Unlocking the Power of Agile Methodology: A Journey to Developing Software and Web Applications

Venkata Koteswara Rao Ballamudi

Sr. Software Engineer, HTC Global Services, USA

Corresponding Contact:
Email: venkata.bvk@gmail.com

ABSTRACT

Agile development approaches are becoming increasingly significant in the current circumstances. Throughout the past few years, there has been a rise in the number of software application development projects that use agile approaches. It is a new paradigm of research that crosses disciplinary boundaries. This article delves into the iterative and collaborative approach of Agile, highlighting its ability to enhance project flexibility, customer satisfaction, and product quality. It showcases real-world examples of Agile implementation and shares best practices for successful adoption. The study covers Agile's introduction, adoption, usage, and analysis in both engineering streams. The remaining portion of the article provides an overview of the critical distinctions between applying agile methodologies to software engineering and web engineering. It explains the constraints inherent in using agile software and web engineering methodologies. Readers will gain insights into harnessing the agility of this methodology to adapt to changing requirements, reduce development risks, and deliver cutting-edge software solutions efficiently. Embrace Agile and embark on a journey toward software development excellence.

Key words:
Agile Methodologies, Traditional Methodologies, Web Engineering, Software Engineering, XP, Scrum, ICONIX, Crystal

9/15/2022 **Source of Support:** None, **No Conflict of Interest:** Declared

INTRODUCTION

Agile development approaches are currently being widely embraced by businesses of all kinds that specialize in software production. According to polls conducted inside the industry, almost every company uses agile methodologies to some degree, and more than half of these companies utilize agile as their primary strategy for developing software.1 Practitioners have reported various positive effects, including a shorter time-to-market, higher customer satisfaction, and lower overall development expenses.2 Managing Agile

projects, in particular in the context of small and medium-sized businesses (SMEs), can be difficult. The product owner and the scrum master face at least two challenges simultaneously: the first is to ensure the quality of the software product, and the second is to assist the efficiency of the team and the process (Ghazi & Glinz, 2017).

Currently, many teams working on software development projects support the development process and the quality of the code and products by utilizing various specialized tools. Some examples of these tools include Jira, GitLab, and SonarQube. This is typically done during a scheduled meeting that looks back on previous events and has everyone on the team. The Scrum Team receives sufficient information from those tools regarding the code quality being developed. Despite this, there is still a gap and a need for additional solutions that represent the effectiveness of teams and the quality of processes. Currently, initiatives for improving processes are primarily based on developers' impressions, and very little support is offered to make process-wise and data-driven judgments (Kaluvakuri, 2022).

The most important contribution that this paper makes is in the form of a group of metrics that measure the Agile software development process (which we will refer to as process metrics from this point forward) in a company that is classified as a small or medium-sized enterprise (SME), as well as a discussion on how those metrics assisted the Scrum Team in the process of developing a commercial product (Maddali et al., 2021). The metrics were created as part of an action research cooperation between researchers and a Polish small-size software development business known as ITTI Sp. z o.o. This collaboration took place while developing the commercial product known as CONTRA by ITTI.

When more than twenty Agile approaches specialists worldwide were brought together for the first-ever eWorkshop on Agile methodologies, they began by defining Agile methodologies. This was the first step in the process of developing agile methodologies. In particular, the working definition of Agile methodologies is given as follows: iterative, which means that it delivers a complete system at the very beginning and then changes the functionality of each subsystem with each new release; incremental, which means that the system as specified in the requirements is partitioned into small subsystems by functionality and new functionality is added with each new release; and self-organizing, which means that the team has the autonomy to organize itself to complete the work it has been given (Song & Zeng, 2014).

The Agile Manifesto outlines four core ideals and twelve guiding principles, which are adhered to by all Agile approaches. They value people and the interactions they have with one another more than they love processes and tools; they prioritize the production of functioning code over the documentation of that code; they involve the client in the development process rather than negotiating contracts; and they are more open to change than they are to stick to a predetermined plan (Kaluvakuri & Lal, 2017). They are constructed using several research approaches. The review structure for each methodology begins with a brief overview, followed by a short description of the methodology's process (life cycle), followed by a listing of the leading project management features. We have yet to include the scaling values for the team size because we know that as the Agile methodologies are scaled up to larger teams, they begin to include more aspects of the rigorous group and, as a result, become less agile (Maddali et al., 2019). For this reason, we have not included the scaling values. The final characteristic is the range of applications for the methodology.

EXTREME PROGRAMMING (XP) LIFE CYCLE

Extreme Programming is a lightweight methodology for developing software based on hazy or rapidly changing requirements. It is intended for use by small to medium-sized teams. When developing software utilizing XP, the first step is for the customer to write down "stories" that define the features and functions of the software. These "stories" are bite-sized chunks of functionality, each requiring approximately one to two weeks to be programmed and tested. The programmers provide the estimates for the stories, and the customer determines, based on value and cost, which tales should be completed first (Roy et al., 2021a). The development process is iterative and performed in small steps. Every two weeks, the development team gives the client working stories they've developed. Next, the client selects another two weeks' worth of work to be completed. The system's capability expands gradually, piece by piece, under the direction of the customer. Rather than judging design artifacts, progress is measured and recorded based on the system's observable behavior. This is done in place of the traditional method. XP relies on evolutionary design strategies to maintain a high-quality software architecture while expanding its capabilities. XP is a compilation of twelve of the industry's most successful practices (Roy et al., 2020).

According to Brewer, the twelve essential practices of XP are as follows: The Planning Game involves collaboration between business and development to generate the most significant amount of business value in the shortest amount of time (Roy et al., 2021c). The planning game can be played on various scales, but the fundamental rules remain the same throughout. The user compiles a wish list of features they would like the system to have. Each feature is outlined as an IIser sto/Y, which provides a name for the feature and a high-level description of what is required. User stories are often written on cards that measure 4" x 6". The developers estimate how much work will be needed to complete each narrative and how much work the team can generate in a set amount of time (the iteration). After that, users select which user stories to build and in what order, as well as when and how frequently production releases of the system should be made (Farid, 2015).

- Start with the smallest and simplest helpful feature set and work up to more significant releases. Launch the product as soon as possible and update it frequently, adding a few new features each time.
- An Organizing Metaphor: Each project has an organizing metaphor, which provides an easy-to-remember naming convention by employing a straightforward and universally understood account of the system's functions.
- Always choose the most straightforward Design: This rule states that we should always select the most detailed design that does the job. Because the needs will differ in the morning, we should perform the actions required to satisfy today's criteria.
- Continuous Testing: Before adding a new feature, the programmers develop a test for it to ensure that it works properly. The task will be finished when the suite is executed. Unit and acceptance tests are the two varieties of tests used in XP.
- The process of refactoring involves removing any duplicate code that was generated during a development session.
- Using a technique known as "pair programming," all of the production code is written by two programmers simultaneously working on a single machine. This allows for reviewing while the code is being written.
- Ownership of Collective Code: No person "owns" a module alone. It is expected that any given developer will be able to work on any given portion of the system code at any given moment.

- Continuous Integration: Every modification is incorporated into the system's software daily, at the very least. Before and after the integration, the tests have to be fully functional one hundred percent of the time.
- The forty-hour workweek allows programmers to leave work at the appropriate moment. On-site Customer: The Development team has continuous access to a real live customer who will be using the system. On-site customers are located at the exact location of the Development team. When dealing with commercial software with many clients, a customer proxy—typically the product manager—is employed instead.
- Coding Standards: All programmers adhere to the same set of coding standards. If everything goes according to plan, one should not be able to tell by looking at the code that it has been modified in any way (Del et al., 2014).

It is iterative, based on the software industry's best practices, and emphasizes design. Additionally, it places a high priority on communication, simplicity, and feedback. It is much more prescriptive than other Agile methodologies; pair programming is quite controversial due to an added overhead on resources, though it can be argued that it leads to fewer defects and reduced cycle time. XP docs do not address the issue of deployment. These limitations include no design documentation, which can be a problem at the maintenance stage; it differs from other Agile methodologies in that it is much more prescriptive and needs to address the deployment issue. The management process, risk, and measurement procedures are all areas in which XP performs poorly.

ICONIX Development Process Life Cycle

ICONIX's analysis and design power is built on the Unified Modeling Language (UML), a medium-sized software development technique. It is claimed to be located someplace in the middle of RUP and XP. ICONIX provides a streamlined software development method that includes a minimal set of diagrams and processes that a project team may use to get from use cases to code fast and efficiently. These diagrams and techniques can be found in the ICONIX Software Development Kit (Insfran et al., 2014).

The ICONIX method begins with domain modeling, which entails defining the things in the actual world that will serve as the vocabulary for the use cases. This task is performed before any other step in the ICONIX process. Because of this, a significant portion of the team is immediately and meaningfully involved in the project. Use cases in ICONIX comprise short, simple pieces of text that capture the functional needs in a manner that is simple enough for anyone to comprehend.

ICONIX has three key features. First, the method simplifies UML. Second, the method is highly traceable. At every stage, we reference the requirements. No procedure permits us to wander too far from user demands. This allows items to be tracked using design artifacts. Third, it's gradual and iterative. Developing the domain model and identifying and analyzing USC examples need numerous iterations (Kaluvakuri & Amin, 2018).

The dynamic model (USC cases, robustness analysis, and sequence diagrams) refines the static model incrementally across iterations. The approach does not require formal milestones or extensive bookkeeping. Instead, refining efforts leads to natural milestones as the team acquires knowledge and expertise. The project monitoring and control system develops independently as work progresses.

SCRUM METHODOLOGY LIFE CYCLE

In object-oriented communities, the Scrum framework has been used for some time. Along with XP, Scrum is one of the Agile techniques with the most widespread adoption. The core tenet of Scrum is that defined and repeatable processes are effective only when applied to the solution of restricted and repetitive problems by defined and repeatable people in defined and repeatable settings, which is impossible. A project is divided into iterations of thirty days each, called sprints, to address specified and repeatable processes. Before the sprint begins, the functionality required for that sprint is defined, and the team is tasked with delivering it. The goal of this phase of the sprint is to achieve requirements stabilization. The principles of project management are emphasized in Scrum. The word "Scrum" comes from the sport of rugby, which describes what happens during a "Scrum" as "players from each team clumping closely together... in an attempt to advance down the playing field."

The three primary stages of the Scrum process are the pre-sprint planning phase, the sprint itself, and the post-sprint planning stage. Planning is done before the sprint: The "release backlog" is a repository for all the work that must be completed on the system. During the planning before each sprint, some features and functionality from the release backlog are chosen and moved into the "sprint backlog," a prioritized collection of activities expected to be finished during the subsequent sprint. Sprint: Once the pre-sprint planning phase has been completed, the teams are "told to sprint to achieve their objectives" and given their sprint backlogs (Kamepally & Nalamothu, 2016). The tasks within the sprint backlog have been frozen and will continue to be inflexible for the remainder of the sprint. The team members decide the tasks they wish to work on and then start the development process. Scrum requires that participants participate in brief sessions daily. Scrum meetings are held every morning to improve communication and inform customers, developers, and management on the project's progress. These meetings also identify any problems that may have arisen and keep the entire team focused on a common goal. Post-sprint meeting: Following the conclusion of each sprint, a post-sprint meeting is held to evaluate the progress of the project and present the existing system. The following are the core tenets of Scrum:

- The use of smaller working teams with the goals of maximizing communication, minimizing overhead, and maximizing the sharing of implicit and informal knowledge.
- The ability to adjust to changes in the marketplace (users/customers), whether technical or otherwise, to deliver the highest quality product feasible.
- Regular "builds," also known as the production of executables, can be inspected, adjusted, tested, and recorded to be expanded upon.
- The separation of individual tasks and team responsibilities into separate, low-coupling partitions, often known as packets.
- Continuous testing and documentation of a product throughout the construction process.
- The capacity to "declare done" on a product whenever this requirement arises.

CRYSTAL METHODOLOGIES LIFE CYCLE

Crystal Methodologies emphasize planning and project management. Alistair Cockburn's early 1990s Crystal Methodologies. Crystal Methodologies address the issue of inadequate communication in product development. Okburn believes face-to-face encounters can replace written documentation, reducing reliance on written work products and increasing

system delivery likelihood. More regular running, tested system slice delivery allows this (Insfran et al., 2014). Highsmith says Crystal Methodologies focus on people, interaction, community, skills, abilities, and communication as first-order performance impacts. The process is crucial but secondary. Like a gemstone, Cockburn's methods are "crystal"—each facet represents a different version of the process around an identical core. Other methods assigned colors in rising cloudiness. The most Agi Ie form is Crystal Clear, then Crystal Yellow, Orange, Red, etc. Crystal Kind relies on several participants, emphasizing communication differently. More persons in the project mean more opaque crystals. Methodologies become stricter as project criticality increases. Methodologies can also be adjusted for productivity or legality.

Every Crystal methodology starts with a fundamental collection of roles, work products, procedures, and notations. This entire collection is then enhanced when the team size increases, or the methodology becomes more stringent. More constraints always result in a less agile process; nonetheless, Highsmith emphasizes that even with the additional constraints, they are still elegant due to a shared mentality.

AGILE METHODOLOGIES IN WEB ENGINEERING

Software Engineering does not include the capabilities of the web and the Internet. The systematic creation of web applications is accomplished by applying software engineering concepts in conjunction with Internet and web technologies. When implementing established approaches, web engineering presents some difficulties due to the increased number of components that must be focused on (Perera, 2011). The following are some areas for improvement of traditional processes for web engineering, such as Waterfall and Spiral, which lead to the application of agile methods in web engineering. The following are some of the restrictions that are placed on typical web engineering practices:

- A method that needs to be more rigorous and systematic.
- The user does not want the entire system to be available to them.
- More costs are incurred because the system is not developed within the allotted period. A lack of capacity for expansion and difficulty in maintaining,
- Failure to achieve the performance requirements results in the waste of resources.
- The creation, testing, evaluation approaches, quality assessment, and control should be paid more attention.
- Relying heavily on its various methods of personal growth and development.
- There is a need for more awareness regarding the stages of its life cycle.
- Requirements investigation, re-design, development (including coding), administration, measurement, and ongoing maintenance.
- Requires significant enhancements as well as judgments regarding the system's design.
- This is not an event; instead, it is a practice. Concerns are warranted regarding the process by which they are manufactured and the long-term quality and integrity of the products.
- Please do not rely on the early users to debug the Web App; instead, build rigorous tests and put them through their paces before deploying the system.

The practice of Web Engineering, by its very nature, adheres to several agile concepts. The production of a practical web application requires adhering to all of the agile principles. The three primary models that have been around for the construction of software applications are: There is effectiveness in-

- The software model,
- The business model, and
- The domain model.

On the other hand, a creative design model is essential to construct a web-based application. Agile aids in the development of software in a more productive way. The customer will have uninterrupted access to the software while it is being developed; however, modifications may be made at any point during the project's development, and the team's ability to collaborate will be high.

AGILE WEB ENGINEERING PROCESS (AWE)

The agile web engineering process is one of the procedures that has been proposed expressly for the creation of web applications. It is one of the ways that agile concepts can be applied. Creating web-based apps can present several challenges, but agile web engineering, a lightweight method, can help developers overcome these challenges. AWE contributes to improved application maintenance, implementation, and testing that is performed continuously. The end consumers will be delighted with the solutions that can be obtained with the assistance of this AWE solution. The figure1 depicts the many stages of the life cycle. The business analysis or evolution phase is excellent for constructing a web application. For applications that have already been initiated, we will go on to the evaluation step (Samadhiya & Chang, 2015).

In this AWE, there will be a high level of interaction between users, and Web engineering is a multidisciplinary field that calls for additional evaluations and collaborations. On the other hand, software engineering places a primary emphasis on its usefulness; as a result, there will be fewer evaluations, and software development will take less time. Compared to software engineering, which requires just a small number of teams and makes it very simple for those teams to concentrate on their work together, developing a web application that can communicate with other apps necessitates using a more significant number of teams. The nature of the teams involved in web engineering is that they draw from multiple disciplines, but most of the time, they cannot communicate with one another (Roy et al., 2021b). As a result, rather than presenting extensive paperwork, they prefer to rely on frequent informal communication that takes place face-to-face. Documentation is required, and comprehensive collaboration is not available now. Despite its flaws, the Agile web engineering method has several redeeming qualities, including adaptability, user-centered design, simplicity, iterative and incremental development, strong customer focus, and usability. Put into practice, this agile web engineering approach presents several significant difficulties and restrictions. In contrast, applying agile approaches to the software engineering process is significantly more accessible than for web engineering.

AGILE METHODOLOGIES IN SOFTWARE ENGINEERING

Lean and agile software development assist in adapting to changing needs and prioritize customer collaboration throughout the software engineering process. The team uses a light methodology and a short development cycle. Lightweight software engineering approaches enable flexibility in adapting to changing needs at any point. Due to the limited multidisciplinary focus of software applications, just a few teams are assigned to projects to facilitate decision-making. Since architectural design is minimal, teams can work on it simultaneously without complications. Agile approaches can minimize costs, focus on

current requirements, and eliminate the need for detailed documentation, allowing engineers to focus on requirements (Maddali et al., 2020).

ADAPTABILITY IN WEB ENGINEERING AND SOFTWARE ENGINEERING

This section compares applying agile approaches for web and software engineering. Agile Development was first designed for software engineering but has been adapted for web development (Chatzigeorgiou et al., 2016). Some developers opt for a different methodology, such as the Waterfall Method, instead of agile principles. Agile principles are supported by the high priority of customer happiness and the low cost of speedy development in web development. Web development can support agile development in various ways. Web applications that are released immediately encourage user and client input, facilitating consistent contact between developers and stakeholders. Web engineering and agility are often linked, with Ruby on Rails being a popular agile framework. —Ruby on Rails, an agile web development framework, prioritizes convention over configuration and avoids repetition, simplifying the development process and boosting productivity for online application prototyping (Graciamary & Chidambaram, 2016). Although incorporating agile principles like incremental communication, web engineering requires more teams and a more structured methodology. Agile development can complement prominent web engineering methods like Model-Driven web engineering. However, agile development can be challenging when a company expands and must adapt to new demands. Some argue that agile development has minimal impact on web application success.

For instance, due to limited functionality, agile web engineering prioritizes documentation above software engineering. Emphasis on non-functional needs will be prioritized above functional requirements during communication and testing at the browser level (Kaluvakuri et al., 2020). Agile SE methods aim to reduce development time but face the same challenges as traditional software engineering. Software Engineering lessens the importance of content, such as user interface, graphics, and data presentation. Non-software developers, such as graphic artists, writers, and usability/HMI experts, often develop content. Web engineering transcends SE due to its nature, multidisciplinary teams, big functional teams, and intricate architectural difficulties in Web App projects. Web engineering requires collaboration among several positions. Agile web teams execute the same task with diverse expertise but require more trained human resources. Agile development has numerous software and web engineering applications due to its flexible methods and non-strict rules. These principles can improve the development of almost any application.

CONCLUSION

The many agility applications to web engineering parallel the many other software engineering applications. Because the methods are flexible and the rules do not need to be followed strictly, the agile principles can help improve the development of almost any application. Some agile principles, such as the incremental development evolutionary model, are involved in web engineering. Agile methods of web engineering adopt some agile methods in software engineering; however, some issues are associated with applying agile methodologies in web engineering. There are a few significant differences between software engineering and web engineering. The techniques covered in this paper are given as a solution to some of the most prevalent issues in web engineering. Different firms follow their methodology depending on the environment. All of the proposed techniques have yet

to successfully fix all of web engineering's problems, although some have. However, employing agile methods in web engineering is beneficial for developing smaller online applications. Because of the multidisciplinary character of web engineering and its more significant number of components, agile methodologies in web engineering follow some of the techniques that go beyond agile software engineering. The only difference and difficulty in implementing agile methodologies for web engineering is a continuous change in requirements and some architectural issues.

REFERENCES

Chatzigeorgiou, A., Theodorou, T. L., Violettas, G. E., Xinogalos, S. (2016). Blending an Android Development Course with Software Engineering Concepts. *Education and Information Technologies*, *21*(6), 1847-1875. https://doi.org/10.1007/s10639-015-9423-3

Del, Á. I. M., Palma, J., Túnez, S. (2014). Milestones in Software Engineering and Knowledge Engineering History: A Comparative Review. *The Scientific World Journal*, *2014*, https://doi.org/10.1155/2014/692510

Farid, A. B. (2015). Proactive Software Engineering Approach to Ensure Rapid Software Development and Scalable Production with Limited Resources. *International Journal of Advanced Computer Science and Applications*, *6*(11). https://doi.org/10.14569/IJACSA.2015.061120

Ghazi, P., Glinz, M. (2017). Challenges of Working With Artifacts in Requirements Engineering and Software Engineering. *Requirements Engineering*, *22*(3), 359-385. https://doi.org/10.1007/s00766-017-0272-z

Graciamary, A. C., Chidambaram. (2016). Enhanced Re-Engineering Mechanism to Improve the Efficiency of Software Re-Engineering. *International Journal of Advanced Computer Science and Applications*, *7*(11). https://doi.org/10.14569/IJACSA.2016.071136

Insfran, E., Chastek, G., Donohoe, P., Leite, D. P. J. C. S. (2014). Requirements Engineering in Software Product Line Engineering. *Requirements Engineering*, *19*(4), 331-332. https://doi.org/10.1007/s00766-013-0189-0

Kaluvakuri, S. (2022). Revolutionizing Healthcare: The Impact of Robotics on Health Services. *Malaysian Journal of Medical and Biological Research*, *9*(2), 41-50. https://mjmbr.my/index.php/mjmbr/article/view/680

Kaluvakuri, S., & Amin, R. (2018). From Paper Trails to Digital Success: The Evolution of E-Accounting. *Asian Accounting and Auditing Advancement*, *9*(1), 73–88. https://4ajournal.com/article/view/82

Kaluvakuri, S., & Lal, K. (2017). Networking Alchemy: Demystifying the Magic behind Seamless Digital Connectivity. *International Journal of Reciprocal Symmetry and Theoretical Physics*, *4*, 20-28. https://upright.pub/index.php/ijrstp/article/view/105

Kaluvakuri, S., Maddali, K., Rahimi, N., Gupta, B., Debnath, N. (2020). Generalization of rc-based low diameter hierarchical structured p2p network architecture. International Journal of Computers and Their Applications, 27(2), 74-83. https://isca-hq.org/Documents/Journal/Archive/2020volume27-2.pdf

Kamepally, A. K., Nalamothu, T. (2016). Agile Methodologies in Software Engineering and Web Engineering. *International Journal of Education and Management Engineering*, *6*(5), 1. https://doi.org/10.5815/ijeme.2016.05.01

Maddali, K., Kaluvakuri, S., Roy, I., Liu, Z., Gupta, B., Debnath, N. (2020). Generalizing Chinese Remainder Theorem Based Fault Tolerant Non-DHT Hierarchical Structured Peer-to-Peer Network. *International Journal of Computers and their Applications,* *27*(4), 150-157. https://isca-hq.org/Documents/Journal/Archive/2020volume27-4.pdf

Maddali, K., Kaluvakuri, S., Roy, I., Rahimi, N., Gupta, B., Debnath, N. (2021). A Comprehensive Study of Some Recent Proximity Awareness Models and Common-Interest Architectural Formulations among P2P Systems. *International Journal of Computers and their Applications, 28*(4), 179-192.

Maddali, K., Rekabdar, B., Kaluvakuri, S., Gupta, B. (2019). Efficient Capacity-Constrained Multicast in RC-Based P2P Networks. In Proceedings of 32nd International Conference on Computer Applications in Industry and Engineering. *EPiC Series in Computing, 63*, 121–129. https://doi.org/10.29007/dhwl

Perera, I. (2011). Impact of Poor Requirement Engineering in Software Outsourcing: A Study on Software Developers' Experience. *International Journal of Computers, Communications and Control*, *6*(2), 337-348. https://doi.org/10.15837/ijccc.2011.2.2182

Roy, I., Kaluvakuri, S., Maddali, K., Aydeger, A., Gupta, B., Debnath, N. (2021a). Capacity Constrained Broadcast and Multicast Protocols for Clusters in a Pyramid Tree-based Structured P2P Network. *International Journal for Computers & Their Applications, 28*(3), 122-131. https://isca-hq.org/isca.php?p=2021volume2803

Roy, I., Kaluvakuri, S., Maddali, K., Liu, Z., Gupta, B. (2021b). Efficient Communication Protocols for Non DHT-based Pyramid Tree P2P Architecture. *WSEAS Transactions on Computers, 20*, 108-125. https://doi.org/10.37394/23205.2021.20.13

Roy, I., Kaluvakuri, S., Maddali, K., Liu, Z., Gupta, B., Debnath, N. (2020). Novel Design of Load-Balanced and Fault-Tolerant Multicasting Protocols for PIM-SM. *International Journal of Computers and their Applications,* *27*(4), 158-167. https://isca-hq.org/Documents/Journal/Archive/2020volume27-4.pdf

Roy, I., Rahimi, N., Kaluvakuri, S., Maddali, K., Gupta, B., Debnath, N. (2021c). Design of Efficient Broadcast Protocol for Pyramid Tree-based P2P Network Architecture. *EPiC Series in Computing, 75*, 80-89. https://doi.org/10.29007/8ws1

Samadhiya, D., Chang, W. C. (2015). Toward Meta-Software Engineering Multistage Process. *Applied Mechanics and Materials*, *764-765*, 983-987. https://doi.org/10.4028/www.scientific.net/AMM.764-765.983

Song, X. J., Zeng, Z. L. (2014). Research on Application of Software Engineering Theory in Software Development. *Applied Mechanics and Materials*, *687-691*, 1921-1924. https://doi.org/10.4028/www.scientific.net/AMM.687-691.1921

--0--