Requirement-Driven Architecture for Service-Oriented e-Learning Systems

Rawad Hammad
King's College London
London, UK
Email: Rawad.Hammad@kcl.ac.uk

***Abstract*—The continuous evolving of Technology Enhanced Learning (TEL) requirements, more specifically Functional Requirements, increases the complexity of TEL software system since such requirements cannot be met by one TEL/e-learning solution. In addition to the traditional Virtual Learning Environments/Learning Management Systems capabilities, such Functional Requirements include: video streaming, plagiarism checker for students' submissions, e-portfolio, etc. Therefore, combining various e-learning software systems, solutions, or tools seems more realistic. However, a limited effort has been done to investigate and control the impact of combining different solutions on the quality, i.e., Non-Functional Requirements (NFRs), of the overall e-learning software system. This paper proposes a new approach to elicit, precisely specify, and manage NFRs for TEL software systems. To meet these capabilities (i.e., Functional Requirements and Non-Functional Requirements), this paper also proposes a flexible service-oriented architecture for e-learning systems. The proposed list of NFRs is comprehensive and can be customized to various e-learning systems to meet stakeholders' requirements. Moreover, the proposed architecture needs to be further developed to test its impact on TEL software systems in real scenarios.**

Keywords-Technology Enhanced Learning; e-learning; architecture; Non-Functional Requirements; Software architecture; SOA; Web Services.

I. INTRODUCTION

The continuous rise of using eLearning in higher education increases the complexity of e-Learning/Technology Enhanced Learning (TEL) Software Systems [1]. On one hand, there is a continuous demand for various supplementary capabilities, more specifically Functional Requirements, that cannot be met by one TEL/ e-learning software system only. For instance, in addition to the traditional Virtual Learning Environments (VLE) capabilities, various supportive capabilities are required (e.g., video streaming, plagiarism checker for students' assignments, e-portfolio, etc. Therefore, combining various e-learning software systems, solutions, or tools seems more realistic. On the other hand, a limited effort has been done to investigate the impact of combining different solutions on Non-Functional Requirements (NFRs) of the overall e-learning service or software system. Such Non-Functional Requirements include performance, reliability, availability, recoverability, etc.

Literature evidence shows that Non-Functional Requirements are not properly managed over the Software Development Life Cycle (SDLC) [2]. This applies to NFRs elicitation, specification, documentation, and evaluation. One of the potential reasons behind this is related to the nature of applying TEL solutions in higher education institutions as they focus on Functional Requirements at the expense of Non-Functional Requirements. Also, there is a lack of literature evidence on how NFRs are elicited and specified. Most of the e-learning systems evaluation is performed against the Functional Requirements only (e.g., [3] and [4]). Moreover, NFRs subtle nature makes them challenging to elicit in advance, and most likely to be approached iteratively along software development life cycle [5]. NFRs are very important to software architecture; they are also known as Architecturally Significant Requirements because they have a measurable impact on the architecture of software system [6] [7].

Therefore, this paper investigates the current approaches to manage, more specifically elicit and specify, NFRs over TEL software development life cycle. NFRs management refers to the process of eliciting, specifying, communicating, and controlling Non-Functional Requirements over software development life cycle [8]. Since, NFRs are persistent, rarely changed, this paper focuses on the early stages of NFRs management process. These stages include NFRs elicitation, specification, and communication. Then, it proposes a flexible architecture for e-learning solutions to meet the early-identified NFRs. The rest of this paper is organized as follows. Section II summarizes related work; Section III proposes a new approach to elicit and specify NFRs for TEL systems; Section IV proposes a service-oriented architecture for e-learning software systems; Section V concludes the paper with future research directions.

II. RELATED WORK

There exist different e-Learning/TEL Software Systems, where some of them are: (i) *open source*, such as: Moodle [3], Atutor [9], Sakai [10], and Ilias [11] or (ii) *propriety*, such as: Blackboard [12] and Desire2Learn [13]. Such systems are known as Learning Management Systems (LMSs) or Virtual Learning Environments (VLEs). The current LMSs/VLEs cannot support architectural flexibility, to different extents, due to their monolithic design [14]. LMSs evolved from black box systems, known as first generation LMSs, towards more modular architectural approach, known as second generation LMSs [14]. Much of this evolution was due to the standardization initiatives, such as: Sharable Content Object Reference Model (SCORM) and IMS Global Learning Consortium Learning Design (IMS LD), which allow good level of interoperability between different LMSs, their components, and third-party plugins/tools. For instance, IMS Learning Tools

Interoperability (LTI) is used by many e-learning tools to align or map their configurations with LMS configurations. Meanwhile, various architecture-oriented improvements on Atutor LMS have been introduced [9]. Similarly, more modular structure has been considered in the case of Sakai LMS, especially in relation to service orientation [15]. This has increased the scalability and extendibility of the current LMSs via plugins deployment.

However, such structure is not sufficiently agile. New requested plugin needs planning and deployment procedures and might have impact on systems performance or other related NFRs. In addition, the recent move towards cloud-based e-learning solutions, especially Software as a Service (SaaS), made it more challenging for the current e-learning systems to effectively exchange assets and efficiently co-exist with each other (e.g., sharing hardware resources). Hence, the next section will present a comprehensive and consistent approach to elicit, specify, and communicate NFRs in relation to TEL solutions to consider them later to design a flexible architecture for TEL systems.

III. NON FUNCTIONAL REQUIREMENTS IN TEL

In the light of the above discussion, a good starting point for architecting e-learning solutions is to thoroughly consider its NFRs in a consistent way. Our approach is inspired by one of the most reliable approaches to elicit NFRs, which is the Quality Attribute Workshop (QAW) approach [16], developed by Carnegie Mellon University Software Engineering Institute. Simply, this approach refers to engaging system stakeholders, or their representatives, early in the life cycle to discover the driving Non-Functional Requirements of software system through a series of workshops. Unlike QAW structure that has a rigid structure and lacks the base definitions for NFRs, we opt for a flexible structure for our proposed approach. The structure of the proposed approach must address the following phases: (i) *induction phase*, to introduce the approach to stakeholders, or their representatives, and explain the rationale behind it and who is doing what, (ii) *business view phase*, to introduce high-level Functional Requirements for the proposed solution, (iii) *architecture view phase*, to present the proposed solution architecture at a high-level including useful details (e.g., hardware, certain technologies, etc.), (iv) *architectural drivers phase*, to summarise the key drivers of the proposed solution, which could be high-level capabilities, organizational concerns, etc., (v) *scenario phase*, to divide the audience into groups to brainstorm real scenarios for using the systems, and to validate them, and link them with NFRs, and finally (vi) *precisely specify TEL software system NFRs* based on the NFRS template introduced later in this section (i.e., Tables I and II). Precise specifications of NFRs means to pick up the definition listed in Table I or II, and to add certain parameters to the definition as explained later.

The proposed phases could be conducted as separate workshops, meetings, interviews or other potential formats, which can be better decided by organisational business analysts. Also, phases can be merged together or divided into two or more depending on the context factors that include: the scale of the TEL software system, nature of stakeholders, their technical background and interest, etc. The key role of the business analyst team is to maximise the benefits of stakeholders' engagement to get accurate enough NFRs specifications. One of the central steps here is to avoid natural language-based specification as this may lead to subtle requirement specifications, which cannot be measured. To do so, we opt for a standard-based approach based on ISO 25010 Systems and Software Quality Requirements and Evaluation: (i) *Product Quality (PQ) Model* that measures the static qualities of a certain software system, depicted in Figure 1, and (ii) *Quality in Use (QiU) Model* that measures the dynamic qualities of a certain software system when it is applied in a particular context [17], depicted in Figure 2. Both models have a set of precisely defined list of qualities that can be easily customised to be smart enough for architecting e-learning solutions. In this context, smart means: specific, measurable, achievable, resource and time bound. Also, using a standard coherent set of precisely defined quality characteristics is appropriate for negotiation with industries, especially in Service Level Agreements for SaaS solutions.

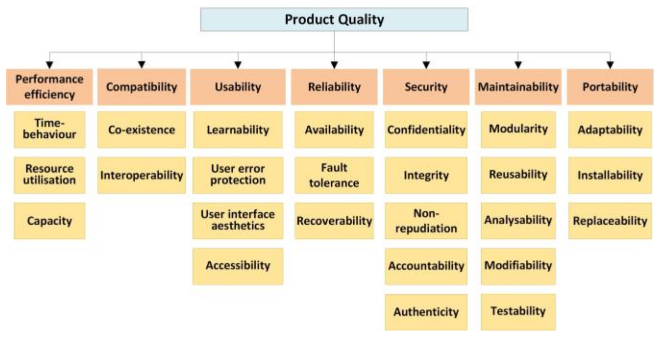

Figure 1. System-related Non-Functional Requirements

The above-depicted NFRs are organised as characteristic (e.g., compatibility) and sub-characteristics (e.g., co-existence and interoperability). The former provides a general definition that does not need to be smart, while the latter (i.e., sub-characteristics) needs further customisation to be smart NFRs. All the above-mentioned product quality-related NFRs are defined [18] in Table 1 below. For readability purpose, different background colour has been given to characteristics (e.g., performance efficiency), while sub-characteristics (e.g., time-behaviour) background colour is white.

TABLE I. SYSTEM-RELATED NON-FUNCTIONAL REQUIREMENTS

Characteristic/ Sub-characteristic	Definition
Performance efficiency	performance relative to the amount of resources used under stated conditions.
Time-behaviour	degree to which the response and processing times and throughput rates of a system, when performing its functions, meet requirements.
Resource utilisation	degree to which the amounts and types of resources used by a system, when performing its functions, meet requirements.
Capacity	degree to which the maximum limits of a system parameter meet requirements
Compatibility	degree to which a system or component can exchange

	information with other systems or components, and/or perform its required functions, while sharing the same hardware or software environment.
Co-existence	degree to which a system can perform its required functions efficiently while sharing a common environment and resources with other products, without detrimental impact on any other product.
Interoperability	degree to which two or more systems or components can exchange information and use the information that has been exchanged.
Usability	degree to which a system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.
Learnability	degree to which a system can be used by specified users to achieve specified goals of learning to use the product or system with effectiveness, efficiency, freedom from risk and satisfaction in a specified context of use.
User error protection	degree to which a system protects users against making errors.
User interface aesthetics	degree to which a user interface enables pleasing and satisfying interaction for the user.
Accessibility	degree to which a system can be used by people with the widest range of characteristics and capabilities to achieve a specified goal in a specified context of use.
Reliability	degree to which a system or component performs specified functions funder specified conditions for a specified period of time.
Availability	degree to which a system or component is operational and accessible when required for use.
Fault tolerance	degree to which a system or component operates as intended despite the presence of hardware or software faults.
Recoverability	degree to which, in the event of an interruption or a failure, a system can recover the data directly affected and re-establish the desired state of the system.
Security	degree to which a system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization.
Confidentiality	degree to which a system ensures that data are accessible only to those authorized to have access.
Integrity	degree to which a system or component prevents unauthorized access to, or modification of, computer programs or data.
Non-repudiation	degree to which actions or events can be proven to have taken place, so that the events or actions cannot be repudiated later.
Accountability	degree to which the actions of an entity can be traced uniquely to the entity.
Authenticity	degree to which the identity of a subject or resource can be proved to be the one claimed.
Maintainability	degree of effectiveness and efficiency in which a system can be modified by the intended maintainers.
Modularity	degree to which a system is composed of discrete components such that a change to one component has minimal impact on other components.
Reusability	degree to which an asset can be used in more than one system, or in building other assets.
Analysability	degree of effectiveness and efficiency in which it is possible to assess the impact on a system of an intended change to one or more of its parts, or to diagnose a product for deficiencies or causes of failures, or to identify parts to be modified.
Modifiability	degree to which a system can be effectively and efficiently modified without introducing defects or degrading existing product quality.
Testability	degree of effectiveness and efficiency in which test criteria can be established for a system or component and tests can be performed to determine whether those criteria have been met.
Portability	degree of effectiveness and efficiency in which a system or component can be transferred from one hardware, software or other operational or usage environment to another.
Adaptability	degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments.
Installability	degree of effectiveness and efficiency in which a system can be successfully installed and/or uninstalled in a specified environment.
Replaceability	degree to which a system can replace another specified software product for the same purpose in the same environment.

Following the above-listed system-oriented Non-Functional Requirements, another complementary set of NFRs is needed to specify the system behaviour in a certain context. Such NFRs describe the quality to which the anticipated system can be used by specific users to meet their demands to achieve specific goals with effectiveness, efficiency, freedom from risk, and satisfaction in specific contexts of use [18]. This complementary list is depicted in Figure 2, and fully described in Table II, as well. Like system-related NFRs, this list is organised as characteristics and sub-characteristics, where the former provides a generic description that does not need to smart, while as the latter needs to be refined to be smart NFRs.

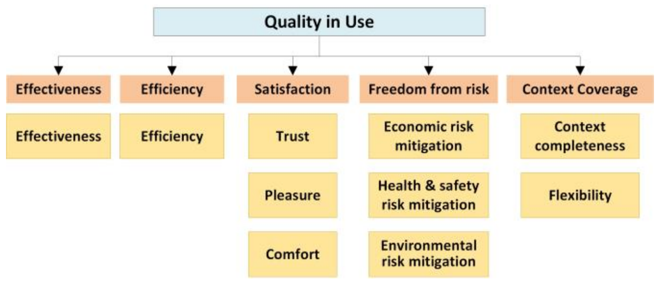

Figure 2. Quality in Use-related Non-Functional Requirements

As depicted in Figure 2, this list is limited to the qualities that can be affected by the context of use. Context of use includes users, tasks, equipment (hardware, software, and material), and the physical and social environments in which a system is used [18]. Some of the system-related NFRs can be affected by the context of use, but generally they are not affected by the context of use.

TABLE II. QUALITY-IN-USE RELATED NON FUNCTIONAL REQUIREMENTS

Characteristic/ Sub-characteristic	**Definition**
Effectiveness	accuracy and completeness in which users achieve specified goals.
Efficiency	resources expended in relation to the accuracy and completeness in which users achieve goals.
Satisfaction	degree to which user needs are satisfied when a system is used in a specified context of use.
Trust	degree to which a user or other stakeholder has confidence that a system will behave as intended.
Pleasure	degree to which a user obtains pleasure from fulfilling their personal needs.
Comfort	degree to which the user is satisfied with physical comfort.
Freedom of risk	degree to which a system mitigates the potential risk to

	economic status, human life, health, or the environment.
Economic risk mitigation	degree to which a system mitigates the potential risk to financial status, efficient operation, commercial property, reputation or other resources in the intended contexts of use.
Health and safety risk mitigation	degree to which a system mitigates the potential risk to people in the intended contexts of use.
Environmental risk mitigation	degree to which a system mitigates the potential risk to property or the environment in the intended contexts of use.
Context coverage	degree to which a system can be used with effectiveness, efficiency, freedom from risk and satisfaction in both specified contexts of use and in contexts beyond those initially explicitly identified.
Context completeness	degree to which a system can be used with effectiveness, efficiency, freedom from risk and satisfaction in all the specified contexts of use.
Flexibility	degree to which a system can be used with effectiveness, efficiency, freedom from risk and satisfaction in contexts beyond those initially specified in the requirements.

Finally, the definitions of the above-mentioned NFRs (i.e., *System*-oriented and *Quality-in-Use*-oriented) are generic enough to accommodate NFRs specifications for a wide range of software systems. For effective TEL system architecture, these NFRs need to be refined to be smart. This means that various thresholds need to be added to these generic definitions based on the NFRs elicitation workshop, explained in Section III. For instance, *Recoverability* will have more specific numbers to describe the conditions in which the system can recover the data affected and re-establish the desired state of the system. This means recoverability requirement specification will look like: "*In the event of interruption or failure, the system must recover the data affected and re-establish the desired state of the system according to the following parameters: (i) Recovery Time Objective (RTO): 30 minutes and (ii) Recovery Point Objective (RPO): three hours*. For clarification, RTO refers to time duration in which users/organisations want to be able to recover the replicated data, while RPO refers to the maximum amount of data that can be lost before causing serious damage to the organisational services. Similarly, *Capacity* NFR needs to be customised with a precise list of parameters, so the refined specification will look like: *the system must be capable of effectively and efficiently performing its functions as expected in the case of having 2500 concurrent users and hosting the contents/activities of 80000 online courses*. In this case, 2500 concurrent users and 80000 courses represent the maximum parameters required by a certain institution. To respond to the early-identified NFRs, a high-level architecture for TEL software system will be proposed in the next section.

IV. The Proposed e-Learning System Architecture

As introduced earlier, NFRs are known as architecturally significant requirements. Ideal software architecture describes the concerned software system through a set of artefacts and relationships between these artefacts. Such artefacts include models, processes, principles, and guidelines that guide the selection, creation, and implementation of software solutions aligned with business requirements. Furthermore, this includes decisions taken during building the high-level architecture of the software system, where these decisions have significant impact on the system quality, performance, availability, etc. [19]. This explains why software architecture is influenced by NFRs, and consequently, justifies investigating NFRs and architecture together. Literature evidence, especially [2], [5]-[7], reveals that the key Non-Functional Requirements that influence software architecture decisions are: performance, compatibility including: co-existence and interoperability, maintainability especially reusability and modularity, adaptation, and flexibility. Moreover, lessons learned from current TEL practices in academic institutions, such as the heavy move towards cloud-based e-learning solutions, puts further emphasis on interoperability and co-existence requirements, because different cloud-based e-learning systems usually share the same hardware environments.

Such requirements can be better met by flexible architecture, such as Service-Oriented Architecture (SOA) that is designed to support flexibility, interoperability, reusability, etc [6]. Therefore, we opt for a service-oriented enabled architecture for e-learning system, where the overall e-learning service is composed of more than one software system, such as LMS (e.g., Atutor, Moodle, etc.), plagiarism checker (e.g., Turnitin), video streaming (e.g., Kaltura), student record system, etc. Some of these systems might be developed in-house, hosted on premises, or provided as a SaaS. Figure 3 depicts the architecture of SOA-enabled e-Learning System. As explained in Figure 3, the proposed architecture is composed of the following three layers: (i) presentation-service layer, that has the necessary set of interfaces to communicate with underneath layers, (ii) business layer, that includes all sub-systems or components (e.g., LMS and video streaming), and (iii) data layer that hosts every possible source of data.

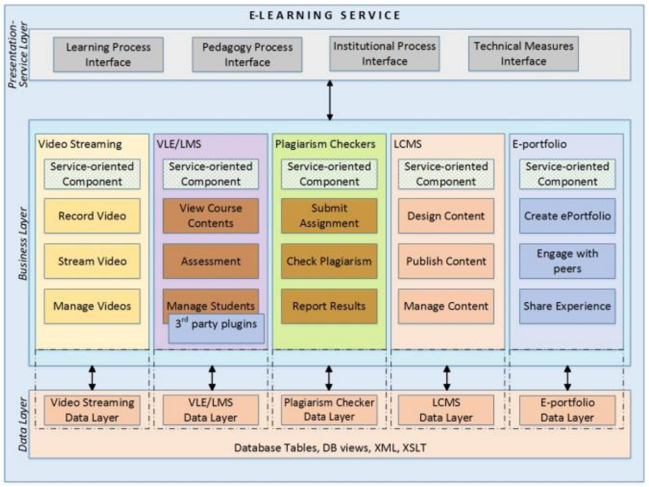

Figure 3. Service-enabled e-Learning System Architecture

The first layer, presentation-service layer, includes the following four components. *The first component*, learning process interface, manages and monitors all learning and teaching processes carried out by students. Such processes mainly include LMS capabilities, such as finding learning contents and managing e-learning activities. To better facilitate this component's job, there is a need to improve the architecture/design of the current LMSs via adding what we call here "*Service-oriented component*". This component allows flexible access to the internal capabilities of the LMS (e.g., looking for certain contents of a particular course or analysing various activities done by a group of users for learning analytics purposes). This component needs significant changes to be introduced to the LMS stretching from architectural design of the intended LMS through specific algorithms that can identify and discover web services that meet users' demands.

The second component, pedagogy process interface, handles learning content through all of its stages (i.e., design, development, publishing, etc.) This can be done through interaction with Learning Content Management System (LCMS) component in the business layer. This component retains complex processes since designing learning contents includes various pedagogical approaches that considerably vary. For instance, designing behavioural-based content, which is instructor-centred contents that contain: a) learning objectives, b) learning contents, and c) assessment exercise is different from designing a social constructive-based content which is driven by students' interactions. Both types of contents are based on underpinning pedagogical models that can be represented via a set of processes that explain the workflow needed to design, develop, and publish learning contents. Similarly, service-oriented component, for LCMS, is needed here to make this process achievable via web services.

The third component, institutional process interface, handles all institutional processes that are related to e-learning, such as assigning roles to e-learning actors (e.g., module leader, instructor, and examiners). This also includes students' enrolments, other administration tasks, tracking other related processes (e.g., financial processes). The automatic execution of these processes is challenging because most institutions have their own business rules that could be complicated due to the wide range of programmes offered by universities and the adopted service models. This component will communicate, via web services, with Human Resources (HR) systems and students' record systems that can provide the necessary information to achieve this task.

The fourth component, integration interface, handles all technical aspects needed for successful e-learning services. One of the most important aspects here is the security because e-learning service, as introduced earlier, is a hybrid service that may combine on-premises software, public cloud, and private cloud. Also, considering the evolving requirements for academic institutions is highly important. This requires continuous monitoring for the current e-learning services, such as doing performance testing and penetration testing. This allows benchmarking for the current level of service, so the institution can investigate the impact of adding additional components to the e-learning service.

As described earlier, each of the above-mentioned interface component, in the presentation layer, liaises with one or more service-oriented component in the concerned sub-system in the business layer. For instance, institutional interface might liaise with one or more than one service component to setup the proper plagiarism check processes that might be dynamic as they differ from undergraduate to postgraduate or lifelong learning programme. This applies to models/tools that use specific learning approaches (e.g., Game-based Learning model [20]). In addition, certain arrangements need to be done at the data level to make sure data are accessible by permitted stakeholders whenever is needed. Despite the fact that Non-Functional Requirements are more persistent, with little changes are expected, there is a need to manage the changes that could happen over TEL software development life cycle. Therefore, it is recommended to use suitable requirement management tool or model to keep the e-learning service reliable and efficient. This includes reviewing the current set of requirements either based on agreed timeframe or whenever we have new requirements from stakeholders. Finally, it is worth mentioning that the early-identified business layer might have extra sub-systems based on the Functional Requirements coming from different departments in the academic institution.

V. Conclusion and Future Work

In this paper, we handled the challenging problem of managing Non-Functional Requirements, more specifically eliciting and specifying NFRs, in the context of TEL. Lessons learned from TEL practices revealed that NFRs are ignored due to many reasons, which could seriously impact the overall e-learning system/service. Therefore, we opt for a comprehensive approach based on ISO 25010 to elicit and specify Non-Functional Requirements. Furthermore, this paper presented flexible service-oriented architecture for e-learning software systems that can better meet the required capabilities (i.e., Functional and Non-Functional Requirements). This work revealed the need to adopt open and flexible architecture for TEL systems. This means that these systems should be designed in a way that is accessible via web service mechanism to allow further agility. Moreover, it highlighted the need to develop service identification and service discovery algorithms that consider e-learning particularities.

References

[1] R. Hammad, M. Odeh and Z. Khan, "Towards a generic requirements model for hybrid and cloud-based e-learning systems," The IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom), 2013, pp. 106-111, Bristol, UK.

[2] J. Eckhardt, A. Vogelsang and D. M. Fernández, "Are non-functional requirements really non-functional? an investigation of non-functional requirements in practice," The 38th IEEE/ACM International Conference on Software Engineering (ICSE), 2016, pp. 832-842.

[3] S. Kumar, A. K. Gankotiya and K. Dutta, "A comparative study of Moodle with other e-learning systems," The 3rd International Conference in Electronics Computer Technology (ICECT), 2011, 414-418.

[4] S. Graf and B. List, "An evaluation of open source E-learning platforms stressing adaptation issues," The 5th IEEE

International Conference on Advanced Learning Technologies (ICALT), 2005, 163-165.

[5] D. Ameller et al., "Non-functional requirements in architectural decision making," IEEE Software, vol. 30, (2), pp. 61-67, 2013, doi:10.1109/MS.2012.176.

[6] L. Chen, M. A. Babar and B. Nuseibeh, "Characterizing architecturally significant requirements," IEEE Software, vol. 30, (2), pp. 38-45, 2013, doi: 10.1109/MS.2012.174.

[7] C. Miksovic and O. Zimmermann, "Architecturally significant requirements, reference architecture, and metamodel for knowledge management in information technology services," The 9th IEEE/IFIP Conference on Software Architecture (WICSA), 2011, pp. 270-279, doi: 10.1109/WICSA.2011.43.

[8] D. Pandey and V. Pandey, "Importance of requirement management: a requirement engineering concern," International Journal of Research and Development A Management Review, vol. 1, (1), pp. 66-70, 2012. ISSN (Print): 2319–5479.

[9] H. Men, J. Liu and J. Han, "Applied research on Atutor," The International Conference on E-Learning, E-Business, Enterprise Information Systems, and E-Government (IEEEE'09), 2009, pp.107-110. ISBN: 978-0-7695-3907-2.

[10] T. Acosta and S. Luján-Mora, "Comparison from the levels of accessibility on LMS platforms that supports the online learning system," The 8th Annual International Conference on Education and New Learning Technologies, 2016, pp. 2704-2711, doi: 10.21125/edulearn.2016.1579.

[11] I. Vlasin and C. Chirila, "Online contest based on integration of activities, adaptability and students cooperation using Ilias LMS," The International Scientific Conference eLearning and Software for Education, 2016, pp. 67-74.

[12] K. Logan and T. Neumann, "Comparison of Blackboard 9.1 and Moodle 2.0," Learning Technologies Unit. University of London, London, UK, 2010.

[13] R. D. Rucker and L. R. Frass, "Migrating Learning Management Systems in Higher Education: Faculty Members' Perceptions of System Usage and Training When Transitioning from Blackboard Vista to Desire2Learn," Journal of Educational Technology Systems, vol. 46, (2), pp. 259-277, 2017, doi: 10.1177/0047239517711954.

[14] D. Dagger et al., "Service-oriented e-learning platforms: from monolithic systems to flexible services," IEEE Internet Computing, vol. 11, (3), pp. 28-35, 2007, 10.1109/MIC.2007.70.

[15] A. G. Booth and B. P. Clark, "A service-oriented virtual learning environment," On the Horizon, vol. 17, (3), pp. 232-244, 2009, doi:10.1108/10748120910993268.

[16] M. R. Barbacci et al., "Quality attribute workshops (QAWs)," International society for Bioelectricity, Shreveport L.A., 2003.

[17] R. Hammad, M. Odeh and Z. Khan, "Towards a model-based approach to evaluate the effectiveness of e-learning," The 9th European Conference on IS Management and Evaluation – ECIME, UK, Bristol, 2015, pp. 111-119, ISBN: 978-1-910810-55-2.

[18] ISO/IEC 25010 "Systems and software engineering–Systems and software quality requirements and evaluation (SQuaRE)–System and software quality models," The International Standard Organisation (ISO), 2011.

[19] N. Medvidovic and R. N. Taylor, "Software Architecture: Foundations, Theory, and Practice," John Wiley & Sons, 1st edition, 2010, ISBN: 9780470167748.

[20] R. Hammad "Game-Enhanced and Process-Based e-Learning Framework," In: Tian F., Gatzidis C., El Rhalibi A., Tang W., Charles F. (eds) E-Learning and Games, July 2017, Lecture Notes in Computer Science, vol 10345, Springer, Cham. pp. 279-284, doi:10.1007/978-3-319-65849-0_30.