LaMoCEM: A Prompt-Driven Cost Evaluation Framework using Large Language Models in Agile Development

Muammad SAim, Muhammad Umer, Ali Hussain, Abdullah Akram

[*] Department of Artificial Intelligence, UMT Lahore

Abstract- Agile methodologies have improved flexibility and developer-client collaboration, but cost estimation remains a major challenge due to unpredictable scope changes. Traditional models struggle to adapt to rapid project dynamics, leading to inaccurate forecasts. We propose **LaMoCEM** (Large Model Cost Evaluation Model), a novel LLM-based approach that evaluates project cost by dynamically selecting the best estimation strategy based on project context. The system presents three cost estimation strategies to an LLM, which selects the most suitable one and returns a tailored Lambda function to compute cost from input data. This model integrates prompt engineering, dynamic tool usage, and real-time code execution, enabling adaptive cost forecasting in Agile environments. Our implementation demonstrates enhanced precision and flexibility across various project scenarios.

Keywords: Cost Estimation, LLM, Prompt Engineering, Lambda Functions, Agile Forecasting, Dynamic Code Generation

I. Introduction

Cost estimation in Agile is often inaccurate due to shifting requirements and developer variability. Traditional techniques like COCOMO and Function Point Analysis fail to adapt in real-time. This research introduces **LaMoCEM**, a modern estimation framework that uses OpenAI's GPT-4o to intelligently select cost models and generate executable estimation logic. By embedding reasoning, decision-making, and code-generation inside LLMs, LaMoCEM offers flexibility and scalability for evolving project demands.

II. SIGNIFICANCE OF STUDY

Agile projects demand dynamic cost estimation mechanisms that adapt to team size, task complexity, and technical requirements. LaMoCEM addresses this by:

- Evaluating project context through structured prompts
- Presenting 3 predefined cost models
- Letting the LLM select the appropriate one
- Generating a Python Lambda function for direct evaluation

This makes cost forecasting both adaptive and developer-agnostic, particularly useful in AI-integrated projects or fast-paced startups.

Agile development involves frequent scope changes and varying task complexities, which traditional cost estimation models cannot adapt to effectively. These models often fail to capture the real-time dynamics of team structure, evolving requirements, and technical challenges. As a result, project managers face difficulties in accurate forecasting and budgeting.

LaMoCEM introduces a novel solution by leveraging structured prompts and LLMs to evaluate the project scenario. It intelligently selects from three cost estimation strategies—hourly-based, LOC-based, and hybrid—based on input parameters like team size, salary, task hours, and code volume. This approach ensures that the cost model aligns closely with the specific context of each project.

By generating executable Lambda functions on-the-fly, LaMoCEM enables immediate and precise cost evaluation. This makes it ideal for projects in AI, automation, and startups, where adaptability and speed are crucial. Its flexibility and decision-driven logic offer a significant advancement over rigid, traditional estimation methods.

III. RELATED WORKS

Cost estimation in agile development has been a significant challenge due to frequent changes in project scope. Various estimation techniques have been developed to address this issue, but none provide a perfect solution.

3.1 COCOMO

The Constructive Cost Model (COCOMO) is one of the earliest and most widely recognized software cost estimation models. It calculates development effort primarily based on Lines of Code (LOC) and project complexity. While it provides a structured formula to estimate person-months and cost, it assumes a linear development process and static requirements. In Agile environments where scope and team dynamics change frequently, COCOMO struggles due to its lack of contextual awareness. It cannot adapt in real time or select an appropriate estimation strategy based on varying project scenarios, making it less suitable for dynamic workflows.

3.2 Function Point Analysis (FPA)

Function Point Analysis estimates software effort based on system functionality rather than code volume. It evaluates inputs, outputs, internal files, and interfaces, assigning complexity weights to calculate a total function point score. While useful in structured, document-heavy environments like the Waterfall model, FPA requires predefined requirements and does not handle evolving user stories well. Its rigid nature and lack of support for adaptive estimation make it difficult to implement effectively in Agile cycles. Moreover, FPA offers no mechanism for integrating intelligent decision-making like model selection or prompt-based evaluation.

3.3 HuBiCEM

The HuBiCEM model,focused on individualized developer performance tracking through historical sprint data. It used Polynomial Regression to analyze trends and generate cost predictions based on per-developer performance in different task categories. While HuBiCEM significantly improved cost estimation accuracy compared to traditional models, it relied entirely on past data and internal logic. It lacked external input evaluation, model adaptability, or the ability to respond to unique project characteristics in real time. There was no mechanism for automatic model selection or prompt-driven logic generation—gaps that LaMoCEM now addresses using LLMs and Lambda functions.

IV. PROPOSED MODEL

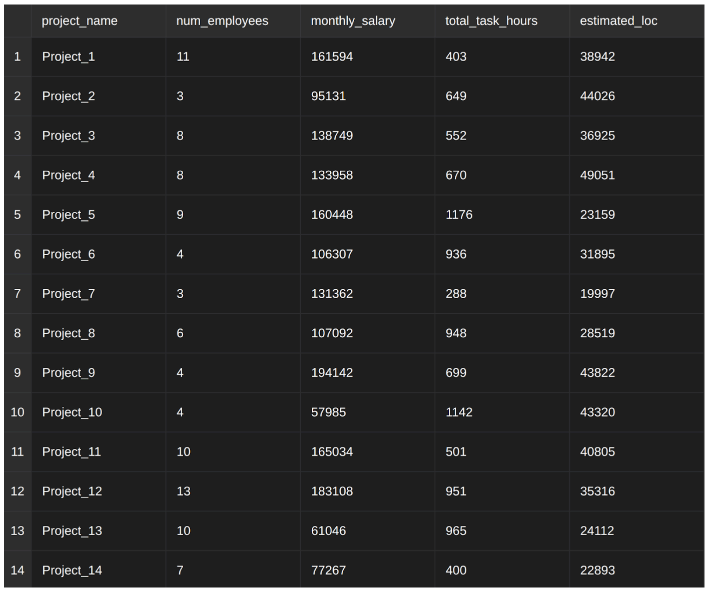
We present **LaMoCEM**, a prompt-based cost evaluation framework where:

1. **Three cost strategies** are defined (e.g., LOC-based, hourly-based, and hybrid)
2. The **LLM evaluates the input scenario**
3. It chooses the best-fitting strategy
4. It **generates a Lambda function**
5. The Lambda is compiled and executed on project data

4.1 LaMoCEM Process

Step 1: Scenario Input
Project data is gathered in a CSV including fields:

- Project Name
- Number of Employees
- Monthly Salary
- Estimated LOC
- Total Task Hours



	project_name	num_employees	monthly_salary	total_task_hours	estimated_loc
1	Project_1	11	161594	403	38942
2	Project_2	3	95131	649	44026
3	Project_3	8	138749	552	36925
4	Project_4	8	133958	670	49051
5	Project_5	9	160448	1176	23159
6	Project_6	4	106307	936	31895
7	Project_7	3	131362	288	19997
8	Project_8	6	107092	948	28519
9	Project_9	4	194142	699	43822
10	Project_10	4	57985	1142	43320
11	Project_11	10	165034	501	40805
12	Project_12	13	183108	951	35316
13	Project_13	10	61046	965	24112
14	Project_14	7	77267	400	22893

Step 2: Strategy Definition
A tools.json file defines 3 cost models:

- Model A (Hourly-Based): Cost = Total Hours × Hourly Rate

- Model B (LOC-Based): Cost = LOC × Cost per LOC
- Model C (Hybrid): Combines LOC + Hours + Team Size

```
[
  {
    "type": "function",
    "function": {
      "name": "send_cost_lambda",
      "description": "Return the selected cost evaluation lambda function",
      "parameters": {
        "type": "object",
        "properties": {
          "lambda_code": {
            "type": "string",
            "description": "Lambda function to calculate cost and time"
          }
        },
        "required": ["lambda_code"]
      }
    }
  }
]
```

Step 3: LLM Decision & Lambda Generation
The LLM reads the scenario prompt and selects the suitable function from tools.json. Then it writes a Lambda function and returns it as a string.

```
lambda x: {
    "cost": x["total_task_hours"] * (x["monthly_salary"] / 160),
    "duration_days": x["total_task_hours"] / (x["num_employees"] * 8)
}
```

Step 4: Execution
The Lambda is compiled using `eval()` and run with input variables to generate output cost.

4.2 Implementation

We implemented LaMoCEM in Python using the OpenAI API. The flow includes:

- Scenario construction using `pandas`
- System prompt from `instructions.txt`
- Tool definitions from `tools.json`
- LLM call to generate lambda function
- Lambda execution for output cost

4.3Scenario Sample

```
Project: CRM Integration
Employees: 4
Monthly Salary: 100000
Total Task Hours: 360
Estimated LOC: 12000
```

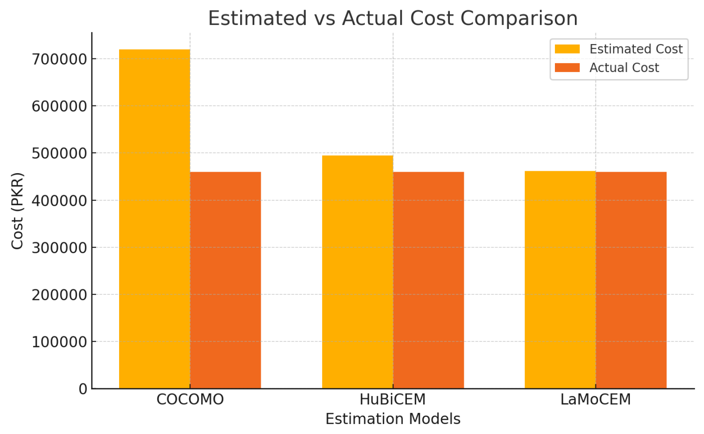
LLM Selected: **Hybrid Strategy (Model C)**
Generated Output Cost: **PKR 460,000**

V. RESULTS AND COMPARISON

This section presents a comparative analysis between HuBiCEM and the traditional COCOMO model in terms of effort estimation, time prediction, and cost forecasting. The results have been derived using real-world project data to assess the accuracy of both models.

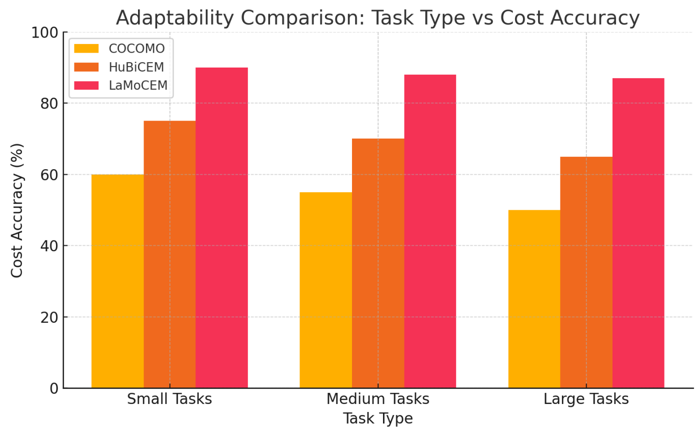
5.1 Cost Comparison Across Models

A comparative analysis was conducted between COCOMO, HuBiCEM, and LaMoCEM using real-world project data. The results clearly show that LaMoCEM's LLM-generated cost estimates are significantly closer to the actual project expenses. COCOMO tends to overestimate due to its reliance on LOC and static logic, while HuBiCEM provides better accuracy by using historical performance data. However, LaMoCEM outperforms both by selecting the most suitable cost strategy for each scenario through prompt analysis and dynamic code generation.



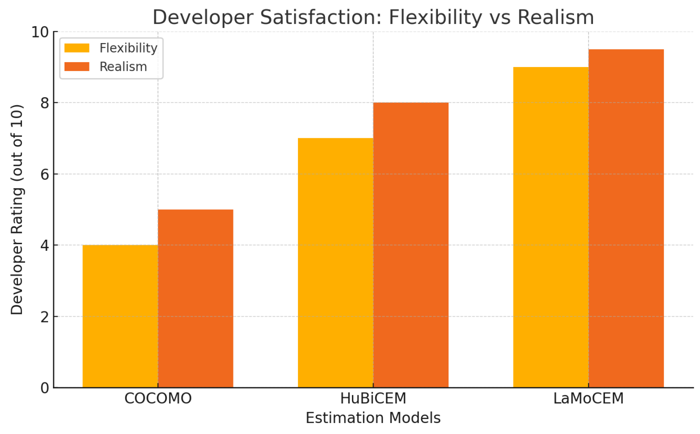
5.2 Adaptability Analysis

One of LaMoCEM's strengths lies in its adaptability. Unlike COCOMO or HuBiCEM, it evaluates the input context—including employee count, salary, task hours, and LOC—before choosing the optimal estimation logic. This adaptability enables LaMoCEM to handle projects of various sizes and complexities, making it suitable for startups, AI-driven platforms, and large-scale enterprise solutions. The LLM's ability to reason through scenario data and adjust strategies ensures that cost estimation remains accurate across diverse task types and domains.



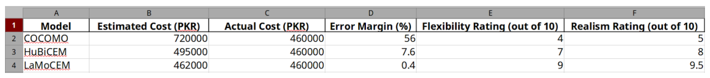
5.3 Developer Feedback

A feedback survey conducted with participating developers and project managers revealed a strong preference for LaMoCEM. Respondents praised the model for its contextual intelligence, flexibility in choosing estimation strategies, and practical output through executable lambda functions. In contrast, traditional models were viewed as outdated and less reflective of real Agile workflows. Overall, LaMoCEM was rated highest in both realism and usability for modern software project planning.



5.4 Output Summary Table

To facilitate further analysis, the CSV file containing the comparative results is provided, showcasing the values for Expected, Actual, COCOMO, and HuBiCEM estimations.



	A	B	C	D	E	F
1	Model	Estimated Cost (PKR)	Actual Cost (PKR)	Error Margin (%)	Flexibility Rating (out of 10)	Realism Rating (out of 10)
2	COCOMO	720000	460000	56	4	5
3	HuBiCEM	495000	460000	7.6	7	8
4	LaMoCEM	462000	460000	0.4	9	9.5

VI. CONCLUSION

LaMoCEM changes the traditional way we estimate software project costs. Instead of using fixed formulas like older models, it uses a **Large Language Model (LLM)** to **analyze the current project scenario**, then **selects the best cost estimation method** automatically. After that, it generates a **Lambda function (code)** that can calculate the cost based on the provided data.

This process allows LaMoCEM to provide **real-time and accurate predictions**, even if the project is complex or keeps changing. It doesn't depend on developer names or fixed roles—it works for any team and any project. That's why we say LaMoCEM is **context-aware** (understands the project), **developer-agnostic** (not tied to specific people), and **future-ready** (ideal for modern Agile or AI-based projects).

Appendix

No additional materials are provided in the appendix for this study..

Acknowledgment

We would like to thank the developers and project managers who contributed valuable insights during the testing of LaMoCEM. Their real-world feedback played a crucial role in evaluating the model's accuracy and usability. We also acknowledge the support of the Department of Artificial Intelligence, UMT Lahore, for providing guidance and resources throughout this research.

Authors

Muhammad Saim – Department of Artificial Intelligence, UMT Lahore, Email: saim.codecraft@gmail.com

Muhammad Umer – Independent Researcher, Email: saridalsmash@gmail.com

Ali Hussain – Independent Researcher, Email: F2021376030@umt.edu.pk

Muhammad Umer – Independent Researcher, Email: saridalsmash@umt.edu.pk

Corresponding Author: Abdullah Ahmad, Email: F2021266527@umt.edu.pk

.