International Journal of Scientific Research in Computer Science, Engineering and Information Technology
ISSN : 2456-3307 (www.ijsrcseit.com)
doi :https://doi.org/10.32628/IJSRCSEIT

Enhancing Software Development Lifecycle with Agile Practices

Chinmay Mukeshbhai Gangani
Independent Researcher, USA

Article Info

Publication Issue :
Volume 3, Issue 7
September-October-2018
Page Number : 555-563

Article History
Accepted: 25/09/2018
Published: 30/10/2018

ABSTRACT

In addition to attempting to ascertain the course of Agile practices' ongoing development, the study aims to comprehend how Scrum may be successfully modified to satisfy the requirements of various organizational settings, particularly in situations involving remote and dispersed teams. Since software develops throughout the course of its life cycle, security considerations should be given particular consideration in software development approaches. The purpose of this article is to equip agile techniques with security actions. The environmental elements of sustainability in computer hardware are addressed by several Green IT initiatives, but there aren't many models, descriptions, or realizations in the field of computer software. According to the research, the evolutionary approach with agile methods has many advantages, including improved customer satisfaction, reduced documentation, higher customer interaction, increased capacity to include emergent needs, and improved risk management skills. To optimize its advantages, we must, however, implement and closely observe such procedures. ICT power usage continues to rise. It is still unclear whether or not the energy savings from ICT outweigh the energy consumption from ICT. The environmental elements of sustainability in computer hardware are addressed by several Green IT initiatives, but there aren't many models, descriptions, or realizations in the field of computer software. In this study, we suggest a general improvement to the software development process that might include sustainability considerations into any software development methodology. This study examines recent research, emphasizes important discoveries, and points out areas that need further investigation. This study seeks to give a better knowledge of the possibilities and difficulties of artificial intelligence (AI) in software engineering via a thorough analysis, providing insights into future research areas and the creation of AI-enhanced development processes.

Keywords :- Agile Practices', Agile Maturity Model (AMM), AI-Enhanced, Green IT, Sustainability, AI-Driven, Automated Test, Training Data, Software Grows, Life Cycle, AI Models.

I. INTRODUCTION

Security is a crucial component of software products and, in some situations, the system's most significant non-functional need. If security flaws are present, using software systems in critical contexts and depending on their functioning might result in a great deal of risks and losses [1]. A growing number of security flaws and vulnerabilities are published annually despite the fact that there are several studies demonstrating novel approaches, strategies, protocols, policies, and instruments to improve software system security. The development team should provide particular attention to security at every step of the software development life cycle since non-functional needs, such as security, cannot be added to a system like functional features [1, 2]. To get more secure software, security operations should be included into iterations of analysis, design, implementation, testing, deployment, production, and product death [2].

Too far, a number of articles have looked at the connection between ICT and the sustainability area. They talk about how ICT affects the environment or how to balance ICT's energy use and savings [3, 4]. In particular, it is still unclear whether ICT's energy consumption is more or lower than its energy savings, such as a result of more effective procedures or scenario simulations. As a result, reducing software-induced energy and resource consumption is essential in light of issues like climate change [3, 4].

As a result, strategies and solutions for creating and using sustainable software are crucial. However, how to include sustainability or environmental considerations into software design and development is not covered in well-known software engineering textbooks. In order to address these obstacles and perhaps produce more sustainable software products, our study suggests a general extension for software development processes that attempts to integrate sustainability concerns into random software development processes [3, 4]. At the moment, the environmental pillar of sustainability—such as energy consumption—is the focus of the example tools, checklists, and recommendations that help professionals use our concept. But if this model works, there may be more diverse rules and supporting instruments in the future, which might help address the social and economic pillars as well [3, 4].

II. APPLICATION OF AGILE PRACTICES IN ABB RESEARCH & DEVELOPMENT

We go over ABB's product development cycle in this part [3, 4]. Additionally, we provide details on the software development procedures that were created and used to technology development initiatives.

2.1 The ABB Product Development Cycle

Figure 1 illustrates the three main stages of ABB's product development cycle. A product idea's potential commercial value and technical viability are assessed during the Feasibility Study Phase [4, 5]. The actions carried out to more thoroughly assess the technical viability and commercial potential of the suggested product are referred to as the Technology Development (TD) Phase. With assistance from the ABB BUs, the ABB CRCs are mostly responsible for carrying out these early development tasks [5, 6]. At ABB, TD projects usually run six to twelve months. A functioning prototype system with the main features of the anticipated final product is the main output from the TD phase.

2.2 ABB Incremental Product Development

An assessment of ABB's software development methods conducted at the beginning of 2001 showed that the Big Design Up Front (BDUF) or classic "waterfall" paradigm was most often used by ABB development teams. The

requirements of ABB's rapid-development TD projects, however, were not always met by the BDUF models [5, 6]. The high documentation burden of BDUF models is made worse by the exploratory nature of the projects and the corresponding volatility of the requirements [6], which frustrates the development teams and raises the costs of adjusting to changes in the technological direction based on preliminary findings from the feasibility assessments.

2.3 Agile Product Development: ADEPT

To improve agility and maturity on TD projects, the Agile Development in Evolutionary Prototyping Technique (ADEPT) was created [5, 6]. Despite considering a number of pre-existing agile processes and lifecycle models, we ultimately chose to build ADEPT by starting with a standard evolutionary lifecycle model and adding a subset of agile principles to it. We have found that rather than trying to switch to a fully agile methodology right away, ABB development teams used to more conventional software development methods are more likely to embrace a more gradual approach to implementing agile practices, as offered by ADEPT.

For the development phase to be maximized, actors must already take into account the future sustainability implications that are anticipated from post-production stages, particularly from the use phase [7]. Actors must thus predict and estimate first, second, and/or third order consequences on sustainability, as well as rebound effects, throughout development. Actors are also encouraged to quantify the sustainability effects of the software development process itself and to set up a continuous improvement procedure that maximizes the effects [7, 8]. The model outlined in the following sections is the result of these results.

2.4 Agile Practices Assessment method and identification of KPAs for improvement

Evaluating the current agile software development processes is the aim of the evaluation approach. Knowledge of agile software development techniques and business case workshops, which emphasize process improvement and provide a roadmap for it, are components of process evaluation [8, 9]. An agile software development methodology and a modified and customizable version of the SW-CMM evaluation questionnaires serve as the foundation for the AMM assessment model. Agile methods, developers, and on-site clients are emphasized [9, 10]. It is anticipated that this procedure will increase understanding and communication, especially by elucidating the true concerns of those engaged in process improvement initiatives. AMM suggested that everyone have a common goal for process improvement and that anybody may take charge of it at any point. Figure 1 below demonstrates how to identify areas that need process improvement [9, 10]. Using the questionnaires that are covered in the section on the agile maturity model, we pinpoint areas that need improvement.

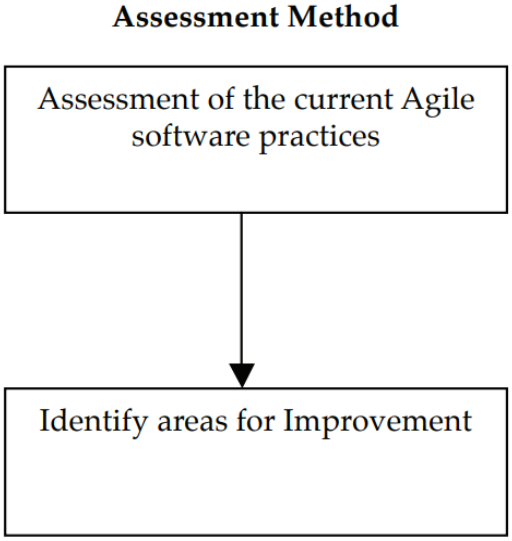

Fig. 1 Areas for improvement assessment framework [9, 10].

III. LIFECYCLES OF SOFTWARE PRODUCTS

Software lifecycle models, sometimes referred to as product lifecycle models in the commercial context, are not appropriate for identifying product lifecycle stages that impact sustainability. These lifecycle models concentrate on business elements (e.g., sales volumes, profitability) and software development or maintenance aspects (e.g., development plans, release plans, service releases) [9]. Therefore, based on a product lifecycle described in ISO/TR 14062, we suggest a software lifecycle model that is influenced by Life Cycle Thinking (abbr. LCT), which is also emphasized in phrases like "from cradle to grave." Raw material extraction, manufacture, distribution, usage, recycling, and product disposal are all covered by [LCT] [9, 10]. This lifespan model has to be modified to include immaterial software items, even though it may be used for material things like computers, vehicles, and dishwashing machines. Figure 2 [9, 10]. As a result, lifecycle stages like raw material extraction and recycling are eliminated since they do not apply to immaterial items. There are two goals for the model: At each step of the software product's lifetime, assign criteria (material, energy, data transmission) that result from the product's effects on sustainability. As early as feasible, establish the foundation for initiatives that maximize these effects, whether favourable or bad [10]. The goal of these initiatives is to provide software that is more sustainable. The model's outline, which is explained below, primarily concentrates on sustainability's environmental components. Future plans call for extensions to the other sustainability pillars. This study focuses on the Development phase, which is the initial lifecycle phase. Here, the software development process makes use of a number of structured techniques and tools. These allow stakeholders to evaluate how the software product affects sustainability over its entire lifespan [11].

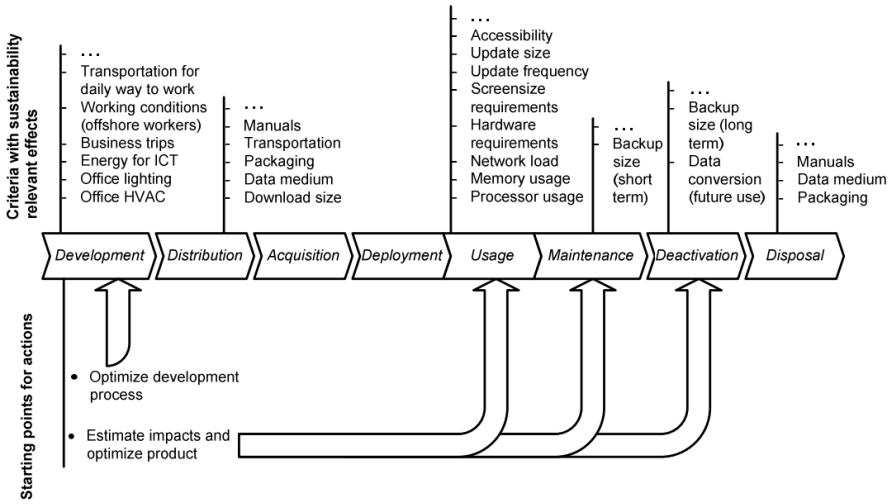

Fig. 2 Life Cycle Thinking inspired lifecycle for Software Products.

Additionally, it enables them to take steps to enhance the software product to maximize its effects [11, 12]. This should result in a software product design that is more sustainable. Both standard and bespoke software are affected by the distribution phase. It considers the effects on sustainability that arise from the packing, the download of software packages, or the development of the data medium [12]. The acquisition phase is the next stage of the lifecycle. Acquisition in this context refers to evaluating many standard software packages, selecting the one that best suits one's requirements, and buying it from a software shop. When choosing software, those in charge should take sustainability factors into account in addition to technical, functional, or licensing requirements [13]. When deploying software products, the deployment lifecycle phase takes into account factors that are pertinent to administrators.

3.1 A Generic Model for Sustainable Software Engineering

Software Engineering General Models A number of systematic process models that are known in software engineering enable the systematic and orderly creation of software products. Requirements analysis, design, implementation, testing, operation, and maintenance are typical stages of a process. The majority of these process models seem to be somewhat dissimilar at first sight [19]. Even when the phases are iterated, have different names, or occur in different order, it is still possible to identify activities that relate to the aforementioned phases when examining the various activities within these models [15]. Most models may be represented using a generic model by removing the waterfall model's restrictions on the sequential sequence of iterations. This broad paradigm permits feedback between stages, performs phases in parallel, and iterates phases as needed [14]. Therefore, without restricting our approach to it, we are utilizing this broad model to demonstrate how our improvements are integrated into development processes.

3.2 Model Overview

Figure 3 describes the general process improvement model for software development that is sustainable. As shown in the activity box [15], we assume a basic phase order without iterations and concurrent phase execution for this overview. The model itself is part of the software product lifecycle model Figure 1's development lifecycle phase. Unlike our software product lifecycle model, this model examines a software product's development phase from an organizational viewpoint [19]. As a result, it serves as a model for procedures and actions that companies should implement.

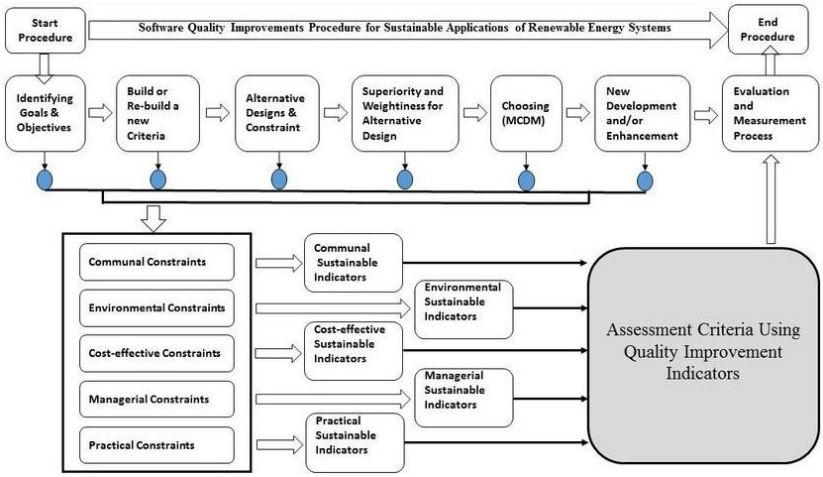

Fig. 3 Process Enhancement Model for Sustainable Software Engineering. [13, 15]

3.3 Sustainability Reviews & Previews Sustainability

In order to maximize the sustainability of the software product under development, Reviews & Previews examine the work completed, evaluate results in light of sustainability concerns, and create measures that are implemented until the next Review & Preview. These studies consider software features that affect sustainability, such as requirements, architecture, or code [3]. Actors examine and evaluate their artifacts (such as requirements, architecture, and code) in a team facilitation style [11] in order to create more sustainable solutions if needed. That's the part about reviewing. The remedies themselves and the prediction that they will be more sustainable are included in the Preview section [16].

3.4 Process Assessment

The development process is regularly quantified via the Process Assessment activity. The concept of lifespan assessment states that it contributes to the lifecycle inventory in certain ways. In order to evaluate the process'

effects on sustainability, various data from the development process is gathered [15]. This activity's goal is not to evaluate the whole software product; it is the goal of the Sustainability Retrospective and Sustainability Reviews & Previews [13]. However, there is presently no advice on how software products and their manufacturing processes may be affected by the LCA technique outlined in the ISO 14040 series.

3.5 Sustainability Journal

The process improvement's information centre is the Sustainability Journal. It is an organized document that changes in tandem with the software project. In addition to documenting Sustainability Reviews & Previews, [18], Process Assessment, and the Sustainability Retrospective, it also publishes the evaluated sustainability effects at the end of the project. In order for stakeholders to readily rebuild the choices made subsequently, sustainability assessments and previews should be as brief as possible while yet include sufficient facts [20, 21]. The journal publishes two reports on the data gathered via Process Assessment activities. If choices are made as a result of the Process Assessment, it first reports to Reviews & Previews and is documented.

3.6 Supporting Tools, Checklists, Guidelines and Educational Material

Tools (such as specialized software and spreadsheets) that facilitate data gathering during Process Assessment and help actors with LCA concerns should be used to support the process modifications. It is essential to take action early in the development phase to lessen the effects of a software product's use phase. This implies that in order to improve, for example, software design or code, actors, such as [22], software architects, and developers, need tools, rules, or clues that provide them ideas and recommendations if effects on sustainability are found. Therefore, we suggest a knowledge base that helps actors locate relevant information [22, 23]. Examples of best practices are provided. The need for educational materials stems from the fact that computer scientists and stakeholders/actors in software development projects typically lack a thorough understanding of sustainability issues, such as [23, 24], first order effects, second and third order effects, or more complex concepts like rebound effects [26, 27]. As a result, it is essential to provide teaching resources that clarify fundamental sustainability ideas, empower stakeholders to evaluate the effects of software and development procedures, and encourage them to take action to reduce adverse effects or enhance favourable ones [29].

IV. INSTANTIATING THE MODEL: BEYOND THE WATERFALL APPROACH

This generic modification aims to be applicable to many software process approaches [28, 29]. Two approaches that adhere to the agile paradigm demonstrate how that may be achieved [14]. Open UP (Open Unified Process) is the first approach, while Scrum is the second.

4.1Tailoring to Open UP

A lean unified process is Open UP. It is said to be a lightweight, agile procedure. It takes into account best practices for software development, such as frequent releases of functional software, teamwork, continuous integration and testing, and iterative development. There are two aspects to the lifecycle's structure [13]. The four lifespan phases—the "Inception Phase," "Elaboration Phase," "Construction Phase," and "Transition Phase"—are represented by the first dimension. Every stage of the lifecycle may undergo several iterations. Activities related to the lifecycle stages are represented by the second dimension [11, 12]. "Identify and Refine Requirements," "Outline the Architecture," "Develop the Architecture," "Develop Solution Increment," and "Test Solution" are the primary tasks associated with software development [19]. Activities are carried out in parallel throughout a

lifecycle phase. The emphasis of the lifecycle phase and the requirements of the operating process determine which actions are carried out and how much labour each activity generates?

4.2Tailoring to Scrum

Every iteration of the agile, iterative Scrum project management methodology aims to produce a potentially shippable product. We refer to these iterations as "sprints" [18]. Sprint Planning Meetings precede each sprint, which concludes with the Sprint Review and Sprint Retrospective. The product owner and the development team decide which features should be included in the next sprint during sprint planning meetings [18, 28]. The team presents the finished features as a potentially shippable product increment during Sprint Review sessions. The product owner then decides whether to accept or reject the implemented features. The goal of the Sprint Retrospective, a team learning technique, is to enhance collaboration and the development process. As a result, the group talks about and considers the previous sprint and decides on modifications for the next one.

V. CONCLUSION

A difficult problem in the realm of software engineering is whether or not the capability maturity models for software and process improvement apply to agile approaches. We explain in this article why and how we have modified the agile maturity model and process improvement framework to concentrate on agile software development methods.

According to the research, Scrum's adaptability, iterative structure, and emphasis on teamwork and user input are essential for improving product development procedures. Organizations must have a continual improvement and adaptation culture in order to successfully use Scrum. "Continuous Scrum," which combines Scrum with DevOps methods, is an example of how the approach may be innovative in software development lifecycles. Additionally, Scrum's capacity to adapt to contemporary work situations is shown by its successful implementation in remote and dispersed teams—as long as team members place a high value on communication and trust.

Although the advantages are obvious, achieving the full potential of AI in software development requires tackling the issues of data quality, explain ability, integration, scalability, and ethical concerns. Future studies should concentrate on resolving these issues, encouraging productive human-AI cooperation, and making sure AI tools are transparent, reliable, and easily incorporated into current processes. By resolving these problems, software engineering may advance and use AI to create software systems that are more inventive, dependable, and efficient.

In order to promote sustainable software product design, we introduced a general model in this work that improves standard software development procedures. In order to accomplish "Sustainable Software Engineering," it presents a number of activities and artifacts, such as sustainability reviews and previews, continuous process assessments, a sustainability retrospective, and a sustainability journal. Guidelines, instructional materials, and supporting tools are included with the model. The model itself does not depend on any particular software development process, even though we first utilized a straightforward waterfall-like approach to demonstrate how the activities and artifacts of the model operate together. Our suggestion for adapting the approach to agile software development techniques highlights this. It is possible, nevertheless, that our paradigm will work better for fast procedures than for intricate, high-ceremony ones. Our next stages include creating and assessing the required sustainability standards for software, testing the model first in student projects and then in real-world projects, and creating the related resources, manuals, and instructional materials. We want to expand our model

in the future. Aspects of broadening include taking into account different stages of the software lifecycle and other sustainability elements, such as the social and economic pillars.

VI. REFERENCES

[1] C. Wallin, F. Ekdahl, and S. B. M. Larsson, Integrating business and software development models, IEEE Software, November-December 2002, 28-33.

[2] K. Auer and R. Miller, XP applied (Reading, Massachusetts: Addison Wesley Professional, 2001).

[3] A. Cockburn, Agile software development (Reading, Massachusetts: Addison Wesley Longman, 2001).

[4] M. B. Chrissis, M. Konrad, and S. Shrum, CMMI®: guidelines for process integration and product improvement (SEI Series in Software Engineering, Pearson Education Inc., 2003).

[5] A. Dagnino and K. Smiley, Agile development in evolutionary prototyping technique (Internal ABB document, 2002).

[6] J. Highsmith, Agile software development ecosystems (Boston, MA: Addison Wesley, 2002).

[7] Hilty, L. M. (2005): Information systems for sustainable development, Idea Group Publishing,

[8] Hershey, Pa. Käfer, G. (2009): Green SE: Ideas for Including Energy Efficiency into your Software Projects. Technical Briefing (TB2). 31st International Conference on Software Engineering, Vancouver. Internet.

[9] Kruchten, P. (2003): The rational unified process. An introduction. 2. ed., Addison-Wesley, Boston. Mocigemba, D. (2006): Sustainable Computing. In: Poiesis & Praxis: International Journal of Technology Assessment and Ethics of Science, volume 4, number 3, pages 163–184. Royce, W. W. (1970): Managing the development of large software systems: concepts and techniques. In: Proc. IEEE WESTCON, Los Angeles, pages 1-9.

[10] Boehm, B., Port, D., Jain, A., and Basili, V. (2002), Achieving CMMI Level 5 Improvements with MBASE and the CeBASE Method. [Internet] Cross Talk Journal.

[11] Brodman, J. and Johnson, D (1997), A Software Process Improvement Approach for small organisation and small projects. Proceedings of the 19th International Conference in Software Engineering, 19th may 1997, Boston- MA, ACM Press, pp661-662.

[12] Casey, V. and Richardson, I. (2002), A Practical Application of Ideal Model. Product Focused Software Process Improvement, 4th International Conference (PROFES), December 9-11, Rovaniemi – Finland, Springer, pp.172-184.

[13] Chrissis, M. B., Konrad, M. and Shrun, S. (2003) CMMI: Guidelines for Process Integration and Product Improvement, UK, Addison Wesley.

[14] Glib, T. (2003), Software Project Management Adding Stakeholder Metrics to Agile Projects. The European Journal for the Informatics Professional, IV (4) August 2003, pp.5-9.

[15] Herbsleb, J. D. and Goldenson, D. R. (1996), A Systematic Survey of CMM experience and results. Proceedings of the 18th international conference on Software Engineering, May 1996, Berlin, Germany, IEEE Computer Society, pp.323-330.

[16] Highsmith, J., (2004) Agile Project Management, Creating innovative products, Addison- Wesley.

[17] Ihme, T. and Abrahamsson, P., (2005) The Use of Architectural Patterns in the Agile Software Development of Mobile Applications, International Journal of Agile Manufacturing, Vol. 8, issue 2, 97-112.

[18] Johnson, J., Boucher, K. D., Connors, K. and Robinson, J. (2001), Collaborating on Project Success. [Internet], Software Magazine, Available.

[19] Anand, S., Burke, E. K., Chen, T. Y., Clark, J., Cohen, M. B., Grieskamp, W., ... & McMinn, P. (2013). An orchestrated survey of methodologies for automated software test case generation. Journal of Systems and Software, 86(8), 1978- 2001.

[20] Saeid, H. (2018). Revolutionizing Software Engineering: Leveraging AI for Enhanced Development Lifecycle. International Journal of Innovative Research in Engineering & Multidisciplinary Physical Sciences, 8(1).

[21] Weyns, D., Iftikhar, M. U., De La Iglesia, D. G., & Ahmad, T. (2012). A survey of formal methods in self-adaptive systems. [6] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[22] Junejo, A. A., & Memon, S. (2016). Case study on evolution performance of agile scrum software development life cycle for shopping cart applications. World Journal of Advanced Engineering Technology and Sciences, 10(1), 085-090.

[23] Kamal, F. (2017). Literature survey on kanban: opportunities and challenges. International Journal of Scientific and Research Publications, 10(11), 935-945.

[24] Kaur, K., & Khurana, M. and Manisha. (2016). Impact of agile scrum methodology on time to market and code quality – a case study," 2016 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N), Greater Noida, India, 2016, pp. 1673-1678.

[25] Kautz, K., Johansen, T., & Uldahl, A. (2014). The perceived impact of the agile development and project management method scrum on information systems and software development productivity. Australasian Journal of Information Systems, 18(3),

[26] Khalid, K.A., Abdullah, & Qureshi, M. (2016). Agile software development: Impacts and challenges within distributed teams. International Journal of Multidisciplinary Research and Growth Evaluation, 4(1), 572-579.

[27] Nyandongo, K. M., & Madumo, M. R. (2018). Assessing the effectiveness of the scrum framework and its successful implementation," IEEE 28th International Conference on Engineering, Technology and Innovation (ICE/ITMC) & 31st International Association for Management of Technology (IAMOT) Joint Conference, Nancy, France, 2022, pp. 1-8.

[28] Opt, S., & Sims, C. D. L. (2015). Scrum: Enhancing student team organization and collaboration. Communication Teacher, 29(1), 55-62.

[29] Permana, P. A. (2015). Permana, P. A. G. (2015). Scrum method implementation in a software development project management. International Journal of Advanced Computer Science and Applications, 6(9), 198-204.

[30] Pócsová, J., Bednárová, D., Bogdanovská, G., & Mojžišová, A. (2018). Implementation of agile methodologies in an engineering course. Education Sciences, 10(11), 333.