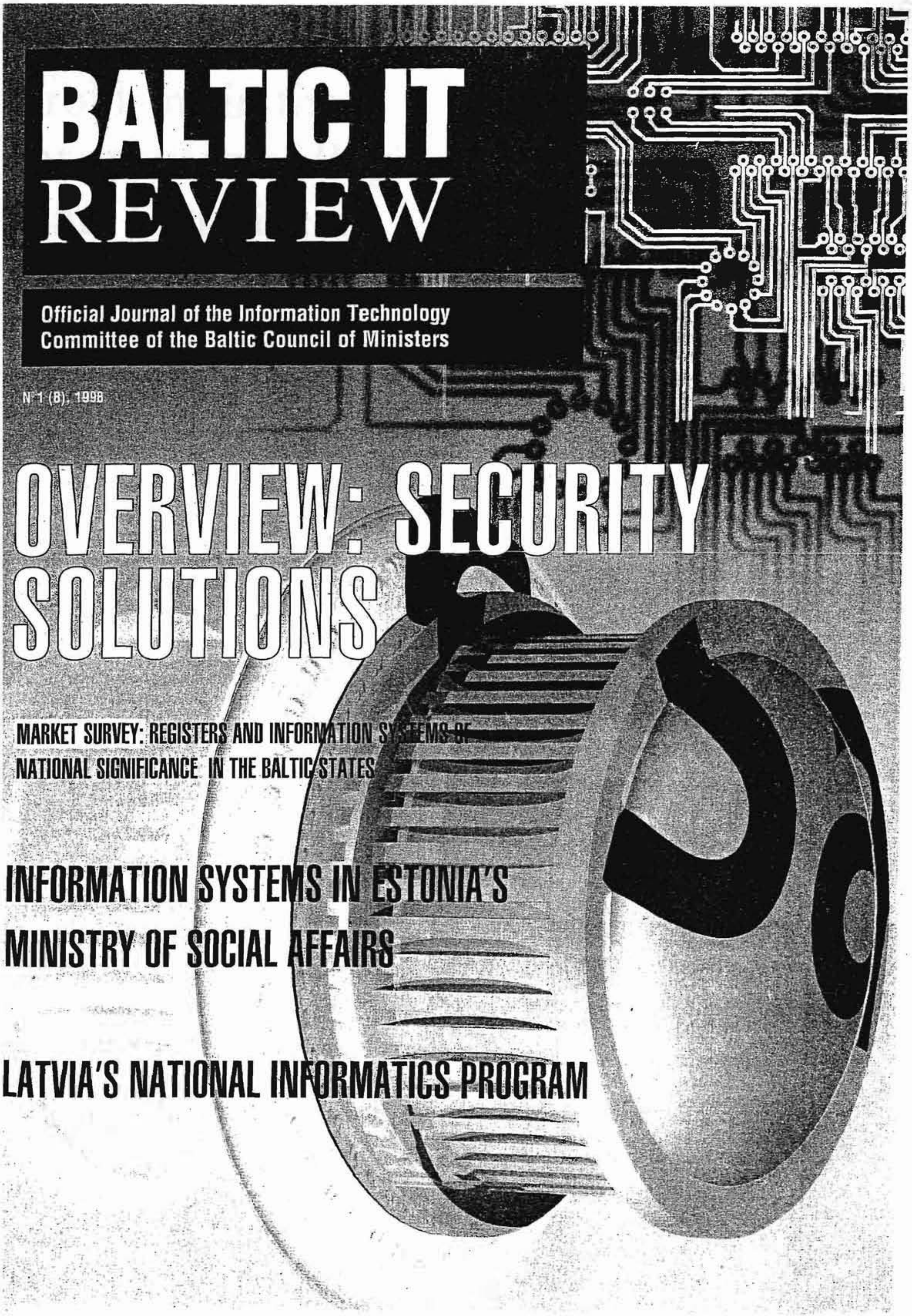
G. Arnicans, G.Karnitis. **Semantics for Managing Systems in Heterogeneous and Distibuted Environment**. H-M. Haav, A. Kalja (Eds), Databases and Information Systems II, Selected Papers from the Fifth International Baltic Conference, BalticDB&IS'2002, pages 149-160, Kluwer Academic Publishers, 2002

G. Arnicans, G.Karnitis. **Semantics for Managing Systems in Heterogeneous and Distibuted Environment**. In Hele-Mai Haav and Ahto Kalja, editors, Databases and Information Systems, Proceedings of the Fifth International Baltic Conference BalticDB&IS 2002, Vol1., pages 51-62, Tallinn, 2002

BALTIC IT REVIEW

Official Journal of the Information Technology Committee of the Baltic Council of Ministers

N°1 (8), 1998



OVERVIEW: SECURITY SOLUTIONS

MARKET SURVEY: REGISTERS AND INFORMATION SYSTEMS OF NATIONAL SIGNIFICANCE IN THE BALTIC STATES

INFORMATION SYSTEMS IN ESTONIA'S MINISTRY OF SOCIAL AFFAIRS

LATVIA'S NATIONAL INFORMATICS PROGRAM

Problems in the Integration of Registers of State Significance in Latvia

Jānis Bičevskis
Head of Department, Faculty of Physics and Mathematics, University of Latvia

Ģirts Karnītis
Student, Faculty of Physics and Mathematics, University of Latvia

Information systems and legal activities which record the most important objects under the authority of the state are called registers of state importance. A megasystem is the consolidation of various registers of state importance, and the megasystem operates on the basis of legal acts which govern individual registers and communications among them. Requirements for the megasystem and its individual registers define data structures, data exchange formats, information accessibility, data quality, security and other aspects. This article analyzes problems which have arisen in the creation and organization of the megasystem.

CHARACTERIZATION OF THE PRESENT SYSTEM

Latvia, like the other Baltic states, has es-ablished registers which provide for the ecording of the more important objects which re governed by the state – natural persons the Population Register), legal persons (the Enterprise Register), real estate (the Real Estate Register), movables (the Movable Property Register), and payment of taxes (the axpayers Register). The condition of these ery important state information systems, acording to a conclusion by an expert commison in July 1997, is not good. There is legal isorder, a failure of the various institutions to ooperate, insufficient veracity of data, and or documentation in the systems.

There is an increasing demand for credile information about the main objects of the ate, and that has led to the current reorgazation of the situation. An Informatics epartment has been established under the spices of the Ministry of Transportation, d among its duties is management of the velopment of the computer sector in atvia. A first draft of a national informatics gram has been elaborated, umbrella leation for the sector has been drafted, and ncrete proposals have been made concern- the integration of individual registers in- megasystem.

E PLAN FOR ESTABLISHING THE GASYSTEM

In order to establish the megasystem, the sting system must first be investigated – the data which the information systems store, the structure of the data, the source of the data, demand for the data, the speed at which data are processed, and the procedures for entering, controlling and processing data. The existing condition must be analyzed with the goal of finding an optimal distribution for the data among the various information systems, of improving the data model with new objects and attributes, of formulating those data processing procedures which are currently lacking, and of restructuring the way in which data are controlled so as to improve their veracity. The result of the analysis must be a proposal concerning all necessary changes in legislation, as well as changes in the existing information systems. Implementation of the proposal will be the next step, and the source of financing might be investments.

After the establishment of the first part of the megasystem, during the course of which the operations of the primary information systems would be brought into order, the consolidation of other information systems around the megasystem could be continued as a second phase of work.

THE CONSTITUTION OF THE MEGASYSTEM

The megasystem consists of three parts:

- A section on laws, standards and recommendations;
- The individual information systems of state importance and the data therein;
- An informational section, or a register of registers.

In the section on laws, standards and recommendations, there must be a statement of the requirements which the state makes vis-a-vis its information systems of state importance:

- The content and structure of information stored in registers;
- The distribution of information among registers;
- Procedures and formats for exchange of information among registers;
- Availability of information to end users;
- Guarantees of data quality and control procedures;
- Recommended specification languages, tools, operating systems and data base management systems;
- Information security requirements and other requirements.

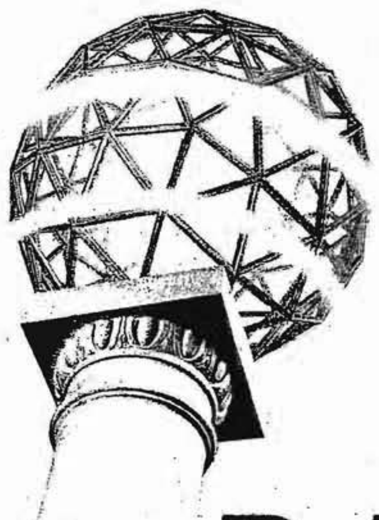
The megasystem must consist of various primary information systems of state importance and the data in them. Existing information systems must be reorganized to a greater or lesser extent in the context of establishing the megasystem. Also needed is a universal search mechanism which would allow any interested party to find the necessary information from all registers, irrespective of the register in which the information is stored. Of course, this process would be governed by access rights and the confidentiality level of the information that is being sought.

Information in the register of registers would concern the various registers (general information, data models, information sources, data processing procedures, etc.), as well as the objects in the registers (natural and legal persons, real estate, movables, etc.). The register of registers would contain a mechanisms for analysis of the use of objects in the various information systems.

Conclusions

In establishing and utilizing the megasystem, the state would obtain many advantages:

- The right of individuals to information would be implemented (an "information kiosk");
- Laws about the functioning of each information system would be put in place;
- Data sources and responsibility for the veracity of data would be specified;
- Duplication of information collection and entry among the registers would be averted;
- Data quality would be improved;
- Documentation of registers would be improved. ❑



Baltic
IT&T
'99

3rd INTERNATIONAL CONFERENCE "INFORMATION TECHNOLOGIES AND TELECOMMUNICATIONS IN THE BALTIC STATES"

Riga, April 28 - 30, 1999, Riga Congress Palace

Information Technology Committee of the Baltic Council of Ministers
DT Media Group

Abstracts of papers from the Baltic IT&T '99 Conference

The Concept of Setting Up a Communications Server

Mr. Guntis Arnicāns, Dr. Jānis Bičevskis, Mr. Ģirts Karnītis, Faculty of Physics and Mathematics, University of Latvia

A communications server is a set of software and computer equipment that allows a wide range of users, both domestically and internationally, to receive information from a variety of sources (government registers, data bases, information systems) through a single contact point. A communications server identifies users, authorizes the use of the respective data, fulfills a request that involves several information sources, and evaluates the cost of the process so that the appropriate financial transaction can be made. A communications server allows users to learn where information is being stored and what kind of information it is, as well as to request and to receive information from various registers without any need for in-depth knowledge about the technical aspects of its storage. The need to establish a communications server became evident when the governments of the Baltic States were setting up their joint data transmission network. In order to allow institutions in one country to obtain information about objects registered in another (enterprises, persons, motor vehicles, etc.), it is useful to receive the necessary data from a single information source, without having to study the data base structures of the other country. The use of the communications server, as has been seen through the elaboration of an integrated state significance information systems project, is also of significance within one country, because it provides a universal resource for information exchange among various information systems.

A communications server is a set of software and computer equipment that allows a wide range of users (both in Latvia and in other countries) to receive information from a variety of sources (government registers, data bases, information systems) through a single contact point. A communications server identifies users, authorizes the use of the respective data, fulfills a request that involves several information sources, and evaluates the cost of the process so that the appropriate financial transaction can be made. A communications server allows users to learn where information is being stored and what kind of information it is, as well as to request and to receive information from various registers without any need for n-depth knowledge about the technical aspects of its storage.

The need to establish a communications server became apparent when the governments of the Baltic States were setting up their joint data transmission network. In order to allow institutions in one country o obtain information about objects registered in another (enterprises, persons, motor vehicles, etc.), it is useful to receive the necessary data from a single information source, without having to study the data base structures of the other country. The use of the communications server, as has been seen through the elaboration of an integrated state significance information systems project, is also of significance within one country, because it provides a universal resource for information exchange among various information systems.

ROBLEM IDENTIFICATION

The need to establish a communications server was noted in the national program "Informatics" [1 and], as well as during the elaboration of two major projects – the Baltic States Government Data ransmission Network (hereafter in the text – the Network) [3 and 4] and the Integrated State Significance nformation System (hereafter – the Megasystem) [5]. The goal in establishing the network is to provide ındamental improvements in the exchange of telecommunications and data among the administrative institutions of the Baltic States. During the first phase of the project (1998 and 1999), universal solution is being set up to provide for the exchange of data among Latvia's Company Register, Motor Vehicles Register and Lost Motor Vehicles Register, as well as between these registers and the related international information structures. So far this has involved three concrete activities:

1) Accession of the Latvian Company Register to the European Business Register (EBR);

2) Cooperation between the Motor Vehicles Register and the related European-level structure EuCaris, as well as the establishment of a motor vehicles insurance system in Latvia (the so-called "green cards");

3) Improvements to the system whereby lost and stolen motor vehicles are registered in Latvia, including a connection to the international data bases of Interpol in this area.

During the second phase of this project, between 2000 and 2002, more work will be done to include Latvian registers into the Network and to integrate them into international information structures. In the second phase, the plan is to place the Population Register, the Lost Persons Register, the Lost Personal Documents Register, the Educational Documents Data Base, the Visas Data Base, the State Statistics Information System, the Consular Information System, the Health Care Information System and the Narcotics Information System on the Network.

In a situation where information from various sources is available on the Network, but users have no knowledge about the technical details of storing that information, there is an obvious need for a universal solution, and that is where the communications server comes in. The main requirement for a communications server is that it must allow users to formulate their information requests in a simple way and to receive responses to those requests without having to understand the technical aspects of the process. Users are not, after all, informatics specialists; they are employees of other administrative structures of the state, and there is no reason to think that they know anything about the way in which data objects are distributed among the registers of another country. We can expect both standardized and wholly unpredictable requests in this process. In terms of the urgency of requests, we can expect demands for on-line responses that require rapid response, as well as requests for off-line responses that can take hours or even days to fulfill. Needless to say, in setting up the communications system we must provide for all aspects of information confidentiality and user authorization.

The setting up of the communications system is important not only in the context of the Network, but also in the context of the Megasystem, which is a universal resource for the exchange of information among various information systems within a single country.

THE CONCEPT OF THE SOLUTION

The communications server, which is illustrated in Figure 1, is an Internet resource point. Users of the server can access it via various protocols – HTTP, CORBA, DCOM, SMTP (E-mail) and FTP. The server provides users with an opportunity to find out where information is stored and what kind of information is available, and then to request and receive information from various registers without studying their structure. Because users may have access to sensitive information, users are identified with certificates, and all data transmissions are coded.

Users who wish to have access to sensitive information before work with the system is begun must receive a certificate that corresponds to the X.509 standard. The certificate must issued for a specific period of time (usually one year) by a specialized institution (presumably in Latvia this would occur under the supervision of the Constitutional Defense Bureau). Certificates of this kind contain information that identifies the user, and they are virtually impossible to forge. The certificates are used to code data and to identify the user. Latvia's communications server will use a standard coding protocol such as SSL.

A user of the communications server sends information requests to it and receives responses from it. This can happen both on-line (HTTP, CORBA, DCOM) and off-line (HTTP, E-mail, FTP).

In the on-line regime, work with the communications server is based on the following structure: At the beginning of the process the user is identified. This means that the user sends his or her certificate to the communications server, which reviews it and specifies the user's rights. If the user does not have a certificate, then he or she can access the communications server as a guest and receive a limited amount of information from it. Next the user requests information. The communications server once again identifies the user and, on the basis of the level of the user's authorization, makes the appropriate requests to the data registers, sending the response to the user when it is received. The register receives not only the information request from the communications server, but also the user's certificate, which means that the register itself can identify the user and the user's level of authorization. The result of this is that the register provides only that information to the communications server for which the user is cleared.

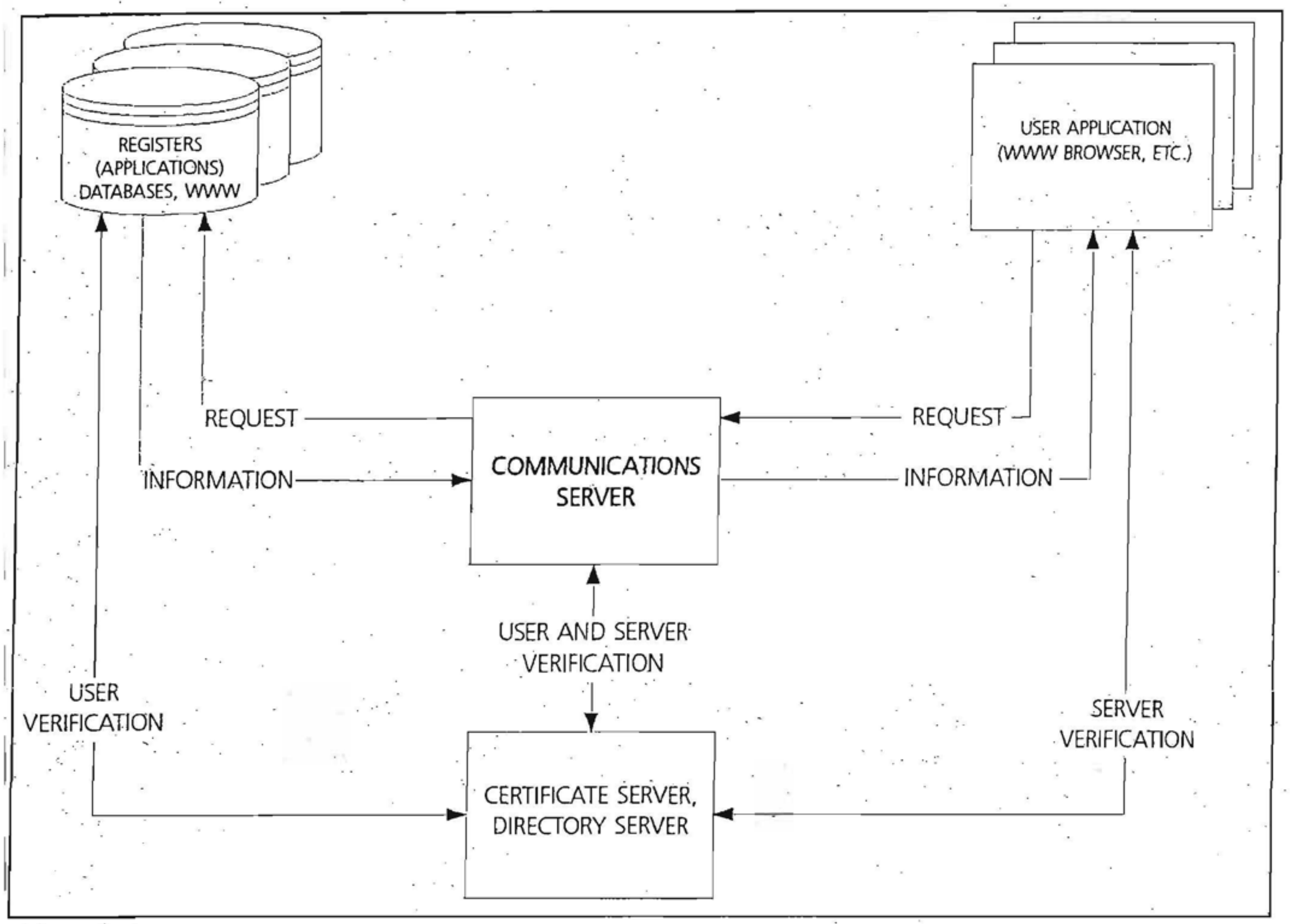

REGISTERS (APPLICATIONS) DATABASES, WWW
USER APPLICATION (WWW BROWSER, ETC.)
REQUEST
INFORMATION
COMMUNICATIONS SERVER
REQUEST
INFORMATION
USER AND SERVER VERIFICATION
USER VERIFICATION
SERVER VERIFICATION
CERTIFICATE SERVER, DIRECTORY SERVER

Figure 1. The operational structure of the communications server

In an off-line regime, the user requests information via HTTP, E-mail or FTP. During periods of time when it is less busy (usually at night), the communications server processes the request – identifies and verifies the user and then requests the respective information from the information registers. The response is sent to the user via E-mail, or it is stored until the user asks for it on-line.

The main advantage of an on-line regime in this process is that information can be obtained immediately when the need arises. This system can be used in cases when the speed at which a response is received is of importance, either from the point of view of the system (e.g., at border control facilities), or from the point of view of the operation (e.g., an application in which the registration number of an automobile is entered and information is received about the automobile from the Road Traffic Safety Department so that it need not be entered a second time).

The advantage of the off-line regime is that registers can even out the volume of work that is required, given that at night there should be relative few on-line requests for information. Off-line requests can also be sent in by users who have dial-up Internet connections, thus reducing costs. It is advisable to make off-line requests less expensive than on-line ones so that users are motivated to use the off-line system.

THE FUNCTIONS OF THE COMMUNICATIONS SERVER

We can specify five main functions for a communications server:

- User identification
- Authorization with respect to the use of information
- Management of user rights
- Fulfillment of requests that involve several information sources
- Evaluation of the costs of each request for billing purposes

USER IDENTIFICATION IN A COMMUNICATIONS SERVER

As was noted before, user identification involves X.509-standard certificates. In order to ensure that the certificate mechanism is operational, a communications system needs both a certificate server and a directory server. The former is a server that belongs to the certifying organization, generating and maintaining electronic certificates – both server certificates (issued to the server) and client certificates (issued to the user). The latter is a server in which the public keys of the certificates are stored, along with information about certificates that have been issued – when a certificate has been issued, to whom it was issued, and whether the certificate is valid or has been revoked.

The directory server is available to any interested party. For example, if a WWW server has been issued a certificate, any WWW user can ascertain that the server is secure. If a WWW client has been issued a certificate, in turn, the WWW server can ascertain that the client is authorized to work with the server. Both the client and the server can check the validity of the submitted certificates by looking them up in the directory server.

Work with certificates in WWW applications involves SSL (Secure Socket Layer) technologies, which are supported by most WWW servers, as well as the main WWW browsers – Netscape Navigator and Microsoft Internet Explorer. SSL technologies provide the following components of secure communications:

1) **WWW server approval:** A user can ascertain the fact that the WWW server is secure and that it can be entrusted with confidential information;

2) **The privacy of information:** The entire information flow between the client and the server is coded, using a unique session key. The session key is coded by the server with the client's public key in order to send the respective information to the client in a secure way. Each session key is used in only one session, which makes it difficult to decode the information without authorization. The information, in other words, cannot be viewed by unauthorized persons, even if it is intercepted on its way between the server and the client.

3) **The integrity of the information:** Both the server and the client calculate the control code on the basis of the content of the information, and if the information has been changed en route, the codes do not match. This means that the receiver of the information sees precisely the same information that was sent by the sender.

Secure data exchange between the WWW server and the client occurs in the following way when SSL technologies are used:

1) The client sends a request for data exchange to the WWW server;

2) The server in response sends its certificate to the client, asking for the client's certificate if appropriate;

3) The client checks the validity of the server certificate through the digital signature of the certificate server, sending the client's own certificate to the server if necessary;

4) When the authorization process is complete, the client sends the session key to the server, coding it with the public key of the server;

5) Both the server and the client know the session key, and further data flow between the server and the client during the respective session is coded with the session key.

The certificates of the server and the client are exchanged quickly and without any involvement by the user. The same is true with respect to an exchange of certificates among other applications.

When information is requested from the communications server (through the WWW or otherwise), the process occurs in the following way:

1) The user is identified through the aforementioned protocol, and the communications server checks the user in the directory server.

2) The communications server has a data base which records user rights, and the server uses this data base to specify the authorization level of the specific user. In carrying out the user's request, the communications server checks the user's rights in its own data base and, if the necessary level of authorization is there, then the request is sent along to the concrete register.

3) The register is also sent identification data about the user who has requested the information.

4) The software in the register checks the information in the directory server and authorizes the user.

5) According to the level of the user's authorization, either the request is carried out and the result is returned to the communications, server, or the communications server is told that the user does not have the right to carry out the request.

6) The communications server returns the result to the user.

A user can also request information from the register directly, without passing through the communications server. In that case the operational mechanism is similar:

1) When the information is requested from the register, the user must supply identifying information (a certificate).

2) The software in the register checks the information in the directory server and authorizes the user.

3) On the basis of the user's authorization and the level of his or her access rights, either the request is fulfilled and the result is sent back to the user, or the user is sent information saying that he or she does not have the right to receive the data.

This mechanism ensures that there is no need for the user to reintroduce identification each time a new request is made. In each session, the user is identified on the first occasion that a request is made with respect to a confidential data source, and in later requests the information is sent on to all of the respective information sources. Another advantage of the mechanism is that there is a centralized method for distributing user rights, as well as a unified policy with respect to this. It's also true that the user's rights do not change depending on the way in which he or she accesses the information – via the WWW, via a different application, or through some other method.

MANAGEMENT OF USER RIGHTS

The rights of users can be divided into several categories:

• The right to obtain information about what is stored in a concrete register – provided that the information is publicly available;

• The right to obtain information about one entry in one table in one register, based on the unique identifier of that particular entry;

• The right to obtain a list of data from one table in one register, selected on the basis of specific criteria;

• The right to obtain a list of data from several tables in a single register (whether the link exists or not);

• The right to obtain information from several tables in one register that are linked through a specific relation, the data being chosen on the basis of specific criteria;

• The right to obtain information about one object from several registers on the basis of the primary key of the object;

• The right to obtain information about the existence of a link among specific objects from various registers;

• The right to obtain a list of data that are selected on the basis of criteria entered by the user, the data coming from several tables in several registers that are mutually linked.

• The obtaining of information can be differentiated at four levels:

• A response as to whether the requested information has been found or has not been found;

• A response as to how many entries have been found;

• The primary keys of objects;

• The data that is being requested.

Each of these levels provides a different volume of information, and there are instances when the jump between proximate levels is quantitative, while in other instances it is qualitative. We could consider four different requests here:

"Does individual X own an automobile?"

"How many automobiles does individual X own?"

"What automobiles does individual X own?"

"Does individual X own automobile Y?"

The management of user rights is intentionally divided up so that it occurs in several places. The communications server has its own user management module, in which it stores information about the right of users to make various kinds of complex requests. Information about the right of a user to receive data from a specific register is stored either in the communications server or in a concrete register. The place where information about user rights is stored is harmonized between the communications server and the register. Because it is expected that before a register issues information, it will want to check the user's rights to use the information, then information about the user's rights with respect to a specific register will usually be stored in that register. From the perspective of centralized management, it would be better if information about user rights with respect to all registers were stored in the communications server. For various organizational reasons, unfortunately, this is either impossible on only partly possible. Information about user rights is stored both in the communications server and in the registers themselves.

The communications server is designed to work with both of these options, as well as with a combination of them, and the following scheme emerges:

- The communications server checks the right of the user to make a request in the first place, as well as the right of the user to seek out a link between objects in various registers;
- The communications server checks whether the user rights with respect to the concrete register are stored in the communications server or the register;
- If the rights are stored in the communications server, then it checks the rights before it sends the request to the register;
- If the information is stored in the register, then the register checks the user rights before it fulfills the request;
- If the rights are not stored in the register, then the register can, if necessary, receive information about the rights from the communications server in order to be able to check the rights of the respective individual to make the request.

Because it is possible for users to connect to the registers not only via the communications server, but also directly from an application, and because it should be true that in both instances the user has the same authorization to obtain information, then the check of whether a user has the right to obtain information from a specific register should occur not in the communications server, but in the register itself.

INFORMATION REQUESTS AND THE OBTAINING AND DEPICTION OF INFORMATION

The basic mission of the communications server is to provide users with access to various information sources so that they can obtain data from them. Let us take a look at the problems that arise in this process, devoting particular attention to the submission of requests and the obtaining of responses, and leaving aside the issue of user authorization, control over data access, registration of who has asked for information and what information has been requested, billing issues and such matters.

INFORMATION SOURCES

An information source or resource facility can be any information system or data base from any organization. There are administrative regulations concerning the organizations, information systems and data bases that are included in the communications server's network of services.

Over the course of time, the number of information sources can reach into the tens or even hundreds of sources. In Latvia alone there are already several dozen government registers, and their number may increase. Communications servers should also provide access to certain foreign information sources, as well as to the data bases of various other organizations in Latvia; these, too, could be included in the range of services provided by the communications server.

The communications server itself does not have an information sources. Each information source is primarily meant to carry out concrete and specific functions inside the respective organization Information systems and data bases that are used in an organization are chosen, designed and optimized specifically for the needs of the respective organization. They may not be aimed at providing information to other entities, but if such an opportunity is intended, then it can be very specific, and many limitations can be applied to it. This means that the communications server must adapt to the information sources, and not vice-versa. Of course some information sources can upgrade their information systems and optimize their data exchange procedures in order to meet the communications server's requirements.

Information sources that are part of the communications server's network can differ in terms of significance and volume. The more significant a data base, the better must be cooperation with it. The size of data bases must also be taken into account, because it has much to do with the respective data processing mechanisms.

Another key issue is the quality and stability of information sources. Information systems can involve a wide variety of technologies, and they are of varying ages. Depending on the resources that have been invested, some are of a higher quality and some – of a lower quality. Of course, it is easier to make contact with a high-quality information system and data base that have been designed with modern technologies than with systems that are old and of a lower quality level. A communications server must certainly be ready to deal with information sources that are unstable, that make errors and that in some instances are not even accessible.

Information systems can be designed with various systems, they may have various data bases, and their use may involve various operating systems and computer technologies. A communications server must be prepared to handle these problems, although this is no longer the worst possible difficulty, given that many different solutions are in existence.

Information can be stored in a wide variety of formats – that is the next issue. The most popular method for data storage is still relation data bases. Object-oriented data bases, static WEB pages and dynamic WEB pages that are generated from an internal format are becoming rapidly more influential. We must not, however, forget other information storage methods such as files of many different structures.

A concrete information unit and a logical group of information units can be doubled, stored in various formats, coded in various ways and stored in such a way that some of the information is kept secret. Information can be contradictory either within a single information system or among various information sources. This means that in the future the field of communications servers will have to involve various laws and data processing algorithms that are based on the technologies or artificial intelligence.

All of these aspects serve to demonstrate how serious is the issue of various information sources being highly varied. It should also be added that this heterogeneity exists among more than just information sources. The same situation can exist within a single register or a single organization.

It must also be remembered that each information source exists fairly independently. It can be updated, changed or liquidated, it can be created anew, its operations can be suspended for a while, or it can be withdrawn from cooperation with a communications server. This means that a communications server must exist in an environment that is not only highly varied, but also is extremely changeable.

USERS

For our purposes, we will say that a communications server user is any subject that wishes to obtain a service from the server.

Users are usually differentiated on the basis of their level of authorization to obtain specific information from specific information sources. These rights are regulated by law and by other normative acts, and they are managed by a specific user management bloc within the communications server.

From the perspective of the communications server, another very important user classification is based on a different aspect – the way in which the user requests information and the way in which the user receives a response. A communications server should be operated on the basis of the principle that it is there for the convenience of users, not vice-versa. This principle means that the server must be ready to receive information requests of a great many varieties and forms, and it must be ready, every time, to provide a response that is convenient for the user in terms of its type and form.

REQUESTS AND RESPONSES

A communications server must be ready to accept information requests that are stated in various ways and forms. The main operational regime for communications servers is an on-line connection, but this can involve a dedicated line to the communications server, dial-up access to the server, or a connection through informational networks (the Internet, the Latvian State Significance Data Transmission Network (VNDPT), or the networks of other national, global or organizational networks). We must also remember other ways to submit a request – E-mail, a request submitted on an electronic information carrier such as a diskette, a written request submitted on paper, or even an oral request.

Responses to various requests can be prepared in the same format as the original request. It should be added, however, that the user must have the right to select the method of response, irrespective of the way in which the request was submitted. Limitations on the ways in which requests and responses are formatted can be specified by administrative regulations, but in terms of technologies, a communications server must be prepared for all kinds of cooperation methods.

The forms of requests and responses can be highly varied. The most popular cooperation form is probably a WEB page, both for requests and for responses. This form of cooperation can be highly varied, and this is underpinned by existing WEB-type applications. The use of special procedures and functions may also be important when the procedure itself has parameters that specify the request and its result (i.e., the response to the desired request as specified by the parameters). Cooperation can also occur in the following forms:

1) Special applications that can work with the communications server;

2) Active objects that can work with the communications server and can be used in the client's applications;

3) Files with requests that are recorded in a specific format or response files in a specific format;

4) A group of files (including even data bases) for the requests and the responses;

5) Paper documents in an agreed format for requests and responses;

6) E-mail, which can be seen as a modification of items 3, 4 and 5 on this list.

It is commonly held that requests from a user can come in a dialogue regime from a human user and in an automated regime where the user is an application on the user's computer.

There must also be plans to work in a synchronous regime (request-wait-response) and in an asynchronous regime (request-processing over a specific period of time-report to the user about the availability of a result-response), because this ensures more efficient work for the user and the communications server alike, especially when it comes to processing large and complex requests.

In work with the user thought must also be given to such aspects as the various levels of preparedness among users, the language of communication, the respective text coding formats, the abilities of the user's computer equipment, operating systems and applications, and limitations in all of these things.

In other words, the main mission and, at the same time, the main problem that a communications server must handle is the way in which many different kinds of requests can be handled, submitting processed information from various information sources that sometimes are not compatible, and submitting a result to the user in the desired type and form.

INFORMATION ABOUT INFORMATION

As the number of information sources available through the communications server increases, an overabundance of information can quickly occur – one in which even the administrators of the communications server can get lost. It is necessary to classify all of the information sources and the information that is contained therein, keeping firmly in mind that information sources can change.

Communications servers must have data source repositories that contain formal descriptions of the sources, their properties, the data that are contained within them and the properties of the data. These repositories must be very flexible, it must be able to change them easily and quickly so that changes in the surrounding environment can be monitored. If there is to be a proper reaction to user requests, other parts of the communications system must be able to adapt to changes in the repository in a dynamic way.

The repository is not, however, meant only for internal use in the communications server. The user, too, must know where and what he can receive (of course, within the limitations of the user's authorization). This means that the communications server must also, so to speak, provide information about information. Using forms and terms that the user can understand, the server must describe the information that can be obtained and the ways in which it can be requested. There must also be efforts to link the various request formulation mechanisms as closely as possible to the repository, thus making easier the work of a user who takes advantage of the communications server's services only seldom.

Users often don't care where and how the desired information is stored. This means that the communications server must satisfy requests that concern information from many different sources. The repository, therefore, must also describe the links between the sources, as well as the ways in which various contradictions among the sources can be resolved, data be converted, etc. The repository must be an entity that makes it possible to consider all of the sources in a communications server to be one, big data base.

THE ABILITIES OF THE COMMUNICATIONS SERVER

A communications server is a dynamic system which must work in a highly changeable external environment. A communications server must be much more flexible and dynamic than a day-to-day system, because it must work with highly heterogeneous external information systems that keep up with rapid technological changes. When it comes to technologies, communications servers must be a step ahead of other systems, because otherwise it may turn out that the communications server ends up unable to perform its functions.

The goal of this paper is not to describe the internal architecture and ideology of communications servers precisely. The establishment of such systems is a very serious process throughout the world these days, and various solutions are being sought out that are linked to the following technologies:

- Distributed Dynamic Systems
- Distributed and Dynamic Objects
- Dynamic Object-Oriented Programming
- Reflection
- Domain Specific Programming Languages
- Artificial Intelligence

Many of these technologies are still quite new, and they are still being developed. This means that not all of them have ready-made tools that support various properties or functions of the technologies. Some tools exist, some are at the prototype stage, while some have already become popular among professionals (this is particularly true of prototype tools that are designed at universities and research laboratories in order to test the latest technologies). In the design of a communications server it is worthwhile to such modern technologies and research results as the Multilanguage Interpreter [6] and the Database Browser Generator [7].

EVALUATION OF REQUESTS FOR BILLING PURPOSES

A billing system is part and parcel of the mechanism whereby a communications server fulfills requests. When a specific request is fulfilled, the system not only does what has been requested, but it also automatically calculates the resources that are used in the process. Within the communications server, a price has been attached to every resource, and it can change on the basis of the volume of information that has been requested, the time of day when the request is filed, etc. The price of each request is calculated automatically and stored in a journal that then is used for billing purposes.

A resource is an information request to a register. The price of resources changes on the basis of the type of the request, the complexity of the request, the register that is involved, etc.

USES OF A COMMUNICATIONS SERVER

There are three major ways to use a communications server:

- As an international resource facility that can be used to access information from Latvian registers;
- As an internal resource facility that can be used to search for information in registers;
- As a way of setting up cooperation among various registers.

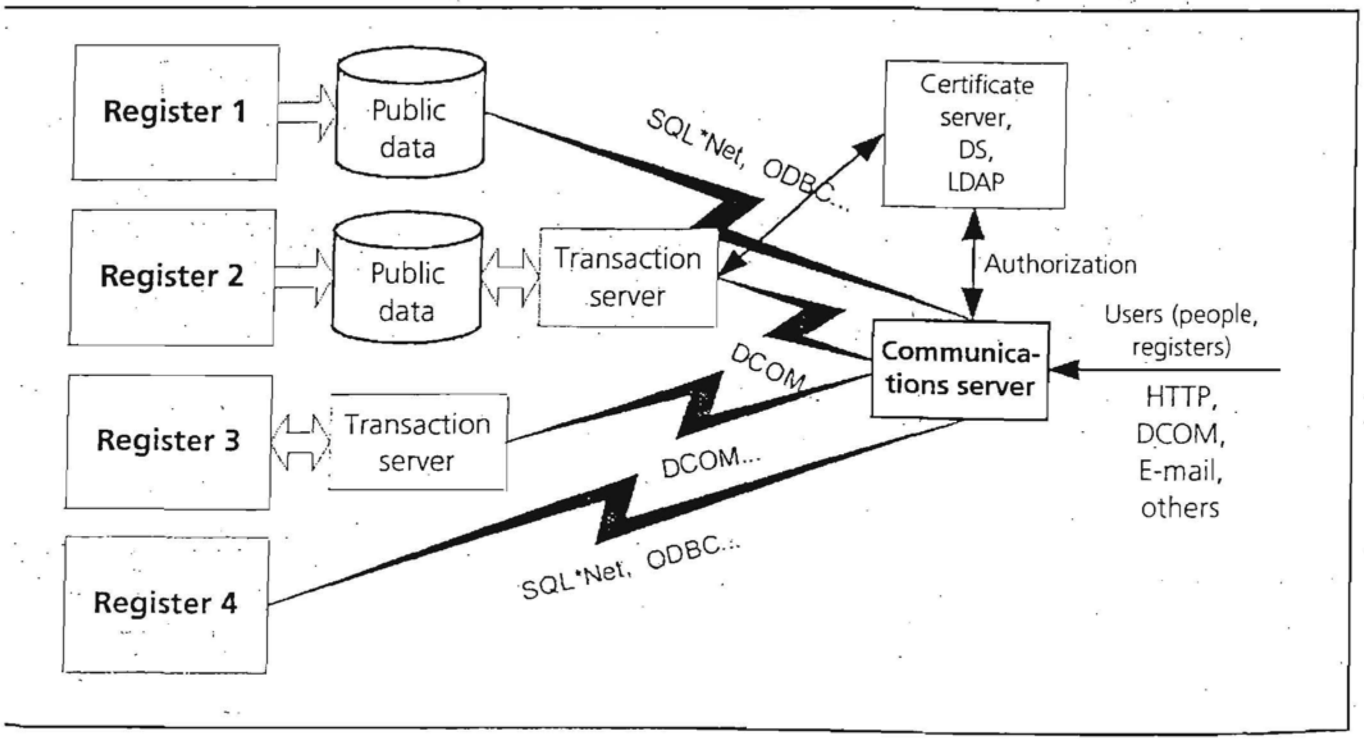


Figure 2. The structure of cooperation between a communications server and other entities

The need to access information from Latvian registers via a single contact facility is the main reason for elaborating the communications server. Of course, this is more than just a trivial solution in which a single Internet application is designed for connection to other registers via their Internet addresses. This simplified design does not deal with the main issue – the ability to collect information from various sources (i.e., various registers) without the user having to hook up to each register separately. The information that a user needs is collected from the various registers by the communications server, and the user himself may be completely unaware of the technical details of this process. Thus the communications server is needed by employees of foreign institutions in order to obtain information that is stored in Latvia's registered.

A second use for the communications server is the fulfillment of domestic information requests in Latvia. The previously described situation in which users do not want to or are unable to understand the technical details of information storage is typical among the personnel of Latvia's administrative structures. Of course, given the fact that access rights to authorization may vary for foreign users and Latvian users, the communications server sets out a unified set of requirements in this area, and solutions are the same for both groups of users.

The third way of using a communications server is to use it in order to exchange information among various registers. It is obviously irrational to maintain communications channels and to conduct information exchange individually with each of many registers that are mutually linked. It is much more rational to set up a centralized contact facility – the communications server – which is linked to all of the registers and through which information is exchanged among them. The general process of information exchange among registers via a communications server is shown at Figure 2.

This diagram shows four ways in which a register can be connected to a communications server. Every register that participates in the data exchange procedure can have its own data base in which those data that are intended for transfer to other registers and for publication can be separated out. The data base can be maintained by a separate computer or server so that approaches to the public data base do not hamper work with the basic data base of the register. Data from the basic data base are regularly copied to the public data base (an automatic replication mechanism). This solution is rational not only from the perspective of using communications channels; it also ensures:

- That the fulfillment of external requests does not hamper the work of the register;
- That there is higher security, i.e., that in the case of unauthorized access, the basic data base is not damaged.

The link between the communications server and the public data base can be implemented on the basis of various technologies, such as DCOM object calls, MS Transaction servers and Oracle SQL*NET. User authorization is provided via a certificate server, a directory server and the Lightweight Directory Access Protocol (LDAP).

REFERENCES

1. The Latvian national program "Informatics", Ministry of Transport, 1998, 211 pp.
2. The Latvian national program "Informatics" (summary), Ministry of Transport, 1998, 60 pp.
3. "The Baltic States Government Data Transmission Network: Conceptual and Methodological Considerations", Rīga, 1998, 11 pp.
4. "The Baltic States Government Data Communications Network. Feasibility Study for a Data Networking Concept to Improve the Interchange of Information Among the Baltic States", Rīga, 1998, 83 pp.
5. "The Integrated State Significance Information System (Megasystem): Conceptual and Methodological Considerations", Rīga, 1998, 16 pp.
6. Arnicāne, V., Arnicāns, G. and J. Bičevskis. "Multilanguage Interpreter", in Proceedings of the Second International Baltic Workshop, 1996, pp. 173-174.;
7. Arnicāns, G. "Application Generation for the Simple Database Browser Based on the ER Diagram", in Proceedings of the Third International Baltic Workshop, 1998, pp. 198-209.

Contact information:
University of Latvia
Raiņa bulv. 29-331, Rīga, LV-1050, Latvia
Tel.: +371 7228226
Fax: +371 7820153
E-mail: bics@lanet.lv

Baltic IT&T 2000

4th INTERNATIONAL CONFERENCE INFORMATION TECHNOLOGIES AND TELECOMMUNICATIONS IN THE BALTIC STATES

The Information Society: The Future for the Baltic Region

Radisson SAS Daugava Hotel, April 6 – 7, Rīga, Latvia

Information Technology Committee of the Baltic Council of Ministers
Data Media Group

Abstracts of papers from the Baltic IT&T 2000 Conference

The Unified Megasystem of Latvian Registers: Development of a Communications Server – the First Results and Conclusions

Mr. Ģirts Karnītis, assistant, Mr. Guntis Arnicāns, lecturer, Prof. Jānis Bičevskis, Head of Department of Computer Science, Faculty of Physics and Mathematics, University of Latvia

This paper describes a development of Communications Server, the first realization version and conclusions. A communications server is a set of software and computer equipment that allows a wide range of users (both in Latvia and in other countries) to receive information from a variety of sources (government registers, data bases, information systems) through a single contact point. A communications server identifies users, authorizes the use of the respective data, fulfills a request that involves several information sources, and evaluates the cost of the process so that the appropriate financial transaction can be made. A communications server allows users to learn where information is being stored and what kind of information it is, as well as to request and to receive information from various registers without any need for in-depth knowledge about the technical aspects of its storage.

INTRODUCTION

The need to establish a Communications Server became apparent when the governments of the Baltic States were setting up their joint data transmission network[6]. One of the main tasks is to obtain information about objects (enterprises, persons, motor vehicles, etc.) without having to study the data base structures in any country. One year ago the concept of Communications Server was defined [1] and project of Communications Server was started in Latvia.

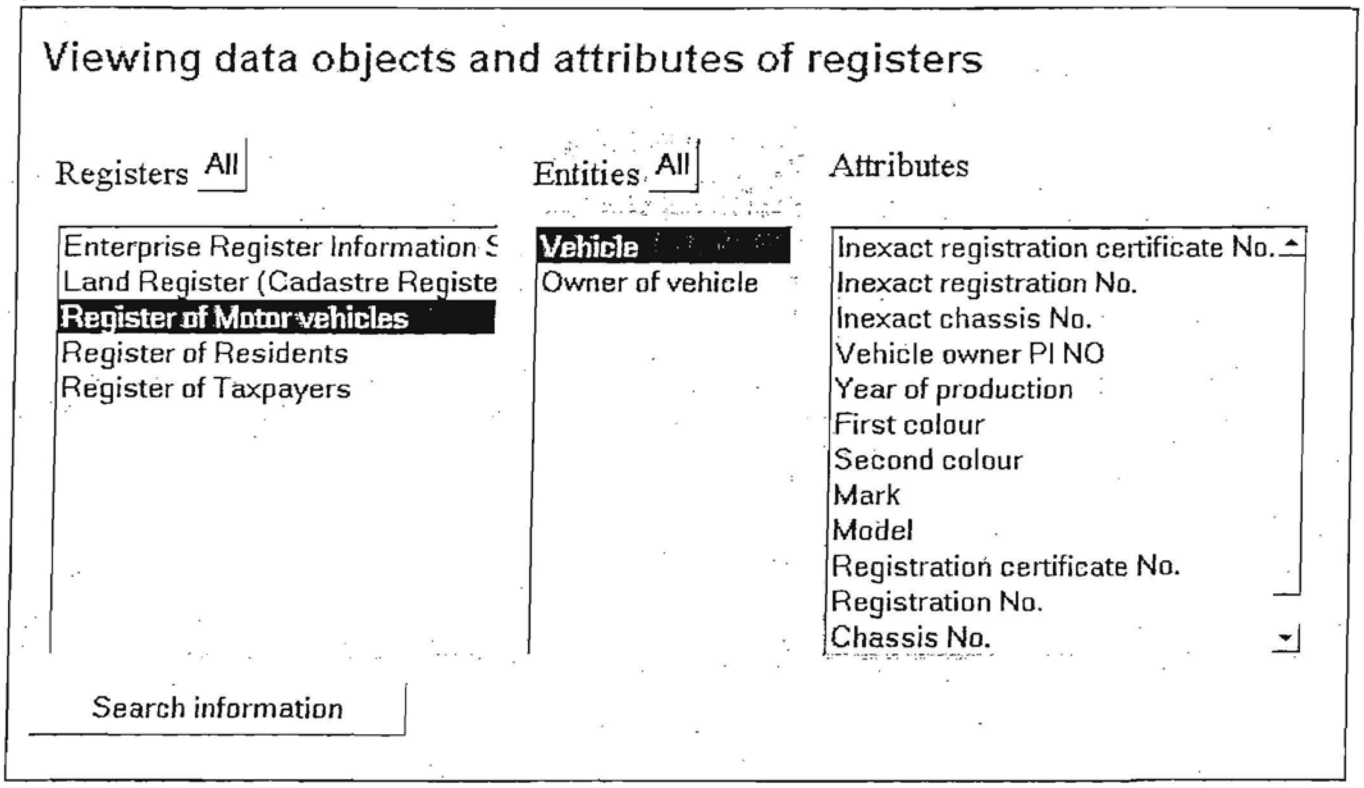
Data retrieval from different autonomous sources has become a hot topic during the last years not only in Latvia but also in all countries or large enterprises. The problem is very complicated and its solution can takes several years and many high-qualified specialists to solve it [2][3][4]. There was made the choice to develop Communications Server step by step in Latvia. Latvia has several dozens of registers and information sources (public and with restricted access). To develop all system at once it is too complex due to, for example, various organizational and technical problems. Design and implementation of all functionality for the Communications Server also takes much time.

CORE OF THE COMMUNICATION SERVER

The main functions for a Communications Server are:

1. User identification
2. Authorization with respect to the use of information
3. Management of user rights
4. Fulfillment of requests that involve several information sources
5. Evaluation of the costs of each request for billing purposes

It is more or less clear how to implement the first three functions, but the largest problems arise to develop last two functions. The original technology was developed to search and obtain data from various data sources during the design phase. This technology bases on WEB technologies and Meta models of data sources[5].



Viewing data objects and attributes of registers

Registers All

Enterprise Register Information S
Land Register (Cadastre Registe
Register of Motor vehicles
Register of Residents
Register of Taxpayers

Entities All

Vehicle
Owner of vehicle

Attributes

Inexact registration certificate No.
Inexact registration No.
Inexact chassis No.
Vehicle owner PI NO
Year of production
First colour
Second colour
Mark
Model
Registration certificate No.
Registration No.
Chassis No.

Search information

Figure 1. Registers and data objects

For the first version of Communications Server was determined several principles or requirements:

- It is possible define new source in couple days
- It is possible to access any type of data source
- It is easy and quickly create primitive services (wrappers) to search and obtain needed data from source
- It is possible to tie related data from various data sources
- It is easy maintain all system (make changes, add new possibilities, etc.)
- The program code have to be simple and small to reduce the possibility to make mistakes
- Initially data is retrieved only from WWW (from end-user point of view)

REGISTER OF REGISTERS

The Register of registers is the information system that contains information of other information systems maintained in Latvia. There is much useful information, such as IS name, content, owner, data model, relations with data objects in other information systems, in database of the Register of registers.

The first version of Communications Server widely uses information stored in Register of registers. For instance, the information searching starts with high level representation of data sources and objects stored in them. See the Figure 1.

We can see what data sources are available, what data objects are available from these sources and what attributes describe each data object. We can start browsing from any data source or data object.

BASIC ADDITIONAL REQUIREMENTS FOR COMMUNICATION SERVER

Various additional aspects and requirements were taken to create first version of Communications Server:

- some data are very sensible (only for authorized and restricted use)
- some data are available for money

For these reasons we keep a close attention to security, to log all activities and to accounting of all retrieved information to calculate accounts between information providers and consumers.

Security is designed to fulfill requirements determined by law, government and information source provider. At present for each user are defined: what data objects (register, information from register, etc.) are accessible, what operations can be done (searching and retrieving) and what templates of WWW

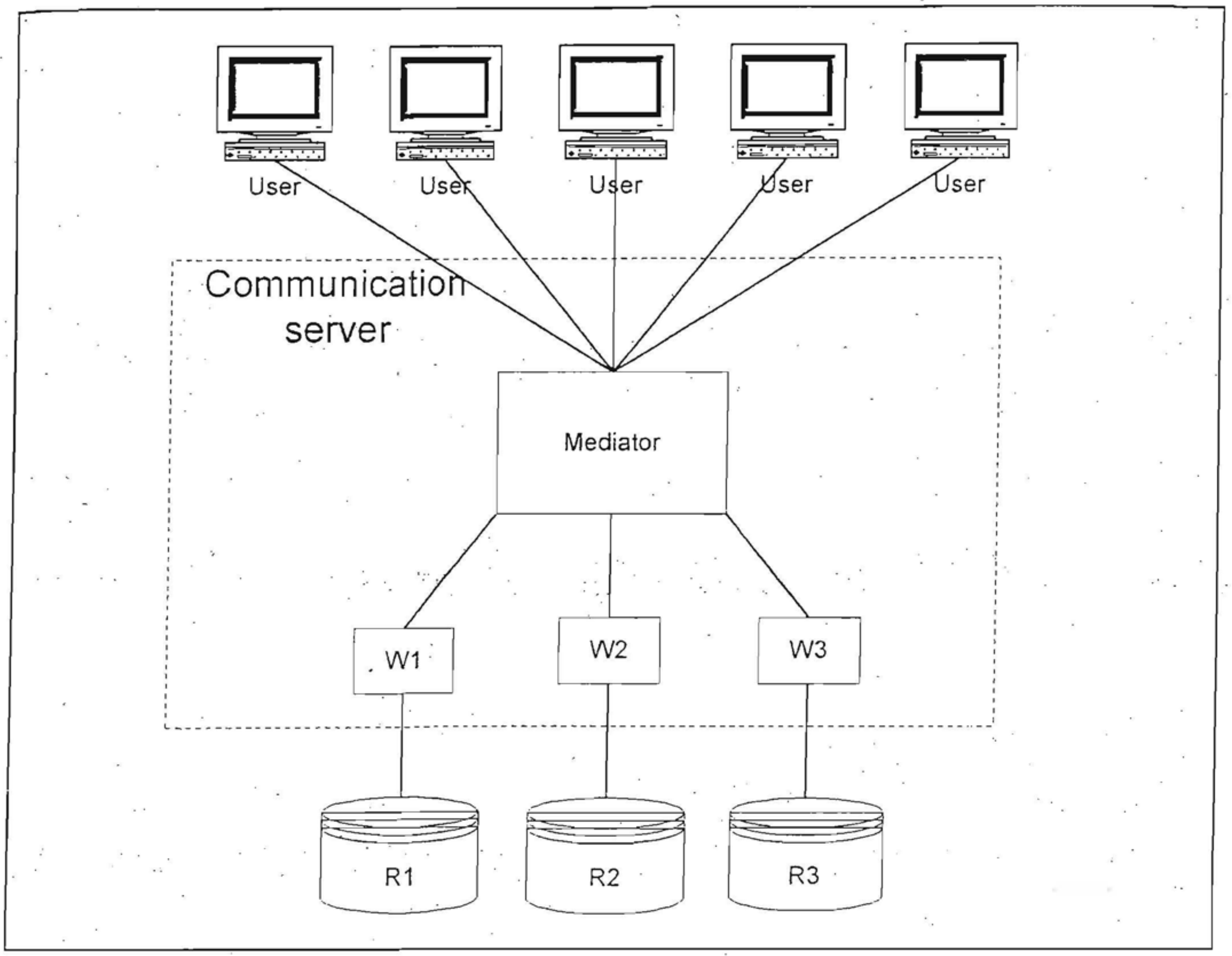


Figure 2. Conceptual shema of Communication Server

pages (data retrieving, combining from various registers and presentation) are available.

All user activities are logged in special journals. The system saves not only the type of activity and user who have done it, but also the request is stored. It is possible to track for any data object (person, for instance) all history - who asked what and what data objects and its attributes were displayed.

We can account costs for information consumers if the cost is defined for some information. Since we are logging any request with details then we can calculate overall accounts for any user and provider.

TECHNICAL SOLUTION

The main task for a Communications Server is to retrieve information from data sources. Let us see the rough view to the implementation principles (Figure 2). User asks the Mediator for information. The Mediator translates requests to set of internal small requests to data sources through wrappers. When the wrapper returns data, the Mediator forms the information presentation and sends the www page to user.

To retrieve information from data source, we have to create special small programs – data wrappers. This approach has the following advantages:

- It allows access data source via different protokols and methds – ODBC, OLE DB, SQL*Net, DCOM, etc.
- Data source usually is made to well suit for specific business tasks, it is not primary made for data access from other system (Communications Server). The access is limited, it is allowed execute some stored procedures to query data. Wrapper allows us to execute only authorized functions.
- Querying data source via functions allows us to have easy transfer real data from data source physical data model to our logical data model (stored in meta database) that is more understandable for the user.
- If the data source changes we need only correct the appropriate wrappers.

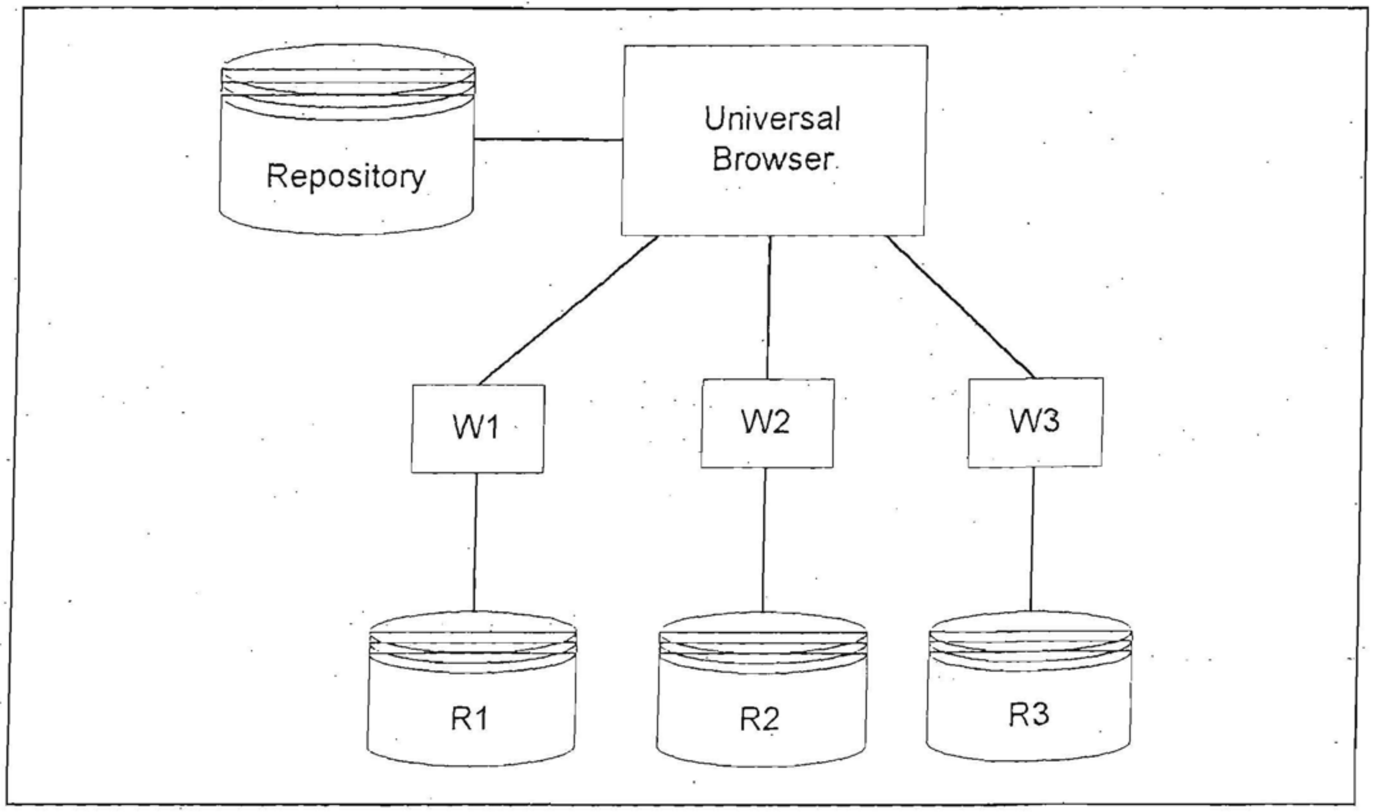
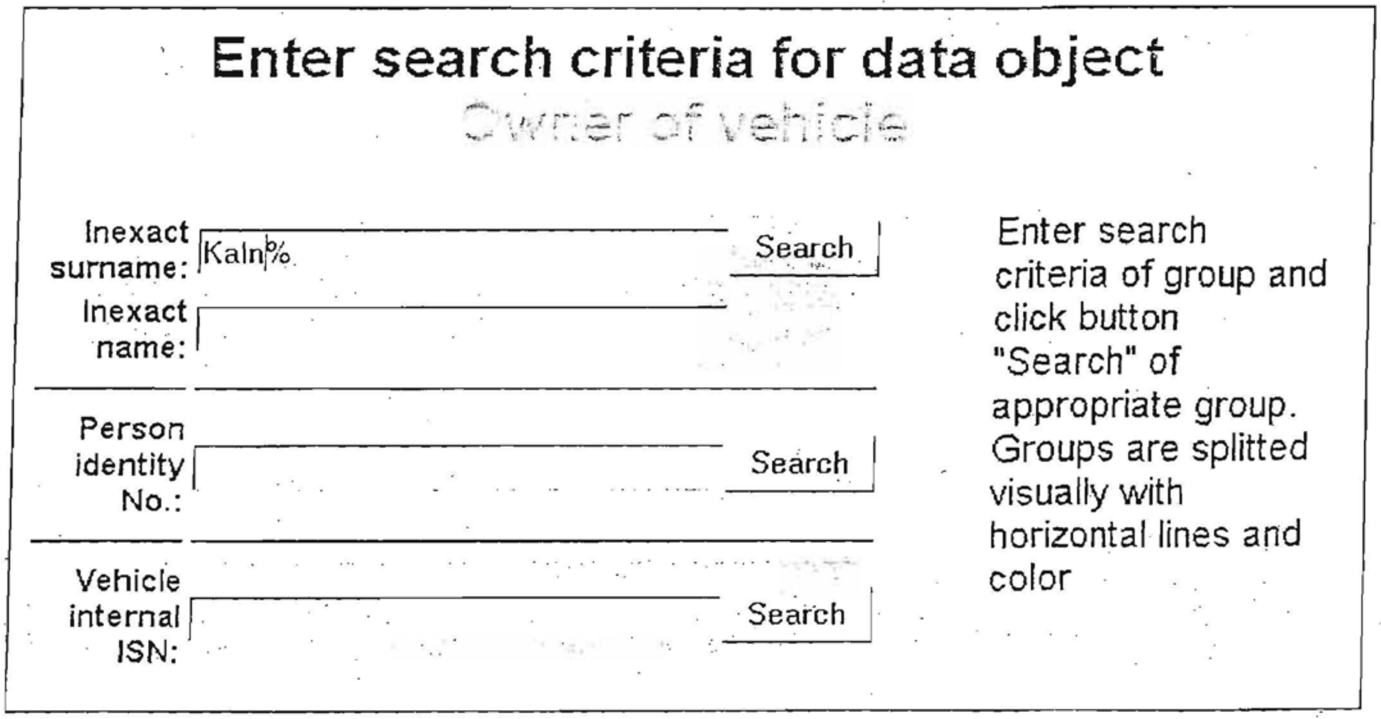


Figure 3. Universal browser

To communicate with user via Internet the special browser is designed that bases on a meta model of data sources. The browser takes the information stored in meta model, generate www pages to communicate with user. We can image browser as Driver and Repository (Figure 3).

Repository is database that stores information about data sources, data objects in sources and relations between them, functions that allows us query source, screen templates (www page structure) and other useful information.

Driver is special program that generate www pages to manage querying at high level and display information. The Driver can analyze relations between data sources and merge together all related information.



Enter search criteria for data object

Owner of vehicle

Inexact surname: Kaln%

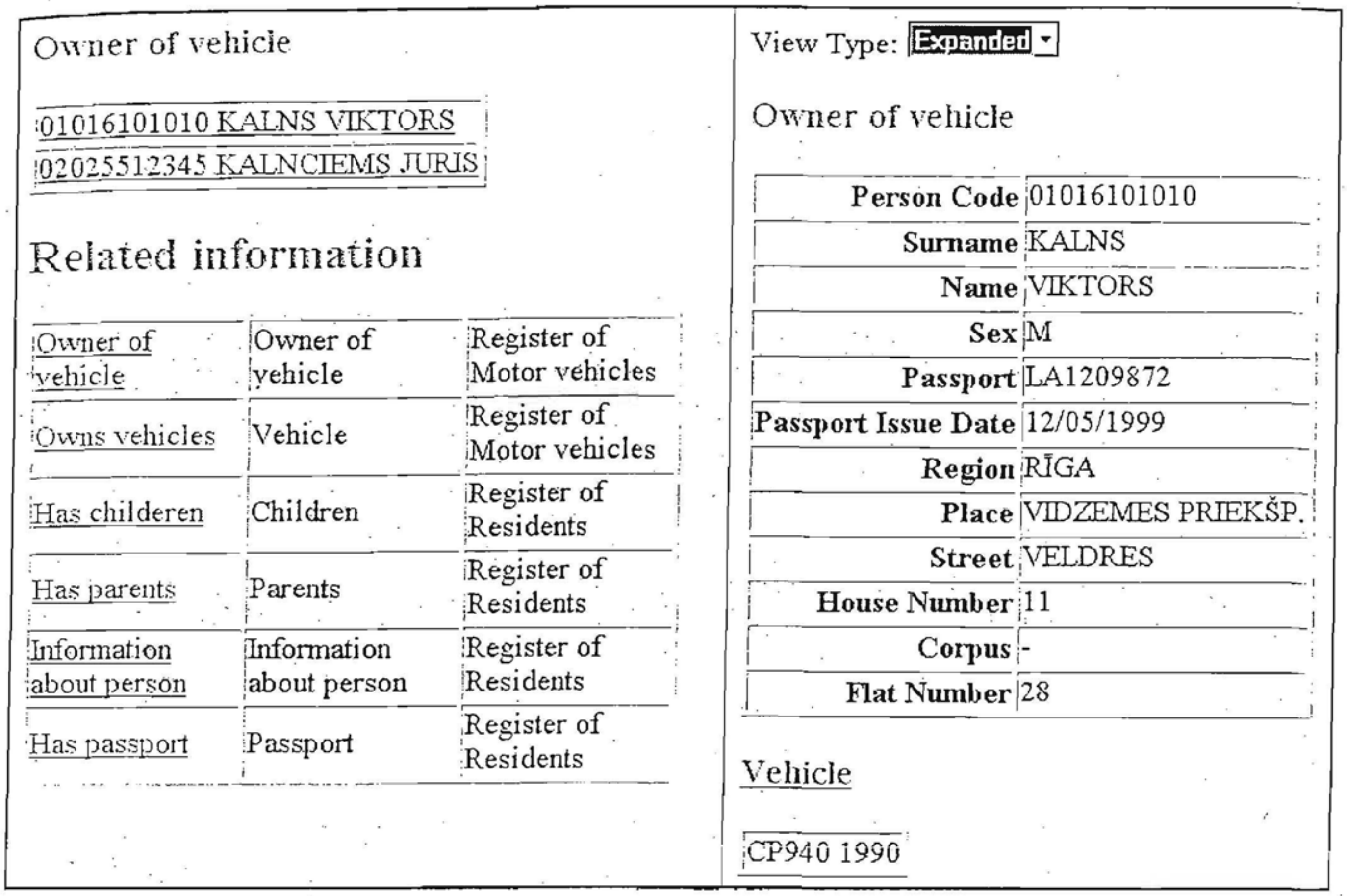
Inexact name:

Person identity No.:

Vehicle internal ISN:

Enter search criteria of group and click button "Search" of appropriate group. Groups are splitted visually with horizontal lines and color

Figure 4. Search criteria input window



Owner of vehicle

01016101010 KALNS VIKTORS
02025512345 KALNCIEMS JURIS

Related information

Owner of vehicle	Owner of vehicle	Register of Motor vehicles
Owns vehicles	Vehicle	Register of Motor vehicles
Has childeren	Children	Register of Residents
Has parents	Parents	Register of Residents
Information about person	Information about person	Register of Residents
Has passport	Passport	Register of Residents

View Type: Expanded

Owner of vehicle

Person Code	01016101010
Surname	KALNS
Name	VIKTORS
Sex	M
Passport	LA1209872
Passport Issue Date	12/05/1999
Region	RĪGA
Place	VIDZEMES PRIEKŠP.
Street	VELDRES
House Number	11
Corpus	-
Flat Number	28

Vehicle

CP940 1990

Figure 5. Information about car owners

DATA SEARCHING AND BROWSING SCENARIO

Let us look at small example how the Communications Server works from end-user point of view. First step is to choose from which register and which data object information will be quered (Figure 1). Then system asks search criteria for the choosen object (Figure 4).

User fills in search criteria and pushes button 'Search', system searches in appropriate the register for necessary information and results are showed (Figure 5).

From this screen user can easily get related information from other registers, for example, if user wants Information about Person from Register of Residents, user needs only to click on appropriate link and appropriate information are showed (Figure 6).

CONCLUSIONS AND FURTHER DIRECTIONS

The prototype of Communications Server was made in the middle of 1999 [7]. 4 registers (with test data) were connected for testing purposes. 2 of them use Oracle as DBMS and 2 others use Microsoft SQL Server. The prototype has shown the effectiveness of designed approach. The prototype of the system was much more powerful, than we expected and can be used as the real system. At present additional improvements is made and the first version of the real system is developed. This version is introduced in real exploitation now.

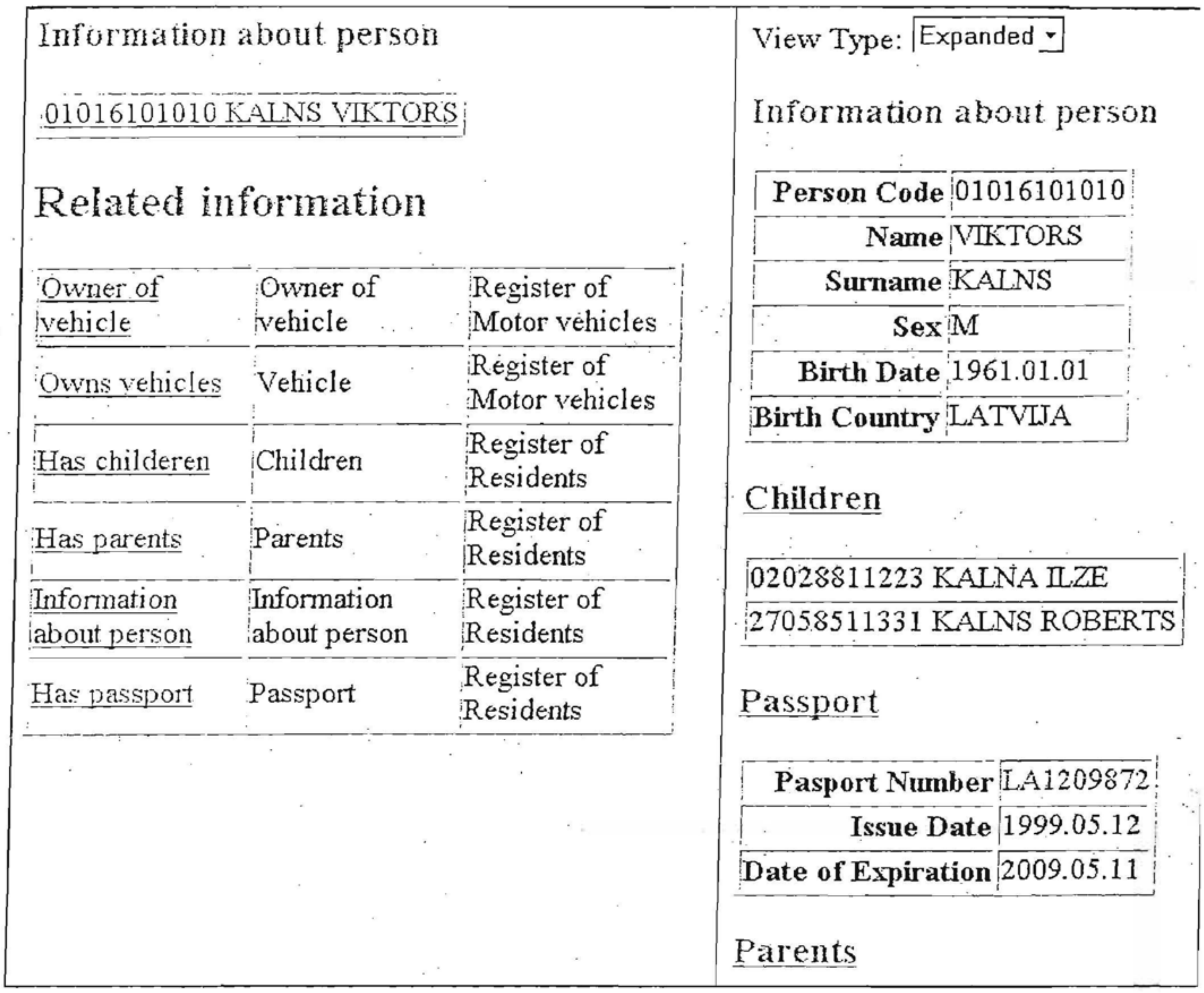
Future direction of our work is to develop a query processor that can take as input SQL-like query and return as output the result queried from multiple data source.

Other direction that already is partially developed – to make Communications Server available not only from WWW browsers, but also from custom programs using XML to query data and return answers.

REFERENCES

[1] Arnicans G, Bicevskis J, Karnitis G, "The Concept of Setting Up a Communications Server", in Abstracts of Papers of 3rd International Conference "Information Technologies and Telecommunications in the Baltic States", pp. 48-57, 1999

[2] Tomasic A, Amouroux R, Bonnet P, Kapitskaia O, Naacke H, and Raschid L, "The distributed information search component (disco) and the World Wide Web" in Proceedings of ACM SIGMOD International



Information about person

01016101010 KALNS VIKTORS

Related information

Owner of vehicle	Owner of vehicle	Register of Motor vehicles
Owns vehicles	Vehicle	Register of Motor vehicles
Has childeren	Children	Register of Residents
Has parents	Parents	Register of Residents
Information about person	Information about person	Register of Residents
Has passport	Passport	Register of Residents

View Type: Expanded

Information about person

Person Code	01016101010
Name	VIKTORS
Surname	KALNS
Sex	M
Birth Date	1961.01.01
Birth Country	LATVIJA

Children

02028811223 KALNA ILZE

27058511331 KALNS ROBERTS

Passport

Pasport Number	LA1209872
Issue Date	1999.05.12
Date of Expiration	2009.05.11

Parents

Figure 6. Information about person

Conference on Management of Data, Tuscon, Arizona, 1997, Prototype Demonstration.

[3] Haas L. M, Miller R. J, Niswonger B, Tork·Roth M, Schwarz P. M, Wimmers E. L, "Transforming Heterogeneous Data with Database Middleware: Beyond Integration", Data Engineering Bulletin 1999

[4] Hammer J, Garcia–Molina H, Ireland K, Papakonstantinou Y, Ullman J, Widom J, "Information translation, mediation, and Mosaic–based browsing in the TSIMMIS system", in Proceedings of ACM SIGMOD International Conference on Management of Data, 1995, Project Demonstration.

[5] Arnicans G, "Application generation for the simple database browser based on the ER diagram", Proceedings of the Third International Baltic Workshop, pp.198-209, 1998.

[6] "The Baltic States Government Data Transmission Network: Conceptual and Methodological Considerations", Rīga, 1998, 11 pp.

[7] www.mega.lv

Contact information:
Datorikas Institūts
Raiņa bulv. 29-220
Rīga, LV-1050
Latvia
Tel.: +371 7503383
Fax: +371 7503531
E-mail:girts@di.lv

Vilnius Gediminas Technical University
Institute of Mathematics and Informatics
Lithuanian Computer Society

DATABASES& INFORMATION SYSTEMS

PROCEEDINGS
OF THE

4th
IEEE
INTERNATIONAL
BALTIC
WORKSHOP

Edited by
Albertas ČAPLINSKAS

Vol. 1

Vilnius
Lithuania
May 1-5
2000

Vilnius "Technika" 2000

8. References

[1] Burdett, D. Internet Open Trading Protocol Version 0.9.9. The Open Trading Protocol Consortium, 1998.

[2] Christoffel, M. Pulkowski, S., Schmitt, B., Lockemann, P. Electronic Commerce: The roadmap for university libraries and their members to survive in the information jungle. *ACM Sigmod Record*, 27(4), 1998, pp. 68-73.

[3] Christoffel, M. A Trader for Services in a Scientific Literature Market. In *Proceedings of the 2nd International Workshop on Engineering Federated Information Systems (EFIS'99)*, Kühlungsborn, 1999, pp. 123-130.

[4] Hewlett Packard. *E-Speak - the platform for E-services*. http://www.e-speak.hp.com.

[5] IBM. *DB2 Digital Library*. http://www-4.ibm.com/software/is/dig-lib/about.html .

[6] JavaSoft. Java Remote Method Invocation Specification. Technical Report, Sun Microsystems, 1997. http://www.javasoft.com/products/jdk/1.1/docs/guide/rmi/.

[7] Karlsruher Virtueller Katalog. http://www.ubka.uni-karlsruhe.de/kvk.html.

[8] MeDoc- The Online Computer Science Library. http://medoc.informatik.tu-muenchen.de/english/medoc.html.

[9] Microsoft. DCOM Technical Overview. Technical Report, Microsoft Corporation, Redmond, 1996.

[10] Object Management Group. CORBA 2.0/IIOP Specification. Technical Report PTC/96-03-04, Framingham Corporate Center, Framingham (MA), USA, 1996.

[11] Pulkowski, S. Making Information Sources Available for a New Market in an Electronic Commerce Environment. In *Proceedings of the International Conference on Management of Information and Communication Technology (MICT'99)*, Copenhagen, 1999.

[12] Pulkowski, S.: Intelligent Wrapping of Information Sources: Getting Ready for the Electronic Market. In *Proceedings of the 10th VALA Conference on Technologies for the Hybrid Library*, Melbourne, 2000.

[13] Rachlevsky-Reich, B., Ben-Shaul, I. et. al. GEM: A Global Electronic Market System. In *Information Systems*, 24(6), 1999, pp. 495-518.

[14] Schmitt, B., Schmidt, A. METALICA: An Enhanced Meta Search Engine for Literature Catalogs. In *Proceedings of the 2nd Asian Digital Library Conference (ADL'99)*, Taipei, 1999.

[15] Stanford Digital Library Project. http://www-diglib.stanford.edu/diglib/.

[16] Stevens, W.R. *TCP/IP Illustrated, Volume 1, The Protocols.* Addison Wesley, Reading, 1995.

[17] Wang Baldonado, W., Winogred, T. Hi-Cites: dynamically created citations with active highlighting. In *Proceedings of the International Conference on Human factors in computing systems (CHI '98)*, Los Angeles, 1998, pp. 408-415.

[18] World Wide Web Consortium. Extensible Markup Language Recommendation. 1998. http://www.w3.org/TR/1998/REC-xml-19980210.

Heterogeneous Database Browsing in WWW Based on Meta Model of Data Sources

Guntis Arnicans, Girts Karnitis

University of Latvia
Faculty of Physics and Mathematics
Rainis Blvd. 19, Riga LV-1459, Latvia
garnican@lanet.lv, girts@di.lv

Abstract

This paper describes a development principle and technique for a simple universal multiple database browser. The browser operates by getting information from metamodel of data sources and actual data from legacy data sources. Every element such as entity, field, relation is mapped to some component of HTML page with appropriate structure and layout. Many templates of information layouts can be created allowing to dynamically change HTML page to acceptable user interface. The wrappers are used to provide browser with actual data and to act as mediators between data sources and browser. This approach allows to quickly describing new data sources, creating wrappers, making modifications later and managing data browsing in a simple unified style. The browser architecture is flexible enough to incorporate data sources with a variety of data models and query capabilities by various protocols. It is possible to select logically tied information from all available legacy data sources.

Keywords: Web-based information system, distributed information system, metamodels, database browsing.

1. Introduction

Data retrieval from different autonomous sources has become a hot topic during the last years. For instance, there are such data sources as enterprise register, register of pledges, register of state orders. When some state institution wants to order something from private business, civil servants are interested to know whether applicants are registered, whether they have pledges and what is their financial situation. Civil servants need information system that can collect related information from different Data Sources (DS) and show it.

We have found some such systems [2], [3], [4] that allow to do data querying from different data sources. All those systems are very complex, with their own query processor, but without universal user end. We decided to make a simple Universal Browser (UB) that acts on DS model during development of Megasystem and Communication server [5], [6].

Main ideas of the UB are described in [1], where the idea of database browsing based on the ER model is described. Our approach is a modified UB, that can browse multiple DS, which can be made in different technologies and with limited access rights and possibilities. Access to the DS is made via wrappers.

2. Repository of conceptual data models of data sources

Repository is a database that contains information about data sources (DS) and the links between them – the specific ER model. Repository also contains a description of functions that can be executed by DS.

2.1 Metamodel of repository

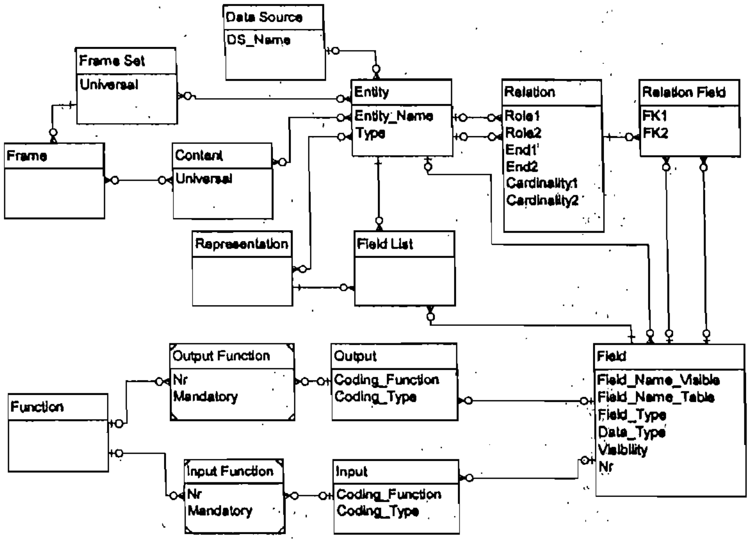


Figure 1. Metamodel of repository

Figure 1 shows an ER model of Universal Browser's (UB) repository. There are different parts in this model that are used for different purposes:

- Entities Data Source, Entity, Field, Relation, Relation Field contain DS models and information about entities and relations.
- Entities Function, Input, Input Function, Output, Output Function contain information about functions that query information from DS and input and output fields of these functions.

- Entities Representation and Field List contain information about visual representations for each entity i.e. what fields in what order have to be shown. For instance, let us take the entity Citizen that contains information about a person. In *short* representation fields PK, Name, Surname are visible, but in *long* representation fields PK, Name, Surname, Address have to be shown.
- Entities Frame Set, Frame and Content contain information about visual representation.

2.2 Conceptual model of data source

DS is a real existing legacy data source that exposes its data to other systems. Any DS can be made with different technologies, and expose its data in different ways. Any DS has some functions that can be executed to get information from DS. It is not necessary for the user to know technical details of DS to get information from it. The user needs a simple and understandable logical information representation that is related to the objects from the real world.

For example, information about cars can be stored in many tables in the real system. We are interested in conceptual data model, without technical details. It means a car can be represented with one entity in the conceptual model.

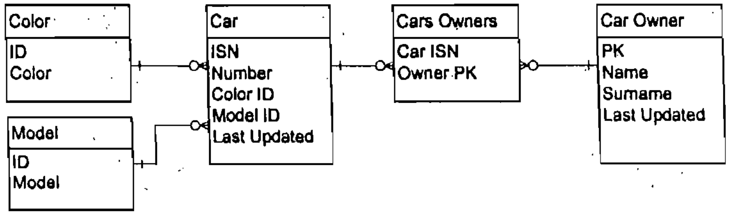


Figure 2. Example of physical data model

There can be such technological fields in the real database, which are necessary for the real system functioning, but they are not interesting for user and are not shown in the conceptual model.

There are two types of fields in the conceptual data model of DS:

- Fields that can be queried with some function,
- Fields from which we cannot query information. It means there are no functions where any of those fields are outputs. Usually these fields are not showed to the user, and they are used as input fields for some function. These fields are also used to link different entities.

There are links between DS entities, which means that, if you know information from one entity, you can get information from the other entity. There are links between entities, if such functions exist, which can query information from DS, using as an input information from other entity. For example, if you know some information about the person (especially person's PK (person code)), you can query the information about the person's passport. It means there is a link from person to passport. This function returns the passport number and the issue date. On the other hand, if you know the passport number, you can't get the passport's owner PK, because there is no function that returns this information. It also means that there can't be a link from passport to citizen.

2.3 Logical links between the data sources

There are entities of different types used to link together information from different DS. These entities are used as base class of DS entities and do not belong to any DS. For instance, the entity *Person with PK* is such a base class. This base class has only one field PK. This field is primary key for similar objects that concern person for most of DS. If you know the PK you can get the information related to the person information from the appropriate DS. For instance, *Person with PK* links together information from the entities *Citizen*, *Passport*, *Car* and *Car Owner* (**Figure 3**).

2.4 An example of repository

Two DS and one base class are given in **Figure 3**.

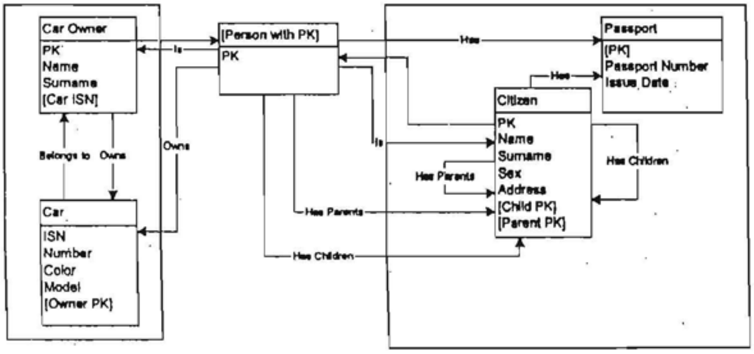


Figure 3. Example of conceptual model of data sources

Fields in square brackets are invisible fields used for search purposes only. Solid line with arrows means if you know information from the entity that is a starting point of the arrow, you can get the related information from the entity that is at the opposite end of the arrow. Interrupted line shows the relation between *normal entity* and *base class entity*. The values of arrows are shown in **Table 1**.

Table 1. Description of Relations

End1	PK1	End2	PK2	Relation name
Citizen	PK	Citizen	Child PK	Has Parents
Citizen	PK	Citizen	Parent PK	Has Children
Citizen	PK	Passport	PK	Has
Citizen	PK	Person With PK	PK	
Car Owner	PK	Car	Owner PK	Owns
Car Owner	PK	Person With PK	PK	
Car	ISN	Car Owner	Car ISN	Belongs To
Person With PK	PK	Car Owner	PK	IS
Person With PK	PK	Car	PK	Owns
Person With PK	PK	Passport	PK	Has
Person With PK	PK	Citizen	PK	Is
Person With PK	PK	Citizen	Child PK	Has Parents
Person With PK	PK	Citizen	Parent PK	Has Children

3. Browsing principles

General idea for dynamic browsing of various data sources is to generate Web pages with predefined information layout and functionality, get data from data sources and put them into page.

A web page consists of a set of frames (Frame) – FrameSet. The Frameset has a prefixed count of Frames, its layout and sizes. We can define as many as we need different FrameSets to organize and display information for the user. The FrameSet is a view to related data from one or many data sources. One of the Frames is the main Frame. The information in any other Frame is logically connected with data in the main Frame. The Frames can contain controls to manage the content in the other Frame.

The layout of the Frame is defined by rule, lets call it Content. Theoretically the Content is a formula or function: *Content(frameEntity, filterExpr)* where *frameEntity* is any entity from the metamodel of data sources and *filterExpr* is logical expression that filters data from appropriate data source. The Content defines: 1) what is the structure and principles of layout, 2) what data from metamodel and from actual legacy data sources are required to display information, 3) what actual instances of the defined entity are retrieved, 5) what controls are used to manage the content of the other Frame or to open the other FrameSet and 5) what related entities are involved from the same or any other data source. If we have various predefined Contents, then we can dynamically apply any Content to the Frame and get another data presentation for the same *frameEntity* and *filterExpr*.

4. Defining the Content of Frame

Let us assume that Content is the function ***Content(frameEntity, filterExpr)***. Let us detennine the means how we can define Content. We introduce the following data types:

- **entity** - determines the entity from the metamodel,
- **field** - determines the field of the entity from the metamodel,
- **relation** - determines the relation for two entities from the metamodel,
- **record** - determines the actual data from the data source for one fixed instance of the entity,
- **value** - determines the actual data of the field for one fixed instance of the entity,
- **string** - determines the character string,
- **list** - determines the list of elements with any other allowed data type, we denote such types by the element type followed by postfix "List",
- **updateAction** - determines the action that updates Frame
- **navigateAction** - determines the action that navigates browsing to another FrameSet
- **sObject** - determines the HTML object that contains string to display,
- **aObject** - determines the HTML object with assigned some action to perform,
- **fObject** - determines the HTML object that is formatted for displaying,
- **frame** - determines the Frame,
- **frameSet** - determines the FrameSet,
- **view** - determines the list of fields that must be displayed.

Let us rewrite the Content as a function ***Content(entity, expr(entity))***.

Let us introduce several additional functions to work with the metamodel and data sources, and to format HTML page.

<u>Functions to work with the metamodel</u>:

1. ***SourceName(entity) →string*** – returns the source name the entity belongs to
2. ***EntityName(entity) →string*** – returns the entity name
3. ***RelationList(entity) →relationList*** – returns all direct relations from the given entity to another entity (including itself) from the same data source
4. ***MetaRelationList(entity) →relationList*** – returns all indirect relations from the given entity to another entity from all available data sources
5. ***FieldList(entity, view) →fieldList*** – returns the list of all the fields of the entity
6. ***RelationName(relation) →string*** – returns the name (role) of the relation
7. ***FieldName(field) →string*** – returns the name of the field

8. *RelationEntity(relation)) →entity* – returns the entity at the opposite end of relation

Functions to work with data sources through wrappers:

1. ***RecordList(entity, expr(entity))→recordList*** – returns the list of instances (records) of the entity according to the given filtering expression
2. ***ValueList(record, view)→valueList*** – returns the list values of the given entity instance (record)
3. ***Value(value)→string*** – returns the field value as character string

Functions to work with the list:

1. ***List(element_1, element_2, ..., element_i)→list_1*** – returns the list of given elements and the list type *list_1* is appropriate to the element type
2. ***IterateList(n%list_1, function(n%))→list_2*** – returns the list *list_2* that has as elements the results applying the given function. The function is executed with each parameter *n%* that is taken from the list *list_1* denoted by the identifier *n%* (n is any unique integer) and the list type *list_2* is appropriate to the function return type
3. ***Concatenate(list_1, list_2)→list_3*** – returns the concatenation of two lists with the same element type.

Functions to format HTML page:

1. ***SO(string)→sObject*** – creates sObject from the character string
2. ***StringListObject(stringList, separatorString)→sObject*** – creates sObject from the list of character strings separated by *separatorString*
3. ***Update(frame, entity, expr(entity), content)→updateAction*** – activates information update into the frame with the given entity, filter expression and layout
4. ***Clear(frame)→updateAction*** – clears the given frame
5. ***Navigate(frameSet, entity, expr(entity), content)→navigateAction*** – navigates to another FrameSet and update main Frame with the given entity, filter expression and layout
6. ***Link(sObject, navigateAction, updateActionList)→aObject*** – converts sObject into aObject and assign the navigation action and set of update actions to it. Any of action parameters may be empty.
7. ***AO(sObject)→aObject*** – converts sObject into aObject with empty action
8. ***FO(aObject)→fObject*** – converts aObject into fObject without any special formating
9. ***HorizontalTable(aObjectListList)→fObject*** – creates fObject from the list of lists, this frame object is displayed as table, and internal lists are placed in rows
10. ***VerticalTable(aObjectListList)→fObject*** – creates fObject from the list of lists, this frame object is displayed as table, and internal lists are placed in columns

11. *ListBox(aObjectList)→fObject* – creates fObject from the list, this frame object is displayed as listbox

12. *Horizontal(fObjectList)→fObject* – creates new fObject by arranging the given list horizontally

13. *Vertical(fObjectList)→fObject* – creates new fObject by arranging the given list vertically.

Only frame objects with the type fObject may be displayed in the Web page.

5. Data wrappers

Function *RecordList* must be implemented to get information from DS. The technology we use is simple, but effective. UB gets information from DS via Wrappers. This approach has the following advantages:

- It allows to access DS via different protocols and methods – ODBC, OLE DB, SQL*Net, DCOM.
- DS usually are made well suited for specific business tasks. DS are not primary made for data access from UB. The access to DS data usually is limited, it is allowed to execute some stored procedures to query data. Wrapper allows us to execute only authorized functions.
- Querying DS via functions allows us to have easy transfer real data from DS physical data model to logical data model that is more understandable for the user.

Information about functions is stored in the UB meta database: defined input and output fields for each function. Each input field may be mandatory or optional.

During development of the prototype, we discovered some rules for function implementation and developing conceptual model of DS.

- First rule - it is desirable to have input and output fields from one entity. It simplifies development of DS model and wrappers.
- Second rule - two approaches possible for making DS model and functions. One approach is that we already have functions, and we make conceptual data model of DS using the first rule. In case DS is a system we maintain and own, it is often possible to make functions according to conceptual data model of DS. In such a case we make conceptual data model of DS at first and then we make data access functions according to conceptual data model and the first rule. It is helpful to make two types of functions:

1. The function gets information identifying the object from DS by some search criteria. For example, get person's PK by its name and surname (might be partial). The answer usually is a list of person's identifying information according to search criteria.
2. The function gets information about one object from one entity by its identifier. An example - get all information about the citizen by its PK.

For instance, we have two functions for the entity Citizen:

1. Input data – Name, Surname (might be partial). Output data – PK, Name, Surname (full).
2. Input data – PK. Output data – PK, Name, Surname, Address.

There are also 2 functions to get information about the citizen's parents and children:

3. Input data – Parent PK. Output data – Children PK, Name, Surname.
4. Input data – Child PK. Output data – Parents PK, Name, Surname.

There is a procedure that implements the function ***RecordList***. This procedure gets the entity and filter expression as input and returns data from DS as output. In our implementation this procedure gets information from the meta database about functions that can be executed over entity from which we need information. In our implementation the filter expression is fields and corresponding values for these fields, e.g. PK="123456-111111". There is "brute force" algorithm that finds functions we can execute e.g. those are functions that have enough input data from the filter expression to be executed, executes these functions and returns result. There can be, of course, other implementations.

DS data access via wrappers allows to connect new DS to our system easily and quickly. We have to write a new wrapper and add information about new DS to the meta database. With some experience the writing of wrappers is easy and fast process, and there is no need to make any modification in DS.

6. Templates for Web page structure and functionality

The design of FrameSet and Frames is based on template principle. With some experience the new FrameSets and Frames can be developed quickly. The design has two main steps – FrameSet structure planning and creating formulas for Frame Contents. We give some templates and ideas how the Web pages can be designed. The above given functions are used.

6.1 Simple entity instance presentation in table

The first column contains field names and the second – field values

A(entity, record) = VerticalTable(List(A1, A2))
A1 = IterateList(1%FieldList(entity, view), AO(SO(FieldName(1%))))
A2 = IterateList(2%ValueList(record, view), AO(SO(Value(2%))))

PK	12121211111
Name	Andris
Surname	Kalns
Sex	M
Address	Riga, Liepu 1-12, LV-1000

Figure 4. Example of entity instance presentation

6.2 Entity instance presentation as text

Instance field values are concatenated according to select *view*.

B(record) = FO(AO(SO(StringListObject(B1, " "))))
B1 = IterateList(3%ValueList(record, view), Value(3%))

12121211111 Andris Kalns M Riga, Liepu 1-12, LV-1000

6.3 Entity relations presentation in vertical list

Each relation is represented as relation name concatenated with entity name at the opposite relation end.

C(entity) = Vertical(IterateList(4%RelationList(entity), C1))
C1 = Horizontal(List(C2, FO(AO(SO(" "))), C3))
C2 = FO(AO(SO(RelationName(4%))))
C3 = FO(AO(SO(EntityName(RelationEntity(4%)))))

Has Passport
Has Parents Citizen
Has Children Citizen

Figure 5. Example of relations presentation

6.4 All relation presentation in table

The data about all relations (relation name, entity name and data source) are placed in table with headings.

D(entity, expr(entity))=HorizontalTable(Concatenate(D1,D2))
D1 = AO(StringListObject("Relation", "Entity name", "Data source"))
D2 = IterateList(5%MetaRelationList(entity),List(D3, D4, D5))
D3 = AO(SO(RelationName(5%)))
D4 = AO(SO(EntityName(RelationEntity(5%))))
D5 = AO(SO(SourceName(RelationEntity(5%))))

Relation	Entity name	Data source
Is	Citizen	Register of Residents
Has	Passport	Register of Residents
Has Parents	Citizen	Register of Residents
Has Children	Citizen	Register of Residents
Is	Car Owner	Register of Motor vehicles
Owns	Car	Register of Motor vehicles

Figure 6. Example of relation presentation

6.5 An example of FrameSet

Let us look how a FrameSet can be built. Let us assume FrameSet *FRS_1* with 4 Frames – *FR_1, FR_2, FR_3, FR_4*. FR_1 is used to list instances of entity, FR_2 – to show details of fixed instance in FR_1, FR_3 – to list all relations to other entities in all data sources, FR_4 – to show details of another related entity instances for FR_2 or FR_4. See Figure 7.

At first let us create three presentations or Contents (E, F, G) for viewing entities. We use formulas created before in this paper.

- Content formula E() for Frame FR_4 (from FR_4 we can update all Frames in FRS_1)

```
E(entity, expr(entity)) = Vertical(E1, E5)
E1 = Horizontal(List(FO(E2), FO(AO(SO(" "))), FO(AO(SO(SourceName(entity))))))
E2 = Link(SO(EntityName(entity)), E3, E4)
E3 = Navigate("FRS_1", entity, expr(entity), "")
E4 = List(Clear("FR_2"), Update("FR_3", entity, expr(entity), ""), Clear(FR_4))
E5 = Vertical(IteateList(6%RecordList(entity, expr(entity)), A(entity, 6%)))
```

- Content formula F() and G() for Frame FR_2 (from FR_2 we can update this frame or update FR_4)

```
F(entity, expr(entity)) = Vertical(H, FO(AO(SO(" "))), E5)
G(entity, expr(entity)) = Vertical(H, FO(AO(SO(" "))), Vertical(E5, G1))
G1 = C(entity), where C3 is substitute with G2 in all places (we have added the action)
G2 = FO(Link(SO(EntityName(RelationEntity(4%))), NULL, G3))
G3 = List(Update("FR_4", RelationEntity(4%), expr(RelationEntity(4%)), "E"))
H = ListBox(List(Link("Presentation F", NULL, H1), Link("Presentation G", NULL, H2)))
H1 = Update("FR_2", entity, expr(entity), "F")
H2 = Update("FR_2", entity, expr(entity), "G")
```

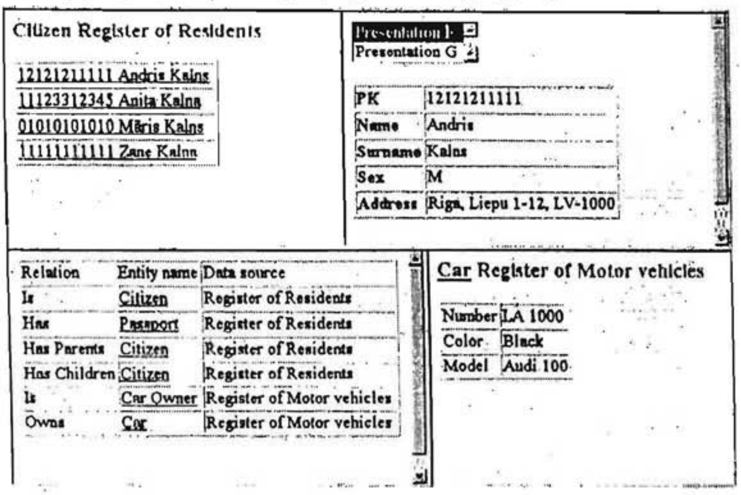
- Content formula I() for Frame FR_1 (from FR_1 we can update FR_2, FR3, FR_4)

```
I(entity, expr(entity)) = Vertical(I1, I2)
I1 = Horizontal(List(FO(EntityName(entity)), FO(AO(SO(" "))), FO(AO(SO(SourceName(entity))))))
I2 = HorizontalTable(IterateList(7%RecordList,Link(B(7%), NULL, I3))
I3 = List(Update("FR_2", entity, expr(entity) and expr(7%), "F"),
        Update("FR_3", entity, expr(entity) and expr(7%), ""), Clear("FR_4"))
```

- Content formula J() for Frame FR_3 (from FR_3 we can update FR_4)

```
J(entity, expr(entity)) = D(entity, expr(entity)), where D4 is substitute with J1 in all places (we have add the action)
```

J1 = Link(SO(EntityName(RelationEntity(5%))), NULL, J2)
J2 = List(Update("FR_4", RelationEntity(5%), expr(RelationEntity(5%)), "E"))



Citizen Register of Residents

12121211111 Andris Kalns
11123312345 Anita Kalna
01010101010 Māris Kalns
11111111111 Zane Kalna

Presentation F
Presentation G

PK	12121211111
Name	Andris
Surname	Kalns
Sex	M
Address	Riga, Liepu 1-12, LV-1000

Relation	Entity name	Data source
Is	Citizen	Register of Residents
Has	Passport	Register of Residents
Has Parents	Citizen	Register of Residents
Has Children	Citizen	Register of Residents
Is	Car Owner	Register of Motor vehicles
Owns	Car	Register of Motor vehicles

Car Register of Motor vehicles

Number	LA 1000
Color	Black
Model	Audi 100

Figure 7. Example of WWW page

7. Conclusions and future directions

The prototype of the UB is made during developing Megasystem and Communication Server. Four registers test databases are connected to the UB for testing purposes. Two of them use Oracle as DBMS, other two use Microsoft SQL Server.

The UB prototype shows the effectiveness of our approach and is being initiated as first version of the real system.

There are many aspects that are very important in real life application, but not covered in this article – security, user authorization, logging, query cost calculation. All these features are incorporated in the UB.

The UB is useful in many large organizations having many autonomous data sources as a browser for these systems with integrated view.

Future direction of our work is to develop a query processor that can take as input SQL-like query and return as output the result queried from multiple data sources. Other directions of future work – to make CS available not only from WWW browsers, but also from custom programs using XML to query data and return answers.

8. References

[1] Arnicans G, "Application generation for the simple database browser based on the ER diagram", Proceedings of the Third International Baltic Workshop, pp.198-209, 1998.

[2] Tomasic A, Amouroux R, Bonnet P, Kapitskaia O, Naacke H, and Raschid L, "The distributed information search component (disco) and the World Wide Web" in Proceedings of ACM SIGMOD International Conference on Management of Data, Tuscon, Arizona, 1997, Prototype Demonstration.

[3] Haas L. M, Miller R. J, Niswonger B, Tork Roth M, Schwarz P. M, Wimmers E. L, "Transforming Heterogeneous Data with Database Middleware: Beyond Integration", Data Engineering Bulletin 1999.

[4] Hammer J, GarciaMolina H, Ireland K, Papakonstantinou Y, Ullman J, Widom J, "Information translation, mediation, and Mosaicbased browsing in the TSIMMIS system", in Proceedings of ACM SIGMOD International Conference on Management of Data, 1995, Project Demonstration.

[5] Arnicans G, Bicevskis J, Karnitis G, "The Concept of Setting Up a Communications Server", in Abstracts of Papers of 3rd International Conference "Information Technologies and Telecommunications in the Baltic States", pp. 48-57, 1999.

[6] www.mega.lv

Databases and Information Systems

Fourth International Baltic Workshop, Baltic DB&IS 2000 Vilnius, Lithuania, May 1–5, 2000 Selected Papers

Edited by

Janis Barzdins

Institute of Mathematics and Computer Science, University of Latvia, Riga

and

Albertas Caplinskas

Institute of Mathematics and Informatics, Vilnius

KLUWER ACADEMIC PUBLISHERS

DORDRECHT / BOSTON / LONDON

Heterogeneous Database Browsing in WWW Based on Meta Model of Data Sources

Guntis Arnicans, Girts Karnitis

University of Latvia
Faculty of Physics and Mathematics
Raina Blvd. 19, Riga LV-1586, Latvia
garnican@lanet.lv, girts@di.lv

Abstract

This paper describes a development principle and technique for a simple universal multiple database browser. The browser operates by getting information from metamodel of data sources and actual data from legacy data sources. Every element such as entity, field, and relation is mapped to some component of HTML page with appropriate structure and layout. Many templates of information layouts can be created allowing to dynamical changing of HTML page to acceptable user interface. The wrappers are used to provide browser with actual data and to act as mediators between data sources and browser. This approach allows to quickly describing new data sources, creating wrappers, making modifications later and managing data browsing in a simple unified style. The browser architecture is flexible enough to incorporate data sources with a variety of data models and query capabilities by various protocols. It is possible to select logically tied information from all available legacy data sources.

Keywords: Web-based information system, distributed information system, metamodels, database browsing.

1. Introduction

Organisations, both governmental and business, have to manage large amount of information stored in some form of databases or files. One of the main problems to deal with information managing is the weak interoperability between various databases and information systems. Especially this problem is serious when we want organise collaboration between the information systems of various organisations.

In nowadays a significant fraction of new information systems or services bases on the Web solutions. Usually developers use Web applications to organise communications between data source and data consumer (user) but data sources sometimes remain the old ones from the current or previous information systems. This leads to the operation with very heterogeneous data. To deal with problems the metadata of the data sources (data structure, content, attributes, etc.) are used to describe the heterogeneous information models. This approach supports the creating of very dynamical systems and it is easy to maintain system in the rapidly changing world.

In this paper we describe some results achieved during the development of two projects - the Integrated State Significance Information System (Megasystem) and the Baltic States Government Data Transmission Network (Network) [2, 5]. The goal of these projects is to provide fundamental improvements in the exchange of telecommunications and data among the administrative institutions of the Baltic States. The principles described in this paper were used to build up the first implementation of *Communication server.* A Communication server is a set of software and computer equipment that allows a wide range of users to receive information from variety of sources (governmental registers, databases, information systems) through a single contact point. Among the other significant functions the Communication server fulfils a requests that involves several information sources, merges together information, allows users to learn where information is stored and what kind of information it is, and to receive information from various registers without any need for in-depth knowledge about the technical aspects of its storage.

Data retrieval from different autonomous sources has become a hot topic during the last years in the other countries and large enterprises also. There are many different approaches to deal with this task. For instance, the systems described in [3, 4, 6, 7] allow data querying from different Data Sources (DS). All those systems are very complex, with their own query processor, but without universal user end. The development of these systems consumes many resources (time, money, people).

Our first aim was to make a simple Universal Browser (UB) that acts on model of data sources and is very useful in practice (relative to consumed development resources). Main ideas of the UB are described in [1], where the idea of database browsing based on the ER model is described. Our approach is a modified UB that can browse multiple DS, which can be made in different technologies and with limited access rights and possibilities. Access to the DS is made via wrappers. Information retrieval bases on logical data models, information between different data model are tied via special logical data entities. The simple means are offered to obtain information and display it on WWW page – the set of functions that allows to create executable formulas.

2. Repository of Conceptual Data Models of Data Sources

Repository is a database that contains information about data sources (DS) and the links between them – the specific ER model. Repository also contains a description of functions that can be executed by DS.

2.1 METAMODEL OF REPOSITORY

Figure 1 shows an ER model of Universal Browser's (UB) repository. There are different parts in this model that are used for different purposes:

- Entities *Data Source, Entity, Field, Relation, Relation Field* contain DS models and information about entities and relations.
- Entities *Function, Input, Input Function, Output, Output Function* contain information about functions that query information from DS and input and output fields of these functions.
- Entities *Representation* and *Field List* contain information about visual representations for each entity i.e. what fields in what order have to be shown. For instance, let us take the entity *Citizen* that contains information about a person. In *short* representation fields *PK, Name, Surname* are visible, but in *long* representation fields *PK, Name, Surname, Address* have to be shown.
- Entities *Frame Set, Frame* and *Content* contain information about visual representation.

2.2 CONCEPTUAL MODEL OF DATA SOURCE

DS is a real existing legacy data source that exposes its data to other systems. Any DS can be made with different technologies, and expose its data in different ways. Any DS has some functions that can be executed to get information from DS. It is not necessary for the user to know technical details of DS to get information from it. The user needs a simple and understandable logical information representation that is related to the objects from the real world.

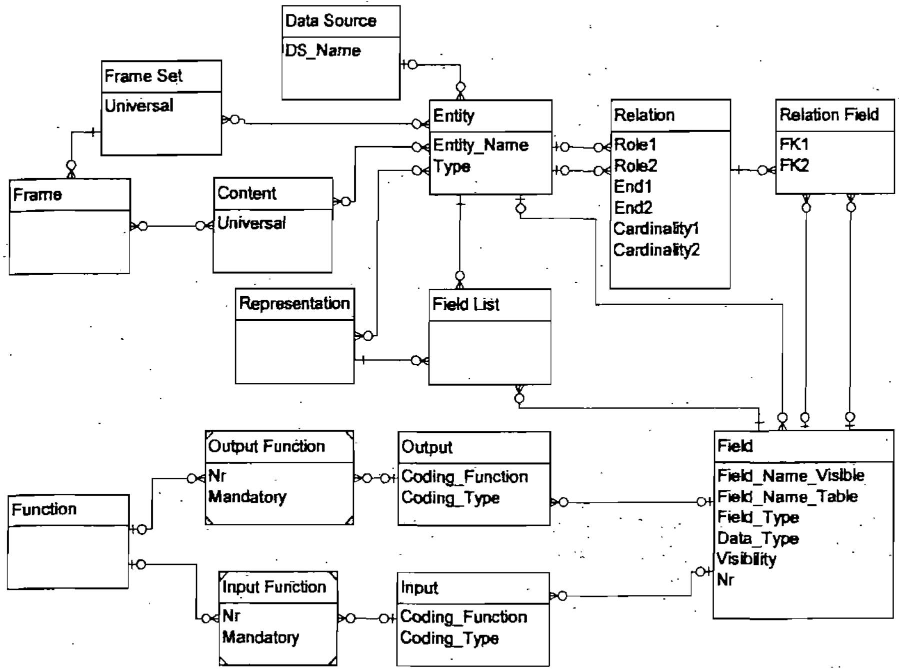


Figure 1. Metamodel of repository

For example, information about cars can be stored in many tables in the real system. We are interested in conceptual data model, without technical details. It means a car can be represented with one entity in the conceptual model.

There can be such technological fields in the real database, that are essential for the real system functioning, but they are not necessary for user and are not shown in the conceptual model.

There are two types of fields in the conceptual data model of DS:

- Fields that can be queried with some function,
- Fields from which we cannot query information. It means there are no functions where any of those fields are outputs. Usually these fields are not showed to the user, and they are used as input fields for some function. These fields are also used to link different entities.

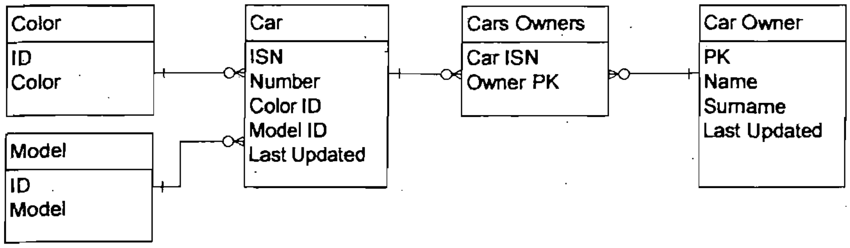


Figure 2. Example of physical data model

There are links between DS entities, which means that, if you know information from one entity, you can get information from the other entity. There are links between entities, if such functions exist, which can query information from DS, using as an input information from other entity. For example, if you know some information about the person (especially person's PK (person code)), you can query the information about the person's passport. It means there is a link from person to passport. This function returns the passport number and the issue date. On the other hand, if you know the passport number, you cannot get the passport's owner PK, because there is no function that returns this information. It also means that there cannot be a link from passport to citizen.

2.3 LOGICAL LINKS BETWEEN THE DATA SOURCES

There are entities of different types used to link together information from different DS. These entities are used as base class of DS entities and do not belong to any DS. For instance, the entity *Person with PK* is such a base class. This base class has only one field PK. This field is primary key for similar objects that concern person for most of DS. If you know the PK you can get the information related to the person information from the appropriate DS. For instance, *Person with PK* links together information from the entities *Citizen*, *Passport*, *Car* and *Car Owner* (Figure 3).

2.4 AN EXAMPLE OF REPOSITORY

Two Data Sources and one base class are given in Figure 3.

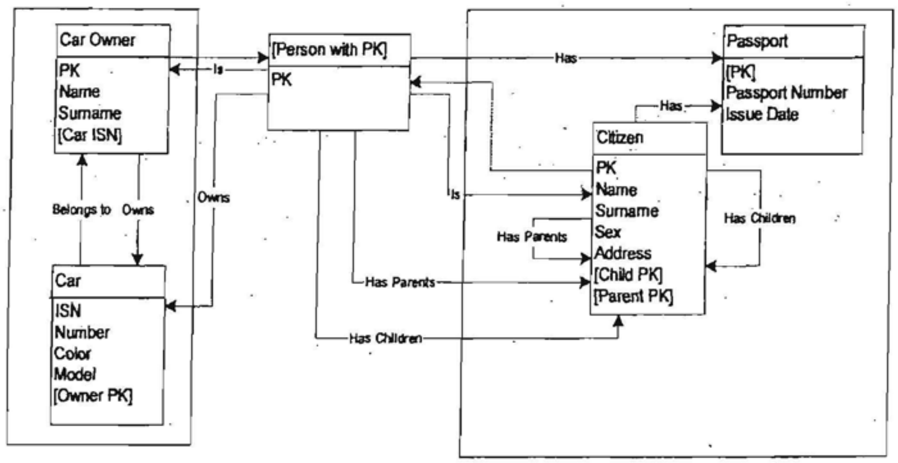


Figure 3. Example of conceptual model of data sources

Fields in square brackets are invisible fields used for search purposes only. Solid line with arrows means if you know information from the entity that is a starting point of the arrow, you can get the related information from the entity that is at the opposite end of the arrow. Interrupted line shows the relation between *normal entity* and *base class entity*. The values of arrows are shown in Table 1.

Table 1. Description of relations

End1	PK 1	End2	PK2	Relation name
Citizen	PK	Citizen	Child PK	Has Parents
Citizen	PK	Citizen	Parent PK	Has Children
Citizen	PK	Passport	PK	Has
Citizen	PK	Person With PK	PK	
Car Owner	PK	Car	Owner PK	Owns
Car Owner	PK	Person With PK	PK	
Car	IS N	Car Owner	Car ISN	Belongs To
Person With PK	PK	Car Owner	PK	IS
Person With PK	PK	Car	PK	Owns
Person With PK	PK	Passport	PK	Has
Person With PK	PK	Citizen	PK	Is
Person With PK	PK	Citizen	Child PK	Has Parents
Person With PK	PK	Citizen	Parent PK	Has Children

3. Browsing Principles

General idea for dynamic browsing of various data sources is to generate Web pages with predefined information layout and functionality, get data from data sources and put them into page.

A Web page consists of a set of frames (Frame) – FrameSet. The FrameSet has a prefixed count of Frames, its layout and sizes. We can define as many as we need different FrameSets to organise and display information for the user. The FrameSet is a view to related data from one or many data sources. One of the Frames is the main Frame. The information in any other Frame is logically connected with data in the main Frame. The Frames can contain controls to manage the content in the other Frame.

The layout of the Frame is defined by rule, lets call it Content. Theoretically the Content is a formula or function: ***Content(frameEntity, filterExpr)*** where *frameEntity* is any entity from the metamodel of data sources and *filterExpr* is logical expression that filters data from appropriate data source. The Content defines:

1) what is the structure and principles of layout,
2) what data from metamodel and from actual legacy data sources are required to display information,
3) what actual instances of the defined entity are retrieved,
4) what controls are used to manage the content of the other Frame or to open the other FrameSet,
5) what related entities are involved from the same or any other data source. If we have various predefined Contents, then we can dynamically apply any Content to the Frame and get another data presentation for the same *frameEntity* and *filterExpr*.

4. Defining the Content of Frame

Let us assume that Content is the function ***Content(frameEntity, filterExpr)***. Let us determine the means how we can define Content.

We introduce the following data types:

- **entity** - determines the entity from the metamodel,
- **field** - determines the field of the entity from the metamodel,

- **relation** - determines the relation for two entities from the metamodel,
- **record** - determines the actual data from the data source for one fixed instance of the entity,
- **value** - determines the actual data of the field for one fixed instance of the entity,
- **string** - determines the character string,
- **list** - determines the list of elements with any other allowed data type, we denote such types by the element type followed by postfix "List",
- **updateAction** - determines the action that updates Frame,
- **navigateAction** - determines the action that navigates browsing to another FrameSet,
- **sObject** - determines the HTML object that contains string to display,
- **aObject** - determines the HTML object with assigned some action to perform,
- **fObject** - determines the HTML object that is formatted for displaying,
- **frame** - determines the Frame,
- **frameSet** - determines the FrameSet,
- **view** - determines the list of fields that must be displayed.

Let us rewrite the Content as a function ***Content(entity, expr(entity))***.

Let us introduce several additional functions to work with the metamodel and data sources, and to format HTML page.

- Functions to work with the metamodel:
  1. ***SourceName(entity) →string*** – returns the source name the entity belongs to.
  2. ***EntityName(entity) →string*** – returns the entity name.
  3. ***RelationList(entity) →relationList*** – returns all direct relations from the given entity to another entity (including itself) from the same data source.
  4. ***MetaRelationList(entity) →relationList*** – returns all indirect relations from the given entity to another entity from all available data sources.
  5. ***FieldList(entity, view) →fieldList*** – returns the list of all the fields of the entity.
  6. ***RelationName(relation) →string*** – returns the name (role) of the relation.
  7. ***FieldName(field) →string*** – returns the name of the field.
  8. ***RelationEntity(relation)) →entity*** – returns the entity at the opposite end of relation.

- Functions to work with data sources through wrappers:
  1. ***RecordList(entity, expr(entity)) →recordList*** – returns the list of instances (records) of the entity according to the given filtering expression.
  2. ***ValueList(record, view) →valueList*** – returns the list values of the given entity instance (record).
  3. ***Value(value) →string*** – returns the field value as character string.

- Functions to work with the list:
  1. ***List(element_1, element_2, ..., element_i) →list_1*** – returns the list of given elements and the list type *list_1* is appropriate to the element type.
  2. ***IterateList(n%list_1, function(n%)) →list_2*** – returns the list *list_2* that has as elements the results applying the given function. The function is executed with each parameter *n%* that is taken from the list *list_1* denoted by the identifier *n%* (n is any unique integer) and the list type *list_2* is appropriate to the function return type.
  3. ***Concatenate(list_1, list_2) →list_3*** – returns the concatenation of two lists with the same element type.

- Functions to format HTML page:
  1. ***SO(string)→sObject*** – creates sObject from the character string.
  2. ***StringListObject(stringList, separatorString)→sObject*** – creates sObject from the list of character strings separated by *separatorString*.
  3. ***Update(frame, entity, expr(entity), content)→updateAction*** – activates information update into the frame with the given entity, filter expression and layout.
  4. ***Clear(frame)→updateAction*** – clears the given frame .
  5. ***Navigate(frameSet , entity, expr(entity), content)→navigateAction*** – navigates to another FrameSet and update main Frame with the given entity, filter expression and layout.
  6. ***Link(sObject, navigateAction, updateActionList)→aObject*** – converts sObject into aObject and assign the navigation action and set of update actions to it. Any of action parameters may be empty.
  7. ***AO(sObject)→aObject*** – converts sObject into aObject with empty action.
  8. ***FO(aObject)→fObject*** – converts aObject into fObject without any special formatting.
  9. ***HorizontalTable(aObjectListList)→ fObject*** – creates fObject from the list of lists, this frame object is displayed as table, and internal lists are placed in rows.
  10. ***VerticalTable(aObjectListList)→fObject*** – creates fObject from the list of lists, this frame object is displayed as table, and internal lists are placed in columns.
  11. ***ListBox(aObjectList)→fObject*** – creates fObject from the list, this frame object is displayed as listbox.
  12. ***Horizontal(fObjectList)→ fObject*** – creates new fObject by arranging the given list horizontally.
  13. ***Vertical(fObjectList)→fObject*** – creates new fObject by arranging the given list vertically. Only frame objects with the type fObject may be displayed in the Web page.

5. Data Wrappers

Function ***RecordList*** must be implemented to get information from DS. The technology we use is simple, but effective. UB gets information from DS via Wrappers. This approach has the following advantages:

- It allows accessing DS via different protocols and methods – ODBC™, OLE DB™, SQL*Net™, DCOM™, COM+™, XML, HTTP.
- DS usually are made well suited for specific business tasks. DS are not primary made for data access from UB. The access to DS data usually is limited, it is allowed to execute some stored procedures to query data. Wrapper allows us to execute only authorised functions.
- Querying DS via functions allows us to have easy transfer real data from DS physical data model to logical data model that is more understandable for the user.

Information about functions is stored in the UB meta database: defined input and output fields for each function. Each input field may be mandatory or optional.

During development of the prototype, we discovered some rules for function implementation and developing conceptual model of DS.

- First rule - it is desirable to have input and output fields from one entity. It simplifies development of DS model and wrappers.
- Second rule - two approaches possible for making DS model and functions. One approach is that we already have functions, and we make conceptual data model of DS using the first rule. In case DS is a system we maintain and own, it is often possible to make functions according to conceptual data model of DS. In such a case we make conceptual data model of DS at first and

then we make data access functions according to conceptual data model and the first rule. It is helpful to make two types of functions:

1. The function gets information identifying the object from DS by some search criteria. For example, get person's PK by its name and surname (might be partial). The answer usually is a list of person's identifying information according to search criteria.
2. The function gets information about one object from one entity by its identifier. An example - get all information about the citizen by its PK.

For instance, we have two functions for the entity Citizen:

- Input data – Name, Surname (might be partial). Output data – PK, Name, Surname (full).
- Input data – PK. Output data – PK, Name, Surname, Address.

There are also 2 functions to get information about the citizen's parents and children:

- Input data – Parent PK. Output data – Children PK, Name, Surname.
- Input data – Child PK. Output data – Parents PK, Name, Surname.

There is a procedure that implements the function ***RecordList.*** This procedure gets the entity and filter expression as input and returns data from DS as output. In our implementation this procedure gets information from the meta database about functions that can be executed over entity from which we need information. In our implementation the filter expression is fields and corresponding values for these fields, e.g. PK="123456-111111". There is "brute force" algorithm that finds functions we can execute e.g. those are functions that have enough input data from the filter expression to be executed, executes these functions and returns result. There can be, of course, other implementations.

DS data access via wrappers allows connecting new DS to our system easily and quickly. We have to write a new wrapper and add information about new DS to the meta database. With some experience the writing of wrappers is easy and fast process, and there is no need to make any modification in DS.

6. Templates for Web Page Structure and Functionality

The design of FrameSet and Frames is based on template principle. With some experience the new FrameSets and Frames can be developed quickly. The design has two main steps – FrameSet structure planning and creating formulas for Frame Contents. We give some templates and ideas how the Web pages can be designed. The above given functions are used. Formulas are logically divided into several subparts only for easier understanding. Some formulas use subparts of other previously defined formulas. The example of visual presentation for each formula is given.

6.1 SIMPLE ENTITY INSTANCE PRESENTATION IN TABLE

The first column contains field names and the second – field values. The field values are retrieved according to the selected *view*.

```
A(entity, record) = VerticalTable(List(A1, A2))
A1 = IterateList(1%FieldList(entity, view), AO(SO(FieldName(1%))))
A2 = IterateList(2%ValueList(record, view), AO(SO(Value(2%))))
```

PK	12121211111
Name	Andris
Surname	Kalns
Sex	M
Address	Riga, Liepu 1-12, LV-1000

Figure 4. Example of entity instance presentation

6.2 ENTITY INSTANCE PRESENTATION AS TEXT

Instance field values are concatenated according to the order of the selected *view*.

```
B(record) = FO(AO(SO(StringListObject(B1, " "))))
B1 = IterateList(3%ValueList(record, view), Value(3%))
```

12121211111 Andris Kalns M Riga, Liepu 1-12, LV-1000

Figure 5. Example of entity instance presentation as text

6.3 ENTITY RELATIONS PRESENTATION IN VERTICAL LIST

Each relation is represented as relation name concatenated with entity name at the opposite relation end.

```
C(entity) = Vertical(IterateList(4%RelationList(entity), C1))
C1 = Horizontal(List(C2, FO(AO(SO(" "))), C3))
C2 = FO(AO(SO(RelationName(4%))))
C3 = FO(AO(SO(EntityName(RelationEntity(4%)))))
```

Has Passport
Has Parents Citizen
Has Children Citizen

Figure 6. Example of relations presentation

6.4 ALL RELATION PRESENTATION IN TABLE

The data about all relations (relation name, entity name and data source) are placed in table with headings.

```
D(entity, expr(entity))=HorizontalTable(Concatenate(D1,D2))
D1 = AO(StringListObject("Relation", "Entity name", "Data source"))
D2 = IterateList(5%MetaRelationList(entity),List(D3, D4, D5))
D3 = AO(SO(RelationName(5%)))
D4 = AO(SO(EntityName(RelationEntity(5%))))
D5 = AO(SO(SourceName(RelationEntity(5%))))
```

Relation	Entity name	Data source
Is	Citizen	Register of Residents
Has	Passport	Register of Residents
Has Parents	Citizen	Register of Residents
Has Children	Citizen	Register of Residents
Is	Car Owner	Register of Motor vehicles
Owns	Car	Register of Motor vehicles

Figure 7. Example of relation presentation

6.5 AN EXAMPLE OF FRAMESET

Let us look how a FrameSet can be built. Let us assume FrameSet *FRS_1* with four Frames – *FR_1, FR_2, FR_3, FR_4*. See Figure 8.

FR_1 (upper left) – to list instances of entity,
FR_2 (upper right) – to show details of fixed instance in FR_1,
FR_3 (lower left) – to list all relations to other entities in all data sources,
FR_4 (lower right) – to show details of another related entity instances of FR_2.

At first let us create three presentations or Contents (E, F, G) for viewing entities. We use formulas created before in this paper.

- Content formula E() for Frame FR_4 (from FR_4 we can update all Frames in FRS_1)

```
E(entity, expr(entity)) = Vertical(E1, E5)
E1 = Horizontal(List(FO(E2), FO(AO(SO(" "))), FO(AO(SO(SourceName(entity))))))
E2 = Link(SO(EntityName(entity)), E3, E4)
E3 = Navigate("FRS_1", entity, expr(entity), "")
E4 = List(Clear("FR_2"), Update("FR_3", entity, expr(entity), """), Clear(FR_4))
E5 = Vertical(IteateList(6%RecordList(entity, expr(entity)), A(entity, 6%)))
```

- Content formula F() and G() for Frame FR_2 (from FR_2 we can update this frame or update FR_4)

```
F(entity, expr(entity)) = Vertical(H, FO(AO(SO(" "))), E5)
G(entity, expr(entity)) = Vertical(H, FO(AO(SO(" "))), Vertical(E5, G1))
G1 = C(entity), where C3 is substitute with G2 in all places (we have added the action)
G2 = FO(Link(SO(EntityName(RelationEntity(4%))), NULL, G3))
G3 = List(Update("FR_4", RelationEntity(4%), expr(RelationEntity(4%)), "E"))
H = ListBox(List(Link("Presentation F", NULL, H1), Link("Presentation G", NULL, H2)))
H1 = Update("FR_2", entity, expr(entity), "F")
H2 = Update("FR_2", entity, expr(entity), "G")
```

- Content formula I() for Frame FR_1 (from FR_1 we can update FR_2, FR3, FR_4)

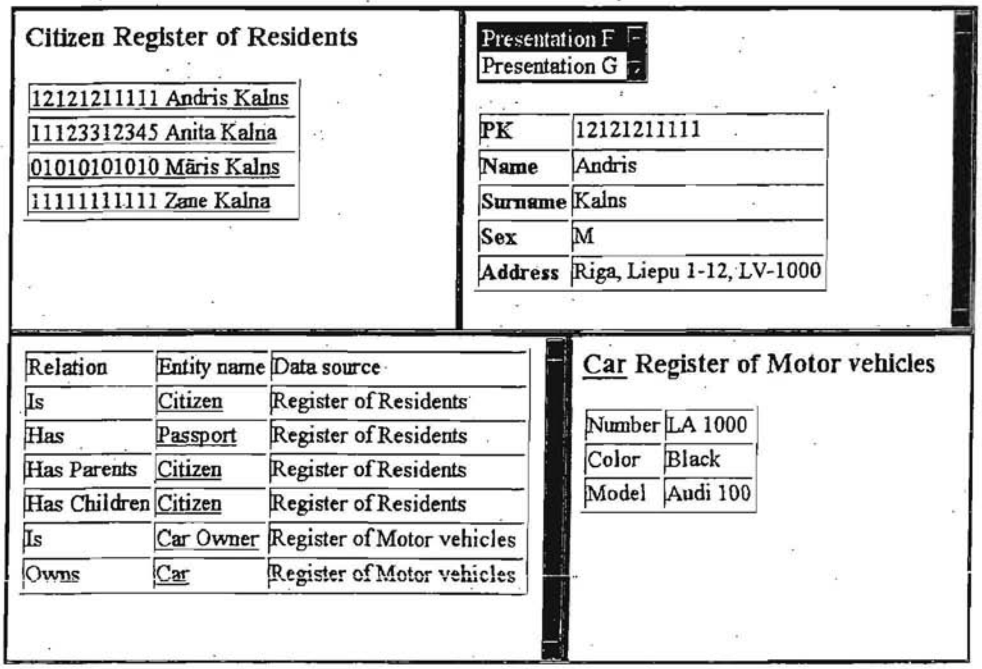
```
I(entity, expr(entity)) = Vertical(I1, I2)
I1 = Horizontal(List(FO(EntityName(entity)), FO(AO(SO(" "))), FO(AO(SO(SourceName(entity))))))
I2 = HorizontalTable(IterateList(7%RecordList,Link(B(7%), NULL, I3))
I3 = List(Update("FR_2", entity, expr(entity) and expr(7%), "F"),
Update("FR_3", entity, expr(entity) and expr(7%), ""), Clear("FR_4"))
```

- Content formula J() for Frame FR_3 (from FR_3 we can update FR_4)

J(entity, expr(entity)) = D(entity, expr(entity)), *where D4 is substitute with J1 in all places (we have add the action)*

J1 = Link(SO(EntityName(RelationEntity(5%))), NULL, J2)

J2 = List(Update("FR_4", RelationEntity(5%), expr(RelationEntity(5%)), "E"))



Citizen Register of Residents

12121211111 Andris Kalns
11123312345 Anita Kalna
01010101010 Māris Kalns
11111111111 Zane Kalna

PK	12121211111
Name	Andris
Surname	Kalns
Sex	M
Address	Riga, Liepu 1-12, LV-1000

Relation	Entity name	Data source
Is	Citizen	Register of Residents
Has	Passport	Register of Residents
Has Parents	Citizen	Register of Residents
Has Children	Citizen	Register of Residents
Is	Car Owner	Register of Motor vehicles
Owns	Car	Register of Motor vehicles

Car Register of Motor vehicles

Number	LA 1000
Color	Black
Model	Audi 100

Figure 8. Example of WWW page

7. Conclusions and Future Directions

The prototype of the UB is made during developing Megasystem and Communication Server. Four state significance registers test databases are connected to the UB for testing purposes. Two of them use Oracle™ as DBMS, other two use Microsoft SQL Server™.

The UB prototype shows the effectiveness of our approach and is being initiated as first version of the real system at present time.

Our approach differs from other systems by several aspects:

- We have developed simple universal user-end that still allows us to show to users information in many different ways. We achieved this goal by implementing user-end using formal formulas.
- UB operates using logical models of DS. Related objects from these models are bound together with base classes that do not belong to any particular DS.
- We transfer physical model of DS to our internal logical representation which is much more comfortable for end-user. We do it by using of data wrappers.
- Our approach allows us to maintenance system and to connect new DS or modify existing one without interrupting operation of Communication server.

There are many aspects that are very important in real life application, but not covered in this article – security, user authorisation, logging, query cost calculation. All these features also are incorporated in the UB. The UB is useful in many large enterprises having many autonomous data sources as a browser for these systems with integrated view.

Future direction of our work is to develop a query processor that can take as input SQL-like query and return as output the result queried from multiple data sources. Other directions of future work – to make Communication server available not only from WWW browsers, but also from custom programs using XML to query data and return answers.

References

1. Arnicans, G. Application generation for the simple database browser based on the ER diagram. *Proceedings of the Third International Baltic Workshop Databases and Information Systems*, Riga, 1998, pp. 198-209.
2. Arnicans, G., Bicevskis, J., Karnitis, G. The concept of setting up a communications server. *Abstracts of Papers of 3rd International Conference Information Technologies and Telecommunications in the Baltic States*, 1999, pp. 48-57.
3. Haas, L. M., Miller, R. J., Niswonger, B., Tork Roth, M., Schwarz, P. M., Wimmers, E. L. Transforming heterogeneous data with database middleware: beyond integration. *Data Engineering Bulletin*, 1999.
4. Hammer, J., Garcia-Molina, H., Ireland, K., Papakonstantinou, Y., Ullman, J., Widom, J. Information translation, mediation, and Mosaic-based browsing in the TSIMMIS system. *Proceedings of ACM SIGMOD International Conference on Management of Data*, 1995, Project Demonstration.
5. *Megasystem - Integrated State Significance Information System*. http://www.mega.lv.
6. Singh, N. Unifying heterogeneous information models. *Communications of the ACM*, **41**(5), 1998, pp. 37-44.
7. Tomasic, A., Amouroux, R., Bonnet, P., Kapitskaia, O., Naacke, H., and Raschid, L. The distributed information search component (disco) and the World Wide Web. *Proceedings of ACM SIGMOD International Conference on Management of Data*, Tuscon, Arizona, 1997, Prototype Demonstration.

Zbornik D 5. mednarodne multi-konference
Proceedings D of the 5th International Multi-Conference

INFORMACIJSKA DRUŽBA IS'2002
INFORMATION SOCIETY IS'2002

Vzgoja in izobraževanje v informacijski družbi
Education in Information Society
Uredili / Edited by
Vladislav Rajkovič
Tanja Urbančič
Mojca Bernik

Razvoj in prenovitev informacijskih sistemov
Development and Reengineering of Information Systems
Uredil / Edited by
Ivan Rozman

Sodelovanje in informacijska družba
Collaboration and Information Society
Uredila / Edited by
Marjan Heričko
Matjaž B. Jurič

Upravljanje v informacijski družbi
Management in Information Society
Uredil / Edited by
Cene Bavec

SMART INTEGRATED MEGA-SYSTEM AS A BASIS FOR E-GOVERNANCE

Guntis Arnicans, Prof. Janis Bicevskis, Prof. Edvins Karnitis, Girts Karnitis
Department of Computer Science
University of Latvia
Raina blv. 19, LV-1050 Riga, Latvia
Tel: +371 7228226; fax: +371 7820153
e-mail: garnican@lanet.lv, bics@di.lv,
Edvins.Karnitis@sprk.gov.lv, girts@di.lv

ABSTRACT

Principles and basic informatics tools for modernization of governance in Latvia are described in the paper. Ensuring access to well-developed information services for everyone should be envisaged as a tool for democratic development and functioning of society. Development of ICT directly affects political/governance procedures also, usage and management of public sector information become the base for all governance procedures. Interconnection and interoperation of public information systems, development of smart Mega-system, integration of national information resources of Latvia in Transeuropean telematic networks become components of unified process.

UNIVERSAL INFORMATION SERVICE

Public sector information, its usage and management is the real base for all governance (both G2G and G2C) procedures [1]. It ranks high among various types of information by its amount and significance, the public sector is the most important (and very often the single) collector and producer of information content.

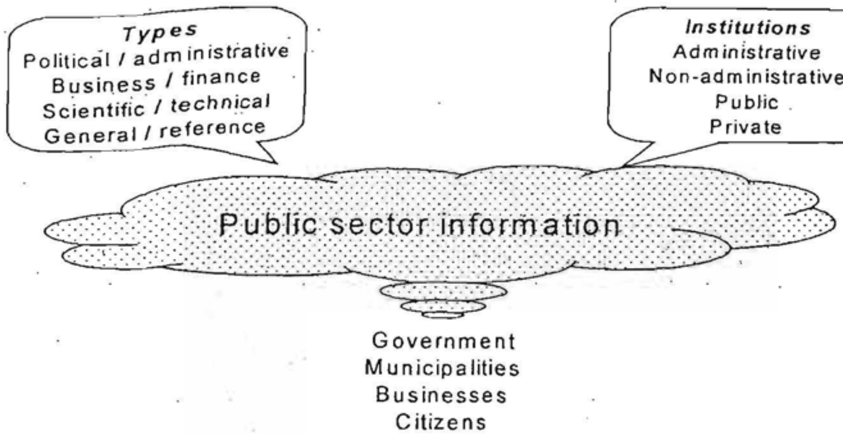


Figure 1. *Public sector information*

Processing and usage of information that can be deemed as being the *information of national significance* is the most significant component [2]. Within this term we shall mean various information that is necessary for state or regional administration, for the development of national economy, for management of financial, educational and social processes. The main subjects of this type of information are real estate and movable property, legal and private persons, substantial for the country processes (legislation, statistics, finances, health care, etc.).
In order to ensure wide and active usage of public sector information, Latvia's approach is developed by the National Program *Informatics* and several other more detailed conceptual documents [3, 4]. In principle it is similar to EU concept, but more extended and methodologically more advanced. The Program implies under the *universal information service* a general access to information services for everybody in an order as set by normative acts without any discrimination, a long-time service of a defined quality at an affordable price.

Ensuring technical access to telecommunications and data transmission networks is well known as the *universal telecommunications service*. Exactly inclusion of the Internet access shows clear forward-looking vision for Latvia: developed data transmission services, convergence of all kinds of information and communications services.

In addition universal information service means ensured access to all types of public sector information (and first of all to information of national significance). An electronic delivery of information is envisaged (on-line or broadcasting, magnetic, optical or another carrier, etc.). Services can be provided on demand or to be interactive. A number of different information services are components of the universal information service -- full set of business and finance information services, availability of data collected in national and municipality information systems (IS), library and reference information services, reference and entertainment services, etc.

A number of bounded up and interdependent subprograms of the National Program *Informatics* are directed to development of the universal information service. The Program includes both macro level strategy (policy of the development) and micro level measures (a number of applications and projects).

Although there are number of common principles in provision of the public sector information to all end-users, many important differences exist too. General access of any citizen to the public sector information (G2C) differs from utilization of the information for state governance (G2G), there are different information compositions, level of confidentiality, demands for completeness, correctness, updating of information. Therefore side by side with common methodological principles, different approaches are used for development of information processing and provision systems and services.

THE MEGA-SYSTEM: ADVANCED TOOL FOR ADMINISTRATION OF THE COUNTRY

Creation of corporative sectoral IS for interconnecting related institutions on national and international scale (e.g., EU Programme IDA [5]) are important activities that are going on in number of countries. The next step – interoperability of IS, interchange of data between sectoral information systems/networks and handling requests that require processing data from various IS.

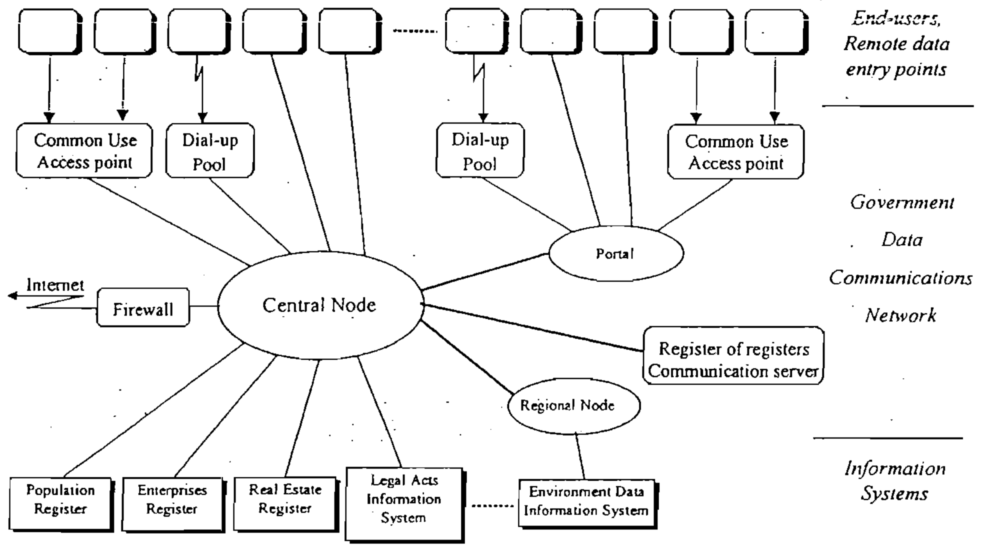


Figure 2. *The Mega-system*

Because a drastic improvement of quality and full interoperability of all IS are vital for the development of e-Government, all set of public IS in Latvia is being developed as a logically unified and technologically distributed information processing Mega-system with a common data field as well as unified user's interface, access principles and authorization procedures. Several basic principles are implemented into the Mega-system:

- the Mega-system is a set of separately functioning harmonized IS;
- all objects of national significance (persons, cars, real estate, legal entities, etc.) must be registered in IS;
- data must be fixed electronically in the place where they are originated; each object is allowed to be registered only in one of primary registers; the source of the information on the object as well as responsibility for its quality must be defined;
- all IS must use information on particular object from corresponding primary register, it is not allowed to duplicate data entry; it is allowed only to keep copy of the data from the basic register for improvement of access;

- the registration certificate of any object (passport, certificate of legal entity, etc.) must be issued only as the result registration of objects; it is not allowed to repeat manual information input from registration certificate or othe documents.

Creation of the Mega-system is not only technological decision, in fact it means solving of number of various informativ legal, organizational problems first of all, among them:

- to analyze existing data flows, to formulate functions of the Mega-system and to distribute them among IS, t formulate demands on systems and their data structure;
- to define subjects of various IS and the amount of stored information, as well as institutions that are responsible for th collection, processing and distribution of data;
- to define a unified user interface, access principles and authorization procedures;
- to elaborate several intercompatible informative models for implementation by local authorities;
- to ensure data quality and security as well as interoperability with EU IS; to elaborate a methodology for data verification;
- to determine the principles of electronic archives.

The integration of primary registers has realized. In addition to various IS the Mega-system includes a portal as a gateway information resources, a *register of registers* for collection and distribution of meta information (formal and informal description of objects, data models, data flows, etc.) on all components of the Mega-system as well as *communication server* common central access point to information resources of the Mega-system. Other IS are being attached to the developed centr core of the Mega-system gradually as far as they are prepared. For this purpose development of the IS is being continued, and primary data entry is taking place in many systems, even as other data are already being used.

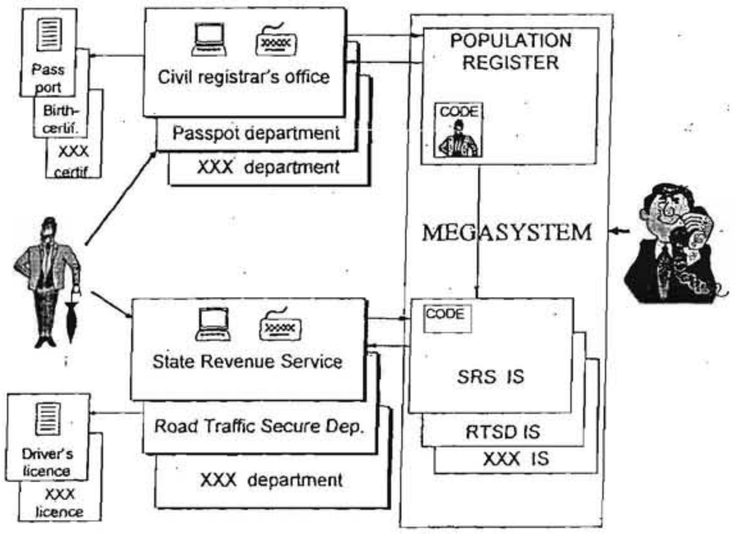


Figure 3. *The Mega-system: data flows*

All end-systems (various IS, their remote data entry and access points, end-users of information) are interconnected through high speed *Government Data Communications Network*, that is an essential communications element for development of th Mega-system. This Network must provide operative and reliable interoperability of all interconnected systems, therefore requirements to the Network include:

- high security and reliability level -- there must be uninterrupted action time, undistorted data transmission, a guarante of several levels of confidentiality and security of information;
- high speed data transmission, some of real time systems need guaranteed channel capacity;
- presence of a common gateway to public data transmission network (the Internet environment) which contains reliable security system.

On the basis of the Mega-system during following years the G2G usage of traditional paper documents will be changed to usage of data base files, when an event or fact is assured not by a paper document, but by a record in a database. Each record in the IS will become legally approved document.

COMMUNICATION SERVER - A CENTRAL ACCESS POINT TO INFORMATION RESOURCES

Communication server is a set of hardware and software that provides a universal resource for information exchange amon various information systems and other G2G transactions within the Mega-system as well as allows a wide range of users to eceive information from a variety of public IS through a single contact point. The need to establish a communication server ecomes apparent when it is necessary to interconnect a lot of IS and to retrieve information from number of systems in unified ay.

nformation becomes available on the Network, but users (most of them are employees of administrative structures) shouldn't ave no knowledge about the technical details of information storage. There is an obvious need for a universal solution, and hat is where the communication server comes in. The main requirement for the communication server is that it must allow sers to formulate their information requests in a simple way and to receive responses to those requests without necessity to nderstand the technical aspects of the process and knowledge on distribution of data objects among the IS (by interconnection ith the register of registers).

or these purposes the communication server identifies users, authorizes the use of the respective data, manages users rights, ulfills requests that involves usage of several information sources. It allows users to learn where information is being stored nd what kind of information it is, as well as to request and to receive information from various registers without any need for -depth knowledge about the technical aspects of its storage. National ID card is a crucial part to implement person's lentification in communication process, there might be desirable to implement not only ID cards for citizens but also ID card r legal entities.

he communications server is an Internet resource point. Users of the server can access it via various protocols – HTTP, XML, ORBA, DCOM, SMTP and FTP. The server provides users an opportunity to find out where information is stored and what inds of data are available, to request and receive information from various IS without studying their structure. Because users ay have access to confidential and sensitive information, they are identified with certificates, and all data transmissions are oded.

HE MEGA-SYSTEM AS A TOOL FOR INTERNATIONAL INTEROPARABILITY

ultilingual and multicultural Europe has a particular interest in international cooperation and united activities, because ndividual national markets for information services are mostly small and ineffective. Competition on the global scale also equires a common European strategy and intercoordinated development. For this reason, the European Commission sees ollaboration among all European states – including the associated Central and Eastern European Countries as an important omponent of integrated information policy [6]. Integration of national information resources of Latvia in Transeuropean formation systems and networks is going on.

n order to further develop national IS of Estonia, Latvia and Lithuania, to prepare their future informative and technological onnection with European IS, prime ministers of the Baltic States in 1997 made the decision to create a *Baltic Governmental ata Communications Network*. The Baltic Network is considered as expanding of the Mega-system and the Government Data ommunications Network on international scale.

he concept envisages to develop the Baltic Network as a pilot stage for integration of national IS in Transeuropean systems. ll principles of the Mega-system structure and user access, data structure and interfaces are being developed so as to allow for tegration of the Latvia's IS into the Transeuropean corporate telematic networks. International expansion of the Mega-system volves the creation of resources-points and interfaces for international interconnection of IS, while maintaining the basic inciples of the Mega-system.

entralized resources of the Mega-system – register of registers and communication server support both local and international formation services. Data on international resources, that are available for Latvia's end-users, are included in the register of gisters. Common communication server can be used for authorization and access to national information resources for foreign d-users equally with local users.

ch approach corresponds to basic principles of the IDA Programme, it is the basis for successful participation of Latvia in the ogramme. The Programme consists of one central network connecting countries and local networks for each country, it quest one central access point for each country. The Mega-system serves as Latvia's local network and communication server ves as Latvia's single access point.

number of national IS are already participating in activities of international systems, they are pioneers among CEES at esent. The Enterprise Register has been joined to the European Business Register in order to support international financial ations and investment processes, as well as business cooperation and foreign trade. Vehicles Register has already been nnected to European Car Register. A number of another national IS are participating in the activities of international systems.

GUIREMENTS TO THE NATIONAL PORTAL: TO BRING ADMINISTRATION CLOSER TO CITIZENS AND SINESSES

via has adopted the EU recommended approach to the *level of electronization* of all G2B, G2C and G2G services identifies different levels [7]:

- level 1: the provision of information – data on services are available on the Internet;
- level 2: interactivity – forms and documents can be downloaded;

- level 3: multi-directional interactivity – client authorization is enabled, and forms and information can be submitted electronically;
- level 4: processing of transactions – full handling services, including the taking relevant decisions and the making payments.

National portals in many countries ensure the first level, although the quality of the information that is provided is not always guaranteed. The possibility to download forms and documents is also fairly common (e.g., United States, France, Estonia) because this does not demand excessively complicated technologies.
There are different situations at the third and fourth levels. There are only very few countries that have resolved the client authorization problem, national laws on digital signatures and electronic documents have not been adopted yet. E-transactions are most commonly offered through Great Britain's *UK Online* and Singapore's *eCitizen* programs.
Usually national portals contain more than one way of looking of stored information. The following organizational types of information can be identified:

- around everyday themes (UKOnline, Danmark.dk, eCitizen Singapore);
- by regions (Danmark.dk);
- by sectors in a catalogue-type principle (in nearly all national portals);
- around the country's administrative structure (Bundesregierung, FirstGov, etc.);
- separately for citizens, businesses and foreigners (Canada).

In general e-governance means that the government shifts to a more advanced model of functioning. There are changes in the structure of the government and in relations among government institutions. The portal must be seen as an instrument in e-government, but by no means it cannot be seen as the tool that actually implements e-government. The Mega-system (including the portal as an interface) will provide government *back office* functionality.
The Latvia's national portal has been developed as a unified access point for information services what are provided by public agencies and institutions [8]. The first version of portal has been developed and provides first level services. The first and second levels services is provided directly by portal, while the third and fourth levels – in cooperation with the communication server.
The Latvia's national portal is being established defining three groups of individuals with different needs – citizens, business people and officials. A system have reciprocal links for all of them. It is important also to define the links between all categories of users as well as various levels of officials. Several organizational types of information are being developed at present [9].
As a result of analysis of the Latvia's situation and the possible demands of users, the basic principles have been defined:

- decentralization of information;
- the national portal is a portal of links;
- high quality (completeness, correctness, actuality) of the content;
- access to the public IS (in cooperation with the register of registers and the communication server);
- opportunities for contacts with officials;
- possibility to download forms and documents;
- autentification of users and personalization of content;
- availability of services in several languages (Latvian is mandatory);
- development of tools for support and maintenance of portal.

The development of the Latvia's portal is going on, the first version is available at the WWW [10]. The process of fulfilling portal with actual data is in progress at this moment.

OPENNESS AND TRANSPARENCY - THE MODEL OF E-GOVERNANCE

Rapid evolution of ICT make possible to change general requirements to the governance principles and procedures in line with development of e-democracy.
It is very hard task to control state institutions nowadays. Usage of ICT will help to introduce number of important for democratic society principles:

- openness -- quantity and quality of information that any public institution provides to society, especially on-line;
- transparency -- possibility to track how the institution acts and how it is making decisions;
- interactivity -- possibilities that citizens and businesses have to contact with any level institution, its readiness to the fast reaction and dialog; possibility to offer opinions as well as to influent decisions;
- control, audit, inspection – public possibility to monitor and to control institution from outside.

Governance will become fully democratic only if every member of society will have possibility to get easy detailed information on administrative processes that are important for welfare of citizens and efficiency of businesses [11]. Citizens or officials initiate any process. While developing the Mega-system, it becomes possible to define and identify set of processes any person

SEMANTICS FOR MANAGING SYSTEMS IN HETEROGENEOUS AND DISTRIBUTED ENVIRONMENT

Guntis Arnicans and Girts Karnitis
University of Latvia

Abstract: The problem of legacy systems collaboration is being solved. Particularly we look at the collaboration as workflow in a distributed and heterogeneous environment. Attention is paid to the description of semantics for workflow process definition languages. There are many solutions how semantics can be decomposed into logical fragments, but the problem of obtaining reusable components that are easy to compile into desired specific semantics still remains. We evolve the division of semantics by semantic aspects whose description is based on abstract data types (pre-built components) and connectors (meta-programs to produce the glue code) between them. This paper offers a way in which semantic aspects are linked with the intermediate representation of a program, and performing of semantics is provided. We mix together various semantics aspects to get a desirable semantics.

Keywords: workflow, programming language specifications, semantics, interpreter, compiler, reusable components, domain specific languages, tool generation.

1. Introduction

Nowadays new technologies are emerging in the government sector allowing to speak about the e-Government. The processes are one of the core components of e-Government [1]. We stated that there is practically no automation of processes in the governmental institutions that organize collaboration between legacy systems among various organizations and institutions. Document flows are manual, or by email. The automated workflows have to be introduced to make the document turnover faster and to improve the provided service for citizens.

In [2] workflow is defined as "the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules."

The workflow management system is defined as "a system that defines, creates and manages the execution of workflows through the use of software, running on one or more workflow engines which is able to interpret the process definition, interact with workflow participants and where required, invoke the use of IT tools and applications."

In the latest years many researchers and developers have paid attention to a problem how to organize the collaboration between legacy systems, and the exploitation of workflow is one of the most popular solutions [3,4]. Various workflow process definition languages have been created which can be considered as domain specific languages.

The workflow implementations commonly are based on one fixed semantics, like most of the programming languages. We are interested in various semantics for a particular workflow, for example, common workflow semantics, a statistical data gathering semantics, a semantics for debugging and simulating purposes or its composition, therefore we need not only a compiler or an interpreter, but the necessity for specific supporting tools becomes a burning question due to demands for high software quality. An interesting topic is changing of semantics for active instance of workflow on the fly.

In our approach semantics are connected to syntax elements via semantic connectors that naturally allow linking legacy systems into collaborative workflow and allow to define or to execute multiple different semantics simultaneously. Actually, each semantic implementation is a tool, similarly to the principle in [5]. We present fragments of semantic description for simple programming language to demonstrate usefulness of this approach for a wide class of programming languages, and ideas how to implement a simple workflow description language.

2. Implementation of Domain-Specific Language

According to Kinnersley's investigation [6], there were more than 2000 exploited languages in 1995, and most of them were classified as domain-specific language (DSL). Together with the growth of DSL many implementations and maintenance problems arise (e.g. [7] analysis of common problems and large annotated bibliography; [8] particular languages and problems). Unfortunately, formal semantics descriptions lose their position because of a weak support to solve practical problems [9, 10,11].

Like natural language, the programming language definition consists of three components or aspects [12, 13]: syntax deals with questions of superficial form of a language, semantics deals with the underlying meaning of a language, pragmatics deals with the practical use of a language. The syntax and semantics of a language can be formalized, and both formalizations together form formal specification of a programming language.

The formalisms for dealing with the syntax aspect of a programming language are well developed. The theory of scanning, parsing and attribute analysis provides not only means to perform syntactical analysis, but to generate a whole compiler as well. There is a lot of problems with practical use of semantics formalisms. Recently the criticism of classical formalism has arisen from the difficulty of using formal methods. The main problem to use widely in practice the formal specifications of programming languages is that specifications become too complex, too abstruse to manage them, often it is impossible to express all needs, and in the end – who verifies and proves the correctness of the specification?

Summarizing the best practices in compiler construction we can declare that most of commercial compilers (interpreters or other tools that deal with programs) are written without using any formalisms or only the first phases (scanning and parsing) to exploit some formalisms [10].

Let us look at the language description again, try to divide it into smaller parts and see, what we can obtain from that. Traditionally the first decision is to separate syntax from semantics, and semantics consists of two parts: static semantics and dynamic (run-time) semantics. But we should divide syntax and semantics further, eliminate reusable components and provide a mechanism to stick all things together.

Syntax components are more or less visible: basic elements (for instance, terminals and nonterminals, if we parse program) connected with some relations (for instance, edges in the parse tree or abstract syntax tree).

To divide semantics into pieces we offer to split it by semantic aspects. Here are some examples of semantic aspects: program control flow management (e.g. loops, conditional branching), execution of commands or statements (e.g. basic operations, assigning), dealing with symbols (e.g. variables, constants), environment management (e.g. scopes of visibility), pretty printing of program, dynamic accounting of statistic, symbolic execution, specific program instrumentation, etc.

We are interested in any formalism to deal with syntax, because we want to make intermediate representation (IR) of program or structured information. The situation is clear what refers to conventional programming languages. But our goal is a workflow implementation, and we have to take into account other languages, for instance, diagrammatic visual languages (e.g. Petri nets, E-R diagrams, Statecharts) and the state-of-art in this field (e.g. [14]).

Our approach has borrowed some principles from attribute grammars, for instance, the ways to link semantics with syntax [11], modular decomposition and reuse of specification [15], distributed computing in real time (e.g. Communicating Timed AG [16]).

We found out that many formalisms of semantics use abstract data types (ADT). ADT is a collection of data type and value definitions and operations on those definitions which behave as primitive data type. This software design approach decomposes problem into components by identifying the public interface and private implementation. A typical example is Stack, Queue, Symbol table [17, 13].

Recently one of the simple and popular methods to build some simple tool for a programming language has become parse and traverse principle [18] that means to build intermediate representation (IR) of program or information. traverse IR and make appropriate computations at each node. This method is similar to the Visitor Pattern [19]. Other useful patterns are also developed [20, 21]. Besides, there exist also nontraditional traversal strategies [e.g. 22]. Many solutions can be obtained from Component collaboration [e.g. 23].

We have taken into account our experience in building prototypes of multi-language interpreter [24]. A Multi-Language Interpreter (MLI) is a program which receives source language syntax, source language semantics and a program written in the source language, and then it performs the operations on the basis of the relevant semantics.

3. Principles of Semantic Definition and Implementation

3.1 Runtime Principles

Let us assume that we have fixed some formalism to describe the syntax of our language (e.g. BNF). Now we can define the language syntax and develop a language parser (e.g. by using Lex/Yacc). The parser creates intermediate representation (IR) of program (e.g. Parse tree), and IR is based on a desirable structure and contains any needed information about the syntax (e.g. node type (nonterminal), name (name of nonterminal), value (terminal value) etc.).

To perform semantics at runtime we choose a principle of parse and traverse. The Traverser that realizes our chosen traversing strategy (e.g. leftdepth tree traversing) has to be created for our IR representation. The computations that have to be done at each node visited by the Traverser are defined in Semantic Connectors (SC). They use predefined data structures with operations to establish the cooperation between Legacy Systems (LS) (Figure 1).

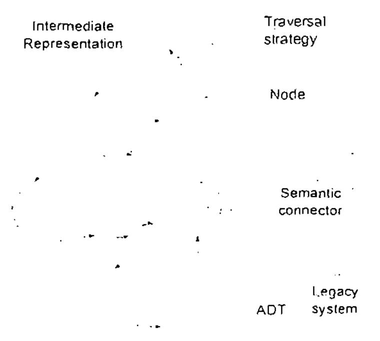


Figure 1. Runtime correspondence between syntax and semantics

Actually, most of work performing semantics is done via operations over various Abstract Data Types (ADT). In such a way we hide most of implementation details and concentrate mainly on logic of semantic aspect. The consequence of this approach is that we can choose the best physical implementation of ADT for the given task. For instance, Stack can be implemented in a contiguous memory or in a linked memory. The instances of ADT can be distributed objects in a heterogeneous computing network.

A concept of a *semantic connector* or simply *connector* is introduced to connect the instances of ADT and LS in a desirable environment. Connector is a meta-program that introduces a concrete communication connection into a set of components, i.e. it generates the adaptation and communication glue code for a specific connection. This concept is adopted from similar problem: how to connect pre-built components in a distributed and heterogeneous environment [25].

3.2 Semantic Definition Principles

Similarly to patterns in [11] we choose a correspondence *Nonterminal with visiting aspect* = *Semantic connector* to establish the relationship between syntax and semantics. *Nonterminal with visiting aspect* means that we distinguish computations performed at nonterminal node considering an aspect of node visiting (e.g. PreVisit or PostVisit). Any connector can see any instance of ADT or LS of the semantic aspect it (connector) belongs to.

The main problem is to find a good way to define semantics and obtain semantic connectors for the definition. After exploring various approaches how semantics can be described and organized, we suppose that semantic aspect is a good basic component for constructing whole semantics according to our goals. The conceptual components of a semantics description and relationships with other concepts are represented in Figure 2.

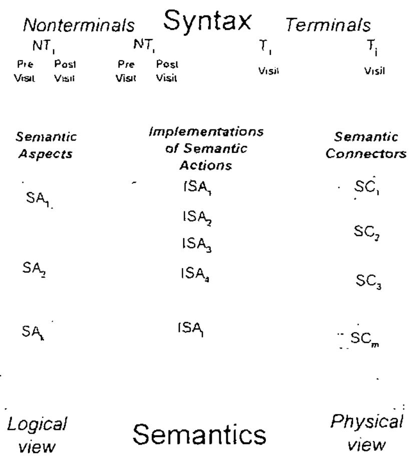
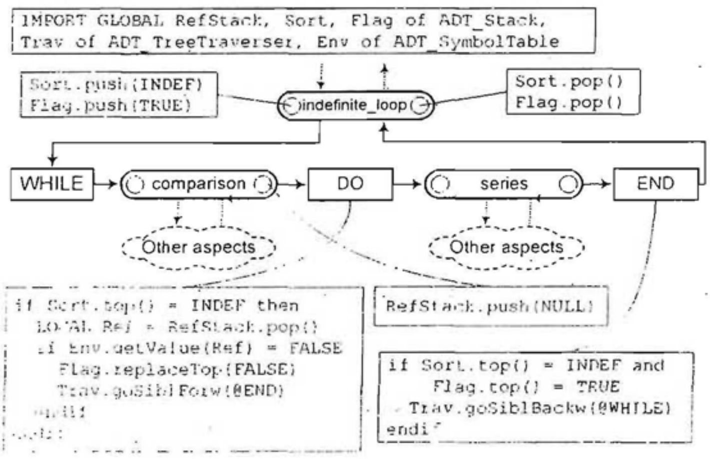


Figure 2. The conceptual schema of semantics.

Semantics can be observed from two different sides – a logical definition view and a physical runtime view. From the logical viewpoint semantics consists of Semantic Aspects (SA). The SA states what syntax elements (terminals; nonterminals) are involved, and what actions have to be performed traversing internal representation and visiting the corresponding node to implement the semantic aspect. Let us define the concept *Semantic Action* that denotes the action performed to implement SA while visiting a corresponding node, and the concept *Implementation of Semantic Action* (ISA) that denotes a meta program which implements the semantic action. In our example SA_1 involves nonterminal NT_1 and terminal T_1, and ISA_1 is performed while *previsiting* NT_1 and ISA_2 – while *visiting* T_1.

The example of semantic aspect INDEFINITE LOOP is given in Figure 3. There are various nonterminals and terminals organized by some syntax description. The arrows represent a traversal strategy. The small circles represent the semantic actions, and the rectangles connected to the circles contain the implementation of semantic action (meta program). A left circle into nonterminal stands for PreVisit, and a right circle - for PostVisit. All used abstract data types (ADT) are defined within semantic aspect. Another example of semantic aspect is given in Figure 4.



```
IMPORT GLOBAL RefStack, Sort, Flag of ADT_Stack,
Trav of ADT_TreeTraverser, Env of ADT_SymbolTable
```

```
Sort.push(INDEF)
Flag.push(TRUE)
```

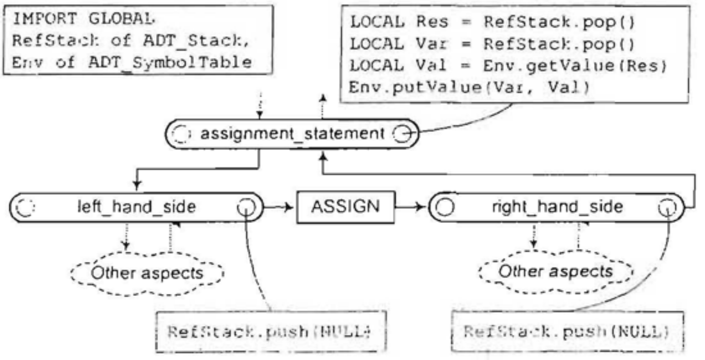
```
Sort.pop()
Flag.pop()
```

```
if Sort.top() = INDEF then
  LOCAL Ref = RefStack.pop()
  if Env.getValue(Ref) = FALSE
    Flag.replaceTop(FALSE)
    Trav.goSiblForw(@END)
  endif
endif
```

```
RefStack.push(NULL)
```

```
if Sort.top() = INDEF and
     Flag.top() = TRUE
  Trav.goSiblBackw(@WHILE)
endif
```

Figure 3. Semantic aspect INDEFINITE LOOP. It "goes through" series and back to WHILE until the comparison sets NULL reference or reference with value FALSE



```
IMPORT GLOBAL
RefStack of ADT_Stack,
Env of ADT_SymbolTable
```

```
LOCAL Res = RefStack.pop()
LOCAL Var = RefStack.pop()
LOCAL Val = Env.getValue(Res)
Env.putValue(Var, Val)
```

```
RefStack.push(NULL)
```

```
RefStack.push(NULL)
```

Figure 4. Semantic aspect ASSIGNMENT. It takes reference to a variable and reference to a value from the stack, and assigns the value to the variable. Pushing of references is simulated, real references will be pushed by other aspects and simulating will be excluded

Let us look at the physical view. The semantic connector contains all corresponding semantic actions having to be executed while visiting syntax element (IR node). For instance, visiting any node with the name T_1 we have to execute the semantic connector SC_2 that contains the implementation of the semantic action ISA_2. Similarly, SC_1 is some composition of ISA_1 and ISA_4.

3.3 Obtaining Semantics from Semantic Aspects

From the logical point of view semantics is a composition of semantic aspects with concrete linking to instances of abstract data types, legacy systems and traverser that performs a traversal strategy over fixed intermediate representation of program or structured information. We cannot simply stick all SA together risking to get senseless semantics. A composition of semantic aspects is operations over a set of implementations of semantic actions with the aim to get one set of connectors that correspond to the new mixed semantic aspect (Table 1).

Table 1. Fragment of semantics description for simple imperative language

```
 compatible with ir_type ParseTree, traverser_type ParseTreeTraverser
 ...
 syntax elements (program, expression, VARIABLE, ...)
 semantic actions (<PROGRAM> program PreVisit {ENV.prepareProgEnv()},
            <PROGRAM> program PostVisit {...}, ...)
 ...
 global Trav of ADT_TreeTraverser, Env of ADT_SymbolTable
 create DataStack, OperatorStack, CanCreateVar, LoopSortStack, LoopCounterStack,
 LoopFlagStack, IfFlagStack of ADT_Stack, InputFile, OutputFile of ADT_FILE
 ...
 compose aspect <COMPOSED SA>  // composes semantic aspects from predefined aspects
 (<PROGRAM>)
 append (<ELEMENT>
  replace RefStack with DataStack // replaces stack for collaborating work
  rename INTEGER Visit with CONSTANT Visit) // renames according to PAM syntax
 append (<ASSIGNMENT>
  replace RefStack with DataStack
  rename left_hand_side PostVisit with VARIABLE Visit,
      right_hand_side PostVisit with expression PostVisit
  ignore left_hand_size PostVisit)    // ignore pushing of NULL reference
 append (<INDEFINITE LOOP>
  replace RefStack with DataStack,
   Sort with LoopSortStack, Flag with LoopFlagStack)
 ...
 end compose aspect
 ... // other aspect are defined and composed together
 link for <COMPOSED SA> Trav to TreeTraverser, Env to SymbolTable
 use aspect <COMPOSED SA> with traverser TreeTraverser
```

The obtaining of semantics for the fixed syntax is achieved in several steps: 1) select predefined semantic aspects or define new ones for desired semantics, 2) rename syntax elements and traversing aspect in the selected semantic aspects with names from fixed syntax and traversing strategy, 3) rename instances of abstract components to organize the collaboration between semantic aspects, 4) make composition from semantic aspects, 5) specify the runtime environment and translate the meta-code to the code of the target programming language, and 6) compile the semantics.

After obtaining meta-semantics (Table 1) meta-code is translated into the target programming language, taking into account the target language (e.g. C++), the implementation of abstract components (e.g. Stack), the operating system (e.g. Unix), the communications between components (e.g. CORBA), runtime components type (e.g. DLL), etc. The translation may be done by hand or automatically (desirable in common cases).

By replacing ADT names we achieve an independent working for some semantic aspects or collaboration between them through common instances of ADT. Another way to get a new semantics is to combine semantic aspects as whole black-box unit. Self-evident method is to execute several semantic aspects sequentially, for instance, we perform static semantics first and dynamic one after that. Instances of ADT can be shared and one semantic aspect can use the results of others. More complex is a parallel execution of many semantics where we need to organize synchronization via instances of ADT.

4. Workflow Case Study

To demonstrate our approach we use a very simple workflow definition language that syntax is described with BNF (Table 2). We have two types of generic statements for describing tasks in a workflow – universal statements and specific statements.

Table 2. Fragment of BNF for simple workflow definition language

workflow	-> series
series	-> statement | series ; statement
statement	-> generic_stm | cond_stm
cond_stm	-> IF compar THEN series ELSE series FI
compar	-> expr relation expr
expr	-> const | var
generic_stm	-> universal_stm | specific_stm
universal_stm	-> *u_stm_type name*
u_stm_type	-> DCOM | CORBA | WEBSERVICE | MANUAL
specific_stm	-> s_stm_type name
s_stm_type	-> ASK | ANSW

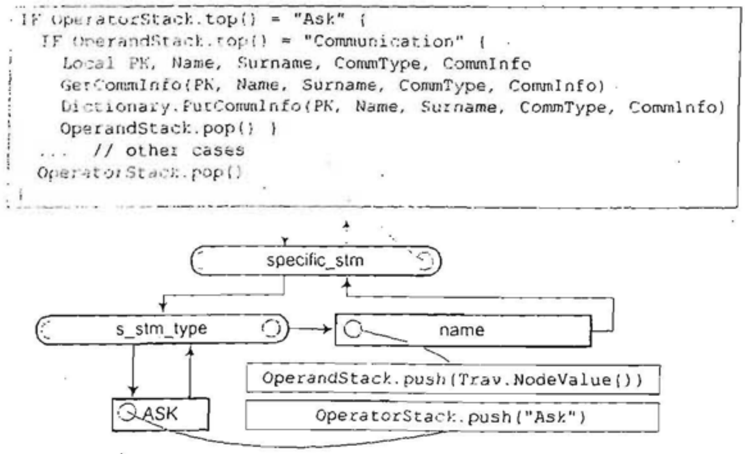
The *universal statements* are used to collaborate with external applications. The universal statement type describes the connection type:

DCOM, CORBA, WEBSERVICE means automatic processing, but MANUAL means, that a human handles this operation. The *specific statement* is used to communicate with a person – usually with a citizen who uses the particular service. There are two types of specific statements. ASK gets information from a person, ANSW sends some information to a person.

Lets us take a look at the following simple workflow:

```
WEBSERVICE Application_writing_and_submitting
ASK Communication
DCOM Application_data_control_and_update
IF Is_data_control_and_updating_successful = True THEN
   DCOM Printing_of_passport
   ANSW Positive_answer
   MANUAL Passport_handing_out
ELSE
   ANSW Negative_answer
FI
```

The purpose of this workflow is to issue a new passport for a person. The workflow has the following activities - a citizen fills in an application form and submits it to the official. It can be a paper form or a web based application. The official or the application asks from the person the communication type and address, and records data into workflow environment. Then the official verifies the correctness of the citizen's filled in form with the data in the Population Register, and if all data are correct, then the passport is issued and delivered to the citizen. Otherwise a negative answer is sent to the citizen. An example of one semantic aspect of this workflow is given in Figure 5.



```
IF OperatorStack.top() = "Ask" {
  IF OperandStack.top() = "Communication" {
    Local PK, Name, Surname, CommType, CommInfo
    GetCommInfo(PK, Name, Surname, CommType, CommInfo)
    Dictionary.PutCommInfo(PK, Name, Surname, CommType, CommInfo)
    OperandStack.pop() }
  ... // other cases
  OperatorStack.pop()
}
```

```
OperandStack.push(Trav.NodeValue())
OperatorStack.push("Ask")
```

Figure 5. Semantic aspect SPECIFIC STATEMENT ASK

5. Conclusions

We have presented ideas for establishing a framework to deal with different collaboration problems between legacy systems. The problem is reduced to describing the collaboration (e.g. workflow) by DSL and building various tools (various semantics) for this DSL.

There are many application generators that automatically produce conventional compiler and interpreter, but we need not only those ones. It is necessary to obtain various supporting tools that are based on the language text processing. The existing formal semantics are not well accepted by language or tool designers. We have made an attempt to search for a compromise to minimize this gap. The latest related works in this field to establish tool-oriented approach are mentioned in [5], [26], [27].

We have offered ideas how semantics can be decomposed into reusable parts. and specific semantics can be composed from them, and how execution is organized. We delegate most of the actual work for semantics to pre-built components (ADT). Our approach allows minimize semantics descriptions for an easier management and provides good implementation in, possible parallel, distributed and heterogeneous environment. Our approach is based on the experience received by constructing prototypes of a multi-language interpreter for conventional programming languages.

Due to practical needs, we lose some precision and benefits of classical semantic formalisms. The next step is to finish the formalization of our approach and to compare it with other formalisms, especially with attribute grammars. Other activities have to be the designing of useful collection of abstract data types.

References

[1] E. Karnitis. E-Government: An Innovative Model of Governance in the Information Society. *Baltic IT&T Review*, 1, 2001

[2] Layna Fischer, editor. The Workflow Handbook 2001, Published in association with the Workflow Management Coalition, 2000

[3] Workflow Standards and Associated Documents, http://www.wfmc.org/standards/docs/Stds_diagram.pdf

[4] Workflow/BPR Tools Vendors http://www.waria.com/databases/wfvendors-A-L.htm

[5] J. Heering and P. Klint. Semantics of Programming Languages: A Tool-Oriented Approach. *ACM SIGPLAN Notices*, 35(3):39-48, March 2000.

[6] W.Kinnersley, ed., The Language List. 1995. http://wuarchive.wustl.edu/doc/misc/lang-list.txt

[7] A Deursen, P. Klint, and J. Visser. Domain-Specific Languages: An Annotated Bibliography. *ACM SIGPLAN Notices*, 35(6):26-36, June 2000.

[8] Special issue on domain-specific languages. *IEEE Transactions on Software Engineering*, 25(3), May/June1999.

[9] David A. Schmidt. Programming Language Semantics. In Tucker [28], pp.2237-2254.

[10] Kenneth C. Louden. Compilers and Interpreters. In Tucker [28], pp.2120-2147.

[11] J. Paakki. Attribute Grammar Paradigms – A High-Level Methodology in Language Implementation. *ACM Computing Surveys*, 27(2):196-255, June 1995.

[12] Frank G. Pagan. *Formal Specification of Programming Languages: A Panoramic Primer*. Prentice-Hall, 1981.

[13] K. Slonneger and B. L. Kurtz. *Formal Syntax and semantics of Programming Languages: A Laboratory Based Approach*. Addison-Wesly, 1995.

[14] F. Ferrucci, F. Napolitano, G. Tortora, M. Tucci, and G. Vitiello. An Interpreter for Diagrammatic Languages Based on SR Grammars. *Proceedings of the 1997 IEEE Symposium on Visual Languages (VL '97)*, pages 292-299, 1997.

[15] U. Kastens and W. M. Wait. Modularity and reusability in attribute grammars. *Acta Informatica 31*, pages 601-627,1994.

[16] T. Matsuzaki and T. Tokuda. CTAG Software Generator Model for Constructing Network Applications. *Proceedings of the Asia Pacific Software Engineering Conference*, pp.120-127, 1998.

[17] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. *Compilers: Principles, Technigues, and Tools*. Addison-Wesley, 1986.

[18] C. Clark. Build a Tree – Save a Parse. *ACM SIGPLAN Notices*, 34(4):19-24, April 1999.

[19] E. Gamma, R. Helm, R. Johnson, and J. Vlisides. *Design Patterns: Elements of Reusable Software*, pages 331-334. Addison-Wesley, 1995.

[20] J. Ovlinger and M. Wand. A Language for Specifying Recursive Traversals of Object Structures. SIGPLAN Notices, 34(10):70-81, 1999. *Proceedings of the 1999 ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages & Applications (OOPSLA '99).*

[21] T. Kühne. The Translator Pattern – External Functionality with Homomorphic Mappings. *Proceedings of the Tools-23: Technology of Object-Oriented Languages and Systems*, pp. 48-59, 1997.

[22] B. Biswas and R. Mall. Reverse Execution of Programs. *ACM SIGPLAN Notices*, 34(4):61-69, April 1999.

[23] M. Mezini and K. Lieberherr. Adaptive Plug-and-Play Components for Evolutionary Software Development. SIGPLAN Notices, 33(10):97-116, 1998. *Proceedings of the 1998 ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages & Applications (OOPSLA '98).*

[24] 24. V. Arnicane, G. Arnicans, and J. Bicevskis. Multilanguage interpreter. In H.-M. Haav and B. Thalheim, editors, *Proceedings of the Second International Baltic Workshop on Databases and Information Systems (DB&IS '96)*, Volume 2: Technology Track, pages 173-174. Tampere University of Technology Press, 1996.

[25] U. Aßmann, T. Genßler, and H. Bär. Meta-programming Grey-box Connectors. *Proceedings of the Technology of Object-Oriented Languages and Systems (TOOLS 33)*, pp.300-311, 2000.

[26] M. Mernik, M. Lenič, E. Avdičauševič, and V. Žumer. Compiler/Interpreter Generator System LISA. *Proceedings of the 33rd Hawaii International Conference on System Sciences - 2000*, pp.10, 2000.

[27] A. M. Sloane. Generating Dynamic Program Analysis Tools. *Proceedings of the Autralian Software Endineering Conference (ASWEC''97)*, pp.166-173, 1997.

[28] Allen B. Tucker, editor. *The computer science and engineering handbook*. CRC Press, 1997.