ALPHAS OF LOW CODE DEVELOPMENT PROJECT: BEEAPEX CASE

ROBERT LESKOVAR,[1] WIELAND SCHWINGER,[2] WERNER RETSCHITZEGGER,[2] ALKIVIADIS TSIMPIRIS,[3] ATHANASIS ANGEIOPLASTIS,[3] DIJANA OREŠKI,[4] VJERAN STRAHONJA,[4] MICHAL KVET,[5] KAROL MATIAŠKO,[5] MONIKA SOŃTA,[6] JACEK MAŃKO,[6] ALENKA BAGGIA[1]

[1] University of Maribor, Faculty of Organizational Sciences, Kranj, Slovenia
robert.leskovar@um.si, alenka.baggia@um.si
[2] Joannes Keppler University, Johannes Kepler University, Institute for telecooperation, Department of Cooperative Information Systems, Linz, Austria
wieland.schwinger@jku.at, werner@ifs.uni-linz.ac.at
[3] International Hellenic University, Department of Computer, Informatics Telecommunications Engineering, Serres, Greece
atsimpiris@ihu.gr, aagiop@gmail.com
[4] University of Zagreb, Faculty of Organization and informatics, Varaždin, Croatia
dijana.oreski@foi.unizg.hr, vjeran.strahonja@foi.hr
[5] University of Žilina, Faculty of Management Science and Informatics, Žilina; Slovakia
michal.kvet@uniza.sk, karol.matiasko@uniza.sk
[6] Kozminski University, Department of Management in the Networked & Digital Societies, Warsaw, Poland
msonta@kozminski.edu.pl, jmanko@kozminski.edu.pl

Abstract Software projects of all kinds should deliver results according to agreed term and conditions. However, they are prone to de-rail and be terminated without satisfactory and in-time outputs. According to the literature review most failures in software project occurs before producing a source code. The first part of the paper gives a short overview of Essence – a generalized and novel approach to construct a tailored software development methodologies and the abstract levels of project health attributes - alphas. The ultimate goal for project team is to provide high quality software under the given constraints. Examination of the alphas which present the kernel of Essence is performed on specific low-code development Erasmus+ project BeeAPEX. Then we present BeeAPEX project alphas. Finally, the states of the alphas in observed project are estimated to get an overview of the project progress and to identify the gaps between plan and realization.

Keywords: essence, software engineering, BeeAPEX, low-code programing

University of Maribor Press

DOI https://doi.org/10.18690/um.fov.3.2023.44
ISBN 978-961-286-722-5

1 Introduction

Evaluation of the success of software development projects is most often limited to the three aspects: time, costs, scope. The literature in software quality often addresses a project diamond that include: time, quality, scope and cost (e.g., Akbar et al., 2017). But project diamond can be evaluated when the project is finished. Software, as defined by ISO, IEC and IEEE (see ISO/IEC/IEEE Std. 90003:2014 (ISO/IEC/IEEE, 2014) is a »collection of components necessary to ensure proper operation, and efficient maintenance during its life cycle. The components are: 1) computer programs (code), 2) documentation, 3) data necessary for its operation and maintenance (including standard tests), and 4) procedures« (Galin, 2018).

The issue of the success of software development projects has been known in the discipline of software engineering for decades. In the famous article "Critical Success Factors in Software Projects", Reel (1999) states that majority of failed projects fall into ten categories: project managers do not understand the user's needs, the scope of the project is inadequately defined, project changes are poorly managed, the chosen technology changes during the course of the project changes, business needs change, production deadlines are unrealistic, users resist the solution, the project sponsor disappears, lack of competence in the development team and project managers ignore good practices. Among these ten reasons, there are seven of them, which are noticeable at the moment when not a single line of code is written. More than two decades later top reasons for failed projects did not disappear. So we are wandering if there is a mean to detect project derailment during the project and not only post-mortem.

An article by Dendere (Dendere et al., 2021), citing media reports and professional literature, finds a global trend of failed healthcare projects related to digital transformation. The main cause is supposed to be project management, more precisely the choice of a suitable project management approach ("Adopting a suitable project management approach is a major factor for achieving success because managing a project using an unsuitable methodology can severely damage the chances of success."). The article implicitly favors agile approaches over traditional ones, but without empirical evidence.

Rasheed (Rasheed et. al., 2021) considers the greater success of software projects that used the agile development method not be the direct result of agility. The authors claim that the high proportion of successful software development projects using agile methods is more the result of the size of the projects (or rather small size). There are also few examples of successful use of the agile approach in large, complex projects. Authors highlight the use of the SAFe (Scaled Agile Framework) agile methodology. In their opinion, requirements engineering is a key part of the project life cycle, although they state, that they failed to identify the cause of as many as 50% of unsuccessful projects. Among identified causes of failed software projects are: lack of technical competence (13%), incomplete requirements (12%), changing requirements (12%), poor user engagement (7%) and poor project initiation (6%).

In the past decades, the supremacy battle of development methodologies has always been fought between the traditional methodologies based on the life cycle and "novelties" such as agile development, SCRUM, XP, FDD, TDD and SAFe. But many studies have shown that the practice, the way the developers work, is the most important for the success of the project. Organizations that reach a higher level of maturity of the development process (Chaudhary & Chopra, 2017) are more likely to complete the project and satisfy customers' needs within the estimated time and with the estimated resources. Good practices also include software engineering standards. Examples are: requirements specification in the standard ISO/IEC 29148:2018 (ISO/IEC/IEEE, 2018) or software quality attributes and quality in use in ISO/IEC 25010:2017 (ISO/IEC, 2017). The discipline of software engineering needs a general, flexible and accepted methodology, which should not tie developers, but should enable them to work efficiently and effectively. We will therefore consider one such approach – Essence that enables the construction of a software development methodology, which is: tailored to development team, free of ties to any methodology and free to utilize any efficient method or practice and enables the assessment of the software development progress during the course of the project.

2 Essence

Essence originates from the SEMAT initiative (Software Engineering Method and Theory), which brought together a group of experts with the aim to establish the field of software engineering as a rigorous scientific discipline. Based on their work, the international non-profit consortium OMG (Object Management Group) produced a standard that defined common elements, a language and a framework for creating methods in software engineering (Park et al., 2018).

Jacobson et al. (2019) consider Essence as a real milestone in the field of software engineering. The most recent version 1.2 was released in 2018 by OMG (Object Management Group, 2018). Both SEMAT and Essence originate from the issues of the software engineering discipline – the search for mature software development practices that result in successful projects and a high-quality product. The source estimates that there are around 20 million developers worldwide and over 100 thousand different methods. Almost every group of developers works in their own way. New methods are constantly appearing. One such example is the A-Z Model - the improvement of the development process and consequently the quality of the software, (Akbar et al., 2017). It is based on the SDLC (traditional) approach. The authors claim that more accurate time-boxing of activities in each phase of the life cycle increase the probability of a successful project and a quality product. The most important features of the Essence – kernel and language are independent of the approach, methodology or methods. In Chapter 4 of Essence (Object Management Group, 2018) alpha is defined as: **An essential element of the software engineering endeavor that is relevant to an assessment of the progress and health of the endeavor. Alpha is an acronym for an Abstract-Level Progress Health Attribute.** The associations between seven alphas are depicted on Figure 1.

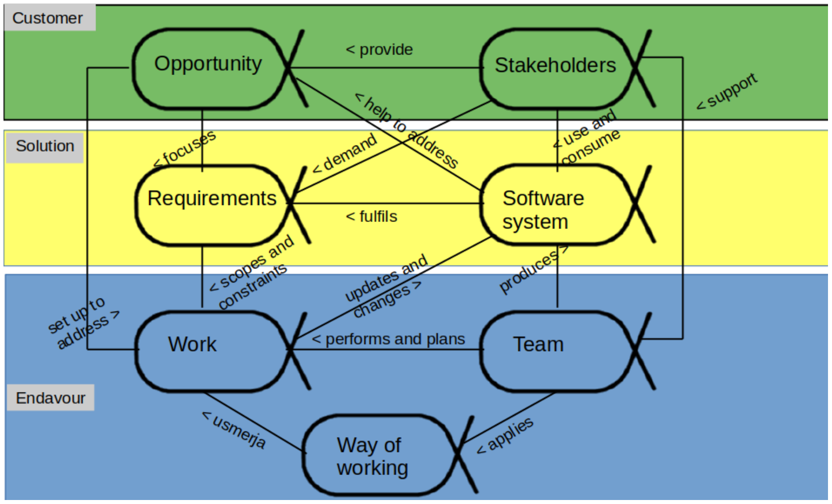

Figure 1: The associations between kernel alphas (source: Object Management Group, 2018)

Essence (Object Management Group, 2018) define alphas states and conditions for transition as follows:

Stakeholders: *Recognized* (Stakeholder groups identified; Key stakeholder groups represented; Responsibilities defined), *Represented* (Responsibilities agreed; Representatives authorized; Collaboration approach agreed; Way of working supported & respected), *Involved* (Representatives assist the team; Timely feedback and decisions provided; Changes promptly communicated), *Agreement* (Minimal expectations agreed; Rep's happy with their involvement; Rep's input valued; Team's input valued; Priorities clear & perspectives balanced), *Satisfied for Deployment* (Stakeholder feedback provided; System ready for deployment) and *Satisfied in Use* (Feedback on system use available; System meets expectations)

Opportunity: *Solution Needed* (Solution identified; Stakeholders' needs established; Problems and root causes identified; Need for a solution confirmed; At least one solution proposed), *Value Established* (Opportunity value quantified; Solution impact understood; System value understood; Success criteria clear; Outcomes clear and quantified). *Viable* (Solution outlined; Solution possible within constraints; Risks acceptable & manageable; Solution profitable; Reasons to develop solution understood; Pursuit viable), *Addressed* (Opportunity addressed; Solution worth deploying; Stakeholders satisfied), *Benefit Accrued* (Solution accrues benefits; ROI acceptable)

Requirements: *Conceived* (Stakeholders agree system is to be produced; Users identified; Funding stakeholders identified; Opportunity clear), *Bounded* (Development stakeholders identified; System purpose agreed; System success clear; Shared solution understanding exists; Requirements format agreed; Requirements management in place; Prioritization scheme clear; Constraints identified & considered; Assumptions clear), *Coherent* (Requirements shared; Requirements' origin clear; Rationale clear; Conflicts addressed; Essential characteristics clear; Key usage scenarios explained; Priorities clear; Impact understood; Team knows & agrees on what to deliver), *Acceptable* (Acceptable solution described; Change under control; Value to be realized clear; Clear how opportunity addressed; Testable), *Addressed* (Enough addressed to be acceptable; Requirements and system match; Value realized clear; System worth making operational), *Fulfilled* (Stakeholders accept requirements; No hindering requirements; Requirements fully satisfied)
Software System: *Architecture Selected* (Architecture selection criteria agreed; HW platforms identified; Technologies selected; System boundary known; Decisions on system organization made; Buy, build, reuse decisions made; Key technical risks agreed), *Demonstrable* (Key architectural characteristics demonstrated; System exercised & performance measured; Critical HW configurations demonstrated; Critical interfaces demonstrated; Integration with environment demonstrated; Architecture accepted as fit-for-purpose), *Usable* (System can be operated; System functionality tested; System performance acceptable; Defect levels acceptable; System fully documented; Release content known; Added value clear), *Ready* (User documentation available; System accepted as fit-for-purpose; Stakeholders want the system; Operational support in place), *Operational* (System available for use; System live; Agreed service levels supported), *Retired* (Replaced or discontinued; No longer supported; No authorized users; Updates stopped)

Team: *Seeded* (Mission defined; Constraints known and defined; Growth mechanisms in place; Composition defined; Responsibilities outlined Required commitment level clear; Required competencies identified; Size determined; Governance rules defined; Leadership model selected), *Formed* (Enough members recruited; Roles understood; How to work understood; Members introduced;

Individual responsibilities accepted and aligned to competencies; Members accepting work; External collaborators identified; Communication mechanisms defined; Members commit to team), *Collaborating* (Works as one unit; Communication open and honest; Focused on mission; Members know each other), *Performing* (Consistently meeting commitments; Continuously adapting to change; Addresses problems; Rework and backtracking minimized; Waste continuously eliminated), *Adjourned* (Responsibilities fulfilled; Members available to other teams; Mission concluded)

Work: *Initiated* (Required result clear; Constraints clear; Funding stakeholders known; Initiator identified; Accepting stakeholders known; Source of funding clear; Priority clear), *Prepared* (Commitment made; Cost and effort estimated; Resource availability understood; Risk exposure understood; Acceptance criteria established; Sufficiently broken down to start; Tasks identified and prioritized; Credible plan in place; Funding in place; At least one team member ready; Integration points defined), *Started* (Development started; Progress monitored; Definition of done in place; Tasks being progressed), *Under Control* (Tasks being completed; Unplanned work under control; Risks under control; Estimates revised to reflect performance; Progress measured; Re-work under control; Commitments consistently met), *Concluded* (Only admin tasks left; Results achieved; Resulting system accepted), *Closed* (Lessons learned; Metrics available; Everything archived; Budget reconciled & closed; Team released; No outstanding, uncompleted tasks)

Way of Working: *Principles Established* (Team actively support principles; Stakeholders agree with principles; Tool needs agreed; Approach recommended; Operational context understood; Practice & tool constraints known), *Foundation Established* (Key practices & tools selected; Practices needed to start work agreed; Non-negotiable practices & tools identified; Gaps between available and needed way of working understood; Gaps in capability understood; Integrated way of working available), *In Use* (Practices & tools in use; Regularly inspected; Adapted to context; Supported by team; Feedback mechanisms in place; Practices & tools support collaboration), *In Place* (Used by whole team; Accessible to whole team; Inspected and adapted by whole team), *Working Well* (Predictable progress being made; Practices naturally applied; Tools naturally support way-of-working; Continually tuned), *Retired* (No longer in use; Lessons learned shared)

3 BeeAPEX project

BeeAPEX is an acronym for the BEE with APEX (Better Employability for Everyone with APEX) Erasmus+ project which aims to support the digital transformation of higher education institutions through the development of the digital readiness, resilience and capacity of educators and students. Modernization of IT curricula and courses in the areas of front-end and back-end design of applications and databases can build up higher education institutions' capacities and adopting a more inclusive approach to digital literacy. Duration of the project is from 01. 11. 2021 to 31. 10. 2023. Expected results are: an advanced, open access, extra-curricular bachelor-level course on low-code development of front- and back-end applications and databases (lectures, practical exercises, assessments, exams, and independent learning for a total effort of 75 hours); documentation needed to get the course recognized and awarded with 3 ECTS; an e-book on low-code programming with APEX, a short course for independent learning on low-code development of front- and back-end applications and database; a 5-day training for professors, lecturers and postdoctoral teaching assistants; pre-recorded webinars on industry and employment trends; case studies created jointly with local SMEs on low-code programming; academic articles; a project website with all relevant information about the project activities, outputs and results,; a showcase of selected students' project work; dissemination and multiplier activities; more inclusive and student-centered IT education. Let's make and overview of Alphas.

3.1 Stakeholders

Essence definition: The people, groups, or organizations who affect or are affected by a software system.

The project consortium comprises six academic partners which are actively involved in project implementation. Oracle Academy is associate partner which supports the project through the provision of free services, consultations and dissemination of results. IT sector is consumer of graduates. It's role as reviewer and adviser makes feedback loop between development team and economy. Teachers and students as the target audiences have a tester role by using learning material and providing qualitative and quantitative feedback to the developers. Developers also use the project results. One important stakeholder is national agency (CMEPIUS) which represents European Commission as a financer of the Erasmus+ program and this particular project.

3.2 Opportunity

Essence definition: The set of circumstances that makes it appropriate to develop or change a software system.

According to the European Commission's Higher Education Modernization Agenda, higher education should enhance individual potential and should equip graduates with the knowledge and core transferable competences they need to succeed in high-skill occupations. The agenda identifies the slow updating of curricula to the changing needs in the wider economy as one of the challenges higher education institutions need to resolve. The BeeAPEX project addresses the discrepancy between graduates' knowledge and the IT skills demanded on the labor market. In order to reach and educate a higher number of students, the provision of IT training in higher education should be reconsidered to expand to an increasing number of students enrolled in non-technical study programs, and the use of digital and blended learning tools and methodologies enabling self-regulated and collaborative learning activities should be enhanced. The possession of development and coding skills is traditionally associated with concrete IT professions such as front-end developer, back-end developer and web developer. Because of the advancement of the digital economy, however, these skills are increasingly needed and used in a wide range of other professions in which case they are complementary, not primary, for the performance of one's job. The project's objective is to contribute to the digital transformation of six higher education institutions through the development of the digital capabilities of their educators and students, in particular through the development of blended learning resources and the development of novel teaching and collaborative capacities. Firstly, the digital capabilities of teaching and research staff is to be enhanced through the organizing of training on and expertise-exchange in the areas of digital course creation, low-code programming, and front-end and back-end development of applications and databases. Secondly, the project is to contribute to the more inter-connected development of curriculum and blended learning in the area of low-code programming. The main project result would be the creation of new courses (a short digital and an advanced extra-curricular course) available for all bachelor-level students enrolled at the higher education institutions. On business level, it would enable local businesses to have access to a better educated workforce.

3.3 Requirements

Essence definition: What the software system must do to address the opportunity and satisfy the stakeholders.

It is necessary to provide high quality education material on low-code programming (in Oracle Application Express – APEX), which include textbook, software, data and videos. Textbook must provide student of any discipline clear guides to:

- get started with APEX low-code environments, prepare a database, navigate through APEX, exchange data, generate web and mobile application, manage reports, forms and menus, facilitate teamwork in APEX, find the benefit of sample and starter applications and to manage packaged and multilingual web applications in APEX
- develop own application inspired by twelve cases from real life of different sectors of economy. Each case must include: business view, problem definition, use case textual, semi-structured and graphical presentation, data model development process (narrative description, logical, relational, physical) via forward and possibly reverse engineering and finally interface design. Each business case must have developed supplementary learning material: exported application and video guides. Exported application must enable teachers and students to easy install and study the code and the data.

All textbook chapters must include at least three questions and answers which could be easy modified into questions for quizzes. Twelve business cases provide the foundation for flexible design of short courses on low-code web application development depending on the user (teacher, student) background, competences and available time. The most common form of short course would involve 25 hours of student effort while in marginal situation only 1 hour could be practiced.

Other project deliverables are less software dependent as they are means of dissemination of main project results.

3.4 Software system

Essence definition: A system made up of software, hardware, and data that provides its primary value by the execution of the software.

Twelve business cases included as textbook chapters result in low-code web applications. Development environment is Oracle APEX as one of the most viable and capable tools. The constraint is that application can't run on other databases, however the argument of always free availability to anyone, resolves the risk of locking to one vendor. Primary value of each application is that principles of data driven web application development in different contexts can be transferred thus enabling students to reuse and enhance the competences in the real business environment. Applications are focused on: internet news for employees, catalogue of local plants, user authorization and management, small innovation system, business process management, exchange of plants and seeds, book review management system, calculation of bill of material, nutrition and diet management, office hours scheduling, telecommunication services billing and car rental. Each application uses distinct database tables to prevent the data mess.

3.5 Team

Essence definition: A group of people actively engaged in the development, maintenance, delivery or support of a specific software system.

The team members are affiliated in six countries and universities. Each project partner organized sub teams of two to five web developers, reviewers, testers, documentation writers, designers, database administrators. Due to physical distance are rare live meetings, the heavy use of videoconferencing and team collaboration systems is a must. BeeAPEX web site is founded on Moodle LMS with integrated BBB videoconferencing system. Currently there are 44 registered users, grouped by participating institutions and stakeholders.

3.6 Work

Essence definition: Activity involving mental or physical effort done in order to achieve a result.

The main BeeAPEX project activities are:

1. low-code web application development involves designing, database modelling, database administration, database programming, application testing (own)
2. application documentation, reviews and application testing (others)
3. reviewing of textbook chapters, supplementary materials, videos
4. project administration, coordination among members, task progress monitoring etc.

3.7 Way of working

Essence definition: The tailored set of practices and tools used by a team to guide and support their work.
Development environment include the following set of practices and tools:

1. Oracle APEX as a primary tool for development. Besides the Oracle Academy development workspace can be obtained: a) free workspace on apex.oracle.com, b) free instance within OCI, Oracle Cloud Infrastructure, c) on-premise APEX d) developer virtual machine or e) docker. The advantage of apex.oracle.com and OCI is that the latest version (at the time of writing the version was 22.2) is on disposal. No backward compatibility is assured.
2. Oracle JDeveloper Studio Edition 12.2 and higher to draw use-case diagrams.
3. SQL Developer Data Modeler 21.4.2 and higher develop logical and relational data models. Since it allows forward and reverse engineering, developer can generate diagrams from SQL scripts.
4. git 2.35 on developers' computers

5. dedicated GitLab server as remote repository. Repository contains all developer branches, which are merged into origin.
6. Texstudio or TexMaker for writing documentation
7. Video capturing program and video editor. Recommended video capture programs are Free Cam (for Windows) and Kazam (Linux) while video editor is KDEnlive (Windows and Linux)
8. Each BeeAPEX project member and contributors to the textbook can use remote desktop accounts where all development tools are installed.

Git remote repository enables accelerated delivery of main project results. Textbook chapters are assigned to project partners (3-5 each). Each partner is to provide translation of the textbook in local language.

4 Estimation of BeeAPEX project Alphas

Twelve months after kick-of meeting, three live meetings, a lot of communication (videoconferences, email, phone, collaboration via Moodle) and developers endeavor, our estimation of the project alphas can be summarized:

Stakeholders: *Agreement* (Minimal expectations agreed; Rep's happy with their involvement; Rep's input valued; Team's input valued; Priorities clear & perspectives balanced). The project consortium is stable, Oracle Academy continuously support dissemination of the intermediate results, IT sector is involved through advising and providing feedback. Teachers and students are included in testing. Developers exchange knowledge. National agency (CMEPIUS) provides adequate support for this particular project.

Opportunity: *Addressed* (Opportunity addressed; Solution worth deploying; Stakeholders satisfied). Procedure of accreditation of extracurricular course started by project partners where applicable. Some partners decided to include developed contents in existing already accredited courses which exceeds initial expectations of the project. This project already contributes to the digital transformation of six higher education institutions through the development of the digital capabilities of their educators and students.

Requirements: *Addressed* (Enough addressed to be acceptable; Requirements and system match; Value realized clear; System worth making operational). At the time of writing, we approximate that more than half of requirements are already fulfilled. Most of chapters in the textbook are under peer review. One chapter is not submitted yet, all twelve applications are submitted. Some reworks and preparation of supplementary learning material is expected. Provided results already enable teacher to design short courses.

Software System: *Demonstrable* (Key architectural characteristics demonstrated; System exercised & performance measured; Critical HW configurations demonstrated; Critical interfaces demonstrated; Integration with environment demonstrated; Architecture accepted as fit-for-purpose).

Twelve applications are ready for demonstration and all agreed areas of the textbook are covered.

Team: *Collaborating* (Works as one unit; Communication open and honest; Focused on mission; Members know each other). The team members are collaborating and utilizes established environment. Overall, no major changes in team occurs. One project partner had issue with skilled developer however with the support of other team members, the progress is noticeable.

Work: *Under Control* (Tasks being completed; Unplanned work under control; Risks under control; Estimates revised to reflect performance; Progress measured; Re-work under control; Commitments consistently met). Low-code web applications are development and all undelaying activities performed. Application documentation is in progress. At the time of writing more than 300 pages are submitted in agreed form. Regular video meetings are held, communication via phone, emails and collaboration tools in Moodle is appropriate.

Way of Working: *Working Well* (Predictable progress being made; Practices naturally applied; Tools naturally support way-of-working; Continually tuned). Working with git presented new way of working and some team members had no prior experience. Therefore, some fear at the beginning of the project was present. Also, Latex environment posed extra effort for some team members. Despite remote desktop environment is available, but the utilization is low because developers prefer working on their equipment. Git remote repository integrates all software artefacts satisfactory. Current size of the repository is 238 MB in 9 branches of English version has 109 directories and 806 files.

5 Conclusions

Essence as a generalized and novel approach to construct a tailored software development methodologies and toolset for monitoring the project progress was applied for this research. Alphas of Essence were easy to understand and apply in the particular BeeAPEX project. We applied only pre-defined alphas and conclude they describe the most important aspects of low-code development project well. We are confident that Essence provides efficient and effective approach for Erasmus+ projects with any kind of software deliverables. Estimation of the state of the alphas is rigorous. All conditions must be meet to progress to the next state (level). Such approach is conservative and known from SEI Capability Maturity Model on. Our estimation of BeeAPEX project alphas was straight forward and fast. Visualization of the states would further contribute to usability with less effort and with greater impact. The estimation of the states of alphas twelve months after project kick-of gives the team some evidence-based confidence that project progresses according to plan.

Acknowledgements

The authors would like to acknowledge the support given by the European Commission through the Action Erasmus + Better Employability for Everyone with APEX (project ID 2021-1-SI01-KA220-HED-000032218), co-funded by the Erasmus+ programme of the European Union. The European Commission support for the production of this publication does not constitute endorsement of the contents which reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Also, authors express acknowledgement to Oracle Academy Program Manager Mr. Darko Jureković for continuous support in project results dissemination.

References

Akbar, M. A., Sang, J., Khan, A. A., Fazal-E-Amin, Nasrullah, Shafiq, M., Hussain, S., Hu, H., Elahi, M., & Xiang, H. (2017). Improving the quality of software development process by introducing a new methodology-Az-model. IEEE Access, 6, 4811–4823. https://doi.org/10.1109/ACCESS.2017.2787981

Chaudhary, M., & Chopra, A. (2017). CMMI for Development. In CMMI for Development. https://doi.org/10.1007/978-1-4842-2529-5

Dendere, R., Janda, M., & Sullivan, C. (2021). Are we doing it right? We need to evaluate the current approaches for implementation of digital health systems. Australian Health Review, 778–781. https://doi.org/10.1071/AH20289

Galin D. (2018). Software Quality: Concepts and Practice. IEEE Computer Society, John Wiley & Sons, Hoboken, NJ

ISO & IEC. (2017). ISO/IEC 25010 Software Quality Model. (https://www.iso.org/standard/35735.html)

ISO/IEC/IEEE. (2018). ISO / IEC / IEEE 29148 Systems and software engineering — Life cycle processes - Requirements engineering.

Jacobson, I., Lawson, H., Ng, P.-W., McMahon, P. E., & Goedicke, M. (2019). The Essentials of Modern Software Engineering.

Object Management Group. (2018). Kernel and Language for Software Engineering Methods (Essence). 2007 4th IEEE International Workshop on Visualizing Software for Understanding and Analysis, Versión 1.2, 300. https://www.omg.org/spec/Essence/1.2

Park, J. S., Jang, J., & Lee, E. (2018). Theoretical and empirical studies on essence-based adaptive software engineering. Information Technology and Management, 19(1), 37–49. https://doi.org/10.1007/s10799-016-0273-5

Rasheed A., Shehryar T., Aiman Aslam N., Zafar B. (2021). Requirement Engineering Challenges in Agile Software Development. Mathematical Problems in Engineering (2021, May). DOI:10.1155/2021/6696695

Reel J.S. (1999), Critical success factors in software projects, IEEE Software, vol. 16, no. 3, pp. 18-23, May-June 1999, doi: 10.1109/52.765782.