GAME CHANGER FOR ONLINE LEARNING DRIVEN BY ADVANCES IN WEB TECHNOLOGY

Manfred Kaul, André Kless, Thorsten Bonne and Almut Rieke
Hochschule Bonn-Rhein-Sieg, University of Applied Sciences, 53757 Sankt Augustin, Germany

ABSTRACT

Almost unnoticed by the e-learning community, the underlying technology of the WWW is undergoing massive technological changes on all levels these days. In this paper we draw the attention to the emerging game changer and discuss the consequences for online learning. In our e-learning project "Work & Study", funded by the German Federal Ministry of Education and Research, we have experimented with several new technological approaches such as Mobile First, Responsive Design, Mobile Apps, Web Components, Client-side Components, Progressive Web Apps, Course Apps, e-books, and web sockets for real time collaboration and report about the results and consequences for online learning practice. The modular web is emerging where e-learning units are composed from and delivered by universally embeddable web components.

KEYWORDS

web technology, modular web, web components, client-side component model, mobile web, embedded collaborative learning

1. INTRODUCTION

The initial development of the big e-learning platforms such as Moodle, Blackboard or the German open-source platform ILIAS started over 20 years ago. The underlying web technology is the LAMP architecture (Linux, Apache, MySQL and PHP) or Java or similar server-centric architectures. Since then many technological revolutions have taken place on all levels: Even on the lowest level, the underlying Internet communication protocol HTTP has changed into SPDY, becoming HTTP/2. HTTP/1 being client pull only, with HTTP/2 server push enters the stage. Additionally, new Web protocols like Web Sockets have emerged. The web programing language JavaScript began as a little scripting language for event handlers in a single web page and has never been designed for large systems programming. Nowadays, JavaScript is the most used programing language on the web (W3Techs 2017), and large systems with millions of lines of code have been built successfully with JavaScript. The innovation rate in the JavaScript language standard ECMAScript itself has increased to a new release every year. A new JavaScript library is released every 8 minutes somewhere around the world (npmjs 2017). The JavaScript package manager npm is offering nearly half a million packages by now, with almost half a billion downloads every day (npmjs 2017). The standards body for web technology, the World Wide Web Consortium (W3C) is hardly keeping up with the technological advances of the big technology leaders in industry, so that the browser vendors started their own consensus platform, the WHATWG, (WHATWG 2017). Many new web technologies arise constantly in all subparts of the Web. The WWW is a completely different technology today, but the former big e-learning platforms are very hard to change and to adapt to the rapid advances in Web technology. Legacy systems become a bottleneck for innovation demands in e-learning for the same reasons.

In 2015, OECD marked the current state of e-learning technology as ineffective in improving student learning outcomes (OECD 2015). The Horizon Report (NMC 2015) argues, that course apps (apps dedicated to a single specific course) are the new way to go, because current e-learning platforms are too limited and inflexible. In order to push the boundaries of what is possible, course apps provide more features and a richer environment on mobile platforms for anytime anywhere learning experience: On mobile devices, cameras, microphone, GPS and additional sensors are available to enrich the learning experience. Social and interactive capabilities lead to a richer engagement and active learning. The underlying technologies of

mobile web and mobile apps have a deep impact on online learning, teaching tools, learning habits, and teaching and learning workflows.

In our e-learning project "Work & Study", funded by the German Federal Ministry of Education and Research (Work & Study 2017), we have experimented with several new technological approaches such as Mobile First, Responsive Design, Mobile Apps, Web Components, Client-side Components, Progressive Web Apps, Course Apps, e-books, and web sockets for real-time collaboration. In this paper, we report about the results of our research and consequences for online learning practice, and how they change the online learning and teaching experience.

2. GAME CHANGER

In this chapter we address the most important game changer for online learning driven by advances in web technology.

2.1 Advances in Web Technology

When Tim Berners-Lee invented the World Wide Web (WWW) in 1989, he had an open, democratic, distributed platform for reading hypertexts in mind helping the scientific community to organize large volumes of rapidly changing texts on the Internet (Berners-Lee 2010). Starting from this first read-only system, the WWW has taken a tremendous journey towards a dynamic, readable and writable world wide platform not only for texts, but for all kinds of media and software. Today, the WWW is a *software on demand* delivery platform: With a web page, the browser also loads the appropriate software turning static text into engaging interactive media. This development took several steps and is still continuing to change the WWW as we know it. The world witnessed the beginning of dynamic web applications with web forms, into which the user fills in data. This data is uploaded to the server and processed by server-side engines (mostly PHP and Java), storing data into a server database. In the "LAMP age" dynamic behavior was server-side. Today, the WWW architecture turns to the "JavaScript age" with most dynamic behavior being client-side in the browser and even the server turning to JavaScript (Driscoll 2011).

Tim Berners-Lee being the Director of the WWW Consortium (W3C 2017) realized the importance of standardization of the browser in order to defeat cross-browser issues. Thereby, all software delivered to any browser of any vendor has a standard interface. In contrast, the server-side has never been standardized in such a fundamental way. Therefore, there is no general vendor neutral standard interface for web server plugins. Nevertheless, the e-learning community has continued to deliver e-learning content to server platforms.

The big e-learning management systems (LMS) such as Moodle, or the German open-source platform ILIAS or OLAT (Online Learning and Training) from Switzerland were developed more than 20 years ago in the LAMP age, and are server-centric. The e-learning community has produced a lot of software, teachware and learnware on these platforms with server-side integration. But without server-side standards, software remains bound to a single platform or vendor. This is well known as platform or vendor lock-in. Exchange across platform or vendor boundaries is hard and very costly. Server-to-server platform exchange formats such as SCORM often turn out to be too limited (Born 2015).

In the WWW industry the browser is the common standardized delivery platform and is rapidly enhanced by new features built directly into the platform. The need for additional libraries is decreasing at an enormous speed, because the needed features are built into the browser directly. "#UseThePlatform" is a popular Twitter hash tag in the web developer community reminding of the native capabilities of the browser: The browser supports more and more features natively. This reduces the need for additional platforms.

Therefore, in this paper we propose a focus shift from the "e-learning server platform" paradigm to the paradigm "The browser is the main platform": The browser has become so powerful, that the main part of the functionality can be implemented there. Thereby, cross vendor and cross platform issues are shifted to and resolved by W3C browser standards on a global industry scale. The e-learning industry can benefit from this paradigmatic shift by reducing own standardization burden.

Along with this shift, the amount of client software (JavaScript delivered to the browser via the web page) has increased dramatically. In order to manage the ensuing large scale of complexity, modularization is

urgently needed. Unfortunately, the founders of the web did not foresee this new turn in the evolution of the web; the web standards did not embrace modularization at all. In the beginning, web pages were seen as rather limited text with very little JavaScript helpers. Therefore, the need for modularization was not envisioned.

Partitioning large scale web software into tiny reusable components is the next step. The W3C fosters this step via the W3C web components standard "v0" that started in 2011 and "v1" in 2016 (WebComponents.org 2017). W3C web components lay the groundwork for browser software modularization. The web components consists of four features which can be used independently or all together: (1) Custom Elements (2) Shadow DOM (3) HTML Template and (4) HTML imports. With (1) the markup language HTML with tags such as <h1> for first level header, <p> for paragraph etc. can be extended by your own custom tags like <login-button>, <my-navigation>, <my-quiz> or <my-lecture>. HTML has become extensible. With custom elements, the e-learning community creates its own specific HTML extensions for its own e-learning needs. Thereby a "*Domain Specific Language for e-learning*" (E-Learning DSL) is constructed as HTML extension. In this DSL, complex e-learning applications are easily composed from basic e-learning building blocks, just as web pages are composed from nested HTML tags. (In our project, we have already started to develop several custom elements for e-learning and composed complex learning units from them, as described in the next chapter.) With (2) the shadow DOM the document object model is encapsulated and scoped. For details see (WebComponents.org 2017).

The semantics of your own custom element is given by your own web component implementation. The implementation of a custom element needs programming skills, whereas the use and remix of web components become as easy as using HTML tags. The complexity of web components is hidden behind custom tags. Publishing web components on a marketplace (WebComponents.org 2017), programers upload their web component software and e-learners use the web components simply via custom element and HTML import. For mere usage, no programing skills are needed. Both tasks are separated clearly, programing and usage.

Meanwhile, Internet usage has changed completely since the advent of the mobile devices such as the smartphone. Mobile Internet usage tripled in 5 years and 2013 surged 54%. Proportions are shifting more and more towards mobile ever since (comScore 2016). Apple generates more revenue via iPhone than all other products. Google´s CEO declared "Mobile First!" as the new company strategy. Web products have been designed for mobile devices in the first place, but also adapt themselves dynamically to other devices with other screen sizes. Using Responsive Design, the adaptation process works continuously starting from smallest screen sizes up to largest wall displays smoothly without friction.

Besides the mobile web, native apps installed from app stores and running natively on the mobile hardware have become commonplace and even surpasses the usage of the mobile web: On mobile devices, native apps generate 87% of internet traffic and the mobile web via browsers only 13% (comScore 2016). Apps being closed environments solely governed by their vendor, a huge threat to the open and democratic nature of the WWW has emerged. One reason for this disaster was, that web apps had poor performance compared to apps. Even worse, native apps only had full access to the device hardware (camera, micro, geolocation, etc), whereas web apps were restricted (sandboxed). This severe disadvantage of the mobile web led to the mentioned serious drawback in usage numbers. Therefore, leading browser manufacturers started to develop advanced browser technology to keep up with modern mobile device capabilities and native app performance. The umbrella term for the advanced browser technology is *Progressive Web App* (PWA), (Google 2016). With PWA, the exclusive advantages of native apps vanished: The mobile web became as performant and feature rich as native apps: PWA include offline mode, add-to-homescreen, push notifications, caching, pre-fetching, fast start-up and more features without any needed installation (Google 2016). The mobile web is striking back and regaining its battlefield.

A PWA has a large sophisticated software base that can best be managed by composing it from loosely coupled modular units. Here web components come in once more, returning the open and democratic nature of the web again inviting everybody to contribute.

The up coming modular web is a big game changer in digital learning as well as in the web in general. It introduces a new way of thinking about learning contents, applications, packaging and delivery. Instead of big monolithic platforms which wall learning contents and applications inside their borders, web components introduce an open way of packaging contents with application software deployable to every LMS with HTML support, to any web page or even HTML-based e-book, e.g. in EPUB3 format. The web components approach to e-learning is similar to the learning objects approach (Beck 2008), but is solely based on web

standards. Packaging learning contents with application behaviour (via the underlying JavaScript software), web components are self-contained: each component can be taken independently. Contents can be delivered cross platform and cross vendor. Even inter-platform communication and realtime collaboration become feasible. They are reusable: a single component may be used in multiple contexts for multiple purposes. They can be aggregated: components can be grouped into larger collections of content, including traditional course structures. In figure 1, a complex e-learning component which contains a video, a quiz, commenting and rating, is aggregated from basic web components, each containing encapsulated HTML, CSS and JavaScript, just like Lego® building bricks.

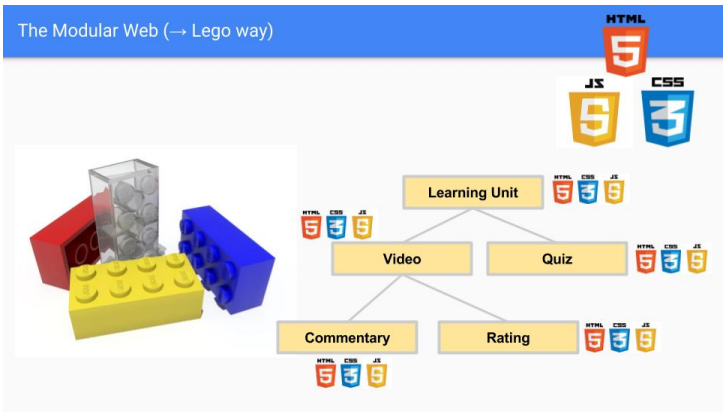

Figure 1. The Modular Web

In comparison to learning management systems (LMS), our web components based approach to learning contents, applications, packaging and delivery has several advantages: An LMS is no longer required, but when working with an LMS, or with several LMSs, web components fit in nicely: They enhance the platform via embedded components, but do not disturb the normal operation of the LMS. Even inter-LMS communication and collaboration become feasible via web components: For example, embedding the same chat component in different, heterogenous LMS enables students to chat across LMS borders, still remaining in their old well-known learning environment. In international projects this situation occurs quite often: Partner universities use different LMSs, thereby hindering students to collaborate seamlessly across borders.

Table 1. Comparison of the LMS and Web Components Approach to E-learning

	LMS	**W3C Web Components**
Standardization	SCORM, AICC, PENS, ...	W3C-Standard
software architecture	monolithic	loosely coupled web components
Component Market	LMS specific, if any	components used via markets
Reusability of Components	low, restricted to LMS, if any	high, across LMS
cross platform delivery	via SCORM, if available	directly deliverable
inter platform communication	via AICC, if available	directly available
inter platform collaboration	N/A	directly available
Composability of aggregates	LMS specific, if any	directly available via standard

LMS contents may be standardized by SCORM, AICC or PENS, but cross platform delivery is still complicated and platform dependent, as shown in our migration study (Born 2015). On the other side, W3C browser standards are developed on a global industry scale and after a small delay of engineering work, they are implemented in most modern browsers. Cross-browser incompatibility was a painful issue back in the last decade, but for several years now, major browser vendors have invested heavily in unification via WHATWG and standardization via W3C. Standardization of JavaScript is continuously refined on a yearly basis by ECMA (TC39 2017). However, in case of old browsers, small JavaScript libraries called "polyfills" are available to redeem the gap (Lawson et al. 2011). It is important to highlight that the cross platform issues no longer have to be solved by the LMS community, but are redeemed on a larger industry scale by browser manufacturers.

There already are some successful markets for learning components. For example, "learningapps.org" is a very successful marketplace for learning components with nearly 2.3m components and almost 1.7m registered users. Over 200 million learning apps have already been downloaded(LearningApps 2017). But learningapps.org was founded in 2010, when web components were not yet available. Therefore, the technology of learning apps is restricted to iFrames only: Every app is isolated within its iFrame and cannot communicate with other learning apps or the surrounding context, the web page and its CSS style rules. This restricts composability drastically: Complex web components as shown in fig. 1 cannot be composed from this kind of learning apps. Nevertheless, the huge success of learningapps.org is an excellent proof of the learning apps concept, which also contributed to our own concept.

Compared to LearningApps.org being based on iFrames, our own approach is based on the latest W3C web components standard. Since 2011 the web standard "v0" for W3C Web Components has been work in progress. The consensus of all browser vendors was finally achieved in 2016 with the "v1" web component standard, (WebComponents.org 2017). A highly advanced solution and concrete implementation of the component based approach is our own JavaScript framework for building and running web components, called *ccm*. Since 2015 we have already implemented seven complete semester e-learning courses taught at our university using *ccm* as a basis and many web components were built with *ccm* to meet e-learning needs.

2.2 Client-side Component Model (CCM)

The “Client-side Component Model” (CCM) is a model for composing, embedding and running web components inside the browser. Our implementation *ccm* consists of a single JavaScript framework *ccm* and many *ccm* components. In order to work smoothly in most browsers without polyfills, *ccm* uses only two W3C "v1" web component standards: Custom Elements and Shadow DOM. In this chapter *ccm* is explained in detail.

The *ccm* framework provides two services, one for embedding *ccm* components inside any web-based content and the other service for data management. The framework is delivered as a tiny JavaScript file (24kB minified, 9kB zipped), uses standard JavaScript (ES5) and has no dependency to other resources or frameworks. The *ccm* framework is published as free software under MIT licence (github.com/akless/ccm). It is loaded automatically when a *ccm* component is used.

A *ccm* component is a tiny JavaScript file which works with standard HTML, CSS and is based on the *ccm* framework. It may be composed of other *ccm* components and use other JavaScript frameworks. From multiple *ccm* components, complex components and web apps can be constructed. There are no restrictions in complexity and application domain. Every *ccm* component is embeddable in every web-based content.

With a *ccm* component, content as well as the underlying software for rendering and running is packaged in a single unit. The content maybe e-learning content or general web contents.

ccm components for quiz, fill-in-the-blank text, chat, team building, slidecast, commentary, rating, user input, user authentication, data logging and unit tests have already been developed and tested. All these components are published as free software under MIT Licence.

A marketplace has been developed as *ccm* component itself where all published *ccm* components are collected and information about them presented. This information contains basic metadata like author, licence, version and the public URL of the component JavaScript file.

There are three different ways to use a *ccm* component, which are described in detail: (1) Declarative via HTML Tag, (2) Functional via JavaScript, (3) Interactive via Bookmarklet. (1) Declarative via HTML Tag: This works in two steps. Firstly, the *ccm* component must be loaded using a HTML <script> tag. This introduces in a new HTML tag representing the new W3C Custom Element. Secondly, the custom tag is put at any place inside the web page any number of times, thereby embedding the custom element(s) into the web page. The component specific HTML attributes of the tag and the component specific inner HTML tags are used for setting up the configuration data. If the configuration data is stored in a database or a JSON file and the URL is specified as attribute, it is loaded directly.

```
<!DOCTYPE html>
<script src="js/ccm.unit.js"></script>
<script src="js/ccm.video.js"></script>
<script src="js/ccm.comment.js"></script>
<script src="js/ccm.rating.js"></script>
<script src="js/ccm.quiz.js"></script>

<ccm-unit>
  <h1>Learning Unit 1: WWW</h1>
  <ccm-video key="cuoZenpQveQ">
    <ccm-comment></ccm-comment>
    <ccm-rating></ccm-rating>
  </ccm-video>
  <ccm-quiz key="WWW"></ccm-quiz>
</ccm-unit>
```

Figure 2. The e-learning DSL-representation for fig. 1 becomes valid HTML via *ccm*

(2) Functional via JavaScript: This works in two steps. Firstly, the *ccm* framework must be loaded using a HTML <script> tag. Secondly, the *start* method of the *ccm* framework is called for starting a *ccm* component. The method *start* needs the URL of the selected component JavaScript file and its configuration data.

(3) Interactive via Bookmarklet: A bookmarklet is a browser bookmark enriched by JavaScript. Our *ccm* marketplace provides a bookmarklet for each published component. So a web user can use such a bookmarklet on any web page to add a new draggable and resizable web page area with the embedded component in it by a simple drag-and-drop action.

A *ccm* component is embeddable on-demand and cross-domain inside any web-based content. On-demand means that a component is not only embeddable when a website is loading, it can also be included later. Cross-domain means that components must not be located on the same server where the actual website comes from, but it can be located on any other web server. With both aspects, any web user is able to embed a component in any currently viewed web page. The embedding of a *ccm* component works without iFrame. The CSS, JavaScript and HTML ids and classes are capsuled in Shadow DOM.

Like the Lego system, *ccm* components are recombinable. This results in a dependency tree. For example, the component for rendering a learning unit reuses the components for quiz and video and the video component reuses components for commentary and rating (fig. 1). These dependencies are automatically solved recursively and asynchronously by the *ccm* framework at runtime. The framework ensures that all dependent resources are loaded in parallel and no resource is loaded twice. Any dependent resource and data can be loaded cross-domain.

The *ccm* framework and all *ccm* components are versioned and use Semantic Versioning (semver.org). The same *ccm* component can be embedded multiple times in the same web page and also different versions of a component without any conflicts and side effects. That is because each component and version has its own namespace inside a web page. It is also possible to use different versions of the *ccm* framework in the same web page; this ensures backward compatibility.

Each *ccm* component can be provided as mobile web app in two steps: The component is embedded inside the <body> tag of a blank web page. Appropriate HTML <meta> tags to display the web page on mobile devices as native app are added. Then, a user can open the web page and store it as mobile web app on the home screen of a mobile device. In modern browsers the web app specification is stored in a separate standardized manifest file instead of the <meta> tags.

The *ccm* framework provides a service for component developers for data management. It allows the usage of *ccm* datastores. A *ccm* datastore can manage datasets in one of three data levels and can also be used autonomously of *ccm* components for easy data management. The different data levels are described below. *ccm* datastores are intended to be universal and provide a simple uniform API for basic CRUD operations to create, read, update and delete data sets. On the first level the data will be managed in a local object. Then all managed datasets are fugitive data which are gone when leaving the website. On the second level the data will be managed in a client-side database. Than all managed data are still there after page reload. This is specially interesting for offline functionality. On the third level the data will be managed in any server-side database of choice. The server must have a *ccm* compatible interface. Then all managed datasets are stored persistently on a server and they are not bound to a specific client. Different network protocols are possible

for communication between client and server. In case of real time communication with the WebSocket protocol for a *ccm* datastore with data level 3, the server informs every active client about changing data sets. Then a *ccm* component which uses such a datastore can react according to these changes, mostly by updating the frontend immediately.

If *ccm* components are using the same *ccm* datastore with data level 3 and WebSocket protocol, then the components are able to exchange data in real time. If the *ccm* components are used in different domains or web-based platforms, this real time communication is cross-domain. This is the technical groundwork for digital learning defined as *Embedded collaborative learning* (ECL): Think of a learning unit that is developed in form of a *ccm* component, so it can be used inside any web-based content. That means it can be embedded inside different virtual learning environments like web-based learning platforms, mobile web apps and websites. Say the learning unit reuses a *ccm* component like chat or team building which is using cross-domain real time communication. This implies that we have real time collaboration between students and educators across virtual learning environments. For example, one student uses the learning unit from the educator website, another one from the university learning platform, another one from a mobile web app and a last one from the actual opened website via Bookmarklet. Despite the different access points, the students can collaborate in real time, which results in embedded collaborative learning across different platforms and devices.

ECL Definition: Embedded collaborative learning is digital learning with learning modules being used collaboratively and autonomously of the platform. The modules are embeddable in any web-based learning environment. Thus, collaboration can occur between all users independently of the web-based learning environments itself (Kless 2015).

Publishing these learning units under a free licence results in an interactive and collaborative Open Education Resource (icOER). Every *ccm* component published as free software can be used in OERs and enrich them with that interactive or collaborative elements. If an icOER uses ECL, a web user can participate in the collaboration from wherever the icOER is used. That means the collaboration is not limited to a single web-based environment, for example a website, an app, a course or even an e-learning platform.

Educators need a way to build web applications for digital learning without the need of computer science knowledge. For this reason, a component developer can build an input mask, in which educators specify configuration data for the component. These input masks are packaged as *ccm* component themselves and are retrieved and used as the other *ccm* components as well. We call these components *factory components*. For every component, a separate factory component serves the need for input of configuration data. Additionally, an educator can also build learning apps in the declarative way by using *ccm* component specific HTML custom tags and attributes.

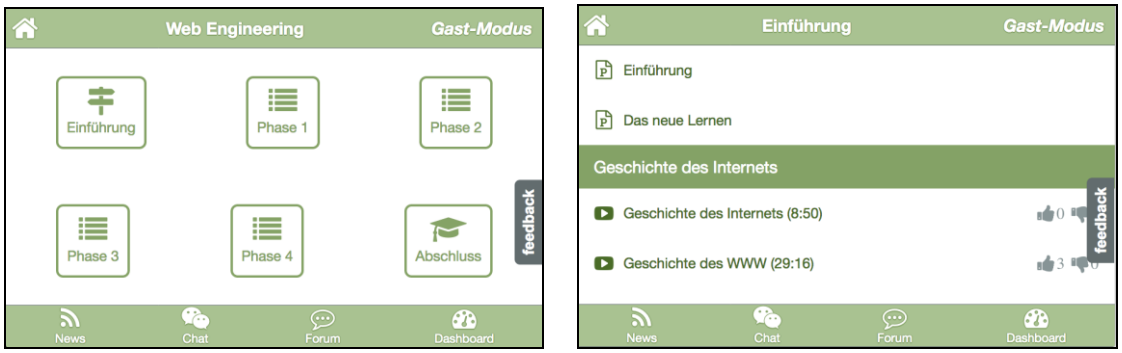

Figure 3. Web Engineering WebApp as *ccm* component aggregated from many *ccm* components

Ccm components are so powerful as one can build complete web apps from them. In our project we have built two course apps as *ccm* component by aggregating existing *ccm* components, specifically components for chat, user input, forum, menu, posts, pie chart, quiz, kanban board, team building, video and rating. A course app is a web app specifically dedicated to a single course. In CCM, a course app is built as a *ccm* component and at the same time a stand-alone mobile web app that can be loaded in any browser just like a web page. Links to the according web apps are http://kaul.inf.h-brs.de/ccm/klr_guest.html and http://kaul.inf.h-brs.de/ccm/we_guest.html. Students can access the course app inside the university learning management system (LMS) or directly via mobile web app or simply via the website. Real time collaboration via chat, kanban board and team building works across all different access points. That means, the course apps realizes ECL.

From *ccm* components complex course material can be aggregated. In our project we developed some OERs based on *ccm* components. The following link is a website where all OERs developed in our project are embedded: akless.github.io/akless/we/.

In our project four universities with different LMSs develop a joint online study program. To overcome the heterogeneity *ccm was the best solution*. *Ccm* components are reusable in every web-based learning platform, even if the platform does not know it. This was implemented in both legacy web-based learning platforms of our partnering universities, ILIAS and OpenOLAT, in parallel. Figure 4 shows the usage of the *ccm* component for kanban board in both LMSs. The changed content inside the kanban board in ILIAS is immediately updated cross-plattform in realtime at the corresponding kanban board in OpenOLAT.

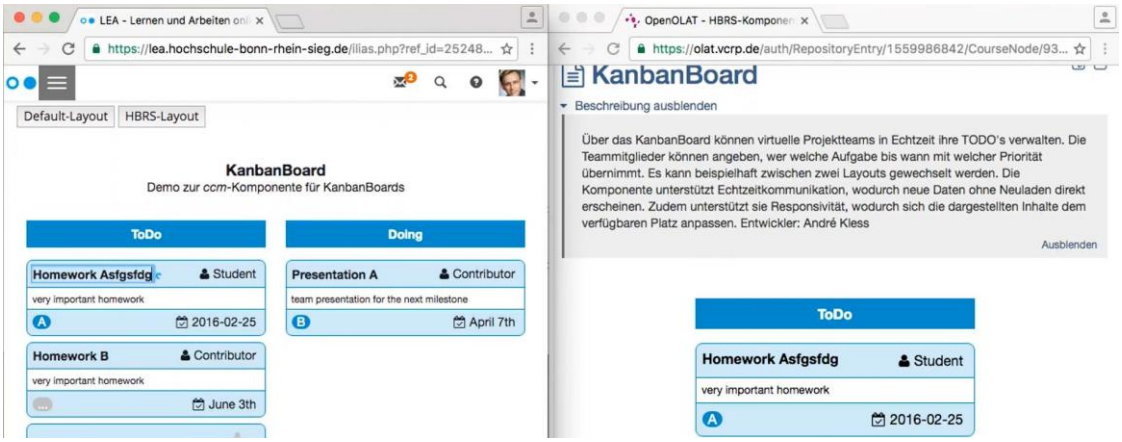

Figure 4. *ccm* component for kanban board embedded in two different LMSs, ILIAS and OpenOLAT

Legacy platforms typically offer some kind of plug-in application programming interface (API) for extending their functionality or for customizing the platform. Plugins were heavily used at our university for customizing the central e-learning platform which serves all educators and students on campus. Unfortunately, with every new release of the platform all custom plugins have to be re-engineered, which turns out to be a very costly procedure. Therefore, our e-learning administrators had to reduce the custom plugins to a bare minimum, which stands in stark contrast to the urgent demand of educators and students who want even more functionality of their platform. Our many years of experience in e-learning support shows that plugins are a too expensive for customizing or extending our platform.

Our solution is the use of HTML5 technology instead. In the HTML5 pages *ccm* components are embedded, thereby, extending the functionality of the legacy platform without necessarily using its API. The new functionality is added to the platform without altering the platform itself. A user cannot distinguish between functionality delivered by the platform and those functions delivered by *ccm* components. As an additional benefit, when changing the underlying platform the HTML5 pages and the *ccm* components can be migrated to the new platform without any re-engineering. If multiple platforms are used, *ccm* components can be used across platforms. In addition, in our project there was a special demand for the use of a best-of-breed wiki platform, because the wiki of the standard e-learning platform was not sophisticated enough. Therefore, all e-learning components, all quizzes and fill-in-the-gap games had to be migrated to the best-of-breed wiki. By using *ccm* components, this task could be achieved without any re-engineering. By focusing on the browser as the target platform instead of the e-learning server platform, technology becomes much more standardized, loosely coupled and can be migrated between different projects, contexts and platforms without re-engineering.

2.3 Apps, Course Apps and interactive E-Books

One of our first course apps we composed from *ccm* components as basic building blocks was for a course in Managerial Accounting in 2015. By using the *ccm* framework we reused *ccm* components that were previously developed for other courses or other purposes. This specific app includes course content quizzes, a forum to communicate with others, a team building component to ensure that participants in the course can find other teammates easily and also focusses on videos that are embedded as the main format to provide the course content. We chose videos as the leading format, because we believed that the success of massive open online courses on platforms like Udacity or Coursera is driven by this visual format.

In 2017 the course evaluations showed the following results: The students were very excited about using a course app. However, they preferred not to use the videos as provided, but additional textbooks for learning. Based on the evaluations and personal feedback, we saw a need to develop a new more text-oriented format for this Managerial Accounting course. The idea of creating an e-book was born. But most e-books are read-only texts with few interaction. We also wanted to offer videos, quizzes, the possibility to give feedback or marks and to communicate with other students or educators within the e-book. Therefore, we developed a course app with the look-and-feel of an e-book instead. According to (NMC 2015), the key features of course apps are mobility, interactivity, engaging design, and integrated analytics. In order to gain experience with other frameworks and to compare it to *ccm*, the open source framework Ionic based on Angular 4 was chosen. Our course app built in Angular combines e-book functionality with interaction components. Ionic enables the development of native apps for all major app stores and PWAs using a single code base. In the pilot version of our e-book, we included videos and quizzes, a forum, a function to write notes and offered links that were embedded directly in the text.

The pilot version of the app was named "@boox" and can be found under "atboox" in the app stores (Apple, Google), which is also available as PWA under www.atboox.de/app. At the time of writing, we are evaluating the use of the app in two courses with about 60 students. Only two participants had actually requested a paper version of the e-book content. Nearly all of the students are nomadic learners who prefer using apps from the stores. Some students preferred to use the PWA on their laptops for working at home or at the university. All in all, evaluations showed a very positive acceptance of this new type of format.

Compared to *ccm*, Angular is much more complex. Angular comprises both, a module and a component concept. Both concepts are solely for decomposing app complexity top-down, not for bottom-up reusability. Angular is about conquering complexity, *ccm* about avoiding and hiding complexity. Transfer of components between Angular projects turned out to be more difficult than between *ccm* projects. Our *ccm* approach is much more oriented towards global reusability and bottom-up design: *ccm* apps are composed bottom-up from reusable *ccm* components offered on a global component marketplace. This fits well with the typical workflow of educators, who have no programming skills: Educators (1) first look on the *ccm* market place for suitable *ccm* components, (2) adapt them via factory components according to their needs and (3) aggregate them building up a composed learning unit. Instead, designing and programming a complex Angular app would not be feasible for them.

2.4 Changes in Learning Environments

The aim of our e-learning project „Work & Study" is developing a new bachelor's degree in Business for non-traditional students.. Four universities of applied sciences in Germany are involved in this research project committed to developing blended learning modules. The challenge encountered was that all partner universities are using different learning management systems or even other platforms: e.g. OpenOLAT, ILIAS or PbWorks, a commercial wiki system. This kind of heterogeneity does not contribute to a convenient teaching and learning environment.

By using *ccm* we were able to develop online learning content and applications that are available in all virtual learning environments. Educators just have to develop their content once with *ccm* and can use it in any platform if needed. *ccm* components can also be combined with each other, quite contrary to apps. This allows educators to go beyond the constraints of one institution and empowers the individual educational setting, giving new freedom to place or move assignments to any place on the web. Embedded collaborative learning is reality in „Work & Study". ECL is needed not only between partner universities but also in international projects, in today´s work settings and in lifelong learning: The digital world forces a new set of skills for learning in both educational and work settings (*digital literacy*). The demand is continuously increasing for highly skilled tasks, for problem solving capabilities and interpersonal skills. Andrew Palmer states "Technological change demands stronger and more continuous connections between education and employment" (Palmer 2017). Lifelong learning is essential nowadays and will continue to play a role in the future. This demands a new technology enabling learning to take place at anytime, anywhere across devices and platforms, across institutional boundaries and work environments.

Ccm provides web components for teaching and learning media that can easily be placed anywhere on the web via *embed code*. This process is easy and sustainable: Once a learning module unit has been developed, the embed code is ready for use on the web. This kind of embedding is well accepted by YouTube users and bloggers. With *ccm* all other kinds of interactive media become embeddable as well. Compared to YouTube that provides a video player, *ccm* offers a universal *web component player*.

Ccm also provides a modular solution to *Learning Analytics*: A *ccm* component for logging browser events can be flexibly combined with any learning component, thereby tracking, filtering and storing browser interactions of this learning component. The collected data can be analyzed with the usual statistical tools afterwards. We are also planning *ccm* components that display analytics for educators.

3. CONCLUSION

The latest advances in web technology have dramatic impact on the way we can search, retrieve, edit, package, recombine, deliver and run e-learning content and applications. Based on the latest W3C standard for web components, we propose a modular kind of e-learning, concerning content, application and media and software packaging as well, which we implemented via our *ccm* framework developing many *ccm* components dedicated to e-learning. From these *ccm* components as basic "Lego-like" building blocks, we have aggregated complete semester courses, course apps and interactive and collaborative Open Educational Resources (icOER). Embedding *ccm* components in web pages, LMSs or e-books results in cross-platform deployment of interactive e-learning content and applications as well as inter-platform collaboration between heterogeneous platforms, yielding embedded collaborative learning (ECL) .

ACKNOWLEDGEMENT

The authors thank their colleague, Regina Brautlacht, for improving the English of the paper, and the German Federal Ministry of Education and Research (BMBF) for funding our research, ref. no. 16OH21056.

REFERENCES

Beck, R., 2008. "What Are Learning Objects?", in: *Learning Objects*, Center for International Education, University of Wisconsin-Milwaukee, http://www4.uwm.edu/cie/learning_objects.cfm?gid=56 on 31 May 2017.

Berners-Lee, T., 2010. Long Live the Web; in: *Scientific American*, Dec. 2010.

Born, C., 2015. *Migration from the ILIAS LMS to the Drupal CMS*, Bachelor Thesis, H-Bonn-Rhein-Sieg.

comScore, 2016. *The State of The U.S. Mobile Market*, 2016 report, http://www.aaaa.org/.

Driscoll, M., 2011. *Node.js and the JavaScript age*, https://gigaom.com/2011/04/11/node-js-and-the-javascript-age/.

Google, 2016. *Progressive Web Apps*, https://developers.google.com/web/progressive-web-apps/ on 1 July 2017.

Kless, A., 2015. *Eingebettetes kollaboratives E-Learning*, Master Thesis, Univ. of App. Sci., Bonn-Rhein-Sieg.

Lawson, B.; Sharp, R., 2011. *Introducing HTML5*, Introducing Polyfills, pp. 276–277.

LearningApps, 2017. *Operation Statistics*, http://verein.learningapps.org/, 1 July 2017.

NMC, 2015. Course Apps, An NMC Horizon Project Strategic Brief, *The New Media Consortium, Volume 2.4*, October 2015, https://www.nmc.org/publication/2015-nmc-strategic-brief-course-apps/ , 25 May 2017.

npmjs, 2017. *Node Package Manager*, https://www.npmjs.com/ .

OECD, 2015. *Students, Computers and Learning: Making the Connection*, OECD Publishing, Paris.

Palmer, A., 2017. Lifelong learning is becoming an economic imperative, in: *The Economist*, 12 Jan 2017.

Pichai, S., 2017. *Google I/O 2017 Keynote*, https://www.youtube.com/watch?v=TUyY2Ng0Yjc , on 25 May 2017.

TC39, 2017. *TC39 - ECMAScript®*, http://www.ecma-international.org/memento/TC39.htm.

WebComponents.org, 2017. *Web Components*, https://www.webcomponents.org/ on 31 May 2017.

WHATWG, 2017. *Web Hypertext Application Technology Working Group (WHATWG)*, https://whatwg.org 1 July 2017.

Work & Study, 2017. *Work and Study - offene Hochschulen Rhein-Saar*, joint project of the universities of applied sciences Koblenz, Worms, Bonn-Rhein-Sieg, Saar; Federal Ministry of Education and Research, ref. no. 16OH21056, http://www.work-and-study.info .

W3C, 2017. Tim Berners-Lee, https://www.w3.org/People/Berners-Lee/ .

W3Techs, 2017. https://w3techs.com/technologies/overview/client_side_language/all .