Informatics in Education, 2005, Vol. 4, No. 2, 215–252

Several Aspects of Technical and Pedagogical Evaluation of Virtual Learning Environments

Eugenijus KURILOVAS

Centre for Information Technologies in Education, Ministry of Education and Science
Suvalkų 1, LT-01306 Vilnius, Lithuania
e-mail: eugenijus.kurilovas@itc.smm.lt

Received: August 2005

Abstract. Currently virtual learning environments (VLEs) and learning objects (LOs) repositories are under active implementation into general education and vocational training system in Lithuania. The article aims to review LOs interoperability standards development tendencies as well as to compare VLEs under existing well-developed pedagogical and technical evaluation frameworks in order to suggest the most suitable one for wider implementation to support active socio-constructivist pedagogies in in-service teacher training and overall in Lithuanian general education and vocational training systems.

Key words: learning objects, e-learning platforms, virtual learning environments, socio-constructivist pedagogies, evaluation, interoperability standards, learning design.

Introduction: VLEs to Support Social Constructivist Pedagogies

Today an increasing interest for so-called “active” and “rich” pedagogies that originate in various socio-constructivist schools of thought exists. There are multiple reasons for this. So called “traditional” pedagogies are very efficient for “knowledge transmission”, but often lead to isolated and superficial knowledge which is difficult to integrate and to apply. In the modern changing world there is an increasing need that students become better general problem solvers and better group workers. Finally there is a pressure to make learning more fun in order to spark both students individual interest. In the heart of a rich, active and open pedagogical scenario are student activities mediated through products. In more simple terms: students have to create something. In analogy to research projects we seek knowledge gain through the medium of artefacts like written text (memos, publications, etc.). The reason that Dewey, Papert, and others have advocated learning from projects rather than from isolated problems is, in part, so that students can face the task of formulating their own problems, guided on the one hand by the general goals they set, and on the other hand by the ’interesting’ phenomena and difficulties they discover through their interaction with the environment (Schneider, 2004). Powerful learning environments that aim at the development of general problem skills, deeper conceptual understanding and more applicable knowledge include according to Merriënboer

and Pass (2003) the following characteristics: "(1) the use of complex, realistic and challenging problems that elicit in learners active and constructive processes of knowledge and skill acquisition; (2) the inclusion of small group, collaborative work and ample opportunities for interaction, communication and co-operation; and (3) the encouragement of learners to set their own goals and provision of guidance for students in taking more responsibility for their own learning activities an processes".

Modern and active pedagogies are more successful if the teacher can profit creatively from information and communication technology (ICT) according to his and his student's needs. ICT have support potential for most of the functions provided by an educational system. Several pedagogical-technical models are currently available and sometimes in competition. Examples are "open-resource-based learning" (using simple web technology), neo-instructionalism (e-learning platforms), collaborative learning (using computer-supported collaborative learning systems or groupware systems), tele-teaching (using increasingly sophisticated conferencing systems).

There are different kinds of ICT tools and systems to support various pedagogies – so-called e-learning platforms, Virtual Learning Environments, Learning Management Systems, Content Management Systems, etc.

Implementation of VLEs in General Education and Vocational Training in Lithuania

VLEs in Lithuania have been used primarily for distance learning in universities, but currently they are being used increasingly as a supplement to traditional classroom based teaching. Educational institutions seek ways to use VLEs to make teaching more effective on campus as well, e.g., in general education and vocational training systems. The Centre for Information Technologies in Education (CITE), the body of the Ministry of Education and Science responsible for implementation of ICT in Education Programmes in Lithuania, currently uses WebCT platform to provide distance learning courses for teachers on Pedagogical part of Teachers' Computer Literacy Standard. ATutor Learning Content Management System was purchased by CITE for basic schools under the framework of Education Development Programme. Several innovative VLEs implementation projects are under implementation in CITE at the moment. They are:

- *Distance learning courses for gifted and special educational needs children.* The aim is to implement 10 collaborative distance learning courses on different subjects for gifted children as well as children with special educational needs. VLE Moodle as a VLE designed to support social constructivist learning model was chosen for implementation of these courses.
- *CALIBRATE* (*Calibrating e-Learning in Schools*). This project financed by FP6-2004-IST-4 brings together 17 partners to carry out a multi-level project designed to support the collaborative use of VLE and exchange of learning objects (LOs) in schools. Here I'll define LO as 'any digital resource that can be reused to support learning' (Wiley, 2000).
- *Programme for ICT implementation in vocational training.* The aim of this in-service teacher training project is to train about 3000 teachers of vocational schools

how to use LOs and VLE during the lessons and for distance learning. 12 collaborative distance learning courses based on purchased LOs and VLE would be prepared for teachers in 2006; and training would be implemented in 2006–2007. The problem of choosing VLE most suitable for implementation of the project is still open.

The aim of this article is to review LOs interoperability standards development tendencies as well as to compare the aforementioned and other probably suitable VLEs under existing well-developed pedagogical and technical evaluation frameworks in order to suggest the most suitable one for wider implementation to support active social constructivist pedagogies in in-service teacher training and overall in Lithuanian general education and vocational training systems.

1. Learning Activities and Learning Design

For pedagogical innovation using e-learning tools to become a reality, both software designers and educators alike need to shift their thinking from a focus on course management to include the design of learning activities themselves (Britain and Liber, 2004).

Significant steps have already begun to be taken in this direction in the development world. The first major contribution was the development of EML (educational modelling language) by Rob Koper and his colleagues at the OUNL in the Netherlands. The work involved in the development of EML has more recently fed into the construction of the IMS Learning Design (LD) Specification in January 2003. The development of EML which began in 1998 was spurred by dissatisfaction with the prevailing model of e-learning to date; that is a heavy focus on content or LOs and an inclination towards information transmission as the overriding yet implicit pedagogical model. The OUNL team, by contrast wanted to provide a language to explicitly model the interactions involved in a given teaching and learning situation so that this could be incorporated into the design of a learning activity. EML was developed as the notational system for modelling 'units of study', which is their abstraction of a learning activity (e.g., a course, a module, a lesson etc).

This is a departure from the prevailing LOs model of e-learning design, which is centred on units of content and metadata rather than units of activity. The major problem with the LOs model is that it fails to provide a coherent framework that can express semantic relationships between the LOs in an educational context. EML is designed to provide a way to type objects according to their pedagogical use, derived from a pedagogical meta-model.

The main contribution of EML to the e-learning community at large is that it has played a core role in the development of the IMS LD specification. The primary aim of the LD specification is to allow teachers or designers to describe a learning design in a standardised way that means it could be 'run' in a variety of learning-design aware players or environments.

Unfortunately there are currently almost no environments that can take an existing learning design and run it; besides there is a paucity of tools available to assist in creating a learning design. However, there are several recent developments that are worthy of mentioning here.

- The first of these is LAMS (Learning Activity Management System) – http: //www.lamsfoundation.org/. LAMS is a learning design inspired system for the creation and running of learning designs in the form of sequences of learning activities.
- The second development is the RELOAD project – www.reload.ac.uk. This project funded under the JISC X4L Program is engaged in producing tools for the creation, editing and running of both LOs and learning activities that implement the appropriate IMS / SCORM specifications. The project is implementing IMS content packaging, simple sequencing and learning design specifications in a suite of open-source tools including a package editor based on the existing PackageIt, a SCORM player for running SCORM 1.3 content and the Colloquia VLE.
- The third development in the space is CALIBRATE (Calibrating e-Learning in Schools) project financed by FP6-2004-IST-4. CALIBRATE brings together eight Ministries of Education, leading research institutions, validation experts, technology providers and SMEs to carry out a multi-level project designed to support the collaborative use and exchange of LOs / resources in schools. The project will strengthen the integration of ICT research in an Enlarged Europe by initially building on three successful IST projects in order to: further develop and implement an open source brokerage system architecture as the basis for a European Learning Resource Exchange (LRE); develop an open source learning toolbox to support the collaborative use of learning resources accessed via the LRE; carry out research into new ways to improve the semantic interoperability of learning resource descriptions; validate CALIBRATE project results in up to 100 schools using an advanced validation methodology. The CALIBRATE Brokerage System implementation developed in the project will provide a strategic level of integration at an infrastructure level. It will act as a catalyst for the development of learning content repositories in all countries but particularly in some new member states including Lithuania where the development of national repositories is still at an early stage. Participating in CALIBRATE will allow these countries to leverage the expertise and knowledge gained in a number of IST projects and in old and new member states that have already made significant progress towards implementing large educational content repositories at a national level. Work in the project on semantic interoperability and the development of an open source, learning toolbox will also make a significant contribution towards consolidating the expertise of leading researchers from old and new member states who are active in the fields of collaborative learning, curriculum mapping, Topic Maps, and IMS Learning Design.

These tools currently still in development point a new direction away from the primacy of content management in VLEs towards systems that make activity-centred learning design using interoperability standards a reality. There are still a variety of difficult

technical issues to overcome in creating learning designs in one system and running them in another. However, the fact that this work is underway and is feeding into the further development of the interoperability specifications is a positive step.

Further Development of Interoperability Specifications and Standards

There has been the vast amount of work globally that has gone into developing an infrastructure for interoperability through the production and refinement of learning technology standards specifications, notably by the IMS Global Learning Consortium, although a number of other bodies are also involved. This work is seen by many in the field as essential to the future of e-learning by aiming to ensure that software systems can work with each other, that learning content and student information can be transported and reused and repositories can be searched using standard metadata.

According to Bill Olivier of CETIS, there are two main motivations for the development of interoperability standards. The first is portability of content and data exchange. If we buy or develop a piece of content we want it to run on a variety of different platforms. The second is to lower the cost of systems integration by allowing existing systems to work with new e-learning systems. Without standards we are dependent on bespoke integration which is time consuming and expensive or lock-in to proprietary systems.

Technology should adapt to fit teaching not the other way round. So by implementing interoperability standards, it may be possible to put together different e-learning tools for different contexts within the same institution, the same department or even the same module. This would allow different subject departments, for example, to choose different e-learning tools that suit the particular teaching approach, instead of having to use the institution's chosen VLE. The aim is to put control of how teaching happens back into the hands of the educator not the software designer.

Importantly, work in defining agreed technical standards for data interchange means that small to medium content providers and software developers can compete effectively with the big players and lock–in to proprietary systems can be avoided, where once one major system has been adopted, it becomes impossible to migrate to another without major costs being incurred to transfer (or re-enter) data that has been saved in a proprietary format. Interoperability allows systems from different vendors to work together (interoperate), and as a consequence, has paved the way for the development of open frameworks for e-learning. The idea of these is that instead of having to buy into a single monolithic VLE application institutions will be able to put together a variety of lower-level components which will provide more flexibility and better suit their e-learning needs.

2. Framework for the Pedagogical Evaluation of VLEs

The aim of the Framework for the Pedagogical Evaluation of VLEs developed by Sandy Britain and Oleg Liber in 1999 and revised in 2004, is to help readers to analyse e-learning tools without being distracted by the details of user interface objects and components. The Framework provides a means by which reflection on aspects of pedagogical process can

be structured, and then how e-learning systems encourage or discourage these can be evaluated.

The Framework does not seek to provide a "true" evaluation of e-learning tools and systems; there inevitably remains an interpretive aspect. The Framework does, however, provide a means by which discussion about specific process aspects of tools and systems can take place in a structured way, and hopefully result in better choices and design decisions.

Education providers using VLEs and other ICT tools for e-learning have two primary aims:

- to enhance the quality of teaching and learning by allowing teachers to use pedagogies that are not possible with large numbers in a face to face environment;
- to manage the delivery and administration of programmes of learning through an electronic (on-line) medium. This includes management of groups of students.

The first of these is a more difficult problem, but (perhaps as a consequence of this) much more effort so far has been put into the second.

The Framework suggests that these two aims are intrinsically linked and that *how* a particular VLE is designed and constructed for the purposes of management can have a profound impact on how likely it is to constrain or facilitate the use of alternative pedagogies.

The Framework as formulated in the Britain and Liber (1999) report was constructed out of two different theoretical models: The **Conversational Framework** (Laurillard, 1993) which is a well-known model of effective teaching practice for academic learning and the **Viable System Model (VSM)** (e.g., Beer, 1985) which is a model for the design and diagnosis of effective organisational structures drawn from management cybernetics. These two models are complementary, with the first providing a model for incorporating effective teaching and learning practice into an e-learning environment and the second providing a number of criteria from an organizational perspective which influence whether the system will facilitate or inhibit the ease with which a pedagogic model such as the conversational model can be used within that system.

A concrete example will clarify this point. If a VLE is constructed on the basis that all learning activities have to be created and sequenced in advance of a course beginning, then there is no way that a teacher can create and add a learning activity to the course on the basis of a preceding conversation at the level of concepts with the students (a basic idea of the conversational model). Alternatively if all the students in one group are treated as belonging a single 'class' object to which learning activities are assigned in the VLE, then there is no possibility for creating individual learning activities.

2.1. *The Viable System Model*

The key premise on which the Viable System Model (VSM) is based is that it is *VARIETY*[1] that threatens to overwhelm organisations and it is this variety that needs to be

[1]Variety is a measure of complexity used in cybernetics and comes from the Law of Requisite Variety Ashby (1956). Simply put, the law of requisite variety states that the variety of a controller must match the variety of the system to be controlled.

managed by any organisation. This is particularly relevant to precisely those issues that education is facing today and that e-learning, it is hoped, will help alleviate. That is to say: how can the quality of educational provision be maintained in the face of increasing student numbers and increasing diversity. The cybernetic answer to this problem is that the variety of educational provision must be increased to match the increased variety presented by the environment. The VSM proposes a number of communication channels by which the variety of the controller might be increased. These channels were elaborated in the (Britain and Liber, 1999) report to yield criteria for the evaluation of VLEs in the effective management of large numbers of students so that resource intensive pedagogic models such as the conversational model could be used in an e-learning context. These criteria are:

- resource negotiation,
- adaptation,
- self organisation,
- monitoring,
- individualisation.

The diagram (Fig. 1) illustrates the relationship between a management system and a controlled system using these channels.

Recursive Properties of the VSM

Course management can be viewed from a number of perspectives or levels:

- programme level,
- module level,
- individual level.

As a tool of analysis the VSM is ideally equipped to accommodate changes in level of focus, as it is a basic mathematical feature of the model that the same general entities and relationships remain valid at any level of analysis. Whether it is a single organism or a

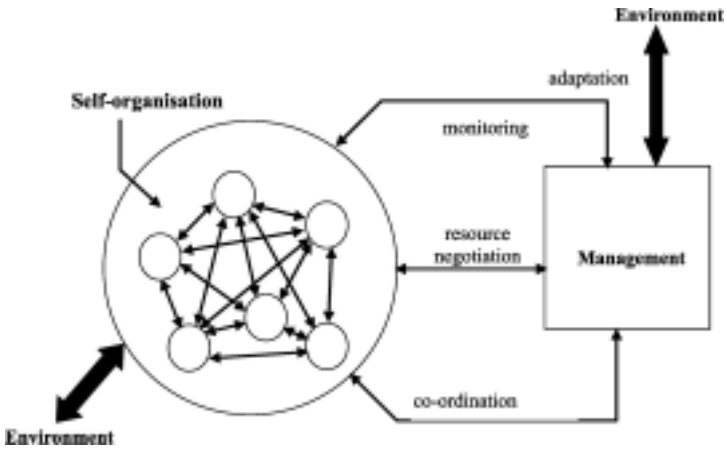

Fig. 1. The relationship between a management system and a controlled system.

global corporation or an entire national economy, the same general principles of viability are the same, i.e., that the variety of the controller must match the variety of the system to be controlled. And at any level of analysis (so the model states) the same sub-systems and channels of control and coordination are involved in the process of managing variety. The model is inherently recursive.

Using the VSM

These three recursive levels – the course or module level, the programme level, and the learner level – correspond to the themes:

- supporting pedagogical innovation using e-Learning (module level),
- supporting institutional management of programmes (programme level),
- supporting students management of their own learning (learner level).

Technology brings new opportunities for managing complexity, where there were previously none. The choices we make affect both the pedagogy and the flexibility available to learners.

2.2. *The Conversational Framework*

There are a number of key characteristics of the conversational model as applied to academic learning, drawn from Laurillard (1993). This model emphasises the importance of dialogue in effective academic learning. Laurillard argues that the nature of much academic learning is largely defined by the acquisition of complex concepts and the creation of conceptual distinctions. This cannot be achieved by a pedagogy based around the one-way presentation of content; there needs to be a two-way dialogue between teacher and student at the level of conceptions. According to the conversational model this dialogue is central to academic learning. The process of learning is supported by the creation of interactive 'micro-worlds' (learning activities) in which the student can actively engage in practice that enhances and reinforces the ideas that have been formulated through discussion. The model emphasises that these activities should be created and adapted on the basis of the conceptual dialogue, rather than pre-set in advance. The final key aspect of the model is that opportunity for reflection is provided as part of the teaching and learning process.

The primary workflow actions that take place through the interactive medium are as follows:

1. Teacher presents/redescribes conception.
2. Student presents/redescribes conception.
3. Teacher sets up micro-world activities.
4. Student interacts with micro-world activities.
5. The system provides feedback on the action.
6. Student modifies actions in light of feedback.

The conversational model as shown in the diagram of Fig. 2 depicts the workflow between tutor and student during learning. Certain activities (in centre) are interactive and take place through some medium. Other activities (in right and left) are internal to

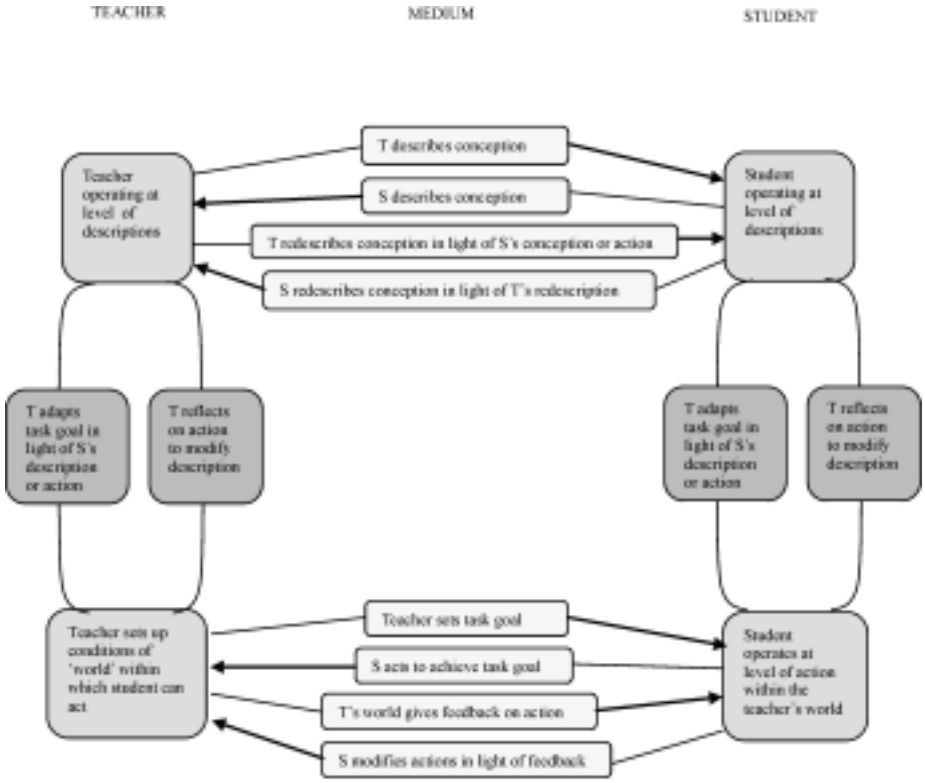

Fig. 2. The conversational framework (adapted from (Laurillard, 1993)).

either the student or the teacher. If we suppose that the medium involved is a VLE, then this model provides a clear set of requirements for evaluating the system's suitability for supporting the processes that form the basis of interactive learning.

The conversational model and VLEs

This relatively simple framework offers much potential as a methodology for evaluating VLEs. In order to make this scheme work, all interactions must be assumed to take place via the medium of the VLE. It is then possible to make judgements about how well the VLE handles each of the individual interactions labelled in the model. The sort of questions one might ask are the following: How does the learning environment allow a teacher to present a conception? What does it mean for a teacher to set up a micro-world within a VLE? How can the student interact with the micro-world?

The model raises further questions about the mechanisms that support conversations. How easy is it to track conversations relating to a particular issue? Can conversations be enhanced by presentation of additional resources? Yet other questions relate to the overall flexibility of the system. How adaptable are micro-world structures once they are in use? How easy is to tailor them to individual students needs?

We might also want to ask whether the learning environment provides any extra tools to support reflection by the student on the relationship between the conversation at the level of descriptions and the activities they have completed or whether it provides structuring to assist the student in modifying their actions.

Evaluation criteria for VLEs from the Conversational Model

In order to evaluate a VLE using the conversational framework we need to establish what tools are provided within the software to allow dialogue and action to mutually influence each other to allow modification of both conceptions and actions on the part of the student as described above.

Additionally we need to be able to say what level of structuring is provided for each interaction. For example if e-mail is used as the tool for communication between tutor and student in a given VLE we need to know how the use of e-mail is embedded within the context of the dialogue about a specific topic or set of learning goals.

Another issue that quickly becomes apparent is that the notion of a 'micro-world' takes on a different meaning in the case of VLEs than more traditional forms of courseware. In essence, the VLE provides the tools for a teacher to build a micro-world by allowing the teacher to construct learning activities enriched by multimedia resources and simulation programmes.

The crucial point from the perspective of the conversational framework is that the teacher should be able to construct the learning activity following a conversation with the student at the level of conceptions and the identification of a learning goal for the topic in question. Thus for any given learning environment we should consider how well the tools provide for both structuring conversations and actions and also how well they allow for integrating dialogue with actions.

- **Discursive Tools**
  All of modern VLEs contain tools for conducting conversations. Naturally these rely to a great extent on e-mail and asynchronous discussion groups. It is important to consider how well the VLE leverages e-mail technology to support the conversation as an integral part of learning. For example, is the conversation accessible directly from the learning topic within the course structure or does the user has to move out of the course work in order to continue the conversation? Does the e-mail or conferencing tool allow attachments to be included with messages? And if so, can the attachment be extracted and embedded into the users' course work structure? Does the tool allow learning goals to be specified and recorded on the basis of the conversation? Ideally the agreed learning goal should be in a prominent location with respect to the topic of learning.
- **Adaptability**
  How easy is it to adapt the activities associated with a learning topic according to the needs of an individual student or student sub-group as revealed by the conversation? This raises the question of whether students in the same group can be easily differentiated within the VLE once a course or learning activity is in progress.

- **Interactivity**
  A basic pre-requisite for a VLE is that it should be interactive. It is not enough for material to be presented to a student and then be tested on it. A VLE should allow the students to restructure the presented material, add resources of their own, annotate material, launch and run simulations etc. In other words the student should not merely be a passive observer of the 'micro-world' constructed by the teacher, but should be pro-active in shaping the 'world'.
- **Reflection**
  How does the VLE allow the teacher to help the student link detailed feedback on their actions to the topic goal? A concept-mapping tool might be a helpful feature in this respect. Alternatively, contextualised discourse for every level of the topic structure should be possible.

2.3. *The Revised Pedagogical Framework*

Let us take two models, the VSM and the Conversational Framework, and tease out a number of key questions that these suggest need to be answered when evaluating VLEs.

Part 1: The Module

1. What tools does the system provide for teachers to present/express their ideas to students?
2. What tools does the system provide for students to articulate their ideas to teachers and other students?
3. Can teachers and learners extend/change their presentations during the module's time period?
4. A VLE is not a single tool; it is a structuring and coordination system containing a variety of tools. These questions are about the model of teaching and learning interactions that forms the basis of the system.
   - Can a module be structured sequentially and / or hierarchically over time?
   - What facilities are there to organise learners in a variety of ways in the module (whole group, small groups, or individuals)?
   - What types of learning activity are supported by the system?
   - What underlying pedagogical model(s) or approach(es) does the system encourage?
5. How are the 'rules of the module' expressed and made evident to the student? By this we mean such things as the learning outcomes, the obligations of the learner and the mutual commitment teacher and student make (e.g., the amount of time a teacher will spend messaging each week, the number of assignments a learner will be expected to complete, etc.)
6. What facilities are there to monitor how well learning is progressing on the module?
7. What can learners do on their own, outside of the purview of the teacher?
   - Can they find and manage resources – do they have their own file stores or repositories?

- Can they talk to other students (other than in the main module discussion), create their own discussions, create their own learning activities involving peers?
- Can they locate people with similar interests outside of their own module, course, year or institution? I.e., is the information about people available?

8. To what extent is it possible for the teacher to adapt the module structure once teaching is underway?
   - Can you add / change / delete resources?
   - Can you add / change / delete fragments of module structure?
   - Can you add / remove people? Can you split them into different groups?
   - Can you create and assign resources or learning activities to individuals?

Part 2: The Student Level

1. How is your system student-centred?
2. Does the system provide time management / planning / organisation tools for the individual student to organise their work?
3. Can the student get a view of current and completed modules? Can they drill down into their completed modules and view a record of work they completed?
4. Can a student monitor his or her own activity? For example can they obtain statistics on what they are spending most time on, whether their time is being evenly shared or not? Can students provide feedback on the quality of the module?
5. How are you thinking about Personal Development Planning (PDP)? Can a student do PDP within your system? Are there module-maps they can use to choose modules from? Can they show register interest in / actually sign up to new modules within the system?

Part 3: The Programme Level

1. Can you see the whole degree or programme within the system as a coherent entity rather than just a collection of unrelated modules? In other words can you obtain a view at programme level? If so what can you do at this level?
2. Does the system allow specification of programme rules for delivering a module? Does it permit or provide a space for negotiation between programme managers and module tutors on resource questions?
3. Can the performance of a module be monitored by the programme manager? What facilities are provided for this? Can guests be signed up to allow QA examination or peer observation of module activities?
4. Can the programme be adapted from within the system? Does the system provide tools for new modules to go through design, development and validation and then be added to a programme?
5. How does the system support teachers working on different modules to coordinate their activities and assist each other?

2.4. *Pedagogical Evaluation of VLEs against the Criteria*

In (Britain and Liber, 2004) the authors has evaluated a number of commercial e-learning platforms (such as WebCT Vista, Blackboard Academic Suite, Granada Learnwise version 3, FirstClass) and more "constructivist" tools and VLEs (LAMS, COSE, Moodle).

I'll comment more properly evaluation results on module and student levels of the most typical proprietary (WebCT) and Open Source (Moodle) VLEs. Currently these levels are more relevant for Lithuanian educational institutions.

WebCT – Vista
Website: `www.webct.com`

Module

1: Presentation and re-presentation of key concepts and ideas
Tutor presentation tools: The tutor can create a sequential activity structure (known as a learning module) based around a table of contents. The activity sequence can include a whole range of tools: simple content pages, discussion tools, assessment tools, media library objects, SCORM objects, glossaries, syllabus, whiteboards etc. There is an HTML editor for creating content. You can also link out to remote objects such as a reading list on a library system.
Student presentation tools: From a student point of view any of the tools can be made available to a group or individual through the permissions model. Group assignments can be published, either by the instructor or student.
Adaptation: Any of the authored content or presentations can be readily adapted on the fly.

2: Coordination of people, resources and activities
Hierarchical and sequential structuring: you can have both views. A learning module is a structured rule-driven sequence of content and activities. But it can also be just an unordered collection. You can create any combination of hierarchical and sequential structuring.
Creating student groups: it is possible to create any grouping of students and to change or make new groups at any point. So you could spontaneously create an assignment workgroup for example and assign different materials to different subgroups.
Pedagogical orientation: transmission model basically.

3: Resource negotiation and agreement
It can be up to the academic what description they put alongside an object; you can display the inherent rules with some objects – when it is going to be displayed, to whom it is available and when. That also works from an academic and instructional point of view. You could for example have a personal discussion forum between tutor and student with the thread resulting in the delivery of a piece of content to a student or it could be the student that decides what content to select following the discussion – so there is support for that process.

4: Monitoring of learning
Tracking facilities. You can get information about what tools have been used and when, how many discussions, what groups of contents, how many emails, what grades did students get at different times, right down to the individual student sessions, tracking both information and activity.

5: Self organization amongst learners
Students have their own file-space and can search for and manage their own resources.
Students can't set up their own learning groups but they can self-subscribe to ones that have been created by the academic.
Vista has 'people links' so anywhere you see someone's name you can send them an email. You can find out who else is online at the time and who is in the same module.

6: Adaptability of module and system
Templates can be defined at any level from institution level to the course level and e-learning modules can be derived from these highly flexible templates. Changes can be made to content within these templates, which cascades to the courses where they are used. Content design and course layout however is not restricted to being developed within templates; work can be carried out at the course level as well. If desired content can be created within content repositories and referred to from within courses so enabling content re-use and collaborative development.

Learner

1: Learner-centredness
There are some trends in student centredness, e.g., a move from a transmission model to an activity-based model. Students can be involved in the decision about what they study, self-enrolment, formation of study groups – those tools and functionality are available. Some of this falls between the student information system and the VLE though.

2: Coordination of people, resources and activities
Yes, students can view their current and completed modules and view a record of their work.

3: Time management/planning
Students have the ability to define their own calendar, which can also be integrated with a campus-wide calendar. They can see how far through the content they are compared to the group as a whole. They can see the timeline they are within with reference to a particular subject. Students can see who has replied to their posts. They have their own file area, so they can have a draft portfolio for their own work. They can drag and drop into their file area from their desktop with webdav.

4: Monitoring own learning
Students can see their own grades. They can give private feedback to instructors. You can have anonymous surveys. It's planning to introduce anonymous marking in the next version.

5: Adaptation/reflection
Students can register their interest in a learning module based on an outline of what the module contains, and then it becomes fleshed out when they sign up. You can have a syllabus embedded within a LO if you want. It is largely dependent on the organisational model you want to implement. WebCT Vista incorporates powerful API's that enable both commercial and open source portfolio applications to integrate with resources and student information within the VLE.

Commentary
Positioned as an Academic Enterprise System, the main benefits of Vista over and above the Campus product are at the level of the support for institutional and cross-institutional level management of degree programmes and support for student management of their own learning.

Overall system lacks support for pedagogical flexibility and innovation. The improvements in Vista over 1st generation VLE products are:

- A new roles and permissions model means that any of the tools that can be used by tutors can also be made available to students.
- Student groups and sub-groups can be created or adapted on the fly.
- Content or presentations can be created or adapted on the fly.
- A learning module can be any combination of hierarchical or sequential structuring.
- Optional use of templates facilitates the process of course design and updating.
- Students can self-subscribe to groups.

Monitoring is still based on tracking facilities of debatable information value. Student self-organisation is limited in that they cannot set up their own groups, although they can self-subscribe to those created by lecturers. Also profiling information is limited because of privacy issues. Although it is collected within the system, it is not made available.

MOODLE (Modular Object-Oriented Dynamic Learning Environment)
Website: `www.moodle.org`

Description:
Moodle is an open-source VLE that is very similar in many respects to the course management components of the major commercial VLEs. The main difference is that philosophy behind Moodle has its roots in Social Constructionism, so there are both specific tools to support constructivist learning and other common features that are constructed in such a way as to correct limitations the creator perceived in the main commercial systems. Being open-source means that it can be downloaded and used by individual teachers or departments without necessarily requiring an institutional decision to purchase it. It also means that it is open and extensible by its developer community.

Module
1: Presentation and re-presentation of key concepts and ideas

Creating a course: Moodle offers three different formats for a course template set for the site: weekly, topic or social-oriented formats. Content can then be uploaded to the course and activities added in the order that students will use them.
Tutor presentation tools: Resources (content), forums, journals, quizzes, assignments, surveys, choices, chat, workshops, user profile, etc. The resource module primarily contains HTML pages and other documents, but you could add a resource that is an external program, there is an API to allow this. Forums are at the core of Moodle. Since Moodle uses a rich text editor - an html editor with wysyig display, the forums get used for all sorts of things. You can rate the posts in forums according to different scales (e.g., rating posts on the extent to which they demonstrate separate or connected knowing) and that is an educational feature. There is an activity module and a journal module – the journal is structured, not just a simple text box. There are surveys but you have to use the validated instruments that are included, you can't construct your own yet. Workshops are peer-assessed assignments.
Student presentation tools: All of the above apart from resources (currently these are teacher only). It's planning to change that to allow students to build up portfolios and the authors would like a file space for them to take their materials from one institution to another. Versioning and access control are introduced. Forums are by far the most used tool, and allow arbitrary attachments, HTML formatting, etc.
Adaptation: Absolutely. Anything can be adapted. But there is a 30-minute time-period for reflection that allows changes to forum posts after that period they become fixed.

2: Coordination of people, resources and activities
Sequential/hierarchical structuring: The system encourages you to lay out your module in a sequential order. A module outline is required.
Organising groups of learners: Whole group, separate subgroups, visible subgroups, and individuals.
Types of learning activity: Various: Connected discussions with optional peer evaluation, Reflective journals, Reading, Glossary/Encyclopaedia writing (students can build up a glossary and any of those entries automatically link from any text throughout the system), Chatting, Peer-evaluated assignments, Quizzing.

3: Resource negotiation and agreement
This is left to the teacher to express, using the tools provided. Each unit within the course can have an advance organiser to help focus activities. The teacher can set an introductory activity, e.g., a forum at the beginning and a journal entry for private thoughts about the course, in order to establish the direction for the course.

4: Monitoring of learning
Each student has an activity report. This is a complete report with all the activities in the same format as presented to the student – you can see what they've read, what they've posted, all in context. There is also a grade-book for the graded activities.

5: Self organization amongst learners
Student file upload: Yes, via forums and glossaries. Shortly a less structured storage system for students will be implemented.

Communication with others creation of discussions and learning activities: Yes, within existing activities. E.g., if the teacher creates an "Open forum" then anyone can do anything in there they like. So students can't set up their own forums but can make a new thread within an existing forum. Teachers can go anywhere at the moment.
Locating people: Very much so. There is currently no information on people outside the course. Moodle.org provides a means for development of learning communities – it is used for tutors at the moment but could be used for students.

6: Adaptability of module and system
This is a prominent feature of Moodle. You can adapt the resources, the course structure, and the people groups. Resources and learning activities can be assigned to groups, but not individuals – unless you create a group of one.

Learner

1: Learner-centredness
Every action is tagged with person's photo and name. This makes the person very prominent, promoting understanding of each individual. This helps teachers to remind themselves who the student is when in communication with them. All a student's enrolled courses are shown on the front page (once they log in).
2: Coordination of people, resources and activities
The activity report (which students can be allowed to see) is very comprehensive, as is the grade-book. These show what is happening in a particular module. There is no overview of modules.
3: Time management / planning
Courses are laid out in an intuitive display to indicate the passage and times of all the activities in the course. To enhance this, the calendar will contain all relevant dates and allow the student to add his own one. The authors are trying to get useful information in the calendar rather than just a display with the date on. Forums give you a sequential flow of activity – you can use your email box as an interface to go back through time with html links to bits of the website.
4: Monitoring own learning
Yes, students can view a record of their own activity using the activity report and they can provide feedback on the quality of a course or sub-unit.
5: Adaptation / reflection
PDP tools: The support for the bigger picture outside individual courses is planning to add.

Commentary
Moodle is primarily a course management VLE in the same tradition as the commercial VLEs with a number of distinguishing features. It is open-source and free. It has a community of developers and educational practitioners centred on the website moodle.org. This means it is a customisable and affordable VLE product especially suited to small colleges, schools and individual lecturers or departments who want to set up e-learning

courses but don't have a budget for license fees. It is designed from a constructivist perspective and has a number of specifically constructivist tools, e.g., reflective journals, peer group assessments, the ability to rate posts on a dimension which represents the level of connected vs. separate knowing demonstrated by the poster. It is designed to be modular and developers are encouraged and supported in developing new activity modules to complement the ones provided as standard with the system. Information about people is important in Moodle and a users profile including their photo appears where ever their name appears.

Moodle at the module level is similarly flexible and adaptable and it does contain a variety of tools and features designed to encourage thoughtful pedagogical practice in an environment which is easy to navigate, use and administer.

There are several systems that have been designed to support an innovative pedagogical approach using technology. COSE, LAMS, and Moodle were evaluated under the Framework by the authors. Each of these systems is quite different. Although COSE and Moodle encourage a constructivist approach to reaching and learning, they encapsulate quite different models. Moodle has some specific constructivist tools but is very similar in many ways to the pattern of use encouraged by the commercial VLEs. COSE embodies a strongly constructivist model and the pattern of use it encourages is quite different from the other tools but is not as familiar and therefore easy to use for general purpose course management as some of the other systems. LAMS takes a radically different approach again to any other system.

Systems such as Moodle, COSE and LAMS are not intended to be institutional level systems and are more directed towards specific teaching and learning situations, thus these systems and others like them occupy a different space within an institutional e-learning environment. The ability for individual lecturers or departments to use tools like these is important for the diversity of teaching and learning approaches using e-learning, but does not sit well with the desire of many institutions to have a single VLE for the entire institution. Whilst the implementation of enterprise standards again should make it possible for multiple VLEs to be used across an institution, it is generally not possible to map all the data from one system to another as a COSE object for example may well have data fields that simply do match up to any WebCT fields.

3. Technical Evaluation of VLEs

Suggested framework for technical evaluation of VLEs is based on the Methodology of Technical Evaluation of Learning Management Systems (LMSs) presented in the document 'Technical Evaluation of selected Learning Management Systems' (Review). The document was published in May 2004, and it is a part of the *Evaluation of Learning Management Software* undertaken as part of the *New Zealand Open Source Virtual Learning Environment* project. One of the goals of the project was to select a best-of-breed Open Source LMS for development and large-scale deployment. The criteria for the Review expand on a subset of the criteria, focusing on the technical aspects of the systems.

I'll properly comment and renew the evaluation results of 3 LMSs surveyed in the Review:

ATUTOR *Platform*: Apache, PHP, MySQL. *Version*: 1.3.3

> ATutor is a promising system that provides good documentation, ease of installation, and strong potential for development. While the user interface may not seem intuitive to many, the overall functionality is good (and wide open/modular) and the development team is committed to standards. The system is also install-friendly and receptive to new language versions.
>
> ATutor is one of very few LMS' that support LO repositories. ATutor is very strong on standards and can import external content in IMS / SCORM format.
>
> ATutor scores highly for openness. It has many features and rates highly for usability, including accessibility for learners with disabilities.

ILIAS *Platform*: Apache, PHP, MySQL. *Version*: 3 Beta

> Ilias is an object-oriented course management system. Its user interface revolves around a desktop metaphor. Widely used in Europe and China, it has good internationalisation features. Version 3, under development, incorporates support for IMS/SCORM packages.

MOODLE *Platform*: Apache, PHP, MySQL/PostgreSQL. *Version*: 1.2 Beta

> Moodle is one of the most user-friendly and flexible open source courseware products available. It has excellent documentation, strong support for security and administration, incorporates SCORM package and is evolving towards IMS LD standard. The key to Moodle is that it is developed with both pedagogy and technology in mind. Moodle has a robust development and user community. Great with languages although some development may be needed for robust handling of MathML and enhanced tracking features. *Still, this program receives a high recommendation.* Moodle is a student-centred course management system designed to help educators who want to create quality online courses. The software is used all over the world by universities, schools, companies and independent teachers.

3.1. *The Goals of the Review and Major Evaluation Criteria*

The objectives of the Review were:

- to define detailed technical criteria for the evaluation of Open Source LMSs;
- to gain understanding of the design, architecture and implementation details of the short-listed LMSs, with a view towards long-term development and maintenance;
- to evaluate the short-listed LMSs against set criteria;
- to engage members of the Open Source community in the process where relevant.

Major evaluation criteria

Working with the *Evaluation of Open Source Technologies* paper, and with a view towards long-term development and maintenance, the overall criteria for the Review were:

- overall architecture and implementation,
- interoperability,

- cost of ownership,
- strength of the development community,
- licensing,
- internationalisation, localisations,
- accessibility,
- document transformation.

3.2. *Methodology*

The following approach was used:

Develop Technical Evaluation Criteria

Based on the Initial Evaluation of Open Source Technologies paper provided by Open Polytechnic of New Zealand, the criteria focused and expanded on the technical aspects of the systems. For each criteria rated, a discussion was provided, indicating what was covered, its importance, and a check-list of the aspects observed. Each of the overall criteria was presented with a brief discussion, with summary tables providing a weighted rating.

Deploy and Evaluate

The short-listed systems were deployed in a test-bed environment, where they were used and maintained for the duration of the Review. The deployment stage involved set-up, configuration, branding and minor customisation, thereby providing practical experience with each tool. As much as possible, the evaluation criteria were checked against the actual performance of the system, rather than the published feature list.

Given the goals of understanding the challenges of long-term maintenance and development, every chance to study the feasibility of changes and extensions was used.

3.3. *Criteria: Overall Architecture and Implementation*

To assess software systems for long-term operation, development and support, the key criteria are simple: quality of architecture and craftsmanship. This is comparable to considering other complex man-made objects for their long-term viability.

The modern definition of software architecture defines it as the structure which comprises software elements, their properties, and relationships.

3.3.1. *Scalability of the System*

Scalability indicates a system's ability to maintain quality performance or service under an increased system load by adding resources.

To scale vertically or *scale up*, means to add resources to a single node in a system, such as adding memory or a faster hard-drive to a computer. To scale horizontally or *scale out*, means to add more nodes to a system, such as adding a new computer to a clustered software application. A truly scalable system must also *scale down* to more constrained resources, to allow small scale deployments.

The three LMSs in the Review are strikingly similar in their scalability profile. While most web applications have a common architectural model, the LMSs evaluated use the

Table 1
System is scalable

System is scalable	ATutor	Ilias	Moodle	Comment
Scale up: high end services	Excellent	Excellent	Excellent	Similar architecture
Scale up: allow cashing	Good	None	Incompleted	Hard to achieve with PHP-based solutions
Scale out: multiple app servers	Good	Good	Good	Better performance and availability. Similar limitations – session handling and data directory
Scale up: multiple database servers	Good	Good	Very good	Better availability and reliability, less performance. All support MySQL with log-shipping. Moodle also supports PostgreSQL with log-shipping and phased commits. None segregates read-write from read-only
Scalability constraints	DB writes	DB writes	DB writes	Similar architecture

same middleware and RDBMS solutions. They embody, however, different strategies database usage, caching, and memory management. These can have a significant impact on scalability and performance (Table 1).

3.3.2. *System is Modular and Extensible*
The Review considered a system to be modular when its implementation is comprised of modules with high cohesion and low coupling. This is a strong indicator of the inherent maintainability and adaptability of a software system.

In short, coupling is a measure of how interrelated two software components are. Cohesion is a measure of how related the functions performed by a software component are. High coupling implies that, when a component changes, changes to other components are likely. Low cohesion implies difficulty in isolating the causes of errors or places to adapt to meet new requirements (Table 2).

Modular architecture
Does the application have a modular architecture? An ideal architecture has simple but well defined internal layout, and the modules have small, well defined interfaces, high cohesion and loose coupling.

The modular architecture must be documented, simple and elegant. There is no requirement that is must be object-oriented or functional. The performance/resource trade-offs must be acceptable, and it should not impose artificial bottlenecks or inefficiencies.

ATutor: Monolithic architecture. All functionality resides in the core of the application. Extensions must be made part of the application, and are tightly coupled.

Table 2

System is modular and extensible

System is modular and extensible	ATutor	Ilias	Moodle	Comment
Has module architecture	No	Yes, suffers tight coupling	Yes, simple and complete	Moodle achieves the goal of loose coupling/high cohesion. Ilias has a tightly coupled architecture
Core functionality in modules	No	Yes, inconsistent	Yes	
Solid support for modules	No	No	Yes	
Internal API	No	Ad-hoc	Yes, well documented	
External API	No	No	No	Possible to develop external APIs

Ilias: Architecture is confusing and undocumented. Each module is implemented as a class, exposes an arbitrary interface, and behaves as a special case.

Moodle: Simple, straightforward and complete module architecture. The code is split in modules (*mod* directory) and libraries (*lib* directory). Modules must expose certain entry points as functions and particular files. Authentication is modular and segregated from the rest of the modules.

Core functionality in modules

Highly modular applications provide most of their end-user services as modules.

Experience indicates that systems that do not follow this rule hint at a module infrastructure without extensive use, therefore incomplete and immature.

ATutor: There are no modules as such. All functionality resides in the core application.

Ilias: Most functionality resides in ad-hoc modules without interface consistency.

Moodle: Most of the end-user functionality is in modules. While roughly 50% of the code is in core libraries, 30% is in modules.

Solid support for modules

Solid support for modules means that the core application sets up a framework of services. Modules must have means to register with the framework (for user-level navigation, compatibility checking, etc.), set up database tables to store their data, schedule backend processes, retrieve user data, presentation templates, and offer configuration interface (either for the user or the administrator). If the application is being upgraded, modules must be able to execute upgrade routines.

Additionally, for modules to maintain their loose coupling with the core application, the framework must offer other well documented services such as retrieval of configuration data, database access, logging, authentication and authorization.

ATutor: There is no framework in place.

Ilias: There is no framework in place.

Moodle: The framework is well developed and mature. Modules can set-up their own DB tables, run upgrade scripts, cron scripts, etc.

Internal API

Internal mechanisms must be documented and expose functions or methods that act as entry points to trigger its functionality. These must follow documented (or well known) conventions, and be consistent and straightforward.

ATutor: There is no well defined internal API.

Ilias: Internal classes are documented using PHPDoc, but do not constitute an organized API.

Moodle: The modules infrastructure imposes a standard API for modules. Libraries are well documented.

External API

Key services are exposed to suitably authenticated programs that need to be integrated with the LMS but are not on the same server or are not part of the same application. This must be done over an RPC or ORB interface; popular implementations of such interfaces are XML-RPC and SOAP.

None of the systems reviewed exposed an external API. It is possible, however, to build it for any of the three system using existing PHP libraries.

3.3.3. *Multiple Installations on a Single Platform*

Can one hardware set-up – at any point of the scalability range – handle multiple instances of the system independently? To achieve this configuration files (covering hostname, database, installed path and, data directory) and logging services must be abstracted.

Ideally a single installation of the program can be made to run with different configuration files.

ATutor: Can run multiple instances in different URLs. Configuration is restricted to a single file and well abstracted. Logging services are provided by Apache and can be segregated per instance.

Ilias: Can run multiple instances in different URLs. Configuration is spread across several files, but well abstracted, and prepared to run several instances from the same set of configuration files. Logging services are provided by Apache and also internal logging facility; can be segregated per instance.

Moodle: Can run multiple instances in different URLs. Configuration is restricted to a single file and well abstracted. Logging services are provided by Apache and can be segregated per instance.

Scripted administration

It must be possible to automate the creation, administration and deletion of server instances using common Unix scripting tools. Scripted upgrades should be possible.

ATutor: Possible to manage through shell scripts, some parts of the installation/upgrade process may need extra work.

Ilias: Possible to manage through shell scripts, configuration files and database set-up more complex.

Moodle: Simple to manage through shell scripts. Version upgrades can be scripted across many instances.

Table prefixes

The use of configurable table prefixes allows many application instances to share one database using segregated database tables. This benefits allows many instances to be run in environments limited to one database (virtual hosting at an ISP, for instance), and also derive a marginal benefit in the use of persistent database connections.

ATutor: Yes.

Ilias: No.

Moodle: Yes.

3.3.4. *LMS Has Reasonable Performance Optimisations*

Strategies that maximize performance and make efficient use of resources are clear signs of good architecture and craft. Furthermore, application systems must be deployed at reasonable cost for large user bases. Strategies and practices relevant to the PHP engine can and should be used to ensure the system delivers acceptable performance and scales well (Table 3).

3.3.5. *Look and Feel is Configurable*

See Table 4.

3.3.6. *Security*

The security of a web-based system consists of strategies that address three interdependent but well defined areas (Table 5).

Server security covers strategies to secure the server against attack. Applications must be designed to avoid being an avenue of attack against the server.

User security and privacy must be preserved; the LMS system must keep communication and data private and avoid exposure of user's computers (client computers) to attacks.

Table 3

System has reasonable performance optimizations

System has reasonable performance optimizations	ATutor	Ilias	Moodle	Comment
Compatible with precompiled code caches	Yes	Yes	Yes, limited compatibility	Moodle is not compatible with one of the code caches available
Persistent connections	Yes	Yes	Yes	

Table 4
Look and feel is configurable

Look and feel is configurable	ATutor	Ilias	Moodle	Comment
Headers and footers customisable/use themes	Yes, untidy	Yes, limited	Yes, as themes	Ilias and ATutor have an untidy setup for headers/footers, where several files must be edited
CSS and other applied styles are customisable/use themes	Good, as themes	Good	Good, as themes, well documented	
Theme scope	Site-wide, with course and user themes	User theme	Site-wide theme	Centralized theme control is better for wide-scale deployments

Table 5
Security

Security	ATutor	Ilias	Moodle	Comment
User privacy-security: SSL integration	Good	Good	Good	
Server security: permission lock-down	Good	Bad	Good	Ilias requires lax permissions on config files, and cannot be easily managed otherwise
App security: authentication	Good	Good	Good	PHP Session management needs to be configured tightly
App security: authorization	Limited	Good	Limited	
App security: logging and monitoring	Bad	Bad	Bad	None offers logging of abuse attempts
App security: validation of input	Good	Weak	Good	Security issues in input validation were identified during the review

Application security covers restrictions on what users can do (to prevent attacks and misuse) and data privacy. To prevent unintended access, the application must have mechanisms for *authorization* (checking of credentials) and *authentication* (checking of right to perform an action).

In these three areas, secure applications implement strategies that monitoring and log attack attempts and minimise exposure.

Security strategies involve the whole stack, from the operating system up through the libraries, framework, components and application layers. The Apache/PHP framework

Table 6
Modular authentication

Modular authentication	ATutor	Ilias	Moodle	Comment
Authentication is modular	No	Yes, PEAR Auth	Yes, own modules	
Interoperates with standard authentication back-ends	No	Yes	Yes	Ilias offers a wider range of authentication back-ends
Interoperates with external authentication ticket system	Yes	Yes	Yes	All use PHP's session management, which can use an external system
Authentication services API	No	No	No	

provides outstanding security features, which these applications should take advantage of.

3.3.7. *Modular Authentication*
Institutions deploying a LMS are highly likely to have a pre-existing infrastructure, one that is already maintaining authentication data, and maintaining segregated user lists is inefficient and error-prone. It is also very likely that a LMS will be coupled with other tools, web-based or otherwise, such as blogs and content management systems. Such scenarios make modular and flexible authentication a key feature, which was evaluated separately from overall system modularity (Table 6).

3.3.8. *System is Robust and Stable*
Robustness and stability are key concerns when deploying applications in large scale and long term. These criteria are observed not only in the passive count support issues during operation, but also on the proactive strategies taken by the application developers (Table 7).

3.3.9. *Installation, Dependencies and Portability*
Ease of installation and maintenance is critical for a scenario of wide deployment, covering both large-scale centrally managed installations and stand-alone installations in the Local Area Network of a tertiary education organization. Considerations cover portability, robustness, freedom and overall ease of installation (Table 8).

3.4. *Criteria: Interoperability*

Systems are seldom deployed in a vacuum. Deployment scenarios include integration with pre-existing systems and networks talking a myriad of protocols.

Table 7
System is robust and stable

System is robust and stable	ATutor	Ilias	Moodle	Comment
Cross-browser support	Good	Good, requires frames	Good	Ilias does not work correctly with screen readers and text-only browsers
Not dependent on JavaScript	Yes	Yes	Yes	Moodle loses WYSIWYG HTML editor
No maintenance downtime	Good	Good	Good	
Backup	Good	Good	Excellent	Moodle implements an XML based backup/restore facility in addition to standard DB backups

Integration of authentication systems is by far the most likely scenario, which is explored fully under 'Modular Authentication'. Integration efforts other than authentication also benefit from a clean, modular, architecture, discussed under 'System is modular and extendible'.

3.4.1. *Integration is Straightforward*
Ease on integration depends as well on the availability of libraries that implement common protocols. This is actually a feature of the framework, which the three systems share.

The three layers of the stack that comprises the framework have a very good availability of protocol libraries. The PHP library repository (PEAR), plus libraries and modules available outside of PEAR, have a wide spectrum of modules implementing internet-related protocols.

Additionally, there is a small range of Apache modules that implement common protocols, such as LDAP, Kerberos and RADIUS. Finally, there are hardly any modern protocols without an available (and open) implementation on Linux.

Put together, the Linux-Apache-PHP framework provides one of the most comprehensive development platforms available.

3.4.2. *LMS Standards Support (situation in July 2005)*
The system is compliant with common standards for LO creation, retrieval and use, i.e., SCORM 1.2, IMS, DC metadata (Table 9).

LO standards (IMS packaging, SCORM)
The LMSs are rated on their ability to use externally supplied LOs packaged in IMS format (also known as SCORM 1.2), their ability to provide SCORM 1.3 runtime environment support for LOs that are SCORM-aware. Additionally, the ability to export content in IMS packages was evaluated in the Review.

ATutor: Can export course material as an IMS/SCORM 1.3 packages.

Table 8

Installation, dependencies and portability

Installation, dependencies and portability	ATutor	Ilias	Moodle	Comment
Overall portability	Widely portable	Widely portable on Unix, limited on Win32	Widely portable	
Dependencies are robust	Yes	XML dependencies not robust	Yes	
Dependencies can be replaced	Yes	Some	Yes	XML-related dependencies cannot be replaced in Ilias
Dependencies are FOSS	Yes	Required	Yes	Some optional dependencies for Ilias require commercial and non-FOSS licenses
Dependencies are easily installed	Yes	Required	Yes	Some optional dependencies (Java) require separate installation on Debian systems
System is easily packaged	Yes, packages not available	Partially. Packages available	Yes, Debian packages available	

Table 9

LMS standards support

LMS standards support	ATutor	Ilias	Moodle	Comment
LO Standards	IMS/SCORM export	IMS import	SCORM 1.3	Moodle has an IMS LD module in development
Search and retrieval standards	No	No	No	ATutor's future plans mention TILE

Ilias: Can use externally generated IMS package. There is no SCORM 1.3 runtime environment support.

Moodle: There is SCORM 1.3 environment support.

Future plans mention IMS LD. In his presentation in Braga in May 2005 Martin Dougiamas, lead programmer of Moodle, explained that "Moodle has an implicit sequencing of activities, which are listed. The exporting of this structure as an LD file is fairly simple. The file contains the standard tags, and it would be good to describe as much as possible using them. Anything that is not can standard be added using XML

processing instructions, although these are only really useful for Moodle at the moment because they are Moodle specific. For example, in Moodle we have the glossary activity. Not many other systems have anything like that, but the export could be made in a way which would be useful, even without the full functionality. However, the importing is trickier. Some things which can be done in LD are not yet in Moodle, but they have been part of the plans for some time. Moodle 1.5, has just been released, and this the culmination of work carried out since August 2004. A lot of core development has been done, making it possible to target tiny elements and restyle them. 1.6 is the next version, and it seems reasonable that we can get LD export for that, in a few months. 1.6 also includes a database module similar to Filemaker for making collections of information, creating a photogallery with text, comments etc. Blogs are very popular, and there are some interesting ideas for their use in Moodle. If you have blogs in an educational environment then you can have rss streams of a single student, or a group, or a site, or an institution. For each entry in the blog it is possible to target who it is for. There are also plans to improve the user profile, making it more like a homepage. At the moment it is rather a dashboard look. It will be able to access LAMS transparently from Moodle, but they are going to be separate systems. They will be easy to work with together. Version 2 coming up in 2006 will be a big break, with major core changes, and it is planned to include LD input. We will add conditional activities. For every activity there will completion criteria, and a second criterion is that the activity is only available when this other activity is done. You can still be free and easy, but there will be the option of using them. This ties in well with LD. The way that roles work will also be reviewed. We currently have hard coded roles, but the idea is to make them completely configurable. The administrator can make a new role, say ”parent”, and assign a role to a user in an activity. LD import would require this. The current groups’ implementation in Moodle is rather basic: each activity can choose whether to use groups or not. The aim is to improve that to the kind of level that LD is talking about, with people in activity groups”. He has said that LD is the only standard that he has really felt comfortable with. “SCORM adds very little to functionality, but is necessary, for example you can’t run Moodle in Italy unless you are SCORM compliant.

Other improvements which are not LD related is networking between Moodles. It would be great to have a button for community, where the system asks you some questions, and then you get logged in automatically to a Moodle server where you can share ideas, content, etc. The administrator can choose if they want their learners going off to Moodle.org, but by default it’ll be on. It’s also intended to make it possible to have one code base that appears to be many different sites. There will also be a documentation management system, which will enable students to have space to store assignments, where they can link to them. Hive’s Harvest Road is being incorporated, but in the future it would be good to have an OS back end too. Moodle is described as ”social constructionist”, which means that it is a learning system where learners can create things for each other, they can see each other making things for each other, and the system also tells you what is going on in the system. If you are on a website which reminds you about what is going on then activity heats up. In forums there is always a reminder to ask questions. All of that is social constructivism. There is a lot more which could be done to support this and plenty of ideas for things which could be done better”.

Table 10
Cost of ownership

Cost of ownership	ATutor	Ilias	Moodle	Comment
Implementation	Low cost, automated	Mostly automated	Low cost, automated	Ilias required manual configuration of each host. Others are automated
Development and support	Complex. Code lacks structure. It is unclear where should new functionality be added	Unclear, complex	Straightforward, well supported	Moodle architecture is straightforward and documented, and finding and fixing bugs proved to be easy. It's easy to follow the logic, additional modules can be created based on the existing ones
Hardware and licensing	No licensing costs, reasonable HW costs	May have licensing costs, reasonable HW costs	No licensing costs, reasonable HW costs	
Maintenance	Low cost, automated	Mostly automated	Low cost, automated	

3.5. *Criteria: Cost of Ownership*

Architecture, modularity, code quality and community support all factor in the long-term costs. This section summarises partially the findings in those specific areas (Table 10).

3.6. *Criteria: Strength of the Community*

Dynamic development communities are a widely recognized feature of successful open source projects. Such communities are nontrivial to develop, yet they are a key component of a software project health and longevity.

3.6.1. *Installed Base and Longevity*

The number of installations is large, with "quality" organizations using the system.

ATutor: Originally released in December 2002, early versions date back to January 2002. In May 2004 known installations list only reported 18 servers, the actual installed based was likely to be much larger, at least by an order of magnitude.

Ilias: Originally released in July 2001, early versions date back to mid-2000. In May 2004 known installations list reported over 100 servers, most of them at large universities.

Moodle: Originally released in August 2002, early versions date back to November 2001. In 2003 reported to have over 1000 known installations, several of them in major universities.

3.6.2. *Documentation*

The system has high quality documentation and online help. Documentation should be available targeted towards end-users (teacher, student, author, and administrator), systems administrators and programmers.

ATutor: Good end-user documentation.

Ilias: Limited end-user documentation. Some developer documentation.

Moodle: Has good end-user, administrator and developer documentation.

3.6.3. *End-user Community*

Responsiveness of the support community is good. Historical numbers of the user and developer mailing lists/forums are strong and show growth. Spirit in the mailing lists/forums is positive.

Unfortunately, the communities are centred on web-based forums instead of the traditional Usenet or mailing lists. This makes measuring community size by traffic nearly impossible without access to the forum database.

ATutor: Medium-sized user community, helpful and active.

Ilias: Medium-sized user community, helpful and active.

Moodle: Strong user community (over 50.000 users from 120 countries in July 2005), helpful and active.

3.6.4. *Developer Community*

There is an active developers community, helpful and with a good understanding of the application internals. There is one or more maintainers with CVS access or equivalent. There are many contributors credited in the Changelog.

ATutor: There is no developer community. The development of ATutor was closed, and this has prevented a community from forming.

Ilias: There is a small but active developer community.

Moodle: There is an active developer community. Several developers have CVS access, and there is a well populated credits list.

3.6.5. *Open Development Process*

The development process is open, with access to the version control repository, nightly releases, a bug tracker, more than one core developers listed in the Changelog.

ATutor: Closed development process, only offers access to nightly builds, and a bug forum.

Ilias: Very open development process, with access to CVS, bug tracker and other tools.

Moodle: Very open development process, with access to CVS, bug tracker and other tools.

3.6.6. *Commercial Support Community*

It is desirable that the application has a community of commercial companies offering related services and support.

ATutor: Only the original development company is offering ATutor-related services.

Ilias: Has not yet gained critical mass for related commercial offerings.

Moodle: Moodle.com, some ISPs and independent contractors offer Moodle-related services.

3.7. *Criteria: Licensing*

To guarantee the future of the project, and of future investments in the project, the system must be truly Open Source. The license must be OSI-Certified, the terms of usage and redistribution well understood and acceptable. The community stakeholders must understand the terms, and act openly and accordingly.

The three systems considered are covered by the GNU General Public License, one of the founding Open Source licenses, and OSI-Certified.

3.8. *Criteria: Internationalisation and Localisation*

Some programs assume text is written in English (ASCII). For people who use non-English languages, these programs are barely usable. And more, though many systems can handle not only ASCII but also ISO-8859-1, some of them cannot handle multi byte characters for Chinese, Japanese, and Korean languages, or combined characters for Thai.

The Review evaluated support for multiple languages at the data handling level and user interface levels.

Internationalisation is needed in the following places:

- displaying characters for the users' native languages;
- inputting characters for the users' native languages;
- handling storage and retrieval of data and files written in popular encodings that are used for the users' native languages;
- using characters from the users' native languages for file names and other items;
- printing out characters from the users' native languages;
- displaying messages by the program in the users' native languages;
- formatting input and output of numbers, dates, money, etc., in a way that obeys customs of the users' native cultures;
- classifying and sorting characters, in a way that obey customs of the users' native cultures;
- using typesetting and hyphenation rules appropriate for the users' native languages.

The Review puts emphasis on the first three items, which are the base of internationalisation support.

The following terminology is used:

Internationalisation means modification of software or related technologies so that software can potentially handle multiple languages, customs, and so on in the world.

Localisation means implementation of a specific language for an already Internationalised software.

3.8.1. *Localisable UI*

Strings (texts) belonging to the UI are managed through an Internationalisation library, such as GNU Gettext, that separates UI strings from the code, and support locales for date and number formats. Must be Unicode compliant and support easy Localisation (tools such as those supplied with gettext are a plus). It is desirable that the UI supports BiDi text (for right-to-left languages). None of the systems used standard libraries, preferring instead to roll their own. These solutions have different tradeoffs between memory consumption, execution speed and ease of maintenance. Not all the systems support date and number format locales.

ATutor: Straightforward localisation scheme using a list of Localised strings held in the database (slow execution, hard to maintain, less memory consumption). Does not seem to support Unicode, it only to supports the ISO-8859 code page. Does not support BiDi text or format locales.

Ilias: Straightforward localisation scheme, using a list of Localised strings stored in specially formatted files that are (slow execution, easy to maintain, low memory consumption) and a locale setting. Supports Unicode – all Localisations are using UTF-8. Does not seem to support BiDi text.

Moodle: Straightforward localisation scheme, using a list of Localised strings stored in PHP-formatted files (fast execution, easy to maintain at the cost of more memory consumption) and a locale setting. Supports Unicode – although most Localisations are using legacy codepages – and BiDi text.

3.8.2. *Localised to Relevant Languages (situation in May 2004)*

It is expected that LMSs have already existing locale and strings for English. It is desirable that they have a wider range of already implemented Localisations using UTF-8.

ATutor: 12 languages available, none uses UTF-8. Lithuanian localisation – partly.

Ilias: 10 languages available, all in UTF-8, with the Simplified Chinese Localisation making use of high characters.

Moodle: 31 languages available, plus seven national Localisations. 66 language packs in July 2005. Lithuanian localisation – partly, interface only; no localisation for glossary and help.

3.8.3. *Unicode Text Editing and Storage*

Where the system allows content editing, it must support editing of wide Unicode characters.

- Data storage must support Unicode.
- Editing interfaces (forms for forums, etc.) must support Unicode.

ATutor: Only supports ISO 8856 posts, some browsers (Mozilla, for instance) work around this limitation HTML-encoding high characters. Other browsers display but not post outside of ISO 8856 characters.

Ilias: Uses UTF-8 as charset, application-wide. This makes support for high characters transparent, at the expense of legacy browsers.

Moodle: With the default ISO-8856, some browsers (Mozilla, for instance) work around this limitation HTML-encoding high characters. Other browsers display but not

post outside of ISO 8856 characters. Setting the locale to use UTF-8 and altering the templates charset setting provides consistent Unicode support.

3.8.4. *Time Zones and Date Localisation*
All dates must be stored with an unambiguous time zone (TZ), preferably UTC. This is to support scenarios where users are in different time zones.

ATutor: Supports only one TZ.

Ilias: Supports only one TZ.

Moodle: Stores time/dates in UTC, and recognizes in which TZ the server and each user is.

3.8.5. *Alternative Language Support*
The system should allow for alternative language versions of the same resource.

Content negotiation support: The system should allow the user to choose directly (or indirectly via browser settings) to access alternative language versions for resources that have them.

ATutor: No support.

Ilias: Supports multiple languages for some content types, doesn't support content negotiation.

Moodle: No support.

3.9. *Criteria: Accessibility*

The Review has focused on the availability of accessibility hooks in the existing implementation.

3.9.1. *Text-only Navigation Support*
The system supports text only navigation and other accessibility hooks, such as:

- hidden links,
- link shortcuts,
- descriptive link texts,
- full support for ALTs for images and rich media.

ATutor: Excellent text-only navigation, including link shortcuts, ALT texts, etc.

Ilias: Cannot be used with text-only navigation, due to frames.

Moodle: Limited or no support for text-only navigation. Images lack ALT texts, navigation does not provide hidden links nor keyboard shortcuts. Some links use ambiguous link texts.

3.9.2. *Scalable Fonts and Graphics*
Graphics and text scale correctly when the user requests large fonts on the browser.

ATutor: Text and images are scalable.

Ilias: Text is scalable, graphics are fixed-size. Due to use of frames, the interface is not usable with large fonts.

Moodle: Text is scalable, graphics are fixed-size.

3.10. *Criteria: Document Transformation*

System should support processes for single source documents creation and multiple output transformation, with a primary goal of authoring content that can be delivered in two specific output formats: web (HTML) and print (PDF).

It is desirable that the authoring facility works with a standard format, such as DocBook.

ATutor: There is no support for format transformations. Content authoring is in pseudo HTML. Content packaging is available, which may help the process.

Ilias: Latest version can perform format transformations from XHTML content to PDF using HTMLDoc. Authoring is in XHTML, but very limited.

Moodle: There is no support for format transformations. Content authoring is in HTML, text and wiki formats.

3.11. *Summary of Findings*

The short-listed Open Source systems show significant differences in their design, architecture and implementation. Regardless of the fact that they share a common underlying framework, the impact of the differences is important.

On the overall evaluation, Moodle shows a clear advantage, particularly in criteria that is critical to the long-term viability of the system. The primary differentiating advantages of Moodle are as follows:

- **System Architecture**

Moodle's main strength is its simple but solid design and architecture. Moodle's architecture sets an excellent foundation, following good practices of low coupling and high cohesion, which the other LMSs fail to achieve. This yields a system that is simple, flexible and effective; and easily accessible to developers.

A modular, extensible architecture is a key criterion. The Moodle approach is pragmatic, using intelligent strategies, e.g., the code is split into modules and standard libraries, and a standard API for modules. Authentication is modular and separate from the rest of the modules. This will allow easier integration with a portal framework, and an interface to student management systems. By comparison ATutor is not modular with authentication strongly tied to the database throughout the system code.

- **Community**

There is a lively developer community built around Moodle, with programmers other than the main maintainer contributing sizeable modules and fixes; a second criterion the other systems fail to meet. The project code is mature. ATutor lacks a true development community possibly due to the closed approach their developers have taken.

Conclusions

Moodle does have limitations, notably it currently lacks IMS support, and its roles and permissions system is limited. However these problems can be fixed, and are part of the project roadmap (Table 11).

Table 11
Summary

Summary	ATutor	Ilias	Moodle	Comment
Architecture and implementation	Weak: no modularity	Complex: tight coupling	Good: high cohesion, low coupling	Considerations in architecture and implementation impact all other areas
Interoperability	Good	Good	Average	Overall interoperability has good potential
Cost of ownership	Medium	High	Low	Correlates with architecture and implementation findings
Strength of the community	Low	Medium	High	Related to openness of process
Licensing	GPL	GPL	GPL	Ilias offers optional features that depend on non-FOSS software
Internationalisation	Weak	Average	Good	
Accessibility	Excellent	Bad	Average	Good accessibility is critical for users of alternative browsers (screen readers, etc.)
Document transformation	No	Average	No	Complex, needs further analysis

ATutor, while strong in features and usability, has serious architectural problems although some features in ATutor warrant further investigation, and may be candidates for porting to Moodle.

Ilias, while promising, has a complex architecture with tight coupling that is hard to work with and debug. The code is new, and lacks maturity. The developer community for Ilias is very small outside the core team. Some features in Ilias deserve to be reviewed for porting to Moodle.

4. Conclusions

Several typical proprietary and Open Source VLEs were properly evaluated in the article against well-developed pedagogical and technical criteria.

WebCT Vista, Blackboard Academic Suite, Granada Learnwise version 3, FirstClass, LAMS, COSE and Moodle e-learning platforms and VLEs were evaluated against pedagogical criteria by Britain and Liber in February 2004. Among them Moodle seems to be the best VLE suitable to use on the module level. It has shown clearly that Open Source VLEs are not less quality on module level than proprietary products while being more

attractive for educational institutions from financial point of view (no licensing fees). *Therefore it would be logical to suggest Lithuanian educational institutions to widely implement Open Source VLEs.*

Moodle was the best among three Open Source VLE evaluated under technical criteria. It's also quite obvious that Moodle would demonstrate better results in comparison with ATutor or Ilias if be evaluated against the Pedagogical Framework criteria. There is also a number of European Open Source VLEs such as Fle3 and Plone / Zope which will be developed during CALIBRATE project in order to obtain more integrated and collaborative learning toolbox. Among them Fle3 was surveyed in Lithuania (Lipeikienė, 2003) and evaluated as slow working environment.

Therefore currently Moodle is the most suitable VLE for wide implementation in Lithuanian general education and vocational training institutions, as well as for teacher in-service training system. Its fundamental advantages in comparison with the other Open Source systems are:

- clear social constructivist philosophy and design;
- modular, extensible architecture;
- wide and lively developer and user community.

References

ATutor Learning Content Management System.
`http://www.atutor.ca/`

Beer, S. (1985). *Diagnosing the System for Organizations*. Chichester, Wiley.

Britain, S., and O. Liber (1999). *A Framework for Pedagogical Evaluation of Virtual Learning Environments*.
`http://www.jisc.ac.uk/index.cfm?name=project_pedagogical_vle`

Britain, S., and O. Liber (2004). *A Framework for the Pedagogical Evaluation of eLearning Environments*.
`http://www.cetis.ac.uk/members/pedagogy/files/4thMeet_framework/VLEfullReport`

IMS Learning Design Best Practice and Implementation Guide. Revision: 20 January 2003.
`http://www.imsglobal.org/learningdesign/ldv1p0/imsld_bestv1p0.html`

Biezunski, M., M. Bryan, S.R. Newcomb (1999). *ISO/IEC 13250:2000 Topic Maps: Information Technology – Document Description and Markup Languages*.
`http://www.y12.doe.gov/sgml/sc34/document/0129.pdf`

Laurillard, D. (1993). *Rethinking University Teaching – a Framework for the Effective Use of Educational Technology*. Routledge, London.

Lipeikienė, J. (2003). Virtual learning environments as a supplement to traditional teaching. *Informatics in Education*, **2**(1), 53–64.

Martinez, M. (2000). *Designing Learning Objects to Personalize Learning*.
`http://reusability.org/read/chapters/martinez.doc`

Mason, R. (1998). Models of online courses. *ALN Magazine*, **2**(2).
`http://www.aln.org/alnweb/magazine/vol2_issue2/Masonfinal.htm`

Van Merriënboer, J.J.G., and F. Pass (2003). *Powerful Learning and the Many Faces of Instructional Design*: *Toward a Framework for the Design of Powerful Learning Environments*. Pergamon, Amsterdam.

Moodle Course Management System.
`http://moodle.org/`

Schneider, D.K. (2004). *Conception and Implementation of Rich Pedagogical Scenarios through Collaborative Portal Sites*.
`http://tecfa.unige.ch/proj/seed/catalog/docs/sevilla03-schneider.pdf`

Technical Evaluation of selected Learning Management Systems (2004).

http://eduforge.org/docman/view.php/7/18/LMS%20Technical%20Evaluation%20-%20May04.pdf

WebCT e-Learning System.
http://www.webct.com/

Wiley, D. (2000). *Connecting Learning Objects to Instructional Design Theory*: *a Definition, a Metaphor, and a Taxonomy*. Utah State University.
http://wiley.ed.usu.edu/docs/astd.pdf

E. Kurilovas has graduated from the Vilnius University as the specialist in the applications of mathematics and obtained the degree of master of mathematics and informatics in 1986. Since 2001 he is a head of the Projects Division of the Centre for Information Technologies in Education under the Ministry of Education and Science of the Republic of Lithuania. He is in charge of organization of international co-operation, preparation of projects of national significance and performance of scientific research on ICT implementation in education.

He is PhD student of informatics engineering in Institute of Mathematics and Informatics and Vilnius Gediminas Technical University since 2003. His present research object is analysis of virtual learning environments for students with special educational needs.

Keli virtualių mokymosi aplinkų techninio ir pedagoginio vertinimo aspektai

Eugenijus KURILOVAS

Šiuo metu Lietuvos bendrojo lavinimo ir profesinio mokymo sistemose yra aktyviai diegiamos virtualios mokymosi aplinkos (VMA) ir mokymo(si) objektų (MO) saugyklos. Straipsnio tikslas yra apžvelgti MO sąveikumo standartų plėtros tendencijas, o taip pat palyginti VMA pagal egzistuojančias gerai išvystytas pedagoginio bei techninio vertinimo gaires bei pasiūlyti tinkamiausią VMA platesniam diegimui bendrojo lavinimo ir profesinio mokymo bei mokytojų kvalifikacijos tobulinimo sistemose aktyviems socialinės konstruktyvistinės pedagogikos metodams palaikyti.