International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 4
Article Received: *25 January 2023* ***Revised:*** *12 February 2023* ***Accepted:*** *30 March 2023*

A Systematic Approach for Categorizing Different Roles in Software Engineering Process Models

Md Asghar Ali[1] and Dr. Ajay Jain[2]
Research Scholar[1] , Research Guide[2]
[1,2]Department of Computer Science & Engineering,
[1,2]Dr.A.P.J. Abdul Kalam University, Indore (MP) , India

Abstract**—**Program frameworks come and go through a arrangement of entries that account for their beginning,introductory advancement, beneficial operation, upkeep, and retirement from one era toanother. This article categorizes and analyses a number of strategies for depicting or modellinghow program frameworks are created. It starts with foundation and definitions of conventionalcomputer program life cycle models that overwhelm most course reading talks and current computer programimprovement hones. This is often taken after by a more comprehensive survey of the electivemodels of program advancement that are of current utilize as the premise for organizing programdesigning ventures and advances.In differentiate to program life cycle models, computer program handle models frequently speak to a organizedgrouping of exercises, objects, changes, and occasions that encapsulate procedures forfinishing computer program advancement. Such models can be utilized to create more exact andformalized portrayals of program life cycle exercises. Their control develops from theirutilization of an adequately wealthy documentation, language structure, or semantics, frequently reasonable for computationalpreparing. In this work, a process model was processed so as to obtain a generalized model.

Index Terms**—***Frameworks, Models, Elective, Encapsulate, Generalized*

I. INTRODUCTION

Traditional software development models have been with us since the early days of software. In this section, we identify four. The classic software life cycle (or "waterfall diagram")and step-by-step improvement models are widely represented in almost allmodern programming practices and software engineering books. The incremental release model is closely related to theindustry practices where it is most commonly found. Models based on military standardshave also transformed some forms of the classic life cycle model into practices required by governmentsubcontractors. All four of these models use coarse-grained or macroscopic characteristicsto describe software. The incremental stages of software development are often described asphases such as post definition, initial design and implementation; they usually havelittle or no properties other than the list of properties that the product of such astep should have.

In addition, these models are independent of the development settings of the organization, choice of programming language, domain of software, etc. In short,traditional models are context-free and not context-sensitive. But since all these life cycle modelshave been around for some time, we call them traditional models andcharacterize each one in turn.Software packages are knowledge-intensive artifacts thatare built incrementally and iteratively by software. Such efforts can be modelled usingsoftware life cycle models. These product development models represent an evolutionary version of traditional software life cycle models.

The changes were due to the availability ofnew software development techniques such as software prototypinglanguages and environments, reusable software, application generators, and documentationsupport environments. The goal of all these technologies is to enable the creation of executablesoftware implementations either early in software development or faster.Therefore,software development patterns in this regard may be implicit in the use oftechnology rather than explicitly stated. This is possible because such models are becomingincreasingly intuitive for developers whose favourable experience with these technologiessupports their use. Thus, a detailed review of these models is most appropriate when suchtechnologies are available for use or testing.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

II. LITERATURE REVIEW

Subbarayudu et. al. states that no one can deny the importance of computer in our life, especially during the present time. In fact, computer has become indispensable in today's life as it is used in many fields of life such as industry, medicine, commerce, education and even agriculture. It has become an important element in the industry and technology of advanced as well as developing countries. Now a day, organizations become more dependent on computer in their works as a result of computer technology. Computer is considered a time-saving device and its progress helps in executing complex, long, repeated processes in a very short time with a high speed. In addition to using computer for work, people use it for fun and entertainment. Noticeably, the number of companies that produce software programs for the purpose of facilitating works of offices, administrations, banks, etc, has increased recently which results in the difficulty of enumerating such companies. During the previous four decades, software has been developed from a tool used for analysing information or solving a problem to a product in itself. However, the early programming stages have created a number of problems turning software an obstacle to software development particularly those relying on computers. Software consists of documents and programs that contain a collection that has been established to be a part of software engineering procedures. Moreover, the aim of software engineering is to create a suitable work that constructs programs of high quality. [1]

Valmohammadi et. al. states that the primary objective of this research is to propose a green process model for software development. The study employed a mixed-method approach, incorporating the interpretive paradigm in the initial stage and the positivist paradigm in the subsequent stage. The research methodology encompassed a systematic literature review, qualitative content analysis, and interviews with academic and industry experts. A conceptual model was proposed, and corresponding hypotheses were derived. The hypotheses were subsequently tested using the structural equation modelling (SEM) technique, aided by SmartPLS software. The dataset consisted of responses from 200 participants who completed the designed questionnaire. Qualitative data analysis was conducted using Maxqda2020 software. The collected data underwent both exploratory factor analysis (EFA) and confirmatory factor analysis (CFA) for analysis purposes. The findings revealed that five key factors, namely "Green readiness," "infrastructure," "methods," "tools," and "emerging trends," Exhibited a noteworthy and favourable impact on green software processes. Furthermore, the variables "tools," "infrastructure," and "emerging trends" were identified as partial mediating factors. Additionally, the dimensions of green readiness, which encompassed "governance," "strategy," "policies," "monitoring," and "stakeholders," were validated and recognized. The findings of this study provide valuable insights to organizations involved in software development, offering a holistic understanding of the importance and interrelationships among the main dimensions of green process modelling. By considering these findings, organizations can enhance their approaches to developing environmentally sustainable software. This study's main contribution is in the context of the software industry.

[2] **Hou et. al.** states that Large Language Models (LLMs) have significantly impacted numerous domains, including Software Engineering(SE). Many recent publications have explored LLMs applied to various SE tasks. Nevertheless, acomprehensive understanding of the application, effects, and possible limitations of LLMs on SE is still in itsearly stages. To bridge this gap, we conducted a systematic literature review (SLR) on LLM4SE, with a particularfocus on understanding how LLMs can be exploited to optimize processes and outcomes. We select andanalyse 395 research papers from January 2017 to January 2024 to answer four key research questions (RQs).In RQ1, we categorize different LLMs that have been employed in SE tasks, characterizing their distinctivefeatures and uses. In RQ2, we analyse the methods used in data collection, preprocessing, and application,highlighting the role of well-curated datasets for successful LLM for SE implementation. RQ3 investigatesthe strategies employed to optimize and evaluate the performance of LLMs in SE. Finally, RQ4 examines thespecific SE tasks where LLMs have shown success to date, illustrating their practical contributions to thefield. From the answers to these RQs, we discuss the current state-of-the-art and trends, identifying gapsin existing research, and flagging promising areas for future study [3] **Neogi et. al.** states that Agility is the keyword if one needs to survive inthis rapidly changing world, keeping pace and at the sametime not losing balance or control which could result in lossof quality of performance. No wonder agile methods haveemerged in the field of software development as well, andthey are gaining popularity in academics also. The bunchesof agile methods evolved so far are not only able to deliversoftware products quickly and easily but also make the agileteam to think quickly and in an intelligent way. Agilemethods are people centric or people driven softwareprocess. The objective of this study is to survey groups ofstudents and gather an opinion regarding their programdevelopment techniques, their preferences in choosingprogramming partners, their views about what affectsquality of a product. The ultimate aim is to identify

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

apattern which we claim is agile like, although most studentsare unaware of the existence of agile process models. [4]

III. PROGRAM EVOLUTION MODELS

Unlike the previous four prescriptive product development models is to develop a descriptive model of software product development. They are:

- Continuous change: a large software system is constantly changing or gradually becoming less useful
- Increasing complexity: As a software system evolves, its complexity increases unless efforts are made to maintain or reduce it
- Basic Law of Program Development: Program development, the programming process, and global measures of project and system attributes. are statistically self-regulating with definable trends and invariants
- Invariant workload: the rate of global work in a large software program is statisticallyinvariant
- Growth limit: over a large active life cycle, the number of changes made to successive releases of program is statistically invariant. However, it is important to note that these are global properties of large software systems, not the causal mechanisms of software.

IV. THE ROLE OF SOFTWARE PROCESS MODELS IN SOFTWARE DEVELOPMENT

Process models play a key role in ensuring the smoothness of software projects. They provide a common language and understanding for all team members, which promotes effective collaboration and communication.Imagine a software development team without a process model. Each team member may have their own way of approaching a project, leading to confusion and inefficiency.When a process model is used, everyone is on the same page with standardized procedures and guidelines.Process models also help mitigate risk by identifying potential pitfalls early in the development phase.Teams can identify potential bottlenecks, dependencies, and risks by breaking down the software development process into manageable steps. This allows them to address these issues proactively, minimizing the likelihood of project delays or failures.Software process models also facilitate project planning and resource allocation.By clearly defining the functions and tasks of each stage of the development process, teams can more accurately estimate the time and resources needed. This helps establish realistic project schedules and budgets.

V. KEY COMPONENTS OF SOFTWARE PROCESS MODELS

Software process models contain several key components that are essential for successful software development. These parts include:

- **Requirements gathering and analysis**

Requirements gathering and analysis is the first step in the software development process. This involves understanding the customer's needs and translating them into clear and concise requirements. This step lays the foundation for the entire project and ensures that the software meets the client's expectations.

- **Design and architecture**

Design and architecture focus on creating software. This includes designing the overall structure, defining modules and components, and creating their relationships. This phase lays the groundwork for the coding and implementation phase.

- **Coding and implementation**

The real development happens in coding and implementation. Developers write code based on design specifications, bringing software to life. This step requires attention to detail and adherence to coding standards to ensure a high-quality end product.

- **Testing and quality assurance**

Testing and quality assurance are crucial to ensure that software meets desired quality standards. Thisphase involves various testing techniques such as unit testing, integration testing and system testing to identify and fix possible errors or problems.

- **Implementation and maintenance**

Development and maintenance include releasing software to end users and providing ongoing support. This phase includes activities such as installation, configuration and user training. Maintenance, such as bug fixes, updates, and improvements, is performed to keep the software functional and up-to-date. By following these key components, software development process models provide a structured approach to software development and ensure that all necessary steps are taken to deliver a successful product.

VII. DIFFERENT TYPES OF SOFTWARE PROCESS MODELS

Several different process models can be used in software. Each model suits different project requirements and teams

and offers unique advantages and disadvantages. Let's look at some popular software process models:

- **Waterfall Model**

The Waterfall Model is a linear, sequential approach that is strictly top-down. It follows a structured flow where each stage of the development cycle must be completed before moving on to the next. This model is ideal for projects with well-defined and stable requirements. One of the main advantages of the waterfall model is its simplicity. The linearity of the process makes it easy to understand and implement. Clear documentation and well-defined milestones make it easy to manage and track progress. But the waterfall model has its limitations. Because it follows a strict order, accepting changes to requirements can be difficult. Lack of flexibility can cause delays and increase costs when changes are needed later in the development cycle.

- **Agile Model**

Agile design, on the other hand, emphasizes flexibility and adaptive design. It encourages iterative development, allowing teams to deliver working software, often in short iterations. This model is suitable for projects where requirements change and customer feedback must be responded to quickly. One of the main principles of the Agile model is collaboration. The development team works closely with stakeholders, constantly seeking feedback and incorporating it into the development process. This iterative approach enables continuous improvement and ensures that the final product meets the customer's expectations. But the agile model also has its challenges. An emphasis on flexibility and repeated iterations can sometimes lead to scope where a project expands beyond its original boundaries. In addition, the lack of a rigorous plan can make it difficult to accurately estimate project schedules and costs.

- **Iterative model**

The iterative model focuses on incremental development where each iteration produces working software. This approach enables early prototyping and testing, enabling feedback-based improvements during development. One of the main advantages of an iterative model is its ability to handle changing requirements. The team can adapt to changes and take feedback at each stage by breaking the development process into smaller iterations. This flexibility ensures that the final product meets the changing needs of the customer. However, the iterative model also has its drawbacks. The need for constant feedback and improvements can sometimes lead to a longer development cycle. Managing multiple iterations simultaneously can be challenging, requiring effective communication and coordination among team members.

VII. CONCLUSION

Understanding programming process models is essential to successful software development. These models provide a structured approach that enables teams to efficiently deliver quality software.By choosing the right model, teams can ensure the smooth progress of their projects through effective risk management and quality assurance practices. So, the next time you start a software development project, consider the right software development process model to help you succeed.Whether an IT professional or new to technology, the Data Institute's software development program is designed to give you hands-on experience in programming, web, and user interface development.

REFERENCES

[1] B Subbarayudu, Srija Harshika D, E. Amareswar, R Gangadhar Reddy, Kishor Kumar Reddy C, " REVIEW AND COMPARISON ON SOFTWARE PROCESS MODELS", International Journal of Mechanical Engineering and Technology (IJMET) Volume 8, Issue 8, August 2017, pp. 967–980

[2] Changiz Valmohammadi, Farkhondeh Mortaz Hejri, "Designing a conceptual green process model in software development: A mixed method approach", International Journal of Information Management Data Insights, 2023 Published by Elsevier Ltd

[3] Xinyi Hou, Yanjie Zhao, Yue Liu, Zhou Yang, Kailong Wang, Li Li,, Xiapu Luo, David Lo, Haoyu Wang, "Large Language Models for Software Engineering: A Systematic Literature Review", ACM Trans. Softw. Eng. Methodol., Vol. X, No. Y, Article 1. Publication date: December 2024.

[4] Madhumita Neogi , Vandana Bhattacherjee, Rupa Mahanti, "A Process Model for Software Development Amongst Students", International Journal of Recent Trends in Engineering Vol. 1, No. 2, May 2009.

[5] Mayank Agarwal, Yikang Shen, Bailin Wang, Yoon Kim, and Jie Chen. 2024. Structured Code Representations EnableData-Efficient Adaptation of Code Language Models. arXiv preprint arXiv:2401.10716 (2024).

[6] Emad Aghajani, Csaba Nagy, Mario Linares-Vásquez, Laura Moreno, Gabriele Bavota, Michele Lanza, and David CShepherd. 2020. Software documentation: the practitioners' perspective. In Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering. 590–601.

[7] Lakshya Agrawal, Aditya Kanade, Navin Goyal, Shuvendu K Lahiri, and Sriram Rajamani. 2023. Monitor-GuidedDecoding of Code LMs with Static Analysis of Repository Context. In Thirty-seventh Conference on Neural InformationProcessing Systems.

[8] Baleegh Ahmad, Shailja Thakur, Benjamin Tan, Ramesh Karri, and Hammond Pearce. 2023. Fixing Hardware SecurityBugs with Large Language Models. arXiv preprint arXiv:2302.01215 (2023).

[9] Wasi Uddin Ahmad, Saikat Chakraborty, Baishakhi Ray, and Kai-Wei Chang. 2021. Unified pre-training for programunderstanding and generation. arXiv preprint arXiv:2103.06333 (2021).

[10] Toufique Ahmed, Kunal Suresh Pai, Premkumar Devanbu, and Earl T Barr. 2023. Improving Few-Shot Prompts withRelevant Static Analysis Products. arXiv preprint arXiv:2304.06815 (2023).

[11] Ajmain I Alam, Palash R Roy, Farouq Al-Omari, Chanchal K Roy, Banani Roy, and Kevin A Schneider. 2023. GPTCloneBench:A comprehensive benchmark of semantic clones and cross-language clones using GPT-3 model andSemanticCloneBench. In 2023 IEEE International Conference on Software Maintenance and Evolution (ICSME). IEEE,1–13.