DISTRIBUTED DATA MANAGEMENT SYSTEM FOR THE NEEDS OF DISTRIBUTED INTERNET- BASED PERFORMANCE-CENTERED ENVIROMENT FOR LEARNING AND DEVELOPMENT

Dimitar Tokmakov, Nevena Mileva

ABSTRACT
This paper presents the design of Distributed Data Management System (DDMS) for the needs of Distributed Internet-based Performance-centred Environment for Learning and Development (DIPELD) which applies the model of Internet based performance support systems with educational elements (IPSS_EE). DDMS facilitates the cooperation among educational resources and systems which are located in different geographical locations in different local servers. The main purpose of the work presented is to define the structure and model of DDMS as well the technologies needed inside for the practical implementation of DDMS. The system is based on Communication Server implementing distributed database storage system that provides data and file replication in an environment of high data mobility and heterogeneous host capabilities. In this paper we also propose a new method for data base merge replication between server nodes over the Internet by implementing a software module called DDMS.

KEYWORDS

Performance Support Systems, e-Learning, Individualized Learning, Distributed Data Management System, Data replication

INTRODUCTION

The technological landscape of modern E-Learning is dominated by so-called learning management systems (LMS) such as Blackboard, WebCT or Moodle. Learning management systems are powerful integrated systems that support a number of activities performed by teachers and students during the e-Learning process.

Internet-based Performance Support Systems with Educational Elements (IPSS_EE) is an innovative approach for task-performance independent e-Learning and development of new instruments in instructional design of Internet-based courses for engineering education. The IPSS_EE research was under two-year project (http://www.ipss-ee.net) within the European program Socrates/ Minerva.

Internet-based Performance Support System with Educational Elements (IPSS_EE) is an integrated electronic environment, which is available via Internet and is structured to provide individualized online access to the full range of information, guidance, advice, data, images, tools and software to permit the user to perform a task (Mileva and Tzanova 2000). The performance-centered approach of thinking up offers new opportunities for the educational and training organizations and calls principle changes in the instructional design of course materials. Developed IPSS_EE Environment gives up the course

designers and teachers a possibility to create IPSS_EE courses without knowing the theory and organisation behind performance-centered approach (Mileva and Tzanova 2002).

Internet-based performance support system (IPSS) is one of the most advanced concepts in the domain of the educational e-Learning initiatives. It is aimed at providing just-in-time, just enough and the point of need support to learners in order to deal with complex authentic tasks in the context of problem-based learning. However the existing performance support systems solutions are short in exploring fully the advantages of Internet in managing distributed pedagogical resources in the most effective and efficient way. There are at least two types of determinants explaining this situation in designing, developing and implementing Internet-based performance support systems. One is pedagogical and the second is technological. The *pedagogical* reason reflects the lack of operational definition of distributed learning in relation to performance support concept, which leads to incomplete functional specifications and inadequate development solutions.

DISTRIBUTED PERFORMANCE CENTERED ENVIROMENT

In the Distributed Performance-Centered Environment (DPCE) multiple users can interact with distributed content in real time, and furthermore it is distributed, running on several servers which are connected by a network using a series of client server applications.DPCE have many characteristics that can be exploited in the educational process and especially in distance learning(Richards at al 2002).

Distributed learning requires students to get structured support in term of background information, examples, demos, simulations, procedures, and software from distributed sources with no time and place constraints. Distributed IPSS introduces the concept of distribute instruction as well. It applies embedded content management facilities with tools, templates, and guidelines for designing courses from learning objects in a shared repository.

Our own partnership experience in designing, developing, evaluating and implementing IPSS in higher engineering and vocational institutions across Europe indicated the following technological problems caused by centralized data-base: not accessible data-base in case of bad or missing Internet connection with the server; directed information flow due to a lack of alternatives; problems to insert multi-language content to the data-base from PC without the specific coding table; huge amount of information and need for server specification in language and subject matter; security problems.

An overview of the current practice of IPSSs reveals that most of the applications do not address sufficiently the growing need for individualized learning. An IPSS for distributing learning should provide conditions for matching the content to learners' individual characteristics such as level knowledge, learning styles, cognitive efforts and cognitive modalities to list but a few. Individualization should be based either on pre-assessment of these constructs and then assigning learners to a particular adaptive track, or monitoring learning progress for additional adjustment.To address these issues we developed the Distributed Performance-Centered Environment.

The Distributed Performance-Centered Environment consists of one or more local IPSS_EE servers, connected with a unified communication server, via the Internet communication environment for allocation of the educational resources and creating of a unified database repository and contents of the certain courses, maintained in the communication server. These are the main characterizations of the DPCE:

- The local IPSS_EE servers are independent of one another. They contain educational courses according to various scientific subjects.
- The local servers are geographically allocated worldwide.
- A repository of all educational courses is maintained, by replication of the databases of local servers, as well as the contents of the courses by replication of content files.
- The distributing environment for the databases replication and content files is the Internet.

- The operational systems of the separate local servers can be heterogenic (Linux, Solaris, Windows and others).
- The software of the DPCE consists of 2 main components – consumer and communication, as the second one is a subject of the present article.
- The replication system, synchronization and transfer of files are completely automatic.
- The software of the system for the distributed databases management is independent, regarding its platform.

The structure of the above-described DPCE can be seen in fig1.

DPCE enables instructors, developers, and learners to become consumers of, and contributors to a network of learning object repositories. DPCE enables individuals to collect and manage learning objects, perhaps creating portfolios of their personal learning experiences to reduce the transience of the e-Learning experience.

DISTRIBUTED DATA MANAGEMENT SYSTEM FOR THE NEEDS OF DPCE .

The architecture and the model of DDMS can be seen in fig.2.The client-server application represents a cover of the databases system and is purposed to transport and replicate data and files with contents from the local IPSS_EE servers toward the communication server and backward. The data transfer is accomplished by specially created **communication protocol**, and TCP/IP executes their transfer through the Internet. In the specific development, the hosting server operational system is Linux, kernel 2.6 both for the local IPSS_EE and communication servers.

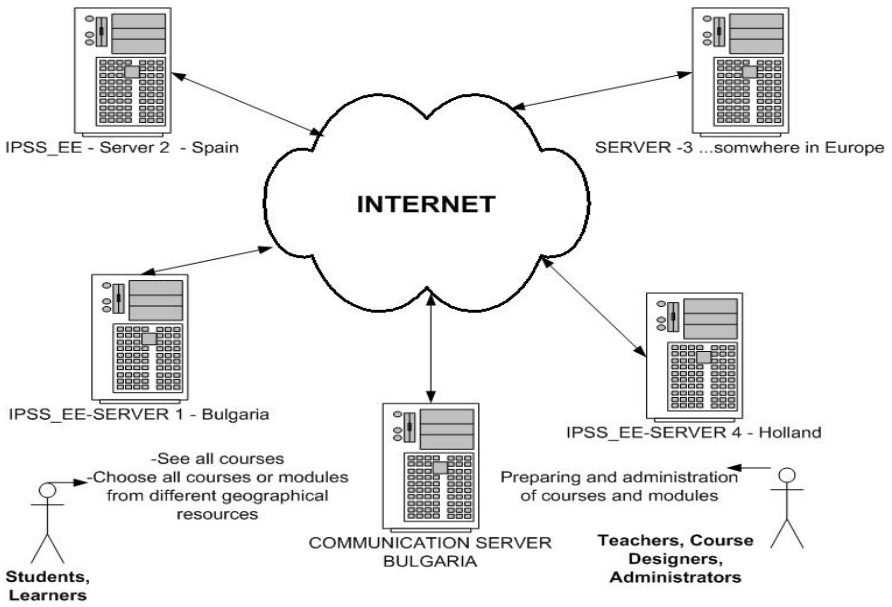

Figure 1. DPCE structure

The system`s configuration allows to be used another operational system, for instance Windows, because software solutions with open code are used, possessing ports to Windows, Linux, Solaris and others. The software components of the distributed IPSS_EE servers are identical. The only difference between them is the cover of the databases system represents a type **client** application, which is synchronized via the communication sever, by messages exchange.

The designed system for distributed databases maintenance (fig. 2) is a client-server application and aims to exchange information between certain servers. It could satisfy simpler structures such as

"request-response", and also much more complicated solutions in the area of distribution programming. It includes also the separation process of one task into parts, and unification of the resources of multiple specialized systems, in order to process those parts and send back the results. The dynamic set up of such tasks is possible, which allows their adding, starting and quit, with no disturbing the client's or the server's work. Such task, in the concrete case, is the replication of databases from the distributed IPSS_EE servers to the communication server and backward, as well as transfer of the files with contents between the communication server and the IPSS_EE servers. This task is accomplished automatically and only when there is some change in the tables of databases and addition of new contents to some of the servers. The communication between the separate stations itself is accomplished on messages basis. Once created, they are ready for registration in a special "messages box". Each client, connected to the system, has to care for the download of the messages and also to process the information they carry.

The code is easily transferred and independent from the operational system, as all settings and the complete information, necessary for the normal and all value work of the client or the server, are contained in configuration files. The application has been worked out under Java. The communication between the separate stations itself is accomplished on messages basis. Once created, they are ready for registration in a special "messages box". Each client, connected to the system, has to care to download messages directed to it, and also to process the information they carry. The code is easily transferred and independent from the operational system, as all settings and the complete information, necessary for the normal and all value work of the client or the server, are contained in configuration files. The application has been worked out under Java.

The first task of the server – when it is restarted – is to recognize the secure socket layer (**SSL)** keys and the passwords for accessing them. When it is fulfilled, the system's work parameters have to be located, as they are read by configuration file. This includes TCP port, name, IP-address, which assistance systems to be activated and with what parameters, and others.

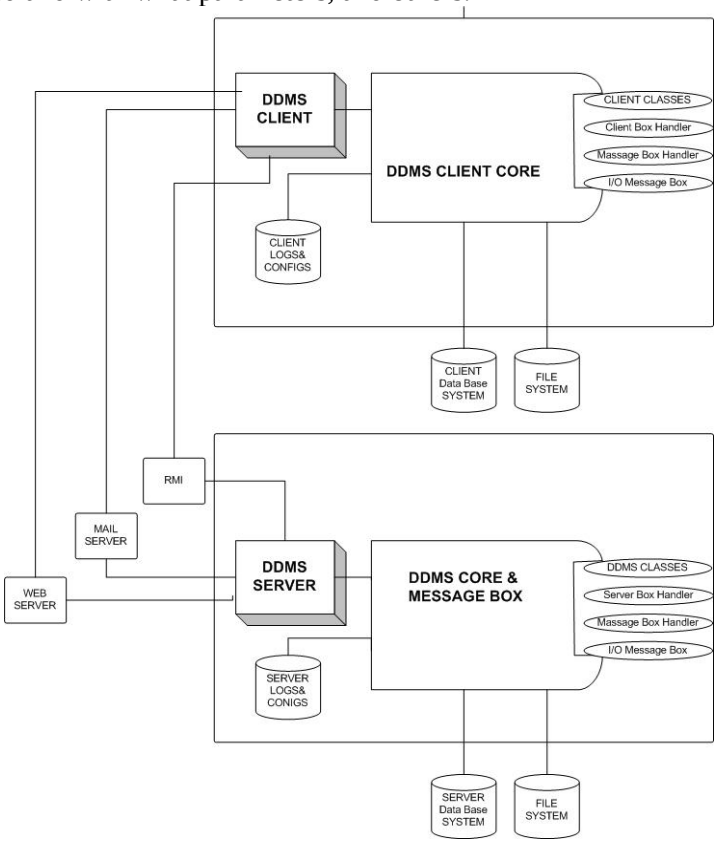

Figure 2

.

The work regime is also defined, and the log-mechanism, in order for the first generated messages to be recorded. At this stage, the appropriate server activity is already possible and the next step is creating the Remote Method Invocation (RMI) register. At the end, the **ServerBoxHandler** is started, and it establishes connection with the **SecureRmiSocketFactory,** modified by us, and initializing the **RemoteMessageInteface,** and the server passes to awaiting regime, as it checks for new messages.

It is possible now for the client to establish connection. Initially, the process runs like the previous one. The SSL keys and their passwords are again allocated, the configuration files are read and the initial client`s parameters are set up. It is again necessary to be loaded and set the log system. After all these actions are accomplished, the **ClientBoxHandler** is started with no obstacles. Here, the process of connecting to the RMI register is a bit more complicated. A special significance is that in the background, there is another connection established. It is on port 80 at TCP protocol where the stub and skeleton files are downloaded. They are necessary for the RMI communication.

Once ensured with those files, the client just has to start the **Naming lookup** method of RMI and to register the own copy of the Remote interface. When this is accomplished, then the server connection is established and a process of concordance the work of both parties is started. For the purpose, a set of office messages is used, and they are collected in the **system.protocol** pack**.** The first step the client should take is to request a registration. For its successful completion, the server must permit it and to relevantly make the registration. The criteria are access rights of the client, as well as whether he already has established active connection or not. The next step is negotiation for a unified communication protocol. The client sends its version, and if does not coincide with the server's one, a command for switching off is received. If everything here is appropriate, **ClientBoxHandlera** is in charge, initializing the task registration system and passes into a state of messages waiting.

When the task for the database review is activated, it prepares the information necessary for update or complete **dump**, if it is started for the first time. This information is archived, with the purpose to diminish the size of the file transferred and via own messages, it starts to communicate with the server. The server extracts from the archive the files received and shows changes made in them. In this way, the merge replication of the mySQL database is accomplished, which is used for the IPSS_EE servers.

Both the client and the server have almost identical structure. Commonly, it is signed in a core, which is the 'message box', within a few classes, operating with it, as well as the classes for configuration of the system for the application documentation, for the file system access, and a part of those for communication establishment and classes for archiving and extracting archives of data.

Besides them, the server disposes with a few systems, caring of its normal work and makes corrections on its 'behavior'.Here is also the RMI register, to which – via web-server – all clients are connected.

In the base is the core, implementing functions on the 'messages box'. Such functions are downloading and registering of messages. Literally, the 'box' exists only on the server`s application and represents an area of objects.
The main classes, working on the core, are:
ServerBoxHandler and **ClientBoxHandler** (relatively to the server and the client). These are classes, enhancing the Java class **Thread**. Their major task is to original initialization of the system`s parameters and passing to regime of watching the 'box'. It is guaranteed by the method **run** of the class, which works as a self-independent one, and is synchronized with the aim object access to not be implemented at one and the same time more than one process.

MessageHandler and **NonSystemMessageHandler.** These classes analyze and process information, entering the system. The first one process not system messages, and the second one – system. Their work principle is identical. They both have one and the same method, called **handleMessage().** It compares the ID of incoming messages and executes the relevant part of the code. If there is no such ID number, then an error alert is shown.

The system for tasks registration is **Job Control System (JCS) -** fig 3**.** This system is the core of all other systems. Its purpose is the registration and management of various tasks, which has been started on the client or server. For creating a task to be fulfilled by the relevant station is necessary for the **Job** class to be widened. The system management is accomplished by the **JobControlSystem** class, by which its starting and quitting is possible. Here are the functions for starting and quitting of a given task as well.

System for documentation of the application`s behavior.

As it name gives the hint, it records in files the complete necessary information, generated from the appropriate work of the client, and also from errors occurred during the application process.

System for sending the logs via email.

It is a system for automatic sending of the **log** files of a preliminary configured **email**. The purpose is for a more powerful control to be ensured, at a distance of the application. It is also possible a compulsory activation of the mechanism, aiming the immediate information receiving in the email box set. **LazyMessageSystem :** In order to avoid circumstances when a given low-priority message to stay for longer on the 'box''s bottom, it is provided – through interval, set by the user – for the mentioned system to be activated.

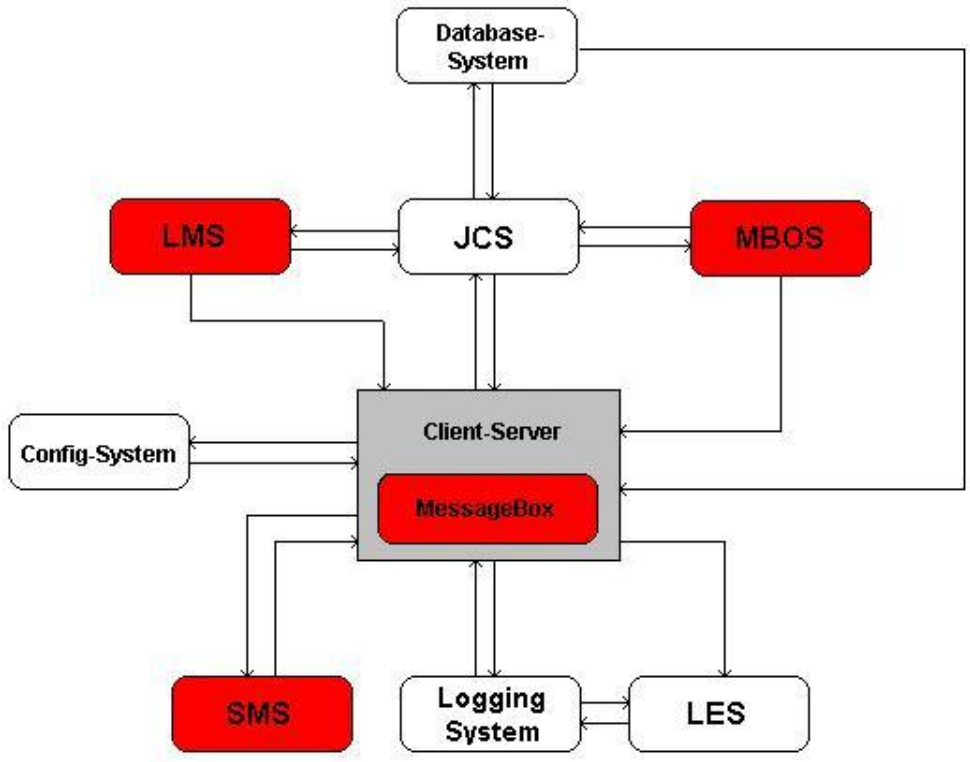

Figure 3. LMS – Lazy Message System; LES - Log to E-mail System; SMS- Store Message System; JCS- Job Control System; MBOS – Message Box Overrun System

System for overloading the box.

It is represented in the **MessageBoxOverrunSystem** class. Its purpose is to follow the number of messages in the box to not exceed a number, preliminarily defined by the user. If it happens, the system is activated and changes the rights of all clients so, until the box is not unloaded, they cannot register messages in it.

Store message system
The purpose of this part of the application is to transfer unsent messages to the file system. If a signal for abortion is sent to the server, but there is still information not downloaded in the box, this system is activated and transfers it to a file.

System for database messages processing.
It accomplishes the communication between the client`s and server`s part of the **DatabaseJob** as the task is registered in the System for Task Control.

From part of the server, the process is manually started, i.e. the client itself sends a message that there is new information and it is necessary for the database to be updated. This message can be used for original initialization or for changes made. In fact, the server receives and a file, containing those changes. The file is archived, it carries the client`s name and is saved in a special directory of the file system. The process has to extract its archive and inform the sender that it has successfully received the information, in order both parties to transfer the file worked out into the archive folder for updates already made.

From part of the client, **JobControllSystem** (JCS) activates the task as it happens by preliminarily defined interval. When the moment comes and the configuration file activates the process, all databases, listed for watching in **JCS** are read and archiving and transfer of files to the server is started. These files are **binary** logs, containing all changes on the databases, made after the latest activation of the system. After that, a message is generated, and it informs that the transfer is complete and it is the turn of the other party to actualize the changes. The client also saves the files sent in an archive folder. The user sets the interval on its own, and then archive files are liable to termination of the file system.

NETWORK ASSURANCE AND COMMUNICATION PROTOCOL

The main task of the server is to work independently, expecting and satisfying incoming requests. In order to reach the server, the client must know the machine`s name the server works on, as well as the port or the socket, on which it follows, and also the correct way of sending a request. For each client, established connection is kept a report for all current connections it has ever made, their status is checked, and the requests progress and all resources of the server are compensated, when the connection is over. For achieving those goals and for complete establishment of the communication process between both parties, the usage of objects serialization and their transfer via the net was necessary to be implemented. The serialization here has been fulfilled through the Remote Method I. It is a completely developed architecture for distributed programming, and is applicable for tasks much more complicated than the 'request-reply' type of programming.

Relating the usage of RMI services, the remote client first has to receive reference toward an object, registered in it. After receiving that information, RMI cares of the following:

- Object serialization
- Object transfer, using its own 'wire' protocol
- Management of errors, specific for the network
- Security management

COMMUNICATION PROTOCOL

The so-called communication protocol is ensured with multiple system messages. They are analyzed and processed by the relevant recipient. They are foreseen to be high-priority in comparison with that of the non-system, which guarantees their faster fulfillment and therefore, the faster actualization of the information they bring.
In order to be used the RMI on SSL was necessary the re-encrypt of many of the Net classes.

The server controls the rights of each client. A list of permitted IP addresses is made and all attempts for establishing connection of clients unlisted are abandoned**.** The clients themselves have at disposal access levels, purposed to limit the change of the relevant parameters of the server, by users who are not authorized to do so.

SYSTEM MONITOR

For following the work of the client-server is ensured separate software. It is developed in two variants – a text variant, in which the information is presented in the console itself, and graphic variant, in which – besides statistics represented – can be also used some graphics of the processes accomplished. The monitor provides information about server box load, active jobs, databases replicated, files transferred etc. A screenshot of the monitor software is shown in fig.4.

CONCLUSION

This paper proposes an architecture and technological implementation for DDMS for the needs of Distributed e-Learning Performance – Centered Environment based on distributed reusable intelligent learning activities that integrate the benefits provided by modern LMS and educational material repositories with the power of distributed architectures. The DPCE addresses the new trends in higher education by providing a web-based environment for the sharing of learner-centered resources.

Figure 4. DDMS Monitor

We have implemented the core functionality of the system by using some rather simple approaches to implement the required actions of building a learning object repository in the communication server, as well as creating a client-server architecture for distributing them over the Internet.

REFERENCES

Brusilovsky, P. (2004) KnowledgeTree: A distributed architecture for adaptive e-learning. In: Proceedings of The Thirteenth International World Wide Web Conference, WWW 2004 ,New York, NY, 17-22 May, 2004, ACM Press, pp. 104-113

Elmasri R., Navathe B, (2000) Fundamentals of Database Systems, Addison-Wesley, USA

Garcia V., Gutl.C. , Modritscher F. (2004) EHELP - Enhanced E-learning Repository: The Use of a Dynamic Background Library for a Better Knowledge Transfer Process. Proceedings of the International conference ICL2004, ISBN 3-89958-089-3 September 29-October 01, Villach Austria

Hamilton, C. (2001, October). Software combinations for learning object repositories. Paper presented at CANARIE E-Learning Workshop, Toronto

Mileva N., Tzanova S. (2002) "Using performance-centred approach in Internet-based vocational education". II European Conference on Information Technology in Education and Citizenship: A Critical Insight, Barcelona, June 26-28.

Mileva N., Tzanova S. (2000), Performance Support Systems with Educational elements in Students' Learning Process. ED-ICT 2000 International Conference on Information and Communication Technologies for Education. Vienna, December 6. pp. 55-59

Mileva, N., Tokmakov D., Stoyanova, S. Internet-based Performance Centred Environment for Individualized Learning in the Framework of Training/Learning Partnership between Industry and Higher Education, Proceedings of the International conference ICL2004, ISBN 3-89958-089-3 September 29-October 01, Villach Austria.

Richards G., McGreal R., Hatala M., and Friesen N.(2002) The Evolution of Learning Object Repository Technologies: Portals for On-line Objects for Learning, Canadian Journal of Distance Education vol.17, No.3,67-79

Salvachua, J., Quemada J., Fernandez, B., Huecas, G. (2003) EducaNext: A Service for Knowledge Sharing, UPGRADE, Special issue on e-Learning for Borderless Education, Vol. IV, issue no. 5, October 2003

Simon, S. Retalis, S. Brantner: (2003) Building Interoperability among Learning Content Management Systems, in: Proceedings of the 12th World Wide Web Conference, May, 2003.

W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmr, and T. Risch.(2002) Edutella: A p2p networking infrastructure based on rdf. In Proceedings of the Eleventh International World Wide Web Conference, Honolulu, Hawaii, May, 2002.

Dimitar Tokmakov
Assistant Professor
Plovdiv University "Paisi Hilendarski"
ECIT Department
24 Tzar Assen Str.
Plovdiv 4000
BULGARIA
Email: tokmakov@pu.acad.bg

Nevena Mileva
Associate Professor
Plovdiv University "Paisi Hilendarski"
ECIT Department
24 Tzar Assen Str.
Plovdiv 4000
BULGARIA
Email: nmileva@pu.acad.bg