ULAM Headless Course Format

ver. e6539fe

Mateusz Wojczal. November 2021

Contents

ULAM Headless Course Format

The Abstract

Current e-learning does have a huge technological dept and do not responding to market needs as fast as other segments. The main reason is the obsolete formats like SCORM that are widely used which does not separate data layer from presentation one. There is a need from market of existence of better formats.

The Introduction

Current e-learning formats does not separate data from presentation layers. Current e-learning content are not portable and are not designed to age well.

With separation of layers the content can be displayed in modern way everything some

new devices is used. If SCORM courses where designed in this fashion back in 2000s it would be straigh forward to convert them to any devices, like mobile phones, smartwatches, smart tvs, etc. Because of wrong design decision we're stuck with this format and obsolete courses.

Back in the days when Advanced Distributed Learning was creating SCORM adapting older AICC HACP desktop format most of the personal computers used the same browser, on the same operation system with common 1024x768 pixel resolution. If there were variation to this statement they were minimal. Browser were not able to do much more then to show server response in HTML format after client request. Everything showed in browser window was rendered by server and even if there was separation of layers it happened only on server side.

Organizations that are working on e-learning standards are responding to market needs very slowly. Their latest specification `cmi5`, which does solve many of the issues, is already 6 years old and not commonly adapted - the most popular format SCORM 2004 4th Edition was published in 2009.

The headless approach seems to be solving all of the issues that modern e-learning and LMSes do have. The separation of content and it's players allows to create courses that works well on any device and do age well. Course designed in this favour most likely will be able to be played on device not yet used.

History of e-learning formats

The most popular e-learning formats are created and managed by the Advanced Distributed Learning (ADL) Initiative from the Office of the United States Secretary of Defense.

Before e-learning was used in the web browser enviroment there was AICC's format created in 1993. First widely used format was AICC HACP released in 1998 which later evolved into SCORM 1.0 that was released in year 2000.

SCORM which is an abbreviation of Sharable Content Object Reference Model since this day is the most popular e-learning package standard. Since version 1.0 to latest SCORM 2004 4th Edition this format is a collection of standards and specifications for web-based e-learning. The format itself describes communications between client side content and a host system and how to package whole course into ZIP files that are called "Package Interchange Format."[1]. Latter is a ZIP package that contains HTML files and XML manifest.

Since SCORM introduced many issues The Experience API, also known as Tin Can API or xAPI was released and later cmi5 format that provides a set of rules intended to achieve interoperability in a traditional Learning Management System environment.

xAPI specification removes content for it description, and allows the content to send "statements" based around [actor] [verb] [object], or "I – did – this" to a Learning Record Store (LRS) which can be part of Learning Management System but can live on their own or as part of another system.

The table below [2] summarizes the comparison of each standard:

Format	Release Date	Pages	Widely Used	Run-Time	Packaging	Me
AICC HACP	Feb 1998	337	Yes	Yes	Yes	

[1] Technical Specification 4th Ed.. SCORM. Retrieved 2017-05-22.
[2] A timeline and description of the eLearning standards.. SCORM

Format	Release Date	Pages	Widely Used	Run-Time	Packaging	Me
SCORM 1.0	Jan 2000	219	No	Yes	Yes	
SCORM 1.1	Jan 2001	233	No	Yes	Yes	
SCORM 1.2	Oct 2001	524	Yes	Yes	Yes	
SCORM 2004 "1st Edition"	Jan 2004	1,027	No	Yes	Yes	
SCORM 2004 2nd Edition	Jul 2004	1,219	Yes	Yes	Yes	
SCORM 2004 3rd Edition	Oct 2006	1137	Yes	Yes	Yes	
SCORM 2004 4th Edition	Mar 2009	1162	Yes	Yes	Yes	
IMS Common Cartridge	Oct 2008	135	No	No	Yes	
IMS LTI	May 2010	25	In Academic LMSs	Yes	No	
The Experience API (xAPI)	April 26, 2013	85	Not Yet	Yes	Partial	
cmi5 (a companion to xAPI)	June 1, 2016	48	Not Yet	Yes	Yes	

What is Learning Management System - LMS

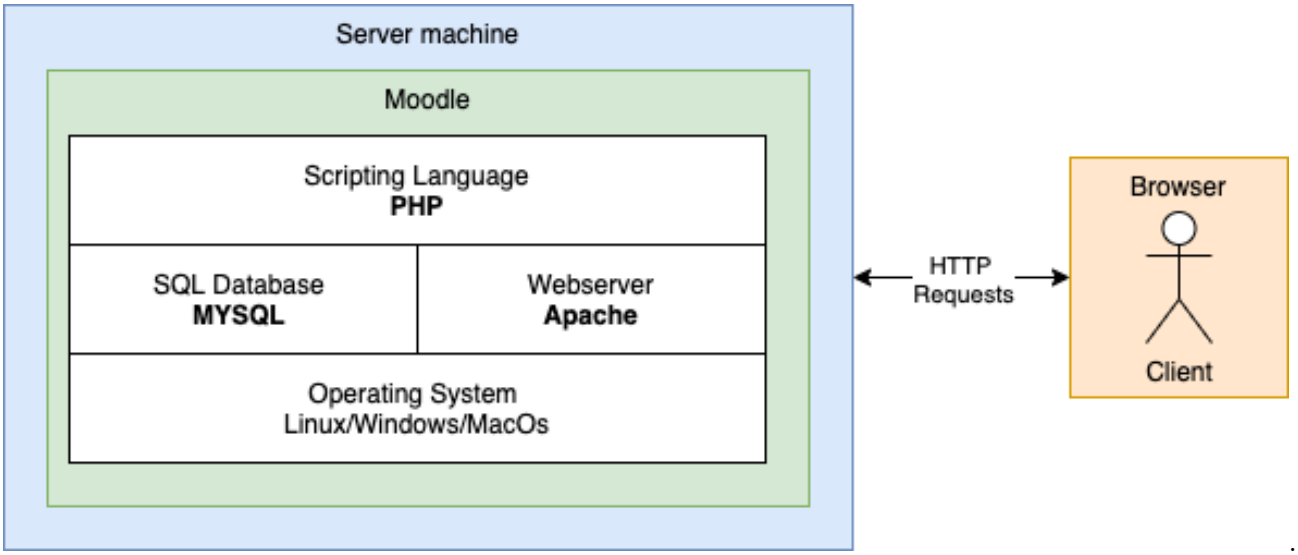
Web accessible application that takes care of administration, documentation, tracking, reporting, automation, and delivery of educational courses, training programs, or learning and development programs is called Learning Management System. LMS systems are kind of software that manage e-Learning.

The most popular LMS is Moodle, released on 20 August 2002 because it's available for free as open course software, distributed under the GNU General Public License.

Moodle is program written in PHP that is being served by machine that use PHP. That means that all of the actions for administrators, course creators, students and any other roles does require to connect to machine (server) that serves Moodle. This is a monolith architecture, which means that all moodle components are PHP based working on one machine that parses moodle source code every time there is a request from the browser. Components of the program are interconnected and interdependent in a tightly-coupled architecture.

Most other popular LMS works very similar, as they monolith architecture is the most popular among the LMS

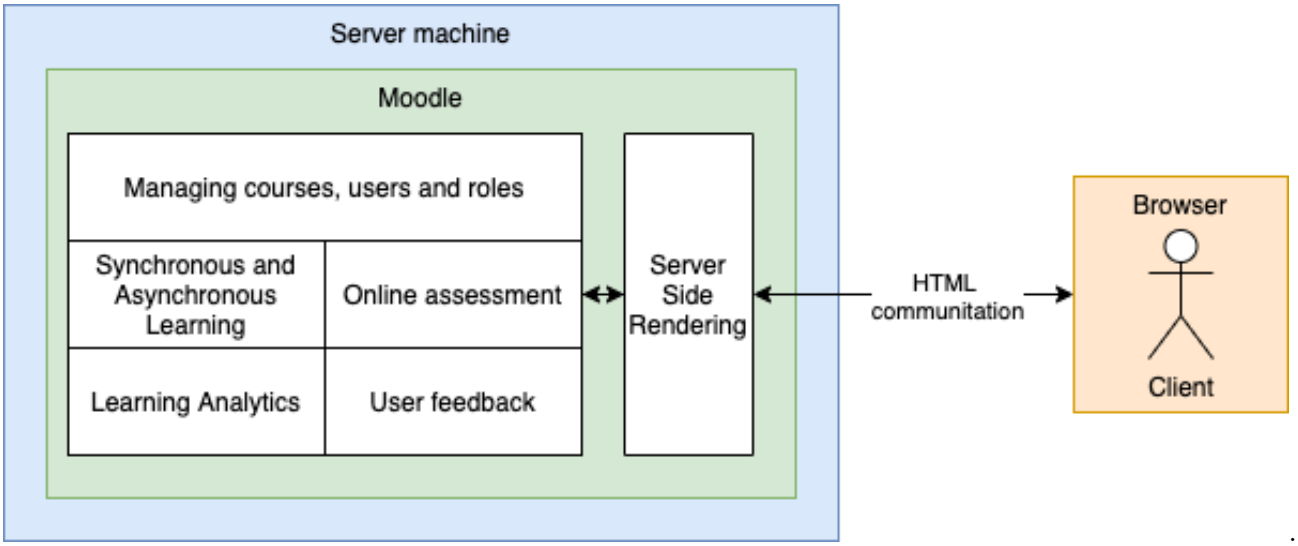
Monolith Architecture It the diagram below there is Moodle monolith architecture



All the LMS Features that includes

- Managing courses, users and roles
- Online assessment
- User feedback
- Synchronous and Asynchronous Learning
- Learning Analytics

are handled directly from the server, the response is prepared before being sent in HTML format by PHP preprocessor, the client gets the HTML already rendered document.



.

- aicc 1993, Scorm 1999, scorm 2004, xapi (tincan) 2013, cmi5 (cmi5: xAPI with Rules)
- cmi5 defines how the LMS and the content will communicate using the xAPI Learning Record Store (LRS).
- https://scorm.com/scorm-explained/business-of-scorm/scorm-versions/
- https://www.slideshare.net/Nine_Lanterns/interoperability-lti-and-experience-api-formerly-tincan
- definition
- how does it work monolith struture
- diagram
- https://scorm.com/what-is-an-lrs-learning-record-store/?utm_source=google&utm_medium=natural_search

How does publishing course look like

- scorm is closed format

Bad parts

- no offine (needs a server)
- same domain
- no speparationn betrween layers
- when designed there was no mobile devices
- Assignable Unit (AU) must have launchURL that basically is course starting point
- why cmi5 is not enough (http://aicc.github.io/CMI-5_Spec_Current/flows/lms-flow.html)

- why xapi is good

What is headless

PWA, Serverless, JSON (vs XML)

Ulam Format

Why separaion is good https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

Universal Learning Asynchronous Model

Good parts from cmi5 & xapi

- xapi verbs
- au
- json > xml

HEadless

Players

Import/Export

Offline

Guideline #1: A clear new important technical contribution should have been articulated by the time the reader finishes page 3 (i.e., a quarter of the way through the paper).

Guideline #2: Every section of the paper should tell a story. (Don't, however, fall into the common trap of telling the entire story of how you arrived at your results. Just tell the story of the results themselves.) The story should be linear, keeping the reader engaged at every step and looking forward to the next step. There should be no significant interruptions -- those can go in the Appendix; see below.

Aside from these guidelines, which apply to every paper, the structure of the body varies a lot depending on content. Important components are:

- **Running Example:** When possible, use a running example throughout the paper. It can be introduced either as a subsection at the end of the Introduction, or its own Section 2 or 3 (depending on Related Work).
- **Preliminaries:** This section, which follows the Introduction and possibly Related Work and/or Running Example, sets up notation and terminology that is not part of the technical contribution. One important function of this section is to delineate material that's not original but is needed for the paper. Be concise -- remember Guideline #1.
- **Content:** The meat of the paper includes algorithms, system descriptions, new language constructs, analyses, etc. Whenever possible use a "top-down" description: readers should be able to see where the material is going, and they should be able to skip ahead and still get the idea.

The Conclusions

In general a short summarizing paragraph will do, and under no circumstances should the paragraph simply repeat material from the Abstract or Introduction. In some cases it's possible

to now make the original claims more concrete, e.g., by referring to quantitative performance results.

Future Work

This material is important -- part of the value of a paper is showing how the work sets new research directions. I like bullet lists here. (Actually I like them in general.) A couple of things to keep in mind:

- If you're actively engaged in follow-up work, say so. E.g.: "We are currently extending the algorithm to... blah blah, and preliminary results are encouraging." This statement serves to mark your territory.
- Conversely, be aware that some researchers look to Future Work sections for research topics. My opinion is that there's nothing wrong with that -- consider it a compliment.

The Acknowledgements

Don't forget them or you'll have people with hurt feelings. Acknowledge anyone who contributed in any way: through discussions, feedback on drafts, implementation, etc. If in doubt about whether to include someone, include them.

Citations

Spend the effort to make all citations complete and consistent. Do *not* just copy random inconsistent BibTex (or other) entries from the web and call it a day. Check over your final bibliography carefully and make sure every entry looks right.

Appendices

Appendices should contain detailed proofs and algorithms only. Appendices can be crucial for overlength papers, but are still useful otherwise. Think of appendices as random-access substantiation of underlying gory details. As a rule of thumb:

- Appendices should not contain any material necessary for understanding the contributions of the paper.
- Appendices should contain all material that most readers would not be interested in.

Grammar and Small-Scale Presentation Issues In general everyone writing papers is strongly encouraged to read the short and very useful The Elements of Style by Strunk and White. Here's a random list of pet peeves.

- Just like a program, all "variables" (terminology and notation) in the paper should be defined before being used, and should be defined only once. (Exception: Sometimes after a long hiatus it's useful to remind the reader of a definition.) Global definitions should be grouped into the Preliminaries section; other definitions should be given just before their first use.
- Do not use "etc." unless the remaining items are completely obvious.
  - Acceptable: *We shall number the phases 1, 3, 5, 7, etc.*
  - Unacceptable: *We measure performance factors such as volatility, scalability, etc.*

  (**Exercise:** The above rule is violated at least once in this document. Find the violations.)

- Never say "for various reasons". (Example: *We decided not to consider the alternative, for various reasons.*) Tell the reader the reasons!
- Avoid nonreferential use of "this", "that", "these", "it", and so on (Ullman pet peeve). Requiring explicit identification of what "this" refers to enforces clarity of writing. Here is a typical example of nonreferential "this": *Our experiments test several different environments and the algorithm does well in some but not all of them. This is important because ...*

  (**Exercise:** The above rule is violated at least once in this document. Find the violations.)
- Italics are for definitions or quotes, not for emphasis (Gries pet peeve). Your writing should be constructed such that context alone provides sufficient emphasis.

  (**Exercise:** The above rule is violated at least once in this document. Find the violations.)
- People frequently use "which" versus "that" incorrectly. "That" is defining; "which" is nondefining. Examples of correct use:
  - *The algorithms that are easy to implement all run in linear time.*
  - *The algorithms, which are easy to implement, all run in linear time.*

Mechanics

- Always run a spelling checker on your final paper, no excuses.
- For drafts and technical reports use 11 point font, generous spacing, 1" margins, and single-column format. There's no need to torture your casual readers with the tiny fonts and tight spacing used in conference proceedings these days.
- In drafts and final camera-ready, fonts in figures should be approximately the same font size as used for the text in the body of the paper.
- Tables, figures, graphs, and algorithms should always be placed on the top of a page or column, not in the body of the text unless it is very small and fits into the flow of the paper.
- Every table, figure, graph, or algorithm should appear on the same page as its first reference, or on the following page (LaTex willing...).
- Before final submission or publication of your paper, *print it* once and take a look -- you might be quite surprised how different it looks on paper from how it looked on your screen (if you even bothered to look at it after you ran Latex the last time...).

Versions and Distribution

- Many papers have a submitted (and later published) conference version, along with a "full paper" technical report on the web. It's important to manage versions carefully, both in content and proliferation. My recommendation is, whenever possible, for the full paper to consist of simply the conference version plus appendices. The full paper should be the only public one aside from conference proceedings, it should be coordinated with latest (final) conference version, and modifications to the full paper should always overwrite all publicly accessible previous versions of it.
- I believe in putting papers on the web the minute they're finished. They should be dated and can be referenced as technical reports -- it's not necessary to have an actual technical report number. Never, ever put up a paper with a conference copyright notice when it's only been submitted, and never, ever reference a paper as "submitted to conference X." You're only asking for embarrassment when the paper is finally published in conference Y a year or two later.