

LCG Middleware at the KIPT CMS Linux Cluster

M. V. Voronko, S. S. Zub, L. G. Levchuk, D. V. Soroka

NSC Kharkov Institute of Physics and Technology, Ukraine

Problems associated with storage, processing and analysis of huge data samples expected in experiments planned at the Large Hadron Collider (LHC) are discussed. Current status of the KIPT CMS Linux Cluster (KCLC), which is a part of the Moscow distributed regional center for the LHC data analysis, is outlined. Configuration of the LHC computing Grid middleware at the KCLC is described. Participation of the KCLC in the CMS Monte-Carlo event production is outlined.

1. Introduction

The revolutionary development of computing facilities and networks that the society has witnessed for the last decade opens new opportunities for solving long-standing problems in science. This progress has already boosted [1] research activities in molecular biology (molecular modeling for drug design), neuroscience (brain activity analysis), geosciences (earthquake simulations and predictions), theoretical physics (non-perturbative treatment of strong interactions based on the lattice quantum chromodynamics (QCD)), and other fields on frontiers of the contemporary science. The complexity of these problems predetermines the needs of huge computing resources to be involved in the investigations and calls for a broad cooperation of scientists on an international or even global scale. Furthermore, such cooperation implies not only joint intellectual efforts aimed at solution of a scientific problem itself, but also establishment of principally new concepts of computation with possibility to allocate world-widely distributed resources to an application job. The latter aspect is addressed by the Grid technology, which was put forward in late ninetiess (see, e.g., Ref. [2] and references therein) and emerges in the beginning of the new century as a novel universal computational environment enabling integration of processors and storage elements of systems belonging to diverse organizations in widespread locations.

One of the major challenges that promote fast development and implementation of the Grid infrastructures comes from the high-energy physics (HEP), where experimentalists encounter exorbitant amounts of data which have to be securely stored, quickly processed and thoroughly analyzed. Nowadays, the bulk of HEP data is provided by experiments performed on particle accelerators at FNAL, SLAC and BNL (USA), DESY (Germany), and KEK (Japan). However, the situation is going to be dramatically harder in case of experiments to be carried on at the Large Hadron Collider (LHC), world's largest accelerator, which will be put into operation at CERN (Switzerland) in 2007.

The LHC nominal luminosity of $10^{34} \text{ cm}^{-2} \text{ s}^{-1}$ corresponds to 10^9 proton-proton collisions per second. Four detectors installed at the beam crossing points will register particles produced in these interactions. In case of the Compact Muon Solenoid

(CMS) detector [3], about 10^{-7} of the total event flow will be selected by a multi-level trigger for the off-line event processing and analysis. Thus, the data should be archived in a high performance storage system (SS) with the rate of ~ 100 Hz. Since the size of one CMS event written to the SS is supposed to be ~ 1 Mbyte, about 100 Mbyte of the information per second (or more than 1 Pbyte annually) has to be transferred to the SS.

The CMS physics program includes search for the Higgs boson, supersymmetry partners to known elementary particles and rare decays of B-mesons. A discovery of signals manifesting an evidence for such "new physics" requires an immense amount of data to be processed and thoroughly analyzed. Typically, the integral luminosity of $\sim 10^5 \text{ pb}^{-1}$ is needed in order to separate those manifestations against a huge background (see, e.g., Ref. [3]). It means that 10^9 CMS events (or 10^{15} bytes of information) have to be processed and analyzed. All this sets hard requirements upon data acquisition and storage systems, terms of computing and networking.

Furthermore, in preparation of the experiment, a lot of computational work is needed in order to optimize the trigger system and provide its physical contents, i.e., justify selection of given data samples and rejection of others. This activity implies "complete" simulation of the experiment starting from event generators (Monte-Carlo tools realizing existing theoretical models for description of high-energy particle interactions), through detailed simulation of the detector response, to the final stage which is event reconstruction and analysis. Certainly, to accomplish this work, considerable computing power is required.

Major challenges for the scientific community involved in the LHC experiments are supposed to be met via communication and collaboration at a distance, network-distributed computing and software development, remote data resource management and physics analysis. It suggests development of a Grid infrastructure, which would take into account mentioned above peculiarities and purposes of these experiments. The work on this project known as LHC Computing Grid (LCG) [4] is currently under way. Within this project, a complex multi-tier architecture of regional centers (RC's) is created [5] for data storage, processing and analysis (see Fig. 1). Interaction of processes running in different RC's is supposed to occur through the network, what is provided by a sort of connectivity software called middleware. Installation and proper configuration of the LCG middleware at a RC is not a trivial task, since this software is a prototype version subjected to permanent modification, rather than a completed product. Then, there are no universal recipes of tuning the Grid services, which would take into account the variety of system environments. Instead, one has to go on in the configuration process step by step, in order to avoid possible system conflicts. Moreover, participation of remote systems is needed to assure that the LCG middleware is correctly installed and configured at a given RC.

In this paper, we outline the procedure of LCG middleware configuration at the KIPT CMS Linux Cluster (KCLC). The KCLC is a specialized PC farm devised for conducting activities on Monte-Carlo simulations within the CMS physics program and CMS data analysis. The first stage of the KCLC (see Ref. [6]), which now enters the Moscow distributed RC (see Fig. 1), was constructed in 2001. Information about its current performances is given in the next section. Configuration of the LCG middleware elements at the KCLC is described in Sect. 3. Some use of these elements

is concerned in Sect. 4, where our CMS simulation activity is briefly described. Sect. 5 contains concluding remarks.

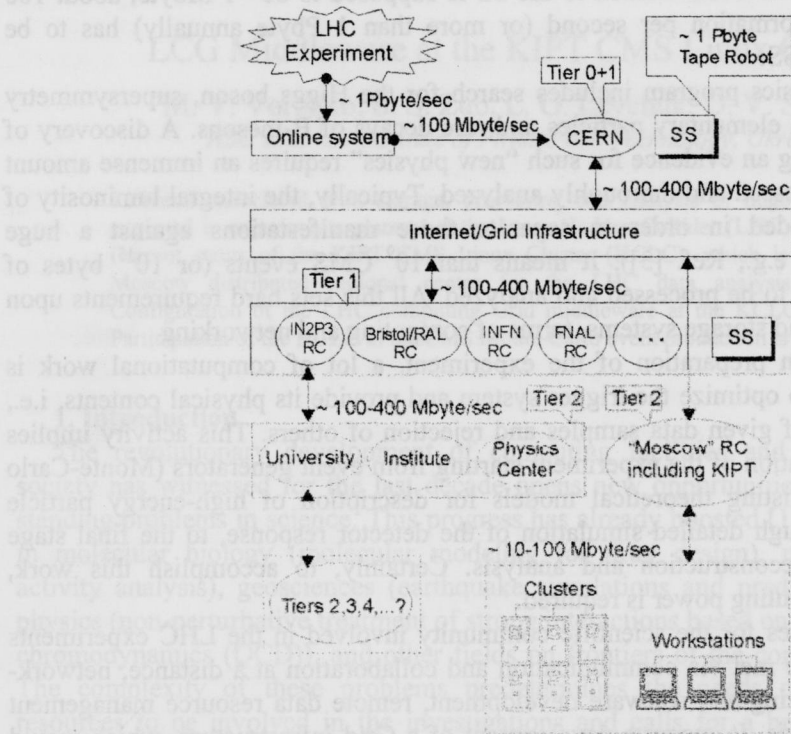


Fig.1. Hierarchy of Grid structures for LHC experiments.

2. KIPT CMS Linux Cluster

The KCLC (see Fig. 2) is a specialized PC farm for computing within the CMS physics program. Eleven dual nodes, viz., 4x800 MHz, 4x1000 MHz, 10x1400 MHz Pentium III, 2x2.0 GHz Xeon and 2x2.4 GHz Xeon processors are allocated for interactive or batch jobs providing the total PC farm CPU power of ~38 Gflops. The overall hard disk drive (HDD) storage is 2.5 Tbyte including two (0.4 Tbyte and 0.9 Tbyte) RAID arrays. System RAM is 9 Gbyte.

The nodes run Linux as the operation system (OS) and the Portable Batch System (PBS) (see, e.g., Ref. [7]) as a batch system. Other important components of our cluster are the network file system (NFS) and network information service (NIS). They provide the joint access to such resources as CERNLIB (including PYTHIA and GEANT), ROOT, PYTHIA, GEANT and LHC++ and to specific CMS software such as CMSIM (a GEANT-based package for simulation of the CMS detector response), ORCA (an object oriented tool for CMS event reconstruction). Versions of the programs are permanently refreshed according to CMS collaboration current demands.

The PBS is used as the cluster batch job and system resource management package. It accepts (see details in Ref. [7]) a batch job (a shell script with some control

attributes) preserves and protects the job until running, runs the job and delivers output to the submitter. The PBS allows one to administer flexibly the system resources while carrying on the computing and may be configured to support jobs run on a single system, or many systems grouped together. It can load processors of the cluster nodes in an optimal way (in accordance with an administrator policy) and select, e.g., the highest-priority execution jobs.

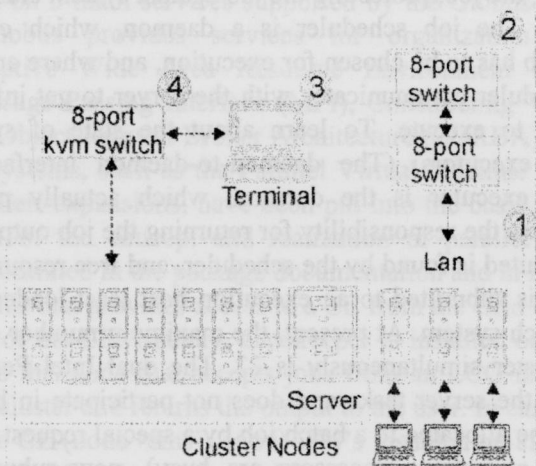


Fig.2. KCLC layout: 1 – server and nodes; 2 – two ethernet switches; 3 – terminal; 4 – kvm switch.

The configuration of the PBS at the NSC KIPT CMS Linux cluster is presented in Fig. 3.

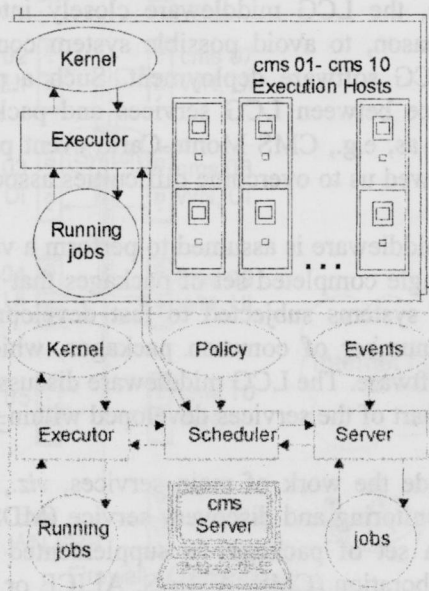


Fig.3. Use of PBS at the KCLC.

The batch system consists (see Ref. [7]) of a command shell and three daemons: the job server, the job scheduler and the job executor, with the latter being activated on every host allocated for execution. The commands are used to submit, monitor, modify and delete jobs and are available at each of the 11 nodes of the cluster. They communicate through the network with the job server. The server main function is to provide proper processing of the "events", i.e., such services as receiving/creating a batch job, modifying the job, protecting the job against system crashes and placing the job into execution. The job scheduler is a daemon, which contains a "policy" controlling which job has to be chosen for execution, and where and when it has to be submitted. The scheduler communicates with the server to get information about the availability of jobs to execute. To learn about the state of system resources, it addresses the job executors. (The daemon-to-daemon interface occurs via the network.) The job executor is the daemon, which actually places the job into execution. It also takes the responsibility for returning the job output to the user. Once a new job to be executed is found by the scheduler, and free resources are available in the system, the job is submitted to an execution host least loaded at the moment as estimated by the batch system. At present, the maximum number of non-parallel jobs executed on the cluster simultaneously is 22. The dual Intel Pentium III 800 MHz computer, performs the server tasks and does not participate in batch executions by default, though can be allocated to a batch job by a special request. If there are no free nodes (i.e., all 22 execution processors are busy), new submitted jobs are put (depending on computing resources requested) into one of 5 queues. When a free processor becomes available, it is immediately allocated to a job from the queue corresponding to the least amount of requested resources.

3. Grid Middleware at the KCLC

As mentioned above, the LCG middleware closely interacts with the system environment. For that reason, to avoid possible system conflicts, we performed a manual (step-by-step) LCG software deployment. Such a procedure helped us to provide a proper interface between LCG services and packages needed for CMS computational tasks such as, e.g., CMS Monte-Carlo event production (see Sect. 4). On the other hand, it allowed us to overcome difficulties associated with low external network bandwidth.

Of course, any Grid middleware is assumed to perform a variety of functions, and, in general, there is no single completed set of packages that would provide all these tools. This is typical of systems subjected to fast development and modification. However, there exist a number of common packages, which are included in the majority of Grid-based software. The LCG middleware discussed here uses the Globus tool kit [8] and the most part of the services developed within the European Data Grid (EDG) project [9].

These packages provide the work of main services, viz., GridFTP, GateKeeper, Globus Job Manager, monitoring and discovery service (MDS), locallogger, Broker, MyProxy, etc. The main set of packages is supplemented by software, which is specific for a given collaboration (CMS, ATLAS, ALICE or LHCb) participating in the LCG project.

Generally speaking, Grid components can be either single computers or clusters. Cluster option is supported by the LCG, in particular, via inclusion of the LCG-

specific interface to the Open PBS [7] called *lcgpbs* in the middleware. Accordingly, earlier PBS configuration described in Ref. [6] has been altered at the KCLC by the *lcgpbs* wrapper in order to provide a due operation of the LCG services.

Furthermore, a system in the LCG may include nodes having different functions. In case of the KCLC, we have nodes of the Computing Element (CE), Worker Node (WN), Storage Element (SE), and User Interface (UI) types.

The CE is based on 5 main services supported by the Globus package [8]. At the top level, the Globus provides services for organization of metacomputer infrastructure: Adaptive Wide Area Resource Environment (AWARE), version MPICH of the Message Passing Interface (MPI), Constructing Virtual Environment (CAVE), Common Object Request Broker Architecture (CORBA), etc. Principles and libraries of earlier systems, such as the Parallel Virtual Machine (PVM), the "plane" MPI, Condor, and their expansions, have been put into the basis of the package. The detailed description of the concept and realization of parallel programming with MPICH methods is outlined in the package documentation and in Ref. [10].

The Globus service that provides access to a CE from the Grid is called *gatekeeper*. It is the "frontend" of the CE. Usually, Gatekeeper is installed on a computer with a real IP address. It is this node that accepts jobs from an LCG user, distributes them over the WN's of a cluster and returns the output to the user. In case of the KCLC (see Fig. 4), we have one CE (node 'cms') and 9 WN's installed on all nodes of the farm, except for the SE ('cms05'). As shown on Fig. 4, the WN services can be configured also on a node running the CE. At the KCLC, the WN installed on the CE node can be allocated to a job through a special request.

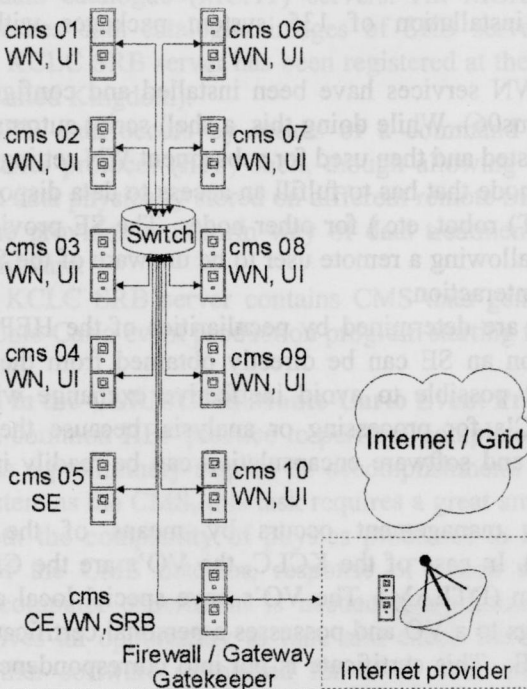


Fig. 4. KCLC with Grid elements.

The next Globus service needed for the CE configuration is the MDS based on the *ldap* protocol. It provides an information support to the local batch system and corresponds to the external LCG structures about the system status. The system data are collected through the so-called Berkley data-based information index (BDII) or Grid information index (GIIS) server. The information is taken by the server from clients of the Grid resource information service (GRIS). The GRIS has been configured on all of the KCLC nodes.

The most important service responsible for the data transfer between LCG nodes is GridFTP. At present, the GridFTP meets all the requirements established for contemporary global networks.

As mentioned above, the LCG middleware contains a Globus wrapped version of the OpenPBS needed for interaction of the CE with the WN's. The Globus service that accomplishes this interface is the Globus resource allocation manager (GRAM).

At last, one needs to configure a very important EDG service called *locallogger*, which provides authorization of users on the CE and secure operation of the system.

In general, CE installation at the KCLC required deployment of 132 system packages and adaptation of 21 configuration files.

The WN's are the nodes, where jobs, actually, run. However, a user does not interact directly with them. All work of the WN's occurs behind the CE and is hidden from the user. The WN's should have access to user's packages (e.g., software specific for a given experiment), while the CE manages their operation via the *gatekeeper* and the batch system. In case of the WN, no special Globus services (like the ones used by the CE) are needed. Nevertheless, Globus utilities (GRIS, GRAM, etc.) remotely collect information about WN resources for the CE. The WN setup at the KCLC caused installation of 135 system packages with adaptation of 8 configuration files.

First of all, the WN services have been installed and configured on one of the KCLC nodes (*viz.*, cms06). While doing this, a shell script automating this procedure was developed and tested and then used for subsequent WN setting up on other nodes.

The SE is a Grid node that has to fulfill an access to data disposed on an SS (large disk array, tape (DLT) robot, etc.) for other nodes. The SE provides a uniformity of such an access, with allowing a remote user to be unaware of the SS technical details, number and internal interaction.

The SE functions are determined by peculiarities of the HEP tasks. Information about data location on an SE can be directly obtained from the corresponding CE name. This makes it possible to avoid ineffective exchange with big data arrays between different RC's for processing or analysis, because the principle of data, computing resources and software encapsulation can be readily implemented in this scheme.

LCG file access management occurs by means of the so-called Virtual Organizations (VO's). In case of the KCLC, the VO's are the CMS and the CERN Grid deployment team (DTEAM). The VO's have special local accounts at the SE. Any LCG user belongs to a VO and possesses a personal certificate mentioned in the *grid-mapfile* at the SE. This certificate is put into correspondence with a local user with name that coincides with the VO name.

The SE is burdened with more system services than the CE or WN. In addition, it actively uses such Globus services as *gatekeeper*, GridFTP, Globus Job Manager,

MDS, *tomcat*, etc. Adaptation of 14 configuration files was needed for SE installation at the KCLC.

The UI is any node, which runs software providing direct access to Grid resources. It is that machine that gives an opportunity for a user to communicate with the Grid through the LCG command interface. The UI is used to submit jobs and get results of computations.

To run the UI functions, *openssh* must be installed and properly configured on the system. In case of a cluster, it is important that the *ssh* configuration would support batch job submission and processing. At the KCLC, this is achieved by a passwordless node-to-node access based on the host-based authorization provided by a special *sshd* daemon configuration.

One of the main Grid problems is security. It is solved not only by special protocols, but also through a package of measures, which include obligatory use of personal and other certificates, authorization with a password and ordinary restriction of access to ports of an element performed by a system of special package filters called *firewall*. All these measures are consistent with the current LCG middleware version installed at the KCLC, which is LCG_2_2_0.

In addition to the LCG middleware described above, we configured at the KCLC also the storage resource broker (SRB) [11], which is a separate (non-LCG) Grid element. The SRB is client-server middleware that enables a uniform interface for connecting to heterogeneous data resources over a network and data replication. At the KCLC, both SRB server and client services have been running since 2002.

Information about all data accumulated on world-spread SRB servers is contained on so-called metadata catalogue (MCAT) servers. An MCAT server stores and synchronously refreshes data catalogue images of SRB servers registered at this MCAT server. The KCLC SRB server has been registered at the MCAT server of the Bristol/RAL RC (United Kingdom).

An access to SRB data occurs by means of a command interface resembling "classical" file transfer protocol (FTP) shell, though allowing a user to manipulate pseudo-locally with data physically stored on different remote SRB servers. Similar to the SE, this scheme provides a uniform way of data treatment and hides technical details of the SS hardware.

At present, the KCLC SRB server contains CMS data generated on the cluster within the CMS Monte-Carlo event production program starting from 2002.

4. Participation in the CERN/CMS Monte-Carlo Event Production

It has become a common HEP practice to perform complete computer simulation of an experiment as a preliminary step of its accomplishment. Certainly, in case of such a complex system as the CMS, this task requires a great amount of computation. It is caused by both the complexity of physics processes to be simulated and the elaborate design of the CMS detector, response of which should be adequately reproduced. Such computer experiment is needed to optimize the detector trigger system. Then, it gives an opportunity to tune and check the integrity of the LCG middleware and other software developed for CMS data distributed storage and analysis. Also, an important objective of this activity is studying software tools developed by the collaboration for event reconstruction and training of physicists intending to participate in data processing and analysis.

Participation of the RC's in the complete computer simulation of the CMS experiment occurs via the so-called CMS Monte-Carlo event production. Computing resources of universities and research centers from more than 30 countries are involved in solution of this problem.

The scheme of the complete event production chain is displayed in Fig. 5. The computer experiment starts from the event generation (module CMKIN), performed by the PYTHIA simulation tool (see Ref. [12]) that implements the Lund version of the quark-parton model of proton-proton interactions. Detector response simulation (block CMSIM/OSCAR) is introduced by the Fortran-based package CMSIM performing complete GEANT [13] (version 3) simulation of passing particles through the CMS detector. Optionally, this step can be fulfilled with the OSCAR [14] package, the detector simulation object-oriented software, utilizing the C++-based GEANT_4 toolkit. The CMSIM/OSCAR simulation is the most CPU-time-consuming part of the production cycle. The average multiplicity of particles produced in proton-proton collisions at the LHC energy reaches several hundreds, and the history of each of such particles and its possible decay products should be followed up to the particle disappearance (due to decay, stopping or escaping the detector volume). A typical production job generating 250 CMS events takes more than 24 hours of a 1 GHz Pentium III processor. The result of this step is information about tracks of all particles passed through the detector. Based on this information, the signals in the CMS compartments are evaluated (step HITS). Digitization of these signals (stage DIGI) and reconstruction of particle tracks (RECO) with (optionally) taking into account possible pile-up events are accomplished by the ORCA tool [15].

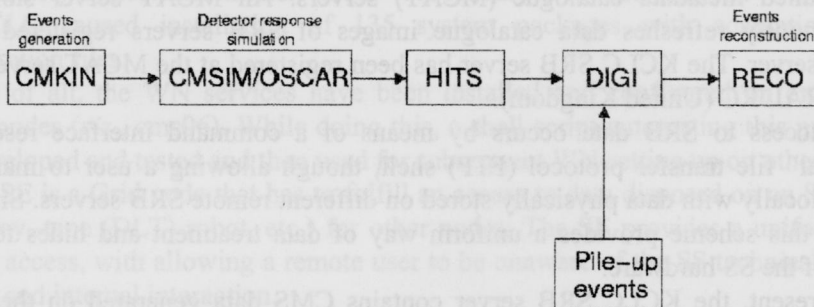


Fig.5. Complete cycle of the CMS event generation/reconstruction.

The KIPT/CMS Group has been participating in the event production runs since 2002 employing the KCLC. A production center has to have a cluster of computers capable of running batch jobs and a special CMS simulation software. Apart from the mentioned above physics software, the latter includes a number of special production tools.

An RC receives a request to simulate the events in the form of an assignment with a unique identification number (ID). This process is maintained with the help of a reference database (RefDB) located at CERN, which generates requests, distributes assignments and traces their status. When an assignment is received, it should be processed with a complex of Unix shell, Python and Perl scripts called McRunjob. Of course, McRunjob has to be properly configured with taking into account the system environment of the PC farm and the peculiarities of the given assignment. It takes the

assignment ID as an input and downloads the relevant data from the RefDB creating a number of jobs, with each of them being a complete entity capable of running independently. It also keeps the track of jobs by updating their status in the RefDB and using the BOSS software [16], which is a wrapper for the cluster batch system providing an interface to the MySQL database.

In case of the KCLC, the jobs are submitted to a PBS queue, which distributes them for execution on different nodes of the cluster. If some of these jobs fail for some reason, they can be resubmitted later. The BOSS provides real time monitoring and book keeping for jobs submitted and running on the cluster and storing the information in the MySQL database located on the KCLC CE node ('cms').

The complexity of installation and configuration of the production software is associated with difficulties of adaptation of these complex tools to a specific platform with the peculiarities and nuances typical of a cluster system. Besides, versions of the physics packages must be agreed upon in order to get results that can be checked and reproduced.

When a production job is running, a summary file for this job is generated by the production software, which contains complete information about the computation process (versions of all physics packages actually used, initial random number, names and sizes of all generated files, execution host name, date and time of the beginning and the end of the execution, resources [RAM and CPU time] used, etc.). Upon completion of the job, the summary file is automatically either e-mailed directly to CERN, or written to the MySQL database to be processed later by a script transferring the relevant information to the CERN RefDB. In any case, information in the RefDB is updated on a regular basis. This provides permanent monitoring of the production activity in the RC's. On the other hand, the summary files give an opportunity to check or reproduce the production results, if some of them appear to be questionable or incomplete.

Reliability of the production results is also achieved through the so-called validation assignments (VA). Before running a production assignment, it is mandatory for a production center to ask for the VA from the RefDB and run it. The VA examines the system environment and correctness of the software versions. The resulting VA summary files submitted to CERN must coincide with sample data. Only in this case the RC is admitted to the physical event generation and can get the production assignment.

Concise statistics of KCLC participation in the CMS Monte-Carlo production runs is presented in Table 1. More than 600000 events have been generated at the KCLC within this program for the two years, with the majority of these data being the results of the most cumbersome production step CMSIM. The CMSIM data occupy ~1 Mbyte/event of the HDD space. There are also generated (CMKIN) and reconstructed (RECO) events in the form of the so-called HEP-standard "ntuples", as well as HITS files, which are comparable in the size per event with the CMSIM data. A physical content of a production assignment is given by the corresponding dataset name. For instance, dataset 'hi02_qcd80_90' means generation of QCD jets in the transverse momentum range $p_T=80-90$ GeV/c requested by the HI ("heavy ions") physics group in 2002, while 'jm03_Wjets_20_50' refers to production of weak $W^\pm$ bosons with $p_T=20-50$ GeV/c accompanied by QCD jets in proton-proton collisions at the LHC

energy requested by the JetMET (“jets and missing transverse energy”) group in 2003. More detailed information about each of the datasets listed in Table 1 can be found in the CERN RefDB.

In conclusion, we would like to mention that, in addition to participation in the CMS event production program, the KCLC is intensively used in our studies of possibilities to observe a heavy Standard Model Higgs boson in decays $H \rightarrow ZZ \rightarrow ll\nu\nu$ with the CMS detector. Details concerning this activity can be found, e.g., in Ref. [17].

Table 1. Results of KCLC participation in CMS Monte-Carlo event production in 2002–2004.

Data Set	Assignment ID	Number of events generated	Data type
Hi02_photon100 110	1829	1000	CMSIM
Hi02_qcd80 90	1644	5000	CMKIN, CMSIM
Hi02_qcd100 110	1645	1000	CMKIN, CMSIM
eg02_BigJets	1958, 1959, 1960, 1961	85250	CMKIN+CMSIM, HITS, DIGI, RECO
Bt03_qcd120 170th	2245	80000	CMKIN
mu03_tt2mu_Mll1000	2549	500	CMKIN
eg03_Wenu_calibration	2840	99500	CMSIM
Jm03_Wjets 20 50	3024	100000	CMKIN
eg03_Zee	3047	100000	CMSIM
Tt_ch_140_ib20	5344, 5345, 5481	150000	CMKIN, CMSIM, HITS

5. Conclusion

An enormous data flow expected in the LHC experiments forces the HEP community to resort to the LCG, a Grid-based model of data storage, processing and analysis. Accordingly, a worldwide infrastructure of LCG RC's is created, with the KCLC being a part of the Moscow distributed RC.

The KCLC is a specialized PC farm constructed at the NSC KIPT for computer simulations within the CMS physics program and preparations to the CMS data analysis. Currently, it combines 22 CPU's, and its total HDD storage is 2.5 Tbyte.

These performances allow us to participate in the CMS Monte-Carlo event production representing “complete” simulation of the CMS experiment. Such computer experiments require huge computation resources, and their accomplishment is distributed world-widely over the RC's. For the time being, more than 600000 events have been generated on the KCLC in such production runs. The most part of the generated data are stored at (and can be fetched from) the KCLC SRB server.

To proceed further with our preparations for the CMS experiment, the installation and proper configuration of the LCG middleware at the KCLC was needed. This is not a trivial task, since this software is neither universal nor “complete” product, and one has to go on in the configuration process step by step, in order to avoid system conflicts. At present, the LCG middleware that supports the CE, WN, SE, and UI functions for the KCLC nodes has been installed. A number of special system services had to be configured in each case. The employed at the KCLC procedure of manual step-by-step LCG software installation and configuration as well as the scripts

developed for its implementation are applicable for LCG middleware deployment on other computing centers scheduling allocation of their resources for tasks of LHC data processing or analysis.

To our knowledge, the KCLC is, at present, the only PC farm in Ukraine, which has been constructed to meet the HEP computing needs, and which is actively used in computation activities for the LHC experiments. Introduction at the KCLC of the computation concept based on the novel Grid technology is also pioneering for the country. So, we hope that our experience can be useful for construction of other computing systems of the similar type, and, on the other hand, our work is promotional for introduction of the Grid computing technology in Ukraine.

Further development of the KCLC is planned with considerable increase of its capacities and deeper integration into the LCG structures.

6. Acknowledgements

The authors thank Prof. P. V. Sorokin for his permanent interest to this work and helpful discussions.

REFERENCES

1. <http://www.gridcomputing.com>.
2. Foster I., Kesselman C., and Tuecke S. The Anatomy of the Grid Enabling Scalable Virtual Organization // Int. J. High Perform. Comput. Appl. 15(3), (2001).
3. The Compact Muon Solenoid Technical Proposal, - CERN/LHCC 94-38, 15 December 1994.
4. Lamanna M. The LHC computing grid project at CERN // Nucl. Instr. and Meth. A. (in press); <http://lcg.web.cern.ch/LCG>.
5. Stickland D. CMS Computing and Core-Software Report, - Proc.VIth Annual RDMS CMS Collaboration Meeting, "Physics Program with the CMS Detector", IHEP, Moscow, Russia, December 19, 2001, p.293-308.
6. Levchuk L.G., Sorokin P.V., Soroka D.V., and Trubnikov V.S. NSC KIPT Linux Cluster for Computing within the CMS Physics Program // Problems of Atomic Science and Technology, Ser. "Nuclear Physics Investigations". Kharkov 2002. N2(40), p.49-51.
7. <http://www.openpbs.com/>.
8. <http://www.globus.org>.
9. EDG User Guide, - IST-2000-25182, 15 January 2003, p.7.
10. MacDonald N., Minty E., Harding T., Brown S. Writing Message Passing Parallel Programs with MPI, - Edinburgh Parallel Computing Centre, The University of Edinburgh.
11. The Storage Resource Broker, - San Diego Supercomputing Center (SDSC), <http://www.npaci.edu/DICE/SRB>.
12. Sjöstrand T., Eden P., Friberg C., Lönnblad L., Miu G., Mrenna S. and Norrbin E. // Computer Physics Commun. 135 (2001) 238.
13. Brun R., Goossens M., and Zoll L. GEANT Users Guide, - Program Library Long Writeup W5013. CERN, 1993.

14. OSCAR, Object oriented Simulation for CMS Analysis and Reconstruction, <http://crnsdoc.cern.ch/oscar/>.
15. Object Oriented Reconstruction for CMS Analysis, - CMS IN 199/001.
16. Grandi C., Renzi A. Object Based System for Batch Job Submission and Monitoring (BOSS), - CMS Note 2003/XXX.
17. Levchuk L.G. On the Possibility to Observe with CMS a Moderately Heavy Higgs Decaying via $H \rightarrow ZZ \rightarrow ll\nu\nu$, - 8th Annual RDMS CMS Collaboration Conference, JINR, Dubna, December, 2003.

Моделирование составных поверхностей в системах символьной математики

П. Г. Доля

Харьковский национальный университет им. В.Н. Каразина, Украина

The method of compiling of uniform analytic formulas for the equations of the continuous surfaces composed of various sections with the given equations is proposed. The general equation of a piecewise-ruled surface stretched on a frame consisting of a set of not intersected space curves constructs.

1. Введение

При расчете и проектировании конструкций, составленных из элементов сложной геометрической формы, возникает задача строгого математического описания контуров их поверхностей. Один из подходов, применяемых в САПР и системах символьной математики, состоит в использовании множества графических примитивов. Однако он плохо приспособлен для использования в расчетах. Конструктивный математический подход к решению этой задачи был предложен В. Л. Рвачевым [1,2], который дал метод построения неявных уравнений составных чертежей. Еще один подход к решению этих задач дает теория сплайнов [3,4]. В работах автора [5,6] предложен способ математического описания составных кривых и приведены общие формулы для построения явных и параметрических уравнений кусочно-гладких непрерывных функций и кривых. В настоящей работе этот метод применяется для построения единых аналитических выражений для уравнений составных поверхностей. Выводятся также общие формулы для построения единых аналитических уравнений кусочно-линейчатых поверхностей.

В основу предлагаемой методики положено использование введенной в работе [5] функции:

$$P(t, a, w) = \frac{1}{2w} (w + |t - a| - |t - a - w|) \quad (w \neq 0). \quad (1)$$

Функция $P(t, a, w)$ является непрерывной по переменной t . При $t < a$ функция равна 0, при $t > a + w$ она равна 1, а внутри интервала $a \leq t \leq a + w$ функция линейная $\frac{1}{w}(t - a)$ (если $w > 0$).

2. Параметрическое уравнение кусочно-линейчатой поверхности

Пусть дано семейство непересекающихся пространственных криволинейных отрезков с уравнениями $\vec{r} = \vec{r}_k(\tau)$, $\tau \in [\tau_0^k, \tau_1^k]$ ($k=0, 1, \dots, n$). В каждом из

уравнений сделаем замену переменных $\tau = \tau_0^k + \frac{\tau_1^k - \tau_0^k}{u_1 - u_0}(u - u_0)$. Получим новые

уравнения кривых $\bar{r}_k(u) = \bar{r}_k\left(\tau_0^k + \frac{\tau_1^k - \tau_0^k}{u_1 - u_0}(u - u_0)\right)$, в которых параметр u

изменяется в одинаковом для всех криволинейных отрезков интервале $[u_0, u_1]$.

Соединим прямолинейным отрезком начальные точки соседних криволинейных отрезков и будем его двигать так, чтобы он все время соединял точки, имеющие одинаковые значения параметра u . Между каждой парой кривых получим линейчатую поверхность. Эти поверхности стыкуются друг с другом по кривым исходного семейства, образуя некоторую непрерывную кусочно-линейчатую поверхность. Составим параметрическое уравнение этой поверхности.

Положим, что кривые исходного семейства являются координатными линиями $v = \text{const}$ образованной кусочно-линейчатой поверхности. Назначим каждой кривой значение параметра $v = v_k$ так, чтобы значения v_k образовывали монотонно-возрастающую последовательность. Тогда уравнение кусочно-линейчатой поверхности будет иметь следующий вид:

$$\bar{R}(u, v) = \bar{r}_0(u) + \sum_{k=1}^n (\bar{r}_k(u) - \bar{r}_{k-1}(u)) P(v, v_{k-1}, v_k - v_{k-1}), \quad (u_0 \leq u \leq u_1, v_0 \leq v \leq v_n). \quad (2)$$

Доказательство. Рассмотрим вектор-функцию $\bar{R}(u, v)$ на интервале $v_{p-1} \leq v \leq v_p$ при некотором фиксированном p ($1 \leq p \leq n$). В сумме (2) все функции $P(v, v_{k-1}, v_k - v_{k-1}) = 1$ при $k < p$, а при $k > p$ $P(v, v_{k-1}, v_k - v_{k-1}) = 0$. Поэтому (2) принимает вид

$$\begin{aligned} \bar{R}(u, v) &= \bar{r}_0(u) + \sum_{k=1}^{p-1} (\bar{r}_k(u) - \bar{r}_{k-1}(u)) + (\bar{r}_p(u) - \bar{r}_{p-1}(u)) P(v, v_{p-1}, v_p - v_{p-1}) = \\ &= \bar{r}_{p-1}(u) + (\bar{r}_p(u) - \bar{r}_{p-1}(u)) P(v, v_{p-1}, v_p - v_{p-1}) \end{aligned}$$

Учитывая (1) и то, что $v_{p-1} \leq v \leq v_p$, последнее выражение преобразуется к виду

$$\bar{R}(u, v) = \bar{r}_{p-1}(u) + (\bar{r}_p(u) - \bar{r}_{p-1}(u)) \frac{v - v_{p-1}}{v_p - v_{p-1}}.$$

Это значит, что на любом отрезке $v_{p-1} \leq v \leq v_p$, вектор-функция $\bar{R}(u, v)$ представляет уравнение линейчатой поверхности, натянутой между кривыми $\bar{r}_{p-1}(u)$ при $v = v_{p-1}$ и $\bar{r}_p(u)$ при $v = v_p$. Если $v \leq v_0$, то в формуле (2) все функции $P(v, v_{k-1}, v_k - v_{k-1})$ равны нулю, и мы получаем, что $\bar{R}(u, v) = \bar{r}_0(u)$. Если $v \geq v_n$, то в формуле (2) все функции $P(v, v_{k-1}, v_k - v_{k-1})$ равны единице, и

мы получаем, что $\bar{R}(u, v) = \bar{r}_0(u) + \sum_{k=1}^n (\bar{r}_k(u) - \bar{r}_{k-1}(u)) = \bar{r}_n(u)$. Т.о., выражение (2)

дает уравнение непрерывной кусочно-линейчатой поверхности, проходящей последовательно через пространственные кривые $\bar{r}_0(u), \bar{r}_1(u), \dots, \bar{r}_n(u)$ при

$v = v_0, v_1, \dots, v_n$. Фактически кривые семейства $\{\bar{r}_k(u)\}$ являются ребрами образованной поверхности.

Отметим, что с помощью формулы (2) строятся уравнения полной поверхности таких тел как куб, пирамида, конус, уравнение поверхности цилиндра вместе с верхним и нижним основанием и уравнения поверхностей многих многогранников. Например, поверхность куба можно представить как кусочно-линейчатую поверхность натянутую между 4 кривыми. Первой кривой семейства будет точка – центр верхнего основания. Второй – квадрат, образующий контур верхнего основания. Третьей – квадрат, являющийся контуром нижнего основания. Четвертой – точка, лежащая в центре нижнего основания. Для куба с вершинами в точках $(1,1,1)$, $(-1,1,1)$, $(-1,-1,1)$, $(1,-1,1)$, $(1,1,-1)$, $(-1,1,-1)$, $(-1,-1,-1)$, $(1,-1,-1)$ параметрическое уравнение поверхности имеет следующий вид

$$x(u, v) = \frac{1}{2} (1 - |u| + |u-1| + |u-2| - |u-3|) \cdot (|v| - |v-1| - |v-2| + |v-3|)$$

$$y(u, v) = \frac{1}{2} (1 - |u-1| + |u-2| + |u-3| - |u-4|) \cdot (|v| - |v-1| - |v-2| + |v-3|)$$

$$z(u, v) = -|v-1| + |v-2|,$$

при $0 \leq u \leq 4$, $0 \leq v \leq 3$. Эти уравнения построены по формуле (2) с последующей заменой функций $P(v, v_{k-1}, v_k - v_{k-1})$ их выражениями (1). Предварительно были составлены параметрические уравнения квадратов (контуров верхнего и нижнего оснований) в соответствии с формулами работы [5].

3. Уравнение составной поверхности

Уравнение составной поверхности, секции которой заданы уравнениями в параметрическом виде. Пусть на плоскости UV задана ортогональная сетка $[u_0, u_1, \dots, u_n] \times [v_0, v_1, \dots, v_m]$ и в ее ячейках $\Delta_{ij} = [u_i, u_{i+1}] \times [v_j, v_{j+1}]$ определены вектор-функции $\bar{r}_{ij}(u, v) = (x_{ij}(u, v), y_{ij}(u, v), z_{ij}(u, v))$ ($i=0, 1, 2, \dots, n-1$, $j=0, 1, 2, \dots, m-1$), представляющие уравнения некоторых поверхностей. Положим, что для составной поверхности в области $[u_0, u_n] \times [v_0, v_m]$ выполняется условие непрерывности. Т.е. для соседних вектор-функций $\bar{r}_{ij}(u, v)$ выполняются условия $\bar{r}_{ij}(u_{i+1}, v) = \bar{r}_{i+1j}(u_{i+1}, v)$ при $v_j \leq v \leq v_{j+1}$, и $\bar{r}_{ij}(u, v_{j+1}) = \bar{r}_{i,j+1}(u, v_{j+1})$ при $u_i \leq u \leq u_{i+1}$. На прямоугольнике $[u_0, u_n] \times [v_0, v_m]$ требуется построить единое аналитическое уравнение поверхности, состоящей из секций с уравнениями $\bar{r}_{ij}(u, v)$ в областях Δ_{ij} .

Составим уравнение каждой секции поверхности в следующем виде

$$\bar{R}_{ij}(u, v) = \bar{r}_{ij}(u_i + (u_{i+1} - u_i)P(u, u_i, u_{i+1} - u_i), v_j + (v_{j+1} - v_j)P(v, v_j, v_{j+1} - v_j)). \quad (3)$$

Поверхность с этим уравнением $\bar{R}_{ij}(u, v)$ в области прямоугольника $\Delta_{ij} = [u_i, u_{i+1}] \times [v_j, v_{j+1}]$ совпадает с поверхностью, имеющей уравнение $\bar{r}_{ij}(u, v)$ в этой области. Действительно, для любой точки прямоугольника $(u, v) \in \Delta_{ij}$

выполняются следующие соотношения $u_i + (u_{i+1} - u_i)P(u, u_i, u_{i+1} - u_i) = u$, $v_j + (v_{j+1} - v_j)P(v, v_j, v_{j+1} - v_j) = v$ и вектор-функция $\bar{R}_{ij}(u, v)$ совпадает с $\bar{r}_{ij}(u, v)$. Любые значения (u, v) , лежащие вне этой области, отображаются в точки границы секции. Например, если $u > u_{i+1}$, $v_j < v < v_{j+1}$, то имеет место $u_i + (u_{i+1} - u_i)P(u, u_i, u_{i+1} - u_i) = u_{i+1}$ и $v_j + (v_{j+1} - v_j)P(v, v_j, v_{j+1} - v_j) = v$. Тогда $\bar{R}_{ij}(u, v) = \bar{r}_{ij}(u_{i+1}, v)$, т.е. точка поверхности $\bar{R}_{ij}(u, v)$ находится на границе секции поверхности $\bar{r}_{ij}(u, v)$. Аналогично проверяется, что любые значения $(u, v) \notin \Delta_{ij}$ отображаются в граничные точки секции. Т.о., вектор-функция $\bar{R}_{ij}(u, v)$ представляет параметрическое уравнение ij -секции поверхности с краем. Ее можно назвать криволинейной гранью составной поверхности.

Построим единое уравнение поверхности, составленной из секций поверхностей с уравнениями $\bar{R}_{0q}(u, v), \bar{R}_{1q}(u, v), \dots, \bar{R}_{n-1q}(u, v)$ ($q=0, 1, \dots, m-1$). Для этого сложим уравнения этих секций следующим образом

$$\bar{R}_q(u, v) = \bar{R}_{0q}(u, v) + \sum_{i=1}^{n-1} (\bar{R}_{iq}(u, v) - \bar{R}_{iq}(u_i, v)). \quad (4)$$

Покажем, что для точек $(u, v) \in [u_p, u_{p+1}] \times [v_q, v_{q+1}]$ имеет место равенство $\bar{R}_q(u, v) = \bar{r}_{pq}(u, v)$. Для этого преобразуем (4) следующим образом

$$\begin{aligned} \bar{R}_q(u, v) = \bar{R}_{0q}(u, v) + \sum_{i=1}^{p-1} (\bar{R}_{iq}(u, v) - \bar{R}_{iq}(u_i, v)) + (\bar{R}_{pq}(u, v) - \bar{R}_{pq}(u_p, v)) + \\ + \sum_{i=p+1}^{n-1} (\bar{R}_{iq}(u, v) - \bar{R}_{iq}(u_i, v)). \end{aligned}$$

При $i < p$ $\bar{R}_{iq}(u, v) = \bar{r}_{iq}(u_{i+1}, v_q + (v_{q+1} - v_q)P(v, v_q, v_{q+1} - v_q)) = \bar{R}_{iq}(u_{i+1}, v)$ т.к. $P(u, u_i, u_{i+1} - u_i) = 1$. При $i > p$ имеем $P(u, u_i, u_{i+1} - u_i) = 0$ и, следовательно, $\bar{R}_{iq}(u, v) = \bar{r}_{iq}(u_i, v_q + (v_{q+1} - v_q)P(v, v_q, v_{q+1} - v_q)) = \bar{R}_{iq}(u_i, v)$. Выражение для $\bar{R}_q(u, v)$ принимает следующий вид

$$\bar{R}_q(u, v) = \bar{R}_{0q}(u_1, v) + \sum_{i=1}^{p-1} (\bar{R}_{iq}(u_{i+1}, v) - \bar{R}_{iq}(u_i, v)) + (\bar{R}_{pq}(u, v) - \bar{R}_{pq}(u_p, v))$$

Условия стыковки соседних секций поверхности, имеющие вид $\bar{r}_{iq}(u_{i+1}, v) = \bar{r}_{i+1q}(u_{i+1}, v)$ ($v_q \leq v \leq v_{q+1}$), теперь можно записать в форме $\bar{R}_{iq}(u_{i+1}, v) = \bar{R}_{i+1q}(u_{i+1}, v)$, поскольку $\bar{R}_{ij}(u, v) = \bar{r}_{ij}(u, v)$ для любых точек $(u, v) \in [u_i, u_{i+1}] \times [v_j, v_{j+1}]$. Поэтому в выражении для $\bar{R}_q(u, v)$ сокращаются все слагаемые, кроме одного. Получаем, что для точек $(u, v) \in \Delta_{pq}$ $\bar{R}_q(u, v) = \bar{R}_{pq}(u, v) = \bar{r}_{pq}(u, v)$. Это значит, что поверхность с уравнением $\bar{R}_q(u, v)$ при $v_q \leq v \leq v_{q+1}$ на отрезках $u_p \leq u \leq u_{p+1}$ совпадает с поверхностями соответствующих pq -секций.

Поверхность с уравнением $\bar{R}_q(u, v)$ также можно рассматривать как некоторую секцию поверхности. Сложим эти уравнения (4) следующим образом

$$\bar{R}(u, v) = \bar{R}_0(u, v) + \sum_{j=1}^{m-1} (\bar{R}_j(u, v) - \bar{R}_j(u, v_j)). \quad (5)$$

Нас интересует значение этого выражения в точке $(u, v) \in \Delta_{pq}$. Разложим сумму (5) на части.

$$\begin{aligned} \bar{R}(u, v) = \bar{R}_0(u, v) + \sum_{j=1}^{q-1} (\bar{R}_j(u, v) - \bar{R}_j(u, v_j)) + (\bar{R}_q(u, v) - \bar{R}_q(u, v_q)) + \\ + \sum_{j=q+1}^{m-1} (\bar{R}_j(u, v) - \bar{R}_j(u, v_j)) \end{aligned} \quad (6)$$

Если $j < q$ и $v_q \leq v \leq v_{q+1}$, имеем $P(v, v_j, v_{j+1} - v_j) = 1$, и для любого $u \in [u_i, u_{i+1}]$ будет $\bar{R}_j(u, v) = \bar{R}_{ij}(u, v) = \bar{r}_{ij}(u_i + (u_{i+1} - u_i)P(u, u_i, u_{i+1} - u_i), v_{j+1}) = \bar{R}_{ij}(u, v_{j+1})$. Если $q < j$ и $v_q \leq v \leq v_{q+1}$, имеем $P(v, v_j, v_{j+1} - v_j) = 0$, и для любого $u \in [u_i, u_{i+1}]$ будет $\bar{R}_j(u, v) = \bar{R}_{ij}(u, v) = \bar{r}_{ij}(u_i + (u_{i+1} - u_i)P(u, u_i, u_{i+1} - u_i), v_j) = \bar{R}_{ij}(u, v_j) = \bar{R}_j(u, v_j)$.

Последняя сумма в выражении (6) обращается в 0, и мы получаем

$$\bar{R}(u, v) = \bar{R}_{p0}(u, v_1) + \sum_{j=1}^{q-1} (\bar{R}_{pj}(u, v_{j+1}) - \bar{R}_{pj}(u, v_j)) + (\bar{R}_{pq}(u, v) - \bar{R}_{pq}(u, v_q)) \quad (7)$$

Отметим, что при выводе последней формулы мы использовали доказанный выше факт, что для любой точки $(u, v) \in \Delta_{ij}$ $\bar{R}_j(u, v) = \bar{R}_{ij}(u, v)$.

Условие стыковки соседних секций $r_{ij}(u, v_{j+1}) = \bar{r}_{ij+1}(u, v_{j+1})$ при $u_i \leq u \leq u_{i+1}$ можно записать в виде $\bar{R}_{ij}(u, v_{j+1}) = \bar{R}_{ij+1}(u, v_{j+1})$. Действительно, при $u_i \leq u \leq u_{i+1}$ вектор-функции $\bar{R}_{ij}(u, v_{j+1})$ и $\bar{R}_{ij+1}(u, v_{j+1})$ совпадают с вектор-функциями $r_{ij}(u, v_{j+1})$ и $\bar{r}_{ij+1}(u, v_{j+1})$, которые равны. Покажем, что равенство $\bar{R}_{ij}(u, v_{j+1}) = \bar{R}_{ij+1}(u, v_{j+1})$ выполняется и при других значениях u . Действительно, для $u < u_i$, $P(u, u_i, u_{i+1} - u_i) = 0$ и мы получаем $\bar{R}_{ij}(u, v_{j+1}) = \bar{r}_{ij}(u_i + (u_{i+1} - u_i)P(u, u_i, u_{i+1} - u_i), v_{j+1}) = \bar{r}_{ij}(u_i, v_{j+1})$. Аналогично получаем $\bar{R}_{ij+1}(u, v_{j+1}) = \bar{r}_{ij+1}(u_i, v_{j+1})$. Но $\bar{r}_{ij}(u_i, v_{j+1}) = \bar{r}_{ij+1}(u_i, v_{j+1})$, поэтому для $u < u_i$ выполняется равенство $\bar{R}_{ij}(u, v_{j+1}) = \bar{R}_{ij+1}(u, v_{j+1})$. Аналогично, для $u_{i+1} < u$ также имеет место равенство $\bar{R}_{ij}(u, v_{j+1}) = \bar{R}_{ij+1}(u, v_{j+1})$. Следовательно, равенство $\bar{R}_{ij}(u, v_{j+1}) = \bar{R}_{ij+1}(u, v_{j+1})$ выполняется для любых значений u .

Учитывая это условие стыковки соседних секций, после сокращений в соотношении (7) остается только одно слагаемое $\bar{R}_{pq}(u, v)$. Т.о., получаем, что $\bar{R}(u, v) = \bar{R}_{pq}(u, v) = \bar{r}_{pq}(u, v)$ для точек $(u, v) \in \Delta_{pq}$. Это значит, что поверхность

с параметрическим уравнением $\bar{R}(u, v)$ при $u_p \leq u \leq u_{p+1}$, $v_q \leq v \leq v_{q+1}$ совпадает с поверхностью соответствующей pq -секции.

Если в формулу (5) подставить выражения (4), то получим

$$\begin{aligned} \bar{R}(u, v) = & \bar{R}_{00}(u, v) + \sum_{i=1}^{n-1} (\bar{R}_{i0}(u, v) - \bar{R}_{i0}(u_i, v)) + \\ & + \sum_{j=1}^{m-1} \left(\left(\bar{R}_{0j}(u, v) + \sum_{i=1}^{n-1} (\bar{R}_{ij}(u, v) - \bar{R}_{ij}(u_i, v)) \right) - \left(\bar{R}_{0j}(u, v_j) + \sum_{i=1}^{n-1} (\bar{R}_{ij}(u, v_j) - \bar{R}_{ij}(u_i, v_j)) \right) \right) \end{aligned} \quad (8)$$

После перегруппировки слагаемых получаем

$$\begin{aligned} \bar{R}(u, v) = & \bar{R}_{00}(u, v) + \sum_{i=1}^{n-1} (\bar{R}_{i0}(u, v) - \bar{R}_{i0}(u_i, v)) + \sum_{j=1}^{m-1} (\bar{R}_{0j}(u, v) - \bar{R}_{0j}(u, v_j)) + \\ & + \sum_{j=1}^{m-1} \sum_{i=1}^{n-1} (\bar{R}_{ij}(u, v) - \bar{R}_{ij}(u, v_j) - \bar{R}_{ij}(u_i, v) + \bar{R}_{ij}(u_i, v_j)) \end{aligned} \quad (9)$$

Т.о., формула (9) (или (8)) с учетом (3) дает аналитическое представление составной поверхности, если параметрические уравнения ее секций $\bar{r}_{ij}(u, v)$ заданы. Отметим также, что точки (u, v) , лежащие вне прямоугольника $[u_0, u_n] \times [v_0, v_m]$, отображаются в граничные точки составной поверхности.

Обычно интервалы изменения аргументов (u, v) вектор-функций $\bar{r}_{ij}(u, v)$, представляющих уравнения секций, заданы на произвольном прямоугольнике $u_0^{ij} \leq u \leq u_1^{ij}$, $v_0^{ij} \leq v \leq v_1^{ij}$. Чтобы эти прямоугольники согласовать с соответствующими прямоугольниками $[u_i, u_{i+1}] \times [v_j, v_{j+1}]$ предварительно перед использованием формул (3) в уравнениях поверхностей $\bar{r}_{ij}(u, v)$ следует сделать замену переменных $u = u_0^{ij} + \frac{u_1^{ij} - u_0^{ij}}{u_{i+1} - u_i} (u' - u_i)$, $v = v_0^{ij} + \frac{v_1^{ij} - v_0^{ij}}{v_{i+1} - v_i} (v' - v_i)$.

Формулы (8), (9) всегда представляют уравнение непрерывной поверхности (если исходные вектор-функции $\bar{r}_{ij}(u, v)$ непрерывны) даже, если не выполняется условие стыковки соседних секций. В этом случае результирующая поверхность будет совпадать только с начальной секцией поверхности $\bar{r}_{00}(u, v)$, а остальные секции будут деформированы и пристыкованы к начальной так, чтобы результирующая поверхность была непрерывной.

Уравнение составной поверхности, секции которой заданы уравнениями в явном виде. Пусть на плоскости XY задана ортогональная сеть $\omega_{xy} = [x_0, x_1, \dots, x_n] \times [y_0, y_1, \dots, y_m]$ и в ее ячейках определены функции $z_{ij}(x, y)$ ($i=0, 1, 2, \dots, n-1$, $j=0, 1, 2, \dots, m-1$), представляющие явные уравнения некоторых поверхностей. Известно, что эти поверхности пересекаются по кривым, расположенным над линиями сети ω_{xy} так, что выполняются условия непрерывности составной поверхности в виде: $z_{ij}(x_{i+1}, y) = z_{i+1j}(x_{i+1}, y)$ при $y_j \leq y \leq y_{j+1}$, и $z_{ij}(x, y_{j+1}) = z_{ij+1}(x, y_{j+1})$ при $x_i \leq x \leq x_{i+1}$. На прямоугольнике $[x_0, x_n] \times [y_0, y_m]$ требуется построить единое аналитическое уравнение

поверхности, составленной из секций поверхностей, имеющих уравнения $z_{ij}(x, y)$ над областями $\Delta_{ij} = [x_i, x_{i+1}] \times [y_j, y_{j+1}]$.

Для этого вначале составим уравнение секций поверхности в следующем виде

$$Z_{ij}(x, y) = z_{ij}(x_i + (x_{i+1} - x_i)P(x, x_i, x_{i+1} - x_i), y_j + (y_{j+1} - y_j)P(y, y_j, y_{j+1} - y_j)) \quad (10)$$

Каждая такая поверхность над областью прямоугольника $\Delta_{ij} = [x_i, x_{i+1}] \times [y_j, y_{j+1}]$ совпадает с исходной, поскольку для любой точки из прямоугольника $(x, y) \in \Delta_{ij}$ аргументы функции в правой части формулы (10) совпадают с x и y . Над полосами влево, вправо, вниз и вверх от этого прямоугольника поверхность $Z_{ij}(x, y)$ является цилиндрической, образованной переносом соответствующей граничной кривой секции вдоль одной из координатных осей. Над угловыми квадрантами результирующая функция $Z_{ij}(x, y)$ постоянна и равна значению исходной функции $z_{ij}(x, y)$ в углах секции.

Повторяя предыдущие построения, получаем следующую формулу

$$Z(x, y) = Z_{00}(x, y) + \sum_{i=1}^{n-1} (Z_{i0}(x, y) - Z_{i0}(x_i, y)) + \sum_{j=1}^{m-1} (Z_{0j}(x, y) - Z_{0j}(x, y_j)) + \sum_{j=1}^{m-1} \sum_{i=1}^{n-1} (Z_{ij}(x, y) - Z_{ij}(x, y_j) - Z_{ij}(x_i, y) + Z_{ij}(x_i, y_j)) \quad (11)$$

Она представляет уравнение составной поверхности, секции которой заданы явными уравнениями $z = z_{ij}(x, y)$ над ячейками сети Δ_{ij} . Доказательство формулы (11) полностью повторяет предыдущее доказательство с той лишь разницей, что вместо вектор-функций $\vec{r}_{ij}(u, v)$ и $\vec{R}_{ij}(u, v)$ рассматриваются скалярные функции $z_{ij}(x, y)$ и $Z_{ij}(x, y)$.

4. Заключение

С помощью введенной функции (1) и формулы (2) можно составлять единые аналитические выражения для уравнений кусочно-линейчатых поверхностей. Метод позволяет построить уравнения полных поверхностей таких тел как куб, пирамида, конус, уравнение поверхности цилиндра вместе с верхним и нижним основанием и уравнения поверхностей многих многогранников. Формулы (3), (9) и (10), (11) дают способ построения уравнений непрерывных составных поверхностей в форме единых аналитических выражений, если уравнения отдельных секций (криволинейных граней) поверхности заданы. Использование этих формул в системах символьной математики позволяет генерировать уравнения широкого класса кусочно-гладких непрерывных составных поверхностей. Алгоритмы настоящей работы реализованы в некоторых процедурах пакета расширения PscFunctions системы символьной математики MAPLE, созданного автором. Он представлен на интернет сайте www.maplesoft.com/products/thirdparty/PSCFunctions. Формулы настоящей

работы имеют применение в научной графике для изображения дву- и трехмерных тел и различных полей на их поверхности. Поверхности тел в системах инженерной компьютерной графики также могут описываться с использованием предложенных в работе уравнений.

ЛИТЕРАТУРА

1. Рвачев В.Л. Геометрические приложения алгебры логики. – Киев: Техніка, 1967. – 212с.
2. Рвачев В.Л. Теория R-функций и некоторые ее приложения. – Киев: Наук.думка, 1982.-552с.
3. Фокс Ф., Пратт М. Вычислительная геометрия. Применение в проектировании и на производстве. – М.: Мир, 1982.
4. Роджерс Д., Адамс Дж. Математические основы машинной графики. – М.: Мир, 2001.
5. Доля П.Г. Параметрические уравнения кусочно-гладких непрерывных кривых.// Вестник Международного Славянского Университета. Харьков. Серия “Технічні науки”, т.5, 2002, №7.
6. Доля П.Г. Моделирование кусочно-гладких непрерывных функций и кривых.// Вестник Харьк. нац. ун-та., - 2005.- № 661. Сер. ”Математическое моделирование. Информационные технологии. Автоматизированные системы управления”, вып.4. – С.97-103.

Visual Semantic Query Formulation and Execution in UnIT-NET IEDI

E. Dzhurinsky, V. Ermolayev, N. Keberle, V. Vladimirov
Zaporozhye National University, Ukraine
JDevelop.com, Ukraine

UnIT-NET Infrastructure for Electronic Data Interchange (IEDI) is the multi-layered distributed software system for providing intelligent ontology-driven information retrieval from distributed, heterogeneous, autonomous information resources (IRs). IEDI is constructed according the ideology of mediator-wrapper architecture with single centralized mediator. The paper presents the results of evaluation of the UnIT-NET IEDI research prototype. Two autonomous distributed IRs - "University Entrant" IR of Zaporozhye National University and "Dean's Office" IR of V. Karazin Kharkiv National University were used as a testbed. Experiments have shown practical applicability of the approach proposed for the IEDI architecture. One of the stages of the evaluation procedure is the assessment of the quality of interface between users and IEDI mediator. The paper analyses lessons learned from the evaluation of UnIT-NET IEDI user interface component – Query Formulation Interface. The analysis proves the usability of the QFI; however there are refinements which have still to be considered in the future work.

1. Introduction

The goal of a data integration system is to provide a uniform interface to various data sources [1], and to enable users to focus on specifying the questions they have.

UnIT-Net¹ IEDI is the software infrastructure providing for the Electronic Data Interchange between the Universities and the State Bodies of Ukraine. More precisely, IEDI is the multi-layered distributed software system comprising the software servers, services, components and tools for providing intelligent ontology-driven information retrieval from distributed, heterogeneous, legally and physically autonomous IR in the frame of the organizational network of the National Higher Education System.

The task of IEDI mediator developed within UnIT-Net project is to provide uniform, consistent and user-friendly interface to retrieve information from distributed autonomous information resources (IRs).

The paper reports on the evaluation of UnIT-NET IEDI research prototype implementation. One of the stages of the evaluation procedure is the assessment of the quality of interface between users and IEDI mediator. The paper presents the study of capabilities of Query Formulation Interface (QFI).

QFI – is a graphical tool which is the UnIT-NET IEDI user interface – uses visual presentation of mediator domain ontology and information resources' ontologies in the form of a tree and in the form of graph; it allows to add concepts, properties and

¹ UnIT-Net: IT in University Management Network. TEMPUS/TACIS project MP-JEP-2010-2003.
<http://www.unit-net.org.ua/>

additional constraints to a user query under construction; it stores user query using the concept of "basket"; it generates initial query to the mediator, initiates sub-queries extraction for the underlying IRs, collects responses and visually presents the responses to the user.

The paper is structured as follows: Section 2 discussed the IEDI framework and its main tasks; Section 3 describes IEDI environment chosen to bring the framework to life; Section 4 describes QFI in details; in the Section 5 presented are the results of experiments with IEDI; Section 6 presents the analysis of experiment results; Section 7 surveys the related work; Section 8 concludes the paper.

2 UnIT-Net IEDI Framework

From the variety of organizational platforms for integration of heterogeneous data sources (mediator-wrapper [1], federated database systems [2] etc) for UnIT-NET IEDI purposes only mediator-wrapper architecture gives the balance of centralized control and to a certain end a freedom in work with autonomous IRs.

Shortly describe the UnIT-NET IEDI architecture (see Fig. 1, also [3], [4]).

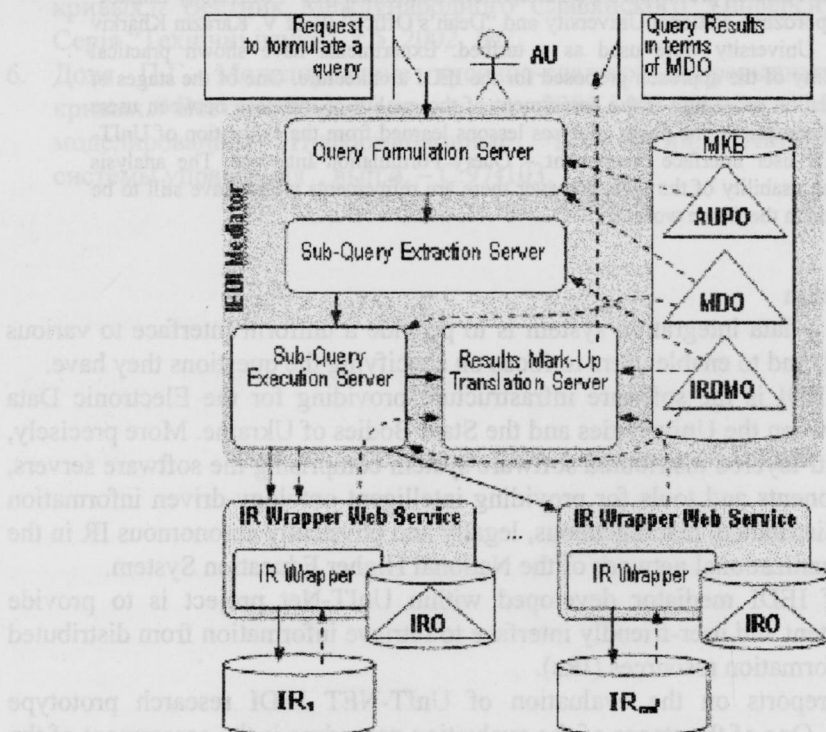


Figure 1. The reference architecture of the UnIT-Net IEDI

There is one IEDI mediator, its main tasks are:

- querying distributed autonomous semantically heterogeneous IRs (including user query formulation, query rewriting, results mark-up)
- registering IR
- maintaining coherent semantic descriptions

IRs are autonomous, distributed, heterogeneous and communicate with IEDI mediator by means of intelligent IR wrappers implemented as web services.

Each IR due to its autonomy has its own representation of the conceptualization of the domain, formally specified by IR conceptual schema. The hierarchy of IEDI ontologies has been developed to integrate these local views into the global view of the mediator.

Two main approaches to integrate data and answer queries without materializing a global schema [5] are “global-as-view” and “local-as-view”. Global-as-view approach prescribes for every domain conceptual schema concept to have a view over the IRs. On the contrary, local-as-view approach provides for each concept of an IR a view over the domain conceptual schema. Both approaches have their advantages and drawbacks; e.g. global-as-view is tolerant to answering complex queries and less flexible for maintaining frequently changed IRs; local-as-view approach is flexible with respect to changes in IRs, but its operation time increases exponentially to the number of IRs involved.

UnIT-NET IEDI explores “global-as-view” approach [6] and query rewriting technique ([7], [8]) to reformulate queries over IRs.

Domain knowledge is structured in the hierarchy of ontologies (see also [4], [9]). Each IR has own Information Resource Ontology (IRO), containing all essential (from the point of view of IR provider) concepts and properties of underlying IR domain. The goal of Mediator Domain Ontology (MDO) is to collect and merge all the knowledge from IROs, possibly adding necessary levels of abstraction.

Rewriting technique [1] makes use of mapping rules applicable to initial query to obtain new queries. In IEDI mapping rules are stored in mapping ontologies. IR-Domain Mapping Ontology (IRDMO, see Fig. 2a) stores mappings between IRO and MDO. IRO-IR Schema Mapping Ontology (see Fig. 2b) is constructed for every IR to store mappings of IR ontology terms to underlying structures of IR.

IRDMO is constructed in a way to contain the minimally necessary mappings for the MDO-IRO pairs. Only the mappings of non-inherited slots are stored for each MDO concept. IEDI mediator uses Late Binding technique [9] to detect all necessary slots of concepts during formulation of sub-queries to IROs.

All ontologies in UnIT-NET were developed with help of ontology editor Protégé 3.0² and are presented in OWL-DL [10], thus they have a sound and formal basis to verify consistence of ontologies.

3 IEDI Environment

All functional components of IEDI architecture are implemented in Java as web services. IR wrapper web services and IEDI mediator web service are run on Apache Tomcat³ Web server.

² Protégé 3.0 is available at the URL: <http://protege.stanford.edu>

³ Apache Tomcat Web server is available at the URL: <http://jakarta.apache.org/tomcat/index.html>

ConceptMappingRule		
MDOConcept	IRO	IROConcept
Person	Faculty	Student
Person	Faculty	Lecturer
Person	Entrants	Profile
Social-Category	Faculty	Category
...	...	...

a) IRDMO structure and the fragment of the knowledge base

SlotMappingRule				
MDOConcept	MDOSlot	IRO	IROConcept	IROSlot
Person	first-name	Faculty	Student	Name
Person	first-name	Entrants	Profile	name
Social-Category	descr-of-Social-Category	Faculty	Category	Category Name
Department	descrOfDepartment	Entrants	Department	descrOfDepartment
Department	descrOfDepartment	Faculty	Faculty	DescrOfFaculty

Rule			
concept	slot	table	field
Student	belongs To Category	tblStudents	ID Category
Student	belongs To Group	tblStudents	ID Group
Sheet	resultForStudent	tblSheets	ID Student
Sheet	MarkAtSheet	tblSheets	Mark
...	...	...	...

b) IRO-IR Schema Mapping Ontology structure and the fragment of the knowledge base

RelationshipMappingRule		
primaryTable	foreignTable	expression
tblCategories	tblStudents	tblCategories.ID=tblCategories.ID Category
tblGroups	tblStudents	tblGroups.ID=tblStudents.ID Group
tblStudents	tblSheets	tblStudents.ID=tblSheets.ID Student
...	...	...

Figure 2. The structures of mapping ontologies

Query Formulation Interface is made as separate GUI client desktop application working together with the mediator and operating on Apache Tomcat Web server as well. All the functional components of IEDI architecture (QFI, servers inside the mediator) may be distributed. Communications between functional components are based on SOAP and implemented with help of Apache SOAP processing service⁴. The architectural details of generic IR wrapper server are presented in [11].

Jena⁵ 2 API – Java-based framework for building Semantic Web applications – was chosen to store and query IEDI ontologies. It provides a programmatic environment for RDF, RDFS and OWL, including a rule-based inference engine⁶. Jena is an open-source project and has good support level from developers' team.

⁴ Apache SOAP processing service is available at the URL: <http://ws.apache.org/soap/>

⁵ Jena 2 API is available at the URL: <http://jena.sourceforge.net/>

⁶ Description of Jena features can be found at the URL: <http://simile.mit.edu/reports/stores/index.html>

Internal language for querying IEDI ontologies is RDQL [15], as far as RDQL has implementation in Jena 2 API. The comprehensive survey of all existing query languages for RDF is given in [12].

User interface for query formulation – QFI – was written as Java SWING graphical user interface application.

4 Visual Query Formulation with QFI

In IEDI it is not required that a user codes his or her queries in RDQL notation. Instead, the tool for ontology-driven Query Formulation Interface (QFI) is provided by the IEDI mediator.

The general requirement to QFI was that a user should have a visual ontology-based interface which allows him or her to interactively choose the terms of the MDO, to apply constraints to these terms and, thus, to formulate the query. The user query should be generated automatically from the set of the chosen ontology elements (concepts, properties constrained with the required terms).

The aim of visual query formulation interface in UnIT-Net is twofold.

First, QFI serves as the tool for domain ontologies navigation and learning. This allows user to be aware of the semantics of concepts and questions they will further pose to the mediator.

Second, QFI embeds all the mechanisms for subsequent querying the knowledge stored in mediator/resources. This allows user to pose queries with no extra knowledge on query languages inside the mediator.

Let's walk through the presentation of the following sample query “*Entrants from Zaporozhye, and their schools descriptions*” in QFI (see Fig. 3).

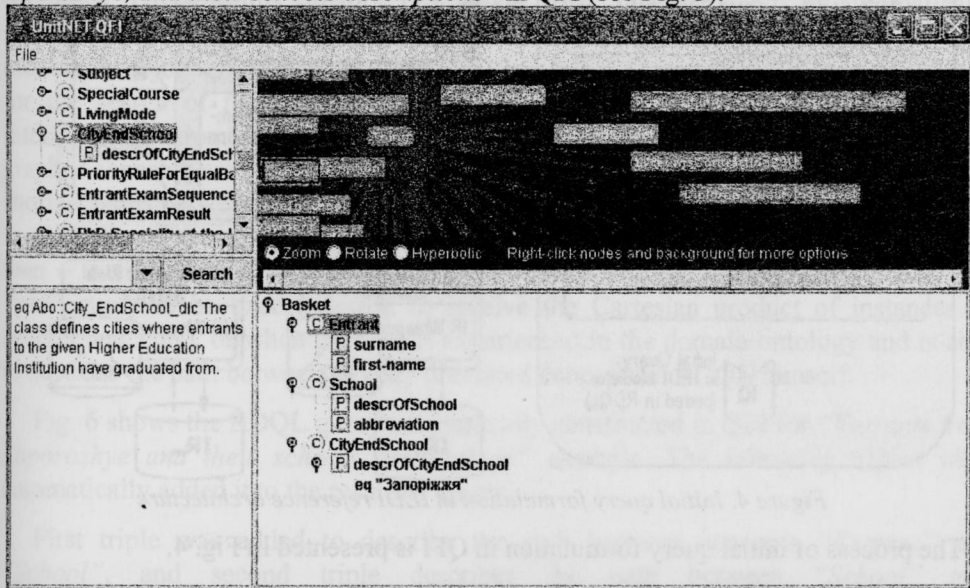


Figure 3. Visual presentation of the query “*Entrants from Zaporozhye and their schools descriptions*”

Left upper frame shows the taxonomy of loaded ontology (taken from .owl file). To ease the understanding of the ontology only datatype properties are presented visually in the QFI for each concept – only these properties may appear with value

restrictions in the query. For example, concept “School” has only two datatype properties – “descOfSchool” and “abbreviation”, and the other properties of this concept are object properties, relating “School” to other concepts in the ontology.

Right upper frame visualizes the ontology. QFI allows three modes of interface regulated with radio buttons and a scroll bar: “zoom” – to visually zoom the neighborhood of the concept; “rotate” – to rotate the whole structure around selected concept; “hyperbolic” [13] – to view the structure in the “fish eye” manner. The latter technique assigns more display space to a portion of the hierarchy while still embedding it in the context of the entire hierarchy. This supports a smooth blending between focus and context, as well as continuous redirection of the focus.

Left bottom frame shows comments (taken from `rdf:comment` tags) to the selected concept or slot. Right bottom frame visualizes the contents of the current query.

QFI according to the classification of visual query systems [14] explores diagram-based approach for domain knowledge and query structure visualization. QFI is suited to construct queries only from terms already existing in domain ontology. Query results’ presentation is form-based and query formulation is visualized as a kind of “unconnected path” visualization strategy [14].

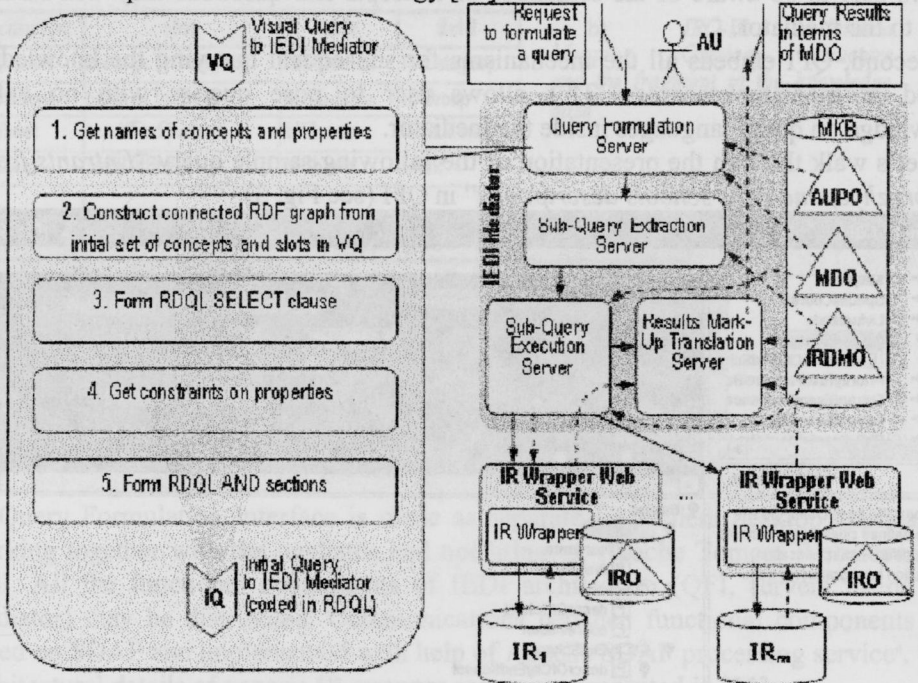


Figure 4. Initial query formulation in IEDI reference architecture

The process of initial query formulation in QFI is presented in Fig. 4.

Visual Query (VQ) is constructed by the user with help of visual interface of QFI. We borrow the concept of a “basket” from e-commerce applications to store elements of the visual query. Serializing Java objects into XML is done with help of XStream⁷.

⁷ XStream available at URL: <http://www.xml.com/pub/a/2004/08/18/xstream.html>

```

<set> <model><concepts><basketclass>
  <cls>
    <className>CityEndSchool</className>
  </cls>
  <children>
    <basketproperty>
      <parent class="basketclass" reference="." />
      <property>
        <propertyName>descrOfCityEndSchool</propertyName>
      </property>
      <constraints>
        <basketconstraint>
          <constraintAction>eq</constraintAction>
          <constraintValue>3anopax</constraintValue>
          <parent class="basketproperty" reference="." />
        </basketconstraint>
        <basketproperty> <children> <basketclass> ...
      </constraints>
    </basketproperty>
  </children>
</model>
<generatedRDQL>...</generatedRDQL></set>

```

Figure 5. Mark-up for Visual Queries basket content

The user adds necessary concepts to the basket, assigns additional restrictions on the values of properties. The fragment of the basket content for query “Entrants from Zaporozhye and descriptions of their schools” is shown in the Fig. 5.

After the query is built visually, QFI generates initial RDQL query. The main problem here comes from the fact that the user is free to choose arbitrary concepts from the ontology. It may happen that the graph constructed over these concepts is unconnected. As it was shown in [9] such graph corresponds to a “bad” initial query, resulting in redundant responses from IRs and rather artificial content of these responses. To avoid this situation QFI makes “bad” queries “good” by automatically searching the paths between concepts and incorporating necessary parts of ontology (concepts, object properties) into the resulting Initial Query (IQ). Dijkstra algorithm⁸ for the search of the shortest path in a graph has been used to implement this intelligent functionality. This algorithm applies to ontology represented as directed graph – concepts are treated as vertexes, object properties – as edges, and tries to find shortest paths between concepts from user query.

QFI allows to make changes directly to the RDQL presentation of the query, and then – to save a new query. This feature is useful in case when a user really needs “bad” query to be executed (e.g. to receive the Cartesian product of instances of several concepts), or when the user is experienced in the domain ontology and is able to describe the path between directly unrelated concepts in RDQL himself.

Fig. 6 shows the RDQL query automatically constructed in QFI for “Entrants from Zaporozhye and their schools descriptions” example. The following triples were automatically added into the resulting query:

First triple was added to describe the path between concepts “Entrant” and “School”, and second triple describes the path between “School” and “CityEndSchool”.

⁸ Realization of Dijkstra algorithm for shortest path search was taken from JDSL - Data Structures Library in Java, available at the URL: <http://www.cs.brown.edu/cgc/jdsl/>

When the initial query is ready, QFI connects to the IEDI mediator, passes this query and waits for the results. Then QFI performs results' mark-up in terms of the ontology whose concepts were in the query and brings the results to the user.

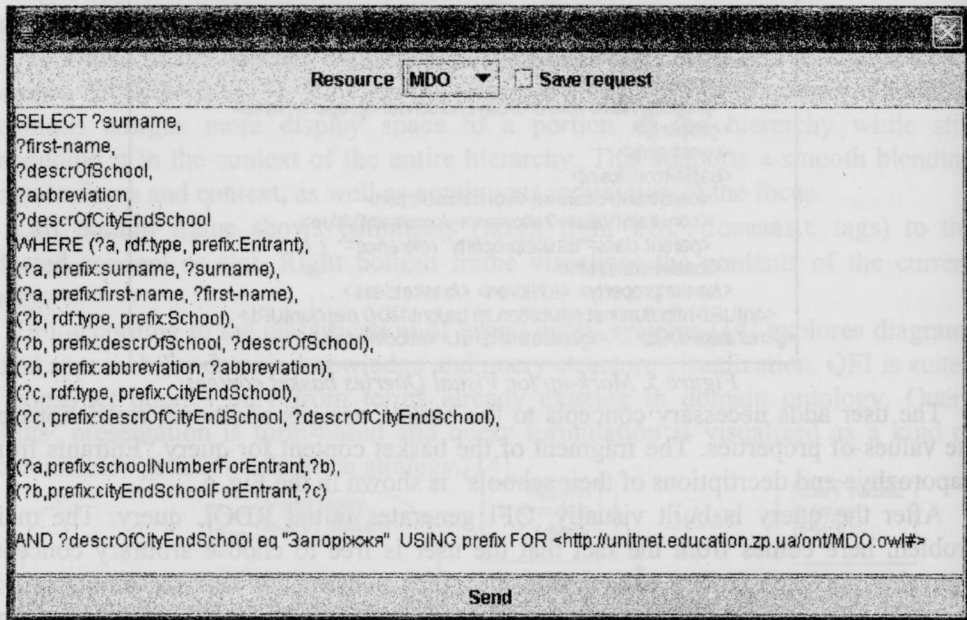


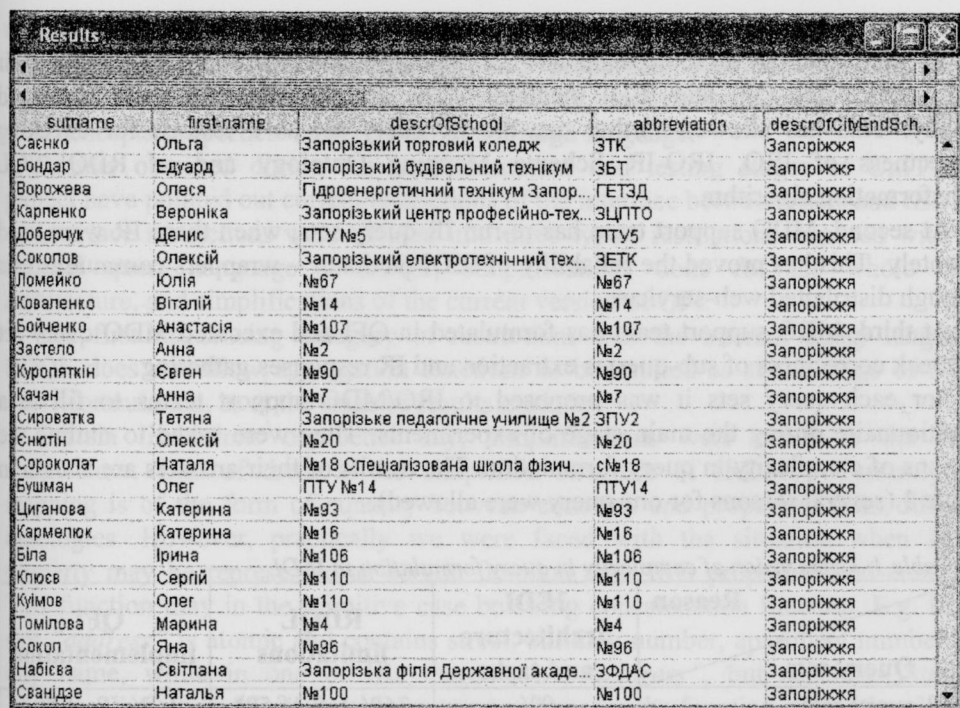
Figure 6. RDQL query generated with QFI

(?a, prefix:schoolNumberForEntrant, ?b) and
(?b, prefix:cityEndSchoolForEntrant, ?c)

In addition, QFI stores queries history to allow user to easily pose frequently used queries. Upon the user request QFI may store the particular responses from IRs as well.

As far as initial query is formulated for the whole MDO QFI initiates ontology-driven sub-queries extraction (see detailed description of the ODSQE algorithm in [9]) to obtain sub-queries to the registered IRs. These RDQL sub-queries are partial in the sense that not every MDO concept/property from the initial query should have correspondences in every registered IR. In spite of this, it is guaranteed that each partial RDQL query is correct and produces result. All peculiarities of sub-queries extraction are reported in [9]. Then IEDI mediator transforms each sub-query into the IR query and sends it to the IR. Current implementation of IEDI uses RDQL-SQL transformation algorithm worked out in [11]. IRO concepts and slots are mapped into table names and field names of the underlying relational IR scheme. This mapping according to the IEDI ontologies hierarchy is stored in IRO-IR Scheme Mapping ontology.

IR wrappers accept the sub-queries, initiate their execution and perform results mark-up in terms of respective IRO. The details of IR wrapper algorithms and realization are reported in [11].



surname	first-name	descrOfSchool	abbreviation	descrOfCityEndSch...
Сасенко	Ольга	Запорізький торговий коледж	ЗТК	Запоріжжя
Бондар	Едуард	Запорізький будівельний технікум	ЗБТ	Запоріжжя
Ворожева	Олеса	Гідроенергетичний технікум Запор...	ГЕТЗД	Запоріжжя
Карпенко	Вероніка	Запорізький центр професійно-тех...	ЗЦПТО	Запоріжжя
Доберчук	Денис	ПТУ №5	ПТУ5	Запоріжжя
Соколов	Олексій	Запорізький електротехнічний тех...	ЗЕТК	Запоріжжя
Ломейко	Юлія	№67	№67	Запоріжжя
Коваленко	Віталій	№14	№14	Запоріжжя
Бойченко	Анастасія	№107	№107	Запоріжжя
Застело	Анна	№2	№2	Запоріжжя
Куропяткін	Євген	№90	№90	Запоріжжя
Качан	Анна	№7	№7	Запоріжжя
Сироватка	Тетяна	Запорізьке педагогічне училище №2	ЗПУ2	Запоріжжя
Єнютін	Олексій	№20	№20	Запоріжжя
Сороколат	Наталя	№18 Спеціалізована школа фізич...	с№18	Запоріжжя
Бушман	Олег	ПТУ №14	ПТУ14	Запоріжжя
Циганова	Катерина	№93	№93	Запоріжжя
Кармелюк	Катерина	№16	№16	Запоріжжя
Біла	Ірина	№106	№106	Запоріжжя
Клюєв	Сергій	№110	№110	Запоріжжя
Куїмов	Олег	№110	№110	Запоріжжя
Томілова	Марина	№4	№4	Запоріжжя
Сокоп	Яна	№96	№96	Запоріжжя
Набієва	Світлана	Запорізька філія Державної акаде...	ЗФДАС	Запоріжжя
Сванідзе	Наталья	№100	№100	Запоріжжя

Figure 7. Response obtained for the query

E.g. query "Entrants from Zaporozhye and their schools descriptions" had the following response (see Fig. 7).

5 Experiments with QFI

Evaluation experiment was planned as follows. There were chosen two autonomous IRs registered into the IEDI mediator – "University Entrants" IR of Zaporozhye National University and "Dean's Office" IR of V. Karazin Kharkiv National University. "University Entrants" IR is run on MS SQL Server 2000 platform, "Dean's Office" IR uses MS Access.

The preparatory stage consisted of construction of experimental query set and installation of IR wrappers on the IRs.

The principle of organizing the query set was that IEDI mediator should correctly answer queries specific to the given IR, and additionally it should be able to answer queries over common concepts of MDO, which have several mappings in IRs.

Each IR support team was asked to prepare a set of queries intrinsic to the IR and formulated in natural language. The size of each query set was approximately 25 queries. As far as frequently asked queries are usually hardwired into the visual interface of information system, most of the queries in the set were taken from that interface. Resulting query sets were formed without any restrictions which might be influenced by the IEDI mediator architecture, ontologies hierarchy and the query formulation language.

Natural language query set over MDO concepts was prepared by MDO support team at ZNU. The main requirement to this set was to intensively use concepts common to both IRs and to explore taxonomy of concepts.

The main stage of the experiment was as follows.

At first, each IR support team has formulated and executed queries from their query sets using QFI and IEDI mediator. The correspondent IR itself was used locally. This has allowed further use the IR query set as a testbed for checking correctness of IRO, IRO-IR Schema Mapping Ontology and of RDQL-SQL transformation algorithm.

At second, MDO support team has re-run IR query sets, when these IR were used remotely. This has proved the reliability of IEDI mediator – wrappers communication through distributed web-services.

At third, MDO support team has formulated in QFI and executed MDO query set to check correctness of sub-queries extraction and IR responses gathering.

For each query sets it was proposed to IRO/MDO support teams to fill in a questionnaire during the main stage of experiments. They were asked to outline the reasons of complexity in query formulation. The results of their answers are shown in Table 1 (several reasons for one query were allowed).

Table 1. Users vision of complexity in query formulation in IEDI

Reason Query sets	IEDI architecture and tasks	RDQL limitations	QFI implementation
IR query sets	0%	36% (18 of 50)	16% (8 of 50)
MDO query set	36% (9 of 25)	36% (9 of 25)	60% (15 of 25)

Both IR support teams were strongly familiar with correspondent domain; however they used QFI with very little respect to its factual capabilities. They expected to get the results in the form similar to their IR native user interface, where the aggregate values (sums, average values etc) are of common use. That's why they've put the restrictions of RDQL on the first place of complexity reasons.

MDO support team was familiar with correspondent domains, and this team was better informed of factual capabilities of QFI and IEDI mediator. Their main task was to pose queries involving concepts from both IRs. They've outlined that IEDI mediator does not merge responses from IRs, but the most complex thing was to compare values from IRs between each other. This has lead to first place of QFI limitations for complexity reasons.

However it should be pointed out that only queries which contained value restriction of non-string data types were failed. The percentage of such queries in the sets was 4% (2 queries of 50).

Maximal delay of response for querying IR situated in the same network with the user and the IEDI mediator was proportional to the response size and did not exceed 5 seconds. For querying IR via remote access both to the mediator and an IR the maximal response delay caused by network connection was 50 seconds.

6 Lessons Learned and Future Research

Experiment with the user interface component of IEDI mediator shows that visual query formulation supplied with native language-oriented interface is very comfortable and easy to use. For novices in the domain it was a bit complex to

understand the context of a particular ontology concept, and they were forced to read comments to definitions of concepts and properties. Additional but expected inconvenience was related with the situation when users navigate and query ontology with concepts/properties' names given in a foreign language.

Results obtained during the implementation and testing stages of UnIT-NET project have pointed out current restrictions of the interface between users and IEDI.

Interface restrictions were divided into three types: restrictions imposed with the internal query language – RDQL, simplifications done for the whole IEDI architecture, and simplifications of the current version of QFI.

The first influencing category of restrictions was the internal query language. RDQL does not allow aggregation functions to be applied as at the stage of initial query formulation, as at the stage of results mark-up.

Certain limitations or, better say, simplifications were made to the IEDI architecture. One of them is the simplified structure of IRDMO. Currently the mapping is of the form of equality between concepts and properties from different ontologies. However, practically we were faced with the situation when MDO property may be represented as certain function over IRO properties and concepts. This function may in the primitive case be string concatenation function. E.g. MDO slot "address" is atomic and contains street, building number, apartment number and city name, while in one of IRs there is no "address", but separate "street", "building", "city", "flat" slots. In more complex case the function should involve the instances of property values. E.g., in one IR the "gender" property of a person has the values "male" and "female", and in the other IR – correspondent values were "m" and "f".

Moreover, as the whole IEDI architecture was not configured for results merging and fusion (it was beyond the scope of UnIT-NET), this has led to the situations when the user had to analyze the responses from the IRs one by one.

QFI limitations were outlined as follows:

- user cannot see object properties of concepts – this leaves the user "outside the kitchen" of initial query formulation
- user unfamiliar with RDQL cannot really approve query prepared by the QFI and he/she is forced to skip the query approval stage
- QFI currently supports value restrictions "equal" and "not equal" for datatype properties. However the whole set of value restrictions provided in the RDQL specification [15] can be easily added to the QFI
- QFI due to limitations of RDQL does not allow aggregation functions to be applied as at the stage of initial query formulation, as at the stage of results mark-up
- QFI currently does not allow to compare one property values with another property values, e.g. in the form " $concept_1.property_1 = concept_2.property_2$ "
- QFI does not allow to choose values of property from the list of values stored in the underlying IR
- QFI does not support all data types, which can be in the underlying domain. Currently only string values are allowed in value restrictions.

In spite of the fact that the highest quantity of negative user feedbacks (see Table 1) was obtained because of limitations of QFI, all these limitations are easy to overcome.

The situation with internal query language restrictions is different. At present, RDQL syntax is fixed. One solution is to choose the RDF query language already possessing all necessary features (at least, aggregation). Such languages as SeRQL[12] and SPARQL[16] seem good candidates for the purposes of UnIT-NET IEDI. Another solution for IEDI is to enhance Results Mark-Up Translation Server (see Fig.1). Queries with calculation of maximum/minimum/count/average values etc. may in that case be performed in two steps: computation of RDQL query result "as is", without aggregation, and then before presenting the result to the user – to calculate requested aggregation functions (on the mediator side, or at the client side).

Negative user feedbacks concerning the limitations of UnIT-NET IEDI architecture appear when the semantics of a user query requires merging of the IR responses. This problem is also of great importance for the IEDI and is one of future research direction for UnIT-NET consortium.

7 Related Work

Mediator-wrapper architecture [17] provides the syntactic and semantic interoperability [1] with help of intermediate layer (mediator) responsible for unification of user interface to distributed heterogeneous IRs presented with their wrappers.

The number of projects in intelligent information retrieval advocates ontology-based knowledge integration from autonomous heterogeneous distributed information resources. The review of state-of-the-art in this direction one can find in [4], [9], [18] and [19].

Functionality and aesthetic presentation of user interface influences future use of the whole system. As it was mentioned in [14], *"...It is worth noting that most people interacting with computers see only the system interface. Thus, it becomes a very important component of a software system from the design phase onwards."* There are a lot of investigations focused on visual query system development for databases. The detailed survey and classification of approaches in that field may be found in [14].

Such approaches are rather strict – they assume that user query may consist only of terms already described in the domain.

However, last decade research projects pay more attention to intelligent query interfaces in order to best capture the semantics of a user query. In the variety of proposed techniques outline the following.

User profiles (see a survey in [20]) are often considered as storages of user personal knowledge.

Intensional navigation in SEWASIE ([21], [22]) is used for incremental user query construction through domain ontology navigation. SEWASIE approach allows refinement of existing query terms or even invention of new query terms basing on existing ontology terms and Boolean connectors. Resulting query is then subjected to satisfiability checking w.r.t. domain ontology with help of a reasoning system supporting domain ontology definition language.

Query rewriting techniques [7], [8], [20] apply a set of rules to reformulate user query into terms of underlying resources.

8 Conclusions

The paper reports on the results of evaluation of the UnIT-NET IEDI research prototype. The focus of the paper is the intelligent visual interface – QFI – the graphical tool aiming to assist a user to formulate his/her queries to the resources registered to IEDI. The evaluation of the proposed solutions of IEDI architecture is made experimentally by the members of UnIT-NET consortium. Experiments have shown practical applicability of the approach proposed for the IEDI architecture. One of the stages of the evaluation procedure is the assessment of the quality of interface between users and IEDI mediator. The paper analyses lessons learned from the evaluation of UnIT-NET IEDI user interface component – Query Formulation Interface. The analysis proves the usability of the QFI; however there are refinements which have still to be considered in the future work.

Acknowledgements

The authors would like to express their gratitude to the members of the UnIT-Net project consortia for their collaborative help in bringing the reported results to life.

REFERENCES

1. Wiederhold G., Genesereth M. The conceptual basis for mediation services. IEEE Intelligent Systems, September/October, 1997, 38-47.
2. Sheth A. and Larson J. Federated Database Systems for Managing Distributed, Heterogenous, and Autonomous Databases. ACM Computing Surveys, 22:3, 183-236, ACM Press, 1990.
3. Ermolayev, V., Spivakovsky, A., Zholtkevych, G.: UnIT-NET IIDE: Infrastructure nationale ukrainienne pour l'intraéchange de données électroniques. Colloque National de la Recherche Universitaire dans les I. U. T. Actes de Colloque, Tome 1. Sciences et Techniques de l'Ingenieur, Nice, May, 6-7, 2004, p. 113-121.
4. Ermolayev, V., et al.: The Infrastructure for Electronic Data Interchange. Reference Architecture Specification. Version 1.0. UNIT-NET Deliverable No D2.2.D.1. URL: <http://www.compscipreprints.com/comp/Preprint/eva/20040228/1>
5. Xu Li, Embley D.W. Combining the Best of Global-as-View and Local-as-View for Data Integration. 3-d Intl. Conference on Information Systems Technology and its Applications (ISTA'2004), Salt Lake City, Utah, USA, July 15-17, 2004, 123-136.
6. Chawathe S. et al. The TSIMMIS project: Integration of heterogeneous information sources. Proc. of the 10th Meeting of the Information Processing Society of Japan, Tokyo, Japan, 1994, 7-18.
7. Baader F. et al.: Rewriting concepts using terminologies. 7th Intl. Joint Conf. on Principles of Knowledge Representation and Reasoning (KR 2000), 2000, 297-308.
8. Lattes V.; Rousset M.-C.: The Use of CARIN Language and Algorithms for Information Integration: The PICSEL System. Intl. J. of Cooperative Information Systems, 9(4), 2000, 383-401.
9. Ermolayev. V., Keberle, N., Shapar, V., Vladimirov, V.: Ontology-Driven Sub-Query Extraction for Distributed Autonomous Information Resources in UnIT-Net

- IEDI. 3-d Intl. Conference on Information Systems Technology and its Applications (ISTA'2004), Salt Lake City, Utah, USA, July 15-17, 2004, 137-150.
10. OWL Web Ontology Language Reference. W3C Proposed Recommendation, 15 December 2003. URL: <http://www.w3.org/TR/owl-ref/> (last checked: 08.07.2005)
 11. Ermolayev, V., Keberle, N., Shapar, V., Vladimirov, V.: Semantically Reinforced Web Services for Wrapping Autonomous Information Resources. Bulletin of V. Karazin Kharkiv National University, – 2004. – № 629. Series “Mathematical Modelling. Information Technology. Automated Control Systems”, Issue 3. – p.56-69.
 12. RDF Query Survey. URL: <http://www.w3.org/2001/11/13-RDF-Query-Rules/> (last checked: 08.07.2005)
 13. Lamping, L.; Rao, R.; and Pirolli, P. A Focus+Context Technique Based on Hyperbolic Geometry for Visualizing Large Hierarchies. Proc. of the ACM SIGCHI Conference on Human Factors in Computing Systems, 401-408. New York: ACM, 1995.
 14. Catarci T., Costabile M.F., Levialdi S., Batini C. Visual Query Systems for Databases: A Survey. Journal of Visual Languages and Computing, 8(2), 1997, 215–260.
 15. RDQL – A Query Language for RDF. W3C Member Submission, 9 January 2004, URL: <http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/> (last checked: 08.07.2005)
 16. SPARQL Query Language for RDF. W3C Working Draft, 19 April 1995. URL: <http://www.w3.org/TR/2005/WD-rdf-sparql-query-20050419/> (last checked: 08.07.2005)
 17. Wiederhold G. Mediators in the Architecture of Future Information Systems. IEEE Computer, 25, 3(March), 1992, 38-49.
 18. Keberle N. Heterogeneous database and knowledge-based integrating systems: the review. Visnyk of the Lviv University. Series Applied mathematics and Computer Science. 2002, No.4, 163-172 (In Ukrainian).
 19. Wache, H. et al.: Ontology-Based Integration of Information - A Survey of Existing Approaches. In: (A. Gomez-Perez, M. Gruninger, H. Stuckenschmidt, M. Uschold) Proceedings of the IJCAI-01 Workshop on Ontologies and Information Sharing, Seattle, USA, August 4-5, 2001, 108-118.
 20. Ermolayev, V., Keberle, N., Plaksin, S., Vladimirov, V.: Capturing Semantics from Search Phrases: Incremental User Personification and Ontology-Driven Query Transformation. In: Proc. of the 2-nd Int. Conf. on Information Systems Technology and its Applications (ISTA'2003), Kharkiv, Ukraine, June 19-21, 2003, 9-20.
 21. Dongilli P., Franconi E., Tessaris S. Semantics Driven Support for Query Formulation. Proc. of the 2004 Intl. Workshop on Description Logics (DL 2004), vol. 104, Whistler, BC, Canada, June 2004.
 22. Catarci T. et al. Usability evaluation tests in the SeWAsIE (SEmantic Webs and AgentS in Integrated Economies) project. Proc. of the 11th Intl. Conf. on Human-Computer Interaction (HCI 2005), 2005.

Алгоритм работы информационного агента в распределенной вычислительной системе и его модель на базе нечетких Е-сетей

Е. В. Дуравкин, Амер Таксин Каламех Абу Джаккар

Харьковский национальный университет им. В. Н. Каразина, Украина

In the article questions of the program intellectual agent model designs by means of fuzzy E-nets were considered. The structure of the node of the distributed computer system which used soft agents was suggested. The algorithms of the agents for data searching and information base analysis which used elements of artificial intelligence were designed. The described agent models were designed by means of fuzzy E-nets. The methods of the performance evaluation of the designed algorithms on models were showed.

1. Постановка проблемы. При разработке распределенных программных систем зачастую модули, выполняющие управленческие функции, реализуют в виде агентов. Основной задачей таких агентов является повышение эффективности управления ресурсами системы, частью которой они являются, путем введения элементов интеллектуального управления в соответствующие алгоритмы. Примерами таких ресурсов могут быть не только информация, но и пропускная способность, память, процессорное время [1]. В настоящее время уделяется большое внимание разработке систем такого класса, однако недостаточное внимание уделяется вопросам совершенствования средств исследования и анализа данных систем. Такое положение дел приводит к тому, разработчики систем, построенных на базе передовых технологий искусственного интеллекта не имеют в своем распоряжении средств для оценки их эффективности на этапах проектирования и отладки, что отрицательно сказывается на их качестве и сроках внедрения.

2. Анализ публикаций. Зачастую для анализа работы интеллектуальных агентов используются традиционные средства имитационного моделирования (автоматы, цепи Маркова, агрегатные схемы) [2,3], в более простых случаях такие системы пытаются описать аналитическими выражениями [4]. Использование данного подхода зачастую приводит к тому, что исследователи получают неверные результаты в следствии ограниченности выбранного средства анализа. Получение неверных результатов в первую очередь связано с тем, что используемые средства не в состоянии учесть основные особенности предметной области, такие как наличие нескольких управляющих и информационных потоков, распределенная система управления, наличие в управляющих алгоритмах элементов искусственного интеллекта.

Как было показано в [3,5] сети Петри и их расширения в значительной мере избавлены от указанных недостатков, а описанное в [6] расширение Е-сетей, так же позволяет учесть наличие элементов искусственного интеллекта.

3. Постановка задачи. В качестве примера, на котором показаны возможности нечетких Е-сетей при разработке моделей в данной предметной области, будет рассмотрена распределенная вычислительная система, структурная схема элемента (узла) которой представлена на рис. 1.



Рис. 1. Структурная схема узла распределенной вычислительной системы

Рассматриваемая система состоит из равноправных узлов, задача которых заключается в обработке данных, в соответствии с алгоритмами вычислительного модуля. В каждом вычислительном модуле могут быть реализованы различные алгоритмы, которые будут требовать данные независимо друг от друга. Для обеспечения потребностей вычислительного модуля в данных в каждом узле предусмотрены программные агенты, осуществляющие поиск необходимой информации на других узлах системы и анализирующие информацию, находящуюся на собственном узле для удовлетворения запросов других узлов. Функции взаимодействия узлов между собой реализованы посредством модуля связи. В некоторых случаях данный модуль так же может состоять из соответствующих программных агентов, однако данные вопросы не являются материалом настоящей статьи.

4. Основная часть. Для начала более подробно рассмотрим алгоритмы работы введенных программных агентов. Задачей агента поиска данных является обеспечение потребностей вычислительного модуля в необходимых данных. В случае, когда для продолжения дальнейших расчетов вычислительному модулю недостаточно данных, находящихся в собственной информационной базе он формирует запрос к агенту поиска данных. В запросе вычислительный модуль должен отразить признаки необходимых ему данных (в данном случае все данные в системе должны быть представлены в виде объектов, содержащих как переменные, так и методы работы с ними). На основании полученного запроса агент поиска данных формирует запрос на поиск данных в системе. После чего передает его модулю связи. По результатам запроса могут быть получены либо положительные, либо отрицательные ответы. В случае получения положительного ответа необходимые данные передаются вычислительному модулю. В случае получения отрицательных ответов от всех узлов системы, агент поиска данных запрашивает дополнительную информацию у вычислительного модуля о данных (например, в ходе работы каких функций

они могут быть получены). С помощью дополнительной информации данных агент поиска формирует новый, расширенный запрос. По результатам такого запроса либо определяется узел – источник необходимых данных и ему высылается заявка на их отправку текущему узлу после генерации, либо принимается решение о том, что вычислительному модулю текущего узла необходимо самому производить их вычисление. Особенностью последнего этапа является то, что по дополнительному запросу несколько узлов системы могут дать положительный ответ. Данная ситуация может возникнуть в силу недостаточно полной информации в расширенном запросе, либо в связи со схожестью работы различных вычислительных функций выполняемых в текущий момент времени на разных узлах. Следовательно, при получении нескольких положительных ответов агент поиска данных должен определить наиболее вероятного генератора искомых данных. Выбор узла – наиболее вероятного генератора искомых данных может быть основан на максимальном совпадении информации содержащейся в запросе, и информации, полученной от вычислительного модуля. Структурная схема алгоритма работы агента поиска данных приведена на рис. 2.

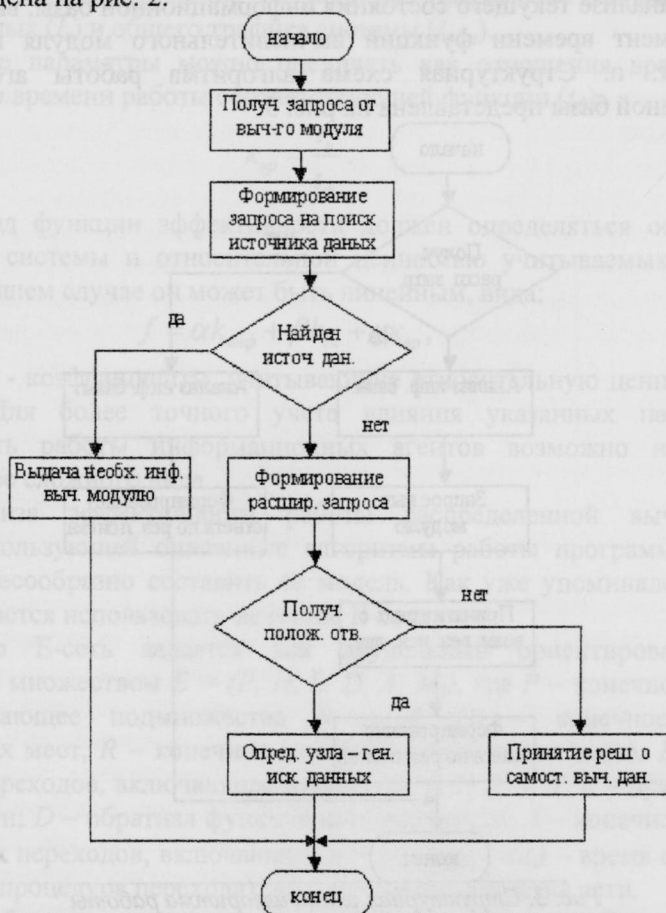


Рис. 2. Структурная схема работы агента поиска данных

Как уже упоминалось, для обработки запросов поступающих от агентов поиска данных на каждом узле реализованы агенты анализа информационной

базы. Задачей агентов является определение наличия необходимых данных, согласно полученного запроса и их отправки запрашивающему узлу. Для решения данной задачи предлагается следующий алгоритм работы агента. Агент может получить запросы двух типов: обычный и расширенный. При получении обычного запроса агент производит поиск необходимых данных в информационной базе своего узла. Поиск производится по информации, содержащейся в запросе (имя или описание объекта). В случае удачного поиска агент высылает требуемые данные узлу – получателю, в обратном случае – отрицательный ответ. При получении расширенного запроса агент производит поиск не только в информационной базе своего узла, но так же посылает запрос вычислительному модулю на предмет вычисляемых данных и работающих функций в настоящий момент времени. После обработки информации, полученной от вычислительного модуля и состояния информационной базы, агент анализа принимает решение, является ли его узел – узлом-генератором искомых данных или нет. И отправляет соответствующий ответ запрашивающему узлу. Принятие решения в данном случае может быть основано на анализе текущего состояния информационной базы, выполняемых в текущий момент времени функций вычислительного модуля используемых данных и т. п. Структурная схема алгоритма работы агента анализа информационной базы представлена на рис. 3.

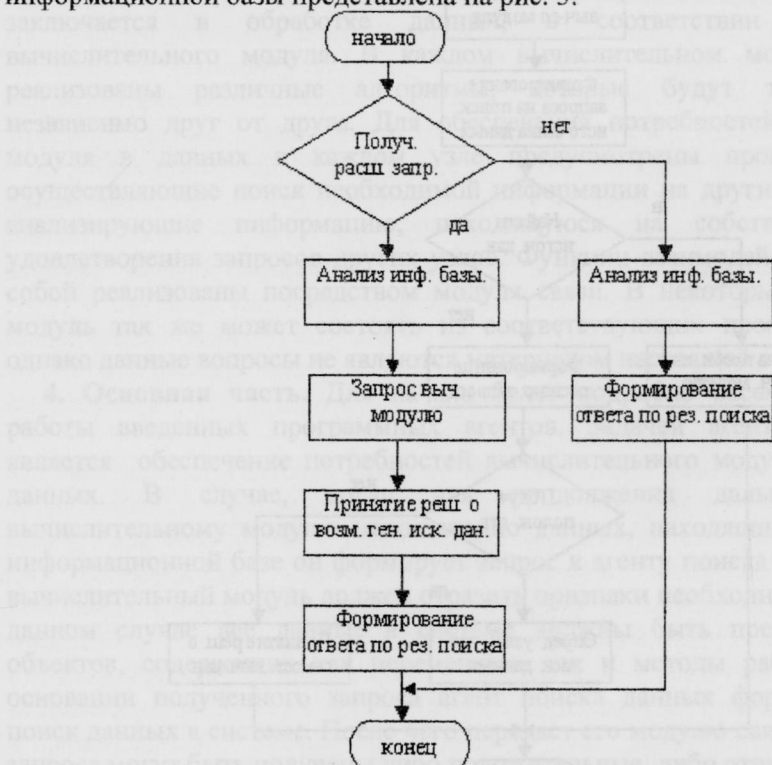


Рис. 3. Структурная схема алгоритма работы агента анализа информационной базы

Основной задачей разработанных алгоритмов является обеспечение максимального значения показателя эффективности системы подготовки данных для вычислений. В качестве такого показателя эффективности может выступать некая функция f которая связывает между собой такие характеристики работы как время поиска данных, полнота доставленных данных и загруженность системы связи. В качестве характеристики полноты доставленных данных можно использовать соотношение:

$$K_{инф} = \frac{V_p}{V_{тр}}, \quad (1)$$

где V_p – реальный объем используемых данных из всех доставленных;

$V_{тр}$ – объем данных, который необходим для работы текущей функции.

Оценить загруженность системы связи можно используя коэффициент

$$K_{св} = \frac{I_3}{I_{общ}}, \quad (2)$$

показывающий соотношение служебного трафика в системе по поиску текущих данных (I_3) и общего трафика системы ($I_{общ}$).

Временные параметры можно оценивать как отношения времени поиска данных (t_n) ко времени работы соответствующей функции (t_ϕ):

$$K_{вр} = \frac{t_n}{t_\phi}. \quad (3)$$

Общий вид функции эффективности должен определяться особенностями исследуемой системы и относительной ценностью учитываемых параметров. Так в простейшем случае он может быть линейным, вида:

$$f = \alpha K_{инф} + \beta K_{св} + \gamma K_{вр}, \quad (4)$$

где α , β , γ – коэффициенты, учитывающие относительную ценность каждого параметра. Для более точного учета влияния указанных параметров на эффективность работы информационных агентов возможно использование функций более сложного вида.

Для анализа эффективности работы распределенной вычислительной системы, использующей описанные алгоритмы работы программных агентов наиболее целесообразно составить ее модель. Как уже упоминалось, для этой цели предлагается использовать нечеткие Е-сети.

Формально Е-сеть задается как двудольный ориентированный граф, описываемый множеством $E = (P, H, L, D, A, M_0)$, где P – конечное множество мест, включающее подмножества B и R , (B – конечное множество периферийных мест, R – конечное множество решающих мест); H – конечное множество переходов, включающее множества T, F, J, X, Y ; L – прямая функция инцидентности; D – обратная функция инцидентности; A – конечное множество характеристик переходов, включающее $a = (\tau(a_i), q)$, ($\tau(a_i)$ – время срабатывания перехода, q – процедура перехода); M_0 – начальная разметка сети.

При разработке моделей с использованием аппарата Е-сетей ситуации принятия решения и выбора моделируются с помощью управляемых МХ и МУ – переходов. Процесс перехода в одно из возможных состояний описывается

предикатом перехода. Наиболее часто предикаты управляемых переходов имеют либо вероятностную функцию срабатывания, либо функцию, зависящую от состояния вектора атрибутов метки, других мест сети или внешних воздействий.

В случае разработки моделей систем использующих элементы интеллектуального управления предложено дополнить возможности описания предикатов управляемых переходов [6]. Используя введенные дополнения в аппарат Е-сетей модель работы информационных агентов рассматриваемой системы может быть представлена в следующем виде. Граф Е-сети, моделирующей работу агента поиска данных представлен на рис. 4.

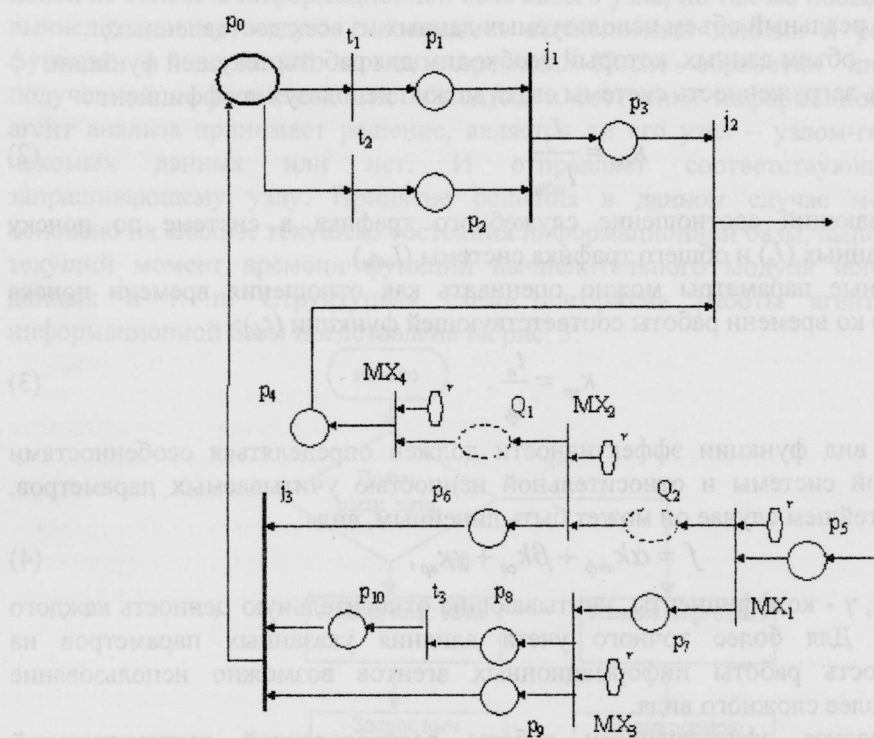


Рис. 4 Граф Е-сети, моделирующей работу агента поиска данных

Элементы разработанной Е-сети соответствуют следующим состояниям и функциям агента. Места: p_0 – макро-место, моделирующее работу вычислительного модуля с агентом поиска данных, в данном макро-месте формируется вектор атрибутов метки, содержащий информацию о искомым данных; p_1 – получена информация от вычислительного модуля для формирования обычного запроса; p_2 – получена информация от вычислительного модуля для формирования расширенного запроса; p_3 – запрос готов для передачи модулю связи; p_4 – отправить заявку на доставку данных выбранному узлу; p_5 – получен ответ на запрос; p_6 – принято решение о необходимости самостоятельно формировать необходимые данные; p_7 – получен ответ на обычный запрос; p_8 – необходимо формирование расширенного запроса; p_9 – получен положительный ответ от одного из узлов системы; p_{10} –

запрос вычислительному модулю на выдачу информации для формирования дополнительного запроса.

Макро-места очереди: Q_1 – хранит метки с положительными ответами на расширенный запрос; Q_2 – содержит метки с ответами на расширенный запрос. Переходы: t_1 – получение информации от вычислительного модуля для формирования обычного запроса; t_2 – получение информации от вычислительного модуля для формирования расширенного запроса; t_3 – формирование запроса вычислительному модулю на дополнительную информацию. j_1 – формирование запроса на поиск данных; j_2 – передача запроса модулю связи; j_3 – передача информации вычислительному модулю по результатам поиска данных. MX_1 – определение на какой из запросов получен ответ; MX_2 – анализ ответов на расширенный запрос; MX_3 – анализ ответов на обычный запрос; MX_4 – выбор наиболее вероятного узла – генератора искоемых данных. Предикаты управляемых переходов определяются следующим образом: g_1 – выбор места в которое будет передана метка определяется значением вектора атрибутов метки различающим ответ на обычный запрос и расширенный запрос; g_2 – в случае если в макро-месте очереди Q_2 у всех меток вектора атрибутов содержат элементы, соответствующие отрицательному ответу, после получения меток от всех узлов одна метка передается в место p_8 , что соответствует необходимости самостоятельно генерировать искомые данные; g_3 – если от всех узлов системы получены отрицательные ответы, то метка передается в место p_8 ; g_4 – по анализу векторов атрибутов меток, находящихся в макро-месте очереди Q_1 определяется номер узла – наиболее вероятного генератора искоемых данных и ему передается заявка на их доставку, остальные метки сбрасываются. При определении предикатов управляемых переходов наибольшие трудности возникают в случае перехода MX_4 . В данном случае сложность вызывают моделирование процедур определения наиболее вероятного узла – генератора необходимых данных. Для этого при формировании ответов на расширенный запрос, агент анализа информационной базы в векторе атрибутов метки должен установить соответствующее значение заданного поля, а так же в случае необходимости, значения дополнительных полей (в общем случае полей, характеризующих вероятность генерации искоемых данных, может быть несколько). В данном случае считается наиболее целесообразным использовать аппарат нечетких множеств, с помощью которых и будет определен предикат соответствующего перехода. Для этой цели должна быть задана некоторая функция предпочтения, значение которой будет определять наиболее предполагаемый узел генератор искоемых данных. Конкретный вид функции предпочтения должен определяться используемыми критериями в исследуемой системе.

Граф Е-сети, моделирующей работу агента анализа информационной базы представлен на рис. 5.

Элементы приведенного графа соответствуют следующим состояниям и функциям моделируемого агента. Места: p_1 – получен обычный запрос на поиск данных; p_2 – получен расширенный запрос на поиск данных; p_3 – по результатам анализа сформирован отрицательный ответ; p_4 – по результатам анализа сформирован положительный ответ; p_5 – анализ информационной базы по расширенному запросу; p_6 – запрос к вычислительному модулю по

выполняемым функциям; p_7 – получены результаты по расширенному запросу; p_8 – формирование положительного ответа по расширенному запросу; p_9 – формирование отрицательного ответа по расширенному запросу.

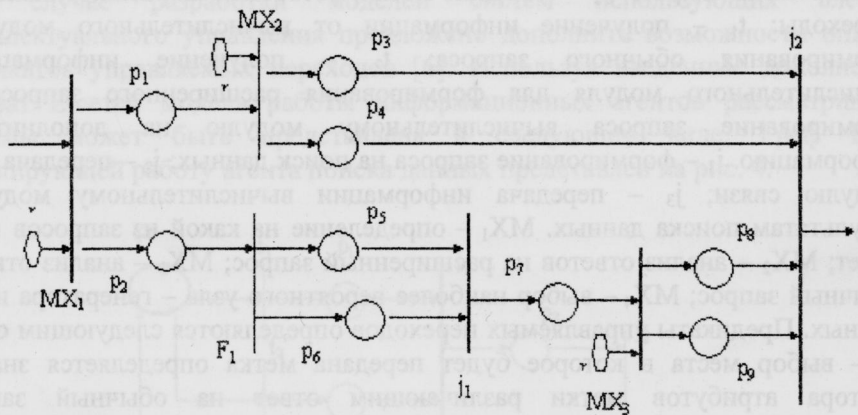


Рис. 5. Граф E-сети, моделирующей работу агента информационной базы

Переходы F_1 – выполнение запросов к информационной базе и вычислительному модулю; j_1 – получение ответов от к информационной базы и вычислительного модуля; j_2 – передача ответов модулю связи; MX_1 – распознавание поступившего запроса; MX_2 – формирование ответа на простой запрос; MX_3 – формирование ответа на расширенный запрос. Предикаты управляемых переходов определяются следующим образом: g_1 – по значению заданного элемента вектора атрибутов метки; g_2 – определяется состоянием элементов, моделирующих информационную базу; g_3 – определяется состоянием вектора атрибутов метки, находящейся в месте p_7 , а так же возможно состоянием неких внешних элементов. Особенностью определения предиката g_3 является то, что он совместно с функцией срабатывания перехода должен задать значения неких параметров вектора атрибутов метки, влияющих на определение узла – генератора данных.

Получение характеристик работы программных агентов и оценка значения функции эффективности (4) производится путем сбора статистики работы E-сети. Так для оценки полноты доставленных данных необходимо подсчитать количество запросов от вычислительного модуля для работы одной функции, а так же объем данных, доставленных избыточно. Данную информацию можно получить путем подсчета количества меток поступивших из макро-места p_0 и анализа элементов вектора атрибутов меток, содержащих положительные ответы на запросы. Оценка коэффициента загруженности системы связи находится путем подсчета числа меток, соответствующих сообщениям в системе и оценки их среднего числа, необходимого для выполнения одного запроса по поиску данных. Оценку временных параметров решения задач поиска данных можно производить путем анализа соответствующих элементов вектора атрибутов метки, инициирующей задачу (поступившей из макро-места p_0) и метки, соответствующей выдаче вычислительному модулю результатов поиска.

После статистической оценки членов функции эффективности производится нахождение ее оценки по соотношению (4).

5. Выводы. Таким образом видим, что разработка моделей работы программных агентов поиска данных и анализа информационной базы с использованием нечетких Е-сетей не представляет особых проблем. При использовании же других средств математического моделирования возникают различного рода трудности. Так например, при использовании аппарата цепей Маркова разработчики сталкиваются со сложностями описания несинхронных, недетерминированных процессов. Так же значительные трудности возникают в описании процесса принятия решения, если при этом в реальной системе используются элементы искусственного интеллекта. При использовании же автоматных моделей у разработчиков отсутствует возможность описания взаимодействующих параллельных процессов, что так же значительно снижает адекватность полученных результатов.

ЛИТЕРАТУРА

1. Распределенные системы. Принципы и парадигмы. – Э. Таненбаум, М. ван Стеен. – СПб.: Питер, 2003. – 877с.
2. Герасимов Б. М., Тарасов В.А., Токарев И.В. Человеко-машинные системы принятия решений с элементами искусственного интеллекта. – К.: Наукова думка, 1993. –183 с.
3. Применение микропроцессорных средств в системах передачи информации: Учеб. пособ. для вузов по сп-ти АСУ/ Я. Советов, О.И. Кутузов и др. – М.:Высш. шк., 1987. –256 с.:ил.
4. Пospelов Г. С. Искусственный интеллект – основа новой информационной технологии. – М.: Наука, 1988. – 280 с.
5. Леоненков А. В. Нечеткое моделирование в среде MATLAB и fuzzyTECH. – СПб.: БХВ – Петербург, 2003. 736с.
6. Дуравкин Е. В. Использование аппарата Е-сетей для моделирования систем управления с элементами искусственного интеллекта. Вісник Харківського національного університету. №629 Серія «Математичне моделювання. Інформаційні технології. Автоматизовані системи управління». С.5-10

One Way to Guarantee the Stable Behaviour of a Software System by Preventing Unauthorized External Intrusions

G. M. Zholtkevych, I. T. Zaretska

V. N. Karazin Kharkiv National University, Ukraine

A new software design pattern of architectural level called Chest is introduced in the paper. It allows guaranteeing the stable behaviour of a complex system by encapsulating its manager into a protecting shell class with static interface. Clients can use system functions only via Chest class which delegates real work to the system manager class. Full description of this pattern is presented in the paper. The description includes pattern structure, relationship between participants, problems of realization, and examples of code and cryptosystem application.

Introduction

The problem of good design and architectural solutions reuse is in the centre of software engineering research. Successive reuse of such solutions (including the programming code) is one of the key factors in providing software development efficiency and software product quality. In fact reuse ensuring in the process of software development was one of the uppermost goals when the object-oriented paradigm of analysis, design and programming was being created [1 – 3]. As practice shows the object oriented approach to the decomposition of knowledge domain (object oriented decomposition) in contrast to the functional decomposition makes it possible to recognize invariants of a structure and behaviour which are called software patterns [4]. As founders of the object-oriented paradigm stated determining the typical interactions between objects of the designed system can be regarded as one of its quality metrics. If this aspect is given focus on the design stage the architecture of the system gets more compact, easy and flexible than if patterns were ignored (see for example the foreword to [4]). This paper describes one software pattern of architectural level, which has been discovered by authors during their work in the TEMPUS TACIS MP JEP 23010-2003 project. This pattern allows developer to protect his software subsystem from access by any methods except for those allowed by him. Such solution guarantees stable behaviour of a system, which is defined by its implementation. It is especially important for the subsystems critical in view of their information security or in view of their effect on the system environment.

Problem description and main ideas

Let us consider some subsystem with definite number of functions and a manager to control its performance in accordance to definite logic and rules. For example computer aided manufacture systems or systems of information exchange crypto protection are of such kind. Exactly the stability of system behaviour is the crucial factor for such systems. It is important to find an architectural solution to protect your system

and its manager from external impact as well as to prevent any changes in the predefined behaviour of executable components of the system. One of such architectural solutions is offered in this paper. Its main idea is to prohibit any direct access to system functions and to its manager behaviour from external objects by encapsulating them into some "protective shell" which strictly regulates user rights. This solution can be regarded as a software pattern of system or architectural level. According to its purpose we call it Chest or Coffe.

Pattern description

We will use the standard scheme to describe the pattern [4].

Pattern name and group: **Chest**, system or architectural group.

Purpose: to guarantee security and stability of complex system performance.

Also known as: **Coffe**.

Motivation. Let us consider some system for cryptographic security of information exchange between hosts in a global network. The system itself has a number of functions using definite algorithms and interacting in definite way to secure information. Certainly it is not advisable and moreover quite dangerous to permit any user an immediate access to these system functions. It would not only aggravate his work but could cause breaking information integrity or secrecy. A natural solution to this problem is to introduce a manager class, which would control the complex process of interaction between system functions. But if any user is permitted an immediate access to the manager class he could try to crack the system by extending (inheriting) or just replacing the manager class. Certainly it is possible to restrict user access to the manager class only by "allowed" methods but still the problem of manager instances creating and destroying remains unsolved. To impose this responsibility on a user (i.e. to make public the manager class constructor and destructor) means to give a user an opportunity to decide independently when and which instance of the manager class to create. Such an "excessive freedom" does not contribute to the reliability of the system performance.

The proposed solution eliminates the mentioned above disadvantages due to a new class **CryptoSystem** that works like a protecting shell. It encapsulates the instance of the manager class so it is possible to create or destroy this instance or to execute its methods only from inside this **CryptoSystem** class. No user can directly access either the system functions or its manager but interacts with the cryptosystem only via allowed for him interface of the **CryptoSystem** class (fig. 1 at the next page). Such architecture deprives user any possibility to intrude into the system work.

Usability. This pattern is used if

- it is necessary to prohibit the direct access of a user to the system functions;
- the system functions interact in a complicated way while a user needs only simple standard interface to the system;
- the responsibility to control the system functions interaction is assigned to a separate manager class that should be protected from changing or replacing.

Structure.

Participants (fig. 2 at the next page)

Client – a user of the system who:

- interacts with the system only via the class **Chest** interface;

- has restricted access to the system like starting or stopping the system or some other standard actions;

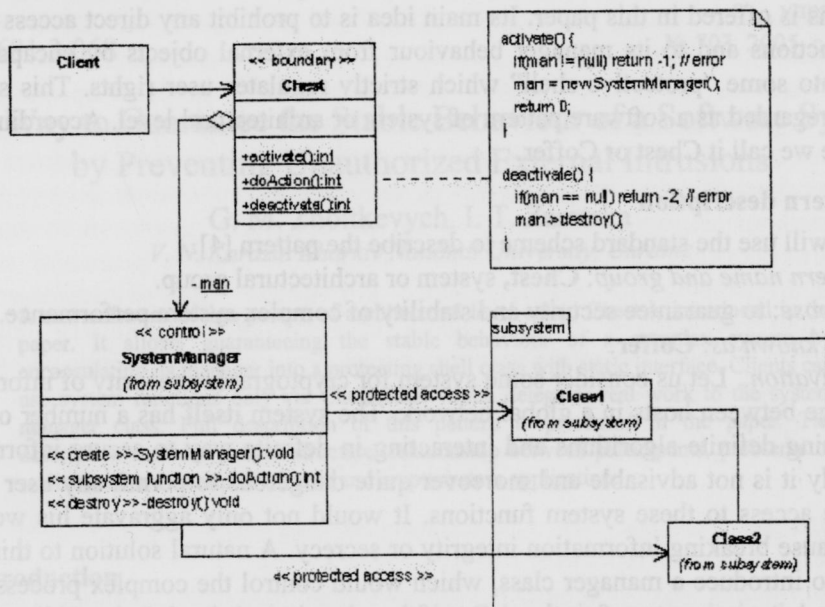


Fig. 1. Cryptosystem structure.

- knows neither about the internal structure of the system nor about the logic of its functions interaction.

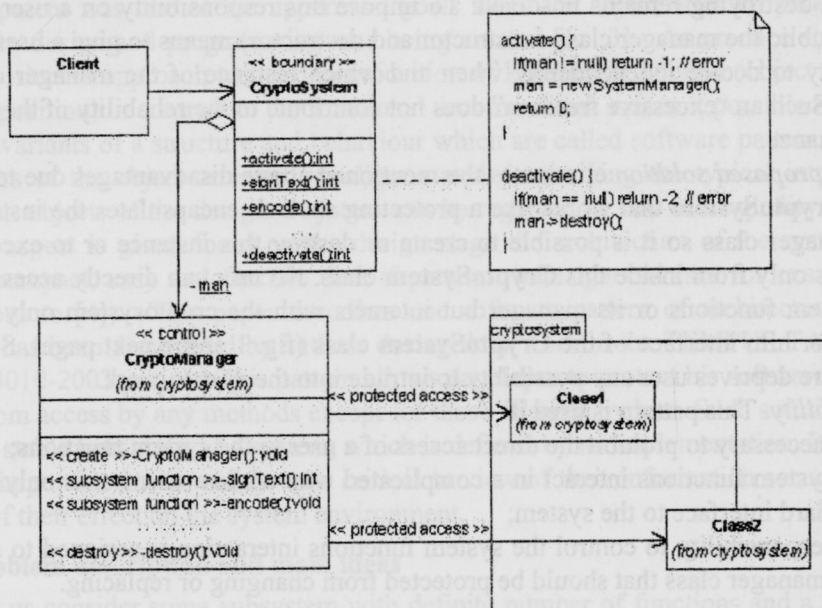


Fig. 2. Pattern Chest structure

Chest – the protecting shell which:

- encapsulates the instance of the class **SystemManager**;

- is responsible for creating the instance of the class **SystemManager** on system start and for destroying it on system stop;
- delegates the responsibility to control the system performance to the instance of the class **SystemManager**;

SystemManager – the manager of the system which:

- is closed for access from outside except for the class **Chest**;
- knows the logic of the system functions interaction and controls it;
- has an immediate access to the functions of the system which are all closed for outside access.

Subsystem – the system which:

- consists of a number of objects of interacting classes;
- has tools to provide the full functionality of the system;
- is closed for access from outside except for the class **SystemManager**.

Relations

- A user starts the system by calling the static method of the class **Chest**.
- The class **Chest** creates the instance of the class **SystemManager**, makes some initializing operations if necessary and delegates all control over the system work to this instance.
- The instance of the class **SystemManager** guarantees the system performance according to the logic and rules known only to this class.
- When the system stops working the class **Chest** destroys the instance of the class **SystemManager**.
- If a user needs to interrupt the system performance he calls the static method of the class **Chest** that correctly releases resources if necessary and destroys the instance of the class **SystemManager**.

Remark. The class **Chest** might have expanded interface implemented by static methods and allowing a user to vary the behaviour of the system within definite limits.

*Advantages and disadvantages of the pattern **Chest**:*

- *all user operations with the system are under the complete control.* Due to the encapsulation of the manager class inside the class **Chest** a user is disabled not only to call the manager class methods but also to create and destroy its instances as well. The interface of the class **Chest** allows user only limited number of operations with the system that cannot destabilize its work. Classes of the system have private interfaces so they are closed for the user access. They can be used only via the system manager. Thus the main requirement of the object-oriented approach, which is encapsulation, is strongly kept;
- *interaction between a user and the system becomes easier.* The interface of the class **Chest** is designed in such a way to supply user by the standard operations with the system. A user has no idea about the internal structure of the system as well as about the logic or implementation of these standard operations on the system level. There is even no need for him to be bothered by creating and destroying the instance of the class **Chest** since its entire interface is static. For a user working with the system is like pressing the buttons on the panel of the microwave oven to cook the meal. He could know nothing about internal structure of the oven and all the more about the properties of physical processes inside it.

- *there could be no creating of redundant objects.* Without the class **Chest** a user would be compelled to interact with the instance of the manager class to make operations with the system. In this case only him would be responsible for creating and destroying such instances, which could lead to creating redundant objects or destroying non-existing ones. With the class **Chest** one instance of the manager class is created at the beginning of the user session with the system and is destroyed at the end of the current session;
- *the manager class and the system classes are simplified.* Since a user has no access to these classes there is no need to handle user errors in their code.
- *the productivity of the system lessens a little.* To reach the required reliability of the system you have “to pay” by introducing a new class with the responsibilities of the protecting shell. This in its turn leads to an additional level of responsibility delegating and code expanding. Nevertheless neither first nor second considerably affects the system performance as the class **Chest** is quite compact and requires no instances to work with. As to the additional method calls they are necessary only for a small number of standard operations with the system from a user side.

Implementation problems

It is advisable to think over the following problems concerned with the implementation of the pattern **Chest**.

1. *How to guarantee the access to the system manager class and to the system functions only via the **Chest** class and to prohibit such access by any other way?*

The implementation of such access depends on the programming language. Usually object-oriented languages have flexible tools for object access rights control. Say C++ allows friend classes or functions to access private and protected data of the class while Java uses packages for this purpose. So in C++ you can declare private or protected all data and methods of the system classes and of the class manager including constructors and destructors. To control the system performance you should declare its manager class or some of its methods as a friend ones to the system classes. As to the shell class **Chest** it should be declared as a friend class to the system manager to be able to delegate to it the responsibilities for the system performance control. In Java to implement the required access you can use the visibility inside the package and default access modifier for data and methods that should be closed from outside. In this case the functionality of the package can be accessed only through the class **Chest** since it is the only class with the public interface.

2. *What is the return value of the class **Chest** methods?*

It is quite common situation when a user would like to know the result of his operation with the system. So it would be better to have the diagnostics of the operation results in the class **Chest**, say by the return values of the interface methods. They could be the values of some enumeration or just string messages. Certainly they should be generated by the system functions and then passed to the system manager, which would return them to the class **Chest** as a result of the delegated responsibility.

Example of code

We consider here some programming code fragments of the pattern **Chest** implementation for the user interaction with the microwave oven. Certainly we are not go-

ing to discuss complex internal mechanisms of this device but restrict ourselves only by details essential for the pattern **Chest**.

C++ code is given below.

```
// enumeration to diagnose the operation result
typedef enum Result {OK, ERROR, FATAL_ERROR};

// enumeration to chose the operating mode
// default mode on start is REHEAT
typedef enum Option {REHEAT, DEFROST, AUTOCOOK};

class MicroWaveChest;

class MicroWaveManager
{
friend class MicroWaveChest;
private:
// methods to control the oven
    Result doStart(int time); // calling the methods
                                //of the MicroWave class to start
                                //the oven for the given time
    Result doStop(); // calling the methods
                                //of the MicroWave class to stop the oven
    Result doSetOption(Option option); //calling
    //the methods of the MicroWave class to set
    //and indicate an operating mode
    displayResult(Result result); //calling the methods
    // of the class MicroWave to display results
    //constructor, destructor
    MicroWaveManager();
    ~MicroWaveManager();
    // data to describe the manager state
    . . . . .
};

class MicroWave
{
friend class MicroWaveManager;
private:
// data to describe the structure and the current
// state of the oven
    . . . . .

// methods to describe working mechanisms
    . . . . .
};
```

```

class MicroWaveChest
{
public:
    static void activate() {
        if (!man) man = new MicroWaveManager();
    }
    static Result start(int time);
    static Result stop();
    static Result setOption(Option option);
    static void deactivate() { if (man) delete man; };
private:
    static MicroWaveManager *man = NULL;
};

Result MicroWaveChest::start(int time)
{
    Result result;
    if (!man) result = FATAL_ERROR;
    else result = man -> doStart(time);
    man -> displayResult(result);
    return result;
}

Result MicroWaveChest::stop()
{
    Result result;
    if (!man) result = ERROR;
    else result = man -> doStop();
    man -> displayResult(result);
    return result;
}

Result MicroWaveChest::SetOption(Option option)
{
    Result result;
    if (!man) result = FATAL_ERROR;
    else result = man -> doSetOption(option);
    man -> displayResult(result);
    return result;
}

```

Here are some possible scenarios of user operations with the oven.

1. Simple reheating for some time, say for 1 minute:
 MicroWaveChest::activate();
 MicroWaveChest::start(60);
 MicroWaveChest::deactivate();
2. Setting the operating mode, say to defrosting, before starting:
 MicroWaveChest::activate();
 MicroWaveChest::setOption(DEFROST);
 MicroWaveChest::start(20);
 MicroWaveChest::deactivate();

3. Stopping the oven before the timeout:

```
MicroWaveChest::stop();
MicroWaveChest::deactivate();
```

Related patterns

To design the system manager which main function is to control the system classes interaction the pattern **Mediator** can be used. To design the interaction between the class **Chest** and the manager class one can use the **Singleton** pattern and elements of the **Memento** pattern. The interaction between the **Chest** class and the system classes via the manager class can be considered in terms of the pattern **Facade** or protecting **Proxy** with slightly changed conditions for the access to real subjects. The manager class can use the pattern **State** to control the system behaviour.

Known applications

This pattern has been used in the process of the cryptographic security system development within the TEMPUS TACIS MP JEP 23010 – 2003 “UnIT – Net in universities management” project.

To complete its description we give here the interface of the class **Chest**. In this application its name is **Cryptosystem**.

```
class Cryptosystem
{
public:
    static Result activate();

    // this method defines the document file for
    // further processing
    static Result setActiveText(const char* location);

    // this method forms signed posting on the base
    // of the document or adds the signature to
    // the posting
    static Result signDocument();

    // this method checks the validness
    // of the signatures in the posting
    //and returns the list of invalid signatures
    static char* validateSignature();

    // this methods restores the document by its
    // posting
    static Result restoreDocument();

    // this method encodes the posting
    static Result encodePosting();

    // this method decodes the posting
    static Result decodePosting();
```

```

// this method saves the active document in a form
// which corresponds its current state
static Result saveActiveText(const char* location);

static Result deactivate();
};

```

The detailed description of the issues concerned with the implementation of the cryptographic security system within the UnIT-Net network can be found here: <http://www.unit-net.org.ua>

REFERENCES

1. Pascoe G. A. Encapsulators: A new software paradigm in Smalltalk-80 / Object-Oriented Programming Systems, Languages and Applications Conference Proceedings. – Portland: ACM Press, 1986. – Pp. 341 – 346.
2. Rumbaugh J., Blaha M. and others. Object-Oriented Modeling and Design. – Englewood Cliffs, NJ: Prentice Hall, 1991. – 347 p.
3. Booch G. Object-Oriented Analysis and Design with Applications. Second Edition. – Redwood City, CA: Benjamin/Cummings, 1994. – 753 p.
4. Gamma E., Helm R. and others. Design Patterns. Elements of Reusable Object-Oriented Software. – Addison-Wesley, 2003. – 321 p.

Компьютерное моделирование дифракции электромагнитных волн на ленточных решетках (случай произвольного падения)

Г. Н. Жолткевич, В. В. Хорошун, В. Б. Хохольков
Харьковский национальный университет им. В.Н. Каразина, Украина

Work is devoted to the qualitative and quantitative analysis of distinctive features of diffraction of flat electromagnetic waves of linear polarization on tape grating at general fall in comparison with cases of excitation of grating E- and H- polarized waves.

Постановка задачи

На решетку из бесконечно тонких и идеально проводящих лент (l – период решетки, d – ширина ее щелей) из верхнего полупространства ($z > 0$) под произвольным углом к ней (рис.1) падает плоская электромагнитная волна

$$\vec{E}_{пад} = \vec{E}_0 e^{ik_0(\vec{n}, \vec{r})}, \quad \vec{H}_{пад} = \vec{H}_0 e^{ik_0(\vec{n}, \vec{r})},$$

где $k_0 = \frac{\omega}{c} = \frac{2\pi}{\lambda}$ – волновое число

свободного пространства, λ – длина волны, ω – круговая частота, c – скорость

света в вакууме. (Временной дифференциальный оператор $\frac{\partial}{\partial t} = -i\omega$).

Требуется определить дифракционные поля во всем пространстве, исключая ленты решетки.

Решение задачи

Пусть $\vec{E}(x, y, z)$ и $\vec{H}(x, y, z)$ – искомое дифракционное поле. Так как уравнения Максвелла инвариантны относительно пространственных сдвигов, а решетка переходит в себя при любых сдвигах в направлении оси OX и при сдвигах, кратных периоду решетки l , в направлении оси OY, то векторы

$$e^{-ik_0(\alpha x + \beta y)} \vec{E}(x, y, z) \text{ и } e^{-ik_0(\alpha x + \beta y)} \vec{H}(x, y, z)$$

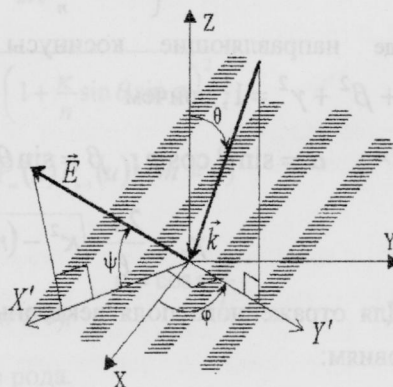


Рис.1 Геометрия структуры
и углы Эйлера

не зависят от x и являются периодическими функциями от y с периодом l . Разлагая эти векторы в ряды Фурье, используя уравнения Гельмгольца для определения вида коэффициентов Фурье и учитывая при этом поведение искомого поля на бесконечности, получим

$$\vec{E}(x, y, z) = \begin{cases} e^{ik_0 \alpha x} \left[\vec{E}_0 e^{-ik_0 \gamma z} + \sum_n \vec{a}_n e^{i\gamma_n z} e^{i\beta_n y} \right], & (z > 0), \\ e^{ik_0 \alpha x} \sum_n \vec{b}_n e^{-i\gamma_n z} e^{i\beta_n y}, & (z < 0). \end{cases} \quad (1a)$$

$$\vec{H}(x, y, z) = \begin{cases} e^{ik_0 \alpha x} \left[\vec{H}_0 e^{-ik_0 \gamma z} + \sum_n \vec{A}_n e^{i\gamma_n z} e^{i\beta_n y} \right], & (z > 0), \\ e^{ik_0 \alpha x} \sum_n \vec{B}_n e^{-i\gamma_n z} e^{i\beta_n y}, & (z < 0), \end{cases} \quad (1b)$$

где направляющие косинусы α, β и γ связаны соотношением $\alpha^2 + \beta^2 + \gamma^2 = 1$, причем

$$\alpha = \sin \theta \cos \varphi, \quad \beta = \sin \theta \sin \varphi, \quad \gamma = \cos \theta, \quad \beta_n = k_0 \beta + \frac{2\pi}{\ell} n,$$

$$\gamma_n = \frac{2\pi}{\ell} \sqrt{\kappa^2 - (n + \kappa \beta)^2 - \kappa^2 \alpha^2}, \quad \kappa = l / \lambda.$$

Для отраженного поля векторные коэффициенты $\vec{a}_n, \vec{A}_n$, удовлетворяют условиям:

$$\begin{aligned} \beta_n A_{nz} - \gamma_n A_{ny} &= -a_{nx}, & \beta_n a_{nz} - \gamma_n a_{ny} &= A_{nx}; \\ \tilde{\alpha} A_{nz} - \gamma_n A_{nx} &= a_{ny}, & \tilde{\alpha} a_{nz} - \gamma_n a_{nx} &= -A_{ny}; \\ \tilde{\alpha} A_{ny} - \beta_n A_{nx} &= -a_{nz}, & \tilde{\alpha} a_{ny} - \beta_n a_{nx} &= A_{nz}; \\ \tilde{\alpha} a_{nx} + \beta_n a_{ny} + \gamma_n a_{nz} &= 0, & \tilde{\alpha} A_{nx} + \beta_n A_{ny} + \gamma_n A_{nz} &= 0. \end{aligned} \quad (2)$$

где $\tilde{\alpha} = k_0 \alpha$.

Для коэффициентов прохождения $\vec{b}_n$ и $\vec{B}_n$ получаем аналогичные соотношения путем замены $\vec{a}_n \leftrightarrow \vec{b}_n$, $\vec{A}_n \leftrightarrow \vec{B}_n$, и изменением знака γ_n на противоположный.

Подчиняя искомые поля точным граничным условиям, требующим обращения в нуль тангенциальных компонент электрического поля и непрерывности всего поля на щелях решетки, получаем связь между неизвестными коэффициентами

$$a_{nx} + \delta_{n0} E_{0x} = b_{nx}; \quad A_{nx} - \delta_{0n} H_{0x} = -B_{nx};$$

$$\begin{aligned} a_{ny} + \delta_{n0} E_{0y} &= b_{ny}; & A_{ny} - \delta_{0n} H_{0y} &= -B_{ny}; \\ a_{nz} - \delta_{n0} E_{0z} &= -b_{nz}; & A_{nz} + \delta_{0n} H_{0z} &= B_{nz}. \end{aligned} \quad (3)$$

и две независимые канонические системы сумматорных уравнений, которые методом АМШ [5] в его модифицированной форме [6] сведены к двум бесконечным системам линейных алгебраических уравнений второго рода (СЛАУ-2) следующего вида

$$\begin{cases} -b_{0x} = i\kappa\gamma(b_{0x} - E_{0x})W_0^0(u) + \sum_{n \neq 0} b_{nx} |n| \tilde{\chi}_n W_0^n(u), & (m=0), \\ mb_{mx} = i\kappa\gamma(b_{0x} - E_{0x})W_m^0(u) + \sum_{n \neq 0} b_{nx} |n| \tilde{\chi}_n W_m^n(u), & (m \neq 0), \end{cases} \quad (4a)$$

$$\begin{cases} -A_{0x} = i\kappa\gamma(A_{0x} - H_{0x})W_0^0(-u) + \sum_{n \neq 0} A_{nx} |n| \tilde{\chi}_n W_0^n(-u), & (m=0), \\ mA_{mx} = i\kappa\gamma(A_{0x} - H_{0x})W_m^0(-u) + \sum_{n \neq 0} A_{nx} |n| \tilde{\chi}_n W_m^n(-u), & (m \neq 0), \end{cases} \quad (4b)$$

где

$$\tilde{\chi}_n = 1 + i \sqrt{\frac{\kappa^2}{n^2} (1 - \sin^2 \theta \cos^2 \varphi) - \left(1 + \frac{\kappa}{n} \sin \theta \sin \varphi\right)^2}, \quad (5)$$

$$W_m^n(u) = \frac{m}{2(m-n)} [P_n(u)P_{m-1}(u) - P_m(u)P_{n-1}(u)], \quad (m \neq n) \quad (6)$$

$$W_0^n(u) = \begin{cases} \ln \frac{1+u}{2}, & (n=0), \\ \frac{1}{2n} [P_n(u) - P_{n-1}(u)], & (n \neq 0), \end{cases} \quad u = \cos \pi \frac{d}{l}, \quad (7)$$

где $P_n(u)$ - полиномы Лежандра первого рода.

Из решения редуцированных СЛАУ-2 вида (4a)-(4b) определяются значения коэффициентов b_{nx} и A_{nx} , через которые с помощью формул (2) и (3) выражаются остальные неизвестные амплитуды дифракционных полей. Заметим, что системы (4a) и (4b) являются независимыми только в случае, когда решетка находится в безграничной изотропной среде. При наличии одной или нескольких границ раздела магнитоэлектрических сред подсистемы (4a) и (4b) являются связанными и, следовательно, их решение следует проводить совместно.

Случай Е-поляризации.

В этом частном случае исходная математическая модель имеет вид:

$$\sum_{n=-\infty}^{\infty} b_n e^{in\zeta} = 0, \quad (\Phi < \zeta \leq 2\pi - \Phi), \quad (8a)$$

$$\sum_{n=-\infty}^{\infty} b_n \eta_n e^{in\zeta} = \kappa, \quad (|\zeta| < \Phi), \quad (8b)$$

где $\eta_n = \sqrt{\kappa^2 - (n + \kappa \sin \theta)^2}$, $\kappa = l / \lambda$, λ - длина падающей волны.

Учитывая связь между коэффициентами отражения a_n и прохождения b_n :

$$a_n + \delta_0^n = b_n, \quad (9)$$

где δ_0^n - символ Кронекера, получим математическую модель исходной задачи в ином виде, а именно:

$$\sum_{n=-\infty}^{\infty} a_n e^{in\zeta} = -1, (\Phi < \zeta \leq 2\pi - \Phi), \quad (10a)$$

$$\sum_{n=-\infty}^{\infty} a_n \eta_n e^{in\zeta} = 0, (|\zeta| < \Phi), \quad (10b)$$

Решение системы сумматорных уравнений (8a-8b) может быть получено путем сведения ее к краевой задаче Римана с коэффициентом задачи $G(\varphi) = -1$ и свободным членом $g(\varphi)$, удовлетворяющим условию Гельдера, причем это можно проделать двумя способами, которые отличаются друг от друга различным видом так называемого параметра малости:

1-й способ

$$\chi_n = 1 + i \sqrt{\frac{\kappa^2}{(n + \kappa \sin \theta)^2} - 1}, \quad \kappa = \frac{l}{\lambda} \quad (11)$$

2-й способ

$$\tilde{\chi}_n = 1 + i \sqrt{\frac{\kappa^2}{n^2} - \left(1 + \frac{\kappa}{n} \sin \theta\right)^2}, \quad n \neq 0. \quad (12)$$

Очевидно, что в случае нормального падения плоской волны на решетку (этот случай рассмотрен в работе [5]) оба способа совпадают. В случае наклонного падения применение первого из указанных способов приводит к СЛАУ-2, некоторые матричные элементы которой выражаются через полиномы Лежандра, а остальные - через функции Лежандра первого рода, что значительно усложняет их вычисление [7,8]. Этот недостаток устраняется при введении параметра малости $\tilde{\chi}_n$ вторым способом.

В работе [9] изложен метод решения задачи дифракции плоской Е-поляризованной электромагнитной волны на однослойной многоэлементной решетке (случай нормального падения), основанный на регуляризации эквивалентного исходной математической модели (10a-10b) сингулярного интегрального уравнения с логарифмическим ядром:

$$\frac{1}{\pi} \int_L j(\tau) \sum_{n=-\infty}^{\infty} \frac{e^{in(y-\tau)}}{\gamma_n} d\tau = -1, \quad y \in L \quad (13)$$

В самом деле, вводя в рассмотрение функцию плотности тока на лентах решетки:

$$j(\tau) = \sum_{n=-\infty}^{\infty} a_n \gamma_n e^{in\tau}, \quad (14)$$

из условия (10b) получим, что $j(\tau) = 0$ на щелях решетки, т.е. при $\tau \in [0, 2\pi] \setminus L$. Подставляя в условие (10a) выражение для коэффициентов Фурье $a_n \gamma_n$ из (14), получим СИУ в виде (13), ядро которого является 2π -периодической функцией y и τ и при $y = \tau$ имеет логарифмическую особенность, так как $\lim_{n \rightarrow \infty} \gamma_n = i|n|$ и

$$\sum_{n \neq 0} \frac{e^{in(y-\tau)}}{|n|} = 2 \sum_{n=1}^{\infty} \frac{\cos n(y-\tau)}{n} = -2 \ln \left[2 \sin \frac{|y-\tau|}{2} \right]$$

В указанной работе СИУ в виде (13) решено путем сведения к краевой задаче Римана с коэффициентом задачи $G = -1$ и индексом задачи, равным -1 . В результате решение неоднородной задачи Римана имеет вид:

$$x(z) = \frac{X(z)}{2\pi i} \int_L \frac{g(\tau)}{X^+(\tau)} \frac{d\tau}{\tau - z}, \quad (15)$$

где $X(z) = \sqrt{(z-\alpha)(z-\bar{\alpha})}$ - выделенная в плоскости с разрезом по дуге L однозначная ветвь многозначной функции (каноническое решение задачи Римана), $\alpha = e^{i\Phi}$, $\bar{\alpha} = e^{-i\Phi}$.

Решение в виде (15) существует тогда и только тогда, если выполняется условие разрешимости $\int_L \frac{g(\tau)}{X^+(\tau)} d\tau = 0$,

В итоге получена СЛАУ-2, матричные элементы которой вычислены в работе [9].

Заметим, что решение системы сумматорных уравнений (10a-10b) может быть непосредственно сведено к задаче Римана с тем же коэффициентом задачи $G = -1$, путем введения функций

$$x^+(z) = \sum_{n \geq 0} x_n z^n, \quad z < 1 \quad (16)$$

$$x^-(z) = -\sum_{n < 0} x_n z^n, \quad z > 1$$

голоморфных соответственно внутри и вне окружности $|z| = 1$, причем $x_n = a_n \gamma_n$. Продифференцировав (10a) по ζ и введя аналогично [9] параметр малости

$$\varepsilon_n = \frac{i}{\gamma_n} - \frac{1}{|n|}, \quad n \neq 0 \quad (17)$$

придем к задаче Римана:

$$x^+(e^{i\zeta}) + x^-(e^{i\zeta}) = g(e^{i\zeta}), \quad (18)$$

где $g(e^{i\zeta}) = a_0 \kappa + i \sum_{n \neq 0} n x_n \varepsilon_n e^{in\zeta}$.

Отметим, что решение задачи Римана ищется в классе функций, ограниченных на концах дуг окружности единичного радиуса. Решение получено в виде СЛАУ-2 с матричными элементами, которые выражаются через полиномы Лежандра первого рода.

Результаты расчетов

В качестве примера на рис.2 и рис.3 представлена зависимость модуля коэффициента прохождения основной волны $|b_0|$ от l/λ и обратного параметра λ/l .

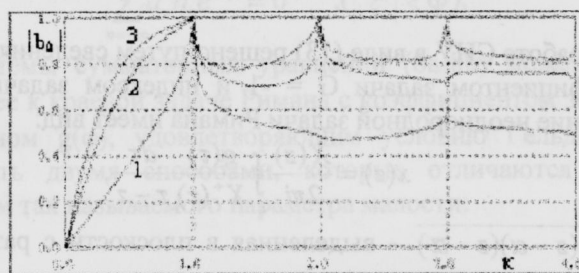


Рис. 2. Зависимость модуля коэффициента прохождения основной волны $|b_0|$ от l/λ

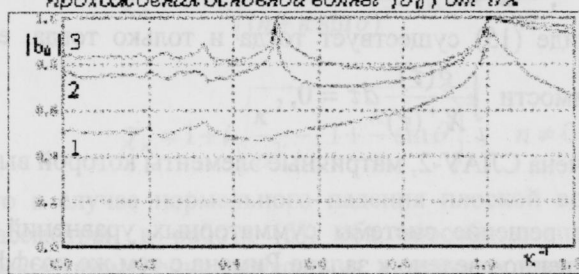


Рис. 3. Зависимость модуля коэффициента прохождения основной волны $|b_0|$ от λ/l

Кривым под номерами 1 – 3 соответствуют значения коэффициента заполнения решетки $d/l = 1/2, 3/4, 5/6$ соответственно. На рис.4 представлена зависимость модуля коэффициента прохождения основной волны от коэффициента заполнения решетки d/l , кривым под номерами 1-3 соответствуют значения $l/\lambda = 1, 2, 3$ соответственно. На рис.5 представлена зависимость модуля коэффициента прохождения от d/l для высших гармоник $n = \pm 1, \pm 2, \pm 3$ при $l/\lambda = 1, 2, 3$.

Результаты вычислений коэффициентов отражения и прохождения во всех рассматриваемых случаях полностью совпадают [10].

Авторы выражают благодарность проф. Ю.В. Ганделю за постоянный интерес к работе и плодотворные дискуссии.

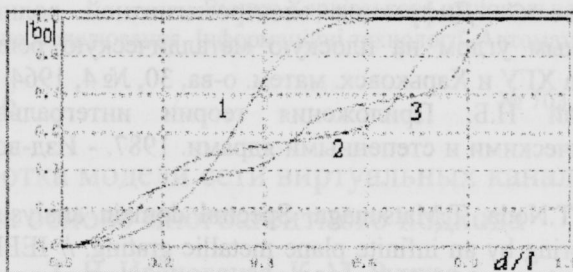


Рис. 4. Зависимость $|b_0|$ от d/l ($l/\lambda = 1, 2, 3$)

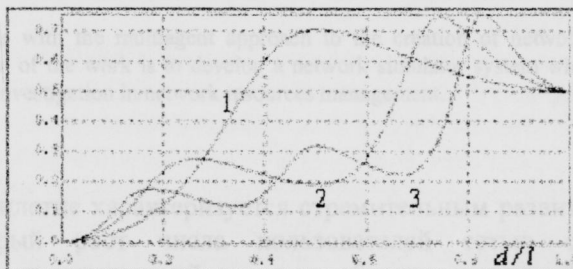


Рис. 5. Зависимость $|b_n|$ от d/l ($l/\lambda = 1, 2, 3$;
 $n = \pm 1, \pm 2, \pm 3$)

ЛИТЕРАТУРА

1. Gandel Yu. The numerically analytic methods for solution diffraction problems on multi-element gratings //2005 5th international conference on antenna theory and techniques, 2005 May, Kiev, pp.42-45.
2. Gandel Yu. The method of dual and singular integral equations in the problems of diffraction by finite gratings //Electromagnetic Phenomena, Kharkov, 1998, Vol.1, No. 2, pp. 220-232.
3. Khoroshun V. The Riemann-Hilbert problem method in the theory of electromagnetic wave diffraction by periodic structures //2005 5th international conference on antenna theory and techniques, 2005 May, Kiev, pp. 424-426.
4. Жолткевич Г.М., Хохольков В.Б. Математичне моделювання явищ дифракції хвиль на стрічкових металевих ґратках. // 36. доп. "Перші Каразінські природознавчі студії".-2004 .Харків.-С.94-96.
5. З.С.Агранович, В.А.Марченко, В.П.Шестопапов. Дифракция электромагнитных волн на плоских металлических решетках // ЖТФ.- 1962.- Т:32, №4.- С. 381-394.
6. В.В.Хорошун. О модификации метода задачи Римана-Гильберта для случая наклонного падения плоской электромагнитной волны на решетку из лент // Вестник ХГУ. Радиофизика и электроника.- 1998.- Вып. 405.- С. 137-141.
7. А.И. Адонина, В.П. Шестопапов. Дифракция электромагнитных волн на плоской металлической решетке с экраном (случай произвольного падения) // РиЭ, 8, № 8, 1963.

8. Е.Н.Подольский. Дифракция электромагнитной волны, падающей под произвольным углом на плоскую металлическую решетку. // Зап. мех.-матем. ф-та ХГУ и Харьковск. матем. о-ва, 30, № 4, 1964.
9. Плещинский Н.Б. Приложения теории интегральных уравнений с логарифмическими и степенными ядрами. 1987. - Изд-во Казанского ун-та, 158 с.
10. K.Uchida, T.Noda, T.Matsunaga. Spectral domain analysis of electromagnetic wave scattering by an infinite plane metallic grating. // IEEE Trans. on antennas and propagation, vol.35, № 1, 1987.