A Roadmap to Convert Educational Web Applications into LTI Tools

José Paulo Leal
CRACS & INESC Tec LA / Faculty of Sciences, University of Porto, Portugal

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

Pedro Ferreirinha
Faculty of Sciences, University of Porto, Portugal

Jakub Swacha
University of Szczecin, Poland

Abstract

This paper proposes a roadmap to integrate existing educational web applications into the ecosystem based on a learning management system. To achieve this integration, applications must support the Learning Tools Interoperability specification in the role of tool provider. The paper starts with an overview of the evolution of this specification, emphasizing the main features of the current stable version. Then, it proposes a set of design goals and milestones to guide the adaptation process. The proposed roadmap was validated with existing applications. This paper reports on the challenges faced to apply it in these concrete cases.

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.2022.12

Funding This paper is based on the work done within the FGPE Plus: Learning tools interoperability for gamified programming education project supported by the European Union's Erasmus Plus programme (agreement no. 2020-1-PL01-KA226-HE-095786).

1 Introduction

Learning management systems (LMS) play a pivotal role in the way instruction is delivered in many schools. These systems manage most of the learning activities in which students must participate. However, some of these activities may be better handled by other educational web applications.

For instance, a typical LMS provides automated assessment features but not on specialized domains such as programming languages. On the other hand, web applications such as Sololearn, CoderByte, CodeWorkout, CodeWars, CodinGame, HackerRank, and LearnJS provide this kind of assessment and would benefit from a tighter integration into the LMS ecosystem.

The Learning Tools Interoperability (LTI) specification [3] provides a standard approach for this kind of integration. It evolved from a simple mechanism to launch web applications from the LMS, to support multiple services to securely interchange data with these applications. LTI is a standard supported by all LMS vendors of reference, and thus maximizes the payoff of adapting an educational web application.

A practical example can be given using CodeWorkout as an external third-party tool provider and Canvas as an LTI-compliant LMS acting as a Tool Consumer. Using LTI, it is easy to seamlessly integrate CodeWorkout as a Canvas Assignment embedded in a Canvas

page. This enables students to code the exercises from within Canvas in the same way that they might do with any Canvas-native assignment, and receive a score in the Canvas gradebook for doing that assignment.

Despite these advantages, adapting an existing educational web application to support LTI is far from a painless process. Designing a system from scratch to use this specification is complex enough. Redesigning an existing system in such as way as to retain all its previous features and simultaneously take the most from LTI integration is even more challenging. Moreover, there are several hurdles to consider, from the configuration of the LMS itself to make it support a given LTI version, to selecting libraries to support the specification on the web application side.

The remainder of the paper is organized as follows. Section 2 provides a background on the LTI specification and surveys the best frameworks and libraries to implement it. Section 3 identifies a set of design goals and milestones to guide the adaptation process. Section 4 reports on the use of a selected LTI library to integrate a code playground into an LMS. Finally, Section 5 summarizes the contributions of this work and points to future directions.

2 State of the Art

Learning platforms have evolved in several dimensions. The interoperability dimension was one of the dimensions that progressed the most [2]. The first platforms were monolithic and closed, with all the features provided by internal and proprietary components. In the last decades, platforms become decentralized as they are built by assembling external components exposed as services. These services are decoupled from their consumers, and thus can easily be reused by several learning processes.

These services led to the formalization of initiatives [8] to adapt Service-Oriented Architectures (SOA) to e-learning called e-learning frameworks. The ultimate goal of these initiatives was to simplify the integration of educational systems, allowing the creation of dynamic educational ecosystems easily adaptable to any learning domain. They were composed of open interfaces to numerous reusable services organized into genres or layers and combined in service usage models. Despite the good intentions, the complexity in formalizing the specifications of the frameworks undermined the adhesion of the major educational players [6].

In parallel, other interoperability initiatives focused on simplifying the educational systems' integration emerged, such as NSDL, POOL, OKI, EduSource, IMS DRI, IMS LTI. Based on these initiatives, several authors presented works on connecting LMS to external applications [9, 2, 1].

2.1 IMS Learning Tools Interoperability (LTI) specification

An initiative that stood out was the IMS Learning Tools Interoperability (LTI) specification [3] defined as a standardized way to integrate third-party content into courses within an LMS or platform. Without LTI, LMS and content providers would have to rely on custom integrations with prejudice to interoperability and scalability.

Nowadays, the IMS consortium recommends the adoption of version 1.3. In short, LTI 1.3 uses OAuth2 and signed messages using JSON Web Tokens (JWTs) to securely authenticate students and pass data between the LMS and the external learning tool.

This version includes LTI Advantage, defined as a package of services that add new features to enhance the integration of any tool with any LMS in the core of LTI 1.3. The three LTI Advantage feature services are Names and Role Provisioning Services, Deep Linking, and Assignment and Grade Services:

- Names and Role Provisioning Services (NRPS): grants the tool access to the list of users and their roles for a specific context (e.g. a course or group). NRPS allows the external tool to request a list of members from the context that launched it. Once the tool receives the membership list, it can read all students and teachers enrolled in the context, even if they have not yet launched the tool.
- Assignment and Grade Services (AGS): allows teachers to sync grades between third-party tools and their LMS. This service returns numeric scores and teacher comments to an LMS gradebook. For instance, a teacher can view when a student has started an assignment, or if it has been completed. The teacher can also receive the status of a grade, even if it requires manual input from the teacher.
- Deep Linking (DL): offers a more streamlined approach to adding LTI links to an LMS. Deep Linking messages allow external learning tools to appear within an LMS as internal tools would. With Deep Linking, external learning tools can now allow teachers to configure specific content or activities within the tool. In LTI 1.1, the whole tool content had to be exposed, even if the teacher only wanted the class to use a specific resource. Deep Linking allows teachers to select whichever content they need and share the link to launch that content with their course. For example, a tool may let a teacher configure a specific chapter from an e-book when their students select a link, rather than force them to launch the tool and then navigate to that chapter.

Adopting LTI 1.3 should be a careful decision. For most of the cases, LTI 1.1 is enough in use cases where a student launches an activity from the LMS, solves it in the external tool and then a grade is reported back to the LMS grade book. However, the new LTI 1.3 features could be applied in relevant use-cases. For instance, considering the use of leaderboards: names and roles obtained from provisioning services could be used to populate leaderboards with the names of students before they submitted for the first time; deep links could be used to embed in LMS content leaderboards generated on-the-fly by the tool.

2.2 LTI implementation tools

Given the popularity of LTI and its consequent adoption by the community, it is worth analyzing the existing options to develop an external application with LTI support and, thus, benefit from secure integration with an LMS while using the services provided by this standard.

Currently, there are several frameworks and libraries (from now on, LTI development tools) to implement a tool provider with preconfigured routes and methods to manage the LTI 1.3 protocol. By using these frameworks, developers may profit from LTI without needing to implement all the security and validation requirements and, this way, concentrate on the tool logic.

The LTI development tools survey was based in the following methodology: firstly, a set of tools were collected using a "lti" keyword to search on GitHub Lab [1] – a site filter for GitHub repositories. Then, tools that weren't update in the last year were discarded. Finally, the five with more stars were selected. The set of tools is the following: ltijs[2], TSUGI[3], LtiLibrary[4], ims-lti[5] and lti-1-3-php-library[6].

[1] https://githublab.com/repositories?q=lti
[2] https://github.com/Cvmcosta/ltijs
[3] https://github.com/tsugiproject/tsugi
[4] https://github.com/LtiLibrary/LtiLibrary
[5] https://github.com/instructure/ims-lti
[6] https://github.com/IMSGlobal/lti-1-3-php-library

The following subsections describe and compare these five LTI development tools. The analysis, summarized in Table 1, is based on three facets: maturity, specification coverage, and language binding.

Table 1 Analysis of the major LTI development tools.

Criteria	Properties	ltijs	TSUGI	LtiLibrary	ims-lti	lti-1-3
Maturity	First release date	May 19	Jun 17	Nov 16	May 20	Aug 18
	Last release date	May 21	Mar 22	Mar 22	May 20	Aug 20
	# Open issues	21	17	7	5	21
	# Pull requests	1	1	-	2	5
	# Commits	438	2879	604	246	110
	# Releases	5	18	14	1	-
	# Contributors	7	24	13	20	8
	# Stars	173	287	72	167	84
	# Forks	29	243	55	105	60
Coverage	LTI 1.0					
	LTI 1.1		X	X	X	
	LTI 1.3					X
	LTI 1.3 (DR included)	X				
Language Bindings	.NET			X		
	JavaScript	X				
	PHP		X			X
	Ruby				X	

Regarding maturity (based on the tools' GitHub repositories), although relatively recent, these tools have been growing with the evolution of the LTI specification. TSUGI and LtiLibrary are the oldest and more frequently updated, taking into consideration commits and releases. However, ims-lti, despite being the most recent, is the second most popular, with a reasonable number of stars. The number of forks of a repository is also relevant. Forking a repository allows to freely experiment with changing it without affecting the original project. Thus, this means more people are using the TSUGI code base to start their projects than any other. The last two tools are those with a lower number of commits and the oldest last release dates. Nevertheless, the lti-1-3-php-library, created by IMSGlobal, has the biggest number of open issues, which is revealing of the great community that the IMS presents.

LTI 1.1 is still the most popular and widespread option. In short, this version defines a standard way to launch a tool provider (typically an external learning tool) within a tool consumer (typically an LMS) and attach some user data, authentication data, and service references. Afterward, the learner uses the external learning tool. Finally, the tool provider calls a Learning Information Service (LIS) to submit the learner grade back to the LMS. Nevertheless, the IMSGlobal library has already implemented version 1.3, supporting all services from LTI Advantage. The ltijs library has the highest coverage level, being the first

library to implement the new LTI Advantage Dynamic Registration Service, which turns the LTI Tool registration flow into a fast, completely automatic process. It exposes a registration endpoint through which platforms can initiate the registration flow.

Finally, most LTI development tools are coded in one of two languages, either PHP or JavaScript (NodeJS runtime). It was expectable as these are the most popular server-side languages. The initial unfiltered set included tools coded in other languages, such as Java and Python, but most of them were in beta state.

3 LTI roadmap – design goals and milestones

This section presents a roadmap to integrate an existing educational web application into a Learning Management Systems (LMS) using the Learning Tools Interoperability (LTI).

We envisioned the object of this integration process as an existing educational web application. This application must be able to provide activities to a group of students similar to a class. It must have structured content to be presented to students as activities. This content can be either expository (PDFs) or evaluative (automated assessment). Finally, the outcome of these activities should be reported back to the LMS for integration in a grade book.

To guide us in the creation of this LTI integration roadmap we considered the following design goals:

- The web application will become an LTI tool provider (TP) that can be launched and interact with an LTI tool consumer (TC) such as an LMS.
- The web application will continue to be usable outside the LTI context, supporting simultaneously LTI and non-LTI users.
- Modifications to the web application user interface will be minimal, desirably none.
- Global configurations of TC (ex: skinning, natural language) should be exposed to the LMS.
- When possible, changes made by LTI users in the web tool will be automatically reported to the TC.
- The adaptation process will start with simple and independent tasks and then proceed to more complex ones that depend on the former.

The adaptation process should profit as much as possible from all LTI features, ranging from the core features from the early versions of the protocol to those proposed by LTI Advantage and integrated into the latest version. The core features include launching the TP in an authenticated session and configuring it from the LMS. The latest features include the Names and Role Provisioning Services (NRPS), Deep Linking (DL), and Assignment and Grade Services (AGS).

The proposed roadmap is organized into three consecutive main milestones. Each milestone is a fully functional and improved version of the web application, providing a deeper integration into the LMS and building on the previous ones. The first aims at the core integration between the TP and LMS, with user ids shared by both systems. The second aims at exposing specific TP content to different LMS activities. Finally, the third milestone aims at reporting student activity on the TP back to LMS. The following subsections detail each of these milestones.

3.1 Entrance

The main objective of the first milestone is to enable a smooth transition between the LMS (e.g. Moodle) and the TP (e.g. Agni). Providing a single-sign-on is a major driving force behind LTI adoption; this feature is available since version 1.0 of this specification. However, entering the TP when coming from the LMS involves more than just authentication and authorization. This milestone is divided into the following sub-milestones.

1. Launch the TP.
2. Expose TP configuration to the LMS.
3. Create an authenticated session.
4. Create users for the activity.

The first sub-milestone is to launch the tool using LTI. This sub-milestone requires configuring the LMS to support LTI with the tool (more on that in the following section) and configuring an activity on the LMS. This sub-milestones is concluded when the activity link is presented on the LMS and, when followed, the tool screen is presented to the user. At this stage, the TP might simply present a welcome or login screen.

The second sub-milestone is to enable the configuration of the TP using LTI custom parameters. The TP may have general configurations to expose to the LMS. For instance, the TP might support different natural languages or some form of user interface customization, such as setting a background color or a logotype. If these configurations are available they may be exposed to LMS using LTI custom parameters. This way, the teacher may configure them when defining the LTI activity on the LMS.

The third sub-milestone is the creation of an authenticated session according to the profile received via LTI. The relevant profiles are teacher and student and these should be mapped to profiles available on the tool. At this stage, the student profile can be mapped to a guest user or generic student. The teacher may be mapped to an administration profile.

Educational systems usually require students to be registered in advance. This registration is necessary since most of these systems record student progress. Many of these systems support groups of students like classes. Ideally, a class or similar should be created and associated with each LTI activity. This class can be automatically created on the first launch of an LTI activity.

Using NRPS the TP can obtain from the LMS the list of users and their profiles. With this approach, all users can be created in batch at the first launch, typically when the teacher tests the activity. In subsequent LTI launches the authorization is reduced to checking if the student was already created.

Students may enroll later in the course after the activity is configured and the class populated on the TP. Hence, if the student is missing during authorization, the process described in the previous paragraph will have to be repeated.

3.2 Content

The main objective of the second milestone is to launch the TP with a specific content. For instance, in a TP with many exercises, the teacher may need to specify a few for a particular LTI activity. This kind of launch opens the TP in any part of it, which gives the possibility of skipping a login screen or showing a restrictive single small part. This can be achieved by configuring the TP activity on the LMS using two different approaches:

1. LTI custom parameter.
2. Deep Linking (DL).

An LTI custom parameter can be used on the activity configuration to specify the content to present to the students. The drawback of this approach is that it requires the teacher to know the parameter name and the content identifier, usually a pathname. If the TP hosts more than a few individual contents, this approach may be too cumbersome.

The alternative is to use Deep Linking to provide a content selector to the teacher when she configures the LTI activity on the LMS. Although integrated on the LMS, the content selector is created by the TP, reflecting its current content. Thus, if the content has many items, it can be organized using a tree widget, for instance. Alternatively or complementary, an incremental search field can be added to facilitate the selection process.

In general, DL is preferable to custom parameters to select content items. However, custom parameters may have other uses in content selection. For instance, a custom parameter may be used to disable navigation on the TP. Using such a configuration, after an LTI launch on specific content, students will be unable to navigate different to other content on that instance.

3.3 Feedback

The main objective of the third and final milestone is to send feedback from the TP back to the LMS. It is an important facet of LTI integration since it allows the instructor to consolidate grades in the LMS, but can be exploited also for other purposes. This milestone is divided into the following sub-milestones:

1. Reporting activity grades.
2. Reporting other information.

One of the tools in an LMS is a *gradebook* – a sort of table where each row refers to a student and each column refers to an activity. The teacher may use this gradebook to manage student grades in a course and grades from LTI activities may be automatically filled in by the TP.

When an LTI activity is created in the LMS, a column for that activity is automatically added to the gradebook. This column has two components: a grade and a description. The grade is a non negative number within a certain range (e.g. 0 to 100) – and the description is a string of characters. Both these values may be null. Using AGS, other similar columns may be added to the gradebook for the same activity.

The objective of the first sub-milestone is to use the automatic column and fill it with the grade of each student that completes the activity. Using AGS, the TP should automatically report grades as soon as the students complete the activity, without requiring any action from the teacher.

The second sub-milestone requires the creation of other columns in the gradebook. These columns may be used to report data besides grades. For instance, the TP may manage HTTP sessions associated with each student and use them to compute the time they spent on the activity. This data may be reported in a new column using the numeric value for the accumulated time and the descriptive text to inform if the student is currently solving the activity.

Although the AGS was designed to reporting student related information and in particular grades, it be also exploited as a general mechanism to return data to the tool consumer. This use may be relevant if the tool consumer is not an LMS but another TP, using the LTI link to integrate this third system with a common single-sign-on domain. In this case the tool consumer will need to implement an AGS to receive the data reported by the TP.

4 LTI Adaptation Example

To validate the proposed roadmap the authors applied it in two different contexts. One is a JavaScript web playground named Agni (previously called LearnJS [5]). The other is a programming exercise authoring tool named FGPE AuthorKit [4]. Subsection 4.1 describes the required LMS and external tools configurations, similar in both cases. Then, Subsection 4.2 describes implementation details specific to Agni, and Subsection 4.3 those related to FGPE AuthorKit.

4.1 Configuration

This subsection details the configurations needed to support the integration of a tool consumer (LMS) with a tool provider.

4.1.1 Tool Consumer (LMS)

The first milestone of the roadmap requires launching the TP from the LMS. To fulfill this objective is necessary to configure an LTI activity on the LMS.

Moodle was the LMS selected for testing the integration of both applications. These applications have in common the implementation language, and both rely on the same integration library – ltijs – introduced in Subsection 2.2. Hence, the initial configuration of both applications as TP in Moodle is very similar and requires the following steps:

1. Create a tool configuration.
2. Set the tool's public key in the configuration.
3. Define the redirection URL(s).

The initial step is to set the tool type `LTI 1.3` and obtain a client ID to register the platform in ltijs. After the platform registration, the tool generates a public key for the handshaking process, to be entered in the `Public Key` of the external tool configuration. Finally, all the URL(s) that are processed inside the context must be entered in the `Redirection URL(s)`. URLs must be registered for security reasons, so that the LMS recognizes all the possible redirections inside its context.

4.1.2 Tool Provider (external tool)

To implement LTI in the tool provider, the ltijs library was the obvious choice due to its full support for the v1.3 specification, including the dynamic registration service. Also both external tools have Node.js – a JavaScript runtime built on Chrome's V8 JavaScript engine – as back-end, which perfectly suits the use of ltijs.

The first step is to install the library npm package and, since this package natively uses mongoDB by default to store and manage the server data, it is also necessary to install it.

After basic installation, it is necessary to set up ltijs into the external tool in multiple steps. Most of them can be mapped to specific methods of the ltijs library, namely:

- **setup** – to configure the database and additional parameters (routes for the reserved endpoints used by ltijs). After the `lti.setup()` method is called, the `lti` object gives you access to various functionalities to help you create your LTI Provider. The `lti.app` object is an instance of the underlying Express server, through this object you can create routes just like you would when using regular Express.
- **deploy** – opens a connection to the configured database and starts the Express server.

- **register** – registers the platform on the tool. Platform manipulation methods require a connection to the database, so they can only be used after the deploy method. A LTI tool works in conjunction with an LTI ready platform (Moodle), so in order for Moodle to display the external tool first page, it needs to first be registered in the tool itself. This way, the tool is able to generate a public key used by Moodle for further connections.

Ltijs behavior is configured through callbacks. For instance, the `onConnect` callback is called whenever a successful launch request arrives at the main app URL, or the `onDeepLinking` callback is called when a successful deep linking request is received.

Every launch generates an `IdToken` that the tool uses to retrieve information about the user and the general context of the launch. The valid `idToken` is then separated into two parts:

- **idtoken** – contains the platform and user information that is context independent (e.g., platform name, user name and email);
- **contexttoken** – contain the context specific information (e.g. activity name, platform endpoints)

Both are stored in the database and passed along to the next route handler inside of a `response.locals` object.

Ltijs needs the correct `idtoken` and `contexttoken` when a tool makes a request. The authentication protocol relies on two items, a session cookie, and a `ltik` token. At the end of a successful launch, ltijs: 1) redirects the request to the desired endpoint; 2) sets a signed session cookie containing the `platformCode` and `userId` information; 3) sends a `ltik` JWT token containing the same platform and user information, with additional context information as a query parameter to the endpoint. When the tool receives a request, it attempts to validate it by matching the information received through the session cookie with the information contained in the `ltik` token. It is important to notice that the `ltik` token must be passed to the tool through either query parameters, body parameters, or as an Authorization header (bearer or LTIK-AUTH-V1).

4.2 Agni Web Playground

Agni (formerly LearnJS) is a Web playground that enables anyone to practice the JavaScript language [7]. In the playground, students access PDFs and videos on different topics, and solve exercises related to those topics with automatic feedback on their resolutions. The playground has two main components: 1) an editor where students code their solutions in an interactive environment and 2) an evaluator to assess students' code based on static and dynamic analyzers. Figure 1 shows the front-end GUI of the playground.

The following subsections discusses three Agni features aligned with the milestones presented at Section 3. The features are exercise sheet launching, leaderboard generation, and score submission.

4.2.1 Entrance – Leaderboard generation

In Agni, every successful launch from the platform to the tool should present two sections: the exercise sheet, so that the student can solve the exercises and a leaderboard with the current state of participant progress.

After a first successfully launch, NRPS is used to automatically create all the platform's users (referred to as members) and their roles within the context of the course where the LTI activity belongs. All members of a platform within the context can be retrieved simply by calling the **getMembers** method as shown in Listing 1.

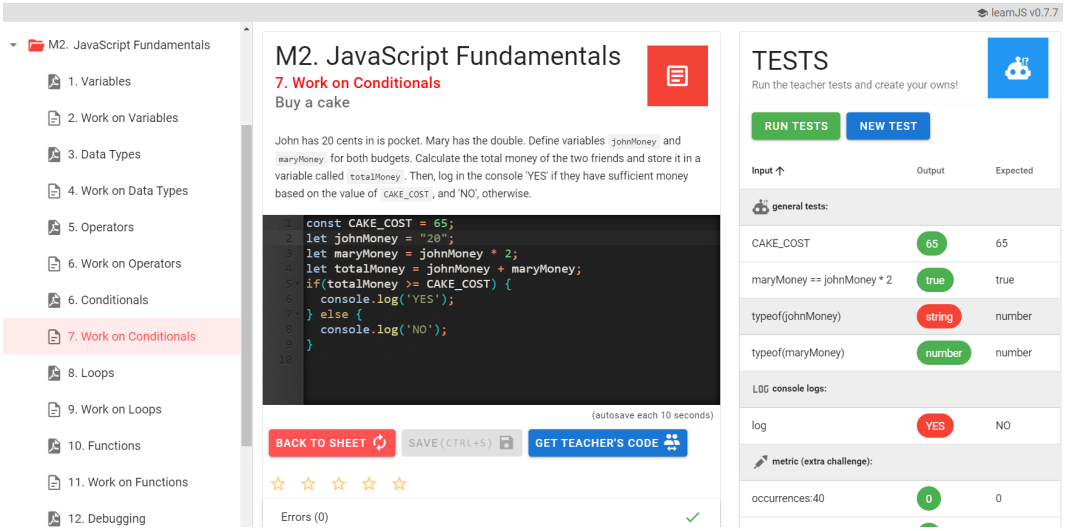

Figure 1 Agni playground User Interface.

Listing 1 Members route.

```
lti.app.get('/members', async (req, res) => {
 // Gets context members
 const members = await lti.NamesAndRoles.getMembers(res.locals.token)
 res.send(members)
})
```

With this approach Agni, could present a leaderboard with the names of students (and respective scores) even before they submitted for the first time.

Despite its popularity, presenting leaderboards also has disadvantages, in particular, it can frustrate all those who recurrently remain in the lower places. Therefore, its use should be restricted, either through the creation of partial rankings or through the use of a boolean custom parameter called **show_leaderboard** which, when set to true, displays the leaderboard, otherwise it inhibits its visualization.

4.2.2 Content – Exercise sheet launching

Agni is a code playground targeted at JavaScript and organized into modules for different topics such as variables, data types, and arrays. Each module has two kinds of resources: expositive, such as videos and PDFs; and evaluative, such as exercise sheets. An exercise sheet contains a set of exercises of different types (e.g. blank sheet, buggy code, quiz).

To configure an exercise sheet in Moodle, the teacher should add an external LTI tool activity (Agni). On the form, press the "select content" button, which will call the `onDeepLinking` callback so that the tool can display a resource selection view. This view with the list of existing exercise sheets did not exist, so it was one of the first challenges to overcome. After selecting the desired sheet, the tool creates a deep linking request message and sends it to the platform, through the `createDeepLinkingMessage` method. Subsequently, when the activity is started in Moodle, the respective exercise sheet is displayed.

Another challenge was controlling the Agni UI, because when launching a specific exercise sheet, nothing prevented the student from selecting the other sheets. For this, a boolean custom parameter called **deactivate_ui** was added. If set to true, the student will only be able to access the associated sheet. Otherwise, they will have access to all the exercise sheets.

4.2.3 Feedback – Scores submission

During the problem solving process, the student's performance score is automatically sent to the Moodle gradebook. In this way, teachers are able to monitor the progress of their students in real time.

Using the Grading Service it is possible to submit grades from Agni to Moodle (with the **submitScore** method), get grades from Moodle (**getScores** method) and even manage Moodle gradebook columns (**getLineItems**, **createLineItem** and **deleteLineItems** methods).

The hardest challenge was to decide at which moment the score should be sent to the platform, since the Agni UI does not have a button to send grades. To overcome this difficulty, it was decided that whenever a student, after submitting his solution for evaluation, leaves the exercise back to the list of exercises, his score is automatically submitted into Moodle gradebook.

4.3 Authorkit

AuthorKit is a programming exercise authoring tool [4]. It is a web application where authors can create programming challenges that can later be retrieved as learning objects by other components of an e-learning environment, such as automated assessment systems. In a strict sense, AuthroKit is not the kind of web application envisioned by the proposed roadmap, since it is not targeted at students. However, by using LTI it can benefit from single-sign-on and use AGS to report data back to the calling system.

As the tool itself does not provide a numeric feedback nor is this feedback relevant in the context of having students and calculating their grade. This was done out of interest for how such a tool could be used in the LTI context, this implementation uses the fact that every grade has a description, and as such, provides the key generated by the tool as a description for a null grade. This key can then be seen through the LMS interface.

AuthorKit is initially set up by a login screen, the objective is to send through an LTI parameter informing that this launch is indeed an LTI one, skipping the login screen and entering the corresponding account from the TC. Afterwards, it is possible to create a programming exercise through the TP content in the TC, resulting in the exercise key as feedback.

5 Conclusion

This paper proposes a roadmap to integrate existing educational web applications into the ecosystem created by a learning management system. To achieve this integration applications must support the Learning Tools Interoperability specification in the role of tool provider. The proposed roadmap was validated with existing applications. This paper reports on the challenges faced to apply it in these concrete cases. As future work it is intended to implement all the services of LTI Advantage in a gamified programming learning environment that is an integral part of an ecosystem to facilitate the teaching of programming and that has been worked on in the context of a European project called FGPE Plus: Learning tools interoperability for gamified programming education (FGPE+)[7].

[7] `https://fgpeplus.usz.edu.pl/`

References

1 Marc Alier, María José Casany, Ariadna Llorens, Jesús Alcober, and Joana d'Arc Prat. Atenea exams, an ims lti application to solve scalability problems: A study case. *Applied Sciences*, 11(1):80, 2020.

2 Claudia-Melania Chituc and Marc Rittberger. Understanding the importance of interoperability standards in the classroom of the future. In *IECON 2019 – 45th Annual Conference of the IEEE Industrial Electronics Society*, volume 1, pages 6801–6806, 2019. doi:10.1109/IECON.2019.8927631.

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

4 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.

5 Ricardo Queirós. Learnjs – a javascript learning playground (short paper). In *7th Symposium on Languages, Applications and Technologies (SLATE 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

6 Ricardo Queirós and José Paulo Leal. Ensemble – an e-learning framework. *J. Univers. Comput. Sci.*, 19(14):2127–2149, 2013. doi:10.3217/jucs-019-14-2127.

7 Ricardo Queirós and José Paulo Leal. Fostering students-driven learning of computer programming with an ensemble of e-learning tools. In Álvaro Rocha, Hojjat Adeli, Luís Paulo Reis, and Sandra Costanzo, editors, *Trends and Advances in Information Systems and Technologies – Volume 2 [WorldCIST'18, Naples, Italy, March 27-29, 2018]*, volume 746 of *Advances in Intelligent Systems and Computing*, pages 289–298. Springer, 2018. doi:10.1007/978-3-319-77712-2_28.

8 Ricardo Queirós and José Paulo Leal. Elearning frameworks: A survey. In *INTED2010 Proceedings*, 4th International Technology, Education and Development Conference, pages 1345–1354. IATED, 8-10 march, 2010 2010.

9 G. Tuparov and D. Tuparova. Approaches for integration of educational computer games in e-learning environments. In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 0772–0776, 2018. doi:10.23919/MIPRO.2018.8400143.