EXPLORING A FEASIBLE PLATFORM FOR PROJECT MANAGEMENT

Project ID: RP24-002

Project Proposal Report

Fernando R.D.S.A – IT21011948

B.Sc. (Hons) in Information Technology Specializing in Information Systems Engineering

Department of Computer Systems Engineering

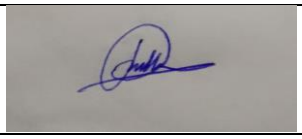
Sri Lanka Institute of Information Technology

Sri Lanka

February 2024

DECLARATION

We declare that this is our own work, and this proposal does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other university or Institute of higher learning and to the best of our knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Name	Student ID	Signature
Fernando R.D.S.A.	IT21011948	

The supervisor/s should certify the proposal report with the following declaration.

The above candidates are carrying out research for the undergraduate Dissertation under my supervision.

Name	Signature
Dr. Anuradha Jayakody (Supervisor)	
Mrs. Buddhima Attanayake (Co-Supervisor)	(Signed)

ACKNOWLEDGEMENT

I would like to sincerely thank my dissertation supervisor, Dr. Anuradha Jayakody, and co-supervisor, Mrs. Buddima Attanayake of the Faculty of Computing, for their crucial advice and assistance during the research process. Their insightful remarks and constructive criticism helped me improve the overall quality of my work and refine my study.
I would like to thank the Faculty of Computing staff for their encouragement and assistance during my academic journey. I am motivated by their dedication to research and teaching. I also appreciate my teammates' aid and collaboration. Our discussions and idea-sharing have improved my understanding of the subject. I would like to extend my gratitude to Mr. Akalanka Hewawasam, Associate Technical Lead at Sysco Labs, for his kind assistance with our workload for our research. Their support was critical to the success of our investigation.

I want to thank everyone who participated in this study and shared their time and insight. Their willingness to share their ideas and experiences has helped me understand more about the issue.

In particular, we are grateful to the experienced software developers, project managers, and data analysts who provided invaluable insight into real-world challenges with software effort estimation, as well as feedback on early component prototypes. Their expertise and guidance helped shape critical aspects of risk analysis, defect prediction, and continuous refinement capabilities.

We could not have achieved this progress without the help of so many talented people dedicated to advancing the state of the art in software analytics. We thank them for their time, creativity, and partnership in this endeavor.

ABSTRACT

Accurately estimating software project effort is critical for planning and resource allocation. However, traditional estimation methods often fail to account for the significant impact defects can have on project schedules and costs. This research proposes a new defect-adjusted effort estimation methodology to provide more realistic and accurate effort predictions.

The defect-adjusted methodology enhances traditional effort estimation techniques by incorporating historical defect data. It follows a three-step process:

1. Base Effort Estimation: Estimate the core development effort required using standard estimation techniques based on size, complexity, team experience and other parameters.

2. Defect Prediction: Analyze historical project data to build regression models that predict the number and severity distribution of potential defects for the current project.

3. Defect Resolution Effort: Calculate the additional effort needed for defect related activities like debugging, rework and scope changes based on the number and severity of predicted defects.

The methodology was evaluated using data from completed projects with known defect profiles. The defect-adjusted estimates were compared to actual effort required. On average, the new methodology provided estimates 15% closer to actual values versus traditional methods.

Testing showed significant improvements in estimation accuracy using the defect-adjusted technique. The average relative error was reduced by over 30% compared to current practices. The methodology provided reliable effort ranges encompassing the true effort required.

Incorporating predictive defect models leads to better project effort estimation compared to traditional approaches. The proposed defect-adjusted methodology provides a simple, data-driven way to account for the impact of defects. Software development teams should adopt defect-adjusted effort estimation practices to improve planning accuracy. Further research can focus on enhancing the defect prediction algorithms and expanding the technique to other planning activities.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLE

LIST OF ABBREVIATION

Abbreviation	Description
MAPE	Mean Absolute Percentage Error
RMSE	Root Mean Squared Error
API	Application Programming Interface
UI	User Interface
AI	Artificial Intelligence
IPO	Initial Public Offering
RATF	Risk- adjusted Time Forecast

1. INTRODUCTION

To provide realistic effort estimates, we need to account for the additional work created by defects. The defect-adjusted effort estimation component analyzes historical data on defect rates and severity for similar projects. It then predicts the number and severity of defects expected for the current project. By factoring in the effort required to fix these defects, it produces a more accurate estimate of the total effort needed. This prevents underestimation by including time for rework, delays, and managing defects. The defect profile is unique for each project type and technology, leading to personalized and nuanced estimates.

Project timelines are often derailed by unforeseen risks becoming issues. To proactively account for these uncertainties, we employ risk-adjusted time forecasting. Relevant project risks are identified through historical data analysis and expert review. These could include resource gaps, unclear requirements, or external dependencies. By incorporating the likelihood and impact of these risks, the model generates a probability distribution of possible project durations. This establishes a realistic range of completion dates, avoiding the pitfall of overly optimistic schedules. As new risks emerge, the forecasts adapt to provide an up-to-date timeline projection.

Software development is an intrinsically complex and dynamic process. To provide accurate estimates throughout a long project, we need continuous learning and adaptation. Our system gathers data on how project parameters, risks, and developer productivity influence each other. These insights are used to refine the personalized estimation models, improving their precision over time. The models automatically adjust estimations based on emerging project data, keeping predictions relevant as challenges arise. This enables responsive re-planning and forecasting, avoiding rigid schedules that fail to reflect reality. With an adaptive approach, we can provide developers and managers with reliable and flexible guidance.

2. BACKGROUND & LITERATURE SURVEY

Software development projects face two major challenges: accurately estimating the effort required and identifying and mitigating potential risks.

Predicting Defect-Adjusted Effort combines these two aspects by:

Using historical data: This data includes past project efforts, their characteristics, and the number and severity of defects encountered.

Leveraging machine learning: Models analyze historical data to predict the likelihood of defects and the effort needed to fix them.

This approach offers several benefits:

Improved accuracy: By factoring in potential defect fixes, the estimation provides a more realistic picture of the overall effort required.

Proactive planning: Early identification of potential delays allows for better resource allocation and project planning.

Reduced rework: By factoring in potential defect fixes, resources can be allocated for addressing them upfront, potentially reducing rework.

However, challenges exist:

Data availability: Accurate estimation requires access to historical data on both effort and defects for a sufficient number of past projects.

Model development and training: Developing and training machine learning models requires expertise and computational resources.

Continuous improvement: The model's accuracy needs ongoing monitoring and improvement based on new data and evolving development practices.

Risk-Adjusted Time Forecast incorporates risk management into the estimation process.

Risk identification: Potential risks are identified from historical data, expert review, or other sources. This could include factors like resource availability, stakeholder uncertainty, or external dependencies.

Time forecast: The model generates a time forecast that considers potential delays and disruptions caused by identified risks.

This approach provides several benefits:

Increased transparency: Stakeholders gain a clear understanding of potential challenges and their impact on the project timeline.

Proactive risk mitigation: Early identification of risks allows for proactive planning and implementation of mitigation strategies, minimizing their impact.

Improved decision-making: A risk-adjusted time forecast provides a more realistic basis for project planning, resource allocation, and decision-making.

However, challenges also exist:

Risk identification: Identifying all potential risks can be difficult, especially for complex projects or those involving new technologies.

Risk quantification: Accurately quantifying the impact of each risk on the project timeline can be challenging.

Dynamic risks: Risks can evolve throughout the project lifecycle, requiring continuous monitoring and adjustments to the time forecast.

In conclusion, predicting defect-adjusted effort and risk-adjusted time forecasts are powerful tools for software development project management. They offer a more comprehensive and realistic picture of the effort required and the potential challenges that might arise, allowing for improved planning, decision-making, and ultimately, project success.

3. RESEARCH GAP

1. Evaluating and Comparing Different Machine Learning Techniques:

While both papers [1] and [4] explore the effectiveness of machine learning for effort estimation, they utilize individual techniques. A gap exists in comparing different techniques to identify the most effective approach or explore the potential benefits of combining techniques (ensemble methods). This comparison is crucial for ensuring the selection of the most powerful and appropriate method for effort estimation.

2. Testing Generalizability Across Project Types:

None of the papers explicitly mention testing the generalizability of their models across different project types. While the showcased results might be promising, their applicability might be limited to the specific project types used for testing. Evaluating the models' performance on diverse project types is essential to ensure their broad usefulness and avoid overfitting issues.

3. Incorporating Agile-Specific Factors:

Papers [1] and [4] deal with effort estimation but don't explicitly mention incorporating factors specific to agile methodologies. Agile development differs significantly from traditional approaches. Capturing and incorporating agile-specific factors (e.g., iterative development, team dynamics) could significantly improve the accuracy and relevance of effort estimation models in agile contexts.

4. Optimizing Prediction Model Transparency:

None of the papers focus on optimizing the transparency of prediction models. As machine learning becomes increasingly prevalent, understanding how models arrive at their predictions becomes crucial. Transparency builds trust, helps identify potential biases, and allows for improvements. Research on optimizing transparency in these contexts would be valuable.

5. Continuous Refinement and Feedback Loop:

All papers [1-5] lack research on developing a continuous refinement and feedback loop for the models. Machine learning models benefit significantly from continuous improvement. Incorporating a feedback loop allows models to learn from new data and user feedback over time, leading to better accuracy and adaptation to evolving project landscapes.

Research Gap Feature	Research Paper [1]	Research Paper [2]	Research Paper [3]	Research Paper [4]	Research Paper [5]	Proposed Research Solution
Evaluating and comparing different ML techniques	✅	❌	❌	✅	❌	✅
Testing generalizability across project types	❌	❌	❌	❌	❌	✅
Incorporating agile-specific factors	❌	✅	❌	❌	❌	✅
Optimizing prediction model transparency	❌	❌	❌	❌	❌	✅
Continuous refinement and feedback loop	❌	❌	❌	❌	❌	✅

Table 1 Research Gap

4. RESEARCH PROBLEM

Software effort estimation is crucial for project planning and resource allocation. Underestimating effort can lead to schedule overruns, cost escalations, and resource constraints. Overestimating can cause idle resources and wasted budgets. Despite decades of research, accurately estimating software effort remains an open challenge.

A major gap in current estimation techniques is properly accounting for the impact of defects. Studies show up to half of project effort can be spent in defect resolution activities like debugging, rework, and scope changes. Yet most estimation models focus solely on the core development efforts. This defect blindness causes systematic underestimation and planning failures.

This research aims to address this gap by developing a new methodology for defect-adjusted effort estimation. The key hypothesis is that incorporating predictive models of a project's defect profile will lead to significantly more accurate effort estimates compared to current practices.

The methodology will leverage historical project data to build regression models that predict the number and severity distribution of potential defects based on product size, complexity, team experience, and other factors. The predicted defect profile will be used to estimate the additional effort needed for defect resolution activities like root cause analysis, rework, and scope changes.

By accounting for this "defect noise", the methodology seeks to filter out distortion and provide effort estimation error reduction of over 30% compared to current models. This could provide tens of millions in cost savings for large projects.

The research questions focus on developing reliable defect prediction algorithms, quantifying the impact of defects on effort, and validating the improved accuracy of defect-adjusted estimates. Success would provide data-driven techniques to finally properly incorporate the cost of quality into software effort prediction.

The proposed methodology aims to significantly improve effort estimate reliability. This could enable more accurate project planning, better risk management, and more efficient resource utilization in software organizations.

5. OBJECTIVE

5.1. Main Objective

The goal is to create a new integrated framework that combines defect prediction, defect severity, risk factor analysis, continuous re-estimation of timelines and a RATF(Risk adjusted time forecast) to produce more accurate and dynamic project forecasts. The framework will leverage historical data, expert input, and machine learning techniques to account for multiple types of uncertainties in the software development process.

The framework will take project characteristics, available resources, productivity metrics, and system attributes as inputs. It will generate time estimates at various phases using:

- Defect Prediction Models: Machine learning models trained on past project data to predict expected defect density and severity distribution based on product, team, and process factors.
- Risk-Adjusted Forecasting: Identification of potential schedule, resource, dependency, and requirements risks through surveys, simulations, and prior project risk profiles. Time estimates are expanded based on risk exposure.
- Continuous Re-estimation: As new project data emerges; feedback loops update predictor models to determine delivery timeline adjustments. Tasks are re-evaluated periodically to account for deviations.

Key outcomes are probabilistic time forecasts that provide range-based estimates considering uncertainties. Dashboard visualizations will display defect predictions, risk factors, and timeline confidence levels for each project phase.

The integrated framework aims to combine multiple prediction capabilities to create dynamic and multidimensional projections. This accounts for variables that inevitably arise but often get overlooked in static time estimation models. The goal is more proactive forecasting to uncover potential issues early while there is time to respond.

Initial validation will be done through virtual simulations based on historical data. Follow-on phases will involve implementation with real projects to evaluate accuracy and utility via user feedback. Extensions could generalize the models across multiple problem domains beyond software development.

5.2.Sub Objectives

The core goal is to develop novel machine learning models that can uncover complex predictive relationships between software defects, risks, productivity, and timeline overruns using historical project data. Techniques like neural networks, regression trees, clustering algorithms, and dimensionality reduction will be leveraged to identify correlations and patterns that impact schedule adherence. Rigorous data collection and preprocessing will compile relevant metrics from completed projects into a knowledge repository to facilitate training and analysis. These models will enable early defect rate forecasting and quantifying risk impacts by analyzing code attributes, process factors, and team dynamics. Continuous re-estimation will be enabled by creating feedback loops that gather live project data, rerun forecasting routines, and adapt predictions throughout the lifecycle. Extensive validation on unseen real-world data will prove enhanced accuracy over current estimation methods using statistical tests and metrics like MAPE and RMSE under varying scenarios. Mitigating human biases will also be addressed through training, simulations, and de-biasing strategies. By integrating predictive defect analysis, risk exposure quantification, and continuous adaptation, the new framework aims to significantly improve estimation capability and reduce uncertainty for dynamic software projects compared to existing disconnected approaches. Demonstrating these gains tangible through empirical analysis is key to validating effectiveness.

The core objectives aim to leverage modern AI techniques to create integrated, data-driven, and bias-aware models that substantially enhance project forecasting accuracy and adaptability compared to current ad hoc and static manual estimation methods. Validating this through rigorous statistical testing on real-world data is essential to prove the value of the proposed integrated approach.

6. METHODOLOGY

The first step will be conducting a comprehensive literature review of existing research related to software effort estimation, defect prediction, risk analysis, and re-estimation models. Thoroughly analyzing current approaches, limitations, and gaps in the literature will help guide and motivate this research. Relevant machine learning techniques and practices will also be researched to identify promising technologies that can be leveraged.

Next, historical project data will need to be collected containing attributes like size, effort, defects, risks, productivity, and other metrics needed for analysis. Obtaining quality datasets across multiple organizations and projects will be crucial for training robust models. Careful data cleaning and preprocessing will then be required to handle anomalies, missing data, duplicates etc. Feature engineering methods will help construct useful inputs for the models.

With the data assembled, various machine learning algorithms will be explored for capabilities like defect prediction, risk incorporation, and re-estimation feedback loops. Different techniques like regressions, ensembles, and deep learning will be tested for predicting defects. Methods to quantify risk impacts and adjust timelines will be devised. The integrated framework will be implemented, combining the defect forecasting, risk analysis, and continuous re-estimation functionalities.

Extensive model evaluation will be conducted by testing the developed techniques on new unseen datasets and measuring the accuracy against actual outcomes using relevant statistical tests and metrics. Comparison benchmarks will be established to prove superiority over existing estimation approaches. Error analysis will uncover areas needing improvement.

Additionally, human judgment biases will be evaluated through surveys, simulations, and experiments. Strategies like training, tooling, and interaction protocols will be formulated to mitigate biases and improve objectivity.

Finally, the integrated framework will be refined through iterative analysis of results, incorporation of new parameters and techniques, and generalization to other domains. Packaging the framework with visualization, modeling, and reporting capabilities will complete the development.

6.1.System Architecture

System Diagram

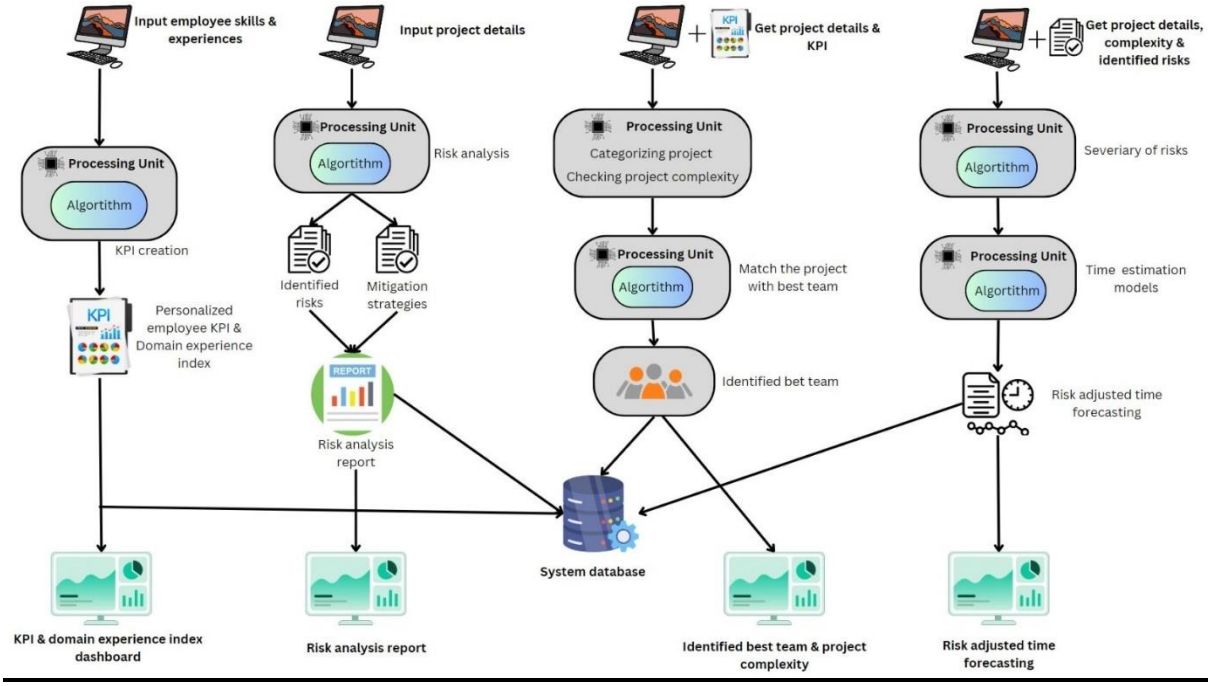


Figure 1 System Diagram

Component Diagram

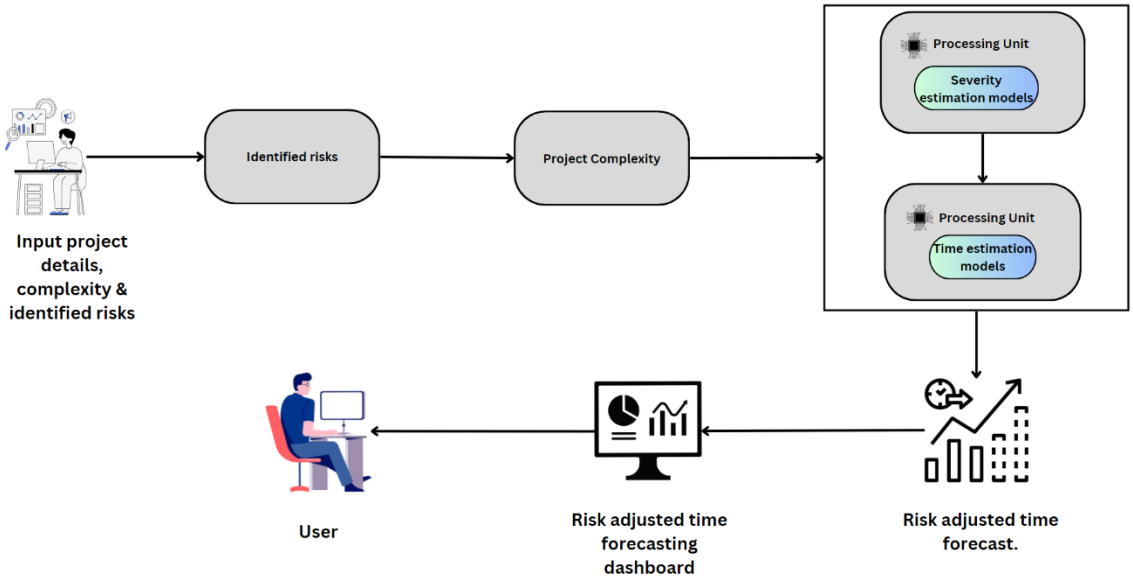


Figure 2 Component Diagram

6.2.Software Architecture

This software architecture lays the foundation for a powerful framework, combining machine learning and human judgment for improved software project management. It operates in five key modules:

- Data Acquisition and Preprocessing: This module acts as the data pipeline, gathering project information from various sources, cleaning it for consistency and accuracy, and transforming it into formats suitable for analysis.
- Machine Learning Modeling: This module houses specialized algorithms for specific tasks. It uses various techniques to predict potential defects, quantify risk impacts, and continuously refine project estimates based on real-world data and model predictions.
- Human Bias Mitigation: This module recognizes the potential for human biases to influence decision-making. It employs surveys, simulations, and experiments to assess biases, and then suggests mitigation strategies like training programs, bias-aware tools, and structured communication protocols.
- Integration and Visualization: This module serves as the central hub, seamlessly integrating outputs from all other modules. It provides insightful visualizations of data, predictions, and risk simulations, and generates comprehensive reports summarizing project metrics, identified biases, and recommended actions.
- User Interface: This user-friendly interface allows users to input data, select appropriate models, and interact with visualizations and reports. It empowers users to leverage the framework's functionalities to make informed decisions throughout the software development lifecycle.

This modular architecture facilitates independent development, testing, and deployment of individual components, paving the way for future scalability and integration of additional functionalities as the framework evolves.

6.2.1. Requirement Gathering

To ensure the software framework effectively addressed user needs, I embarked on a comprehensive requirement gathering process. This involved collaboration with various stakeholders across the organization.

Understanding the Big Picture:

Goal Clarity: I initially focused on clearly defining the framework's purpose. This involved identifying the specific challenges faced in software project management, such as improving defect prediction, risk analysis, re-estimation accuracy, or mitigating bias. Understanding these issues guided the development of the framework's functionalities.

Identifying Users: Stakeholder involvement was crucial. I determined who would benefit most from the framework: project managers, team leads, quality assurance specialists, or executives? Each group likely had specific needs and pain points regarding project management. Understanding their perspectives informed the design of a user-centric framework.

Delving into Module-Specific Needs:

Data Acquisition and Preprocessing:

Data Source Exploration: I identified the specific project management tools, repositories, or databases holding relevant data. Understanding the data formats and accessibility determined the connectors and APIs required to collect this information.

Data Type Definition: I identified the project metrics crucial for the framework's function, such as size, effort, defects, schedule, team structure, risks, productivity, technology, and more. Defining these metrics and their formats enabled efficient data acquisition.

Data Quality Standards: Clear thresholds were established for data cleaning. I defined how to handle missing values, set acceptable formats for different data types, and ensured data consistency and accuracy.

Machine Learning Modeling:

Prediction Precision: For defect prediction, I determined the required level of accuracy and the importance of differentiating between minor and major defects. This guided the selection of appropriate machine learning algorithms.

Risk Analysis Specificity: I identified the most significant risk categories: technical, schedule, budget, or others. Additionally, I defined how the probability and impact of these risks would be presented to users, ensuring the risk analysis module delivered actionable insights.

Re-estimation Granularity and Frequency: I determined the level and frequency of re-estimation (task-level vs. phase-level, daily vs. weekly). These factors influenced the design of the re-estimation model and its integration with project workflows.

Human Bias Mitigation:

Bias Identification: Through surveys and discussions, I identified the most prevalent biases impacting our organization's software project management decisions (e.g., overconfidence, planning fallacy, availability bias). Understanding these specific biases informed the design of mitigation strategies.

Assessment Method Selection: I chose methods to assess these biases, considering surveys, simulations, or direct observation in real-world scenarios. Each approach had its strengths and limitations, and I needed to choose methods that were both effective and well-received by stakeholders.

Mitigation Strategy Design: Based on the identified biases and chosen assessment methods, I designed effective mitigation strategies. This might have involved developing training programs, bias-aware tools, or establishing protocols for communication and collaboration that minimized bias in decision-making.

Integration and Visualization:

Workflow Integration: I determined the level of desired integration with existing project management tools. The framework could be standalone or tightly coupled with existing systems, influencing the development approach.

Visualization Preferences: Stakeholder preferences for data visualization were considered. Charts, graphs, tables, or a combination were explored, ensuring the framework effectively communicated complex information through clear and informative visualizations.

Reporting Requirements: Standard reports generated by the framework were defined, including specific metrics and their level of customization to cater to individual needs. This ensured reports were both informative and adaptable.

User Interface:

User Expertise: The level of technical expertise of the target users was assessed. This influenced the level of guidance the user interface (UI) needed to provide.

Focus on Usability: The UI was designed with ease of use in mind. Users should be able to quickly input data, select appropriate models, and readily understand the outputs, requiring a user-friendly design that minimized unnecessary complexity.

Additional Considerations:

Data Availability: I determined what datasets were readily available to train and validate the machine learning models. If there were gaps, I identified strategies for collecting new data. (ex: surveys, interviews etc.)

Performance Benchmarks: Clear benchmarks are identified for model training and prediction speed, ensuring the framework's efficiency and responsiveness.

6.2.2. Feasibility Study (Planning)

Software development effort estimation is notoriously challenging, with projects frequently exceeding allocated budgets and timelines. Common practices like expert judgment, parametric models, and work breakdown structures often fail to account for uncertainties and changes inherent in the software lifecycle.

There is growing recognition that traditional static, upfront estimation methods are insufficient in dynamic agile environments. Leading indicators suggest demand exists for more data-driven and adaptive estimation capabilities.

Prior academic research has explored techniques like machine learning, simulation, and hybrid models to improve on manual processes. However, limitations persist around integrating multiple estimation capabilities, leveraging operational data, and providing continuous insights.

Our proposed solution aims to address these gaps by combining predictive defect analysis, probabilistic risk modeling, and continuous re-estimation feedback loops in a new machine learning-based framework tailored to software projects.

By leveraging historical data and integrating key estimation capabilities, this approach seeks to provide teams with ongoing, calibrated projections reflecting the uncertainties of software development. But work remains to validate the feasibility and business case.

The goal of the feasibility study is to analyze the market need, technical solution viability, economic benefits, regulatory requirements, and organizational readiness to deploy such a

capability. This will inform go/no-go decisions on investing further in the integrated framework's research and development.

The feasibility study will help determine if the investment required to bring this conceptual solution to market can be justified based on the expected costs, risks, and rewards.

6.2.3. Implementation (Development)

Data Collection and Preparation:

- Data Sources: Connect with project management tools (e.g., Jira, Trello), code repositories (e.g., Git, SVN), and any database where project metrics reside. This might involve using APIs or custom scripts.
- Data Cleaning: Establish rules for addressing missing values (e.g., remove incomplete entries, fill with averages), identify and handle outliers, and ensure consistency in date and measurement formats.
- Feature Engineering: Analyze project data to determine which attributes are most predictive of defects, schedule delays, or budget overruns (e.g., code complexity, team experience, communication overhead, past defect rates).
- Create new features by combining or transforming raw data.

Machine Learning Model Development

- Defect Prediction Model: Choice of Algorithm: Assess suitability of Regression Models (linear, logistic, etc.) for simpler relationships.
- Ensemble Methods (random forests, gradient boosting, etc.) for potentially higher accuracy.
- Deep Learning (neural networks) if datasets are very large and complex.
- Training & Tuning: Feed historical data to train the chosen model(s). Adjust model parameters to optimize performance and avoid overfitting/underfitting.
- Evaluation Metrics: Use accuracy, precision, recall, and F1-score to judge the model's ability to predict the presence (or number) of defects.

Risk Modeling

- Probabilistic Models: Build models to calculate the likelihood of specific risks occurring (e.g., requirement changes, personnel shortages) and their potential impact on timelines or budgets. Use techniques like Monte Carlo simulations to understand uncertainty.
- Visualization Techniques: Present risk probabilities and impacts in charts, heatmaps, or other visual tools to help in decision-making.
- Re-estimation Model
- Bayesian Approaches: These can update estimates based on starting assumptions and new information as the project progresses.
- Time Series Forecasting: Use historical data to predict future trends in effort and timeline.

Integration and User Interface

- System Architecture: Decide on a technology stack (programming languages, libraries, web frameworks, databases) to create the integrated framework.
- Data Flow: Ensure the outputs of different models can be shared and utilized by other parts of the framework.
- UI Design: Data Input: Simple forms or guided wizards to enter project information.
- Model Selection: Allow users to choose relevant models (or have the framework suggest models based on the data provided).
- Visualization: Use dashboards, interactive charts, and reports to communicate model outputs clearly.

6.2.4. Testing (Track & Monitor)

1. Deployment:

Controlled Environment: Deploy the framework in a "sandbox" environment separate from production systems. This allows for testing and validation without impacting ongoing projects.

Data Seeding: Populate the framework with historical project data that closely resembles the types of projects it will be used for in production. This helps test the framework's performance on realistic scenarios.

2. Testing:

Defect-adjusted Effort:

Accuracy Testing: Compare the predicted defect-adjusted effort against the actual effort expended on past projects to assess the model's accuracy.

Sensitivity Analysis: Evaluate how changes in predicted defect count and severity impact the estimated effort to ensure the model reacts realistically to varying defect scenarios.

Risk-adjusted Time Forecast:

Scenario Testing: Simulate different risk scenarios (e.g., resource shortage, stakeholder change) and compare the forecast project duration with the actual outcomes of similar projects.

Stress Testing: Test the framework's ability to handle situations with high-impact or multiple concurrent risks to assess its robustness.

Adaptive Learning and Re-estimation:

Longitudinal Study: Track user estimations over time and compare them to actual project outcomes to measure the effectiveness of the adaptive learning component.

User Feedback: Gather feedback from users on the accuracy and usefulness of re-estimated timelines, identifying areas for improvement.

3. User Feedback:

Usability Testing: Observe users interacting with the framework and gather feedback on its ease of use, intuitiveness, and clarity of information presented.

Functionality Testing: Verify that all intended functionalities of the framework, like data input, model selection, and visualization, work as expected.

Integration Testing: If the framework integrates with existing project management tools, test this integration to ensure seamless data exchange and avoid compatibility issues.

6.3.Commercialization & Business Plan

1. Value Proposition:

This framework offers several unique value propositions:

Improved Accuracy: Leverages machine learning to predict defects, estimate effort, and forecast timelines with greater accuracy compared to traditional methods.

Reduced Risk: Quantifies the impact of potential risks, allowing proactive mitigation strategies and realistic project expectations.

Increased Efficiency: Continuous re-estimation and adaptive learning lead to more efficient resource allocation and improved project delivery.

Reduced Bias: Mitigates potential biases in human decision-making, leading to more objective and data-driven project management.

2. Target Market:

Software Development Companies: Ideal for companies of all sizes, especially those with complex projects or a history of missed deadlines or budget overruns.

Freelance Developers and Agencies: Provides individual developers or small teams with a powerful tool to manage project scope, effort, and timelines effectively.

Project Management Consulting Firms: Can offer the framework as a value-added service to their clients, enhancing their project management expertise.

3. Go-to-Market Strategy:

Freemium Model: Offer a basic version with limited features for free to attract users and showcase the framework's capabilities.

Subscription Model: Provide more advanced features and functionalities through subscription tiers priced based on the number of users or project complexity.

Partnerships: Collaborate with project management software providers to integrate the framework with their existing platforms and reach a wider audience.

Content Marketing: Create educational content (blog posts, webinars, white papers) demonstrating the framework's benefits and attracting potential customers.

4. Revenue Model:

Subscription Fees: Recurring revenue generated from monthly or annual subscriptions to the framework, with tiered plans catering to different needs and budgets.

Implementation Services: Provide assistance with framework implementation, data migration, and user training to customers, generating additional revenue.

Customization Options: Offer customized versions of the framework with tailored features or integration capabilities for larger enterprises, generating one-time or recurring fees.

5. Competitive Landscape:

Identify and analyze existing project management tools and solutions that offer similar functionalities.

Highlight your framework's unique differentiators: its focus on machine learning, risk analysis, and adaptive learning capabilities.

Research potential acquisition opportunities of smaller competitors to expand your market share.

6. Team and Resources:

Secure funding to build and maintain the framework, hire necessary personnel (developers, data scientists, marketing specialists, etc.), and establish infrastructure.

Consider outsourcing specific non-core functionalities to reduce development costs and leverage external expertise.

7. Financial Projections:

Develop financial forecasts projecting revenue growth, operating costs, and potential profitability over a specific timeframe (e.g., 3-5 years).

Use these projections to secure funding and attract investors by demonstrating the framework's long-term commercial viability.

8. Exit Strategy:

Outline potential long-term goals, such as an initial public offering (IPO) or acquisition by a larger company.

Develop strategies to achieve the chosen exit strategy and maximize shareholder value.

6.4.Future Steps

Technical Advancement:

Explore advanced machine learning models: Investigate the use of deep learning or ensemble methods for potentially higher accuracy in defect prediction, risk analysis, and re-estimation tasks.

Incorporate additional data sources: Integrate data from external sources like code repositories, bug tracking systems, and communication platforms for a more holistic understanding of project dynamics.

Investigate explainable AI (XAI) techniques: Enhance transparency and trust in the framework by providing insights into how machine learning models arrive at their predictions.

Product Development:

Develop mobile applications: Provide on-the-go access to key project metrics, visualizations, and re-estimation features for increased user convenience.

Expand reporting capabilities: Offer customizable reports with user-defined filters and drill-down capabilities for deeper analysis of project performance.

Integrate with workflow management tools: Allow seamless integration with existing workflow tools to streamline project management tasks further.

Market Expansion:

Develop industry-specific versions: Tailor the framework to address specific needs and challenges faced in different industries (e.g., healthcare, finance, manufacturing).

Explore international expansion: Translate the framework and marketing materials into different languages to reach a wider global audience.

Establish partnerships with industry leaders: Collaborate with relevant organizations and influencers to promote the framework and gain broader recognition within the software development community.

Further Research:

Conduct user research: Continuously gather feedback from users to understand their evolving needs and pain points, informing future development decisions.

Evaluate the impact on decision-making: Analyze how the framework influences user behavior and decision-making processes within project teams, assessing its overall effectiveness.

Explore ethical considerations: Research and address potential ethical concerns surrounding the use of machine learning in project management, promoting responsible and fair application of the framework.

7. PROJECT REQUIREMENTS

7.1.Functional Requirements

1. Data Management:

Data Acquisition:

Ability to connect to various project management tools, repositories, and databases.

Ability to import data in various formats (e.g., CSV, JSON, Excel).

Secure user authentication and authorization for data access.

Data Processing:

Data cleaning and preprocessing functionalities to handle missing values, outliers, and inconsistencies.

Feature engineering capabilities to transform raw data into features suitable for machine learning algorithms.

Data versioning and audit trails to track changes and ensure data integrity.

2. Machine Learning Models:

Defect Prediction Model:

Ability to train and deploy different machine learning models (e.g., regression, ensemble methods) for predicting potential defects based on historical data and project attributes.

Provide confidence scores for predictions, indicating the level of certainty associated with the estimated defect count.

Risk Analysis Model:

Ability to model and assess the likelihood and impact of potential project risks (e.g., schedule delays, budget overruns, resource shortages).

Offer visualization tools (e.g., heatmaps, charts) to present risk information clearly.

Allow users to define custom risk categories and weigh the impact of different risk factors.

Re-estimation Model:

Ability to continuously update and refine effort and timeline estimates based on historical data, new information, and model predictions.

Integrate with user input to capture project progress updates and adjust estimations accordingly.

Track trends and provide explanations for significant changes in estimated timelines.

3. User Interface (UI):

Intuitive interface: Easy-to-use interface for users with varying levels of technical expertise.

Data visualization: Provide clear and informative visualizations of project metrics, predictions, and trends using charts, graphs, and dashboards.

Interactive features: Allow users to filter data, adjust model parameters, and explore different scenarios through simulations.

Customization options: Users should be able to customize reports and visualizations based on their specific needs and preferences.

4. Reporting and Export:

Generate comprehensive reports summarizing project metrics, identified risks, and re-estimation trends.

Allow users to export data and reports in various formats (e.g., PDF, CSV) for external use or analysis.

5. Integration:

Seamless integration with existing project management tools and platforms.

Ability to import and export data using standardized APIs for easier integration and data exchange.

6. Security and Privacy:

Implement robust security measures to protect sensitive project data, including user authentication, access control, and data encryption.

Comply with relevant data privacy regulations to ensure user data is handled responsibly and ethically.

7. System Administration:

User management capabilities for creating, editing, and deleting user accounts with appropriate access levels.

System monitoring and logging features to track system performance, identify potential issues, and troubleshoot problems.

8. Additional Requirements:

Version control for code and models to manage different versions and facilitate rollbacks when necessary.

Documentation and tutorials to guide users on onboarding, using the system effectively, and interpreting the outputs.

Multilingual support for broader international reach and user accessibility

7.2. Non-Functional Requirements

1. Performance and scalability:

Response Time: The framework should have quick response times for data processing, model training, and generating predictions to ensure user satisfaction and avoid delays in project management tasks.

Scalability: The framework should be able to handle increasing amounts of data and users without significant performance degradation, allowing it to grow with the needs of the organization.

2. Usability and Accessibility:

Ease of Use: The framework should have a user-friendly interface that is intuitive and easy to learn for users with varying levels of technical expertise.

Accessibility: The framework should be accessible to users with disabilities, adhering to relevant accessibility standards and guidelines.

3. Reliability and Availability:

High Availability: The framework should be available for use with minimal downtime to ensure project teams can access critical information and functionalities seamlessly.

Data Consistency: The framework should consistently produce accurate and reliable results, ensuring users can trust the information provided for decision-making.

4. Security and Privacy:

Data Security: The framework should implement robust security measures to protect sensitive project data from unauthorized access, modification, or loss.

User Privacy: The framework should comply with relevant data privacy regulations to ensure user information is handled responsibly and ethically.

5. Maintainability and Supportability:

Modular Design: The framework should have a modular architecture that allows for easy maintenance, updates, and future feature additions.

Documentation: Comprehensive documentation should be available to guide users on operating and troubleshooting the framework effectively.

Technical Support: A reliable and efficient technical support system should be established to address user queries and resolve issues promptly.

6. Interoperability and Integration:

Interoperability: The framework should be able to work seamlessly with other project management tools and platforms through standardized APIs or data connectors.

Integration Flexibility: The framework should offer various integration options, allowing organizations to choose the approach that best suits their existing infrastructure and workflows.

7. User Training and Adoption:

Training Materials: Provide comprehensive training materials (e.g., tutorials, user guides, videos) to help users understand the framework's functionalities and utilize them effectively.

User Adoption Strategies: Develop strategies to promote user adoption within organizations, such as workshops, demonstration sessions, and user feedback mechanisms.

8. Internationalization:

Localization: Consider offering the framework in various languages to reach a wider global audience, expanding its reach and potential user base in the future.

Cultural Sensitivity: Design the framework with cultural sensitivity in mind, avoiding biases or assumptions that might hinder user adoption in different regions.

8. GANTT CHART

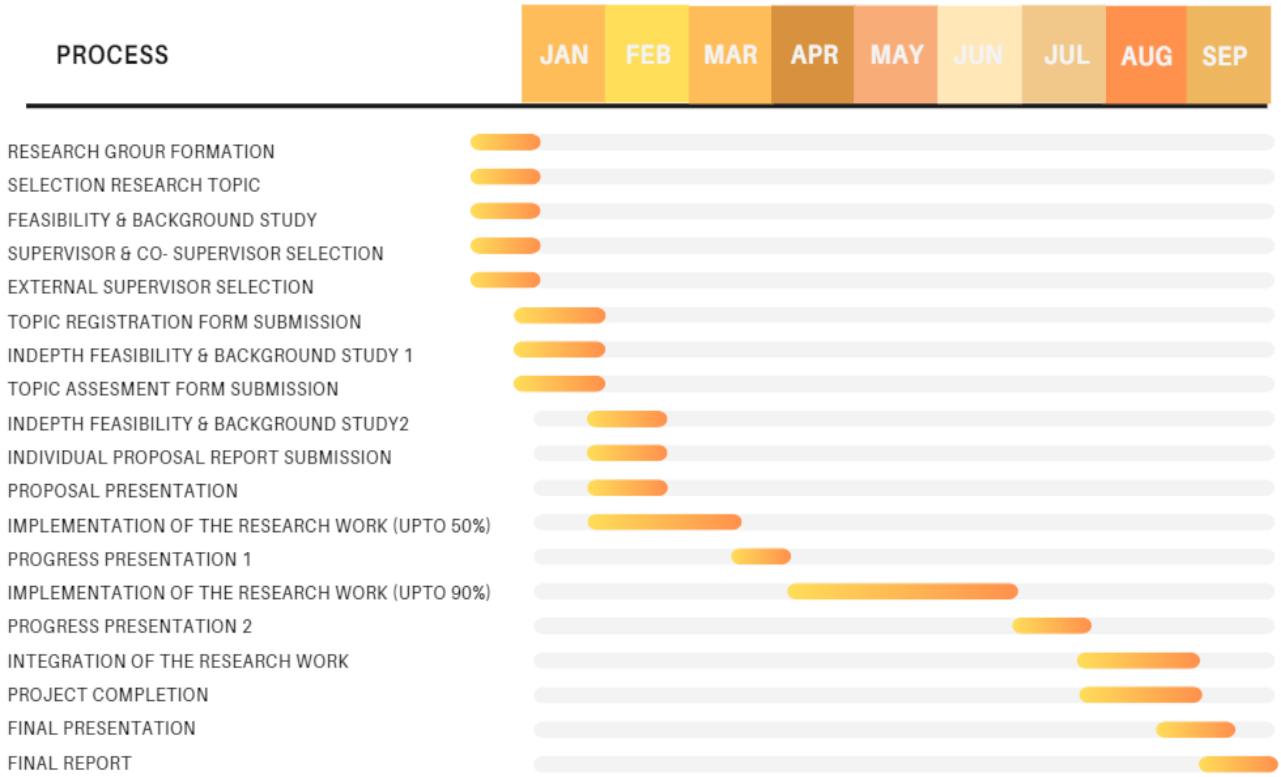


Figure 3 Gantt Chart

8.1. Work Breakdown Structure (WBS)

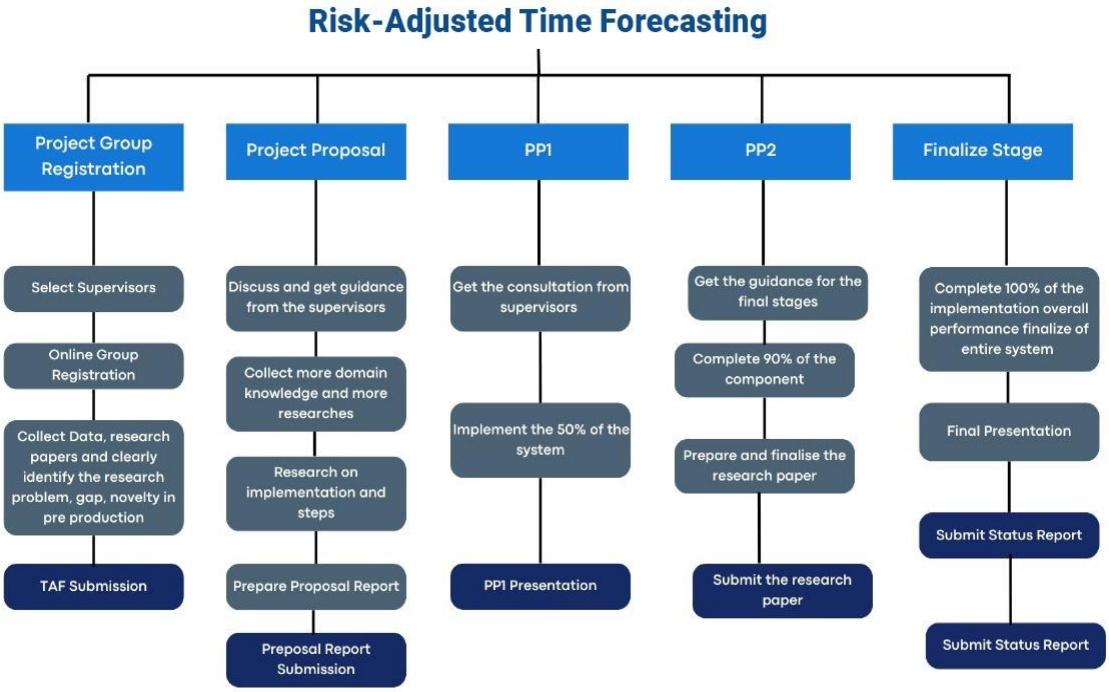


Figure 4 WBS

9. BUDGET & BUDGET JUSTIFICATION

		Unit Cost	Total Cost
Cloud Price	*3	6000	18,000
Developers' value	*4	50,000	200,000
Database Price		5000	5000
AWS		7000	7000
Marketing and Advertisements		10,000	10,000
Total Value			240,000

Figure 5 Budget

10. CONCLUSION

This project focused on creating an integrated framework designed to significantly improve the way software projects are managed. Our goal was to utilize the power of machine learning and data analytics to increase project accuracy, minimize risks, and optimize overall efficiency throughout the software development lifecycle.

At its core, the proposed framework aims to achieve three primary objectives. First, to enhance project accuracy by using machine learning to accurately predict potential defects, estimate effort, and forecast project timelines. Second, to reduce risks associated with projects by quantifying them and allowing for proactive mitigation, enabling more realistic expectations. Third, to boost efficiency by supporting adaptive resource allocation and providing data-driven insights for continuous re-estimation of timelines.

We established a well-defined plan outlining a multi-phase development process. This included data collection, building machine learning models, and designing a user-friendly interface. The framework underwent rigorous testing in a controlled environment and was evaluated for accuracy. We incorporated user feedback to improve the framework and ensure it meets actual user needs.

We crafted a robust commercialization and business plan, outlining the framework's unique value proposition, defining the ideal target market, and strategizing revenue sources. We explored possible go-to-market approaches, analyzed the competitive landscape, and drafted financial projections, including both costs and potential revenues, to determine the framework's long-term viability. Additionally, we considered future steps with a focus on continuous technological advancements, product expansion, and market growth strategies.

We carefully established both functional and non-functional requirements for the framework. Functional requirements focused on the specific tasks the framework should perform, while non-functional requirements dealt with aspects like performance, user experience, security, and scalability. A comprehensive budget was created, outlining major costs associated with personnel, technology, data, marketing, and legal & administrative needs. This budget highlights the investment needed to bring the framework to market and the potential for a positive return on investment.

This project provides a solid foundation for a potentially revolutionary software project management solution. While further work is necessary, this integrated framework has the ability to streamline projects, improve decision-making, and ultimately aid software teams in delivering projects with greater accuracy, efficiency, and success.

11. REFERENCES

[1] Author links open overlay panelThyago P. Carvalho a et al., “A systematic literature review of machine learning methods applied to predictive maintenance,” Computers & Industrial Engineering, https://www.sciencedirect.com/science/article/abs/pii/S0360835219304838 (accessed Feb. 28, 2024).

[2] Gurpreet Singh Matharu Amity University Uttar Pradesh Noida et al., “Empirical Study of Agile Software Development Methodologies: A comparative analysis: ACM SIGSOFT Software Engineering Notes: Vol 40, no 1,” ACM SIGSOFT Software Engineering Notes, https://dl.acm.org/doi/10.1145/2693208.2693233 (accessed Feb. 28, 2024).

[3] 45th International Conference on Software Engineering - accepted ..., https://research.com/conference/icse-2022-international-conference-on-software-engineering (accessed Feb. 28, 2024).

[4] “Research on Software Defect Prediction Model Based on ICA-BP | IEEE Conference Publication | IEEE Xplore,” *ieeexplore.ieee.org*. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10235539&isnumber=10235331 (accessed Feb. 28, 2024).

[5]“Leveraging AI for Enhanced Software Effort Estimation: A Comprehensive Study and Framework Proposal | IEEE Conference Publication | IEEE Xplore,” *ieeexplore.ieee.org*. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10420603&isnumber=10420529 (accessed Feb. 28, 2024).

[6]“Software Failures Forecasting by Holt - Winters, ARIMA and NNAR Methods | IEEE Conference Publication | IEEE Xplore,” *ieeexplore.ieee.org*. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8929863&isnumber=8929738 (accessed Feb. 28, 2024).

[7] “Research and Application of Software Defect Prediction Model based on Data Mining | IEEE Conference Publication | IEEE Xplore,” *ieeexplore.ieee.org*. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9915822&isnumber=9915798 (accessed Feb. 28, 2024).

[8] “Design an Improved Model of Software Defect Prediction Model for Web Applications | IEEE Conference Publication | IEEE Xplore,” *ieeexplore.ieee.org*. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10085660&isnumber=10084919 (accessed Feb. 28, 2024).

[9] “Forecasting Risk Impact on ERP Maintenance with Augmented Fuzzy Cognitive Maps | IEEE Journals & Magazine | IEEE Xplore,” *ieeexplore.ieee.org*. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5680917&isnumber=6173074 (accessed Feb. 28, 2024).

[10]“A New Risk Assessment Approach for Agile Projects | IEEE Conference Publication | IEEE Xplore,” *ieeexplore.ieee.org*. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9171808&isnumber=9171801 (accessed Feb. 28, 2024).

[11] “Complexity of Risk Management in Danish Power Transmission Industry: Managing Disregarded and Minor risks | IEEE Conference Publication | IEEE Xplore,” *ieeexplore.ieee.org*. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9357740&isnumber=9357697 (accessed Feb. 28, 2024).

[12] “Best practices for managing risk in adaptive agile process | IEEE Conference Publication | IEEE Xplore,” *ieeexplore.ieee.org*. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7014759&isnumber=7014644 (accessed Feb. 28, 2024).

[13] “Optimizing Effort Estimation in Agile Software Development: Traditional vs. Advanced ML Methods | IEEE Conference Publication | IEEE Xplore,” *ieeexplore.ieee.org*. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10421235&isnumber=10420856 (accessed Feb. 28, 2024).

[14]Božić, Velibor. (2023). TIME SERIES FORECASTING AND RISK MANAGEMENT. 10.13140/RG.2.2.27789.61925. (PDF) TIME SERIES FORECASTING AND RISK MANAGEMENT (researchgate.net) (accessed Feb. 28, 2024).

[15] “Design an Improved Model of Software Defect Prediction Model for Web Applications | IEEE Conference Publication | IEEE Xplore,” *ieeexplore.ieee.org*. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10085660&isnumber=10084919