Software Engineering Economics Tool - Technical Documentation & User Manual

Table of Contents

1. System Overview
2. Technical Architecture
3. Models and Algorithms
4. Calculations Reference
5. User Manual
6. API Documentation
7. Database Schema

System Overview

The Software Engineering Economics Tool is a comprehensive web application built with Next.js that provides interactive economic analysis capabilities for software development projects. The system integrates four core modules:

- **Cost Estimation**: COCOMO and Function Points methodologies
- **Budgeting & Financial Analysis**: ROI, NPV, IRR calculations
- **Risk Management**: Sensitivity analysis and Monte Carlo simulations
- **Resource Allocation**: Optimal resource assignment algorithms

Technical Architecture

Technology Stack

- **Frontend**: Next.js 14 with TypeScript
- **Styling**: Tailwind CSS
- **Charts**: Chart.js and D3.js
- **Database**: MySQL with connection pooling
- **Deployment**: Vercel platform

Project Structure

```
src/
├── app/
│   ├── budgeting/          # Financial analysis module
│   ├── cost-estimation/    # COCOMO & Function Points
│   ├── risk-management/    # Risk analysis tools
│   ├── resource-allocation/ # Resource optimization
│   └── api/                # Backend API endpoints
├── components/
│   └── charts/             # Visualization components
└── lib/
    ├── models.ts           # Data models
    └── mysql.ts            # Database connection
```

Models and Algorithms

1. Cost Estimation Models

COCOMO (Constructive Cost Model)

Purpose: Estimate software development effort, time, and team size

Algorithm:

```
const coefficients = {
  organic: { a: 2.4, b: 1.05, c: 2.5, d: 0.38 },
  'semi-detached': { a: 3.0, b: 1.12, c: 2.5, d: 0.35 },
  embedded: { a: 3.6, b: 1.20, c: 2.5, d: 0.32 }
};

effort = a × (KLOC)^b × EAF
duration = c × (effort)^d
teamSize = effort / duration
cost = effort × 8000 // $8000 per person-month
```

Parameters:

- **KLOC**: Thousands of Lines of Code
- **EAF**: Effort Adjustment Factor
- **Project Types**:
    - Organic: Small, experienced teams
    - Semi-detached: Medium complexity
    - Embedded: Complex, real-time systems

Function Points Analysis

Purpose: Size-based estimation independent of technology

Algorithm:

```
weights = {
  externalInputs: 4,
  externalOutputs: 5,
  externalInquiries: 4,
  internalFiles: 10,
  externalInterfaces: 7
};

unadjustedFP = Σ(component × weight)
adjustedFP = unadjustedFP × complexityFactor
estimatedLOC = adjustedFP × languageFactor
effort = adjustedFP × 0.04
```

2. Financial Analysis Models

Return on Investment (ROI)

```
ROI = ((Total Cash Flows - Initial Investment) / Initial Investment) × 100
```

Net Present Value (NPV)

```
NPV = -Initial Investment + Σ(Cash Flow_t / (1 + discount_rate)^t)
```

Internal Rate of Return (IRR)

Algorithm: Newton-Raphson iterative method

```
// Iterative calculation until NPV ≈ 0
for (let i = 0; i < maxIterations; i++) {
  npv = -initialInvestment + Σ(cashFlow / (1 + irr)^period)
  derivative = -Σ(period × cashFlow / (1 + irr)^(period + 1))
  irr = irr - npv / derivative
}
```

Payback Period

```
// Time to recover initial investment
cumulativeCashFlow = -initialInvestment
for each year:
  cumulativeCashFlow += annualCashFlow
  if (cumulativeCashFlow >= 0) return period + fraction
```

3. Risk Management Models

Sensitivity Analysis

Purpose: Analyze impact of variable changes on project outcomes

```
for (variableChange from -range to +range by 5%) {
  newValue = baseValue × (1 + variableChange/100)
  impact = (newValue - baseValue) × impactFactor
  totalValue = baseValue + impact
}
```

Monte Carlo Simulation

Purpose: Probabilistic risk analysis using random sampling

Algorithm:

```
for (i = 0; i < iterations; i++) {
  totalCost = 0
  for each variable {
    if (distribution === 'uniform') {
      value = random() × (max - min) + min
    } else { // normal distribution
      mean = (max + min) / 2
      std = (max - min) / 6
      value = normalDistribution(mean, std)
    }
    totalCost += value
  }
  results.push(totalCost)
}

// Statistical analysis
mean = Σ(results) / iterations
median = sortedResults[iterations/2]
p5 = percentile(results, 5)
p95 = percentile(results, 95)
std = √(Σ(result - mean)² / iterations)
```

4. Resource Allocation Algorithm

Purpose: Optimal assignment of resources to tasks

```
// Simplified greedy algorithm
for each task {
  suitableResources = resources.filter(r =>
    task.requiredSkills.some(skill => r.skills.includes(skill))
  )

  bestResource = suitableResources.reduce((best, current) => {
    currentEfficiency = skillMatch / current.hourlyRate
    bestEfficiency = skillMatch / best.hourlyRate
    return currentEfficiency > bestEfficiency ? current : best
  })

  efficiency = (skillMatch / task.requiredSkills.length) × 100
}
```

Calculations Reference

Cost Estimation Formulas

Model	Formula	Variables
COCOMO Effort	$E = a \times (KLOC)^b \times EAF$	E: effort (person-months), KLOC: thousands of lines of code
COCOMO Duration	$D = c \times E^d$	D: duration (months)
Function Points	$FP = \Sigma(\text{components} \times \text{weights}) \times CAF$	CAF: Complexity Adjustment Factor

Financial Metrics

Metric	Formula	Interpretation
ROI	((Gains - Cost) / Cost) × 100	Percentage return on investment
NPV	$\Sigma(CF_t / (1+r)^t) - I_0$	Present value of future cash flows
IRR	Rate where NPV = 0	Internal rate of return
Payback	Time to recover initial investment	Risk indicator

Risk Metrics

Metric	Formula	Purpose
Risk Score	Probability × Impact	Quantify risk level
Expected Value	Σ(Outcome × Probability)	Average expected result
Standard Deviation	$\sqrt{\Sigma(x - \mu)^2 / N}$	Measure of variability

User Manual

Getting Started

1. **System Requirements**
   - Modern web browser (Chrome, Firefox, Safari, Edge)
   - Internet connection
   - No additional software installation required
2. **Accessing the Application**

- Navigate to the application URL
- The home page displays four main modules
- Click on any module to begin analysis

Module Usage Guide

1. Cost Estimation Module

COCOMO Analysis:

1. Select project type (Organic/Semi-detached/Embedded)
2. Enter estimated KLOC (thousands of lines of code)
3. Adjust Effort Adjustment Factor (1.0 = nominal)
4. View calculated effort, duration, team size, and cost

Function Points Analysis:

1. Count external inputs, outputs, inquiries
2. Count internal files and external interfaces
3. Set complexity factor (0.65-1.35)
4. Choose programming language factor
5. Review estimated function points and LOC

Best Practices:

- Use historical data for accurate KLOC estimation
- Consider project complexity when selecting COCOMO mode
- Validate function point counts with requirements

2. Budgeting & Financial Analysis

Setting Up Analysis:

1. Enter initial investment amount
2. Input annual cash flows (up to 10 years)
3. Set discount rate (typically 8-15%)
4. Review calculated ROI, NPV, IRR, and payback period

Interpreting Results:

- **Positive NPV**: Project adds value
- **IRR > Discount Rate**: Project is profitable
- **Shorter Payback**: Lower risk
- **Higher ROI**: Better return

Tips:

- Use realistic cash flow projections
- Consider inflation in discount rate
- Compare multiple scenarios

3. Risk Management

Sensitivity Analysis:

1. Set base value for analysis
2. Define variable range (±percentage)
3. Set impact factor
4. Review sensitivity scenarios

Monte Carlo Simulation:

1. Define risk variables with min/max values
2. Choose distribution type (uniform/normal)
3. Set number of iterations (1000+ recommended)
4. Analyze statistical results

Risk Interpretation:

- **P5/P95**: 90% confidence interval
- **Standard Deviation**: Measure of uncertainty
- **Mean vs Median**: Distribution skewness

4. Resource Allocation

Resource Management:

1. Add team members with skills and rates
2. Define project tasks with requirements
3. Set task priorities and dependencies
4. Run optimization algorithm

Optimization Results:

- View optimal resource assignments
- Check efficiency percentages

- Analyze cost implications
- Review timeline impacts

Scenario Analysis:

- Compare Fast/Balanced/Economic approaches
- Adjust quality/cost/time weights
- Evaluate trade-offs

Advanced Features

Data Visualization

- Interactive charts for all analyses
- Risk matrix visualization
- Cost breakdown charts
- Timeline and resource utilization graphs

Data Export

- Save analysis results
- Export charts and reports
- Share scenarios with stakeholders

Integration

- API endpoints for external integration
- Database storage for project history
- Batch processing capabilities

Troubleshooting

Common Issues:

1. **Negative NPV**: Check cash flow projections and discount rate
2. **High Risk Scores**: Review assumptions and add mitigation strategies
3. **Resource Conflicts**: Adjust availability or add resources
4. **Unrealistic Estimates**: Validate input parameters

Performance Tips:

- Use appropriate iteration counts for Monte Carlo
- Limit sensitivity analysis ranges
- Optimize resource allocation scenarios

API Documentation

Endpoints

```
// Project Management
GET    /api/projects          // List all projects
POST   /api/projects          // Create new project
GET    /api/projects/[id]     // Get project details
PUT    /api/projects/[id]     // Update project
DELETE /api/projects/[id]     // Delete project
```

Data Models

```
interface Project {
  id?: number;
  name: string;
  description: string;
  costEstimation?: CocomoData | FunctionPointData;
  budgeting?: FinancialData;
  riskManagement?: RiskData;
  resourceAllocation?: ResourceData;
}
```

Database Schema

The system uses MySQL with the following key tables:

- **projects**: Main project information
- **cost_estimations**: COCOMO and Function Point data
- **financial_analysis**: ROI, NPV, IRR calculations
- **risk_items**: Risk register entries
- **resources**: Team member information
- **tasks**: Project task definitions

Performance Considerations

- Indexed foreign keys for fast joins
- Optimized queries for large datasets
- Connection pooling for scalability
- Caching for frequently accessed data

Conclusion

This Software Engineering Economics Tool provides comprehensive analysis capabilities for software project economic evaluation. The combination of proven methodologies (COCOMO, Function Points), financial analysis, risk management, and resource optimization makes it a valuable tool for project managers, software engineers, and stakeholders involved in software development decision-making.

For technical support or feature requests, please refer to the project repository or contact the development team.