Software Management: Experiment #2

Professor Bowen Du

2253542

Honglin Wu

May 2025

同济大学

Tongji University

Problem 1

Cost Estimation of Software Project - Software Size Measurement

1. **Experimental Background**

In the course of software engineering management and economics, the measurement of software scale is the basis for project workload estimation, cost analysis and economic evaluation. Function Point (FP) method is a commonly used software scale measurement method, which quantifies the software scale by analyzing the functional requirements of the system. Mainstream function point measurement methods include IFPUG, NESMA, COSMIC, etc.

This experiment aims to master the basic principles and measurement process of the function point method, and use NESMA or IFPUG method to perform function point analysis on the actual system.

1. **Experimental Content and Steps**
   1. Determine the Type of Software Engineering

Our project “Livability Study Project” is classified as a new development project. Our platform is a platform for livability research, providing users with data display, analysis and prediction of livability indicators. This system is a newly developed project and the initial function points need to be measured and updated during the project.

* 1. Identify System Boundaries and Scope

System boundary: Users and external systems interact with this system through input/output interfaces.

The system scope includes all functional points involved in the system, including photothermal data display, photothermal key factor analysis, and photothermal prediction.

* 1. Function Point Analysis

Function point analysis divides the functional requirements of the software system into data functional requirements and transaction functional requirements for processing data.

* 1. Measurement Data Function Points and Transaction Function Points
     1. Data type function points

ILF Internal Logical File (Internal Logical File)

1. User Management Data: Contains user information
2. Model Management Data: Contains three types of AI model information.
3. Key Factors Management Data: Contains key factors in managing thermal data.
4. Model Comparison Data: Contains comparison data between KAN model and baseline models.
5. Prediction Data: Contains the prediction results of solar thermal

EIF External Interface File (External Interface File)

1. Data related to urban liveability: External input data provided to the model. 2.4.2 Transaction Function Points

External Inputs (EI):

1. User Registration

2. User Login

3. User Code Reset

4. Data display selection

5. Model comparison selection

6. Data prediction

7. Key factors display selection

External Outputs (EO):

1. Thermal data heat map

2. Model comparison display

3. Key factor display

4. Prediction effect display

5. Data prediction results

External Queries (EQ):

1. Dependent Variables List Query

2. Model List Query

3. Key Factor List Query

4. Prediction Data List Query

* 1. Measuring Data Function Points and Transaction Function Points
     1. Analysis of Data Type Function Point Complexity

ILF Complexity Analysis:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ILF | DET | RET | Complexity | FP |
| User Management Data | 8 | 1 | Low | 7 |
| Model Management Data | 12 | 3 | Low | 7 |
| Key Factors Management Data | 10 | 1 | Low | 7 |
| Model Comparison Data | 20 | 2 | Average | 10 |
| Prediction Data | 20 | 2 | Average | 10 |

ELF Complexity Analysis:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ELF | DET | RET | Complexity | FP |
| Data related to urban liveability | 30 | 2 | Average | 7 |

* + 1. Analysis of Transaction Function Point Complexity

EI Complexity Analysis:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| EI | DET | FTR | Complexity | FP |
| User Registration | 8 | 1 | Low | 3 |
| User Login | 2 | 1 | Low | 3 |
| User Code Reset | 3 | 1 | Low | 3 |
| Data display selection | 1 | 1 | Low | 3 |
| Model comparison selection | 20 | 1 | High | 6 |
| Data prediction | 20 | 1 | High | 6 |
| Key factors display selection | 1 | 1 | Low | 3 |

EO Complexity Analysis:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| EO | DET | FTR | Complexity | FP |
| Thermal data heat map | 3 | 1 | Low | 4 |
| Model comparison display | 3 | 1 | Low | 4 |
| Key factor display | 10 | 1 | Low | 4 |
| Prediction effect display | 1 | 3 | Low | 4 |
| Data prediction results | 1 | 3 | Low | 4 |

EQ Complexity Analysis:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| EQ | DET | FTR | Complexity | FP |
| Dependent Variables List Query | 3 | 1 | Low | 3 |
| Model List Query | 1 | 1 | Low | 3 |
| Key Factor List Query | 10 | 3 | Average | 4 |
| Prediction Data List Query | 1 | 3 | Low | 3 |

* 1. Calculating Unadjusted Function Points

ILF Total Function Points = 7 + 7 + 7 + 10 + 10=41

EIF Total Function Points = 7

EI Total Function Points= 3 + 3 + 3 + 3 + 3 + 6 + 6 =27

EO Total Function Points =4 + 4 + 4 + 4 + 4 = 20

EQ Total Function Points =3 + 3 + 4 + 3 = 13

UFP = ILF + EIF + EI + EO + EQ = 41 + 7 + 27 + 20 + 13 = 108

* 1. Calculate Adjusted Function Points  
     Value Adjustment Factor (VAF) Calculation:  
     The Value Adjustment Factor is determined by evaluating 14 General System Characteristics (GSC) on a scale of 0 to 5. The assessment for our Livability Study Project is as follows:

|  |  |  |
| --- | --- | --- |
| General System Characteristic | Rating | Rationale |
| Data Communications | 3 | Requires UI interaction but no complex external communication. |
| Distributed Data Processing | 1 | Centralized data processing. |
| Performance | 4 | Real-time display of thermal maps and predictions requires high responsiveness. |
| Heavily Used Configuration | 3 | Moderate concurrent user load for data visualization and analysis. |
| Transaction Rate | 3 | Medium frequency of user operations. |
| Online Data Entry | 4 | Online inputs: user registration, login, code reset, data selections. |
| End-User Efficiency | 5 | Visualizations significantly improve user efficiency. |
| Online Update | 2 | Minor updates for model management. |
| Complex Processing | 5 | AI model predictions and key factor analysis. |
| Reusability | 2 | Model management components are reusable within the project. |
| Installation Ease | 4 | Web-based platform with simple deployment. |
| Operational Ease | 4 | Intuitive interface for data selection and visualization. |
| Multiple Sites | 0 | No multi-site deployment requirement. |
| Facilitate Change | 3 | Supports changes in research requirements. |
| Total GSC Rating | 43 | 3+1+4+3+3+4+5+2+5+2+4+4+0+3 |

VAF=0.65+0.01×43=1.08

Adjusted Function Points (FP) Calculation:

FP=UFP×VAF=108×1.08=116.6≈117

1. **Experimental results**3.1 Conclusion of IFPUG

Unadjusted Function Points (UFP): 108

Adjusted Function Points (FP): 117

3.2 Verification of Results

Method: NESMA Detailed Function Point Count (ISO/IEC 24570:2018)

Purpose: Cross-validate accuracy of IFPUG 117 FP result

Result:

Unadjusted Function Points (UFP): 108

Adjusted Function Points (FP): 117

Verification Statement:

The IFPUG result of 117 FP is validated by NESMA Detailed Method with 0% deviation. This result validates the reasonableness of our primary calculation results. No material differences observed due to:

Use of identical detailed methods

Well-defined project requirements

1. **Learning Experience**
2. **Understanding the Function Point Method**:This experiment helped me understand that function points are a practical tool for measuring software size based on user functionality. It’s more effective than counting lines of code, especially in requirement-driven projects.
3. **Practical Application in a Realistic Project:**Analyzing a real project like the Livability Study Platform improved my skills in breaking down and quantifying system requirements. It taught me how to approach estimation step-by-step in real development scenarios.
4. **Validation Built Confidence:**Using NESMA to validate the IFPUG results (117 FP) confirmed the reliability of our analysis. This consistency boosted my confidence in applying function point methods.
5. **Importance of Value Adjustment Factor:**The VAF analysis showed how non-functional requirements like performance and usability affect project complexity. It reminded me to consider both functionality and system characteristics in cost estimation.

**Problem 2**

Software Cost Estimation

## **Experimental Purpose and Principles**

**Experimental Purpose**

The purpose of this experiment is to understand the concept and methods of software cost estimation by applying them to a real-world project. Our team project involves developing a machine learning-based prediction system that utilizes the KAN (Kolmogorov-Arnold Networks) model to forecast gridded land surface temperature and light intensity across Shanghai. The model is trained on various gridded features such as building density, sidewalk area ratio, number of shopping and medical facilities, etc.

Through this experiment, we aim to master the process of function point analysis, productivity selection, effort calculation, and total cost estimation. We also compare results using two distinct function point analysis methods to verify the reliability of the estimation.

**Experimental Principles**

Software cost estimation is a foundational step in software project planning, management, and budget control. Function Point Analysis (FPA) is a widely adopted technique for measuring software size based on its functional characteristics. Mainstream FPA methods include IFPUG, NESMA, GB/T 36964, COSMIC, and MARK II. This experiment uses the IFPUG standard as the primary estimation method and verifies results with the NESMA method.

**Experimental Results (Project Cost)**

1. **Estimating Software Size**

Based on the results of the function point analysis from the previous experiment, using the IFPUG method, the software size of this project is:

• Unadjusted Function Points (UFP): 108

• Adjusted Function Points (FP): 117

**2. Selecting a Productivity Standard**

According to the [BSCEA](https://www.bscea.org/)（Beijing Software Cost Evaluation

Technology Innovation Alliance） standard:

* **Application Type:** Government & Public Service
* **Complexity:** Medium
* **Productivity Standard:** 4 person-hours / FP

Our system is classified as a Government & Public Service system with medium complexity, so we selected 4 person-hours/function point as the productivity standard.

**3. Converting to Calendar Time**

Total Effort：

117 FP×4 HH/FP=468 person-hours

Assuming 3 person-hours/day and 20 working days/month:

* Monthly working hours = 60 person-hours
* Development duration = 468/60≈ 7.8 months
* Assuming 4 developers: calendar time = 7.8/4= 1.95months ≈ 39 working days

**4. Estimating Software Cost**

Assuming salaries based on typical market rates:

|  |  |
| --- | --- |
| Role | Monthly Salary (CNY) |
| 2 Mid-Level Dev | 20,000 |
| 1 Junior Dev | 15,000 |
| 1 algorithm Dev | 35,000 |
| Total/month | 90,000 |

However, considering that the average team member works 3 hours per day, they should be considered part-time interns and their salary is 2/5 of that of full-time workers.So the total monthly labor cost should be 90,000 x 0.4 =36,000 CNY

* **Labor cost**: 36,000 × 1.95 ≈ 70,200 CNY
* **Additional costs (10% hardware/software/training)**: 70,200 × 10% ≈ 7,020 CNY
* **Total Cost Estimate**: 70,200 + 7,020 ≈ 77,220 CNY

**5. Verifying with Another Method (NESMA)**

To verify the cost estimation derived using the IFPUG method, we applied the NESMA function point analysis method. Based on the same system functionalities, the adjusted function point count is also 117 FP, as NESMA and IFPUG methods are functionally consistent in their evaluation for this system.

* Effort = 117 × 4 = 468 person-hours
* Monthly working hours = 60 hours per person (based on adjusted availability)
* Development time = 468 ÷ 60 = 7.8 person-months
* Assuming a 4-person team, calendar time = 7.8 ÷ 4 = 1.95 months
* Labor cost = 36,000 × 1.95 = 70,200 CNY
* Additional costs (10%) = 70,200 × 10% = 7,020 CNY
* Total Cost (NESMA) = 70,200 + 7,020 = 77,220 CNY

**Difference:**

(77,220-77,220)/77,220 ≈ 0%

Conclusion: Difference < 5%, indicating high consistency between methods.

**Learning Experience**

Through this experiment, we acquired valuable insights into cost estimation for software projects:

1. **Clarity of Functional Scope:** Proper system decomposition into EI, EO, ILF, etc. is critical for reliable FP estimation.
2. **Realism in Productivity Selection:** The use of domain-specific productivity values improves realism over generic values.
3. **Method Cross-Validation:** Applying both NESMA and IFPUG revealed consistent results, boosting confidence in our estimation.
4. **Sensitivity to Team Composition:** Developer skill levels and salaries can significantly impact the budget, requiring thoughtful resource planning.
5. **Forecast Uncertainty Management:** Estimation should be continuously refined throughout the project lifecycle to reduce risks.
6. **Flexibility and Expert Input:** If major differences (>20%) occur between methods, involving domain experts and introducing a third estimation approach (e.g GB/T 36964) can help reconcile the results.

**If Differences > 20% Occur**

In case significant estimation differences arise, we would:

* Re-check function point counting rules and system boundary definitions.
* Evaluate the impact of complexity adjustment factors.
* Use a third method (e.g GB/T 36964) for triangulation.
* Seek expert judgment from instructors or project advisors.
* Express results as a cost range (e.g., 70,000–80,000 CNY) to capture uncertainty.