# Presentation Script
Multi-Schema Software Effort Estimation Using Machine Learning

Phan Hoang Long

January 1, 2026

## Contents

# Slide 1: Title Slide

*[30 seconds]*

Good morning/afternoon everyone. My name is Phan Hoang Long, and I'm from the Department of Computer Science at Chungbuk National University.

Today, I'll present my research on **Multi-Schema Software Effort Estimation Using Machine Learning**, which proposes an enhanced COCOMO II approach with heterogeneous data integration.

*[Pause]*Let's begin.

# Slide 2: Motivation

*[2 minutes]*

First, let me explain **why effort estimation matters**.

**The Problem:**

- Inaccurate effort estimation leads to budget overruns and project failures
- Traditional COCOMO II has limited adaptability to modern diverse projects
- Real-world data is heterogeneous and inconsistent across different organizations

**The Impact is significant:**

- Studies show that 85% of software projects face budget issues
- 78% experience schedule delays
- And 42% are at risk of complete failure

As you can see in the figure, current challenges in effort estimation are severe.

**The Research Gap:** There is a lack of unified framework to handle heterogeneous project data across different sizing schemas like Lines of Code, Function Points, and Use Case Points.

*[Pause]*This is the problem we're addressing today.

# Slide 3: Research Contributions

*[1.5 minutes]*

Our research makes **three main contributions**:

**First - Data Integration:** We developed an automatic normalization pipeline for LOC, Function Points, and Use Case Points schemas. This allows us to integrate data from multiple sources seamlessly.

**Second - Machine Learning Models:** We benchmarked COCOMO II against four ML models: Linear Regression, Decision Tree, Random Forest, and Gradient Boosting.

**Third - Deployment:** We built a REST API for real-world usage, making the system practical and accessible.

**Key Results:**

- We integrated over 320 projects from multiple sources

- Our best model, Random Forest, achieves 38% MMRE and 58% PRED(25)
- This reduces estimation error by 34% compared to the COCOMO II baseline

*[Pause]*These are significant improvements over traditional methods.

## Slide 4: COCOMO II Background

*[2 minutes]*

Let me briefly review the **COCOMO II model** that serves as our baseline.

As shown in the formula, COCOMO II estimates effort using three main components:

**First:** A calibration constant A, which is 2.94 **Second:** Size raised to an exponent E, which includes scale factors like precedentedness and flexibility **Third:** Product of effort multipliers - there are 17 cost drivers that affect the final estimate

From effort, we can derive duration and team size using additional formulas.

**However, the limitation** of traditional COCOMO II is that it assumes homogeneous data and requires manual calibration. This is not scalable for modern diverse projects with heterogeneous data sources.

*[Pause]*This is why we need a machine learning approach.

## Slide 5: Dataset Overview

*[1.5 minutes]*

Now, let's look at our **data sources**.

We collected data from three different schemas:

- **LOC-based data:** 180 projects from NASA COCOMO and COC81 datasets
- **Function Points:** 95 projects from Desharnais and Albrecht datasets
- **Use Case Points:** 45 projects from various sources

In total, we have 320 projects with mixed metrics.

**The Key Challenge** is data heterogeneity:

- Different size metrics across schemas
- Inconsistent effort units - some in hours, others in person-months
- Missing values and different contexts

**Our Solution:** We developed an automatic schema detection and unit normalization pipeline to handle this heterogeneity.

*[Pause]*Let me show you how this works.

## Slide 6: Data Heterogeneity Visualization

*[1.5 minutes]*

This figure illustrates the **challenge of data heterogeneity**.

**Before normalization:** You can see we have incompatible schemas - some projects measured in LOC, others in Function Points or Use Case Points. The units are inconsistent.

**After our unified normalization:** All data is standardized and ready for machine learning.

Our pipeline handles three key aspects:

1. **Unit conversion:** Converting hours to person-months, LOC to KLOC
2. **Missing values:** Using IQR-based outlier detection and median imputation
3. **Schema tagging:** We preserve the origin information for schema-specific modeling

*[Pause]*This preprocessing is crucial for our multi-schema approach.

## Slide 7: Preprocessing Pipeline

*[2 minutes]*

Here's our **end-to-end automated preprocessing pipeline**.

The process follows six key steps:

**Step 1:** Schema detection - automatically identify whether data is LOC, Function Points, or Use Case Points

**Step 2:** Unit standardization - convert all measurements to common units

**Step 3:** Outlier handling using the IQR method to remove extreme values that would skew our models

**Step 4:** log1p transformation for linearization - this helps normalize the skewed distribution of effort data

**Step 5:** Feature scaling and encoding - standardizing numerical features and encoding categorical ones

**Step 6:** Export ML-ready data that can be directly fed into our models

*[Pause]*This pipeline is fully automated and can process new data without manual intervention.

## Slide 8: Experimental Setup

*[2 minutes]*

Now let's discuss our **experimental setup**.

**Models Evaluated:** We compared five approaches:

- COCOMO II as our analytical baseline
- Linear Regression
- Decision Tree
- Random Forest

- Gradient Boosting

**Training Strategy:**

- 80/20 train-test split
- GridSearchCV for hyperparameter optimization
- 5-fold cross-validation to ensure robustness

**Evaluation Metrics:** We use five standard metrics:

- MAE and RMSE - lower is better
- MMRE - Mean Magnitude Relative Error, lower is better
- PRED(25) - percentage of predictions within 25% of actual, higher is better
- R-squared for goodness of fit

According to Conte et al., industry considers MMRE less than 0.25 and PRED(25) greater than 0.75 as acceptable performance.

*[Pause]*Let's see our results.

## Slide 9: Overall Results

*[2 minutes]*

Here are our **main results**.

As you can see from the charts, **Random Forest achieves the best performance across all metrics**.

**Key Findings:**

- Random Forest reduces MMRE by 34% - from 0.58 in COCOMO II down to 0.38
- PRED(25) improves to 58% - meaning 58% of our predictions are within 25% of actual effort
- Ensemble methods - Random Forest and Gradient Boosting - are clearly superior to simpler baselines

The improvement is consistent across all four metrics: MAE, RMSE, MMRE, and PRED(25).

While we don't reach the ideal thresholds yet, this represents **significant progress** over traditional COCOMO II, especially considering the heterogeneous nature of our dataset.

*[Pause]*Now let's look at schema-specific performance.

## Slide 10: Schema-Specific Performance

*[1.5 minutes]*

This chart shows **performance varies by schema** due to data availability.

**LOC Schema - shown in green:**

- 180 samples available
- Provides stable and reliable predictions
- This is our best-performing schema

**Function Points Schema - shown in blue:**

- 95 samples
- Achieves moderate accuracy
- Still good but slightly less reliable than LOC

**Use Case Points Schema - shown in orange:**

- Only 45 samples
- Shows higher uncertainty
- This needs more data collection in future work

The key insight here is that **data quantity matters**. Schemas with more training examples perform better, which is expected in machine learning.

*[Pause]*But our multi-schema approach still works across all three types.

## Slide 11: Error Analysis

*[2 minutes]*

Let's examine **error analysis and model interpretability**.

**Left plot - Actual vs Predicted:**

- Points close to the diagonal line indicate accurate predictions
- Our Random Forest model performs well across all project sizes
- There's no systematic bias - we're not consistently over or under-estimating

**Right plot - Feature Importance:** This shows which features contribute most to predictions:

- **Size metric** is the dominant predictor at 38% importance - this makes intuitive sense
- **Schema type** contributes 22% - this justifies our multi-schema approach
- Other COCOMO II cost drivers also contribute significantly

The fact that schema type has 22% importance confirms that **different schemas carry different information**, and our unified approach successfully leverages this.

*[Pause]*This validates our design decisions.

## Slide 12: Residual Analysis

*[1.5 minutes]*

For **model diagnostics**, we performed residual analysis.
The plots show three important validations:
**Left - Residual Scatter Plot:**

- Random scatter around zero - no systematic bias
- Homoscedastic pattern - constant variance across prediction range
- This indicates our model assumptions are valid

**Right - Residual Distribution:**

- Near-normal distribution
- This allows us to compute reliable confidence intervals
- Small standard deviation indicates consistent predictions

These diagnostics confirm that our Random Forest model is **statistically sound** and not overfitting or underfitting the data.

*[Pause]*Now let's move to the practical deployment.

## Slide 13: Deployment Architecture

*[2 minutes]*

We deployed our solution as a **REST API for production use**.
The architecture has four key components:
**1. Schema-aware Routing:**

- Automatically detects whether input is LOC, Function Points, or Use Case Points
- Routes to the appropriate preprocessing pipeline

**2. Model Registry:**

- Maintains separate trained models for each schema
- Ensures optimal performance per schema type

**3. Traceability:**

- Preserves data source information
- Provides confidence scores with each prediction

**4. Extensibility:**

- Ready for integration with requirement analysis tools
- Can connect to Jira for automated estimation

The API accepts project metrics as input and returns effort, duration, and team size estimates along with confidence intervals.

*[Pause]*This makes our research practically useful.

## Slide 14: Practical Applications

*[2 minutes]*

Let me highlight the **practical applications and use cases**.

**Current Deployment:**

- We have an API endpoint at /api/estimate
- Input: Project requirements or metrics
- Output: Effort, Duration, Team Size plus confidence scores

**Supported Modes:** The system works in four modes:

1. LOC-based estimation
2. Function Point estimation
3. Use Case Point estimation
4. Mixed mode with automatic detection

**Future Extensions we're planning:**

- **NLP Integration:** Extract metrics automatically from requirement documents
- **Story Point Mapping:** Support for Agile projects
- **Jira Plugin:** Real-time estimation in issue tracking systems
- **Continuous Learning:** Feedback loop for ongoing model improvement

**Impact:** Our system reduces manual estimation time by 70% while improving accuracy by 34%.

*[Pause]*This demonstrates real business value.

## Slide 15: Limitations & Future Work

*[2 minutes]*

Now, let me be honest about our **current limitations**.

**Three main limitations exist:**

**First - Data scarcity in UCP schema:**

- Only 45 samples lead to higher uncertainty
- We need more Use Case Point projects

**Second - Context factors not fully captured:**

- Domain-specific calibration may still be needed
- Industry context affects estimation but isn't fully modeled

**Third - Static models require periodic retraining:**

- Technology evolution isn't automatically tracked
- Models can become outdated over time

**Honest assessment:** Our model works best for similar project types. Extreme outliers are still challenging.

**Future Roadmap:**

1. **Data Augmentation:** Collect more UCP projects, possibly use synthetic data
2. **Deep Learning:** Neural networks for complex non-linear relationships
3. **Online Learning:** Incremental updates from project feedback
4. **Multi-modal Input:** Combine metrics, text requirements, and historical data
5. **Uncertainty Quantification:** Probabilistic predictions with confidence intervals

*[Pause]*These improvements will make the system even more robust.

## Slide 16: Conclusion

*[1.5 minutes]*

Let me conclude with a **summary of our contributions**.

**We delivered three main contributions:**

1. **Unified Pipeline:** Automatic normalization for LOC, Function Points, and Use Case Points data
2. **Validation:** Random Forest reduces MMRE by 34% compared to COCOMO II
3. **Deployment:** REST API for real-world practical usage

**Key Takeaways:**

- Data integration is crucial for modern software estimation
- Ensemble machine learning methods are superior to traditional approaches
- Schema-aware modeling handles heterogeneous data effectively

**Impact:**

- Enables data-driven project planning
- Reduces estimation bias significantly
- Provides multi-schema support for diverse organizations

**Thank you for your attention!**

I'm now happy to take your questions and discuss any aspects of this research.

*[Pause]*

# Backup: Anticipated Questions & Answers

## Q1: Why not use deep learning instead of Random Forest?

**Answer:** Great question. We actually considered deep learning, but for three reasons, Random Forest was more appropriate for this problem:

1. **Dataset size:** With only 320 samples, deep learning would likely overfit. Random Forest works well with smaller datasets.

2. **Interpretability:** Random Forest provides feature importance scores, which help us understand what drives effort estimation. Deep learning is more of a black box.

3. **Performance:** In our experiments, Random Forest achieved the best results. Adding complexity doesn't always improve performance.

However, as we collect more data - especially reaching thousands of projects - deep learning would become more viable and is definitely in our future roadmap.

## Q2: How do you handle new COCOMO II cost drivers not in your training data?

**Answer:** That's an important practical question. We handle this in two ways:

1. **Default values:** For missing cost drivers, we use COCOMO II default values of 1.0, which represent nominal effort multipliers.

2. **Schema detection:** Our pipeline identifies which schema the new data belongs to and applies the appropriate preprocessing.

However, if a project has completely new characteristics never seen in training, our confidence scores will reflect higher uncertainty. This is where continuous learning comes in - we can retrain models as new data becomes available.

## Q3: What's the computational cost of your approach?

**Answer:** Excellent question about practical deployment.

**Training time:**

- Random Forest training: approximately 8 seconds on our dataset
- One-time cost, only needed when retraining

**Prediction time:**

- Less than 50 milliseconds per project
- This includes preprocessing and prediction
- Fast enough for real-time API responses

So the computational cost is very reasonable, even for large-scale deployment. A single server can handle thousands of estimation requests per hour.

## Q4: How does your approach compare to recent deep learning papers?

**Answer:** We conducted a literature review, and I can share some comparisons:

**Recent deep learning approaches:**

- Achieve 15-25% MMRE on single-schema datasets
- But require 1000+ training samples
- And typically focus on one schema only

**Our approach:**

- Achieves 38% MMRE on multi-schema heterogeneous data
- Works with 320 samples across three schemas
- Provides practical multi-schema support

The key difference is that we handle **heterogeneous real-world data**, while many research papers use cleaner, single-source datasets. Our slightly higher error rate is the trade-off for greater practical applicability.

## Q5: Can you explain the 34% improvement more clearly?

**Answer:** Absolutely. Let me break down the 34% improvement:

**COCOMO II Baseline:**

- MMRE = 0.58
- This means on average, estimates are off by 58%

**Random Forest:**

- MMRE = 0.38
- Average estimation error is 38%

**Improvement calculation:**

- Absolute improvement: 0.58 - 0.38 = 0.20
- Relative improvement: 0.20 / 0.58 = 34%

So we reduced the estimation error by one-third compared to traditional COCOMO II. This is significant in practice - it means fewer budget overruns and better project planning.

## Q6: What about different software domains - web apps, embedded systems, etc.?

**Answer:** Another excellent question about generalization.

Our current dataset includes projects from multiple domains, but we don't have enough samples per domain to build domain-specific models yet.

**Current approach:**

- We use COCOMO II application type as a feature
- The model learns domain differences implicitly

**Future improvement:**

- Collect domain-labeled data
- Build ensemble models with domain-specific branches
- Allow users to specify domain for better calibration

This is definitely an area for future research - domain adaptation in software effort estimation.

## Q7: How do you ensure the API doesn't get abused or produce wrong estimates?

**Answer:** Security and reliability are critical for production deployment.

**We implement several safeguards:**

1. **Input validation:**
    - Range checks on all metrics
    - Schema consistency validation
    - Reject obviously invalid inputs
2. **Confidence scoring:**
    - Each prediction comes with confidence score
    - Low confidence triggers warning
    - Users see uncertainty estimates
3. **Rate limiting:**
    - API key required
    - Request throttling per user
    - Prevents abuse
4. **Logging and monitoring:**
    - All predictions logged
    - Anomaly detection on inputs
    - Alert system for suspicious patterns

**Most importantly:** We make it clear that estimates are **guidance, not guarantees**. Human judgment should always be involved in final decisions.