World Transactions on Engineering and Technology Education
Vol.1, No.2, 2002

How and why to build and use virtual laboratories

George S. Georgiev†, Hubert Roth‡, Silvia Stefanova†, Georgi T. Georgiev†, Emil Stoyanov† & Otto Rösch‡

Rousse University, Rousse, Bulgaria†
University of Siegen, Siegen, Germany‡

ABSTRACT: This article presents the authors' experiences in building virtual laboratories and provides discussions on issues of importance and relevance with regard to the pedagogy, software and equipment utilised. The authors comment on the potential laboratory structures from the point of view of the Distributed Laboratories' general purposes, which include: the cost effectiveness and platform independence, time and space constraints, as well as the balance between the traditional and virtual forms of teaching. The structure and the organisation of the laboratory are proposed and special attention is being given to the physical instruments, the possibility for their platform independent control from the client-student and their synchronisation in a multitask environment such as Windows.

INTRODUCTION

The rapid development of information technologies has dramatically changed the acquisition, manipulation and transmission of knowledge. They considerably increase the number of opportunities for its procurement by means of information exchange, communication and collaboration by overcoming time and space constraints. Universities' main links in the knowledge acquisition are strongly influenced by these factors. Their functions for the creation, preservation, integration, teaching and application of knowledge find a new interpretation [1]. New forms and models of teaching, such as virtual structures for laboratories, classes and universities, are an alternative of the traditional education methods [2].

Approaches in building virtual labs are discussed elsewhere [6-10]. It is essential that platform independence is ensured in order to obtain access to remote controlled labs and overcoming license restrictions [9]. Although the question of didactical opportunities is not bound to the technical realisation, it remains open for discussion [1][6]. The aims of the present work are firstly to discuss the disputed issue concerning the didactic value of new Internet technologies and to share the authors' views in this respect. Secondly, a new systematic approach for building remote labs is proposed to emphasise the platform independence of virtual instruments and their synchronisation on the physical layer.

DIDACTICAL ISSUE

As with every new technology, some problems arose concerning the realisation and implementation of virtual labs in education, such as an adequate, well mobilised, compact, economically justified and didactic learning environment [3]. The question concerning the didactical value usually remains separate without a clear answer.

The powerful features underlying new technologies give some authors reason to consider that traditional teaching can be completely replaced by these new forms, which utilise multiple levels of teaching materials structuring as hyper pages with connections, the use of simulators and multimedia techniques that help students to learn effectively and experiment remotely without time and place constraints [1][2]. It is thought that the student's isolation, typical for the virtual remote teaching, may be overcome primarily by technical means in contemporary Internet technologies, including online conferences, seminars, discussion groups, opportunities for computer-based quizzes and instant feedback.

A precise and formalised analysis is needed to properly place new didactical meanings in such a way that they stimulate the learner to gain knowledge according to specific actions. It is considered that the approach of actors, as proposed by Ahmavaara, is appropriate for these purposes [4]. This approach is based on a confrontation between human abilities and technical structures, as it is shown that the latter are limited in their reactions and interactions with the surrounding environment.

Firstly, a human thinks about the realisation of aims before undertaking the corresponding actions. The aims of these predetermined actions are dependent on his/her subjective values and knowledge about the situation. Except when expressing individual free will and having personal goals to achieve, a person still complies with objective laws and, in this sense, is a causal system. Combining the two approaches distinguishes a human as an *actor-system*.

In a real situation, an actor does not act on his/her own. For example, when a student-actor is experimenting in a traditional lab, he/she interacts immediately with a teacher-actor. From the traditional educational point of view, it would commonly be

necessary to investigate the interaction of N actors forming a general actor-system that jointly states and acts can be presented as follows:

$$x_s = (x_1, x_2, ..., x_N), f_s = f_1 \wedge f_2 ..... \wedge f_N \quad (1)$$

Even though each actor has as a precondition his own values of states B_{β_i}, his actions are added to the total crossings of the effects in conceivable world's states X (see Figure 1):

$$C_\beta(t) = C_{\beta_1}(t) \cap C_{\beta_2}(t) .... \cap C_{\beta_N}(t) \quad (2)$$

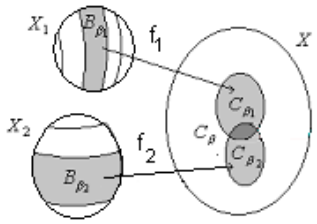


Figure 1: Actor's interaction.

It is shown in [4] that, in order to generate a transition from one state to another, $f_1 \wedge f_2 .... \wedge f_N : X_s \to X_s$ - a basic characteristic of the dynamic systems, a system function $S^t(\beta)$, can be defined such as:

$$x_s(\tau) \to x_s(t) = S^t(T_1^{t\tau}(x_1(\tau), T_2^{t\tau}(x_2(\tau)) ... T_N^{t\tau}(x_N(\tau)) \quad (3)$$

This unifies the efforts of the single actors through merging their joint intents, ie:

$$x(\tau)_s \to x_s(t) = S^t(T_1^{t\tau}(x_1(\tau), T_2^{t\tau}(x_2(\tau)) ... T_N^{t\tau}(x_N(\tau)) \quad (4)$$

This represents the joint requirement for all genuinely self-steering machines to which the person belongs.

A more detailed view of the component structure of (4) shows that the generated new states x_i in the common actor-system depend equally as much on the exits-intents y_i of the individuals, an expression of their own subjective goals, as on the exits-intents $y_j (i \neq j)$ of other actors-participants. Apart from this, the use of different entrances to technical systems separates their internal structure from the external environment. They perceive the external influences through receptors connected to the external entrances. Their internal structure determined as preliminary remains untouched and encapsulated inside the system [10]. This can be generalised as follows:

- The teaching process, comprising an actor-student and an actor-teacher, develops as an action in an actor-system and has its system characteristics.
- Knowledge transmission between the actors (participants in the actor-system) is carried out as a sequence of actions along the chain:

  knowledge $\Rightarrow$ cognitive beliefs/values $\Rightarrow$ norms

  by means of the teacher's values delivered to the student under the operation of merging their intents function $S^t(y_i, y_j, i \neq j)$. This is a bilateral process and is individual from the perspective of the final results according to the component record:

  $x_i(t) = S_i^t(T_i^{t\tau}(x_i(\tau)), T_j^{t\tau}(x_j(\tau)), i \neq j)$.

  Mutual knowledge delivery does not finish the same for all participants. It has an original character and is based on knowledge generation, which is one of the essential functions of universities [2].
- The technical systems that separate themselves from the surrounding environment by means of the external entrances cannot completely deliver their behaviour to the environment's elements-actors. If they are inappropriate mediators between the participants-actors in the teaching actor-system, then they will reduce its effectiveness for the knowledge delivery and generation.
- Humans can explore the environment better than technical systems, perceiving it as a behaviour of co-actors. Humans react on their *intents* throughout the whole time of interaction, rearranging internal cognitive systems (2). This is a continuous process in time and space.

All this does not detract from the usefulness of virtual labs. It is clear from Figure 1 that the intents of the actors operating in a joint actor-system during the teaching process can be merged only if the consequences C_{β_i}, C_{β_j} are crossed; this being a continuous time process (2)(3). Increasing numbers of students and limited resources if time and space are the main limitations in today's universities. Interaction within the teacher-student system can be vastly improved. The teaching process can gain the necessary level of intensification through virtual labs, where students can learn without the above-mentioned constraints.

JAVA COMMUNICATION VIRTUAL LAB STRUCTURE

Platform independence is an important condition for clients-students to obtain wide access to the remote lab client-server modules in Java. Their integrative implementation is a powerful method for dialogue and remote procedures; this is why they provide the basis of the proposed communication structure (see Figure 2). The communication network ports are shown, while the initiator of the connection is the component marked with local port, which is the client. The server is the acceptor and it provides the remote port. Network connections are numerated.

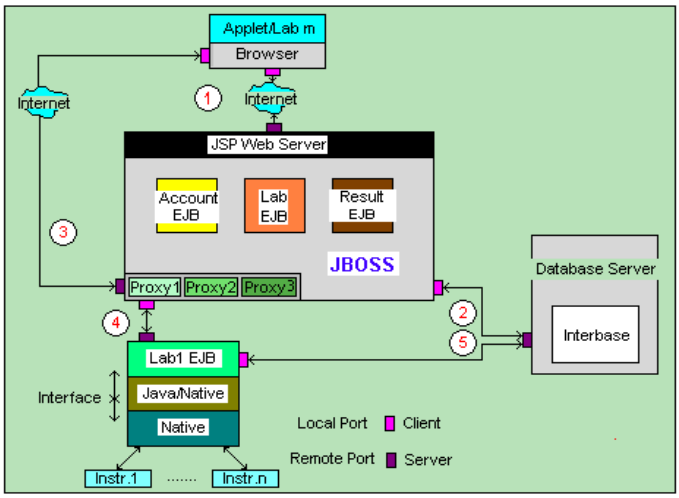


Figure 2: Java communication structure.

The sequence of the client's access to the lab resources is as follows. The central server (*JBOSS* application server: http://www.jboss.org) provides the Web server (Jakarta Tomcat 3.2.x: http://jakarta.apache.org) to which the browser is connected (step 1). Registration, authorisation and information downloads are generated by the JSP engine pages. An internal verification to the database server is realised through executing

an SQL query using *AccountEJB* (step 2). The *AccountEJB* component estimates the validity of the user identification and grants an access to the available laboratory applets. After being downloaded, the desired applet connects to the *JBOSS* central server (step 3) using the associated proxy EJB component. Proxy EJBs are used only as receiver-transmitters of requests for remote procedures execution on the basis of earlier established authorisation (step 4). These execute remote procedures of components located at the laboratory servers (also *JBOSS* EJB Server) that control the devices under test. After finishing the procedure, the laboratory server connects to the database server (step 5) to write the operation results. The final results then return to the calling applet through the proxy EJB.

Locking control and availability of the remotely controlled device is done in the proxy EJBs and the laboratory server. Such a structure allows for *parallel* processing:

- Steps 1 and 2 can be processed by multiple users simultaneously.
- Step 3 to 5 can be processed simultaneously by as many users as the laboratory server allows, usually the number of attached devices to one laboratory server.
- Any of the server components of the system (central server, lab server, database server) may grant access to various users at the same time (eg SQL cl/s connected to the database server) but authorisation is required.

The general model of the software-controlled device is shown in Figure 3. *Controlling unit functions* are the main commands of the controlling system (DAQ system) whereas the *device functions* are in the core of the *model*: they are specific for the system and usually sequence the functions or procedures to the controlling unit. The Graphical User Interface (GUI) is the visual representation of the device. The GUI calls only the device model functions and cannot directly call the controlling unit functions.

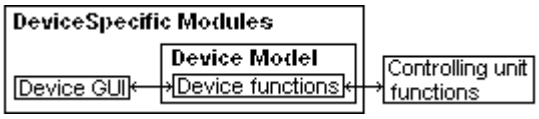


Figure 3: General model of the software-controlled device.

In order to apply this model to the client-server model, it is necessary to separate the parts dealing with user interaction (GUI) and the controlling unit so the client to be responsible for graphic representation and remote *device functions* calling. This means the device model has to be implemented as client and server parts.

The client executes remote procedures that are specific for the Device Under Test (DUT). Functions are located on a server (central, laboratory or node). It does not matter through how many nodes the system call will pass; it finally reaches the laboratory server. In turn, it provides the required system calls as remote functions that are executable by means of an RMI/IIOP mechanism. Generally, all remote functions are encapsulated in EJBs, which call native functions containing code that controls the DUT (see Figure 4).

Figure 4 shows that the client consists of the GUI and its part of the device functions (device model). The server is the controlling unit functions module and it also has part of the device model. Access to the device is arranged as follows: Java is capable of executing native functions through Java Native Interface (JNI). The JNI serves as the glue between Java and native applications, with functions written in C language, and the device can be accessed directly or by calling driver procedures. In this case, access is organised by driver functions to make the system more flexible and standard.

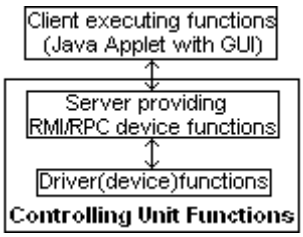
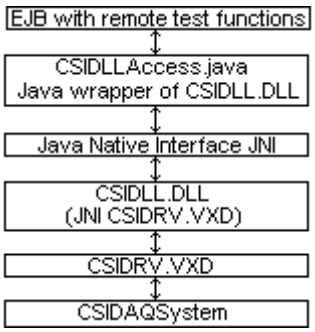


Figure 4: Remote procedures execution.

Figure 5 shows a detailed view of the layers through which a call to the system method passes. Every EJB has functions specific for the device it represents (device model). These functions should be transformed into a sequence of basic commands understandable by the controlling unit (or *measurement system*, marked as CSI). In this case, devices are controlled by the CSI and every EJB has class *CSIDLLAccess*, which allows the core CSI functions (din, aot, etc) to be executed. By means of these classes, it is possible to model and control any device connected to the CSI. *CSYDLLAccess* makes JNI call to a library containing the wrapping code of the CSI driver, manipulating it directly. To complete this discussion, the physical instrumentation layer should be looked at.



Note:
EJB: gives methods executable from the net.
CSIDLLAccess.java: mirror of the library accessing the device driver (acts as a Java bridge to the native functions).
Java Native Interface: integrated in the JVM, translates Java to C data types and object calls.
CSI.DLL: library compiled with JNI support, its functions access the device driver.
CSIDRV.VXD: device driver – system level.
CSIDAQSystem: the controlling unit (physical layer).

Figure 5: Detailed view of the layers and their descriptions.

PHYSICAL LAYER

Implementing real-time processing in multitasking environments such as Windows is very difficult. This is because system resources are granted to each running task in time slices (about $1\,\mu s$) according to the task's priorities. A task can raise its priority but that would impede the performance of other tasks.

Even running at highest priority, a task is not guaranteed to hold the system resources, most importantly the CPU, all the time. Another problem affecting measurement is that multi-tasking environments tasks are heavily protected from each other so that one task can not access memory that may belong to any other task. Intel processor-based PCs protection is implemented by assigning two very different privilege levels to each piece of running code: system (or supervisor) level and user (application) level. The latter level cannot access IO ports (this is especially true for Windows NT), nor process *interrupt* requests. So, any code that accesses hardware directly should work at the system level.

To make matters worse, code running at the system level cannot directly call user level code. Even the simplest measurement of voltage using an ADC takes some time to complete. Most implementations either generate an interrupt request or raise a flag to indicate that measurement has been completed. One way of synchronising is to just wait in the system level code until the flag raises, or an interrupt request is generated (in this case there is really no difference between these two approaches), as illustrated in Figure 6.

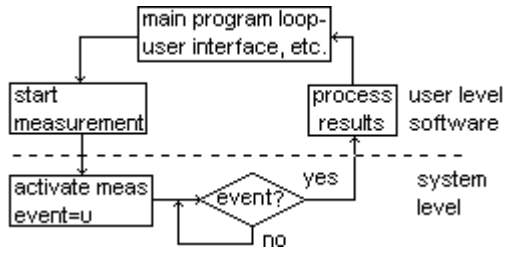


Figure 6: The simplest way of dealing with synchronisation.

This scenario suffers from two major problems. On the one hand, Windows' task scheduler still periodically gives a slice of time from the CPU to the measurement task, even if it is only checking a flag. This would impede performance of all other tasks in the system, especially if the priority of the measuring task is above normal. On the other hand, there is no guarantee that the reaction to the finishing of the measurement will be prompt. In fact the conversion may end while another task occupies the CPU; then Windows may yield the CPU to still another task. In general, several time slices may pass before the measurement task becomes aware of the event that conversion has been completed.

There are at least two ways to deal with these problems. The performance impediment problem can be dealt with entirely, while the second problem can be reduced to a delay of no more than one time slice. One solution is to use Windows' message passing mechanism, as shown in Figure 7.

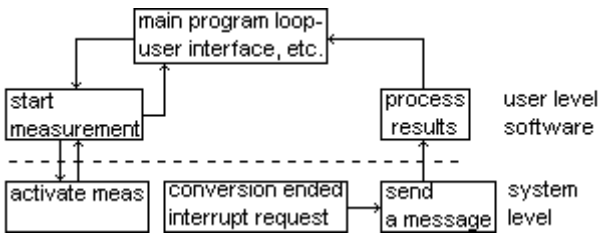


Figure 7: Synchronisation through message passing.

In this case, control is returned to the main program loop as soon as the conversion starts. The main loop continues running; if there is no user activity. The task will again do nothing, but this time it will not be given any resources by the Windows' task scheduler. This is guaranteed by the very way the main program loop is organised in each Windows program. All Windows programs are necessarily event-driven; if, for a given time interval, there are no events concerning a particular program (task), it is placed in an idle state without any performance penalties to other tasks.

Another solution is to use a synchronisation object (see Figure 8). A semaphore is a typical example: Windows offers other objects to accommodate the needs of each particular synchronisation scenario. In this case, an *event* is more appropriate. Here, again, control is returned to the main program loop immediately. However, this time, the main program has to create a second thread, waiting for the event to become signalled. This waiting is again performance penalty free due to the way Windows uses synchronisation objects. This implementation uses the approach with message passing because it is nearer to natural event-driven programming techniques. The reaction to the message can easily be implemented as a method in a class, so the implementation of an object oriented approach is also not a problem.

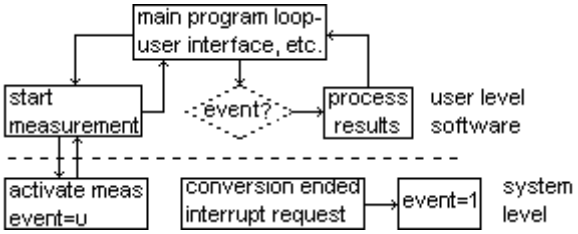


Figure 8: Synchronisation through an event.

CONCLUSIONS

Virtual technical and technological meanings and systems can implement and support only a part of the basic functions of the universities connected to the preservation and delivery of knowledge.

The idea is for universities to turn into *knowledge servers* [1]. This is based completely on new information technologies and limits the interaction between the actors (participants in a common teaching system). It also delivers the opportunity for the main university functions to be realised. The implementation of the preservation, delivery, generation and application of knowledge requires a balanced and reasonable combination of the two teaching forms: traditional and virtual.

REFERENCES

1. Duderstadt, J.J., The future of the university in age of knowledge. *J. of Asynchronous Learning Networks*, 1, **2**, 78-88 (1997).
2. Bourne, J.R. et al, Learning network in engineering education. *J. of Asynchronous Learning Networks*, 1, **1**, 18-23 (1997).
3. Hanna, D., Higher education in era of digital competion: emerging organizational models. *J. of Asynchronous Learning Networks*, 2, **1** (1998).

4. Ahmavaara, A., The impossibility of genuinely self-steering machines: a fundamental theorem on actor-systems. *Cybernetics*, 10, 113-121 (1981).
5. Mesarovic, M.D. and Takahara, Y., *General System Theory: Mathematical Foundations*. New York: Academic Press (1975).
6. Benetazzo, L., et al, A Web-based distributed virtual educational laboratory. *IEEE Trans. on Instrumentation and Measurement*, 49, **2**, 349-356 (2000).
7. Arpaia, P. et al, A measurement laboratory on geographic network for remote test control. *IEEE Trans. on Instrumentation and Measurement*, 49, **5**, 992-996 (2000).
8. Ko, C.B. et al, A large-scale Web-based virtual oscilloscope laboratory experiment. *Engng. Science and Educ. J.*, April, 69-76 (2000).
9. Fortino, G., et al, Distributed measurement patterns based on Java and Web tools. *IEEE Trans. on Instrumentation and Measurement*, 47, **7**, 624-628 (1997).
10. Rodriguez, F. et al, A remote laboratory for teaching mobile robots. *Proc. 1st IFAC Conf., Telematics Applications in Automation and Robotics*, TA, 307-311 (2001).

The Global Journal of Engineering Education

The UICEE's *Global Journal of Engineering Education* (GJEE) was launched by the Director-General of UNESCO, Dr Frederico Mayor at the April meeting of the UNESCO International Committee on Engineering Education (ICEE), held at UNESCO headquarters in Paris, France, in 1997.

The GJEE is set to become a benchmark for journals of engineering education. It is edited by the UICEE Director, Prof. Zenon J. Pudlowski, and has an impressive advisory board, comprising close to 30 distinguished academics from around the world.

The Journal is a further step in the Centre's quest to fulfil its commission of human resources development within engineering through engineering education, in this instance, by providing both a global forum for debate on, and research and development into, issues of importance to engineering education, and a vehicle for the global transfer of such discourse.

In the first five years of the Journal's existence, 220 papers over 1,670 pages have been published, including award-winning papers from UICEE conferences held around the world. Papers have tackled issues of multimedia in engineering education, international collaboration, women in engineering education, curriculum development, the future of engineering education, the World Wide Web and the value of international experience, to name just a few. Other examples include: Vol.3, No.1 was dedicated to papers on quality issues in engineering education; Vol.3, No.3 focused on papers given at the 1st Conference on Life-Long Learning for Engineers; Vol.4, No.2 centred on the German Network of Engineering Education and was the first issue published entirely in the German language; Vol.4, No.3 centred on the achievements of the *2nd Global Congress on Engineering Education*, held in Wismar, Germany; while Vol.5, No.2, had a more regional focus on Taiwan.

The GJEE is available to members of the UICEE at an individual member rate of $A100 p.a., or to libraries at a rate of $A200 p.a. (nominally two issues per year, although each volume has included an extra, complementary issue). For further details, contact the UICEE at: UICEE, Faculty of Engineering Monash University, Clayton, Victoria 3800, Australia. Tel: +61 3 990-54977 Fax: +61 3 990-51547, or visit the UICEE Website at:

http://www.eng.monash.edu.au/uicee