Integration of Computer Science Assessment into Learning Management Systems with JuezLTI

Juan V. Carrillo
CIFP Carlos III, Cartagena, Spain

Alberto Sierra
CIFP Carlos III, Cartagena, Spain

José Paulo Leal
CRACS – INESC-Porto LA & DCC – FCUP, Porto, Portugal

Ricardo Queirós
CRACS – INESC-Porto LA & uniMAD, ESMAD/P. Porto, Portugal

Salvador Pellicer
Entornos de Formación (EdF), Valencia, Spain

Marco Primo
Faculty of Sciences, University of Porto, Portugal

Abstract

Computer science is a skill that will continue to be in high demand in the foreseeable future. Despite this trend, automated assessment in computer science is often hampered by the lack of systems supporting a wide range of topics. While there is a number of open software systems and programming exercise collections supporting automated assessment, up to this date, there are few systems that offer a diversity of exercises ranging from computer programming exercises to markup and databases languages. At the same time, most of the best-of-breed solutions force teachers and students to alternate between the Learning Management System – a pivotal piece of the e-learning ecosystem – and the tool providing the exercises.

This issue is addressed by JuezLTI, an international project whose goal is to create an innovative tool to allow the automatic assessment of exercises in a wide range of computer science topics. These topics include different languages used in computer science for programming, markup, and database management.

JuezLTI borrows part of its name from the IMS Learning Tools Interoperability (IMS LTI) standard. With this standard, the tool will interoperate with reference systems such as Moodle, Sakai, Canvas, or Blackboard, among many others. Another contribution of JuezLTI will be a pool of exercises. Interoperability and content are expected to foster the adoption of JuezLTI by many institutions. This paper presents the JuezLTI project, its architecture, and its main components.

2012 ACM Subject Classification Applied computing → Computer-managed instruction; Applied computing → Interactive learning environments; Applied computing → E-learning

Keywords and phrases programming, interoperability, automatic assessment, programming exercises

Digital Object Identifier 10.4230/OASIcs.ICPEC.2022.9

Funding JuezLTI is Co-funded by the Erasmus+ Programme of the European Union with ref. 2020-1-ES01-KA226-VET-096004. The European Commission support for the production of this publication does not constitute an endorsement of the contents which reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

1 Introduction

Computing skills will be among the most in-demand skills of this new decade [3]. These skills range from building a cloud platform to maintaining it and require solid knowledge in programming, database, and markup languages.

Most Learning Management Systems (LMS) do not support automated assessment in these domains. Hence, academic environments resort to the best-of-breed learning tools. To integrate these tools into the LMS they use several non-standard approaches, ranging from plugins to ad hoc Web services.

The IMS Learning Tools Interoperability (IMS LTI) specification [4] emerged in the last decade to facilitate the integration of the LMS with external tools. Its main goal is to release learning agents from the burden of having to authenticate themselves in multiple tools to benefit from a more engageable learning experience.

Nonetheless, to the best of the authors' knowledge, there is yet no available open tool, seamlessly integrated with the LMS, supporting the automatic assessment of computer science exercises from different domains, including but limited to computer language programming.

This paper presents the current status of a tool for automatic assessment of computer science exercises, under development as part of an international project within the scope of the Erasmus+ programme, key action: *Cooperation for innovation and the exchange of good practices* [8]. The objective of this project is a tool – called JuezLTI – to support the automatic assessment of markup languages, programming, and databases exercises.

As its name suggests, JuezLTI interoperates with online learning environments such as Moodle, Sakai, Canvas, or Blackboard (among many others), thanks to the LTI standard. The improved interoperability and the pool of exercises to be developed will open it to many institutions.The target audience of JuezLTI is computer science instructors and students (also self-education). All the project outputs will be freely available on the Internet under open-source licenses.

The remainder of this paper is organized as follows. Section 2 surveys both CS learning environments and the models they use for LMS integration. Section 3 presents the JuezLTI architecture and its main components. A particular emphasis is placed on the interoperability ensured by the LTI specification. The final section summarizes the current status and identifies opportunities for future developments.

2 Related work

Learning Management Systems (LMS) play a central role in any e-learning ecosystem. These systems were created to deliver course content, collect student assignments, and evolved to provide more versatile and engaging tools, including exercise assessment.

The Computer Science (CS) domain is an apt example of this evolution. In the last decades, due to the high demand of computing skills, a panoply of web-based interactive exercise systems were created to foster coding practice. In that time, most LMS lacked out-of-the-box support for automatic assessment of programming exercises, which is an important feature for a computer science learning environment. This lack of support meant that teachers and students had to switch between tools to enhance programming language learning. In order to solve these issues, some LMS offer integration of external tools as simple modules known as plugins. Some interesting examples are Virtual Programming Lab (VPL)[1], eLiza [2] and Javaunittest[2].

[1] `https://vpl.dis.ulpgc.es/`
[2] `https://moodle.org/plugins/qtype_javaunittest`

A Virtual Programming Lab (VPL) is a programming assignment management system where students can edit and execute programs, with automatic and continuous assessment. eLiza [2] is a game-based educational tool, developed and used to support the teaching and learning of programming languages and paradigms related to the development of web applications. Javaunittest is a Java question type that allows teachers to create Java questions and test the knowledge of students about Java programming. The students type source code for a given interface in Java and the response gets graded automatically.

Table 1 CS exercises assessment systems integrated into LMS.

Systems	Module[1]	WS[2]	LTI v.1[3]
ACOS			X
Codeboard			X
CodeCheck			X
CodeHS			X
CodeWorkOut			X
DBQA			X
eLiza	X		
javaunittest	X		
PERE		X	
Virtual Progr. Lab	X		
Web-Cat			X

[1] Module = implemented as a plugin

[2] WS = implemented as a web service

[3] LTI 1 = used the LTI specification 1.0 or 1.1

Despite its apparently success, this approach hinder the modularization and interoperability of tools. In a world where LMS were appearing at a fast pace, other specifications were born to overcome the LMS dependency avoiding the need to create for each tool a module for each LMS supported. One notorious example is the IMS LTI specification which offers seamless integration with most popular LMS in the market such as Moodle, Canvas, Blackboard, Sakai, and others. In this case the LMS takes the role of a platform (tool consumer) whereas the external tool serves as a tool (tool provider). A platform can send (anonymous) student information to the external tool and, in return, the tool reports the grades of students back to the platform (LMS). This way, programming exercises can be seamlessly integrated into every platform that supports LTI [9]. With the evolution and consequent acceptation of the specification a big number of computer programming environments started to support LTI. Most popular tools are CodeWorkOut, DBQA and Web-Cat.

CodeWorkOut[3] is an online exercise system to help people learn a programming language for the first time. It is a free, open-source solution for practicing small programming problems. Web-CAT [15] is a flexible, tailorable automated grading system designed to process computer programming assignments. Database Query Analyzer (DBQA) [7] is a tool that illustrates the effects that clauses and conditions have on an SQL SELECT statement using a visualized, data-oriented approach.

Table 1 presents a few of the most popular solutions and their interoperability features organized in three integration flavours: as a plugin, using ad hoc Web Services (WS) and using LTI specification (v1.0/1). As can be seen, most systems support the LTI v1.1 specification

[3] `https://codeworkout.cs.vt.edu/`

for integration with the LMS. The most recent version of LTI (v1.3) is not yet supported by these systems; this may be due to being a recent specification and with a higher degree of complexity than previous versions.

3 JuezLTI

This section presents JuezLTI – a tool integrated into the LMS to assess exercises in languages used in computing. Examples of such languages are markup languages (a domain where few assessment tools are available), database management languages, and programming languages. Subsection 3.1 provides an overview of JuezLTI, presents its architecture, and introduces its components. The following subsections describe the main components types of this architecture. Finally, Subsection 3.5 discusses the integration of JuezLTI on the LMS.

3.1 General overview

The goal of JuezLTI is to provide assessment of exercises in a wide range of languages used in computing within the LMS. To attain this goal JuezLTI relies on a combination of features:

1. the interoperability with the LMS provided by the TSUGI framework;
2. a modular design allowing the incorporation of evaluators for new domains;
3. a generic feedback system to help students to overcome their difficulties;
4. a centralized repository of exercises from where they can be exported and imported.

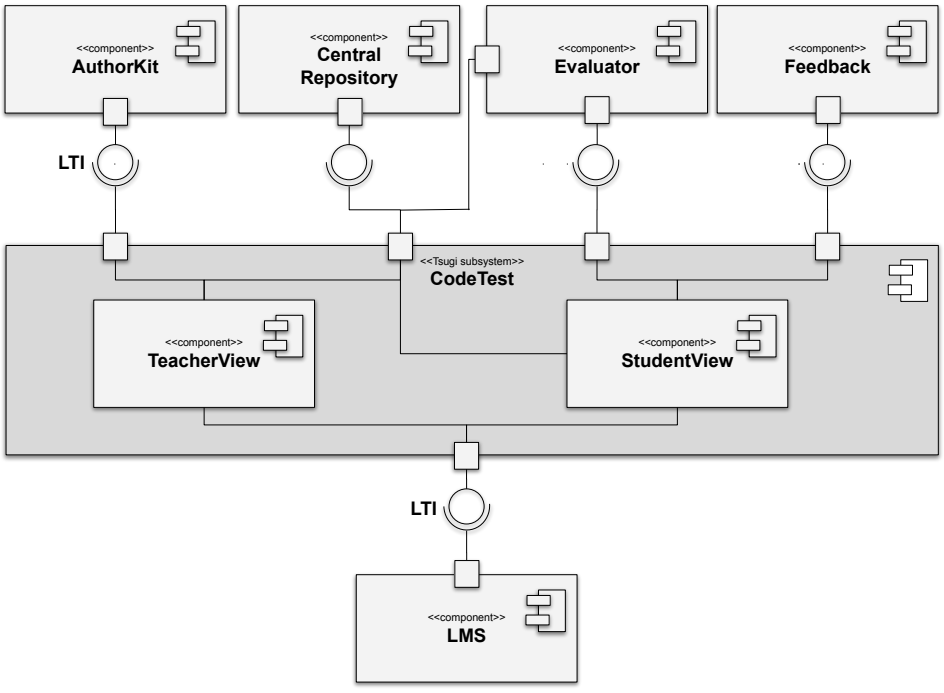

Figure 1 JuezLTI internal and external components.

The architecture of JuezLTI is depicted by the UML component diagram in Figure 1. At the center of this architecture is CodeTest, the main component based on TSUGI [6], the framework that provides LTI support, and its sub-components TeacherView and StudentView.

The former is a web environment where the teacher configures the activity by adding exercises from a central repository. The latter is a web environment where the student attempt to solve the proposed exercises. The TSUGI-based component relies on several kinds of components depicted on the top of the diagram in Figure 1. From left to right they are Autorkit, Central Repository, Evaluator, and Feedback. AuthorKit [12] is an exercise authoring system that can be used by teachers to create exercises for JuezLTI. The central repository acts as a pool of exercises for the system. The evaluator component runs the code of the students and checks the results. Finally, the feedback system provides information about the result of exercise resolution.

JuezLTI is an open-source project distributed under an Apache 2.0 license. It can be installed on the servers of any institution and configured to communicate with the institution's Learning Management System. It is available on GitHub in two different forms: production and development. The former allows the deployment of the entire system (several docker components) on a production server open to the Internet. With the latter, anyone can build the platform locally and contribute to JuezLTI development.

3.2 Exercise resolution

There are several tools described in the literature developed to automatically assess programming exercises, using different approaches such as static or dynamic evaluation of code; some examples are AutoGrader, eGrader, or Kassandra [17]. It's also easy to find tools to grade SQL assessments, such as Learn-SQL, a tool based on the IMS QTI specification [1]; unfortunately, a closed solution. But, despite the massive presence of markup languages in the computer science curriculum, there are few solutions to assess XML exercises.

JuezLTI proposes a tool that supports the evaluation of programming, databases and XML exercises, but that can be extended to other types of exercises given its highly modular architecture.

The module in charge of exercise resolution is called CodeTest. CodeTest relies on other JuezLTI modules for retrieving exercises, evaluating code, and generating feedback. It has web interfaces for two roles: teachers and students. The teacher's role is to create, edit, import, and export a set of exercises. Teachers can also review the results of every single student. Students view exercise statements and have a code editor to introduce their resolution. As JuezLTI supports multiple programming languages, the student can choose the language to solve the exercise, restricted to the context of the exercise. For instance, in a programming language exercise, the student may have the possibility to select among different languages such as Java, Python, or C.

JuezLTI also acts as a central repository of questions and answers. To do that, it stores all the exercises proposed in a non-relational database. External questions can also be imported using the tool AuthorKit.

3.3 Evaluation

The core of JuezLTI is the automated evaluation of different kinds of computer science exercises. According to the kind of exercise, a different module performs the evaluation. Currently, JuezLTI has in development modules to evaluate exercises in programming languages, markup language, and relational databases.

A module responsible for the evaluation of a kind of exercises is called an *evaluator*. Evaluators are micro-services, with their own internal structure, running on their own containers. For instance, a relational databases evaluator can have its own database engine,

as a programming language evaluator may have installed several compilers for the languages it supports. These micro-services share a common API used by CodeTest to invoke them. This API specifies the data sent to evaluators – exercises, student attempts – and return by them – evaluation reports.

Student attempts are plain text files. In contrast, exercises are a complex content, including several files – a statement, one or more solutions , several test cases and other auxiliary files – as well as specialized metadata. For instance, a relational database exercise requesting an SQL query may include a database dump and a reference solution; a programming language exercise may include a set of input files and corresponding expected output. Exercises sent to evaluators are encoded in the YAPExIL [13].

An evaluator returns a report detailing the assessment performed on the student's attempt using the data provided by the exercise. This report includes possible compilation errors, including resource usage - memory and time. It may also include hints associated with test cases if these were provided by the exercise. Evaluation reports are encoded as a JSON document in the Programming Exercise Evaluation Assessment Report Language (PEARL). This specification is an evolution of similar XML-based specification [10] developed specifically for JuezLTI.

3.4 Feedback

The role of the feedback manager is to mediate between evaluators and students to distill effective feedback from evaluation reports. Evaluators produce reports with perfusion of details. This information needs to be summarized to make it understandable to students.

Students typically submit several attempts before solving an exercise. Hence, feedback messages must avoid being repetitive and strive to be increasingly more informative. For instance, a first feedback message may simply acknowledge that most students have difficulty in solving the exercise in their first attempt; a second message may provide a hint associated with a failed test case in the evaluation report; a third may provide an input that causes the program to fail.

On the other hand, automated feedback may also be detrimental to some students if they use it as a sort of oracle that constantly solves their difficulties. The feedback manager must also ensure that the information it provides does not scale up too quickly in reaction to submissions that are too similar to the previous ones.

3.5 Interoperability

JuezLTI uses version 1.0 of the Learning Tools Interoperability standard (LTI) for integration into the LMS. This interoperability standard was proposed by the IMS Global Learning Consortium [14] and is supported by reference LMS vendors such as Moodle, Canvas, or Sakai.

With LTI a set of educational services can be used to extend the functionality of the LMS [11] using a Service Oriented Architecture (SOA) [14]. The LMS becomes a marketplace where the teacher can select the resources from different providers to improve the learning experience.

JuezLTI is based on TSUGI, a framework that simplifies the development and deployment of LTI applications [6]. The first application of TSUGI was a self-contained MOOC for training in Python – py4e.com. It was deployed by one of its authors, Dr. Charles Severance, founder of the Open Source LMS Sakai and currently working for IMS. LTI has been widely used since 2010, not only for code testing purposes, as in [16], but also for providing complex assessments not directly supported by the LMS, such as Dig4E, a tool to learn about standards for creating high-quality digital still images [5].

To use JuezLTI the teacher has to request a key/secret pair on the project's website. With those credentials and the supplied URL, the teacher adds an external activity in the LMS. The students are directly identified in JuezLTI using the information provided by the LMS. Whenever a student or teacher opens the activity, the LMS communicates with JuezLTI using LTI (via the TSUGI framework) to identify the user and present the appropriate interface. Then, the LMS sends the `resource_link_id` property identifying the relevant activity. This way, different activities can be added using the same key. Finally, JuezLTI reports grades of solved exercises back to the LMS using an LTI service.

In short, there is bidirectional communication between JuezLTI and the LMS. JuezLTI stores the information of students and their results and sends the grades obtained back to the LMS.

4 Conclusion

The main contribution of the research reported in this paper is JuezLTI – a platform to assess computer science exercises. JuezLTI is an interoperable, open-source, and modular platform. Usability, easiness of use, and extensibility were its main design goals.

JuezLTI supports exercises in different kinds of languages, including programming, database management, and markup. Due to its modular structure, evaluators for new domains can be easily added. It will have access to a centralized repository of exercises adapted to EQF levels, which will promote its adoption by teaching institutions. The TSUGI framework, upon which JuezLTI is based, supports LTI version 1.0, and future versions of the framework will gain access to more advanced versions of the standard with limited recodification. JuezLTI is a work-in-progress. It is currently under beta testing, and some of its components are still being improved, reflecting the suggestions and bugs reported by early adopters. This project intends to contribute with a collection of exercises compatible with its evaluators. Collecting and generating these exercises is also part of our immediate future work.

References

1 Alberto Abelló, M. Elena Rodríguez, Toni Urpí, Xavier Burgués, M. José Casany, Carme Martín, and Carme Quer. LEARN-SQL: Automatic assessment of SQL based on IMS QTI specification. In *2008 Eighth IEEE International Conference on Advanced Learning Technologies*, pages 592–593, 2008. doi:10.1109/ICALT.2008.27.

2 Míriam Antón-Rodríguez, María Ángeles Pérez-Juárez, Francisco Javier Díaz-Pernas, David González-Ortega, Mario Martínez-Zarzuela, and Javier Manuel Aguiar-Pérez. An Experience of Game-Based Learning in Web Applications Development Courses. In Ricardo Queirós, Filipe Portela, Mário Pinto, and Alberto Simões, editors, *First International Computer Programming Education Conference (ICPEC 2020)*, volume 81 of *OpenAccess Series in Informatics (OASIcs)*, pages 3:1–3:11, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/OASIcs.ICPEC.2020.3.

3 European Commission. Coding – the 21st century skill, 2018. accessed on 20 Jan 2020. URL: https://ec.europa.eu/digital-single-market/en/coding-21st-century-skill.

4 IMS Global Learning Consortium. Learning tools interoperability core specification, 2019. accessed on 14 Apr 2022. URL: http://www.imsglobal.org/spec/lti/v1p3/.

5 Paul Conway and Ann Arbor. Digitization for everybody (Dig4E). *Archiving Conference*, 2020:12–16, April 2020. doi:10.2352/issn.2168-3204.2020.1.0.12.

6 Nikolas Galanis, Marc Alier, María José Casany, Enric Mayol, and Charles Severance. Tsugi: A framework for building PHP-based learning tools. In *Proceedings of the Second International Conference on Technological Ecosystems for Enhancing Multiculturality*, TEEM '14, pages 409–413, New York, NY, USA, 2014. Association for Computing Machinery. doi:10.1145/2669711.2669932.

7 Ryan Hardt and Esther Gutzmer. Database query analyzer (DBQA): A data-oriented SQL clause visualization tool. In *Proceedings of the 18th Annual Conference on Information Technology Education*, SIGITE '17, pages 147–152, New York, NY, USA, 2017. Association for Computing Machinery. doi:10.1145/3125659.3125688.

8 JuezLTI Project Consortium. JuezLTI - automatic assessment of computing exercises using LTI standard, 2021. accessed on 12 Apr 2022. URL: https://juezlti.eu.

9 José Paulo Leal and Ricardo Queirós. Using the learning tools interoperability framework for LMS integration in service oriented architectures. *Technology Enhanced Learning TECH-EDUCATION'11*, 2011.

10 José Paulo Leal, Ricardo Queirós, and Duarte Ferreira. Specifying a programming exercises evaluation service on the e-framework. In Xiangfeng Luo, Marc Spaniol, Lizhe Wang, Qing Li, Wolfgang Nejdl, and Wu Zhang, editors, *Advances in Web-Based Learning - ICWL 2010 - 9th International Conference, Shanghai, China, December 8-10, 2010. Proceedings*, volume 6483 of *Lecture Notes in Computer Science*, pages 141–150. Springer, 2010. doi:10.1007/978-3-642-17407-0_15.

11 Jeff Merriman, Tom Coppeto, Francesc Santanach Delisau, Cole Shaw, and Xavier Aracil Díaz. *Next generation learning architecture*. eLearn Center. Universitat Oberta de Catalunya, 2016.

12 José Carlos Paiva, Ricardo Queirós, José Paulo Leal, and Jakub Swacha. Fgpe authorkit–a tool for authoring gamified programming educational content. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*, pages 564–564, 2020.

13 José Carlos Paiva, Ricardo Queirós, José Paulo Leal, and Jakub Swacha. Yet Another Programming Exercises Interoperability Language (Short Paper). In Alberto Simões, Pedro Rangel Henriques, and Ricardo Queirós, editors, *9th Symposium on Languages, Applications and Technologies (SLATE 2020)*, volume 83 of *OpenAccess Series in Informatics (OASIcs)*, pages 14:1–14:8, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/OASIcs.SLATE.2020.14.

14 Charles Severance, Ted Hanss, and Josepth Hardin. IMS learning tools interoperability: Enabling a mash-up approach to teaching and learning tools. *Technology, Instruction, Cognition and Learning*, 7(3-4):245–262, 2010.

15 Anuj Ramesh Shah. *Web-cat: A web-based center for automated testing*. PhD thesis, Virginia Tech, 2003.

16 Antonio J. Sierra, Álvaro Martín-Rodríguez, Teresa Ariza, Javier Muñoz-Calle, and Francisco J. Fernández-Jiménez. LTI for interoperating e-assessment tools with LMS. In Mauro Caporuscio, Fernando De la Prieta, Tania Di Mascio, Rosella Gennari, Javier Gutiérrez Rodríguez, and Pierpaolo Vittorini, editors, *Methodologies and Intelligent Systems for Technology Enhanced Learning*, pages 173–181, Cham, 2016. Springer International Publishing.

17 Zahid Ullah, Adidah Lajis, Mona Jamjoom, Abdulrahman Altalhi, Abdullah Al-Ghamdi, and Farrukh Saleem. The effect of automatic assessment on novice programming: Strengths and limitations of existing systems. *Computer Applications in Engineering Education*, 26(6):2328–2341, 2018. doi:10.1002/cae.21974.