An Analytical Comparison of Software Project Estimation Techniques

Malav Mehta
Department of Computer Engineering
Mukesh Patel School of Technology Management and Engineering,
Mumbai, India

Vedaant Melkari
Department of Computer Engineering
Mukesh Patel School of Technology Management and Engineering,
Mumbai, India

Avani S. Bhuva
Department of Computer Engineering
Mukesh Patel School of Technology Management and Engineering,
Mumbai, India

Harsh Motiramani
Department of Computer Engineering
Mukesh Patel School of Technology Management and Engineering
Mumbai, India

ABSTRACT
Accurate project estimation marks the basis of successful software development. It provides a gap for proper resource management, and then it installs correct project scheduling and budgeting methods. This review paper highlights six widely used estimation models for software projects. In detail, three methods of effort estimation catered to various types of teams and types of projects respecting traditional or Agile setup were highlighted. The review mentioned the estimation models; such as COCOMO, Source Lines of Code, Wideband Delphi, Expert-Based Estimation, Planning Poker, and Function Point Analysis. It covers the methods, strengths, and weaknesses of each estimation technique discussed below as compared against four estimation techniques based on selection criteria including accuracy, suitability to the size of a project, adaptability, and adequacy. This review will help managers of projects, developers, and stakeholders in deciding on which most suitable strategy is tailored for their requirements through such evaluation of these four methods. Lastly, this paper contributes to the global understanding of how estimation techniques can be strategically chosen to minimize risks and enhance predictability in software engineering projects.

Keywords
COCOMO, Expert-Based Estimation, Function Point Analysis, SLOC, Wideband Delphi, Planning Poker

1. INTRODUCTION
Software estimation techniques play a critical role in the planning and execution of software development projects. They provide a systematic approach to forecasting key project parameters such as effort, cost, time, and resources, enabling teams to make informed decisions and deliver projects successfully. Accurate estimation is crucial for managing stakeholder expectations, mitigating risks, and ensuring that projects stay on track within defined constraints. Over the years, a variety of estimation methods have emerged, each tailored to address specific project needs, complexities, and scales. From algorithmic models like COCOMO to experience-based approaches such as Expert Judgment, and collaborative techniques like Planning Poker, these methods vary in their accuracy, scalability, required inputs, and suitability for different project types. While some techniques emphasize quantitative precision, others focus on flexibility and adaptability, reflecting the diverse nature of software development environments.

In this review, we explore the most widely used software estimation methods, highlighting their unique characteristics, advantages, limitations, and industry use cases. By understanding these techniques, project managers and teams can choose the most appropriate approach to enhance project predictability and achieve better outcomes in today's dynamic and competitive software development landscape.

Accurate project estimation becomes critical in software engineering in regard to managing resources, a timeline, and overall delivery of the project. Every budgeting aspect to how stakeholders are satisfied or otherwise depends on it. Taking into account the messy as well as often unpredictable aspects of software development, it is no wonder that different techniques have been developed just to overcome these challenges-and each technique has its different ways and applications.

This paper is a review and comparison of six leading software estimation methods: COCOMO, Expert-Based Estimation, Function Point Analysis, Source Lines of Code (SLOC), Wideband Delphi, and Planning Poker. Every estimation method has a set of strengths that match certain project needs, team structures, and development environments from traditional waterfall approaches to agile methodologies. These explorations aim to provide broad understanding about the pros and cons of ideal applications of each technique. The aim of the paper is to guide the software practitioners to decide on an appropriate estimation practice for their specific project requirement, which concerns attributes such as precision, user-friendliness, ease of adaptation, and efficiency. The minute comparison done should raise the most appropriate practices for the various kinds of scenarios of software development projects for better decision-making and quality project results.

2. PROBLEM STATEMENT
Estimating project effort, time, and cost in software development presents significant challenges due to the complex and dynamic nature of project requirements. Furthermore, the application of different methodologies across various projects complicates the estimation process. Inaccurate estimations can result in budget overruns, missed deadlines, and dissatisfied stakeholders, ultimately jeopardizing the project's success. Various estimation methods, including COCOMO, Expert-Based Estimation, Function Point Analysis, SLOC, Wideband Delphi, and Planning Poker, exhibit distinct strengths, weaknesses, and applicability.

The absence of a universally applicable approach makes it difficult for project managers and development teams to select the most suitable method for their specific project types. The lack of general guidelines regarding the most effective estimation techniques across diverse project environments

exacerbates this issue. To address this gap, this paper provides a comprehensive comparative review of the prominent methods utilized in software estimation, evaluating their suitability and effectiveness in relation to different types of software projects.

3. ESTIMATION METHODS

The following outlines the different effort estimation methods and their respective details, as derived from the reviewed papers.

3.1 COCOMO (Constructive Cost Model)

The methodology presented in the paper, known as the COCOMO (Constructive Cost Model) suite, seeks to deliver a thorough estimation for the development of software systems as well as software-intensive systems of systems (SoS). Created by Barry Boehm, the Constructive Cost Model is an algorithmic framework that is extensively utilized to assess the cost, effort, and time necessary for software development, taking into account the project's scale and complexity. By modeling the software project through a series of mathematical relationships, it links various project characteristics to the required development effort, thereby providing a systematic and adaptable approach to estimation.

Process:

Variants of COCOMO: The model comprises three variants that are suited to different needs of projects:

1. Simple: It does a pre-approximation from the scale alone.
2. Intermediate: Attributes of the projects beyond software scale, database size and software reliability. These would be used in approximations.
3. Detailed: It relies on the phase to phase nature of the projects in approximating fine-graded elements of the software modules.

Input Factors:

1. Scale Factors: Scale dependent as it adjusts on the scales based on parameters like experience of the teams or constraints of technology set on.
2. Effort Multipliers: Adjustment of efforts. Aspects of a product relate to its dependability; its complexity and speed that it is achieved.

Adjustability:

The capability of COCOMO for the scaling of various types of projects and the support of the different requirements of project scale make this model apt for a project of any size, small-scale systems to large products of software. All the input parameters must be precise enough to ensure good estimation accuracy.

3.2 Expert-based software estimation:

The paper (Much more than a prediction: Expert-based software effort estimation as a behavioral act) discusses some techniques of software estimation, mostly focusing on the technical prediction approach and the behavioral approach. In contrast to algorithmic models like COCOMO, expert-based estimation leverages the knowledge and experience of seasoned professionals. This qualitative approach allows experts to draw on historical data and contextual insights to approximate the effort required for a project.

Process:

1. To Gather Expert Input: There is derivation of estimates which comes from consulting experts or those using historical data along with their professional insights to have a measure of effort needed.
2. Synthesizing Estimates: An agreement-based approach can be used by taking the averages or sums of various experts' estimates to get a consensus number.

Advantages:

This method takes into account specific project characteristics that the metric-based models might miss.

Limitations:

This estimation is subjective as it is based on experts' views, and the accuracy will depend on the appropriateness and richness of experience of the expert. Poor results may occur when adequate experts are not available.

3.3 Function Point Estimation:

The paper Function Point Estimation Methods A Comparative Overview provides a comprehensive overview of various methods for estimating the size of software applications using Function Points (FP). Function Point Analysis (FPA) offers a quantitative method to measure the size of a software project by analyzing its functional requirements. It is particularly effective for data-intensive applications, providing insights into both the inputs and outputs of a system.

Process:

1. Identify the functions: Analyze functional requirement by identifying external inputs, outputs, user interaction files, and external interfaces.
2. Determining Function Points: The functions are assigned complexity weights; the total function points work out to approximate the size of the project.

Usage:

Function Point Estimation especially applies to data-intensive applications and is less applicable when the systems are more computation based. It provides a mathematical or systematic way of size measuring in projects where the handling of data is predominant in the system.

3.4 Source Lines of Code (SLOC)

As cited in the paper *Effort Estimation in Agile Software Development using Story Points*, Source Lines of Code (SLOC) refers to another widely used measure of the size of a project, according to the number of lines of code in a system. It is a foundational measure in estimating effort and cost for many software projects.

Formulation:

SLOC is calculated counting either the physical lines of code or logical statements.

Effort Estimation:

SLOC can be brought into relation with productivity measures using historical data to get an effort estimate.

Drawbacks:

SLOC depends on the project and also varies due to the code style and the programming language that is used. It is apt for simple projects but less suitable for those projects requiring more than the volume of code.

3.5 Wideband Delphi:

Wideband Delphi is an estimation technique, on the basis of the

Delphi method that uses series of repeated rounds of estimating by various experts with the iterative refinement using discussion and consensus building among them.

Process:

1. Individual Estimation: Each person makes his individual estimate in the first round.
2. Group Discussion: All the individual estimates are taken by the group and after that further discussion is conducted with each other to clarify differences.
3. Refining Estimates: Participants update their estimates in the light of collective insight; this cycle repeats again and again until a general consensus is reached.

Benefits:

Wideband Delphi reduces individual biases by offering various views, allowing for iterative refinements to achieve more reliable estimates through consensus built by experts.

3.6 Planning Poker

Planning Poker is an estimation technique based on agreement; however, it is specific to be used in Agile. It is the estimating together of teams using simple, card-based collaboration technique where team members privately pick individually from a deck of cards values to best represent the estimated effort.

Process:

1. Choose: Each team member privates selects a card representing the relative effort best.
2. Discuss: Only the biggest and smallest estimates are those whose members discuss why.
3. Re-Estimation: After discussion, team members refine their choices until they come to a consensus-based estimate

Benefits:

Planning Poker encourages fair participation and eliminates hierarchical bias. This technique's collaborative nature fits well with Agile methodologies, where iterative team input improves the accuracy of the estimations.

4. RELATED WORK

Software effort estimation is a critical area in software engineering, with various methodologies proposed to improve accuracy and effectiveness. Matsubara, Steinmacher, Gadelha, and Conte (2023), in their paper "Much More Than a Prediction: Expert-Based Software Effort Estimation as a Behavioural Act," focus on expert-based software effort estimation, emphasizing the behavioural aspects of estimation and addressing biases in the process. Their work integrates qualitative methods, such as thematic analysis, and discusses collaborative estimation techniques like Planning Poker and the Wideband Delphi method. The study highlights the role of customer expectations and expert judgment in influencing software development efforts.

Coelho and Basu (2012), in their research paper "Effort Estimation in Agile Software Development Using Story Points," explore effort estimation in the context of Agile methodologies, focusing on the use of story points. Their research addresses the challenges and limitations of estimating the size and effort required for software development in Agile environments. The study emphasizes the importance of precise and reliable estimation techniques to ensure the success of Agile projects.

Meli and Santillo (1999), in their paper "Function Point Estimation Methods: A Comparative Overview," provide a comparative overview of function point estimation methods. Their research evaluates various approaches for estimating the size of software applications, stressing the importance of selecting estimation techniques that align with the specific context and objectives of a project. The study offers a detailed classification of methods based on their characteristics and applications.

Boehm and Valerdi (2005), in their work titled "COCOMO Suite Methodology and Evolution," investigate the evolution of the COCOMO suite, a widely recognized model for software effort estimation. Their research discusses the methodologies, scope, and refinements of the COCOMO suite to improve decision-making in software-intensive system development. The study also addresses overlapping activities, missing tasks, and collaborations with industry affiliates to enhance the accuracy of the model. Gandomani, Faraji, and Radnejad (2019), in their paper "Planning Poker in Cost Estimation in Agile Methods," analyze Planning Poker, an Agile estimation method that involves collaboration between developers and business representatives. This technique uses specialized cards to evaluate user stories based on complexity and risk, fostering consensus within teams. However, the study notes a potential limitation in the method's reliance on team members' memory and experience. Munialo and Muketha (2016), in their paper "A Review of Agile Software Estimation Methods," provide a comprehensive review of both traditional and Agile effort estimation methods. Their work examines various techniques, including Planning Poker, Wideband Delphi, and COCOMO, as well as traditional methods like top-down, bottom-up, and analogy-based estimation. The study acknowledges challenges such as scope, complexity, and uncertainty in software project management, offering valuable insights to improve estimation practices.

These studies collectively underscore the variety and progression of software estimation techniques, establishing a basis for enhancing project planning and management across various software development methodologies. The table below summarizes the research conducted from sources [1] to [6].

Table:1 Detailed comparison analysis of different estimation approaches

Estimation Method	Approach	Best Suited For	Required Inputs	Accuracy	Scalability	Advantages	Limitations	Industry Use Cases
COCOMO	Algorithmic model using regression-based equations; different models	Medium to large-scale projects	Project size, scale factors, effort multipliers, historical	Moderate to High; depends on accurate calibration	High	Provides structured, quantitative estimates; flexible for	Dependent on the accuracy of scale factors and multipliers; requires	Government and defense projects, large enterprise

	(Basic, Intermediate, Detailed) for increasing complexity		project data			adjusting based on project scale factors	domain knowledge for accurate tuning	software
Expert-Judgment	Experience-based estimation	Small	Expert	Varies; can be subjective	Low	Flexible, subjective	Subjective; requires experienced experts	Consultancy
Function Point Estimation	Quantifies software size based on functional requirements like inputs, outputs, and interactions	Data-intensive projects with clear functional requirements	Defined functional requirements, complexity weight factors (like user interactions)	Moderate; accurate if functional requirements are clear and detailed	Moderate	Useful for data-centric applications, irrespective of programming language; good for modular development	Time-consuming if functional requirements are complex or unclear; less suited for algorithm-heavy projects	ERP systems, data processing applications, banking systems
Source Lines of Code (SLOC)	Counts lines of code (physical/logical) to estimate effort; often combined with historical productivity rates	Code-centric projects where size can be correlated with effort	Code structure, codebase history, productivity metrics from past projects	Low to Moderate, straightforward but limited by coding style variations	Low	Simple, direct measurement of code volume; often used in maintenance projects with existing codebases	Ignores complexity and design; highly variable across languages and coding styles	Legacy system upgrades, maintenance projects
Wideband Delphi	Iterative, consensus-driven process among experts to refine and converge on estimates	Complex projects needing diverse input	Expert judgment, initial individual estimates, consensus-building process	High; reduces individual bias, results tend to stabilize across iterations	Moderate	Promotes balanced estimates by reducing biases; accommodates input from multiple experts	Time-intensive, requires availability of knowledgeable experts, who may have scheduling conflicts	Aerospace, research-driven projects, cross-functional teams
Planning Poker	Agile-based, collaborative estimation using relative sizing via team-based discussions	Agile projects, iterative development with team consensus	User stories, team knowledge, relative sizing based on Fibonacci sequence	Moderate, often sufficient for iterative sprints, though less precise for large projects	Low	Encourages active team involvement, reduces hierarchy bias, fast for Agile sprints	Limited to relative estimates, precision decreases for larger or highly complex tasks	Software startups, product development, Scrum projects

The above table provides a detailed comparison of various software project estimation approaches, highlighting their unique features, best-suited applications, required inputs, accuracy, scalability, advantages, limitations, and industry use cases. COCOMO, an algorithmic model, is ideal for medium to large-scale projects and offers structured, quantitative estimates, though its accuracy depends heavily on precise calibration of scale factors. Expert Judgment, suited for small projects, relies on experienced professionals, offering flexibility but subjectivity. Function Point Estimation quantifies software size based on functional requirements, making it effective for data-intensive projects, though it may struggle with complex or unclear requirements. Source Lines of Code (SLOC) is a straightforward approach for code-centric projects but is limited by variations in coding styles and ignores complexity. Wideband Delphi employs an iterative, consensus-driven process for complex projects, reducing bias but requiring significant time and expert availability. Lastly, Planning Poker, widely used in Agile projects, fosters team collaboration and quick estimates for iterative development but lacks precision for larger or highly complex tasks. Each method serves specific industry needs, from government and defense to Agile startups, making their selection context-dependent.

Presented here are several charts that illustrate specific details concerning the publications referenced in this paper. These charts depict the distribution of various estimation methods utilized across the papers, the methodologies addressed within them, and the timelines associated with these publications. 6.1 Estimation methods vs. Number of papers

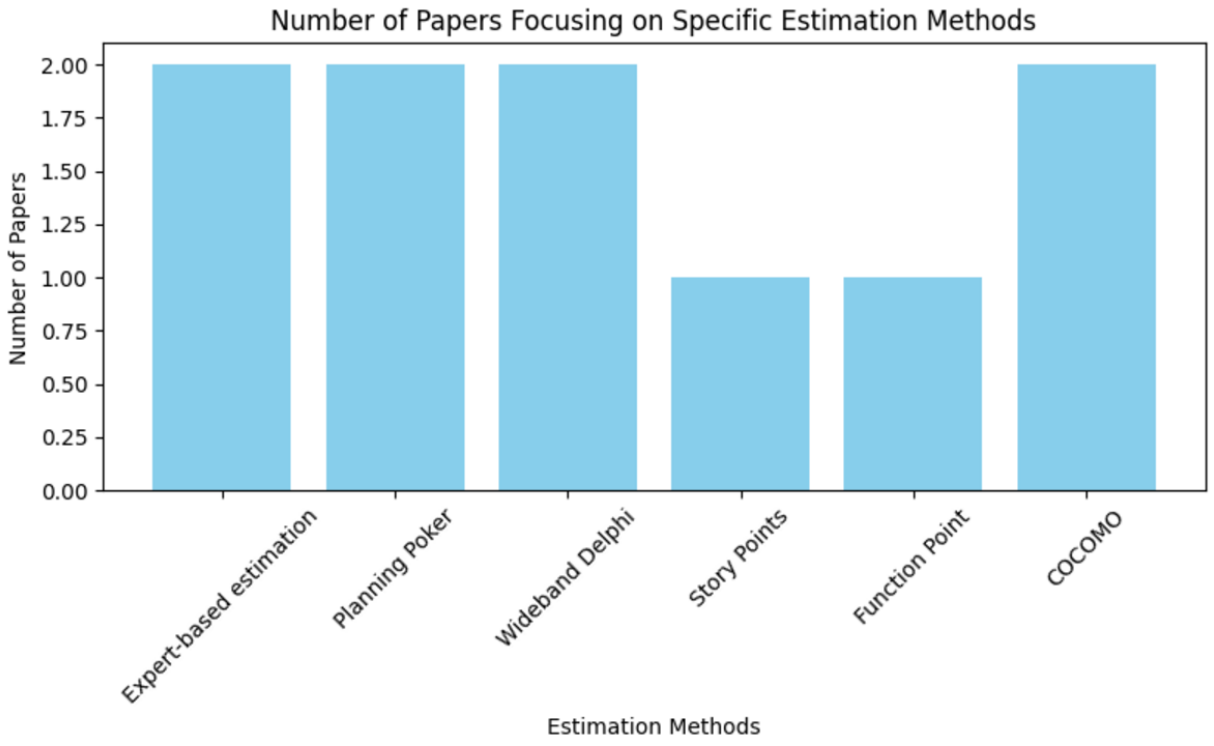

Figure.1 No of publications for the software estimation techniques [1]-[6]

The graph shows the distribution of research papers focusing on various software estimation methods. The methods include Expert-based estimation, Planning Poker, Wideband Delphi, Story Points, Function Point, and COCOMO. Expert-based estimation, Planning Poker, and COCOMO have the highest number of papers, each with approximately 2 papers focusing on them, indicating significant interest and study in these areas. Wideband Delphi and Story Points have moderate attention, with slightly fewer papers devoted to them. Function Point has the least number of papers, suggesting relatively lower focus in comparison to the other methods. This distribution highlights that while some methods like COCOMO and Planning Poker are widely studied, others like Function Point may require more exploration to understand their applicability and effectiveness in software estimation.

The below pie chart illustrates the proportion of various software estimation methodologies covered across research papers. The methodologies include COCOMO, Wideband Delphi, Planning Poker, Function Point, and Story Points. COCOMO, Wideband Delphi, and Planning Poker each account for 25% of the methodologies covered, indicating equal and significant representation in the research papers. Function Point and Story Points each make up 12.5%, reflecting comparatively less attention in the reviewed literature. This distribution suggests that methodologies like COCOMO, Wideband Delphi, and Planning Poker are more prominently studied, while Function Point and Story Points are less frequently explored in the context of software estimation research.

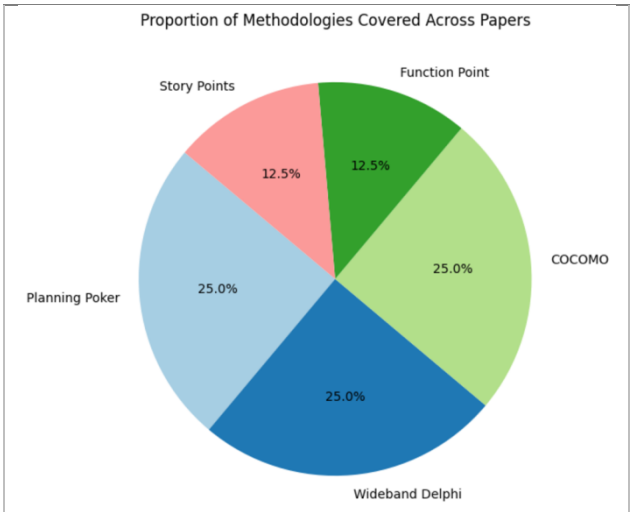

Figure.2

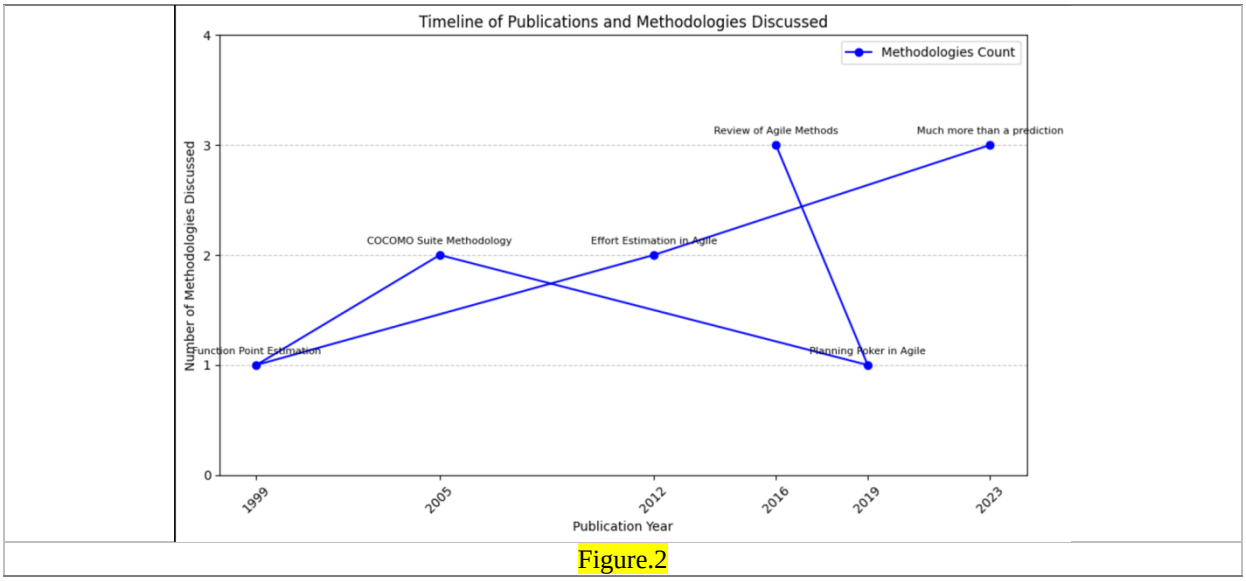

Figure.2

Some overall theory

5. CONCLUSION

Accurate project estimation is increasingly vital for the success of contemporary software development initiatives. As software systems grow in complexity and the demand for timely and cost-effective delivery escalates, the necessity for dependable estimation techniques has reached unprecedented levels. This paper examines six of the most prevalent estimation methods: COCOMO, Wideband Delphi, Planning Poker, Function Point, Story Points, and Expert-based estimation. Each method possesses distinct advantages and disadvantages, rendering them more appropriate for particular project types and requirements. For instance, algorithmic models such as COCOMO provide structured, quantitative estimates, whereas experience-based methods like Expert Judgment offer flexibility and adaptability, particularly for smaller projects. Collaborative approaches, including Wideband Delphi and Planning Poker, are especially effective in fostering team consensus and minimizing bias.

Although no single estimation method is universally applicable to all project scenarios, a thoughtful selection tailored to the specific project context, complexity, and team dynamics can greatly improve predictability, mitigate risks, and facilitate better resource management. The integration of traditional methodologies with emerging trends, such as AI-driven estimation tools and hybrid models, suggests that the future of project estimation will be increasingly efficient and precise. These innovations are expected to empower software teams to make well-informed decisions, adapt to changing project conditions, and achieve high-quality results that align with stakeholder expectations. Ultimately, proficient project estimation is crucial for successful software development, ensuring that projects are completed on schedule, within budget, and with optimal resource allocation.

6. FUTURE WORK

1. *Integration of AI and ML*

The integration of Artificial Intelligence (AI) and Machine Learning (ML) into software estimation practices represents a transformative step forward. AI/ML models have the ability to analyze historical project data, learn from past trends, and predict effort, cost, and resource requirements with greater precision. By leveraging techniques like natural language processing, deep learning, and predictive analytics, these models can automatically adjust to project-specific parameters, making the estimation process more adaptive. For example, AI systems can analyze patterns in user stories, code complexity, and development team performance to produce real-time and highly accurate effort estimations. Additionally, AI-based tools can provide early warnings of potential deviations or risks, enabling proactive intervention and better project management.

2. *Hybrid Estimation Techniques*

Hybrid estimation techniques combine the strengths of multiple methodologies to deliver a more comprehensive and accurate approach to project estimation. For instance, algorithmic methods like COCOMO can be paired with experience-based approaches such as Expert Judgment to provide both quantitative and qualitative insights. Similarly, collaborative techniques like Planning Poker can be integrated with data-driven models like Function Point Analysis to enhance accuracy in Agile environments. Hybrid approaches allow flexibility and adaptability, catering to the unique needs of projects that vary in scale, complexity, and domain. By combining methodologies, organizations can leverage the best of each approach to address the limitations of any single technique, resulting in more reliable and well-rounded estimates.

3. *Real-Time Metrics*

The use of real-time metrics in software estimation is a modern innovation that enables dynamic and continuous updates to project estimates throughout the development lifecycle. By incorporating live data such as team velocity, code churn, defect rates, and sprint progress, real-time estimation tools can adapt to evolving project conditions. This approach ensures that estimates remain accurate and relevant, even as requirements or timelines shift. Real-time metrics foster transparency and accountability within teams, as they provide immediate insights into the project's status and potential challenges. This dynamic updating process not only improves decision-making but also allows for more efficient resource reallocation and risk mitigation as projects progress.

4. *Industry-Specific Frameworks*

Designing customized estimation frameworks tailored to specific industries can address unique challenges and requirements that traditional methods might overlook. For example, in the healthcare sector, estimation techniques need to consider stringent regulatory requirements, data privacy concerns, and complex system integrations. In contrast, financial technology projects may require a focus on high-frequency data processing, compliance standards, and security protocols. Industry-specific

frameworks can incorporate these nuances, enabling more accurate and relevant estimations. By aligning estimation methods with industry-specific needs, organizations can ensure greater predictability, improved resource management, and adherence to sector-specific standards.

5. Scalable Agile Tools

In Agile methodologies, tools like Planning Poker have proven effective for team collaboration and relative sizing of user stories. However, there is a growing need to scale these tools to accommodate larger, more complex Agile projects. Enhancements to such tools can include incorporating advanced visualization, real-time voting across distributed teams, and AI-driven recommendations for story point assignments. Scalable Agile tools can also support broader applications, such as aligning multiple Scrum teams in a Scaled Agile Framework (SAFe) or integrating cross-functional inputs in enterprise-level Agile projects. By improving and expanding the capabilities of Agile estimation tools, teams can maintain the flexibility and adaptability of Agile practices while addressing the challenges of scaling in larger organizations.

This elaboration highlights how these advancements can revolutionize estimation practices, making them more accurate, efficient, and adaptable to modern development challenges.

7. REFERENCES

[1] Matsubara, Patrícia GF, et al. "Much more than a prediction: Expert-based software effort estimation as a behavioral act." Empirical Software Engineering 28.4 (2023): 98.Johnson, R., & Lee, M. (2023). Prediction-Based Cost Estimation Technique in Agile Development. International Journal of Agile Systems, 12(2), 85-102. doi:10.1108/IJAS.2023.0123.

[2] Coelho, Evita, and Anirban Basu. "Effort estimation in agile software development using story points." International Journal of Applied Information Systems (IJAIS) 3.7 (2012).Wilson, G., & Patel, S. (2023). Advancing Cost Estimation in IT Software Projects. Journal of IT Project Management, 18(4), 212-229. doi:10.1145/ITPM.2023.9876.

[3] Meli, Roberto, and Luca Santillo. "Function point estimation methods: A comparative overview." FESMA. Vol. 99. Citeseer, 1999.

[4] Boehm, Barry, et al. "COCOMO suite methodology and evolution." CrossTalk 18.4 (2005): 20-25.

[5] Gandomani, Taghi Javdani, Hamidreza Faraji, and Mahsa Radnejad. "Planning Poker in cost estimation in Agile methods: Averaging vs. Consensus." 2019 5th Conference on Knowledge Based Engineering and Innovation (KBEI). IEEE, 2019.

[6] Munialo, Samson Wanjala, and Geoffrey Muchiri Muketha. "A review ofagile software effort estimation methods." (2016).