ULAM Headless Course Format

ver. 6ea643e

Mateusz Wojczal. November 2021

Contents

Glosarry

lorem ipsum

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.

Evolution of e-learning

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 environment 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.

[1]Technical Specification 4th Ed.. SCORM. Retrieved 2017-05-22.

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

Format	Released	Pages	Widely Used	Run-Time	Pack-aging	Meta-data	Sequen-cing	Works Cross Domain
AICC HACP	Feb 1998	337	Yes	Yes	Yes	No	No	Yes
SCORM 1.0	Jan 2000	219	No	Yes	Yes	Yes	No	No
SCORM 1.1	Jan 2001	233	No	Yes	Yes	Yes	No	No
SCORM 1.2	Oct 2001	524	Yes	Yes	Yes	Yes	No	No
SCORM 2004 “1st Edition”	Jan 2004	1,027	No	Yes	Yes	Yes	Yes	No
SCORM 2004 2nd Edition	Jul 2004	1,219	Yes	Yes	Yes	Yes	Yes	No
SCORM 2004 3rd Edition	Oct 2006	1137	Yes	Yes	Yes	Yes	Yes	No
SCORM 2004 4th Edition	Mar 2009	1162	Yes	Yes	Yes	Yes	Yes	No
IMS Common Cartridge	Oct 2008	135	No	No	Yes	Yes	No	Yes
IMS LTI	May 2010	25	In Academic LMSs	Yes	No	No	No	Yes
The Experience API (xAPI)	April 26, 2013	85	Not Yet	Yes	Partial	No	No	Yes
cmi5 (a companion to xAPI)	June 1, 2016	48	Not Yet	Yes	Yes	No	No	Yes

What is Learning Management System - LMS

What is Learning Management System - LMS

The most popular LMS is Moodle [3], 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

LMS Monolith Architecture

It the diagram below there is Moodle monolith architecture

[2] A timeline and description of the eLearning standards.. SCORM
[3] Moodle - Open-source learning platform | Moodle.org

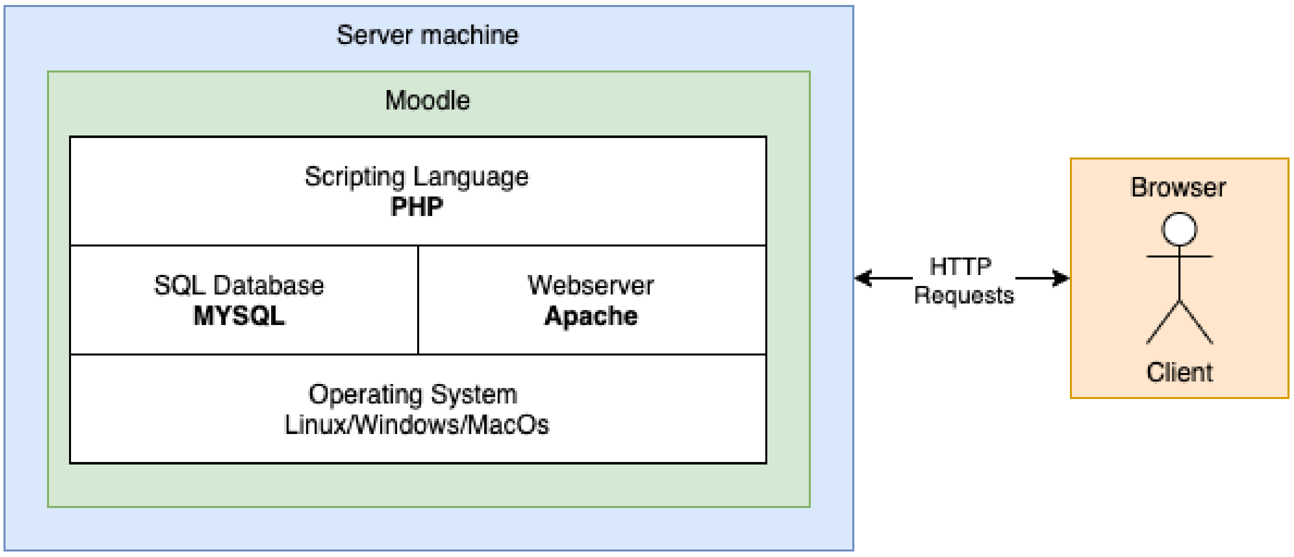

Figure 1: LMS monolith architecture. Moodle technical architecture

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.

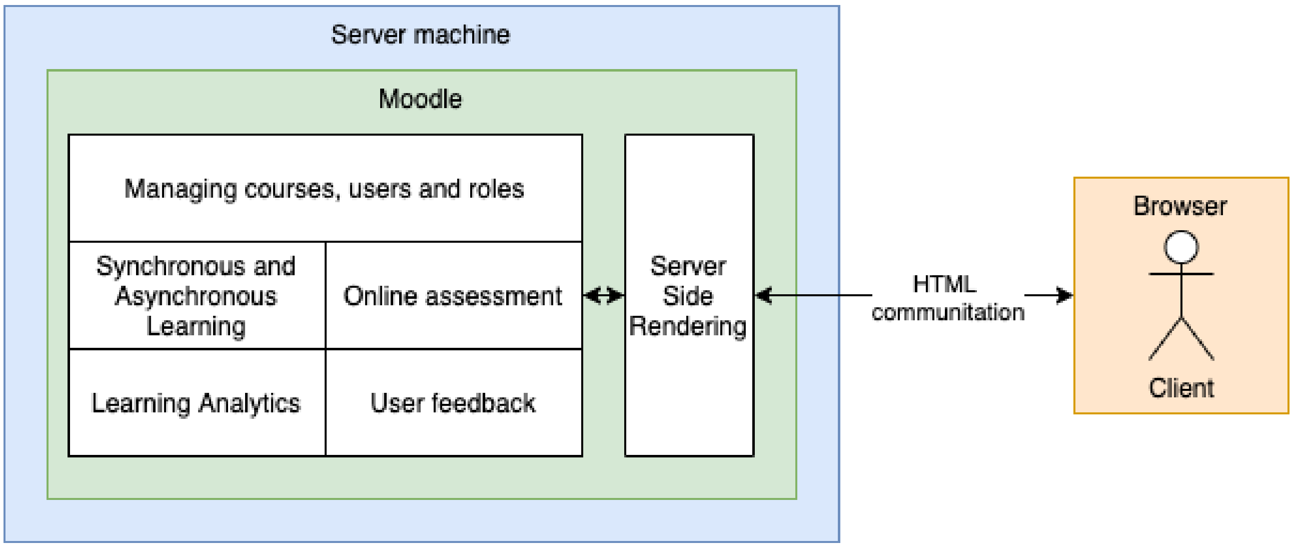

Figure 2: LMS monolith architecture. Moodle functional architecture

All the above means that Moodle and dedicated server is required all the time for all e-learning activities.

Process of publishing the course

Standard way of creating and publishing SCORM compliant course is to follow the steps

1. Creating of a course in an e-Learning authoring tool (like Adobe Captivate [4]) or in from the LMS environment.
2. Course is published as a SCORM package, a ZIP file
3. SCORM package is being uploaded with LMS upload form and prepared to be published
4. LMS publish the course to the students. All results of activities are stored in the LMS

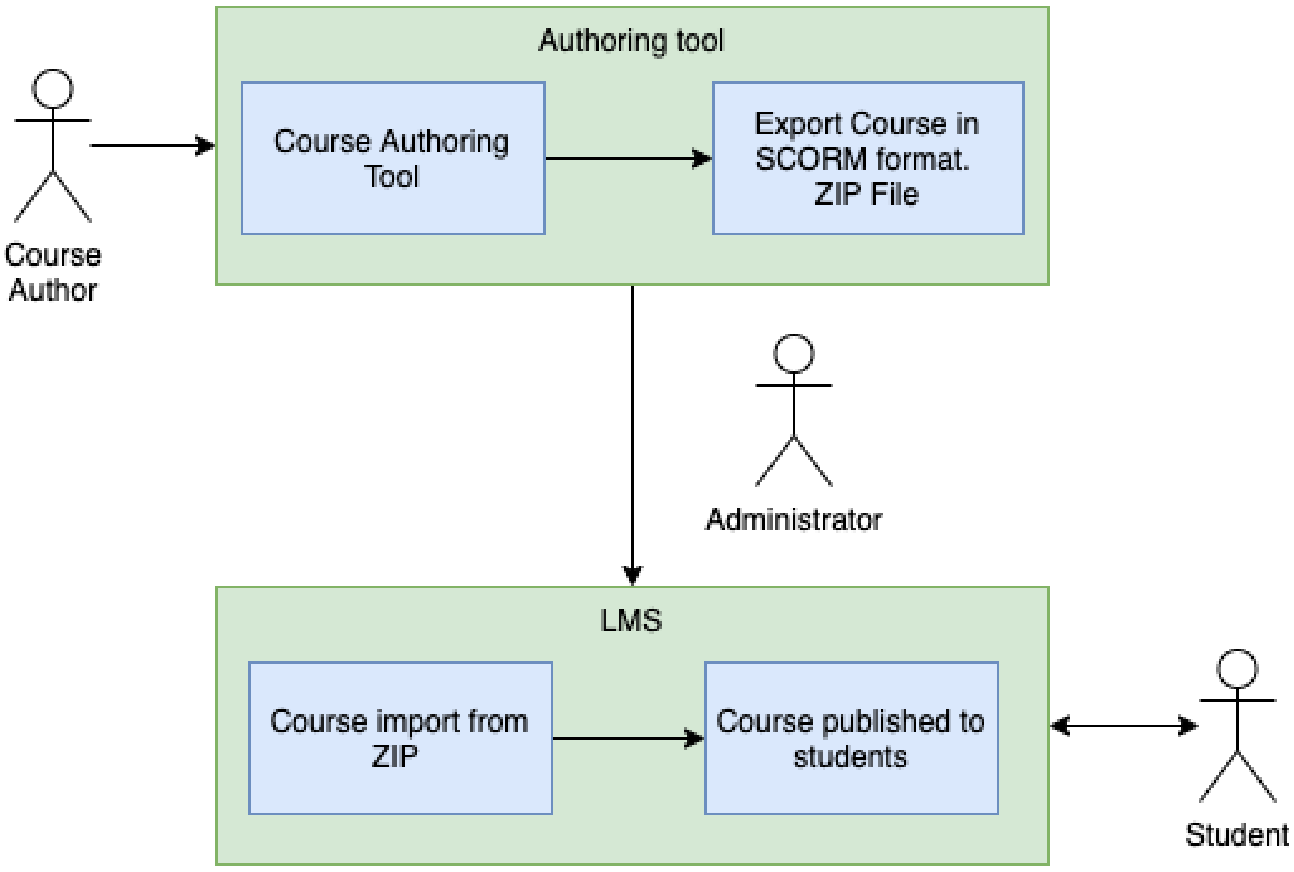

Figure 3: Process of publishing the course

The process above is one direction - which means that SCORM package is closed format, once published it cannot be changed. In order to make any changes, event amending simple typo, the whole process must be repeated - course needs to be changed in authoring tool, then uploaded, etc.

Introduction of Experience API (xAPI) and related technologies

One of the limitation of SCORM that decided about introducing extended formats was capability to track and trace activities from students only within the same LMS. That means that the course and LMS are inseparable.

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)

[4]Adobe Captivate

which can be part of Learning Management System but can live on their own or as part of another system. This was the first step for **Separation of concerns** in e-learning.

Learning Record Store LRS

A Learning Record Store is an external to course application that receives and sends data in JSON format from and to course runtime - it is an essential component in Experience API process flow. What's a big difference is that the specification does not tell how does course is being played (course runtime), it just defines that runtime does communicate with the interface (LRS) though xAPI Statements. The statements are open to extend, each implementation can introduce their own statements.

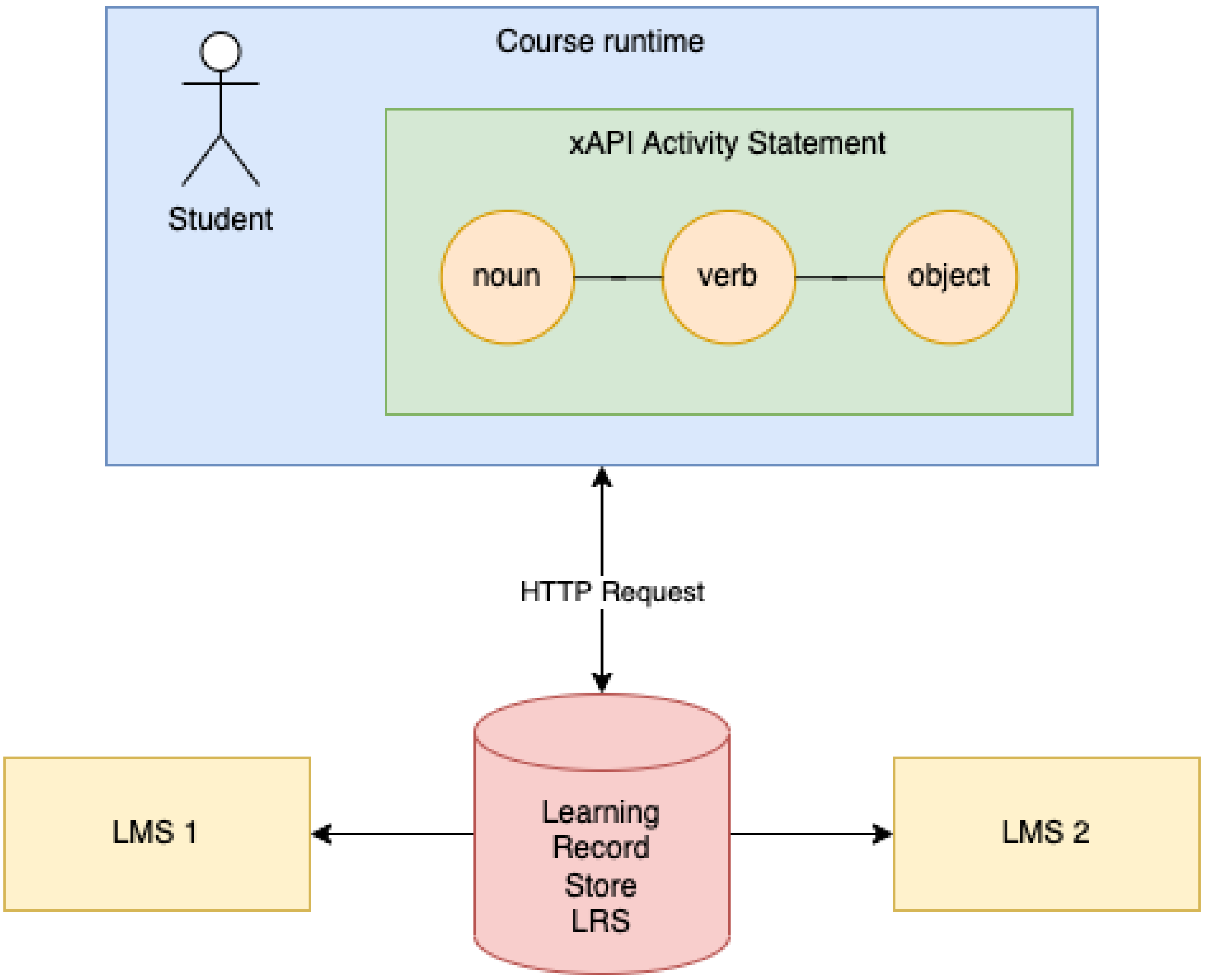

Figure 4: Experience API (xAPI) process flow with Learning Record Store (LRS)

cmi5 Specification

> cmi5 is a "profile" for using the xAPI specification with traditional learning management (LMS) systems. [5]

The Specification of cmi5 is a set of rules providing all the capabilities of SCORM and xAPI. It is similar to SCORM in a way that it also contains XML file manifest, yet it does introduce the Assignable Unit (AU) - separately launchable learning content presentation. The

[5]cmi5: Technical 101 Terminology.

AU is the unit of tracking and management. The AU collects data on the learner and sends it to the LMS.

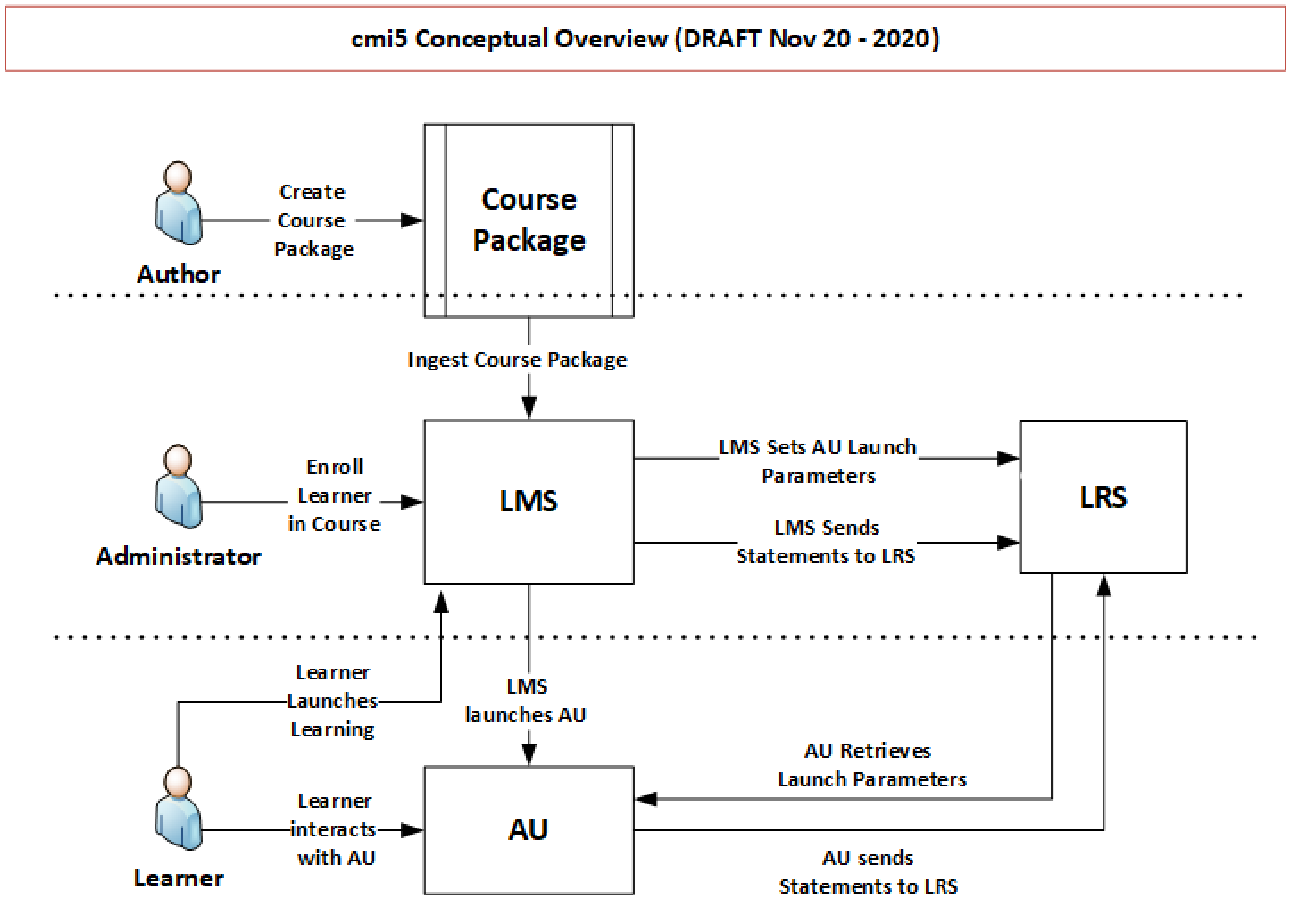
cmi5 Conceptual Overview (DRAFT Nov 20 - 2020)
Author
Create Course Package
Course Package
Ingest Course Package
Administrator
Enroll Learner in Course
LMS
LMS Sets AU Launch Parameters
LMS Sends Statements to LRS
LRS
Learner Launches Learning
LMS launches AU
AU Retrieves Launch Parameters
Learner
Learner interacts with AU
AU
AU sends Statements to LRS

Figure 5: Conceptual Overview of cmi5 [6]

cmi5 also requires Determine Launch Mode, defines some xAPI statements that must appear in correct order. The course itself describes `moveOn` rules

> Setting that captures how a learner moves through the AUs/Blocks of a Course. Determines what is required for an AU to be considered “Satisfied”. Blocks are “Satisfied” when all of their direct descendent AUs or Blocks are “Satisfied”. The Course is “Satisfied” when all of its direct descendent AUs or Blocks are “Satisfied”. [7]

[6]Conceptual Overview of cmi5
[7]cmi5: Technical 101 Terminology.

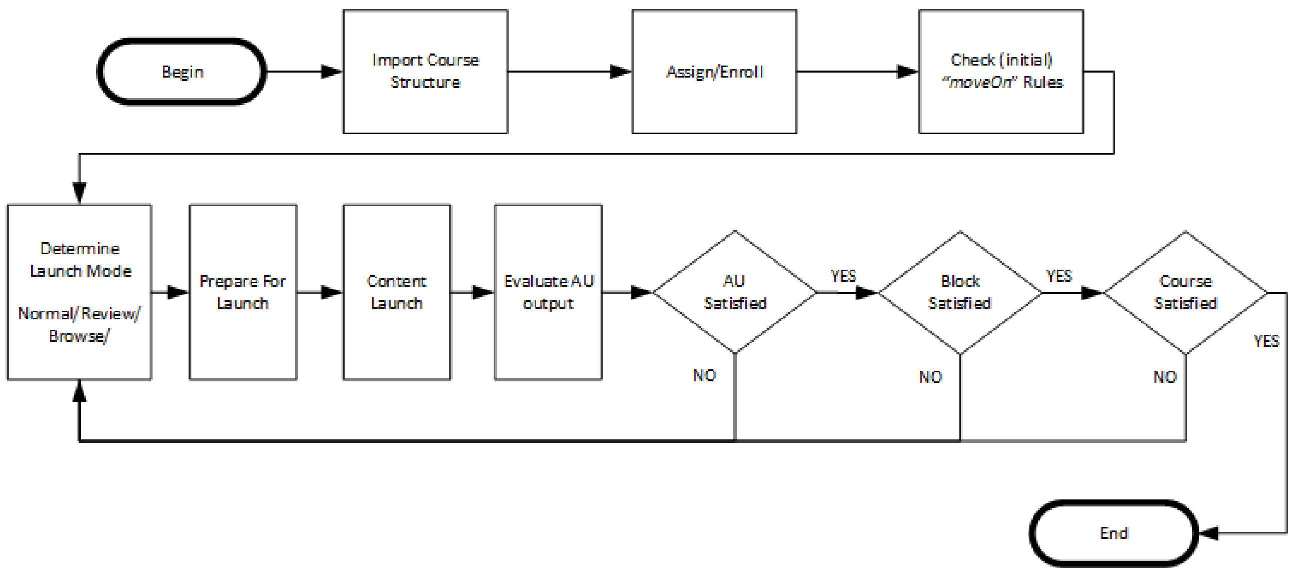

Figure 6: cmi5 Implementation Flow for an LMS [8]

Limitations parts of current standards

Regardless of all the efforts for evolution of e-learning standard there are still limitations:

- SCORM is limited by design, there is no way to improve it implementing **Separation of concerns** design pattern.
- Even with latest standard `cmi5` the **Separation of concerns** is not complete.
- Assignable Unit defines only entry URL for the content, but it does not define the content structure in any way.
- Specification of Assignable Unit (AU) require to have `launchURL` that basically is course starting point. There is no way to extend this to replace `launchURL` with the content itself.
- There is no separation of layers in content delivery. Presentation, data and logic layers are inseparable.
- Courses cannot be played offline as server is required all the time.
- Even though cmi5 provides **Mobile app launch support** functionality there is no specification for that, it is possible yet not defined.

[8]cmi5 Implementation Flow for an LMS

Separation of concerns

The main motivation of introducing new e-learning format is to allow to separate all of the e-learning components into independent elements and follow the **separation of concerns (SoC)** computer science design principle.

> A design principle for breaking down an application into modules, layers, and encapsulations, the roles of which are independent of one another. [9]

Headless

Regular websites and web application works in the way that their own back-end (server side component) and front-end (graphical user interface). Each piece use the same code base and communicate directly on the server machine with each other, making the website as a whole.

Headless web application is an implementation of **separation of concerns (SoC)** design principle of the front-end as stand-alone piece of software, and the back-end that doesn't know anything about way the data that is served will be presented. All the communication happens through API as the bridge between both parts. All parts works separately technically (placed on separate servers) and functionally.

Headless LMS.

In opposition to Monolith LMS Architecture headless LMS is build upon API as a main component. All other components does communicate though this interface. In most cases API and Database are the only parts that require dedicated server.

In Monolithic architecture, frontend component, a presentation layers, requires specific know-how, example: you are obliged to use Moodle template system called Mustache In Headless architecture, frontend is framework agnostic. You can use any frontend framework you want. Furthermore you can use few at once, like React on one domain, vue for course details and Angular for admin panel on other domains.

Headless LMS Architecture

- **separation of concerns (SoC)** design principle, separate all of the components.
- only API require server
- admin panel is serverless
- user app is serverless
- application and admin panel are easy to replicate
- other view layouts (eg native mobile application) are easy to add without changes to other layers
- to implement courses for students there is no need to specialization knowledge.

[9] Blockchain Networks: Token Design and Management Overview NISTIR 8301. National Institute of Standards and Technology

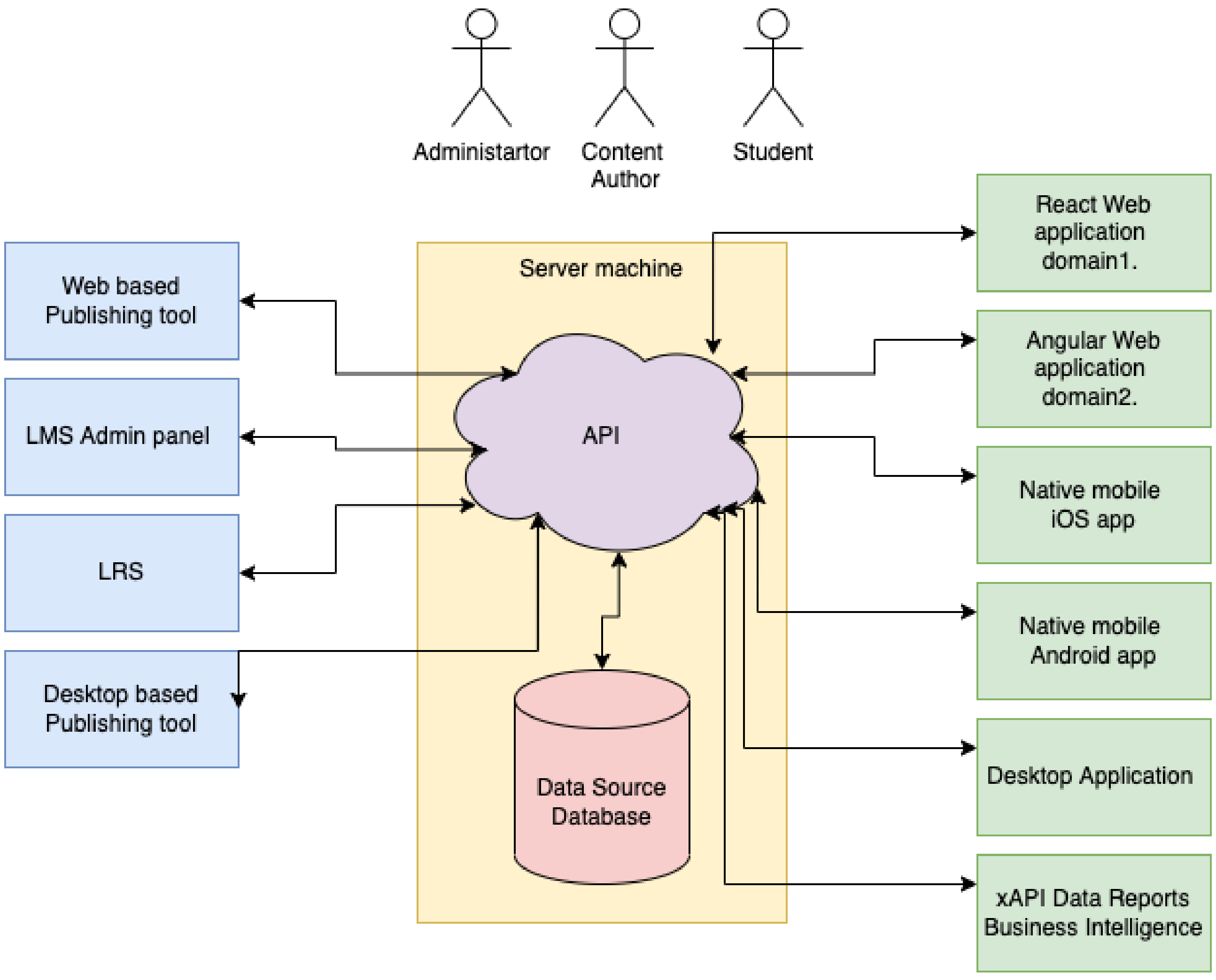
Administartor
Content Author
Student
Web based Publishing tool
LMS Admin panel
LRS
Desktop based Publishing tool
Server machine
API
Data Source Database
React Web application domain1.
Angular Web application domain2.
Native mobile iOS app
Native mobile Android app
Desktop Application
xAPI Data Reports Business Intelligence

Figure 7: Architecture of headless LMS

A Headless LMS is a “Course Repository” that makes content accessible to any platform via an API. We provide blocks to build one, yet you’re free to change those od use your own. Unlike a traditional LMS such as Moodle, a Headless LMS does not dictate where or how content is shown. Also you don’t need any additional software to show a course - it’s just a matter of API communication

A Headless LMS enables teams to deliver omnichannel experiences at scale, globally, without restrictions like templates, devices, or pre-defined technologies. A Headless LMS allows brands and companies to engage with users on any device and format. White label was never easier then with headless. A Headless LMS fits into any preferred tech stack or framework, including most popular ones like React, Angular, and Vue.

Limitations parts of current standards that Headless can improve

- Implementing **Separation of concerns** design pattern is complete.
- New headless formats can defined content structure.
- Presentation, data and logic layers are separable.
- Courses cannot can be played offline as server is not required all the time.
- Other presentation layouts are easy to add without changes to other layers so **Mobile app launch support** functionality is easily achievable.

ULAM Format.

Because of the existing limitation new format **ULAM Universal Learning Asynchronous Model** for course content is needed.

The main motivation for the above are the following statements.

Separation content layer from presentation layer.

All of the most popular e-learning format, mostly SCORM mix view, data and logic layer into one as inseparable. New format should separate all the layers to follow separation of concerns (SoC) design principle.

Use of JSON format instead of XML.

When SCORM was published for the first time JSON format did not exists. During evolution of standards ADL use JSON for `xAPI`. Comparing to XML JSON is light-weight and has an easy to parse data format requiring no additional code for parsing. It is more popular then XML [10]

Easy implementation.

Since SCORM first version implementing the format wasn't trivial for implementation. Until this day SCORM is the most popular, and latest format cmi5 has problems with adaptations [11]. New format must be easy to implement.

Format that age well.

SCORM does not age well. There are many courses that were created in 2010s using obsolete Flash technologies, which are useless now because none of the browsers still support Flash Player. New format must be able to prevent this kind of issues and **Separation of concerns** design pattern is a solution, because it does separate course pure data from presentation.

Simple but open for extension.

New format must be open for extensions which implies that it is possible to create new features without changing old ones.

Standalone.

New format package must be able to be played as course without any external services.

Headless.

The course itself does not know how it will be displayed for a student.

Using well design standards, reject obsolete ones.

From SCORM

- Content packaging as ZIP file
- Importing from ZIP
- Exporting as ZIP
- Sequencing

From xAPI

- usage of JSON Format

[10]Google Trends XML vs JSON
[11]An exciting time to watch xAPI and cmi5 adoption numbers

- xAPI Statements
- Learning Record Store

From cmi5

- Launch Method with `moveOn` Rules
- Manifest with blocks and Assignable Units (AU)
- 9 xAPI Verbs (Launched, Initialized, Completed, Passed, Failed, Abandoned, Waived, Terminated, Satisfied)

Definition

The course is packed in ZIP file that contains `content.json` manifest in main folder.

The course itself consists of at least one lessons which consists of at least topic.

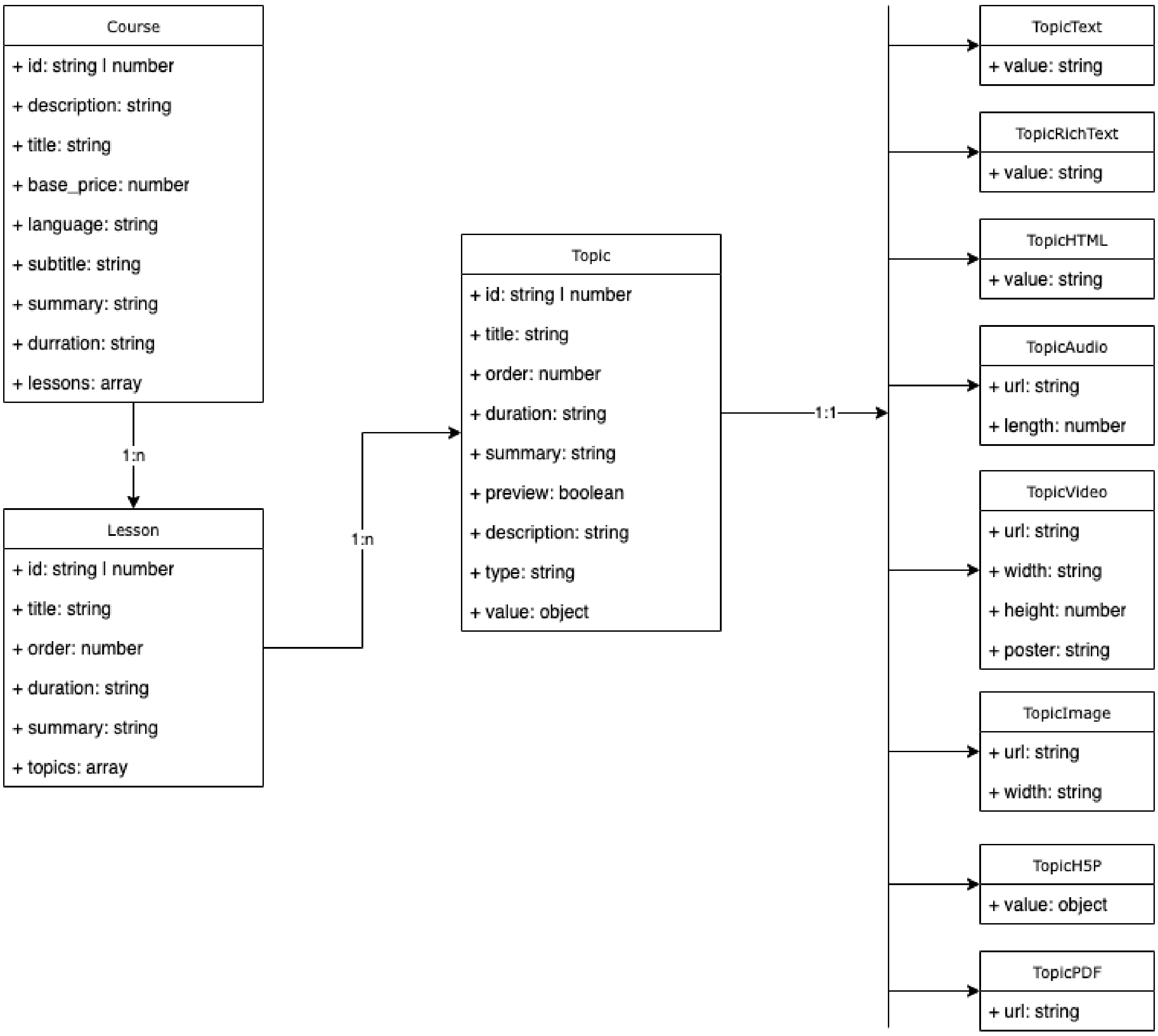

Figure 8: Structure of `ulam` course manifest

The manifest contains the following:

General description of course attributes

Course is defined by

Attribute name	type	description	required
id	number | string	Unique ID of the course. Can be used to identify during import process whether this is a new course or update	no
description	string	Description of the course	no
title	string	Title of the course	yes
base_price	number	Base price of the course. Value 0 means that it's free	no
lessons	array	List of lessons	yes
language	string	Unique ID of the language of the course. Prefered in ISO 639-1 format	no
subtitle	string	Subtitle of the course	no
summary	string	Summary	no
duration	string	How long does the course take. Prefered in ISO 8601 duration format	no

Except of listed attributes the course can be described by unlimited additional fields.

General description of lesson attributes

Attribute name	type	description	required
id	number | string	Unique ID of the course. Can be used to identify during import process whether this is a new lesson or update	no
title	string	Title of the lesson	yes
order	number	Sorting order	no
duration	string	How long does the lesson take. Prefered in ISO 8601 duration format	no
summary	string	Summary of the lesson	no
topis	array	List of Topics	yes

Except of listed attributes the lesson can be described by unlimited additional fields.

General description of topic attributes

Attribute name	type	description	required
id	number | string	Unique ID of the course. Can be used to identify during import process whether this is a new topic or update	no
title	string	Title of the topic	yes
order	number	Sorting order	no
duration	string	How long does the lesson take. Prefered in ISO 8601 duration format	no
summary	string	Summary of the lesson	no

Attribute name	type	description	required
preview	boolean	Can user preview this Assignable Unit without launching the course	no
description	string	Description of the lesson	no
type	string	Type of Topic	yes
value	object	Value of the Topic depending on the Type	yes

General description of Topic types

Topic is describes by type and value. In the first version of the format there are following Topic Types.

Topic type attribute is in string format with namespace, with default namespace `ulam`.

Topic type is an abstract type and polymorphic. This means that has different meaning based on the `type` value.

If the value of the Topic contains references (url) to assets (audio, video, images, etc), they must be placed with relative paths to main `content.json` file.

Text

- Namespace: `ulam\text`
- Value: Unformatted text with newlines

RichText

- Namespace: `ulam\richtext`
- Value: Formatted with Markdown text with newlines

HTMLText

- Namespace: `ulam\html`
- Value: Formatted with HTML text

Audio

- Namespace: `ulam\audio`
- Value: Reference to audio file

Video

- Namespace: `ulam\video`
- Value: Reference to Video file

PDF

- Namespace: `ulam\pdf`
- Value: Reference to PDF file

Image

- Namespace: `ulam\image`
- Value: Reference to Image file

H5P

- Namespace: `ulam\h5p`
- Value: Reference to H5P content

Validating

The manifest must be validated against json schema file. The latest schema is always available in github repository[12].

Extending

Except of listed attributes the manifest can be extended by unlimited additional fields.

For example if there is need to describe course by video path or image it's a matter of adding new fields to the course attributes.

Example

More examples are available in github repository[13].

Packaging.

lorem ipsum

Import

lorem ipsum

Export

lorem ipsum

Example

Implementation

Frontend agnostinc (no Scorm like object)

Content Types

`topic_type` is like `xAPI` word, it can be anything, the standard doesn't specify this.

Content Players

lorem ipsum

[12]Ulam Headless Format Github repository.
[13]Ulam Headless Format Github repository.

comparison with `cmi5`

	cmi5	ulam
use xAPI	yes	yes
Manifest format	xml	json
Defined course type structure	no	yes
Separation of concerns in course data	no	yes
Connection to LRS	required	not required
Mobile friendly	no (only tracking)	yes
Run-Time required	yes	no
Content Package format	yes	yes
Definition of Course launch	yes	yes, same as cmi5
Client Agnostic	yes	yes
Distributed Content	yes	no
Advanced activity tracking	yes	yes
Serverless	no	yes

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

Thank you XXX who edited this article.

Name ulam ….