Dialogue Social Science Review (DSSR)
www.thedssr.com

ISSN Online: 3007-3154
ISSN Print: 3007-3146

DIALOGUE SOCIAL SCIENCE REVIEW

Vol. 3 No. 6 (June) (2025)

A Hybrid Framework For Accurate Software Effort Estimation In Agile And Traditional Development Models

Jahanzaib Ahmed Khan
Assistant Manager, Meezan Bank, Email: jahanzaibkhan169@gmail.com

Mohammad Ayub Latif (Corresponding Author)
Assistant Professor, CoCIS, KIET, Email: malatif@kiet.edu.pk

Muhammad Khalid Khan
Professor and Dean CoCIS, KIET. Email: khalid.khan@kiet.edu.pk

Muhammad Dawood Akram
Senior Lecturer, Department of Computer Science, Bahria University Lahore Campus. Email: m.dawood@bahria.edu.pk

Rizwan Khalid
Senior Lecturer, Department of Computer Science, Bahria University Lahore Campus. Email: Rizwankhalid.bulc@bahria.edu.pk

Syed Mubashir Ali
Associate Professor, Department of Computer Science, Bahria University, Lahore campus, Pakistan, Email: syedmubashir.bulc@bahria.edu.pk

Abstract
The success of software development projects depends on accurate software effort estimation. Plan-driven frameworks frequently employ conventional models like COCOMO and Function Point Analysis (FPA), however these models frequently lack the adaptability needed to change with the needs of a project. Although agile estimation methods, such as T-shirt sizing and Story Points, offer flexibility, they may compromise early accuracy. The hybrid estimating framework proposed in this paper combines adaptive and deterministic techniques to increase accuracy while preserving adaptability. A major government project and a mobile application development endeavor serve as two case studies used to assess the framework. The findings show that when conventional and agile estimate techniques are combined, accuracy and flexibility are improved over when either strategy is used alone. Performance indicators including usability, accuracy, and adaptability are examined. Finally, guidelines for selecting appropriate estimation techniques based on project characteristics are presented.

Keywords: Software Effort Estimation, Agile, Traditional, Hybrid Framework, COCOMO, Story Points, Software Engineering

1. Introduction:

One of the most important elements of software engineering is software effort estimation, as it determines whether software projects will succeed or fail.

Dialogue Social Science Review (DSSR)
www.thedssr.com

ISSN Online: 3007-3154
ISSN Print: 3007-3146

DIALOGUE SOCIAL SCIENCE REVIEW

Vol. 3 No. 6 (June) (2025)

Establishing realistic expectations demands precise estimates, which provide adequate resource allocation and eventually the delivery of the software product within the scheduled time and budget. Inaccurate effort/cost estimation can result in delays, over expenditure, and even total project failure and is a vital area of concern for software project managers and development teams [1], [2] [3] , [4]. The methodology of estimating the effort of software development has changed drastically since years ago, with numerous techniques emerging to suit the purposes of different paradigms in software development.[5],[6]

The intricacy and intrinsic uncertainty of software development processes make effort estimation in software engineering challenging [7],[8]. As opposed to physical engineering, software development is an abstract and intangible process that usually includes high uncertainty, especially concerning effort, time, and resource needed[9]. This can complicate the estimation task since software projects tend to be susceptible to scope change, technology, and user needs change[10],[11]. Traditional methods of software development, such as the Waterfall model are usually complained about because they are not very adaptable in managing such changes, and thus can pose problems in estimating effort[12],[13]. More recent methodologies such as agile have, therefore, come in to overcome such shortcomings through emphasis on flexibility, iterative development, and constant stakeholder input, hence presenting an alternative way of estimating effort and costs [14], [15].

The objective of this paper is to identify and contrast the different effort/cost estimation methods applied in the classical and agile paradigms for development. Classical methods like COCOMO (Constructive Cost Model) and Function Point Analysis (FPA) model, stress initial planning, exhaustive requirements analysis, and utilization of past data to deliver organized estimates [16], [17], [18]. These models have been extensively applied in software engineering, especially in those industries with highly regulated environments, where thorough planning and rigid follow-up of project schedules and budgets are essential. Nonetheless, their dependency on well-defined project specifications and early-stage commitment tends to result in difficulties when presented with dynamic project requirements or changes not previously anticipated [21], [22].

Contrarily, agile methodologies such as Extreme Programming (XP), Scrum, and Kanban has emerged as more flexible means of software development [23], [24]. Agile methodologies are characterized by short, iterative development cycles (sprints), regular releases, and continuous communication with stakeholders and therefore are more flexible and adaptive to change [25]. This adaptability during development also finds its manifestation in a change in cost estimation methodologies. Within Agile projects, the application of measures such as Story Points and T-shirt sizing has become more common because they are simple, easy to execute, and consistent with Agile Principles like iterative delivery and continuous improvement[26],[27]. These practices are often less rigorous and more flexible than conventional methods, permitting development teams to rapidly update estimates according to up-to-date progress and changing project requirements [28].

While conventional effort estimation techniques such as FPA and COCOMO have been demonstrated to be effective in estimating effort and costs for large and clearly defined software projects, they tend not to perform well in agile settings

ISSN Online: 3007-3154
ISSN Print: 3007-3146

where the requirements and scope can change over the course of the project life cycle [29]. Conversely, agile estimation methods, although more adaptable, tend to lack the accuracy and reliance on historical data offered by conventional methods [30]. Consequently, the choice of the right cost estimation approach relies on various factors that involve the scope, size, complexity of the project, as well as the development methodology under use [31]. Hence, the knowledge of the advantages and disadvantages of conventional as well as agile estimation approaches is essential to arrive at intelligent decisions and obtain accurate predictions of cost [32].

This work seeks to give an in-depth analysis of such cost estimation methods, a comparison of their efficiency, simplicity, flexibility, and usability in different software project types. It analyzes the theory behind such estimation methods and then the practical use by industry case studies and the views of experts. Through examining both conventional and agile methods of software effort/cost estimation, it provides insightful analysis regarding how various estimation methods can be utilized in various project settings. The ultimate goal is to provide a set of recommendations for the most appropriate effort/cost estimation method with consideration for the specific needs and constraints of each project.

This research paper's structure is designed to provide readers a thorough grasp of software effort/cost estimates across various development frameworks. The relevant work and literature study are presented in Section 2, which also highlights current approaches and research pertinent to cost estimate in both traditional and agile systems. By highlighting knowledge gaps and providing background information for the topic, this part lays the groundwork for the remainder of the paper. Section 3 discusses the key contributions of this study. Section 4 examines traditional models, their advantages, and disadvantages as it relates to cost estimating techniques inside traditional software development frameworks. Effort estimate in agile development frameworks is examined in section 5, which also examines the impact of agile approaches on estimating precision and flexibility. Section 6 offers useful insights by discussing two different case studies where the HCEF framework can be utilized and finally section 7 concludes the paper by identifying the future directions.

2. Related Work:

Agile approaches, which offer iterative, adaptive, and customer-centered methodologies that contrast sharply with more traditional linear project management models, have become a standard component of modern software development. Agile's adaptability to shifting user needs and project scope is perhaps one of its fundamental principles. Techniques like continuous re-estimation and effective scope management, which have been the subject of extensive research and application, are largely responsible for this flexibility.

- **Continuous Re-estimation in Agile:** Estimating is a one-time task at the beginning of a project in traditional project management. Agile approaches such as Scrum and XP, on the other hand, place a strong emphasis on ongoing re-estimation in order to better track progress and adjust to evolving needs [2]. In order to align estimates with current reality, [4] argue for iterative sprint planning, pointing out the shortcomings of static estimation. This method guarantees that expectations align with stakeholder feedback, task difficulty, and available resources.

Dialogue Social Science Review (DSSR)
www.thedssr.com

ISSN Online: 3007-3154
ISSN Print: 3007-3146

DIALOGUE SOCIAL SCIENCE REVIEW

Vol. 3 No. 6 (June) (2025)

Our project's team was better able to adjust to changing priorities and user feedback by using iterative estimation. The usefulness of agile principles was demonstrated by better scheduling, resource allocation, and stakeholder alignment when effort was re-estimated at each sprint.

- **Managing Scope Flexibility:** One major benefit of agile is its scope management flexibility, which allows teams to adjust to shifting customer requirements without sacrificing budget or timeline. Agile guarantees constant attention to high-value features through the use of prioritized backlogs and iterative delivery. This flexibility, however, can result in scope creep if improperly managed, putting deadlines and profitability at risk. To reduce this risk, [3] stresses stringent prioritizing, regular stakeholder engagement, and a distinct product vision.

Agile's adaptability enabled the team to add new requirements to a project, but if changes were left unchecked, there was a chance that the original objectives would be lost. Scope flexibility can be turned into a strength rather than a drawback when properly handled.

2.1 Suggestions for Further Enhancement

As it enables project managers to effectively plan, schedule, and resource, software cost estimating is a critical activity in the software development process. Many studies have examined different cost estimation techniques, focusing on both traditional and agile approaches. Traditional estimating models like COCOMO and Function Point Analysis (FPA) are popular for their systematic, quantitative methodology to estimate the effort of software development [1]. For effort, cost, and schedule estimation, these models use historical data and pre-defined criteria. In contrast, agile estimating techniques like Story Points and T-shirt sizing are flexible and dynamic, especially in cases where the requirements keep evolving fast. [2]

While comparing Agile and conventional cost estimation methods, [3] point out that conventional methods offer greater accuracy in the case of stable and well-known requirements. But conventional methods do not perform well where the environment is dynamic and where the scope change is common [2] and it supports agile estimation techniques, highlighting their flexibility and ease in iterative development, when customer response and feature modification are usual.

2.2 Data and Metrics

Reliable data and appropriate metrics form the backbone of accurate software effort estimation, in both traditional and agile environments, the use of historical project data, performance indicators, and well-defined metrics is essential to improve estimation accuracy.

In traditional estimation methods such as COCOMO and Function Point Analysis (FPA), quantitative data plays a central role. These methods rely heavily on:

Historical project data (e.g., effort hours, size in KLOC or Function Points)

- Productivity rates
- Complexity factors
- Cost drivers such as team capability, tools, and required reliability

These inputs help produce detailed, formula-based estimates. Metrics such as effort per function point or person-hours per KLOC are commonly used to

Dialogue Social Science Review (DSSR)
www.thedssr.com

ISSN Online: 3007-3154
ISSN Print: 3007-3146

DIALOGUE SOCIAL SCIENCE REVIEW

Vol. 3 No. 6 (June) (2025)

benchmark performance and predict future project needs. In contrast, agile estimation techniques make use of more abstract and team-specific metrics, such as:

- **Story Points**: a relative measure of effort, complexity, and risk for each user story.
- **Velocity**: the average number of story points completed in a sprint, used to forecast future progress.
- **Burn-down and burn-up charts**: visual tools that track remaining work against time.

Agile teams may also use cycle time, lead time, and throughput as operational metrics to monitor and adjust their estimates dynamically throughout the project lifecycle.

One key challenge in both approaches is the availability and accuracy of the underlying data. While traditional methods often benefit from well-documented legacy project data, agile methods rely on consistent team performance and accurate backlog grooming to maintain velocity metrics. The subjective nature of some agile metrics, like story points, may introduce variability, especially across different teams or organizations.

Nevertheless, combining the strengths of both methods using historical data-driven insights from traditional models and the real-time adaptability of agile metrics can significantly enhance the reliability and responsiveness of software cost estimation practices.

2.3 Effect of Altering Requirements on Estimation

- **Derivative Analysis:** A direct comparison of the two approaches (Waterfall and Agile) may be beneficial, particularly emphasizing the variance in the estimation [4]. Contends that the changing scope in Agile projects can result in rework and scope creep, which makes it ever harder to estimate cost. Analogously, [1] explains how the inflexibility of conventional methods, such as COCOMO, cannot handle late-stage adjustments, which causes deadlines to be missed and exceeding budgets [5]. It is also known that although Agile's iterative approach supports changes in requirements, it can also create unrealistic cost forecasts since early estimates do not entirely reflect subsequent adjustments. This is especially concerning for projects such as the development of a mobile application, where feedback from users may require new features or modifications to existing ones, thereby making the prediction of effort difficult.
- **Utilization of Historical Data in Estimation Models:** Traditional models such as COCOMO depend largely on historical data to predict the amount of money and effort required in software development. [1]. [6] indicating that the application of traditional models is more challenging under conditions of unreliable historical data, especially for new projects or technology, for instance, historical data may not be available for estimation of costs for projects incorporating new technologies like block chain or AI, and hence, it is not possible for conventional approaches to give an accurate estimate. Agile estimation methods, however, depend less on historical data. They concentrate on relative estimates and scale using changing project data [2]. But if not monitored constantly, it can lead to less accurate predictions and wastage of resources.

Dialogue Social Science Review (DSSR)
www.thedssr.com

ISSN Online: 3007-3154
ISSN Print: 3007-3146

DIALOGUE SOCIAL SCIENCE REVIEW

Vol. 3 No. 6 (June) (2025)

- **Software Estimation Bias and Its Consequences:** Software estimating bias is yet another problem common to both traditional and agile approaches. Biases that stem from humans like optimism bias and overconfidence could influence the accuracy of cost estimates, as per [7]. For example, [1] explains how project managers tend to underestimate work complexity based on optimism bias in conventional models. In [2] points out the dangers of overconfidence bias in Agile methodologies, whereby team members exaggerate their capacity to deliver work within a certain sprint and therefore underestimate resources and delays. Errors in cost estimates because of bias can lead to higher costs for the project and delayed project completion, affecting both project success and stakeholder satisfaction.
- **Comparison of Cost Estimation Methods:** The following table is a summary of major differences between traditional agile estimation methods on different criteria: Table 1 shows the comparison of traditional and agile estimation techniques.

Criteria	**Traditional Estimation (e.g., COCOMO, FPA)**	**Agile Estimation (e.g., Story Points, T-shirt sizing)**
Accuracy	Higher accuracy in stable environments with well-defined requirements (Boehm, 1981) [1]	Lower accuracy, especially in early stages. Adaptability allows for evolving estimates (Cohn
Flexibility	Limited flexibility; changes in scope often require re-	Highly flexible; accommodates changes
Ease of Use	Requires detailed upfront planning and complex calculations [6].	Easier to implement with less upfront documentation [2].
Dependence on Historical Data	Strong reliance on historical data from similar projects [1].	Minimal reliance on historical data focusing on team experience and
Criteria	Traditional Estimation (e.g., COCOMO, FPA)	Agile Estimation (e.g., Story Points, T-shirt sizing)
Handling of Changing Requirements	Struggles to accommodate late changes, leading to cost overruns [5].	Built to handle changing requirements iteratively, though frequent scope changes can still lead to unpredictability
Bias Impact	Prone to optimism bias and complexity underestimation	Susceptible to overconfidence bias in

Table 1: Comparison of Traditional and Agile Estimation Techniques

- **Solving Estimation Problems:** Some solutions have been offered to mitigate the intrinsic problems of software cost estimation. To achieve a balance between flexibility and accuracy, [5],[25] recommends the adoption of a hybrid framework that integrates aspects of the conventional and agile approaches. Agile projects can, for example, employ coarse-grained classical models initially to obtain an estimate of the scope and cost of the project prior to converting to

ISSN Online: 3007-3154
ISSN Print: 3007-3146

more flexible agile methods once the project is underway and more information has become available.

3. Key Contribution of the Study

The following key findings were derived from an in-depth analysis of traditional, agile, and hybrid software effort estimation techniques and its salient features are highlighted in the sub-sections of this section:

3.1 Improved Accuracy through Hybrid Framework:

The proposed hybrid framework combines the early-stage precision of traditional estimation models with the adaptability of agile methods. This integration enhances overall accuracy in cost and effort prediction across varying project environments [1], [2], [5].

3.2 Context-Driven Estimation Selection:

The suitability of an estimation method is dependent on multiple factors such as project scope, complexity, requirement stability, and team dynamics. The hybrid model allows flexible adaptation based on these parameters [3], [4], [6].

3.3 Reduced Risk via Continuous Re-estimation:

Agile estimation techniques, such as story points and velocity-based forecasting, enable iterative updates that reflect ongoing changes in project scope and progress, thereby minimizing risks associated with fixed early estimates [2], [5].

3.4 Mitigation of Estimation Bias:

Both traditional and agile techniques are prone to human bias, optimism bias in traditional models [1] and overconfidence in agile methods [2][15]. A hybrid framework, through structured planning and continuous feedback, helps mitigate these biases [7],[20].

3.5 Strategic Use of Historical Data:

Traditional methods like COCOMO rely on extensive historical data for accurate cost modeling [1], while agile methods depend more on relative, team-based estimation [2]. A hybrid approach effectively utilizes historical data in the initial phases and agile metrics during development [6].

3.6 Empirical Validation through Case Studies:

The application of the hybrid estimation framework to two case studies, a government project and a mobile app development project, demonstrated superior performance in accuracy, usability, and adaptability compared to standalone approaches [4], [8],[13].

3.7 Guidelines for Technique Selection:

Based on comparative analysis and empirical evidence, the study provides practical guidelines for selecting estimation methods based on project characteristics such as development methodology, requirement volatility, and stakeholder involvement [5], [9].

4. Software Cost Estimation in Conventional Development Frameworks

4.1 Introduction of Waterfall Model

One of the earliest software development methodologies, the Waterfall model is a systematic, sequential process for project development. Every phase like collection of requirements, design, coding, testing, and maintenance has to be completed before proceeding to the next, so its design is very rigid. It is most suited to projects with well-defined requirements at the beginning that will not

Dialogue Social Science Review (DSSR)
www.thedssr.com

ISSN Online: 3007-3154
ISSN Print: 3007-3146

DIALOGUE SOCIAL SCIENCE REVIEW

Vol. 3 No. 6 (June) (2025)

change [1].

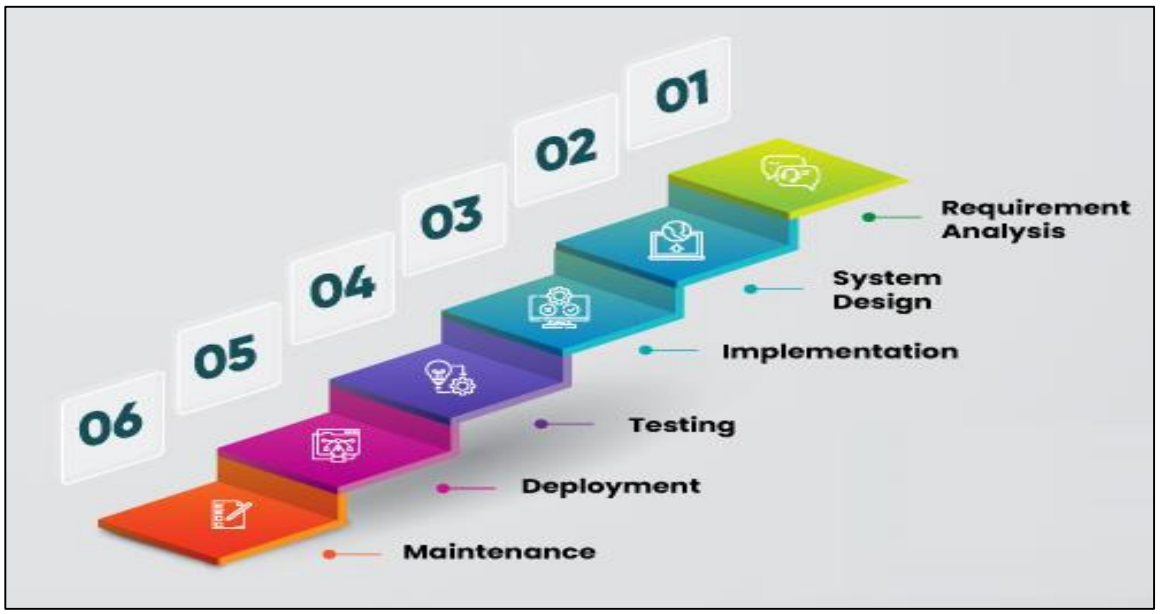

Figure 1. Waterfall Software Development Model

4.2 Suggested Hybrid Cost Estimation Framework (HCEF)

The aim of the suggested Hybrid Cost Estimation Framework (HCEF) is to merge the formal precision of conventional estimation methods with the elasticity and responsiveness of agile estimation methods. The framework captures the weakness of both methods through an added layer model that provides robust initial estimation and ongoing improvement over the project duration.

The HCEF functions in three sequential and iterative layers:

Layer 1: Initial Estimation Layer (Conventional Techniques)

Purpose: Create a planning phase baseline cost and schedule estimate.

Techniques Employed: Any techniques like Function Points, COCOMO etc. that gives the effort in persons-month

Output: A quantified structured estimate in person-months and initial resource allocation.

Layer 2: Adaptive Estimation Layer (Agile Techniques)

Purpose: Improve the cost estimate with the project execution based on agile metrics.

Techniques Used: Story Points: Allocated to user stories in sprint planning. Sprint velocity is applied for calculating capacity and estimation of completion time.

Output: Enhanced delivery forecast and effort estimate from real-time team performance.

Layer 3: Continuous Refinement Layer (Analytics and Feedback Integration)

Purpose: Continuously refine cost and schedule estimates using data analytics, performance trends, and stakeholder feedback to ensure alignment with project goals and evolving conditions.

Techniques Employed:

Dialogue Social Science Review (DSSR)
www.thedssr.com

ISSN Online: 3007-3154
ISSN Print: 3007-3146

DIALOGUE SOCIAL SCIENCE REVIEW

Vol. 3 No. 6 (June) (2025)

- **Earned Value Management (EVM):** To monitor project performance in terms of cost and schedule variances (CPI, SPI).
- **Trend Analysis:** Use historical sprint and release data to identify patterns in velocity, scope creep, or estimation accuracy.
- **Burndown/Burnup Charts:** Visual tools to track progress and predict remaining effort.
- **Feedback Loops:** Incorporate customer, stakeholder, and team feedback to adjust priorities and resource allocation.

Output:

- Continuously updated forecasts based on actual performance
- Early warnings for budget or schedule risks
- Data-driven decisions for mid-course corrections

The table 2 shows the HCEF functions in three sequential and iterative layers and figure 2 shows the HCEF framework. Table 3 shows the comparative advantages of the HCEF framework.

Layer	Purpose	Techniques	Output
Layer 1: Initial Estimation	Establish baseline using structured, historical data	FPA, COCOMO etc.	Initial estimate in person-months, resource plan
Layer 2: Adaptive Estimation	Improve estimates with execution metrics	Story Points, Sprint Velocity	Iterative effort and delivery forecast
Layer3:Continuous Refinement	Real-time adjustment using analytics and stakeholder input	EVM, Trend Analysis, Charts, Feedback Loops	Updated forecasts, risk alerts, informed adjustment actins

Table 2: HCEF functions in three sequential and iterative layers

Figure 2: The hybrid framework

Feature	Traditional Methods	Agile Methods	HCEF (Proposed)

Dialogue Social Science Review (DSSR)
www.thedssr.com

ISSN Online: 3007-3154
ISSN Print: 3007-3146

DIALOGUE SOCIAL SCIENCE REVIEW

Vol. 3 No. 6 (June) (2025)

Accuracy in Initial Estimation	High – Uses models like FPA and COCOMO II for early structured estimation.	Low – Estimations are relative and less precise at project start.	High – Combines traditional baseline (Layer 1) with Agile refinement (Layer 2).
Adaptability to Changes	Low – Rigid structure makes it hard to accommodate evolving scope.	High – Agile supports iterative requirement changes.	High – Agile layer enables flexible updates during execution.
Sprint-Level Tracking	No – Focuses on overall milestones rather than iterative progress.	Yes – Uses sprint metrics like story points and velocity.	Yes – Sprint data in Layer 2 improves tracking and forecast.
Historical Data Use	Required – Needed to calibrate models like COCOMO.	Not Needed – Estimates are based on current team dynamics.	Optional – Layer 1 benefits from historical data, Layer 2 adapts in real time.
Suitable for Dynamic Projects	Poor Fit – Not flexible for frequently changing requirements.	Excellent – Built for adaptive, evolving scope.	Excellent – Supports changing scope with initial structure and Agile updates.
Suitable for Fixed-Scope Projects	Best Fit – Structured estimates align with defined scope.	Weak Fit – Agile may introduce overhead in fixed environments.	Balanced Fit – Provides initial structure with flexibility for change if needed.

Table 3: Comparative advantages of the HCEF

4.3 Mathematical Representation of HCEF:

The **Hybrid Cost Estimation Framework (HCEF)** integrates traditional estimation, agile execution data, and real-time feedback to continuously refine project effort forecasts.

Main Estimation Formula

Let:

- Et = Total estimated effort at time t (in person-months)
- Eo = Initial estimate (Layer 1)
- At = Agile-based adjustment (Layer 2)
- Ft = Real-time performance adjustment (Layer 3)

Then the overall effort estimation becomes:

Et = Eo + At + Ft (Equation 1)

Layer 1: Initial Estimation (Eo)

This layer establishes the baseline estimate using any traditional method. The value of Eo is in person-months.

Accepted Estimation Methods:

- **COCOMO 81**: $Eo = a \times (KLOC)^b$
- **Function Point Analysis**: $Eo = FP \times Productivity\ Rate$
- **Expert judgment or historical analogy**

The estimator may choose any of these methods based on project type, team maturity, or organizational standards.

Layer 2: Agile-Based Adjustment (At)

Dialogue Social Science Review (DSSR)
www.thedssr.com

ISSN Online: 3007-3154
ISSN Print: 3007-3146

DIALOGUE SOCIAL SCIENCE REVIEW

Vol. 3 No. 6 (June) (2025)

This layer adjusts the original estimate based on agile metrics such as story points and team velocity.

Formula:
At = ((S / V) × D × T) / W − Eo (Equation 2)
Where:

- *S* = Total story points
- *V* = Team velocity (story points per sprint)
- *D* = Duration of each sprint (in weeks)
- *T* = Team size (number of full-time members)
- *W* = Average work weeks per month (typically 4.33)
- *Eo* = Initial estimate in person-months

Interpretation:

- If **At > 0**, Agile projections indicate **more effort** than initially planned.
- If **At < 0**, Agile progress suggests **less effort** is needed than the baseline.
- If **At = 0**, Agile delivery is aligned with the initial estimate.

Example 1: Negative Adjustment (At < 0)
Inputs:

- S = 150, V = 10, D = 2, T = 3, W = 4.33, Eo = 25

Steps:

- Number of sprints = S / V = 15
- Weeks = 15 × 2 = 30
- Person-weeks = 30 × 3 = 90
- Person-months = 90 / 4.33 ≈ 20.78
- At = 20.78 − 25 = **−4.22 person-months**

The Agile projection requires **4.22 fewer person-months** than initially estimated.

Example 2: Positive Adjustment (At > 0)
Inputs:

- S = 200, V = 10, D = 2, T = 4, W = 4.33, Eo = 27

Steps:

- Number of sprints = 20
- Weeks = 20 × 2 = 40
- Person-weeks = 40 × 4 = 160
- Person-months = 160 / 4.33 ≈ 36.96
- At = 36.96 − 27 = **+9.96 person-months**

Agile progress indicates **higher effort** than the initial estimate.
Layer 3: Continuous Refinement Using EVM (Ft)
This layer makes real-time corrections based on **Earned Value Management (EVM)** indicators.
Formula:
Ft = α × (1 − CPI) + β × (1 − SPI) (Equation 3)
Where:

- *CPI* = Cost Performance Index = EV / AC
- *SPI* = Schedule Performance Index = EV / PV
- *α* = Weight assigned to cost impact
- *β* = Weight assigned to schedule impact

Interpretation:

Dialogue Social Science Review (DSSR)
www.thedssr.com

ISSN Online: 3007-3154
ISSN Print: 3007-3146

DIALOGUE SOCIAL SCIENCE REVIEW

Vol. 3 No. 6 (June) (2025)

- If CPI and SPI are below 1.0, the project is over budget or behind schedule.
- Subtracting from 1 reveals the **degree of inefficiency**.
- Multiplying by α and β converts that inefficiency into **person-month impact**.

Example of Ft Calculation

Given:

- CPI = 0.90
- SPI = 0.85
- α = 5
- β = 5

Calculation:

- Ft = 5 × (1 – 0.90) + 5 × (1 – 0.85)
- Ft = 5 × 0.10 + 5 × 0.15 = 0.5 + 0.75 = **1.25 person-months**

Ft reflects the **effort increase** needed due to cost/schedule slippage.

Final Estimation Example

Let:

- E_0 = 27
- At = –4.22
- Ft = 1.25

$E_t = E_0 + A_t + F_t = 27 – 4.22 + 1.25 = 24.03$ person-months

The layer summary of each layer is shown in table 4.

Layer	Component	Input	Output	Interpretation
Layer 1	E_0	LOC / FP / Expert	Initial Estimate	Starting baseline
Layer 2	At	Agile metrics	Adjustment	Reflects Agile speed vs. plan
Layer 3	Ft	CPI, SPI	Adjustment	Reflects real-world project conditions

Table 4: Layer summary of each layer of HCEF

5. Effort Estimation in agile development frameworks:

The HCEF equation provides a **dynamic estimate** that improves over time:

$E_t = E_0 + A_t + F_t$ (Equation 1)

It blends traditional accuracy, agile responsiveness, and real-time analytics into one evolving formula.

Software Cost Estimation in Agile Development Frameworks: Agile methodologies are a move away from inflexible, sequential development processes towards more adaptive, iterative cycles. Agile prioritizes responding to change, teamwork, and customer input along the way throughout the development process. Methodologies such as Scrum, Kanban, and Extreme Programming (XP) stress short development cycles (sprints) where features are incrementally delivered, enabling stakeholders to examine and modify requirements in real-time [1]. Because of Agile focus on constant iteration and change, traditional cost estimation methods like COCOMO and FPA may not be suitable. Instead, agile projects require more flexible, less formal estimation techniques that can evolve with the project's needs [2].

Dialogue Social Science Review (DSSR)
www.thedssr.com

ISSN Online: 3007-3154
ISSN Print: 3007-3146

DIALOGUE SOCIAL SCIENCE REVIEW

Vol. 3 No. 6 (June) (2025)

5.1 Story Points

In Agile methodologies, Story Points serve as a unit of relative estimation for assessing the complexity, risk, and effort associated with implementing a user story. Unlike time-based estimation, story points abstract away from hours and instead represent the multidimensional workload inherent in software tasks [4].

5.1.1 Estimation Process

Relative Sizing Using Fibonacci Scale: Story Points are typically assigned using a non-linear Fibonacci-based scale (e.g., 1, 2, 3, 5, 8), which captures increasing uncertainty and nonlinear growth in task complexity [5].

Collaborative Estimation via Planning Poker: During sprint planning, estimation is performed through a consensus-driven approach such as Planning Poker, where each team member proposes a point value, followed by rational discussion and convergence toward a final estimate [6]. This approach mitigates anchoring bias and promotes estimation consistency.

Sprint Velocity Calculation: Teams track their velocity, defined as the aggregate story points completed per sprint, to construct an empirical performance baseline. This velocity metric facilitates capacity planning and supports probabilistic forecasting of project timelines [7].

5.1.2 Challenges

While Story Points are highly effective for tracking progress and aligning expectations, they do not provide precise estimates in terms of cost or duration. The system's reliance on team experience also means that estimates may vary widely depending on the team's familiarity with the project domain [8].

5.2 T-shirt Sizing

Another straightforward estimation method frequently employed in agile projects is t-shirt sizing. To show the relative complexity, tasks are divided into sizes like XS, S, M, L, or XL. And effort involved. This method is often employed in the early stages of the project when detailed requirements are still being clarified [9]. Advantages are as stated under:

• Quick to Implement: T-shirt sizing is an efficient way to estimate work early in the project lifecycle.

• User-Friendly: Teams find this approach simple and intuitive, especially for new agile teams.

5.2.1 Challenges

T-shirt sizing can be a decent place to start, but it's not as accurate as other techniques, which can lead to inaccurate cost estimates for larger Projects or projects with complex requirements [10].

This table 5 includes the T-shirt size code, T-shirt size, and the estimated duration range in hours.

T-Shirt Size Code	T-Shirt Size	Estimated Duration Range (Hours)
XXS	Extra Extra small	0 to 15
XS	Extra small	1.5 to 4
S	Small	4 to 10
M	Medium	10 to 20
L	Large	20 to 36
XL	Extra Large	36 to 50

Dialogue Social Science Review (DSSR)
www.thedssr.com

ISSN Online: 3007-3154
ISSN Print: 3007-3146

DIALOGUE SOCIAL SCIENCE REVIEW

Vol. 3 No. 6 (June) (2025)

Table 5: T-Shirt Size Mapping Table (With Duration Range)

This table 6 lists the T-shirt size along with its corresponding size code and an approximate duration in hours as a variation to table 5.

T-Shirt Size	T-Shirt Size Code	Approx. Duration Hours
Extra Extra small	XXS	1
Extra small	XS	3
Small	S	8
Medium	M	15
Large	L	28
Extra Large	XL	45

Table 6: T-Shirt Size Mapping Table (Without Duration Range)

6. Case Studies

6.1 Government Major Project (Waterfall Model)

6.1.1 Context and Background

Government projects, especially in industries like defense, healthcare, and infrastructure, tend to adopt highly disciplined approaches such as the Waterfall model. This is due to the necessity of high documentation levels, compliance with regulatory requirements, and ensured progress in long-duration, high-risk projects. The linear, sequential nature of the Waterfall approach is seen as being best suited to guaranteeing that all development phases which includes the gathering requirements, design, implementation, verification, and maintenance are well-planned and carried out.

The project in question entails the design of a healthcare information system funded by the government. It was a large-scale project that was supposed to automate patient data management across various modules like patient registration, electronic health records (EHRs), medical billing, and healthcare reporting. Being mission-critical in nature, the system was put under various audits and stringent compliance tests, making the choice of the Waterfall methodology even stronger.

6.1.2 Cost Estimation Process:

To approximate the project's cost, effort, and time, the development team employed two widely recognized industry estimation methods:

Function Point Analysis (FPA): The system was decomposed into separate components in terms of user interactions inputs, outputs, data files, and interfaces. Each component was given a complexity rating (low, medium, high), and the function points were counted accordingly. The function points assisted in quantifying the functionality of the system in a technology-independent manner, giving an early estimate of the size of the system in function points and then the effort was calculated in persons-month by using the historical data.

6.1.3 Challenges:

Rigidity of the Waterfall Model: Halfway through development, newer healthcare regulations necessitated major redesigns of the system's architecture. But as the architecture and design phases had already been accomplished under Waterfall's linear framework, such changes were hard to make without going back on previous phases. This inflexibility resulted in project delays, escalated costs, and a domino effect of needed revisions around the system.

Dialogue Social Science Review (DSSR)
www.thedssr.com

ISSN Online: 3007-3154
ISSN Print: 3007-3146

DIALOGUE SOCIAL SCIENCE REVIEW

Vol. 3 No. 6 (June) (2025)

Incorrect Early Estimates: Early estimates, premised on initial assumptions regarding complexity, did not account for the changing regulatory environment or unexpected technical issues. Consequently, resource and time requirements were actually much greater than originally estimated, leading to the need for rework and extra personnel.

6.1.4 Lessons Learned:

Project Management Flexibility: The project highlighted the requirement for more flexibility in the estimation and planning phases. While the Waterfall model can be applied to projects with stable requirements, the changing landscape of healthcare rules exposed its shortcomings. Implementation of agile aspects, including iterative feedback loops, would have helped enhance flexibility.

Continuous Re-estimation: The project emphasized the need for continuous estimation throughout the life cycle. Constantly revisiting and adjusting estimates based on new information or risks would have allowed for improved resource allocation, risk management, and stakeholder communication.

6.2 Mobile App Development (Agile Methodology):

6.2.1 Context and Background:

Within high-speed and fast-changing sectors like mobile app development, organizations favor agile methodologies more and more, especially scrum. Agile adaptive, inclusive, and elastic features suit it particularly well for projects involving changing requirements and constant updates. The mobile app landscape, with the changing tastes of users and ongoing technological progression, requires development methodologies that are capable of quickly adjusting and changing.

6.2.2 Project Overview:

This case study highlights a fitness mobile app created by a startup whose mission is to enable users to define goals, monitor fitness progress, and obtain personalized workout and nutrition programs. The app further incorporated social functionality through which users could share success and connect. Because of market competition, the startup focused on fast delivery, user-led development, and frequent updates.

First, the team employed T-shirt sizing to make an estimate of the relative complexity of the features. The tasks were allocated as XS, S, M, L, or XL, allowing for rapid prioritization without the need for detailed specifications which is perfect at this stage when requirements were still changing.

As development went on and features were more clearly defined, these sizes were evolved into finer-grained story points (with Fibonacci values like 1, 2, 3, 5, etc.) to enhance progress tracking and sprint planning.

6.2.3 Challenges:

Uncertain Requirements:

When the project started, the client's vision of the app's features was not very clear. This made the initial estimates unclear and the feature set keep changing. The necessity for iterative, user feedback-based development brought in fresh requirements during the project, making it difficult to predict efforts and costs.

Iterative Development and Estimation: Agile iterative approach to development meant that cost and effort projections were revisited in every sprint. While this kept the team adaptable to emerging insights and user requirements,

Dialogue Social Science Review (DSSR)
www.thedssr.com

ISSN Online: 3007-3154
ISSN Print: 3007-3146

DIALOGUE SOCIAL SCIENCE REVIEW

Vol. 3 No. 6 (June) (2025)

it created difficulties in long-term forecasting, particularly where new features were introduced halfway through development.

Stakeholder Impact: Agile focused on ongoing interaction with stakeholders, such as product owners and early adopters. This helped ensure that development stayed in sync with market expectations. But it also meant that feedback loops might initiate scope changes, exerting pressure on estimation accuracy and team capacity. In contrast to government projects, where regulatory agencies control stakeholder input, the stakeholders here were end-users and product managers, focused on user experience and feature innovation.

6.2.4 Lessons Learned:

- **Accept Uncertainty in Estimation:** Initial estimates in agile contexts hardly ever remain so. The team came to accept them as tentative and created estimation habits that could adapt. By employing relative estimation methods such as story points and T-shirt sizing, the team retained visibility of progress while permitting change.
- **Value of Continuous Feedback:** Continuous stakeholder participation and incremental releases improved alignment with user expectations. Over time, estimation got better as the team became clearer about the product domain and the needs of the users, showing the capability of agile in dealing with changing requirements.
- **Desire for Estimation Discipline in Agile:** Although agile encourages flexibility, the team understood that disciplined estimation ceremonies, including sprint planning and retrospectives, are essential to prevent scope creep and keep delivery schedules on track. Finding a balance between agility and estimation discipline was the key to the success of the project.

6.3 Summary of Model Application in Case Studies:

In both case studies, the components of the proposed Hybrid Cost Estimation Framework (HCEF) could have been applied to reflect the contrasting demands of traditional and agile environments. The government project employed Function Point Analysis to deliver early-stage, size-based cost estimation suited to a plan-driven approach. Conversely, the mobile app project adopted Agile-compatible techniques T-shirt sizing and story points to support iterative planning and dynamic re-estimation. These applications demonstrate the need for a hybridized approach capable of adapting to evolving requirements while maintaining estimation accuracy. The observed limitations in both cases validate the relevance of the HCEF framework.

7. Conclusion and Future Work

7.1 Conclusion

Correct software cost estimation remains one of the most critical and difficult aspects of software project management. This paper presents a survey of the estimation techniques in use in both traditional and agile development environments with specific consideration for their relative strengths and weaknesses, including COCOMO, Function Point Analysis, T-shirt sizing, and story point estimation. This paper also introduces and assesses a new Hybrid Cost Estimation Framework (HCEF) that strategically combines deterministic models of estimation with the practices of adaptive approaches in agile estimation. Case studies in government and mobile application projects depict real-world challenges under both methodologies and underline the requirement

ISSN Online: 3007-3154
ISSN Print: 3007-3146

of an agile yet disciplined estimation approach. The work has found major insights in terms of continuous re-estimation need, stakeholder engagement and scope volatility management. The hybrid framework connects predictability of conventional models with the adaptability of agile models to offer a more elastic and realistic approach toward modern software projects.

7.2 Future Work

Many opportunities for future improvements are seen, though the proposed HCEF already offers a strong foundation for estimating software effort and cost. One major avenue would be the deployment and validation of a software product that incorporates HCEF, dynamic user interfaces, AI-driven estimate creation, and interaction with project management platforms like Microsoft Project or JIRA. This would make it possible to easily validate and test, on a wide scale and in as many application settings as possible, the framework to handle the specific cost drivers and estimating challenges of industries such as embedded systems, healthcare, fintech, and edtech, among others, but domain-specific calibration as well. Another possibility of great interest is the enhancement of the AI capabilities of HCEF, perhaps by using state-of-the-art techniques such as deep learning, reinforcement learning, or hybrid neuro-fuzzy systems for improving prediction performance, particularly in complex and data-rich environments. This would raise its credibility and dependability to a greater level, working with software companies and looking at past project datasets to benchmark the framework with industry data. The estimating model can learn to minimize discrepancies between expected and actual results by adding real-time feedback loops from change logs or sprint reviews. Future improvements in HCEF will move it closer to being an intelligent, flexible industry-relevant estimation tool.

References

[1] I. Somerville, *Software Engineering*, 10th ed. Pearson, 2015.
[2] M. Cohn, *Agile Estimating and Planning*. Prentice Hall, 2005.
[3] D. Leffingwell, *Scaling Software Agility: Best Practices for Large Enterprises*. Addison-Wesley, 2007.
[4] M. Jorgensen and M. Shepperd, "A systematic review of software development cost estimation studies," *IEEE Trans. Softw. Eng.*, vol. 33, no. 1, pp. 33–53, Jan. 2007.
[5] S. Park, C. Kim, and E. Lee, "Hybrid cost estimation framework using COCOMO and Agile," *Int. J. Comput. Sci. Inf. Secure.*, vol. 12, no. 9, pp. 1–7, 2014.
[6] V. T. Nguyen et al., "An improved estimation model for Agile projects using AI and historical data," *J. Syst. Softw.*, vol. 180, p. 110980, 2021.
[7] R. Madachy, *Software Process Dynamics*. Wiley-IEEE Press, 2007.
[8] A. Trendowicz and R. Jeffery, *Software Project Effort Estimation: Foundations and Best Practice Guidelines*. Springer, 2014.
[9] S. McConnell, *Software Estimation: Demystifying the Black Art*. Microsoft Press, 2006.
[10] A. Abran et al., *Guide to the Software Engineering Body of Knowledge (SWEBOK)*. IEEE Computer Society, 2014.
[11] L. H. Putnam and W. Myers, *Five Core Metrics: The Intelligence Behind Successful Software Management*. Dorset House, 2003.

ISSN Online: 3007-3154
ISSN Print: 3007-3146

Vol. 3 No. 6 (June) (2025)

[12] M. Sharma, A. Rana, and R. Singh, "Cost estimation models in traditional and Agile environments: A review," *Int. J. Comput. Sci. Inf. Technol.*, vol. 5, no. 3, pp. 4380–4384, 2014.
[13] M. Rizwan et al., "AI-based estimation models in Agile frameworks," *Int. J. Adv. Comput. Sci. Appl.*, vol. 12, no. 6, 2021.
[14] A. Srivastava, "Use of artificial neural networks in software estimation," *Int. J. Comput. Appl.*, vol. 124, no. 3, pp. 5–9, 2015.
[15] P. Padua, "Cost estimation in Agile using fuzzy logic," in *Proc. Int. Conf. Comput. Intell. Comput. Res. (ICCIC)*, 2017.
[16] D. R. Jeffries, "Extreme programming and Agile software development methodologies," *IEEE Softw.*, vol. 19, no. 6, pp. 32–37, 2002.
[17] N. Ramesh and M. Jarke, "Toward reference models for requirements traceability," *IEEE Trans. Softw. Eng.*, vol. 27, no. 1, pp. 58–93, 2001.
[18] P. Berander and P. Jönsson, "A goal question metric based approach for efficient measurement framework definition," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 13, no. 5, pp. 531–547, 2003.
[19] F. Zhang et al., "An adaptive effort estimation approach using fuzzy analytic hierarchy process," *Expert Syst. Appl.*, vol. 36, no. 3, pp. 6829–6837, 2009.
[20] T. Mens, "Software evolution," *IEEE Comput. Soc.*, vol. 39, no. 1, pp. 20–22, 2006.
[21] B. Kitchenham et al., "Systematic literature reviews in software engineering – A systematic literature review," *Inf. Softw. Technol.*, vol. 51, no. 1, pp. 7–15, 2009.
[22] H. Munir, "Comparative study of Agile and traditional project management," *Int. J. Mod. Educ. Comput. Sci.*, vol. 8, no. 11, pp. 1–9, 2016.
[23] M. Ali Babar, "Software architecture knowledge management: Theory and practice," *J. Syst. Softw.*, vol. 82, no. 8, pp. 1239–1240, 2009.
[24] R. S. Pressman, *Software Engineering: A Practitioner's Approach*, 7th ed. McGraw-Hill, 2009.
[25] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*. Addison-Wesley, 2012.
[26] S. Misra and A. Mondal, "Identification of software development risk factors," *J. Inf. Technol. Manag.*, vol. 19, no. 1, pp. 19–26, 2008.
[27] K. Beck et al., "Manifesto for Agile Software Development," 2001. [Online]. Available: https://agilemanifesto.org
[28] International Function Point Users Group (IFPUG), *Function Point Counting Practices Manual*, 2020.
[29] D. Pham and A. Ghanbarzadeh, "Multi-objective optimization using the Bees Algorithm," *Int. J. Intell. Comput. Cybern.*, vol. 1, no. 2, pp. 1–18, 2007.
[30] R. Suryanarayana, G. Samarthyam, and T. Sharma, *Refactoring for Software Design Smells*. Morgan Kaufmann, 2014.
[31] A. Mishra and D. Dubey, "Agile estimation techniques: A comparative study," *Int. J. Comput. Sci. Eng.*, vol. 6, no. 1, 2018.
[32] S. H. Kan, *Metrics and Models in Software Quality Engineering*. Addison-Wesley, 2002.