Migrating Learning Management Systems Towards Microservice Architecture

Pia Niemelä[0000−0002−8673−9089] and Heikki Hyyrö

Tampere University, Tampere, FINLAND https://www.tuni.fi/en

Abstract. Microservice architecture provides on a set of modular, independent and fault-tolerant services. In recent years, new architectures have evolved with an emergence of recurrent, and effective architectural patterns essential in maintaining and scaling microservice-based systems. However, in the domain of education there is a lack of open-source, microservice-based systems that are easily configurable for various teaching, research, and commercial purposes. Preferably, these services should be orchestratable as part of other education-related service compositions as well. In this paper, a study of microservice-based learning management systems is conducted by focusing on two systems that the authors are involved in: WETO and Plussa. We report the current status of these systems through the lens of microservice architecture and draft a proposal for the synthesis of an ideal, decoupled learning management system.

Keywords: Learning management system · migration from monolith · microservice architecture · open data

1 Introduction

Learning environments and Learning Management Systems (LMSs) have been in focus of active development and research. However, the development is led by multinational companies whose topmost interest is commercial. The cross bet between non-profit organizations and commercial actors is far from unseen. In the domain of scientific publishing, it has led to the situation where some data is freely available, and some are behind the paywall of commercial journals. This is an obstacle from the viewpoint of open data: publications on the other side of the wall will get less reads and references. Research Data Alliance (RDA) works against these obstacles and promotes openness and reuse of data [32].

In Finland, opening public data sources started approximately in the beginning of the 2010s, which triggered such initiatives as Open Data [15] and Linked Data [12]: Open Data fosters citizens' participation by increasing the transparency of decision-making mechanisms [15], Linked Data fosters the exploitability of data, by agreeing on meta-data formats in spirit of semantic web [12]. In consequence, public organizations were called to open up their data and provide RESTful web service APIs for its retrieval, i.e., so-called microservices. This call for openness and microservices was promptly responded by National

Land Survey of Finland [22] and Open API Project of Helsinki City, the Capital of Finland. As desired, opening data catalyzed plenty of new business and services in the SME sector, built on top of this data.

In Finland, education is a public service sustained with tax money, where the 2019 government programme states: *An equal society seeks to provide opportunities for every citizen to study to their full potential* [30]. Thus, educational resources and opportunity for learning should be as open and as accessible as possible, and this concerns data gathered during all educational events as well, including learning analytics data[1]. Accordingly, the Ministry of Education funds multiple projects that focus on developing and innovating more accessible learning environments [21]. These projects include Digital Education For All [33, DEFA], FITech [7], and Smart Learning Environments [6, Älyoppi].

DEFA provides a shortcut to university studies: participants can just take individual courses, or compile courses as bigger and meaningful entities, with a bonus of being accepted as a student, dependent on one's success. All the studies are available as MOOCS, enrolling several thousand participants simultaneously. An example of this is *Elements of AI, a prize-winner MOOC implemented by University of Helsinki that introduces artificial intelligence theory with practical exercises. FITech network fills the digital skills gap and prepares ICT professionals for the industry, the initiative is called FITech ICT. It runs for three years and the target output is 4000–5000 ICT-savvy individuals. Smart Learning Environments project develops and expands university-specific e-learning environments (including automatic validation, visualizations and simulations) for national inter-university use. As its sub-project, the material converter from LaTeX to RST, or shortly *LaRST, seeks to find underlying ultimate data format, where Latex is upvoted by mathematicians, in particular. The original Latex content can then be printed as a book or converted to on-line course area with auto-assessed exercises.

In the development of reusable learning materials and flexible infrastructure, the key to success from the technical point of view are such architectural choices that maximize flexibility and the independence of services. By the same token, microservices have trended since 2014 [28] with the motivation of decoupled services, better maintainability, scalability, DevOps support, zeitgeist (*because everybody else is doing it), fault tolerance, easy technology experimentation, and delegation of team and software responsibilities [29]. In migrating towards microservices, it is first essential to identify the core functionalities and then divide them as services in an appropriate manner. Thus, this study asks:

- RQ1: Which microservices are the core functionality of an LMS?
- RQ2: How (micro)service-oriented are the LMSs of Tampere University?
- RQ3: What would a more decoupled architecture of LMS look like?

Next, we introduce our research context, the LMSs used more in detail, followed by a short explanation of the method used. Next chapter summarizes

[1] Privacy disclaimer: after anonymization and pseudonymization etc.

the currently identified state-of-art solution, as the target state of our LMS development. In the Discussions and Conclusions section, we validate our results and ponder critically different architectural choices made and (false or overemphasized) promises attached to them.

2 Method and research context

2.1 Feasibility study

This study follows loosely the method of feasibility study [5]. Essential in the method is the determination of the viability and benefits of a proposal without forgetting to evaluate the problems as well. In the Finnish landscape of different LMSs, the studied systems of Tampere University offering, WETO and Plussa, provide both plenty of data to analyze as well as lessons learned, thus, are a representative sample.

2.2 Higher education LMSs evolving towards micro-services

In the evolution from monolithic to microservice architecture, WETO resides in a more monolithic position than Plussa that can be described largely service-oriented. In comparison with more traditional service-oriented architecture, microservice architecture stands out as being more decoupled, its services can be deployed independently and scaled horizontally with proper load-balancing mechanisms.

Using microservices each learning service can be developed and deployed independently, which makes the exploitation of common resources easier. Developed learning materials exemplify such resources, and they should be as exploitable and open as possible keeping in mind that the funding comes from the Ministry of Education. The target is that reusable parts of learning materials and gathered data will be publicly available for all national actors and service providers mainly in the domain of education and employment services.

2.3 WETO

WETO[2] is an LMS developed and used at Tampere University since early 2000's. Its core features are:

- Basic content management: course pages may be created/edited using a browser, e.g., images and files may be uploaded and linked to the pages.
- Submission management: pages may upload student submissions (e.g. homework or exam answers).
- Grade management: pages may comprise student grades, and grading rules may be defined hierarchically:

[2] An acronym derived from **We**b **T**eaching **O**rganizer.

  - E.g., the top-level page could show overall course grades, its "Exercise"- and "Exam"-sub-pages exercise and exam grades, respectively.
- Automated grading: submissions may be graded automatically. WETO has a built-in support for multi-choice questions; programming tasks can be graded by an external grader.
- Rights management: pages may define permissions for viewing and uploading submissions, these permissions may be time- and student-specific.
- Peer-reviewing: submissions may be peer-reviewed anonymously.
- Basic discussion forum: pages may have a forum where teachers and students can discuss learning-related issues. Students remain anonymous.

WETO is a fairly traditional Java EE 6 Servlet application. A WETO instance needs a single PostgreSQL master database and one or more PostgreSQL course databases. The master database contains global information about users and courses. Course databases contain course-specific information such as teachers, enrolled students, grades and submissions. Each course database may host several courses. In principle, the WETO instance may be connected with one or more heterogeneous external automated graders. A restriction is that each external grader needs to directly manipulate WETO's course databases, which implies the need to implement WETO-specific middleware for any non-WETO-specific grader. The architecture is illustrated in Fig. 1. Evidently, WETO itself is a monolithic application even though the mechanism of exploiting external graders might resemble microservices.

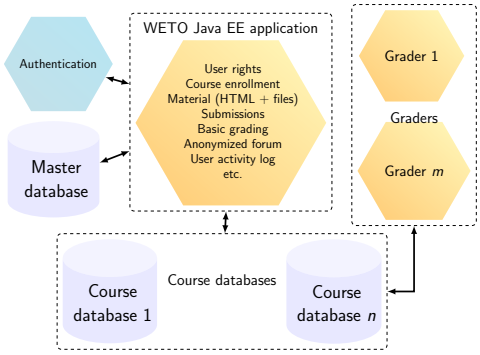

Fig. 1. WETO architecture.

2.4 Plussa

Plussa is Tampere University's application of A+ of Aalto. Plussa divides into two parts: a front that takes care of role-based authentication (students, teachers, admins), and stores grades, the other part is MOOC grader that runs in the background invisible to the user and manages courses and provides such built-in exercise types as multi-choice questions [2]. Teachers interact with the course area mainly through Git version control system, as illustrated in Fig. 2.

The A+ platform manages user authentication and authorization, stores submissions and grading results (feedback and points), manages course configuration and setup, provides course monitoring (student progress and statistics) and web user interfaces for students and teachers. [25].

One major design principle of Aalto A+ was to provide an easily extendable, service-oriented architecture that could be enhanced with different services, for example, with new graders for automated assessment [17]. Tampere University Plussa has been enhanced with such services as Grading Helper [35], and Peer-review component [11] that both are integrated by using the LTI protocol explained in more detail in Ch. 2.4.

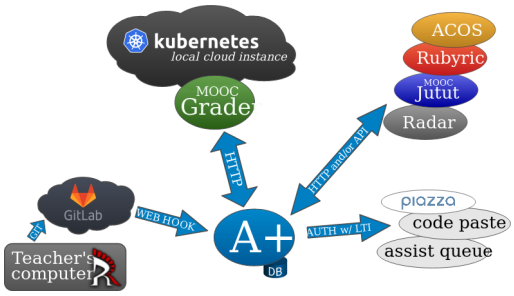

Fig. 2. A+ architecture visualized by the lead developer J. Kantojärvi [16]

In addition to LTI services, A+ LMS can be extended with graders. The graders do not require separate authentication, since a student has already been authenticated by A+ frontend. If the grader runs in a separate server, such as in Fig. 3, this is signified in the exercise configuration with the *submit RST directive and by giving the URL to the exercise. The exercise itself is represented inside an iframe element (1), and after completion when the student presses Grade button (2), grading is done and points are returned to A+, which stores the results (both feedback and points) in its database. In Fig. 3, grading is done by an external grader. Other options exist, e.g., grading within temporary docker containers, or submitting exercises to a separate Kubernetes cluster for grading.

Automatic assessment with graders has been one of the principal goals when constructing A+. The work began with a literature review of different auto-assessment systems. In the review, such features as submission/resubmission, manual assessment options, sandboxing, distribution and availability of different LMSs were examined with the mindset of designing as decoupled assessment system as possible [13].

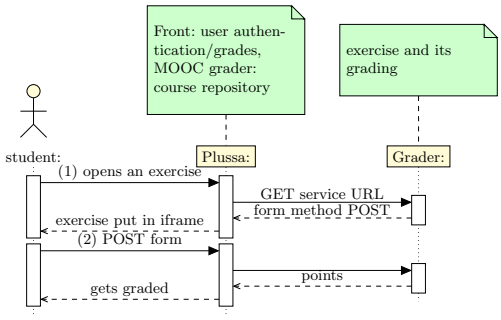

Fig. 3. A Plussa grader running in a separate server.

Synchronous, asynchronous, and static analysis e.g. with Git and SonarQube In their A+ study, Karavirta et al. divide the assessment systems in three categories: synchronous, asynchronous, and static [17]; synchronous meaning assessment during one HTTP transaction, asynchronous allows later grading with the provided unique submission URL; and static implies

batch assessment of stored code files (use, e.g., in Rubyrik). Static assessment covers also Git-based assessment. Students submit new files into their own repositories. In this scenario, the only thing returned to the LMS is the URL of the Git repository.

The industry partners of Tampere University have listed Git skills to be an important asset, which is confirmed by Haaranen's study, which corroborates their high demand in the software industry, and the emphasis on their importance in the literature as well [10]. Haaranen was not content with the literature review only, but also tested Git during a large (ca. 200 students) course and evaluated the results from both instructor's and learners' point of view. The study proved that Git served well for exercise submissions and course material dissemination.

In the strategy of Tampere University, Git system will be exploited even further: continuous integration pipeline is set on, unit tests are automatically run when a submission is pushed to the Gitlab repository and SonarQube static code analysis is available for a course personnel as well as students for self-reflection of the quality of their code. Besides auto-testing, the benefit of the pipeline for continuous integration is that students familiarize themselves with the DevOps development that is another skill appreciated by employers.

Learning analytics and gamification More recent studies in Aalto has concentrated on adding gamified features and learning analytics to the LMS. Gamification includes such new features as earning badges [9], where learning outcome, however, remained controversial in the executed experiments. In addition to simple built-in exercise types such as multi-choice questions and sending a file, A+ was supplemented with three systems for new exercise types, Acos, Jsvee, and Kelmu [26]: Acos is an external server for animated exercise types such as Parson's puzzles and drag-and-drop tasks; Jsvee provides **a visual representation of a notional machine and shows how the program state changes when a program is executed step-by-step*[27]; and Kelmu enables annotated animations. In addition to external services, more embeddable *active elements are developed [23, 24].

In addition to engaging students with gamified user experience, A+ improves teachers' view of the learning process as a whole. Lehtinen et al. note that teachers' want to monitor the expected progress of their students, improve allocation of learning material, identify problematic areas in learning material, and improve interaction with learners [18]. At a minimum, Lehtinen et al. targeted allowing teachers and researchers to access the data collected during learning events. Actual learning analytics was available through the service API that provided information in various formats, and additionally, raw data files were downloadable. Service API was considered beneficial in improving interoperability with different LMSs, e.g. A+ and Moodle.

In LukioPlussa, new graders for math exercises were added as internal graders such as MathCheck, Geogebra, and Abitti. The ease of adding new graders demonstrates that the plug-in architecture of A+ provides effortless extensibility. In addition, interoperability is ensured by using consistently agreed protocols, such as proprietary A+ protocol, and widely adopted LTI standard.

A+ protocol The A+ grader service protocol is defined in the GitHub in Grader's README [1]. The simplified sequence diagram in Fig. 3 represents roughly the process. Notably, the protocol is not RESTful. Essential information is conveyed in HTTP headers, such as status and points. Moreover, unlike REST protocol defines, Create-Read-Delete-Update operations are supported only partially, from HTTP methods only GET and POST are supported. When the exercise is launched, a grader gets the submission URL. The URL can be exploited for asynchronous grading that happens later.

LTI protocol LTI is a specification developed by the IMS Global Learning Consortium whose main purpose is to provide a standard way of integrating learning tools [14]. The IMS Global Learning Consortium is an international, not-for-profit member organization dedicated to enabling the growth and impact of learning technology globally. As a protocol, LTI is more restricted than A+ protocol. In the services concerned, LTI is used mainly at the authentication stage.

The LTI defines a standard way in which LMS can communicate with an off-platform learning service provider. When the LMS aims at using the service, it sends a signed POST message with a browser. In signing the request, OAuth protocol is used, which e.g. enables an application to use e.g. Twitter or Google for authentication. LTI allows single-sign-on, a student needs to authenticate to the LMS only – not individually to each service used in a course. Once student is logged in LMS, external services linked to the platform via LTI will be informed by the platform that the user has already been authenticated. Thus, a student does not need to register to yet another service. For example, MATLAB supports LTI, therefore being integrable to Plussa [20]. Other LTI-services done in Tampere University comprise Grading Helper and Peer-review Platform; they are integrated using Aalto University's Django LTI login module [3].

3 LMS with a decoupled microservice architecture

As discussed in Section 1, one of the main goals of the Smart Learning Environments project is to find ways to share university-specific LMS resources across different universities. Due to the heterogeneity of the systems, and lack of consensus, convergence towards a single common LMS is unlikely, but with the setting of a more modest target, the way to advance this objective would be via migration towards microservice architecture. Highly decoupled and independent microservices would lower the threshold of sharing them among universities, although different universities would still use their own LMSs. Hence, this section sketches out a microservice-based LMS oriented towards flexible reuse of its components.

3.1 The major principles of decoupling

– Page hosting should be minimally restricted. The LMS will be used via a web browser, hence the requirement is that the course pages can be hosted by any

type of web server/service, for example, a generic web-content management system, such as WordPress.
- For the sake of simplicity, the authentication method should be common for all services, preferably with a wide, existing support, e.g.:
  - a token-based method: an authentication server provides the LMS frontend with a token (e.g. JWT) to be included in each request,
  - the widely used LTI protocol (see Section 2.4) qualifies as well.
- LMS features should be split into independent (micro)services that assume minimal mutual knowledge and dependence.
- Each service should provide both a web API and a reference implementation of a frontend component (HTML, JavaScript etc.) for accessing the service. The API should provide facilities to get a response from the service as plain data (preferably in JSON).
  - Many existing LTI-based services send pre-rendered HTML content to the service consumer. Our proposal is more stringent, as we strongly suggest that each service provides also a data-only type API to enable the migration towards microservice architecture.
  - A reference frontend component could lower the threshold for incorporating new services. Preferably, the reference implementation only needs tweaking its context-specific configuration parameters, e.g., application ids and service URLs, to be embeddable as a new service to an HTML page [3].

Fig. 4 sketches our proposal for a microservice-oriented LMS architecture and answers to first research question: The main component is the user/rights manager with a centralized responsibility for maintaining information about course membership roles and more specific user rights. When a user accesses the LMS through a view provided by the host, the system authenticates the user, if necessary. The authenticated user receives an authorization token that will then be attached to all LMS service requests. Each service uses the token to verify the identity of the user and to consult the user/rights manager about the user's roles/rights. This solution is similar to the one proposed by Baier and Allen [4]. Alternatively, roles and rights could be attached to the token, but this fairly common practice may result in invalidated tokens due to outdated rights [4].

Furthermore, the sketch suggests grade management, discussion forums, user activity logs/analytics, and automatically graded exercises as the core functionalities of the LMS. For decoupling, each service should maintain its own database. In service access, a request typically entails course and user ids that are unique, which is enforced by the authentication and user/rights managers. The sketch also depicts nested services: a service itself might employ sub-services; e.g., automated grading may use separate grader services.

Optimistically, this sketch implies independent services that are capable of performing their tasks without employing any other services. Inter-service communication would mandate the implementation of e.g. an asynchronous message queue: a typical solution for service interoperability via HTTP with an inherent decoupling [31].

[3] E.g. assuming that Cross-Origin Resource Sharing etc. have been properly configured to allow AJAX calls to the services.

4 Discussion and Conclusions

This paper reviews the migration of LMSs from monolithic towards more microservice-based architecture. Tampere University LMSs, WETO and Plussa, are taken as starting points, and are then analyzed and synthesized into one modern microservice-based LMS. Next, the results are summarized as responses to the research questions. **RQ1: The core microservices of an LMS** Authentication, management of user and course information, grading, and providing reports and analysis.

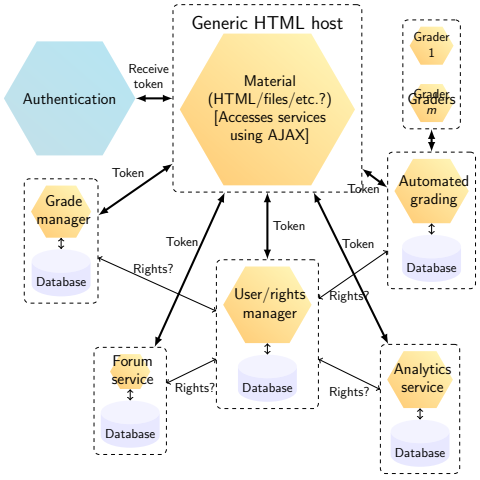

Fig. 4. A decoupled microservice architecture.

RQ2: Current microservice provision of Tampere University LMSs In the continuum from monolith to microservice architecture, WETO is more to the left, but taking steps to service-oriented direction. Plussa mainly waits for an agreed common exercise format and the anticipated outcome of LARS project, i.e., the Latex to RST converter. In addition, teachers' user experience is defective in many respects, leaving a lot of space for improvement.

RQ3: The decoupled target architecture of LMS In particular, easy authentication services are of paramount importance. In general, LMS should provide open, reusable and standard services that can function independently. Standard solutions are promoted, such as LTI for authentication and grading. Other standard solutions include: Sharable Content Object Reference Model (SCORM), a set of specifications for creating and sharing e-learning [34], xAPI as learning analytics metadata [19], and *Learning analytics interoperability [8].

The benefits of microservice architecture become more apparent when the target is to co-operate between multiple universities, such as in Smart Learning Environments project, i.e., when the complexity of a system increases. Then, micro-architecture may facilitate better coordination, re-usability of resources, such as learning material, and orchestration of different services represented in Fig. 4. However, we want to highlight that there are problems ahead and too much unfounded hype: microservice architecture will provide neither a single point of failure nor easy logging, and service orchestration is, in fact, quite complex. Cooperation with many players is not supposed to be smooth, e.g., sudden and poorly informed changes may cause discontinuities in service provision. Thus, realization of this vision requires a lot of work, good will, and co-operation.

References

1. Aalto University: A+ grader service protocols (2016), https://github.com/Aalto-LeTech/a-plus/blob/master/doc/GRADERS.md
2. Aalto University: A+ LMS (2018), https://apluslms.github.io/architecture/
3. Aalto University: Django LTI login (2019), https://github.com/Aalto-LeTech/django-lti-login
4. Baier, D., Brock, A.: Authorization is hard! implementing authorization in web applications and apis (2018), presentation in NDC London 2018. Video available at https://vimeo.com/254635640 (accessed on 31.10.2019).
5. Bowen, D.J., Kreuter, M., Spring, B., Cofta-Woerpel, L., Linnan, L., Weiner, D., Bakken, S., Kaplan, C.P., Squiers, L., Fabrizio, C., et al.: How we design feasibility studies. American journal of preventive medicine **36**(5), 452–457 (2009)
6. eOppimiskeskus: Älykkäät oppimisympäristöt ja niiden sisällöntuotanto (ÄlyOppi) -hanke (2019)
7. Finnish Institute of Technology: Upgrade your knowledge and study ICT courses with FITech! (2019), https://fitech.io/en/upgrade-your-knowledge-and-study-ict-courses-with-fitech/
8. Griffiths, D., Hoel, T., Cooper, A.: Learning analytics interoperability: Requirements, specifications and adoption. Public Deliverable D **7** (2016)
9. Haaranen, L., Ihantola, P., Hakulinen, L., Korhonen, A.: How (not) to introduce badges to online exercises. In: Proceedings of the 45th ACM technical symposium on Computer science education. pp. 33–38. ACM (2014)
10. Haaranen, L., Lehtinen, T.: Teaching Git on the side: Version control system as a course platform. In: Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education. pp. 87–92. ACM (2015)
11. Heino, P.: Peer-review Platform (2018), https://github.com/piehei/prplatform
12. Hyvönen, E., Tuominen, J., Alonen, M., Mäkelä, E.: Linked Data Finland: A 7-star model and platform for publishing and re-using linked datasets. In: European Semantic Web Conference. pp. 226–230. Springer (2014)
13. Ihantola, P., Ahoniemi, T., Karavirta, V., Seppälä, O.: Review of recent systems for automatic assessment of programming assignments. In: Proceedings of the 10th Koli calling international conference on computing education research. pp. 86–93. ACM (2010)
14. IMS Global Learning Consortium: LTI Message (2019)
15. Jaakkola, H., Mäkinen, T., Eteläaho, A.: Open data: opportunities and challenges. In: Proceedings of the 15th International Conference on Computer Systems and Technologies. pp. 25–39. ACM (2014)
16. Kantojärvi, J.: Architecture of A+ LMS (2018), https://apluslms.github.io/architecture/presentation//step-4
17. Karavirta, V., Ihantola, P., Koskinen, T.: Service-oriented approach to improve interoperability of e-learning systems. In: 2013 IEEE 13th International Conference on Advanced Learning Technologies. pp. 341–345. IEEE (2013)
18. Lehtinen, T., et al.: Bootstrapping learning analytics case: Aalto online learning (2017)
19. Lim, K.C.: Using xAPI and learning analytics in education. In: Elearning Forum Asia. pp. 13–15 (2016)
20. MathWorks: MATLAB Grader (2019), https://se.mathworks.com/products/matlab-grader.html

21. Ministry of Education: The Key Projects (2019), https://www.aalto.fi/en/aalto-university/the-key-projects-funded-by-the-ministry-of-education-and-culture
22. National Land Survey of Finland: Maps and Spatial Data (2019)
23. Piitulainen, R.: A+ Active Element (2017), https://apluslms.github.io/events/2017-1st-a-plus-con/active_element.pdf
24. Piitulainen, R.: A+ RST tools (2017), https://version.aalto.fi/gitlab/piitulr1/aplus-rst-tools-ae
25. Riekkinen, M., et al.: Integrating Stratum and A+ functionalities in Moodle: Architecture and evaluation (2017)
26. Sirkiä, T., Haaranen, L.: Improving online learning activity interoperability with ACOS server. Software: Practice and Experience **47**(11), 1657–1676 (2017)
27. Sirkiä, T., Sorva, J.: How do students use program visualizations within an interactive ebook? In: Proceedings of the eleventh annual International Conference on International Computing Education Research. pp. 179–188. ACM (2015)
28. Soldani, J., Tamburri, D.A., Van Den Heuvel, W.J.: The pains and gains of microservices: A systematic grey literature review. Journal of Systems and Software **146**, 215–232 (2018)
29. Taibi, D., Lenarduzzi, V., Pahl, C.: Processes, motivations, and issues for migrating to microservices architectures: An empirical investigation. IEEE Cloud Computing **4**(5), 22–32 (2017)
30. The Government of Finland: Programme of Prime Minister Antti Rinne's Government: INCLUSIVE AND COMPETENT FINLAND – a socially, economically and ecologically sustainable society (2019)
31. de la Torre, C., Wagner, B., Rousos, M.: .NET Microservices: Architecture for Containerized .NET Applications. Microsoft Corporation (2018), https://aka.ms/microservicesebook.
32. Treloar, A.: The Research Data Alliance: globally co-ordinated action against barriers to data publishing and sharing. Learned Publishing **27**(5), S9–S13 (2014)
33. University of Helsinki: Digital Education for All (2019), https://www.helsinki.fi/fi/projektit/digital-education-for-all
34. Vossen, G., Westerkamp, P.: Towards the next generation of e-learning standards: SCORM for service-oriented environments. In: Sixth IEEE International Conference on Advanced Learning Technologies (ICALT'06). pp. 1031–1035. IEEE (2006)
35. Väkevä, E.: Grading Helper (2018), https://github.com/eliisav/gradinghelper