Design Virtual Learning Labs for Courses in Computational Science with use of Cloud Computing Technologies

Alexey Dukhanov¹, Maria Karpova¹, Klavdiya Bochenina¹

¹ ITMO University, Saint-Petersburg, Russia
dukhanov@niuitmo.ru, 91ms91@gmail.com, k.bochenina@gmail.com

Abstract
This paper describes the approach to the design and implementation of a virtual learning laboratory (VLL) with the use of cloud computing technologies within the model of AaaS (Application as a Service). The formal model of composite application and a set of learning models using cloud-based VLL are proposed. The relation to learning objectives in accordance with the revised Bloom's taxonomy was identified for each model. The software tool to automate the creation and configuration VLL, based on the cloud computing platform CLAVIRE are presented. The paper ends with the description of case study implementation with the use of the offered approach.

Keywords: Virtual learning lab, the AaaS model, cloud computing, workflow, learning models, learning objectives

1 Introduction

The rise of eLearning technologies requires new forms and approaches to the design and creation of digital educational resources (DER), including virtual learning labs (VLL), which allow carrying out scientific experiments with the use of one or more pieces of expensive elaborate equipment and software in the remote access mode. Put otherwise, the VLL can be considered as a supporting remote access distributed hard-software complex which can simulate any objects and processes. Usually these products contain application software and packages for numerical simulation of various processes, data sources, results interpretation and visualization means, unique and expensive equipment, and other instruments for solving tasks usually in multi- and trans-disciplinary scientific areas. The cloud computing technologies can be considered as a perspective approach [1] for preparing DER. For DER design automation the following models of cloud computing may be considered.

1. IaaS (Infrastructure as a Service). A university’s computational infrastructure is virtualized and shared for the purpose of solving internal tasks, including special
software installation. This infrastructure may be used as a base for the creation and deployment of VLL.

2. PaaS (Platform as a Service). Virtualized resources with installed required packages are shared, but DER applications are absent.

3. SaaS (Software as a Service). The most common model of VLL representation is as a web-service. The successful examples can be found in [2] and [3].

4. DaaS (Data as a Service). It is an additional model for providing data sources as a cloud service.

5. HaaS (Hardware as a Service). This model is used for sharing unique equipment, e.g. near-field scanning optical microscopes [4] through an appropriate interface.

The amount of interdisciplinary research areas and associated educational programs [5] is rising. The design an inter(multi)- or trans-disciplinary course (module) requires application software and hardware in different scientific areas, which are in distributed heterogeneous computational environment. The VLL should be presented to the end user (e.g. student) as a joint-free application instead of the set of software and hardware. The most eligible model of cloud computing for the considered reasons is the model of AaaS (Application as a Service) [6]. This model allows us to create and use composite applications (CA), which consists of interacting cloud services for solving complex tasks: data source access, computer simulation, results processing and analysis. The most popular technique of CA representation is workflow formalism [7], which allows us to describe the corresponding distributed process in terms of the directed graph. This paper offers the set models of cloud VLL application with the use of the AaaS model [8].

2 Related works

The design and development of VLL for providing simple access to unique approaches and technologies is presented in many works. Most of them deal with virtual machines with installed required application software and emulators of special devices and equipment. The access to these machines is provided with the use of software like a remote desktop. For example, the virtual learning laboratory for a Computer Forensics course is considered [9]. More than ten special applications and six devices were required for the design of this course. The authors of this work were able to use a virtual machine (server) running on a Windows 2003 Server, hardware emulators, and equivalents or trial versions of software. Students are proposed to perform several labs outside of the virtual considered environment. The authors of the paper [10] offer to apply WMware application for virtualizing computer networks in the framework of Computer Networking Laboratories design.

Other works offer to use standard cloud decisions like GoogleDocs, iGoogle Gadgets, and original instruments for the creation of personal virtual learning environments (VLE) [11]. It is a very popular approach because it does not require expensive resources and uses well-known cloud instruments.

An original approach is presented in [12], where the authors developed the architecture for the integration of external tools in VLE. This architecture provides a creation VLE within well-known learning management systems (LMS) with the use of external tools with a special program interface and tools adapters. It allows the user to integrate LMS in very popular instruments, such as Google Documents [13], Dabbleboard [14], and Facebook Live Stream [15] thanks to separately developed tools adapters. As a result, this architecture allows a user to apply any software, including unique application packages, if the corresponding tools adapter exists. The presented demonstration [16] shows that the creation of learning resources within Moodle with the use of “GLUE!” can take five times less time than if it was done without “GLUE!”

As can be seen, the above mentioned works use some type of cloud technology. For example, Computer Forensics Labs were developed with the use of IaaS, PaaS, and “GLUE!” – SaaS and HaaS
(in the case of sharing hardware through an adapter) models. For training which requires the use of unique and expensive equipment (or application software) it is necessary to create a virtual machine and install hardware emulators (trial versions of application software) or to develop corresponding tools adapters.

3 Features of the organization of Virtual Learning Labs with the use of the AaaS model

Based on the AaaS model, VLL is presented as software, which operates in a distributed computing environment. It can provide the following features:

1. CA Design and Execution support in a cloud computing environment. The CA is formed with the use of different proven application packages; therefore, the AaaS model is useful for interdisciplinary researches.
2. The user does not have to install a special local client. An internet browser is enough for working with VLL. A variety of packages is used for solving interdisciplinary problems. Setup and maintenance of these software programs can be very difficult for a user. The software developed in accordance with the model of AaaS solved this problem.
3. Intelligent users support. Self-study of the application package may be difficult, therefore, the user often needs support while performing a lab and during the whole learning process.
4. Ensuring information security within an open-circuit system (the features and location of the prospective user are not defined). Protection against legitimate users, which have authorized access to VLL resources but who may have the ability of unauthorized usage of one or more application packages, is important.
5. The efficiency of the combined use of supercomputer’s resources and the resources of grid and cloud computing environments. This feature enables a user to apply application packages which require unique computational resources and special software. The user must be isolated from the layer of computational resources, and as a consequence resource selection for solving particular tasks must be implemented automatically (for e.g. minimizing cost or time of computing).
6. “Hot plug” of packages and computational resources. This “hot plug” does not require hard efforts of users, such as package/resource developers and/or owners. This feature makes VLL open, portable and scalable. When the package becomes a part of the VLL it is not considered as a competitor and is annotated by the overall system of intelligent support. The method of unified annotation of packages and resources is described in section 6

The AaaS model attracts particular attention in the context of the development of interdisciplinary researches. This model enables the user to design and use “Composite applications” (CA) – a complex of interconnected cloud services which are oriented together to solve a whole problem. The workflow-formalism is the most well-known approach of CA-representation. This formalism describes connections between particular operations in the distributed computational environment using the notations of a directed acyclic graph. The complicated CA is created by means of existing software and is executed in distributed resources of the cloud environment.
4 The Formal model of the composite application

CA is represented by the WF model of the directed acyclic graph (DAG) whose nodes are data processing units with the input and output parameters and arcs are the relationships between the nodes. For example, a node may be the launch of the package, procedure, node to vary parameters, conditional branching, and cycle. Each node of composite application represents a tuple \( N_i = \{ \text{Ins, P, Outs, F} \} \), where \( l = 1..k \), \( k \) - quantity of the WF nodes, \( \text{Ins} = \{ I_i : i = 0..n_i - 1, I_i \in \text{Id} \} \) – the set of identifiers of input parameters of the unit, \( \text{Outs} = \{ O_j : i = 0..m_j - 1, O_j \in \text{Id} \} \) – the set of identifiers of output parameters of the unit, \( P : Type(I_0) \times \ldots \times Type(I_n) \rightarrow \{ \text{true, false} \} \) – a set of all identifiers of parameters), – the predicated defining entering of the given set of input values in the feasible region for a package (i.e. validating a correctness), \( Type: \text{Id} \rightarrow B \) – the operator which defines identifier type, \( F \) – the function translating transmitted input values into output values.

\[
F: \{ x \in Type(I_0) \times \ldots \times Type(I_n) \} \times P(x) \rightarrow \{ y \in Type(O_0) \times \ldots \times Type(O_m) \}. \tag{1}
\]

Relationships of two types between nodes can be set of data relations and control relations. Data relation connects the output parameter of one unit to the input of another, and as a result the order of execution of units is also set. Control of relationships explicitly sets an order of execution of units, and this relationship is used as a base for building controlling constructions, such as condition, cycle, or units of a variation of parameters. In general, the CA is a tuple \( \langle \text{Ins}^{wf}, \text{Outs}^{wf}, N, D, C \rangle \), where

\[
\text{Ins}^{wf} = \{ I_i : i = 0..n_i, I_i \in \text{Id} \} \quad \text{– a set of identifiers of input WF parameters},
\]

\[
\text{Outs}^{wf} = \{ O_j : j = 0..m_j, O_j \in \text{Id} \} \quad \text{– a set of identifiers of the output WF parameters},
\]

\[
N = \{ N_j : \text{Node}, l = 0..k \} \quad \text{– a set of the WF nodes (here Node – the WF node),}
\]

\[
D \quad \text{– a set of data dependencies},
\]

\[
C \quad \text{– a set of dependencies on control. In the proposed model the control relation is a twain} \langle \text{Child}^C, \text{Parent}^C \rangle, \text{Child}^C \in N \quad \text{– a dependent node, and Parent}^C \in N \quad \text{– a node on which}
\]

\[
\text{depends, thus Child}^C \neq \text{Parent}^C. \quad \text{Data dependency is the triple} \langle \text{Child}^D, \text{Parents}^D, F_{\text{conv}} \rangle, \text{where}
\]

\[
\text{Child}^D \in \bigcup_{N_j \in N} \text{Ins}^{N_j} \cup \text{Outs}^{wf} \quad \text{– a dependent node}, \quad \text{Parents}^D \in \bigcup_{N_j \in N} \text{Outs}^{N_j} \cup \text{Ins}^{wf} \quad \text{– a set of nodes on}
\]

which \( \text{Child}^D \) depends, \( \text{F}_{\text{conv}} \) – data conversion function. Data relation is set between each (one) input parameter and several outputs:

\[
F_{\text{conv}}: \text{Type}^{\text{Parents}^D_0} \times \ldots \times \text{Type}^{\text{Parents}^D_z} \rightarrow \text{Type}^{\text{Child}^D}, \quad z = |\text{Parents}^D|. \tag{2}
\]

For whole WF or for each of its unit interpretations the \textit{Interpretation} function can be introduced. This function allows the user to transform a set of input data to a set of output data:

\[
\text{Interpretation(WF)}: \text{Type}(I_i) \times \ldots \times \text{Type}(I_n) \rightarrow \text{Type}(O_0) \times \ldots \times \text{Type}(O_m) \tag{3}
\]

4.1 The model’s interpretation for VLLs

Expressions (1)–(3) can be used in both direct and reverse forms for the description of the information processes occurring while the trainee does the VLL. In particular, the direct form reproduces user actions in the case of selecting and configuring the individual application-oriented packages (1), combining them in CA (2), and can also start and perform the whole CA in a distributed environment (3). One can notice that the function (2) makes the CA as a joint-free application for the trainee. Inverse relationships can be interpreted as criteria for checking trainee actions and their success in the passing stages of VLL.
4.2 The model’s example

Let us consider the example in terms of the model mentioned above. Suppose that we have access to the following packages for (Error! Reference source not found.):

1. making of a satellite image and recording it in the spatial storage in the RAW format – \( N_1 \);
2. converting the satellite image in selected standard format (e.g. “png”-format) – \( N_2 \);
3. processing satellite image in required standard graphics format (e.g. identification of atmospheric cyclones) – \( N_3 \);
4. visualization of the results of processing (e.g. drawing isobars) – \( N_4 \).

![Figure 1: The example of workflow in the terms of formal model](image)

Suppose, that the second package \( N_2 \) “knows”, where the satellite image is in a RAW-format, recorded by the first package, and may access it itself and convert it. Then it can be started after the completion of the first package (dependence on control). The third package uses the output of the second package, and the fourth – the output of the third (data dependencies). The process of solving the problem of atmospheric cyclones identification from satellite images can be represented as a WF, shown in the Figure 1. This WF determines four identifiers for input parameters \( I_1,...,I_4 \) (including configuration options for packages; coordinates of area, that to be photographed, for the first package, two identifiers for output parameters (\( O_1 \) - the visualized results of processing, \( O_2 \) - the image in standard format). The identifier \( C_1 \) means the dependence on control between the first and the second packages (the dotted line in the Figure 1). The identifiers \( D_1, D_2 \) mean the data dependences between packages \( N_2, N_3 \) and \( N_3, N_4 \) respectively.

If the node \( N_3 \) supports the same image format, which the second package returns, then \( F_{conv}^{D_1} \) for \( D_1 \) (2) is the identify function. Let the third package return some matrix, and package \( N_4 \) requires the transposed matrix. Then joint-free relation \( D_2 \) requires the function \( F_{conv}^{D_2} = MTrans \), where \( MTrans \) - operator of transposition of matrices. The functions (1) and (3) are described in the next section.

5 Learning models

Based on formal model (1) - (3), there are three models of learning, which reflect approaches to the use of interdisciplinary VLL.
1) Model "A": interpretation function (3) is applied to the WF as a whole (Figure 2, a), and it is assumed that the CA in the form of WF was previously created and debugged. The student receives immediate access to it without the ability to view and change internal structures in the implementation VLL. This approach, with use of the integrated VLL with access to remotely executing application-oriented packages, is used for training specialists. An advantage of the approach is the possibility of the "seamless" combining of packages as a part of the CA which allows a user to expand the range of tasks and composition of operations of the virtual laboratory based on the same application-oriented packages and computational resources.

2) Model "B": function of interpretation (3) is applied to WF in units, but the user cannot change the WF structure. Figure 2, b shows the process of access to the internal structure of WF for the trainee without the possibility of editing. Obviously the output parameters of the given unit, denoted by the symbol $O$, are the input parameters (marked with the symbol $I$) of the dependent unit. For instance, the output parameter $O_{1}^{N_{1}}$ of the unit $N_{1}$ is also the input parameter $I_{1}^{N_{1}}$ of the unit $N_{3}$. In such a mode the step by step interpretation of the operation of WF and the monitoring of the intermediate parameters is possible. In this model, except for functions (3) and (2) the function of the intermediate interpretation of WF (4) on a node are implemented.

$$\text{Debug}_{N_{i}}(WF) : \text{Type}(I_{1}) \times \ldots \times \text{Type}(I_{n}) \to \text{Type}(O_{1}^{N_{1}}) \times \ldots \times \text{Type}(O_{m}^{N_{m}})$$

where $O_{j}^{N_{j}}$, $j = 1..m$ - the output parameter of a node (unit) $N_{j}$.

![Figure 2: Non-Editable Closed (a) and Opened (b) Workflow](image)

This is the same situation when only the (1) function of interpretation, which is independent for each of the informative units of WF, is under consideration. This approach is oriented on training in
the application of individual application-oriented packages, including preparation of input data, start on execution, and the visualization and interpretation of the results of computation (Figure 2, a). The goal of this model is not so much the study of the interdisciplinary phenomena but the addition of competences of the trainee by skills of using application-oriented packages, which are specific to a particular subject area.

3) Model "C": (1)-(3) functions are applied sequentially to the separate units WF – application-oriented packages, from the beginning to the end. In this case the trainee has an opportunity to research the processes occurring in the interdisciplinary system step by step, analyze the output data of each of the packages, and interpret their impact on the results of the subsequent computation. There are additional actions over WF (adding additional units, variable researches, etc.), for obtaining necessary data for research (Figure 3, a). This operation mode is preferable for VLL because it allows the user to reproduce the "tangible" relationship between objects of research and their components.

Variable research on node \( N_3 \) returns as the results of operation parameters not with single values and with sets of the values, the size of which is equal to the quantity of elements of a variation. In fig. 2a three elements of a variation are presented: \( O_1^{N_3} = \left( O_{1,3}^{N_3}, O_{1,2}^{N_3}, O_{1,1}^{N_3} \right) \). This is the reason for changing the description of the \( N_4 \) and \( N_5 \) nodes: in them the variable search is also defined by the results of the \( N_3 \) node execution or is selected as one value from the received sets. The additional node \( AN_1 \) should also select one value from the received set or accept all values and do a variable search afterwards. In the second case, the output parameter \( AO_1^{AN_1} \) is a set of three values.

![Figure 3: Editable Workflow (a), Checking Workflow (b)](image)

Automations of quality assessment of the performed VLL can be created/modified by trained WF or can be implied with the special checking WF (Figure 3, b). The trainee’s results of working with VLL are gathered in archive \( I_1 \). Check parameters \( I_2 \) can be used if necessary. Node \( N_1 \)
decompresses the archive and transfers to a checking node $N_2$ group of files. After the checking, results are transferred to WF ($\phi$) and to a node of creation of a graphical representation of check results.

For the analysis the consistency of models «A» - «C» we need to define measure of competence, developing with corresponding VLL (each type of VLL develop different ranges of learning objectives). The type of grade scales for competences depends on the chosen educational standards or accepted learning theories. For instance, we have used revised Bloom's taxonomy of learning objectives [17] grading to find correspondences between them and learning models.

In our work, a competence is represented by a set of learning outcomes correlated with VLL, that can be grouped into modules (parts of educational course) correlated with learning objectives.

The model “A” helps to answer for the question “What will happen if…” in relation with considered process with use of one package or closed workflow (retrieving, recognizing, inferring, comparing – Table 1). The model “B” includes properties of the model “A” and also allows to analyze the structure of process by preparing and execution related workflow (learning objectives “Apply” and “Analyze”). The model "C" has the greatest range of the of the objectives descriptions, also including checking intermediate results and reorganizing elements in the new structure WF (learning objectives “Evaluate” and “Create”).

<table>
<thead>
<tr>
<th>Learning objectives</th>
<th>Objective Description</th>
<th>Model A</th>
<th>Model B</th>
<th>Model C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remember</td>
<td>Retrieving, recognizing, and recalling relevant knowledge from long-term memory</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Understand</td>
<td>Constructing meaning from oral, written, and graphic messages through interpreting, exemplifying, classifying, summarizing, inferring, comparing, and explaining</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Apply</td>
<td>Carrying out or using a procedure through executing, or implementing</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Analyze</td>
<td>Breaking material into constituent parts, determining how the parts relate to one another and to an overall structure or purpose through differentiating, organizing, and attributing</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Evaluate</td>
<td>Making judgments based on criteria and standards through checking and critiquing</td>
<td></td>
<td></td>
<td>+</td>
</tr>
<tr>
<td>Create</td>
<td>Putting elements together to form a coherent or functional whole; reorganizing elements into a new pattern or structure through generating, planning, or producing</td>
<td></td>
<td></td>
<td>+</td>
</tr>
</tbody>
</table>

**Table 1:** Learning objectives in relation to VLP learning models

Chosen model of the work with WF shows some student’s qualities like responsibility (e.g. stimulation of initiative responsibility by involvement of the trainee in the process of construction or modification of WF), motivation and organizing skills.

The competences are interconnected and can be classified, thus, we can present it by DAG. For each learning course in the competence-oriented education (that is frequently used) a set of developing competences, is defined. Thus, we can present educational process like a WF that consists of courses and represents relations between competences. The course we can also present as WF, where part of the educational WF is the learning objective oriented node module, and they are connected by a
logical structure of the course by the teacher or expert. Each module contains VLL (or a set of them organized in WF and connected by output and input data) – learning outcomes oriented unit.

6 Tool for automation of Virtual Learning Labs design

Implementation of VLP realizing learning models "A" – "C" is rather time-consuming as it includes not only the creation of composite appendices, but also, first of all – deployment in the cloud environment of required application packages. For the CLAVIRE platform this process includes two steps: installation of application packages on the computational resources in a cloud and their registration in a database of the packages, including the description of different scenarios of use as a part of the VLL. For the description of scenarios we used the domain-specific language (DSL) EasyPackage [8] based on the Ruby language. With this approach the same application package may be represented in a "cloud" differently depending on the target audience. A package’s owner has the ability to restrict usage’s options (e.g. for protection from unauthorized users, representation of demo version of the package).

The intellectual editor is developed for the automation of the process of embedding packages in CLAVIRE and the elimination of the need in the manual description of scenarios for their use in the formal language – the CLAVIRE/PackageManager tool. It provides the interface of the designer, which displays the packet description in the form of a set of fields, each of which describes the corresponding attribute. This tool also provides editing of information about the attributes describing the input and output data of the package and requires lexical and syntax parsing of the file with the packet description. At the user's request for the creation of a new package’s description the complete set of fields and attributes for filling is provided to the user and they are automatically assigned access rights to create a package. To debug and verify the working of the added package in CLAVIRE the task-focused interface (TFI) technology is used.

Intelligent editor CLAVIRE/PackageManager is implemented in JavaScript and HTML languages. It can be used as a Web application via a standard Internet browser. For data transfer and requests the JSON data interchange format is used.

7 Case study implementation

Using the considered approach we designed several case studies for the following e-learning systems in Computational Science:

1. E-learning complex “Computer simulation in nanotechnologies”;
2. E-Science center “Sociodynamics”;
3. Case-study “Public Ground Transport Scheduling”

For the purpose of providing the above mentioned educational services to the users the web-portal [18] was implemented, which supports different kinds of learning materials, such as theoretical information, laboratory assignments, and video tutors and allows the students to construct and execute the composite applications with the help of CLAVIRE tools. Note that all workflows’ execution capabilities are fully available through a web interface (workflows – CA can be prepared as a script on the EasyFlow DSL [8] with use of the CLAVIRE/Ginger component - Figure 4).
Each learning system provides the user with abilities to run a separate package (e.g. for finding of optimal route in some city or calculation of the molecular bonds’ energy), and to obtain the skills of solving monodisciplinary tasks. Also, they allow the user to consider the existing CA or build his own CA (e.g. the complex network modelling and information spread simulation and flash mobs predictions) from packages for different scientific areas and obtain the advanced skills of solving interdisciplinary problems.

8 Conclusions and future work

In this paper we proposed an approach for the design and implementation of virtual learning labs for solving interdisciplinary problems with the use of computational resources and application packages which are in the cloud of distributed computational environments. This approach does not require additional efforts to obtain unique computational resources and the installation of special software. The ability of the options’ restriction provides an opportunity for protection from an authorized user (in the case of a legal user who attempts to apply the package not for its intended purpose). Teachers, students, and experts do not have to install anything on theirs personal computers. The web-browser is enough for working with VLL. This paper shows that with the use of VLL, which is implemented in the framework of the mentioned approach, a teacher (expert) may evaluate the level of a student's skills and knowledge.

In the future we are planning to develop a method for the design of educational courses with the use of VLL. The method will be based on the above mentioned abilities of knowledge’ and skills’ evaluation.

9 Acknowledgement

This work was financially supported by the Government of the Russian Federation, Grant 074-U01.

References

Cloud-Computing-to-Prepare-Your-Lectures. [Accessed 20 01 14].


