Remote Laboratory Java Server Based on JACOB Project

doi:10.3991/ijoe.v7i1.1528

P. Bisták and P. Folvarčík
Slovak University of Technology, Bratislava, Slovakia

***Abstract*—Remote laboratories play an important role in the educational process of engineers. This paper deals with the structure of remote laboratories. The principle of the proposed remote laboratory structure is based on the Java server application that communicates with Matlab through the COM technology for the data exchange under the Windows operating system. Java does not support COM directly so the results of the JACOB project are used and modified to cope with this problem. In laboratories for control engineering education a control algorithm usually runs on a PC with Matlab that really controls the real plant. This is the server side described in the paper in details. To demonstrate the possibilities of a remote control a Java client server application is also introduced. It covers communication and offers a user friendly interface for the control of a remote plant and visualization of measured data.**

***Index Terms*—Client-server systems, JACOB project, Java, MATLAB, Remote laboratories**

I. INTRODUCTION

Influenced by the growth of the Internet the online education has become the regular part of the educational process. In the education of engineers laboratory exercises create the essential part of learning where students can compare theoretical knowledge with practical skills. In the online education classical laboratories are being replaced by remote laboratories [8]. Although using the remote laboratories could not be fully compared with the stay in classical laboratories it can significantly improve the process of getting familiar with the laboratory environment (e.g., virtual tour) as well as it enables acquiring measured data and their processing. Remote laboratories differ from virtual laboratories as they provide students with real data. Virtual laboratories rely on simulations so the data are not real (Fig. 1).

There are several ways how to supply data to remote users. Very often a client server architecture is used for the exchange of data between remote sites. This paper is focused on the server side of a client server application. The structure of the server side can be very different with respect to the services that should be provided to remote clients. This contribution describes the server that is based on the Java programming language and the COM technology for the data exchange. Similar structures have already been described [2-6] but the novelty of this approach consists in the usage of JACOB project results and their modification. This allows to avoid the necessity of using Microsoft Java Virtual Machine (MS JVM) as it was the case in the previous our solutions. For the future this is important also for that fact that from the year 2009 MS JVM is no more supported by Microsoft and so it is very hard to implement it in new Windows operating systems.

II. REMOTE LABORATORY BASIC ELEMENTS

First review basic elements that create a remote laboratory. As it is depicted in the Fig. 1 the remote laboratory consists from a client side and a server side that are interconnected through the Internet. The server side includes the server with its hardware, software, and additional equipment that represents a real system. The basic task of the remote laboratory is to provide clients with possibilities to operate with remote real systems. Sometimes it means only to monitor how the real system behaves or measure its characteristics. But in different cases remote users can interact with the real system and see its responses. This time remote users can actively influence the behavior of the real systems or in other words they can control it. The precondition is that it is possible to control the real system by a computer (local control at the server side). If this is fulfilled then the server must manage the data flow between the client and the real system to enable the remote laboratory will operate correctly.

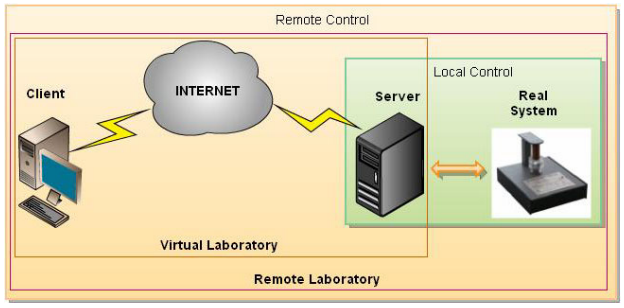


Figure 1. Remote and virtual laboratory structure

III. JAVA SERVER BASED ON JACOB PROJECT

As it was mentioned above the server mediates the communication between clients and the real system. Usually it transfers client's commands to the real system and returns the response data to the client. By this way it operates like a proxy server. But the server can also perform additional tasks, e.g., administration of user accounts and control of a remote laboratory approach. In the following we will discuss the basic functionality of the server and describe the process of the data exchange in details. We will not concentrate on the standard TCP/IP communication between the server and the client that is realized through the Internet. But we will focus on the data flow between the server and the application that directly communicates with the real system. In our case it is the Matlab application. This means that all our real systems are locally controlled by the Matlab.

Using the Matlab under the Windows operating system we have several possibilities to communicate with other applications. If we choose that Java will be the server programming language then these possibilities are limited to few ones.

The old technology of the data exchange called Dynamic Data Exchange (DDE) is no more developing from the Matlab version 5.1. The following technology called Component object model (COM) is well supported in Matlab but not supported in Java. On the other hand the platform independent technology Common Object Request Broker Architecture (CORBA) is supported by Java but not by Matlab. To find the solution of the data exchange between Matlab and Java we have had to look for additional software that would enable this kind of communication. We found the JACOB project that serves for the interconnection of Java and COM technology as it is written in the full title of this project: Java COM Bridge. This determines that from the above mentioned possibilities the COM technology has been finally chosen for the data exchange.

A. COM technology

The COM technology has been developed by Microsoft and so it is limited to Windows operating systems. It defines a binary standard that is language independent and serves for component interoperability. It allows communication between running processes and dynamic creation of objects in programming languages that support this technology (unfortunately, Java does not belong to them). By the use of the COM technology developers and end users can choose application specific components supplied by different producers and integrate them into a complete application solution. The principle of COM consists in a language neutral way of implementing objects. Then these can be used in different environments than they were created in, even across machine boundaries (if we consider DCOM – Distributed COM).

Using the COM technology Matlab can play the role of a controller or be controlled by another component. We will choose the second possibility when the Matlab acts as

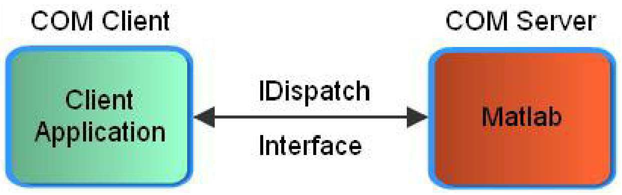


Figure 2. Matlab as the COM Server object

the server (Fig. 2). There are still two ways how to run the Matlab in the server mode. We are not interested in the case when the Matlab runs as the Engine Server. This case is solved by the well-known JMatLink library [7] that also interconnects Java with the Matlab. The disadvantage of this solution is that it does not allow to enter the External Mode of the Matlab that is necessary for compilation of real time algorithms. Therefore we have chosen the second possibility and run the Matlab as the Automation Server that is suitable also for fast real plants with the use of Real Time Workshop.

B. JACOB project

Up to now we have used Microsoft Java in order to approach COM objects [3] but Microsoft finished to support Java in the year 2009 and its software Microsoft Java Virtual Machine has become obsolete. So we have decided to use the JACOB project that offers a possibility to run a Java application in any Java Virtual Machine.

The JACOB project has been developed to create the bridge between Java and COM objects (JACOB - JAva COm Bridge). It enables to call COM Automation components from Java. It uses JNI to make native calls into the COM and Win32 libraries [1]. We have tried to use the results of this project earlier but we have not been successful in getting data from the Matlab workspace. After some time we have discovered that there is a small incompatibility between JACOB and Matlab when arguments of calling methods are misunderstood. After small modification of the JACOB's .dll file we were able to use JACOB libraries and so communicate with the Matlab Automation Server. In the Fig. 3 one can see the structure of the Java server based on the JACOB project. The Fig. 4 shows the server monitor windows during the data transmission.

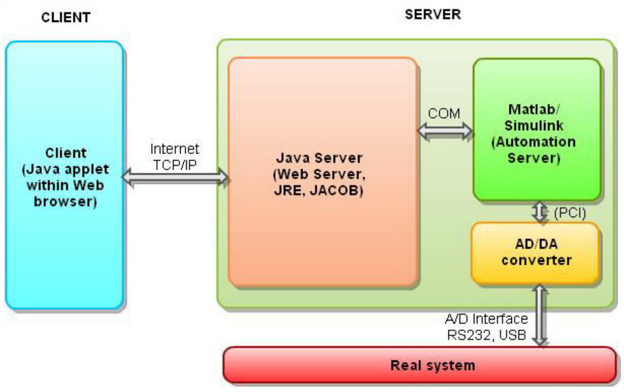
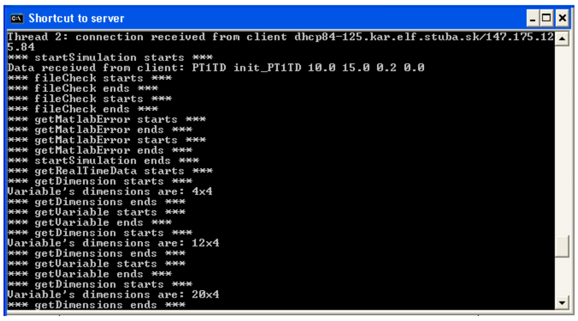

CLIENT
SERVER
Client (Java applet within Web browser)
Internet TCP/IP
Java Server (Web Server, JRE, JACOB)
COM
Matlab/ Simulink (Automation Server)
(PCI)
AD/DA converter
A/D Interface RS232, USB
Real system

Figure 3. Structure of the Java server



```
Thread 2: connection received from client dhcp84-125.kar.elf.stuba.sk/147.175.12
5.84
*** startSimulation starts ***
Data received from client: PT1TD init_PT1TD 10.0 15.0 0.2 0.0
*** fileCheck starts ***
*** fileCheck ends ***
*** fileCheck starts ***
*** fileCheck ends ***
*** getMatlabError starts ***
*** getMatlabError ends ***
*** getMatlabError starts ***
*** getMatlabError ends ***
*** startSimulation ends ***
*** getRealTimeData starts ***
*** getDimension starts ***
Variable's dimensions are: 4x4
*** getDimensions ends ***
*** getVariable starts ***
*** getVariable ends ***
*** getDimension starts ***
Variable's dimensions are: 12x4
*** getDimensions ends ***
*** getVariable starts ***
*** getVariable ends ***
*** getDimension starts ***
Variable's dimensions are: 20x4
*** getDimensions ends ***
```

Figure 4. Server application monitor window

The problem with the above mentioned incompatibility appeared by the calling the method GetWorkspaceData that serves for the transfer of a one-dimensional variable from the Matlab environment. Matlab required the last parameter should be the reference but references of VARIANT types were not implemented in JACOB and when there was an attempt to use them, the system announced an exception. This problem has been corrected by the modification of the JACOB's .dll file. If the method GetWorkspaceData is called its arguments are modified so that the new VARIANT of the VT_R8 type is created and the original one is overwritten by the type VT_VARIANT+VT_BYREF that points to the newly created VARIANT of the type VT_R8. After calling IDispatch.Invoke and its successful termination the newly created VARIANT is copied to the place of the original one and sent to the Java server for processing. A similar procedure has been applied to correct calls of the GetFullMatrix method that serves for the transfer of arrays.

IV. CLIENT APPLICATION

The client application serves as the user interface so it should be designed friendly. First it has to connect to the specified address and port of the server in order to receive data. After the connection is established the data exchange can start. Usually the client sends Matlab commands and waits for responses from the server. The user does not need to know the syntax of Matlab commands. They are mostly hidden under buttons and text boxes of the client application. The responses from the server can be visualized in different forms:

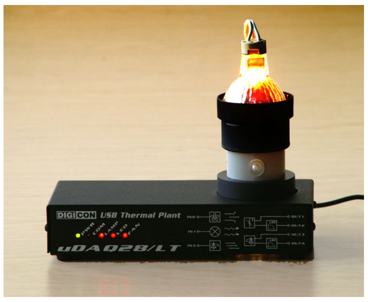
- text messages
- numerical data
- graphs
- animations
- video clips

The client application reads default parameters for each system from a corresponding configuration file (Fig. 5). The configuration file contains information about all systems involved in the remote laboratory and because it is in the textual form it is easy to add a new system by simple copy and past commands to extend the file by a new section. Then it is necessary to modify parameters corresponding to the new system as the name of the system, the Simulink model name, the name of the file with initial Matlab parameters, the desired value, the sampling period, the time of the experiment, the file for saving outputs and the IP address of the server application. Thus the extension and modification of the configuration file provides the flexibility in creation of client applications for new remote laboratories.

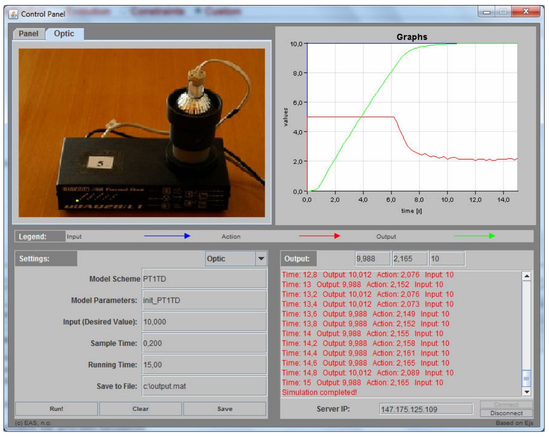
```
*Optic
Model:PT1TD
Parameters:init_PT1TD
w:20.0
Ts:0.20
T:20
Save:c:\output.mat
URL:147.175.125.109
```

Figure 5. Configuration parameters for the optical system

To demonstrate features of the Java server application based on the JACOB project we have used the universal client application for different real systems. From the control point of view the thermo-optical and the hydraulic systems (Fig. 6, 7) belong to the slow systems (their time constants are big enough so the control algorithm can run with a longer sampling period). The other two systems (the rotational pendulum and the magnetic levitation system – Fig. 8, 9) are fast systems and they require to run the Matlab in the mode of the Automation Server.



a)



Settings: Optic

Model Scheme: PT1TD
Model Parameters: init_PT1TD
Input (Desired Value): 10,000
Sample Time: 0,200
Running Time: 15,00
Save to File: c:\output.mat

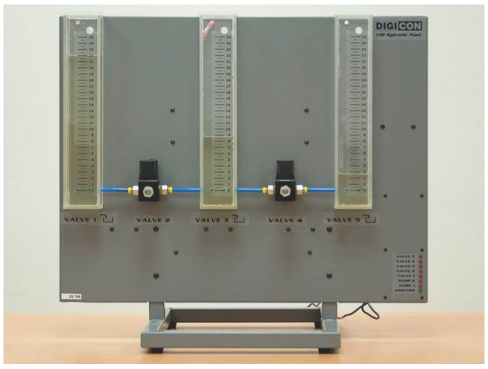
Output: 9,988 2,165 10

```
Time: 12,8  Output: 10,012  Action: 2,076  Input: 10
Time: 13  Output: 9,988  Action: 2,152  Input: 10
Time: 13,2  Output: 10,012  Action: 2,076  Input: 10
Time: 13,4  Output: 10,012  Action: 2,073  Input: 10
Time: 13,6  Output: 9,988  Action: 2,149  Input: 10
Time: 13,8  Output: 9,988  Action: 2,152  Input: 10
Time: 14  Output: 9,988  Action: 2,155  Input: 10
Time: 14,2  Output: 9,988  Action: 2,158  Input: 10
Time: 14,4  Output: 9,988  Action: 2,161  Input: 10
Time: 14,6  Output: 9,988  Action: 2,165  Input: 10
Time: 14,8  Output: 10,012  Action: 2,089  Input: 10
Time: 15  Output: 9,988  Action: 2,165  Input: 10
Simulation completed!
```

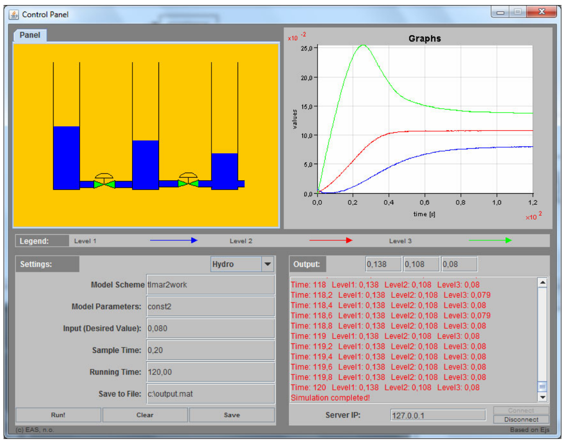
Server IP: 147.175.125.109

b)

Figure 6. Thermo-optical system: a) real system; b) client application



a)



Settings: Hydro

Model Scheme: tlmar2work
Model Parameters: const2
Input (Desired Value): 0,080
Sample Time: 0,20
Running Time: 120,00
Save to File: c:\output.mat

Output: 0,138 0,108 0,08

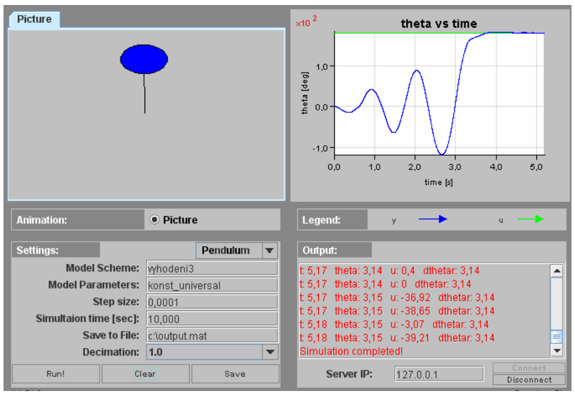
```
Time: 118  Level1: 0,138  Level2: 0,108  Level3: 0,08
Time: 118,2  Level1: 0,138  Level2: 0,108  Level3: 0,079
Time: 118,4  Level1: 0,138  Level2: 0,108  Level3: 0,08
Time: 118,6  Level1: 0,138  Level2: 0,108  Level3: 0,079
Time: 118,8  Level1: 0,138  Level2: 0,108  Level3: 0,08
Time: 119  Level1: 0,138  Level2: 0,108  Level3: 0,08
Time: 119,2  Level1: 0,138  Level2: 0,108  Level3: 0,08
Time: 119,4  Level1: 0,138  Level2: 0,108  Level3: 0,08
Time: 119,6  Level1: 0,138  Level2: 0,108  Level3: 0,08
Time: 119,8  Level1: 0,138  Level2: 0,108  Level3: 0,08
Time: 120  Level1: 0,138  Level2: 0,108  Level3: 0,08
Simulation completed!
```

Server IP: 127.0.0.1

b)

Figure 7. Three-tank hydraulic system: a) real system; b) client application

a)



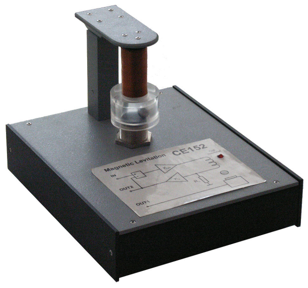
Animation: [✓ Picture]
Legend: y 🟦 u 🟩

Settings: Pendulum
Model Scheme: vyhodeni3
Model Parameters: konst_universal
Step size: 0,0001
Simultaion time [sec]: 10,000
Save to File: c:\output.mat
Decimation: 1.0
Run! Clear Save

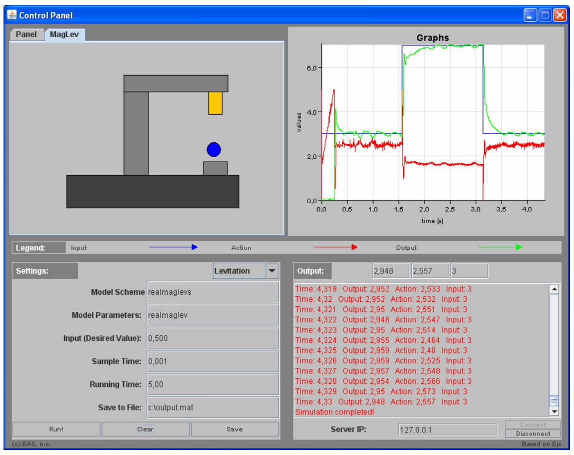
Output:
```
t: 5,17   theta: 3,14   u: 0,4   dthetar: 3,14
t: 5,17   theta: 3,14   u: 0   dthetar: 3,14
t: 5,17   theta: 3,15   u: -36,92   dthetar: 3,14
t: 5,17   theta: 3,15   u: -38,65   dthetar: 3,14
t: 5,18   theta: 3,15   u: -3,07   dthetar: 3,14
t: 5,18   theta: 3,15   u: -39,21   dthetar: 3,14
Simulation completed!
```
Server IP: 127.0.0.1
Disconnect

b)

Figure 8. Rotational pendulum: a) real system; b) client application



a)



Legend: Input 🟦 Action 🟥 Output 🟩

Settings: Levitation
Model Scheme realmaglevs
Model Parameters: realmaglev
Input (Desired Value): 0,500
Sample Time: 0,001
Running Time: 5,00
Save to File: c:\output.mat
Run! Clear Save

Output: 2,948 2,557 3
```
Time: 4,319   Output: 2,952   Action: 2,533   Input: 3
Time: 4,32   Output: 2,952   Action: 2,532   Input: 3
Time: 4,321   Output: 2,95   Action: 2,551   Input: 3
Time: 4,322   Output: 2,948   Action: 2,547   Input: 3
Time: 4,323   Output: 2,95   Action: 2,514   Input: 3
Time: 4,324   Output: 2,955   Action: 2,464   Input: 3
Time: 4,325   Output: 2,959   Action: 2,48   Input: 3
Time: 4,326   Output: 2,959   Action: 2,525   Input: 3
Time: 4,327   Output: 2,957   Action: 2,548   Input: 3
Time: 4,328   Output: 2,954   Action: 2,566   Input: 3
Time: 4,329   Output: 2,95   Action: 2,573   Input: 3
Time: 4,33   Output: 2,948   Action: 2,557   Input: 3
Simulation completed!
```
Server IP: 127.0.0.1
Disconnect
(c) EAS, n.o.
Based on Ejs

b)

Figure 9. Magnetic levitation: a) real system; b) client application

V. CONCLUSION

The newly developed Java server application for the remote laboratories that is based on the results of the JACOB project is no more dependent on the Microsoft Java Virtual Machine. For the future this seems to be very important because Microsoft does not support Java Virtual machine any more. We have to mention that it was necessary to slightly modify the JACOB's files. Finally, we have tested the developed software with several client applications for different real systems and it has been shown that it was stable under different Windows operating systems and different versions of the Matlab software. This means that the classical infrastructure of control engineering laboratories can be converted to the remote laboratories in a convenient way. Students welcome this process as they appreciate visiting remote laboratories very much.

ACKNOWLEDGMENT

The work has been partially supported by the Grant KEGA No. 3/7245/09 and by the Grant VEGA No. 1/0656/09. It was also supported by a grant (No. NIL-I-007-d) from Iceland, Liechtenstein and Norway through the EEA Financial Mechanism and the Norwegian Financial Mechanism. This project is also co-financed from the state budget of the Slovak Republic.

REFERENCES

[1] D. Adler, The JACOB Project, http://danadler.com/jacob/

[2] P. Bisták, "Remote control of thermal plant using Easy Java Simulation," *Proceedings of Int. Conf. on Interactive Computer Aided Learning ICL'06*, Villach, Austria, 2006.

[3] P. Bisták, "Remote Laboratory Java Server for Data Exchange with Matlab Automation Server," *Proceedings of International Conference REV 2008*, Düsseldorf, Germany, 23-25 June, 2008.

[4] R. Safaric,, S. Uran, M. Trunic, I. Hedrih, "Remote controlled mechatronics device via internet using Matlab," *Proceedings of 1st Int. REV Symposium*, Villach, Austria, 2004.

[5] Y. Sheng, W. Wang, J. Wang, J. Chen, "A Virtual Laboratory Platform Based on Integration of Java and Matlab," *Li, F. et al (Eds.) ICWL 2008*, LNCS, vol. 5145, Springer-Verlag Berlin Heidelberg, 2008, pp. 285-295.

[6] V. Žilka, P. Bisták, P. Kurčík, "Hydraulic Plant Remote Laboratory," *International Journal of Online Engineering*, ISSN 1861-2121, vol. 4, Special Issue 1: REV2008, July 2008, pp. 69-73.

[7] S. Müller, H. Waller, "Efficient integration of real-time hardware and Web based services into MATLAB," *Proceedings of 11th European Simulation Symposium*, Erlangen, Germany, 1999.

[8] L. Gomes, S. Bogosyan, "Current Trends in Remote Laboratories," *IEEE Trans. Industrial Electronics*, vol. 56, No. 12, pp. 4744- 4756, December 2009. doi:10.1109/TIE.2009.2033293

AUTHORS

P. Bisták is with the Institute of Control and Industrial Informatics, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, STU FEI Bratislava, Ilkovicova 3, 812 19 Bratislava. (e-mail: pavol.bistak@stuba.sk).

P. Folvarčík is with the Institute of Control and Industrial Informatics, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, STU FEI Bratislava, Ilkovicova 3, 812 19 Bratislava. (e-mail: pavol.folvarcik@stuba.sk).