Model-based approach to automated provisioning of collaborative educational services[⋆]

Raul Llopis Gandia[1], Sławomir Zieliński[2][0000−0002−0824−2608], and Marek Konieczny[2][0000−0002−1167−8568]

[1] Universitat Politecnica de Valencia, Campus d'Alcoi, Spain
[2] AGH University of Science and Technology, Kraków, Poland
raulloga@epsa.upv.es
{slawek,marekko}@agh.edu.pl

Abstract. The purpose of the presented work was to ease the creation of new educational environments to be used by consortia of educational institutions. The proposed approach allows teachers to take advantage of technological means and shorten the time it takes to create new remote collaboration environments for their students, even if the teachers are not adept at using cloud services. To achieve that, we decided to leverage the Model Driven Architecture, and provide the teachers with convenient, high-level abstractions, by using which they are able to easily express their needs. The abstract models are used as inputs to an orchestrator, which takes care of provisioning the described services. We claim that such approach both reduces the time of virtual laboratory setup, and provides for more widespread use of cloud-based technologies in day-to-day teaching. The article discusses both the model-driven approach and the results obtained from implementing a working prototype, customized for IT trainings, deployed in the Małopolska Educational Cloud testbed.

Keywords: model driven architecture, cloud services provisioning, collaborative education, STEM

1 Introduction

Cloud environments, though conceptually straightforward, are perceived as not easy to get started with, because they require significant effort and knowledge to be configured and used properly[9]. For that reason, they are not used in education as frequently as they could be. On the other hand, the proliferation of broadband Internet access in recent years resulted in significant reduction of technological barriers against integrating such services in courses' curricula.

⋆ The research presented in this paper has been partially supported by the funds of Polish Ministry of Science and Higher Education assigned to AGH University of Science and Technology.

As pointed out by the 2020 EDUCAUSE Horizon Report, the numbers of students grew in recent years, but "much of the growth has come from a significant increase in adult learners who are either returning for additional learning or seeking postsecondary credentialing"[4, p.33]. Cloud based collaboration environments provide for addressing the needs of that group, especially when the respective courses are led according to distance or blended distance learning patterns, which are among the students' favourites.

As also stated in the report, by "forming innovative consortia, many smaller institutions have been able to avoid closure". However, as we observe from both from our practice, and from the opinions of people participating in the Małopolska Educational Cloud project, briefly described in section 4.2, it is much easier to form a consortium than to provide added value to the learners. From our perspective, reluctance to exit one's comfort zone is a very important obstacle to wider adoption of modern educational tools. Even if the teachers use cloud services, they are not willing to change the tools they use, mainly because of the time it takes to set up a new environment. Tasks such as installation, configuration and – especially – granting privileges to the students, can be time consuming and error-prone. That discourages teachers from preparing short-lived environments, to be used, e.g., only to illustrate a single topic. Moreover, the typical unstructured way of sharing knowledge, which relies only on instructing people on how to use a particular tool or providing 'howto' documentation, does not result in increased willingness to experiment with new tools.

The approach presented in this paper assumes that cloud-based ICT tools are needed by multiple members of an educational community. We propose an MDA-based service orchestration system, capable of instantiating educational environments (compound services) on the resources possessed by the community, according to teachers' demands. The system relies both on resource pooling, and on pooling of expertise provided by the community members. By 'expertise' we understand both the knowledge possessed by IT staff about particular environment setup, and the knowledge of tools' applicability areas possessed by the teachers. We present a way of providing a structured pool of templates and propose a high level interface for describing educational environments, to make the deployment of new environments straightforward. Aside from hiding the technicalities from a teacher, the presented system automates the installation and configuration tasks, and reduces the setup times from several hours to minutes.

The structure of the paper is as follows. Section 2 surveys the important developments in the subject area. Section 3 describes the processing of user-defined models, which results in creating a ready-to-use service. Section 4 discusses the results of evaluating a proof of concept implementation of the system. Section 5 concludes the paper and points out the directions of future work.

2 Related Work

Since the introduction of MIT OpenCourseWare in 2001, many universities opened their curricula to online communities. The wide adoption of Massive

Open Online Courses (MOOCs) by educational institutions strengthen that process. Nonetheless, the importance of practical verification and experimentation in teaching engineering and computational sciences is crucial and broadly studied [8,10]. Students need to explore topics not only in theory, but in practice as well. As a result, many educational organizations opened the laboratory materials to virtual, remote, or hybrid education. There are many motivations behind that: to increase its reach, to improve curriculum, or to reduce costs [12,13,11].

However, the process of creating educational environments on demand is not widely studied yet. Popular open platforms for MOOCs[3] [4] often require special agreement with educational institutions. Similar requirements apply to platforms that focus on delivering technology-oriented trainings[5] [6]. They also focus on providing a complete curriculum rather than a flexible platform for provisioning educational environments. Some platforms require specialized hardware or software[3,5,6] and may be technologically complex[7]. The majority of mentioned platforms use only public cloud infrastructure and cannot leverage private resources, which functionality is often necessary for an educational organization.

There are solutions which utilize domain-specific languages to support educators in creating labs. However, they are bound to specific execution environments e.g OpenStack platforms [14] or proprietary technologies [7] - which also increases the technology learning curve for educators.

The EEDS, introduced in this article, tackles quite a few of the mentioned problems, as explained in sections 3 and 4.

3 Model driven service orchestration

In this section we describe our approach to supporting educational communities by fostering sharing of knowledge regarding composition of educational environments. We assume that the communities are composed of institutions that provide similar, or at least related, courses to their respective students, and are willing to share their how-to knowledge within the community of teachers. We start from describing and justifying the main architectural elements of of the proposed educational environment deployment system (EEDS), then we describe the service orchestration process.

3.1 EEDS applicability area

Education is a process typically organized in the form of courses. The courses cover a collection of related topics. In our opinion, any tool designed to support education needs to be fitted to a course-topic model, such as the one depicted in figure 1, which also provides a mapping between topics and educational activities.

[3] edX, `https://www.edx.org/`
[4] Coursera, `https://www.coursera.org/`
[5] Vocareum, `https://www.vocareum.com/`
[6] Qwiklabs, `https://www.qwiklabs.com/`
[7] OpenEdx, `https://github.com/edx`

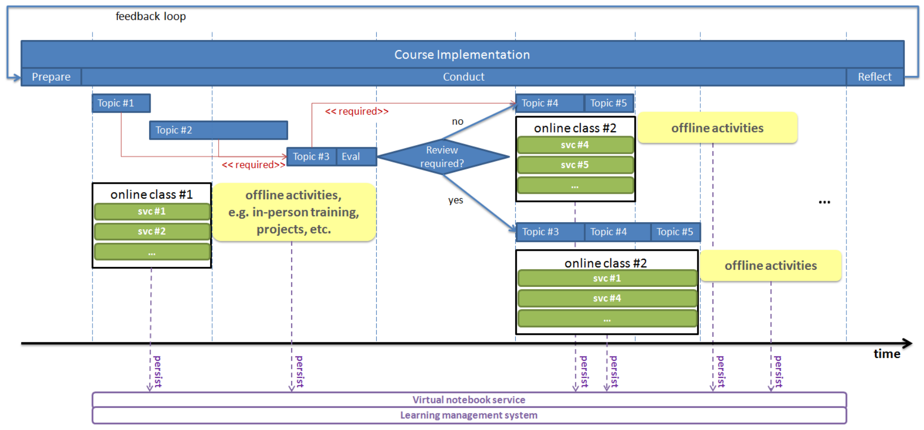


Fig. 1. A model of a partly remote course.

In the introduced model, courses are implemented as sequences of educational activities, and therefore continuity between the activities is one of the key requirements to be addressed by any supporting system. Indirectly, this also implies the need for documenting classes and sharing materials related to the respective topics. These requirements are typically fulfilled by learning management systems (LMSs). Some institutions use also more advanced services, which allow not only to store and share materials, but also provide platforms for organizing and annotating them. We refer to such services as virtual notebooks.

EEDS is designed as a system to be used for setting up educational environments for respective online classes. It is capable of deploying the services needed during an online class, configuring the required virtual machines or containers, distributing the tasks and materials for the students, and collecting the results of their work. The sets of services that constitute educational environments could be different for different online classes, so the state of students' work needs to be kept outside EEDS. That allows for effective reuse of resources – the instances used by a class can be destroyed just after the class finishes and the resources can be available for reuse very quickly. Section 3.2 discusses the matters in more detail.

3.2 EEDS architecure overview

An educational environment is a short-lived complex service, created by an institution for its students. Because of the assumed similarity of topics, it is likely that the same environment is reused multiple times. We claim that proper organization of sharing how-to knowledge (containing service descriptors and contextual information) allows for easier reuse of the environments. Therefore, the most important architectural elements of EEDS (depicted in figure 2) are respective repositories, which are used by the educational environment orchestrator.

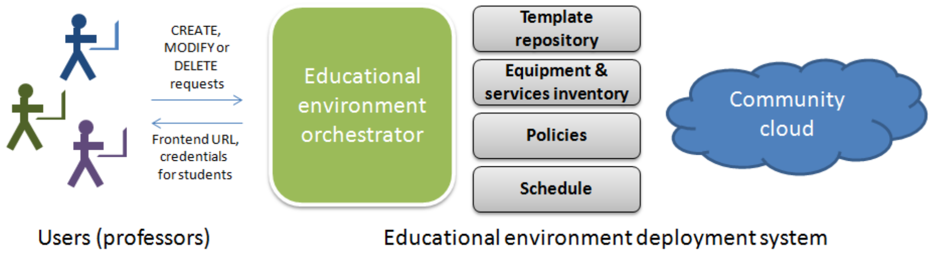

CREATE, MODIFY or DELETE requests
Frontend URL, credentials for students
Users (professors)
Educational environment orchestrator
Template repository
Equipment & services inventory
Policies
Schedule
Community cloud
Educational environment deployment system

Fig. 2. Educational environment deployment system building blocks.

The template repository contains templates which simplify the conversions between Computation Independent Models (CIMs) and Platform Independent Models (PIMs). The templates include technical parameters set to values chosen by IT staff, and presented to the users in an easily understandable form, so that the teachers who request an educational environment are not overwhelmed with details. The inventory of equipment and services simply keeps track of computational nodes and existing instances of basic services, e.g., firewalls, available to the system. The policies repository contains declaratively specified administrative policies, and the schedule organizes the tasks to be performed by EEDS.

A CIM, according to the Model Driven Architecture standard specified by the Object Management Group - "only describes business concepts"[2, p.13]. In EEDS case, the users are responsible for preparing a valid CIM that describes the educational environment they want to use during classes, and submit it to the orchestrator, which takes care of the instantiation process. It is safe to assume that the environment is composed of many services, especially given the typical class phases, which are preparation of tools and tasks, instruction, distribution of the tasks, collection of results, grading and providing feedback, and – finally – persisting the process outcomes. In a typical scenario, the class is expected to use at least an audiovisual connectivity service, a file storage, a learning management system, and a sort of virtual notebooks, which can be treated as separate services. Depending on the topic, additional services may be taken into account, including groupware and collaboration tools. Additional backend services, e.g., authentication, authorization and accounting service are also sure to be used, because the environment's use is both time and user constrained.

The main design guideline of EEDS was to provide an easy to use provisioning mechanism that would cover the details of instantiating complex, cloud-based collaboration environments. That was achieved by providing a common layer of abstraction, expressed in the form of CIMs. That is an important advantage from the point of view of educators who are not adept to rapidly changing contemporary cloud technologies. The CIMs, specified by the professors, undergo conversion into respective PIMs, which in turn are converted to a single or multiple Platform Specific Models (PSMs). Finally, a single PSM is selected for execution by administrative policies. The process is depicted in figure 3.

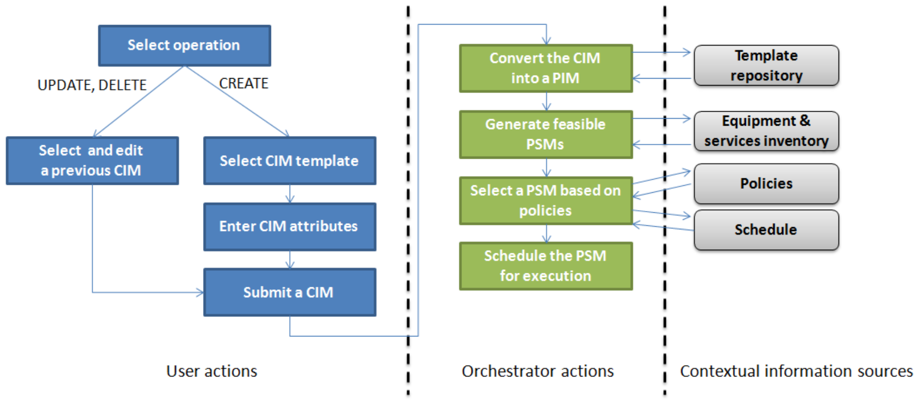
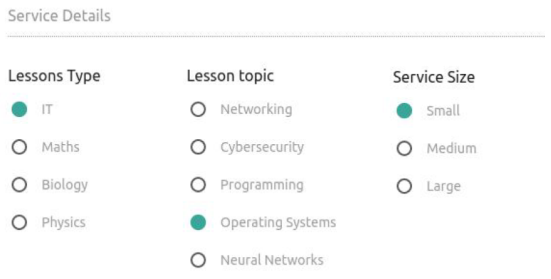

Select operation
UPDATE, DELETE
CREATE
Select and edit a previous CIM
Select CIM template
Enter CIM attributes
Submit a CIM
Convert the CIM into a PIM
Generate feasible PSMs
Select a PSM based on policies
Schedule the PSM for execution
Template repository
Equipment & services inventory
Policies
Schedule
User actions
Orchestrator actions
Contextual information sources

Fig. 3. Simplified workflow of CIM processing.

The process begins when an educator – using a web interface – specifies the model to be processed, as well as the operation to be performed by the EEDS orchestrator. There are three operations that can be selected: creation, modification or deletion of an environment. In the following considerations we focus on service creation. Once the data set describing attributes of a CIM is complete and valid, the request can be processed. Figure 4 contains a fragment of a web interface used to define a CIM.


Service Details

Lessons Type: [✓ IT] Maths Biology Physics

Lesson topic: Networking Cybersecurity Programming [✓ Operating Systems] Neural Networks

Service Size: [✓ Small] Medium Large

Fig. 4. CIM details specification interface.

Using a three-models structure is important, because it allows for handling multiple implementations of the same abstract service, as the PIM – though technical – is by definition technologically agnostic. The conversion process is carried out by using templates, which convert descriptive CIM attributes to technical PIM attributes. The resulting model is still abstract, and can be mapped to one of as many implementations as the particular system offers. The conversion rules are defined by the contents of the template repository. Table 1 shows an excerpt of such rules regarding VM configuration.

Table 1. A template used to convert a VM description from its CIM to PIM form.

Attribute	CIM form	PIM form
vmSize	small/medium/large	architecture, vCPUs, RAM, OS, storage, virtualization
cooperation	isolated/groups/ common	admin and users' accounts, groups, data directories, filesystem privileges
persistence	datastore name	file transfer protocol and file URI
network addresses	n/a	static/dynamic
firewall rules	n/a	rules to be applied

In the example shown in table 1 the educator needs to specify the size of a service (which should be related to the number of students participating in the online class), the mode of cooperation - isolated students, isolated groups or one large group, and a name of datastore to hold the results of the students' work. Those values are converted to more technical ones, but the rules of conversion are not stiff, rather they depend on the subject of the course. For example, different values regarding virtual machine size are assigned to a programming course, and different to a statistical analysis course. At the same stage of conversion, usernames are generated, the data store name converted into a file transfer protocol name and URLs for the files. Moreover, subnet IP address, gateway, DNS, and firewall rules are generated. Note that the requesting user does not need to deal with the network-related details, which – from our observation – is also an important hurdle for many teachers.

After converting the descriptive attributes to technical ones, all PSMs which describe the possible implementations of the educational environment, are generated. In case the equipment and services inventory contains multiple implementations of the requested services, multiple PSMs are generated. Again, the template repository is queried for converting the PIM attributes to attributes specific for the implementation (the respective template contains environment variable names, command line arguments, etc.).

Eventually, administrative policies are applied, and service schedules are consulted to select a single PSM. First of all, the screening policy decides which of the feasible PSMs are within the privileges of the requesting user. Then, the selection policy checks which of the PSMs can be run on the infrastructure at the specified time period, and whether or not it requires preemption of previously scheduled tasks. Finally, a single PSM is scheduled for execution.

In the event all the feasible PSMs do not comply with the policies, the process ends and the requesting user is properly notified by an e-mail message. In order to allow for reacting (e.g., by submitting a modified request), the EEDS defines the minimum time before execution threshold, after which the requests cannot be proceeded. That allows also for including human work in the environment preparation phase in the future.

Once selected, the PSM describing the compound service requested by a teacher is scheduled in a priority queue. After taking such a model out of the queue, the orchestrator generates code for the specific platform, execution of which results in provisioning the requested environment. After the requested usage time, the orchestrator takes care of persisting the results of students' work and destroys the environment.

4 Proof of concept implementation and evaluation

This section describes a proof-of-concept implementation of EEDS[8] – a system that complies with the architecture described in section 3.2, designed for teaching information technologies, but not limited to that area. We start the discussion with a brief description of the implementation, including technologies used in the process (subsection 4.1) to show that the architecture can be implemented using open technologies only. Then we describe the Małopolska Educational Cloud project (subsection 4.2), which may benefit from the service in the future, and which infrastructure was used during evaluation. We continue with describing the environment in which the evaluation was performed (4.3), and present some of the functional evaluation results (4.4) and processing time measurements (4.5).

4.1 Proof of concept implementation

The frontend of the EEDS prototype was implemented using the LAMP Open Source software package (Linux, Apache, MySQL, PHP) which makes up a web service stack to dynamically provide the webform and have control of the request reception system. The stack architecture is based on the Presentation-Business-Data architecture – it distinguishes between three independent modules implemented with different technologies but connected together.

We considered two options for representing the templates and policies: YAML and JSON. We chose YAML mailny because it is visually easier to understand and therefore facilitates the readability of the data (which leads to easier error detection). The readability is especially useful in defining policies, such as presented in the following excerpt, which defines a screening policy for a teacher:

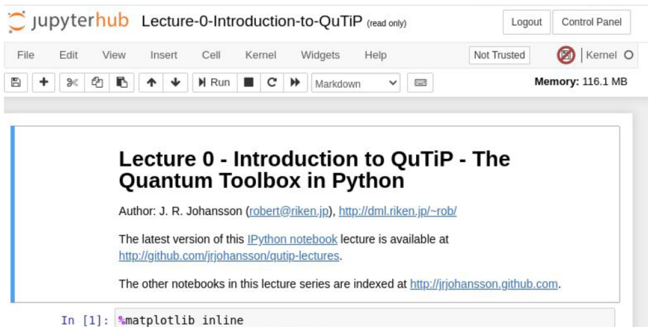
```
maximums:
  teacher:
    small: {users: 15 , groups: 5 , availability: 7 }
    medium: {users: 25 , groups: 8 , availability: 3 }
    large: {users: 35 , groups: 11 , availability: 1 }
```

The prototype is capable of setting up educational environments using either Vagrant (to create a workflow to provision a virtual machine) or Docker (for deployment of containers). The design and architecture of the EEDS are open and allow integration of other environments. The prototype was configured to instantiate two educational environments, based on The Littlest JupyterHub (TLJH), and Antidote SelfMedicate, respectively.

TLJH was chosen as a platform for providing various types of lessons. The software was designed to handle 100 users on a single machine, so it is able to

[8] EEDS Repository: https://github.com/llopisga/cloud-orchestration

satisfy most of the educators' requests in that aspect. TLJH is a lightweight, Docker-based solution, capable of being adapted for arbitrarily chosen topics, presented in the form of Jupyter notebooks[1]. Figure 5 depicts a TLJH-based lesson interface.


jupyterhub Lecture-0-Introduction-to-QuTiP (read only)
Logout
Control Panel
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Kernel
Run
Markdown
Memory: 116.1 MB

Lecture 0 - Introduction to QuTiP - The Quantum Toolbox in Python

Author: J. R. Johansson (robert@riken.jp), http://dml.riken.jp/~rob/

The latest version of this IPython notebook lecture is available at http://github.com/jrjohansson/qutip-lectures.

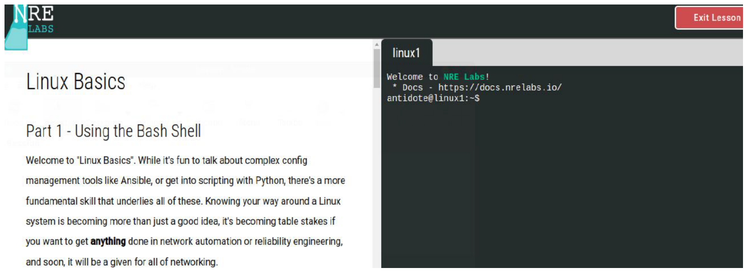
The other notebooks in this lecture series are indexed at http://jrjohansson.github.com.

```
In [1]: %matplotlib inline
```

Fig. 5. TLJH service instantiated for a quantum physics lesson

Antidote (an open source project developed by NRE Labs) aims to facilitate learning network automation, and is provisioned by EEDS as a VM set up with Vagrant. The service splits the lesson window into two sides, containing the lesson text content and a Bash terminal, respectively (see fig. 6). Antidote Selfmedicate allows to implement an identical service by spinning up a virtual machine with Vagrant, to which lessons can be added using YAML configuration files. The lessons may refer to different topics that involve the use of the terminal. Each Antidote-based environment is made up of four layers:

- infrastructure - virtual machine run on Virtualbox,
- MiniKube - a single-node cluster used for providing the lessons on demand,
- Antidote platform - for loading the lessons and providing web interface,
- curriculum - where the lessons are specified in YAML.



Linux Basics

Part 1 - Using the Bash Shell

Welcome to "Linux Basics". While it's fun to talk about complex config management tools like Ansible, or get into scripting with Python, there's a more fundamental skill that underlies all of these. Knowing your way around a Linux system is becoming more than just a good idea, it's becoming table stakes if you want to get **anything** done in network automation or reliability engineering, and soon, it will be a given for all of networking.

```
Welcome to NRE Labs!
 * Docs - https://docs.nrelabs.io/
antidote@linux1:~$
```

Fig. 6. Antidote service instantiated for a Linux basics lesson

4.2 Małopolska Educational Cloud

Małopolska Educational Cloud (MEC) is a community formed by 20 university departments, more than 120 high and vocational schools and several other institutions, including 11 pedagogical libraries and 7 teacher excellence centers. The community was formed to foster collaboration between universities and schools scattered over the Małopolska region (see figure 7). By participating in MEC activities, high school students learn more about particular universities and areas of study before they make decisions regarding their further education. MEC partners organize regular courses on various topics (e.g., information technologies, civil engineering, vocational English), each of which lasts at least one semester. The courses are led collaboratively by university and school teachers, the former being responsible mainly for online classes, the latter for offline activities. The rationale for implementing MEC and user activity analysis is discussed in [15,16] in more detail.

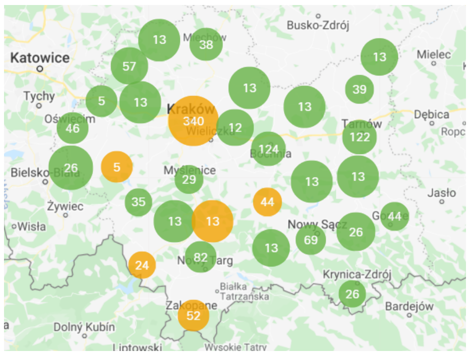


Fig. 7. MEC coverage as of Feb 2021; the numbers denote the quantities of network devices installed at respective locations.

From the beginning, MEC activities are split into:

– **didactic activities**, led by respective groups of institutions, and
– **infrastructural activities**, led by AGH University, the leader of the MEC community.

Didactic activities. Each of the MEC high schools participates in at least two interest groups which organize online courses. Currently, MEC supports over 50 such groups. The groups – consisting of one university and up to five schools – organize online courses every semester. Typically, a group is established for a period of two semesters and involves about 100 high school students.

Infrastructure for resource pooling. MEC resources are offered to the participating schools according to either IaaS or SaaS paradigms. AGH University, the leader of the project, runs a private cloud hosting most of the services on its premises, providing participants with access to storage and computational resources. Additionally, users may leverage public clouds of their choice.

MEC implemented a dedicated overlay network, connecting all project participants. They use the overlay to access MEC services hosted in a private cloud. The list of MEC services includes: audiovisual connectivity, recording and storage, collaborative editing, etc., to be used during or after online classes. The outcomes of the online classes, as well as other materials prepared by or for students, are kept in a social media portal, which plays the role of a virtual notebook (see figure 1). MEC does not provide any unified LMS - the respective institutions use systems of their choices.

MEC is developing its own orchestrator of services, conceptually similar to the one presented in the article. However, a few differences are present. First, the goal of MEC orchestration is to provide an interface for moderating online classes led for people coming from many institutions. That is outside EEDS scope of interest. Second, the MEC orchestrator targets synchronizing the changes in the operational modes of many services, during the online class, while EEDS does not, as it would require specific instrumentation of the orchestrated services. Third, the MEC orchestrator is oriented on audiovisual connectivity as the most important service, and tries to leverage the hardware AV terminals capabilities. EEDS does not follow that pattern, but focuses on technology openness.

4.3 Experimental environment

In order to conduct the evaluation of the EEDS proof-of-concept implementation, we used three virtual machines provided by the MEC IaaS service. One of the machines (master) was assigned a frontend role, while the others (workers) formed a small pool of servers which were used to instantiate the requested educational environments. Table 4.3 contains technical specifications of the VMs.

Table 2. Technical parameters of testbed VMs

VM	Master	Worker 1	Worker 2
vCPUs	2	4	4
RAM	4GB	64GB	64GB
Storage	16GB	16GB	16GB
Operating system	CentOS 8	CentOS 8	CentOS 8
Virtualization	IVT	IVT	IVT

The machines were accessed by the MEC VPN, and were accessible from the schools' internal networks by using the aforementioned MEC overlay network.

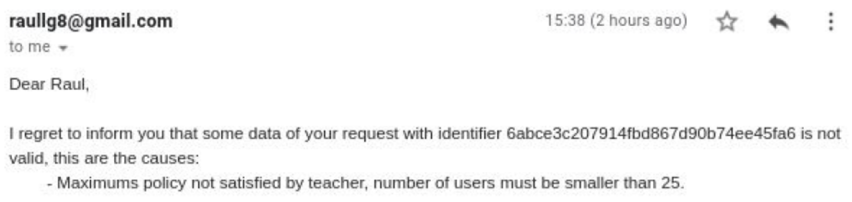
4.4 Functional evaluation

At first, we conducted functional evaluation of the EEDS prototype. Among other features, we tested the orchestrator capability to detect policy breaches (note that policies are applied after the feasible PSMs set is generated). One of the tests that were conducted used four requests, described in table 3. As

Table 3. Most important attributes specified in the initial test requests.

Attribute	Request 1	Request 2	Request 3	Request 4
machine size	small	medium	medium	large
course	IT	biology	maths	IT
topic	operating systems	bioinformatics	calculus	networking
users	8	14	26	14
groups	4	isolated	common	isolated

expected, requests 1,2 and 4 were accepted and resulted in deployment of environments, while request 3 was rejected due to a policy breach (see fig. 8 for rejection message contents). The characteristics of the respective environments generated by the successful requests 1, 2 and 4, are summarized in table 4.


raullg8@gmail.com
to me
15:38 (2 hours ago)

Dear Raul,

I regret to inform you that some data of your request with identifier 6abce3c207914fbd867d90b74ee45fa6 is not valid, this are the causes:
- Maximums policy not satisfied by teacher, number of users must be smaller than 25.

Fig. 8. Automatically generated message indicating a policy breach.

Table 4. Most important attributes of the platforms generated by requests 1,2 and 4.

Attribute	Request 1	Request 2	Request 4
dest. node	node02	node02	node01
service	Antidote	Jupyter	Antidote
virtualization	Vagrant	Docker	Vagrant
resources	2 vCPUs, 8GB RAM	4 vCPUs, 16GB RAM	8 vCPUs, 32GB RAM

4.5 Processing time measurements

We measured deployment and lesson loading times for both kinds of environments the EEDS prototype was capable of creating, and for different machine sizes. The machine sizes and key characteristics are summarized in table 5.

Not surprisingly, both the deployment and lesson loading times decreased with increased machine capabilities. As depicted in fig. 9, the creation times of the Antidote machines were up to 15 minutes - most of the time was spent on downloading the needed packages from Internet repositories. The time it took to load a lesson (the Linux Basics lesson) was decreasing from slightly over a minute to about 30 seconds. The deployment of TLJH took about 5 minutes, and the respective lesson loading time was less than 10 seconds.

Table 5. Characteristics of environments instantiated by EEDS prototype.

Attribute	Antidote 1	Antidote 2	Antidote 3	Antidote 4	TLJH 1	TLJH 2	TLJH 3
machine size	small	medium	medium	large	small	medium	large
vCPUs	2	2	4	6	2	4	8
RAM	4GB	8GB	8GB	12GB	8GB	16GB	32GB
virt. provider	Virtualbox	Virtualbox	Virtualbox	Virtualbox	Docker	Docker	Docker

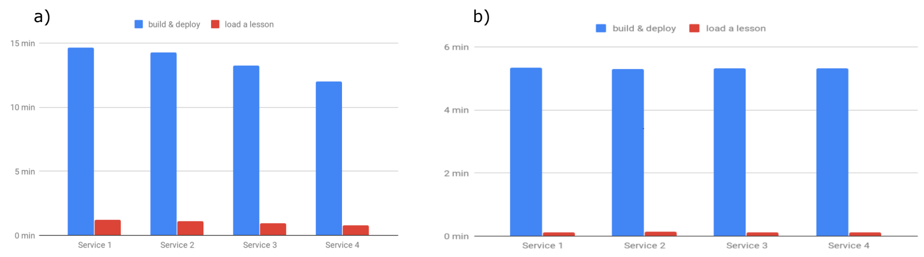


Fig. 9. Deployment and lesson load times for a) Antidote and b) TLJH services.

5 Conclusions

The described work proved that providing an MDA-based platform for deploying educational environments for online classes on arbitrary topics is feasible, and can be accomplished using open technologies only. The presented prototype of the EEDS system proved to be capable to instantiate the declaratively specified environents in a short time, measured in minutes. Therefore we claim that the main goals were achieved. Nonetheless, we see fields for improvement – which we set as our development goals – that could make the system usable on a larger scale. One of them is integrating the system with users' registry and an external authentication and authorization service to free the educators from the task of distributing user credentials. Second on the list is the integration with an LMS. The third goal is to provide the system with a monitoring service that would provide for reflecting upon the services' current and future use.

References

1. A gallery of interesting Jupyter Notebooks, `https://github.com/jupyter/jupyter/wiki/A-gallery-of-interesting-Jupyter-Notebooks`. Last accessed 10 Feb 2021
2. OMG Model Driven Architecture (MDA) Guide rev. 2.0, OMG Document ormsc/2014-06-01 (2014), `https://www.omg.org/cgi-bin/doc?ormsc/14-06-01.pdf`. Last accessed 10 Feb 2021
3. Auer, M.E., Villach, C.: A Toolkit to Facilitate the Development and Use of Educational Online Lab-oratories in Secondary Schools. Seattle, Washington
4. Brown, M., McCormack, M., Reeves, J., Brook, D.C., et al.: 2020 Educause Horizon Report Teaching and Learning Edition. Tech. rep., EDUCAUSE (2020)
5. Cadenas, J.O., Sherratt, R.S., Howlett, D., Guy, C.G., Lundqvist, K.O.: Virtualization for cost-effective teaching of assembly language programming. IEEE Transactions on Education **58**(4), 282–288 (2015)
6. Costa, R., Pérola, F., Felgueiras, C.: μLAB A remote laboratory to teach and learn the ATmega328p μC. In: 2020 IEEE Global Engineering Education Conference (EDUCON). pp. 12–13. IEEE (2020)
7. Demchenko, Y., Belloum, A., de Laat, C., Loomis, C., Wiktorski, T., Spekschoor, E.: Customisable data science educational environment: From competences management and curriculum design to virtual labs on-demand. In: 2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom). pp. 363–368. IEEE (2017)
8. Feisel, L.D., Rosa, A.J.: The role of the laboratory in undergraduate engineering education. Journal of engineering Education **94**(1), 121–130 (2005)
9. Kim, B., Henke, G.: Easy-to-Use Cloud Computing for Teaching Data Science. Journal of Statistics Education pp. 1–18 (2020)
10. Lee, H.S., Kim, Y., Thomas, E.: Integrated Educational Project of Theoretical, Experimental, and Computational Analyses. In: ASEE Gulf-Southwest Section Annual Meeting 2018 Papers. American Society for Engineering Education (2019)
11. Lynch, T., Ghergulescu, I.: Review of virtual labs as the emerging technologies for teaching STEM subjects. In: INTED2017 Proc. 11th Int. Technol. Educ. Dev. Conf. 6-8 March Valencia Spain. pp. 6082–6091 (2017)
12. Morales-Menendez, R., Ramírez-Mendoza, R.A., et al.: Virtual/remote labs for automation teaching: A cost effective approach. IFAC-PapersOnLine **52**(9), 266–271 (2019)
13. Perales, M., Pedraza, L., Moreno-Ger, P.: Work-in-progress: Improving online higher education with virtual and remote labs. In: 2019 IEEE Global Engineering Education Conference (EDUCON). pp. 1136–1139. IEEE (2019)
14. Soceanu, A., Vasylenko, M., Gradinaru, A.: Improving cybersecurity skills using network security virtual labs. In: Proceedings of the International MultiConference of Engineers and Computer Scientists 2017 Vol II, IMECS (2017)
15. Zieliński, K., Czekierda, Ł., Malawski, F., Straś, R., Zieliński, S.: Recognizing value of educational collaboration between high schools and universities facilitated by modern ICT. Journal of Computer Assisted Learning **33**(6), 633–648 (2017)
16. Zygmunt, M., Konieczny, M., Zielinski, S.: Accuracy of statistical machine learning methods in identifying client behavior patterns at network edge. In: 2019 42nd International Conference on Telecommunications and Signal Processing (TSP). pp. 575–579. IEEE (2019)