Lean Architecture for Agile Software Development Business Processing Cycle – "Chunks Processes"

Dr.G. Singaravel[1], S. Sethupathi[2], T. Satheshkumar[3]

[1]Professor, Department of Information Technology, K.S.R. College of Engineering (Autonomous), Tiruchengode, Namakkal District, Tamil Nadu, India. E-mail: singaravel@ksrce.ac.in
[2]Assistant Professor, Department of Information Technology, K.S.R. College of Engineering (Autonomous), Tiruchengode, Namakkal District, Tamil Nadu, India. E-mail: sethupathis@ksrce.ac.in
[3]Assistant Professor, Department of Information Technology, K.S.R. College of Engineering (Autonomous), Tiruchengode, Namakkal District, Tamil Nadu, India. E-mail: sathishkumart@ksrce.ac.in

***Abstract*---** Scrum is one of the most widely adopted frameworks for agile team collaborative implementation ranging from software development to other fields in between. To provides a way of managing work as teams break it down into development tasks that have to be completed within time-boxed intervals known as sprints. Most of the time, the intervals-Carthasigo would-be two weeks and the maximum would be one month. Every day, the actual scrum holds a 15-minute time-boxed stand-up meeting. The scrum team measures progress toward its sprint goal. At the end of the sprint, the team holds two additional meetings: an internal sprint retrospective and a sprint review, where the work done will be presented to stakeholders to obtain their feedback. Typically, a scrum master is the person in charge of a scrum team. Scrum encourages teams to self-organize by promoting tight online cooperation or physical co-location and requiring regular communication among all team members. The continual feedback and adaptability and loose structure rest quite a bit on a notion of requirements instability-meaning that the stakeholders are going to change their minds as the project develops. One of the effects of what is called the design process is architecture, which brings about a challenge in design. Essentially, the proposal of agile patterns of problem-solving consists of parsing the business processing cycle so effectively into "chunks" that can be processed concurrently. A lean approach to get the perfect value to the consumer through a perfect value-creating process that has zero waste.

***Keyword*---** Agile Methodology, Scrum, Lateral Linking Problem, SDLC, Chunks Processes.

I. INTRODUCTION

THE idea behind lean software development is to maximize productivity and reduce waste in the software development lifecycle. The method was inspired by the 1980s about the Lean Manufacturing movement. But these days, it's seen as an essential component of the Agile software development process [1][14]. The fundamental idea of the lean development technique is that waste can be controlled at every stage of the process and efficiencies may be implemented. These occur at the individual, departmental, and interdepartmental levels as well as within the corporation as a whole and between the company and its suppliers and customers.

The Development of Lean Agile

Lean Architecture sheds new light on crucial software development topics that the agile movement has ignored or overlooked. The primary tenet of lean agile methodology is to increase efficiency through waste elimination. In contrast to waterfall project management, which follows a predetermined plan created by a project manager, lean agile aims to eliminate any tasks and activities that don't provide any true value [17]. The makes it possible for all parties involved in a project or product development to work as efficiently as possible.

The Five Principles of "Lean agile"

To applying lean methods, there are five fundamental principles to follow: such as Value, Stream of values, the movement, pulls and completeness.

These guidelines outline a five-step procedure for implementing lean methods in software development teams, manufacturing, and other agile industries. Lean places a high priority on getting rid of waste to increase productivity [11]. This enables teams to prioritize the tasks that provide the greatest value to consumers while continuously improving their procedures. Extreme programming (XP), rapid application development, and other agile frameworks are just a few of the eight software development methodologies that we recently reviewed if you're interested in learning more about how agile concepts fit with other development techniques [2],[16].

II. LITERATURE STUDY

Software development teams use agile approaches to collaborate and adjust to needs that change over time. But over time, this adaptability could result in technical debt (TD), which could introduce errors. Lean management techniques, which prioritize waste reduction and ongoing process enhancement, can be utilized to effectively handle test data in agile software development. This study undertakes a methodical evaluation of the literature on the application of lean and agile methods to the management of TD. 34 publications are identified in the review, which classifies different types of technical debt, identifies lean and agile concepts, and matches technical debt categories with appropriate lean and agile principles. In addition, the CoDVA framework, the LTD framework, and the TAP framework are recognized as the three current technical debt management frameworks. The study comes to the conclusion that companies can benefit from the proper integration of lean principles into agile software development, the researcher identified by the **Surya Seven Y. Simangunsong et al (2023).**

Sadeghi S et al (2022), to Identification, measurement, and prioritization of technical debt are among the tasks involved in technical debt management, or TDM. It is primarily carried out to prevent the software product from becoming less maintainable and evolvable, which would lower team velocity. TDM is still not widely used in software firms, despite its significance. The fierce competition and strong market demand facing software firms means that providing value to customers now trumps the time and effort required for TDM operations. Because the effects of technical debt are unpredictable and become apparent over time, it is more challenging for businesses with very few resources to allocate resources to TDM in a proactive manner. In this work, we provide a lean methodology to help software companies with very little resources embrace TDM.

Abdullah Aldaeej et al (2023), many software organizations now want to embrace large-scale agile methodologies in an attempt to mimic the success of the highly prevalent and successful adoption of these approaches at the team level within the organization. Nevertheless, a review of the literature that is now available finds that (i) there aren't many conceptually sound studies of large-scale agile transformations; (ii) the assumptions behind the studies that do exist are shaky; and (iii) there is an excessively problematic and unsophisticated dependence on simple adherence to agile methodologies. Furthermore, nothing is known about the reasons behind the failure of extensive agile changes. In order to investigate this further, we use normalization process theory (NPT) to look at important issues surrounding the normalization—that is, the process of integrating and maintaining large-scale agile methodologies. To offer a thorough case study of a significant agile transformation and show how failing

III. PROBLEM IDENTIFICATION

The good problem definition helps the team: Self-organise (agile), by providing a directive by which the team will guide its decisions; Focus on the problem at hand and eliminate everything else that does not add value to the problem resolution (lean) [9]. While agile techniques vary in practices and emphasis, they follow the same principles behind the agile manifesto of working software is delivered frequently (weeks rather than months). Working software is the principal measure of progress. Customer satisfaction by rapid, continuous delivery of useful software [3].

IV. PROPOSAL METHODOLOGY

According to Wikipedia, critical thinking is the process of evaluating information through factual analysis. It entails recognizing an issue and using logic and facts to analyse it in order to assess various solutions and choose which is most likely to provide the desired outcome. However, brainstorming is a necessary component of creative thinking [10]. Proposals for problem solutions are made and then examined. Divergent thinking is used in this strategy where you start with a prompt or question and come up with several answers.

An Agile Architect is responsible for defining and maintaining the structure of the solution by making sure that the product and the Product Increments meet the requirements. Agile's low initial planning requirements make it simple to get distracted when providing unexpected or new features [12],[13]. Furthermore, it implies that projects have no end in sight because there is never a precise idea of what the "final product" would look like.

Identifies of Framework in Agile Architecture: Scrum. Among the most popular Agile team frameworks is Scrum. "Framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value" is how the Scrum Guide describes scrum [4],[5]. Determine the problem's underlying cause by gathering data and then consulting with relevant parties. Integrating data from your stakeholders with previously conducted research might provide some understanding of the issue and its causes [6].

The four indicators that make up the IDEAL model are as follows:

- Identify the problem: this involves attempting to pinpoint issues and turn them into opportunities for creative action.
- Define goals: in a problem, specify goals.
- Investigate solutions: investigate different methods for resolving issues.
- Implement the plan: foresee outcomes.

Agile architecture is a set of principles, procedures, and teamwork that promotes the dynamic, evolutionary architecture and design of a system [8]. By embracing the DevOps attitude, this method enables the architecture to continuously grow while meeting the needs of present users.

V. CONCLUSION

The outcome of the process known as design is architecture, and design cannot exist in the absence of a challenge. An explicit written declaration of an issue is called a problem definition, and it is the difference between the intended and present states [7],[15]. To determine the destination before planning the route to get sources. The software developers what issue their product addresses, the majority of the time. Agile architecture is a set of principles, procedures, and teamwork that promotes the dynamic, evolutionary architecture and design of a system. By embracing the DevOps philosophy, this method enables the architecture to continuously adapt while meeting the needs of present users.

REFERENCE

[1] Aldaeej, A., Nguyen-Duc, A., & Gupta, V. (2023, May). A Lean Approach of Managing Technical Debt in Agile Software Projects–A Proposal and Empirical Evaluation. In *International Conference on Agile Software Development* (pp. 67-76). Cham: Springer Nature Switzerland.

[2] Furlan, A., Grandinetti, R., & De Toni, A. F. (2023). Managing the Lean–Agile Paradox in Complex Environments. *Systems*, *11*(5), 258. https://doi.org/10.3390/systems11050258

[3] Bajwa, S. S., et al. (2023). Investigating the Challenges of Scaling Agile in Large Organizations. *International Journal of Project Management, 41*(3), 218–229.

[4] Carroll, N., Wang, X., & Conboy, K. (2023). Agile as the New Normal: Overcoming Barriers in Hybrid Project Environments. *Journal of Software Evolution and Process.*

[5] Stettina, C. J., Garbajosa, J., & Kruchten, P. (2023). *Agile Processes in Software Engineering and Extreme Programming: 24th International Conference on Agile Software Development, XP 2023, Amsterdam, The Netherlands, June 13–16, 2023, Proceedings* (p. 193). Springer Nature. https://doi.org/10.1007/978-3-031-33976-9.

[6] Coplien, J. O., & Bjørnvig, G. (2011). *Lean architecture: for agile software development*. John Wiley & Sons.

[7] Poppendieck, M. (2007, May). Lean software development. In *29th International Conference on Software Engineering (ICSE'07 Companion)* (pp. 165-166). IEEE.

[8] Malvar, E., & Chen, N. (2023, July). Creating Continuous Improvement in Agile Software Development Using Lean Six Sigma. In *2023 Congress in Computer Science, Computer Engineering, & Applied Computing (CSCE)* (pp. 2571-2578). IEEE. https://doi.org/10.1109/CSCE60160.2023.00412

[9] Carroll, N., Conboy, K., & Wang, X. (2023). From transformation to normalisation: An exploratory study of a large-scale agile transformation. *Journal of Information Technology*, *38*(3), 267-303. https://doi.org/10.1177/02683962231164428

[10] Kruchten, P. (2010, May). Software architecture and agile software development: a clash of two cultures?. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 2* (pp. 497-498). https://doi.org/10.1145/1810295.1810448

[11] Sadeghi, S., Akbarpour, A., & Abbasianjahromi, H. (2022). Provide a lean and agile strategy for an antifragile sustainable supply chain in the construction industry (residential complex). *Cleaner Logistics and Supply Chain*, *5*, 100079. https://doi.org/10.1016/j.clscn.2022.100079

[12] Pasuksmit, J., Jiang, F., Thornton, K., Friedman, A., Fuksmane, N., Kohout, I., & Connor, J. (2023, May). Improving Agile Planning for Reliable Software Delivery. In *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)* (pp. 25-26). IEEE. https://doi.org/10.1109/MSR59073.2023.00017

[13] Simangunsong, S. S. Y., Raharjo, T., & Fitriani, A. N. (2023). Lean and Agile Software Development for Managing Technical Debt on A Large-scale Software: A Systematic Literature Review. *The Indonesian Journal of Computer Science*, *12*(6). https://doi.org/10.33022/ijcs.v12i6.3612

[14] Edison, H., Wang, X., & Conboy, K. (2021). Comparing methods for large-scale agile software development: A systematic literature review. *IEEE Transactions on Software Engineering*, *48*(8), 2709-2731. https://doi.org/10.1109/TSE.2021.3069039

[15] Turner, N., Swart, J., & Maylor, H. (2013). Mechanisms for managing ambidexterity: A review and research agenda. *International journal of management reviews*, *15*(3), 317-332. https://doi.org/10.1111/j.1468-2370.2012.00343.x

[16] Walter et al., (2015). Continuous Improvement in Lean-Agile Processes for Large Telecommunications. *International Journal of Software Engineering Practices*, *3*(2).

[17] Zorzetti, M., Signoretti, I., Salerno, L., Marczak, S., & Bastos, R. (2022). Improving agile software development using user-centered design and lean startup. *Information and Software Technology*, *141*, 106718. https://doi.org/10.1016/j.infsof.2021.106718