Keywords
Software Project Delay, Cost Estimation, Goal-Question-Metric, Artificial Intelligence, Machine Learning, Project Management, Integrated Delay-Cost Equation.

Authors

Ahmad Abdullah Alghamdi[1*]
General Directorate of Health Services, Ministry of Defense, Saudi Arabia.
ORCID: 0009-0009-3390-4149
E-mail: ahmad.a.alghamdi@modhs.med.sa

Reem Alnanih[2]
Department of Computer Science, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia, Software Engineering and Distributed System Research Group, King Abdulaziz University, Jeddah 21589, Saudi Arabia.
ORCID: 0000-0002-2428-0356
E-mail: ralnanih@kau.edu.sa

Received:29-05-2026
Revised:27-06-2026
Accepted: 04-07-2026
Doi:10.1922/ejprd.v34i5s.1569

GQM's AI-enhanced framework for proactive forecasting and financial assessment of software project delays

Abstract
Software project delays remain a persistent source of cost escalation, delivery uncertainty, and managerial risk, particularly in dynamic environments where requirements, resources, dependencies, and stakeholder decisions evolve throughout the project lifecycle. Conventional estimation approaches often rely on static assumptions and therefore provide limited support for early delay detection or financial impact assessment. This study develops and validates an artificial intelligence-enhanced Goal-Question-Metric framework, referred to as AI-GQM, for proactive forecasting and financial quantification of software project delays. The framework integrates a dynamic GQM-D measurement structure, machine learning-based prediction, and an Integrated Delay-Cost Equation to convert delay indicators into a financially interpretable risk estimate. A positivist quantitative design was adopted using three empirical sources: a global dataset of 10,000 completed software projects for model training, a separate Saudi sample of 52 completed projects for unseen contextual testing, and survey responses from 111 practitioners for indicator validation and weighting. Random Forest, Long Short-Term Memory, and XGBoost models were evaluated within the proposed framework. The AI-GQM framework achieved 92.5% predictive accuracy and demonstrated strong explanatory power for cost overrun estimation ($R^2 = 0.892$). The Integrated Delay-Cost Equation further supported proactive financial risk interpretation by linking delay severity, inter-factor dependency, prediction outputs, and project-specific risk adjustments. The findings indicate that AI-GQM can improve delay forecasting, cost-risk visibility, and decision support in software project management.

INTRODUCTION

Software projects are central to digital transformation because they enable organizations to modernize services, automate operations, improve decision-making, and deliver technology-driven value. Despite their strategic importance, many software projects continue to experience schedule delays and cost overruns that weaken delivery performance, reduce stakeholder confidence, and compromise the expected value of project outcomes (Choetkiertikul et al., 2015; Ibraigheeth & Fadzli, 2019). Cost estimation remains a persistent challenge in software project management because managers must forecast budgets, schedules, resources, and effort under conditions of uncertainty. This challenge is amplified in global software development environments, where distributed teams, coordination complexity, diverse work practices, and changing project conditions increase the likelihood of estimation inaccuracies (Khan et al., 2021).

Traditional project management and cost-estimation approaches, including algorithmic models such as COCOMO and expert-judgment-based methods, provide useful baseline estimates but often depend on static assumptions, predefined parameters, and historical analogies (Alshammari, 2022). These approaches are therefore limited in their ability to capture the dynamic and non-linear interactions that characterize contemporary software projects. Volatile requirements, evolving technologies, governance complexity, stakeholder dependency, team productivity, and organizational constraints may interact throughout the project lifecycle, producing delays that are difficult to detect early and costly to quantify after they occur. Consequently, many organizations lack a systematic mechanism for anticipating emerging delays and translating them into measurable financial consequences (Garg & Rastogi, 2022).

Software cost estimation is particularly complex because it requires multiple uncertain and continuously changing variables to be incorporated into a single forecast. These variables include requirement changes, technical complexity, project scope, team productivity, quality expectations, and evolving risk conditions. Estimators frequently work with incomplete data, subjective assumptions, and fluctuating project circumstances, requiring continuous adjustment as new information becomes available (Ba'abbad & Qureshi, 2021). In practice, estimation accuracy is further hindered by unclear requirements, evolving scope, complex task structures, stakeholder-related delays, heterogeneous team composition, limited historical data, and inconsistent coordination between technical and managerial roles. These factors reduce the reliability of conventional estimation practices and increase the probability of cost overruns and schedule deviations (Alenezi, 2022; Alhumam, 2025; Alturki & E-AMIN, 2025).

Although extensive research has examined software project delays and cost-estimation inaccuracy, important limitations remain. Many studies identify delay factors descriptively but do not integrate them into predictive and financially interpretable models (Alghamdi et al., 2024; Alotaibi et al., 2016; Binu, 2025; Kula et al., 2023). Conventional estimation approaches often fail to account for dynamic project conditions and evolving delay patterns, while complex machine-learning models may provide strong predictive performance but remain difficult to interpret without dedicated explainability techniques (Lundberg & Lee, 2017).These limitations indicate the need for an integrated, adaptive, and explainable framework that can measure delay factors, predict their likely impact, and quantify their financial consequences in a form that is useful for project managers and decision-makers.

This study addresses this gap by developing and validating an AI-enhanced Goal-Question-Metric (AI-GQM) framework for proactively measuring, predicting, and financially quantifying software project delays. The proposed framework integrates the systematic measurement logic of the Goal-Question-Metric paradigm with the predictive capability of artificial intelligence. It is operationalized through a dynamic measurement model, referred to as GQM-D, an Integrated Delay-Cost Equation (IDCE), and a software-based decision-support tool, the Delay Impact Analyzer (DIA). Together, these components provide a delay-aware and cost-sensitive approach for supporting project managers and project management offices in identifying emerging risks, estimating their likely financial effects, and prioritizing corrective interventions.

Methodologically, the study adopts a positivist, quantitative design based on three distinct empirical sources. A global dataset of 10,000 completed software projects obtained from Zenodo (Zenodo, 2026) was used for machine learning training. A 52-project Saudi context-specific sample was reserved for unseen testing, contextual validation, and practical evaluation. In addition, responses from 111 practitioners were used to validate the GQM-D indicators and support the derivation of weighting factors used in the IDCE. This separation between training data, testing data, and survey-based validation strengthens the methodological rigor of the study and reduces the risk of data leakage or overfitting.

The current study has four main aims. First, it examines the nature and extent of schedule delays and cost overruns in software projects by using the 10,000-project global dataset for model training and the 52-project Saudi context-specific sample for unseen testing and practical validation. Second, it re-engineers the traditional GQM paradigm into a dynamic measurement framework capable of defining and operationalizing delay-related metrics that evolve across the project lifecycle. Third, it formulates and validates the IDCE as a mathematical model that links delay metrics, correlation terms, AI-predicted delay values, and project-specific risk adjustments to a composite delay-cost index. Finally, it designs, implements, and empirically evaluates the integrated AI-GQM model and the DIA tool by assessing their performance in predicting delay risk and cost overruns relative to traditional estimation approaches.

1. RELATED WORK

2.1 Understanding Delay in Software Projects

Project delays represent a form of failure affecting both timeliness and cost, as operational expenses increase with project duration. In software projects, delay refers to the

failure to complete work on schedule, with detrimental implications for cost prediction, making it a critical discussion point. Numerous studies have examined the causes and consequences of software project delays, highlighting a range of technical, managerial, and contextual factors. Binu (2025) identified uncertainty, poor resource management, communication gaps, and optimism bias as key delay factors in several countries, although their study lacked a predictive costing model and a Saudi Arabian context.

Alghamdi et al. (2024) offered empirical insights specific to Saudi Arabia, finding that schedules, teamwork, and timeline planning are critical to project success or failure, but their work remained descriptive without providing a formal costing equation. Kula et al. (2023) presented a multifactor framework linking organizational and technical issues to delivery delays, but they did not focus on cost estimation or adapt the model to Saudi Arabia. Abdalkareem et al. (2021) demonstrated that automating commitment clearance can improve development efficiency, but they did not incorporate costing or budgeting models.

Alotaibi et al. (2016) identified the limited use of tools as a major problem in Saudi project management, but it lacked a cost estimation framework and compatibility with modern DevOps or dynamic environments. Finally, a comprehensive study by Kula et al. (2022) demonstrated the correlation between team size, technical factors, and delay drivers, but it was unable to translate these findings into a dynamic or adaptive cost estimation equation. Overall, published studies reveal a consensus on the main delay factors, but they consistently lack predictive cost estimation models, particularly those specifically designed for the Saudi Arabian software project context.

These findings show that these studies collectively demonstrate that delays are multi-dimensional phenomena arising from the interaction of technical, organizational, and human factors, rather than isolated events. However, they also reveal a significant gap: while factors are well-documented, integrated predictive frameworks remain underdeveloped.

2.2 The GQM approach application multiple domains

The Goal-Question-Metric (GQM) approach has been applied in multiple fields to support systematic measurement and decision-making. In the area of requirements quality, Timoshchuk (2020) used GQM to identify the goals, questions, and measures needed for requirements clarity and completeness, often using natural language processing (NLP) tools, but they did not link requirements quality to cost estimation or predictive modeling. Mansoor et al. (2024) applied the Total Quality Management (GQM) methodology to measure process and delivery maturity, including expanded GQM approaches based on the Capability Maturity Model (CMM) for academic and delivery systems; however, their focus remained on quality rather than cost integration. Regarding agile methodologies and project tracking, Boerman et al. (2015), Kārkliņa and Pirta (2018), and Aldahmash and Gravell (2018) used GQM methodology to track progress and performance indicators in dynamic Agile environments, supporting monitoring and decision-making without cost estimation or providing financial estimation models.

In the field of cybersecurity risk and adaptive systems, Calvo and Beltrán (2024) developed dynamic risk assessment metrics, incorporating adaptability into measurement systems, but limited their scope to risk without cost or budgetary applications. Lehrig et al. (2015) identified scalability, flexibility, and efficiency metrics to improve cloud computing performance, although these were not linked to cost estimation or financial dimensions. Farina et al. (2022) used GQM methodology for process recommendation and pattern selection in recommendation systems, enhancing decision support but remaining conceptual without predictive or mathematical models. Finally, Leme et al. (2024) expanded the scope of the GQM model to include human factors by measuring developer mental health and productivity, but they did not link productivity measures to cost estimation equations.

Taken together, these studies demonstrate the versatility of the GQM model across areas such as requirements, processes, agile methodologies, cybersecurity risks, cloud computing, recommendation systems, and mental health. However, they consistently lack integration with cost estimation, predictive modeling, or financial dimensions, a gap that motivates the design of the dynamic AI model of the GQM adopted in this study.

2.3 Comparative Analysis of Key GQM Studies

The application of the Goal-Question-Metric (GQM) paradigm in software engineering research has been diverse, ranging from quality assessment to predictive modeling. Studies like that of Alghamdi et al. (2025) were pioneering in applying GQM to the specific context of delay and cost estimation in the Saudi software industry. Their quantitative survey successfully demonstrated GQM's effectiveness in linking delay factors to estimation errors. However, the study's reliance on static, retrospectively defined metrics meant its design was primarily descriptive, lacking the capability for dynamic or predictive analysis. This limitation is common in early GQM applications, where the framework is used more for post-mortem analysis than for proactive project control.

In contrast, other researchers have applied GQM to highly specialized domains. For instance, Liu et al. (2022) developed a GQM-based metric instrument to assess tangible and intangible cultural heritage content in commercial video games. Similarly, Shojaeshafiei (2020) developed a hierarchical GQM model for early-stage security vulnerability measurement. While these applications showcase the paradigm's flexibility, their focus is narrow cultural assessment and security, respectively—and they do not address the core project management challenges of schedule and cost overruns.

A significant portion of the literature has focused on extending GQM's methodological rigor. Farina et al. (2022) conducted a systematic review of 422 studies, identifying the strengths and weaknesses of various GQM frameworks and proposing improvement strategies. Similarly, Addakhil et al. (2021) combined GQM with the Analytic Hierarchy Process (AHP) to create a quantitative scale for project complexity. While these works enhance the GQM methodology itself, their contributions remain

largely conceptual or expert-dependent and lack empirical validation on real-world project data, particularly concerning delay dynamics.

Attempts to integrate GQM with other tools have also been made. Chang et al. (2020) developed an automated code review tool using static analysis and GQM-derived quality metrics. Although innovative, this tool is rule-based and non-adaptive, meaning it cannot learn from new data or adjust its metrics to evolving project contexts. Furthermore, its focus is on code quality, not on the broader project management issues of delay and cost.

Other studies have focused on improving goal traceability and decision support. Askarbekuly et al. (2020) combined GQM with the KAOS methodology to improve the traceability of goals from the organizational context, but their work remained detached from the project execution phase. Takai et al. (2020) proposed a continuous modeling process for IoT systems using GQM + Strategies and GSN, but its high complexity limits practical applicability. Saraiva et al. (2020) developed a GQM-based validation model to support decision-making in software measurement programs, but its application was limited in scale and scope.

The synthesis of these studies reveals a clear pattern: while GQM is a powerful tool for structured, goal-oriented measurement, its traditional applications are static, domain-specific, or lack predictive capability. None of the existing frameworks successfully integrate GQM with the adaptive, learning capabilities of Artificial Intelligence to dynamically measure, predict, and financially quantify software project delays. This study directly addresses this gap by proposing the AI-GQM framework, which transforms the static GQM into a dynamic, predictive system trained, tested, and validated using substantial real-world data.

3 METHODOLOGY

3.1 Research Design

This study adopted a positivist, quantitative research design to develop, operationalize, and validate the proposed AI-GQM framework for proactive software project delay prediction and financial impact assessment. The methodology combined structured measurement design, mathematical model formulation, statistical analysis, and machine learning validation. This design was appropriate because the study examined observable project-level delay and cost variables, modeled their relationships empirically, and evaluated predictive performance using quantitative data.

The study used three distinct empirical sources. The first was a global dataset of 10,000 completed software projects obtained from Zenodo (Zenodo, 2026), which was used as the primary training dataset for the machine learning models. The second was a Saudi context-specific sample of 52 completed software projects, which was reserved for unseen testing, contextual validation, and practical evaluation. The third consisted of survey responses from 111 software project practitioners, which were used to validate the GQM-D indicators and support the derivation of weighting factors for the IDCE. This separation between training data, testing data, and survey-based validation reduced the risk of data leakage and strengthened the methodological rigor of the study.

3.2 Data Sources and Samples

The global project dataset comprised 10,000 completed software projects and was used exclusively for machine learning model development, internal validation, and cross-validation. The dataset provided sufficient volume for identifying general patterns in schedule deviation, cost overrun, project complexity, change frequency, and other delay-related indicators. It was not used for contextual evaluation of the Saudi project environment.

The Saudi context-specific sample included 52 completed software projects with available archival records, including planned and actual schedules, cost-related records, change logs, and documented delay information. This sample was not used during model training or hyperparameter tuning. Instead, it served as an unseen testing sample to evaluate the generalizability and practical performance of the trained AI-GQM framework against real completed project records.

The survey sample included 111 practitioners with experience in software project planning, execution, development, testing, control, or managerial decision-making. The survey data were used to validate the relevance of the GQM-D indicators and derive expert-informed weighting factors for the IDCE. Therefore, the survey component supported construct validation and indicator weighting rather than machine learning model training.

3.3 Variables and Measures

The main outcome variables were schedule delay and cost overrun. Schedule delay was measured using delay duration and delay percentage. Delay duration represented the difference between actual and planned project completion time in days, while delay percentage represented the proportional schedule deviation relative to the planned project duration. Cost overrun was measured as the percentage increase between actual and planned project cost.

The predictor variables consisted of delay and cost drivers derived from the GQM-D structure. These included technical, administrative, client-related, and organizational indicators. The validated indicator set included 22 delay-related variables: code quality, requirement complexity, testing tool maturity, system compatibility, design flaws, technical documentation accuracy, approval procedure complexity, management experience, planning accuracy, schedule pressure, resource allocation efficiency, bureaucratic hurdles, scope creep, client response time, vision clarity, stakeholder communication, sudden requirement shifts, work environment quality, executive support, methodology maturity, team skill gap, and automation tool utilization.

The weighting factors used in the IDCE were derived from practitioner responses using the RII. These weights quantified the relative importance of each delay indicator and ensured that the equation reflected both empirical evidence and expert-informed judgment.

3.4 Development of the GQM-D Framework

The traditional GQM paradigm was re-engineered into a dynamic measurement model, referred to as GQM-D. This phase transformed static project measurement into an

adaptive structure capable of reflecting evolving project conditions. The framework translated research goals into operational questions and measurable delay-related indicators, which were then linked to project-level delay and cost outcomes.

The GQM-D framework was structured around three measurement goals: analyzing delay patterns in software projects, developing a predictive index for delay risk, and estimating the financial impact of delay. Unlike conventional GQM applications, where metrics are usually fixed at a specific point in time, the GQM-D model allowed delay indicators, weights, and values to evolve across the project lifecycle.

3.5 Formulation of the IDCE

The IDCE was developed as the mathematical core of the AI-GQM framework. It linked weighted delay indicators, normalized delay values, factor significance, correlation effects, AI-predicted delay values, and project-specific risk adjustments into a composite delay-cost index.

The IDCE was used to estimate delay-related financial impact and classify projects into risk categories. The model incorporated four components: a weighted delay component, a correlation component, a predictive component, and a risk adjustment component. The weighted delay component captured the contribution of validated delay indicators. The correlation component represented interdependencies among delay factors. The predictive component incorporated machine learning outputs. The risk adjustment component accounted for project-specific contextual risk.

3.6 Data Preprocessing and Feature Engineering

Before analysis, all datasets underwent structured preprocessing. Duplicate records were removed, missing values were assessed, and inconsistent entries were screened. Continuous variables were normalized or standardized where required, and categorical variables were encoded before model training. Outliers were examined using statistical thresholds and project-level plausibility checks.

Derived variables were then generated, including delay duration, delay percentage, cost overrun percentage, change frequency, and IDCE-related composite indicators. The final feature set was used for machine learning training, statistical analysis, and external validation.

3.7 Machine Learning Model Development

Three machine learning models were developed and evaluated: Random Forest, LSTM, and XGBoost. Random Forest was used for robust classification and feature importance analysis. LSTM was used to support temporal delay prediction where sequential project information was available. XGBoost was used for structured data prediction and nonlinear relationship modeling. The global dataset of 10,000 completed software projects was used for model training and internal validation. Model development was conducted using 10-fold cross-validation to improve generalizability and reduce overfitting. The 52-project Saudi context-specific sample was excluded from training and hyperparameter tuning and was used only for external validation.

3.8 Statistical Analysis

Statistical analysis was conducted using SPSS version 28. Descriptive statistics were used to summarize the characteristics of the 52-project Saudi testing sample, including sectoral distribution, project duration, project cost, delay duration, delay percentage, cost overrun, and number of changes. Continuous variables were summarized using means, standard deviations, minimum values, maximum values, and interquartile ranges where appropriate.

Correlation analysis was used to examine associations between delay indicators, IDCE values, and actual cost overrun. Regression analysis was used to evaluate the explanatory power of the IDCE in relation to actual cost-overrun outcomes. The coefficient of determination, adjusted R^2, regression coefficients, confidence intervals, and p-values were used to assess model fit and explanatory strength. Survey responses from the 111 practitioners were analyzed to assess the relevance and relative importance of the GQM-D indicators. The RII was used to rank the delay indicators and derive weighting factors for the IDCE.

3.9 Model Evaluation and Validation

Model performance was evaluated using classification and regression metrics. Classification performance was assessed using accuracy, precision, recall, F1-score, and area under the receiver operating characteristic curve where applicable. Regression and forecasting performance were assessed using RMSE, MAE, MAPE, and R^2. The final validation of the AI-GQM framework was conducted using the unseen 52-project Saudi testing sample. Model predictions were compared with actual historical outcomes to assess prediction accuracy, financial estimation performance, and risk classification reliability. SHAP analysis was used to identify the most influential predictors contributing to model outputs and to improve the interpretability of the machine learning results.

3.10 Ethical Considerations and Data Availability

The Saudi project records were anonymized before analysis to protect organizational confidentiality. No personally identifiable information was included in the analytical dataset. Practitioner survey responses were treated confidentially and analyzed in aggregate form. The global training dataset is publicly available through Zenodo (Zenodo, 2026). The Saudi project records and survey data are not publicly released because of confidentiality restrictions; however, anonymized data may be made available from the corresponding author upon reasonable request and subject to institutional approval.

4 THE PROPOSED (AI-GQM) FRAMEWORK

The proposed AI-GQM framework integrates the structured measurement logic of the GQM paradigm with machine learning-based prediction to support proactive assessment of software project delays and their financial consequences. The framework consists of three connected components: the GQM-D measurement model, the IDCE, and the DIA tool. GQM-D defines dynamic delay-related

indicators, IDCE translates these indicators into a composite delay-cost index, and DIA operationalizes the framework as a decision-support environment for estimating delay risk, financial impact, and intervention priorities.

4.1 The GQM-Dynamic (GQM-D) model

The GQM-Dynamic model was developed as a qualitative evolution of the traditional model, specifically designed to accommodate the dynamic and complex nature of delays in software projects (Basili et al., 2007; Monden et al., 2012; Nakai et al., 2014). Unlike conventional GQM applications, which typically define metrics as fixed measurement elements, GQM-D treats delay-related indicators as variables that can change over the project lifecycle in response to temporal, technical, organizational, and contextual conditions.

The model was structured around three measurement goals: analyzing delay patterns in context-specific software projects, developing a predictive index for delay risk, and estimating the financial impact of delay. These goals were translated into operational questions addressing the main drivers of delay, the early prediction of delay risk, and the relationship between delay severity and cost escalation. The resulting metrics include delay rate, delay severity, and delay cost, which together capture the temporal, operational, and financial dimensions of project delay.

In this structure, delay rate measures the percentage deviation from the planned schedule, delay severity reflects the relative importance of delayed tasks or factors, and delay cost links schedule deviation to financial impact. By defining these elements as dynamic rather than static variables, GQM-D enables the framework to update its assessment as project conditions evolve. Table 1 summarizes the main components of the GQM-D delay model, including their notation, mathematical representation, value range, and practical interpretation

Table 1. Summary of the GQM-D Delay Model.

Component	Code	Description	Mathematical Formula	Value Range
Dynamic Weight	wi(t)	Represents the relative importance of delay factor i at time t, which decays over the project lifecycle and can be recalibrated.	$wi\,(t) = w_{i0} \cdot e^{(-\lambda t)} + \Delta w$	[0, 1]
Delay Function	fi(Di)	Converts raw delay time Di (e.g., in days) into a standardized, normalized value between 0 and 1.	$f_i(D_i) = \tanh\left(\frac{D_i}{D_{norm}}\right)$	[0, 1]
Significance Function	gi(Ii)	Reflects the relative importance of the delayed task based on an importance index Ii (e.g., critical path, stakeholder priority).	$g_i\,(I_i) = 1 + \log_{10}(I_i + 1)$	[1, ∞)
Correlation Factor	Cij	Measures the degree to which a delay in task i is correlated with a delay in task j, capturing cascading effects.	$C_{ij} = C_{ov}\,\frac{(D_i, D_j)}{(\sigma_i \sigma_j)}$	[-1, 1]
Composite Index	CI(t)	Aggregates the individual factor contributions into a single, holistic index representing the project's overall delay status at time t.	$CI(t) = \frac{(\sum w_i f_i g_i)}{(\sum w_i)}$	[0, 10]

As summarized in Table 1, the GQM-D model consists of five interrelated components that collectively support delay measurement, normalization, weighting, dependency modeling, and composite index generation.

4.1.1Dynamic Weight (wi (t))

The dynamic weight represents the relative importance of delay factor (i) at time (t). In contrast to static weighting approaches, GQM-D allows factor weights to vary across the project lifecycle. For example, requirement-related factors may have greater influence during early planning and specification phases, whereas testing or integration-related factors may become more influential during later phases. The dynamic weight is defined as:

$$wi\,(t) = w_{i0} \cdot e^{(-\lambda t)} + \Delta w \qquad (1)$$

where wi0 is the initial weight of factor (i), λ is the decay parameter, (t) represents time, and Δw denotes periodic recalibration based on updated project conditions. This formulation enables the model to remain responsive to changes in project phase, risk exposure, and emerging delay patterns.

4.1.2Delay Function (fi(Di))

The delay function converts raw delay values into standardized scores, allowing delay effects to be compared across projects and tasks of different durations. A delay of the same absolute length may have different implications depending on planned task duration, task criticality, and project scale. The normalized delay function is expressed as:

$$f_i(D_i) = \tanh\left(\frac{D_i}{D_{norm}}\right) \quad (2)$$

where Di is the observed delay for factor or task (i), and Dnorm is the normalization parameter calibrated according to project characteristics, such as planned duration or task criticality. The hyperbolic tangent function bounds the normalized delay score between 0 and 1, reducing the influence of extreme delay values on the overall index.

4.1.3Significance Function (gi(Ii))

The significance function captures the relative importance of the delayed task or factor. Delays affecting critical-path tasks, high-priority deliverables, or activities with multiple dependencies may have greater project-level consequences than delays in non-critical activities. The significance function is defined as:

$$g_i(I_i) = 1 + \log_{10}(I_i + 1) \quad (3)$$

where (Ii) represents the importance index of factor or task (i). This index may be derived from task criticality, stakeholder priority, dependency count, or managerial assessment. The logarithmic structure allows significance to increase with importance while limiting the dominance of extreme values.

4.1.4Correlation Factor (Cij)

The correlation factor captures dependency and cascading effects between delay factors. In software projects, delays are often interdependent; for example, design delays may affect development, integration, and testing activities. The correlation factor is calculated as:

$$C_{ij} = C_{ov}\frac{(D_i, D_j)}{(\sigma_i \sigma_j)} \quad (4)$$

Where Cov (Di, Dj) is the covariance between delay factors (i) and (j), and σi and σj are their respective standard deviations. A positive value indicates that delays in one factor tend to increase with delays in another, while a negative value indicates an inverse relationship. This component allows the model to account for systemic delay propagation rather than treating delay factors as independent events.

4.1.5Composite Index (CI(t))

The composite index represents the overall delay status of the project at time (t). It integrates the weighted, normalized, and significance-adjusted contributions of all active delay factors into a single interpretable score:

$$CI(t) = \frac{(\sum w_i f_i g_i)}{(\sum w_i)} \quad (5)$$

where wi, fi, and gi represent the dynamic weight, normalized delay score, and significance score for factor (i), respectively. The resulting index provides a consolidated measure of project delay severity and serves as the main input for the downstream predictive and financial quantification components of the AI-GQM framework.

4.2The Integrated Delay-Cost Equation (IDCE)

The IDCE was formulated to quantify the financial impact of software project delays by integrating weighted delay indicators, correlation effects, machine learning predictions, and project-specific risk adjustment into a single delay–cost index. The equation 6 provides the mathematical link between the GQM-D measurement structure and the predictive layer of the AI-GQM framework.

$$IDCE(t) = \sum[w_i(t).f_i(D_i).g_i(I_i)] + \lambda \sum C_j^2(t) + \alpha \cdot AI_{pred(t)} + \beta \cdot Risk_{adj} \quad (6)$$

where $w_i(t)$ is the dynamic weight assigned to delay indicator (i) at time (t), $f_i(D_i)$ is the normalized delay function, and $g_i(I_i)$ represents the significance function for the delayed task or factor. The first term therefore captures the weighted contribution of delay indicators. In this study, (n) represents the number of validated delay indicators included in the model. If the 22 indicators are used directly, (n = 22); if these indicators are aggregated into broader dimensions, (n) represents the number of aggregated dimensions.

The second term represents correlation effects among delay factors, where $C_j^2(t)$ denotes the correlation coefficient for delay dependency (j), (m) represents the number of modeled correlation relationships, and λ controls the relative contribution of correlation effects to the overall index. The third term incorporates the machine learning prediction, where AIpred(t) is the predicted delay or cost-risk output at time (t), and α represents the confidence weight assigned to the prediction. The final term represents project-specific risk adjustment, where Riskadj captures contextual risk conditions and β controls the sensitivity of the index to these risks.

The resulting IDCE value provides a composite measure of delay-related financial exposure. Higher IDCE values indicate greater delay severity, stronger dependency effects, higher predicted risk, and greater expected cost impact. This index is subsequently used to classify projects into risk categories and support delay-aware financial decision-making.

4.3The Advanced AI-GQM Hybrid Model

The AI-GQM hybrid model is a multi-layered architecture that harmonizes systematic modeling with computational intelligence. It integrates a foundational knowledge base of historical project data with a middle layer (GQM-Dynamic engine) and a top layer of advanced AI systems (Random Forest, LSTM, and XGBoost) to provide transparent and predictive insights into software delays. The IDCE model, supported by the AI-GQM architecture, represents the final analytical framework of this study. It serves as an integrated model that consolidates statistical analysis, expert knowledge, and predictive intelligence into a unified structure capable of explaining and forecasting project delay and cost dynamics. The

architecture is structured into three integrated layers that work harmoniously to achieve the research objectives:

4.3.1Basic Layer (Knowledge & Database)

This layer contains the historical data from the 10,000 global projects for machine learning training and the detailed records from the 52 Saudi context-specific projects for unseen testing, contextual validation, and practical evaluation.

4.3.2Middle Layer (Enhanced GQM-D Engine)

The methodological core where strategic objectives are translated into measurable metrics and where the IDCE equation is implemented to manage the data analysis process.

4.3.3Top Layer (Advanced AI System)

This layer adds the predictive dimension using specialized algorithms to address different aspects of the delay and cost problem.

4.4Integrated Workflow of the DIA Tool

The practical implementation of the AI-GQM framework is operationalized through the DIA tool, which converts project-level data into interpretable delay-risk and financial-impact outputs. As shown in Figure 1, the workflow consists of five sequential stages: data configuration and validation, analytical execution, AI-driven forecasting, recommendation generation, and sensitivity monitoring.

In the first stage, project data are configured and validated to ensure completeness, consistency, and alignment with the GQM-D parameters. In the second stage, the IDCE is applied to estimate the current delay status and associated financial deviation. The third stage incorporates AI-driven forecasting, where predictive models, including LSTM and XGBoost, generate delay-risk and cost-overrun estimates. SHAP-based interpretation is then used to identify the main factors contributing to the predicted outcomes.

In the fourth stage, the tool generates strategic recommendations by prioritizing interventions according to predicted delay severity, expected financial impact, and risk level. In the final stage, monitoring and Monte Carlo simulation are used to assess the sensitivity and robustness of the delay–cost index under alternative risk scenarios. This workflow ensures that the DIA tool produces actionable decision-support outputs rather than purely descriptive summaries. Details regarding the tool design, validation, and implementation are provided in Supplementary File S1.

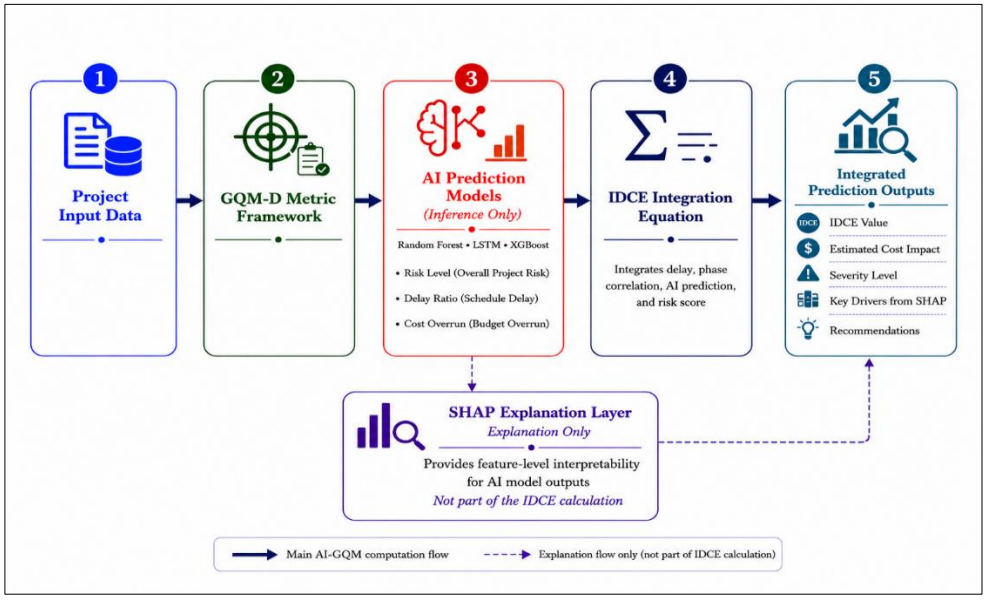

Figure 1. The Integrated AI-GQM Methodological Framework for Software Delay and Cost Estimation, available as online tool at https://fourpoints.sa/Predict/index.html.

5.Data Analysis Procedure

5.1Training, Testing, and Survey Samples

The AI-GQM framework was developed and validated using three distinct empirical sources, each assigned a separate methodological function. The global dataset of 10,000 completed software projects obtained from Zenodo (Zenodo, 2026) was used for training and internal validation of the machine learning models, including Random Forest, LSTM, and XGBoost. Its size provided sufficient statistical volume for identifying general delay patterns, cost-overrun behavior, and nonlinear relationships among project variables.

The Saudi context-specific sample of 52 completed software projects was reserved for unseen testing, contextual validation, and practical evaluation. These projects were excluded from model training and hyperparameter tuning to prevent data leakage. The sample provided archival project records, including timelines, cost-related data, change logs, and documented delay information, allowing the trained framework to be evaluated against real completed project outcomes.

The 111 practitioner survey responses were used to validate the GQM-D indicators and derive weighting factors for the IDCE. These data supported construct validation and expert-informed weighting rather than machine learning model training. This separation between training, testing, and survey-based validation strengthened the methodological rigor of the study by ensuring that model development, empirical testing, and indicator validation were conducted using independent data sources.

5.2Instruments

To identify the core drivers of delay that feed into the Integrated Delay-Cost Equation (IDCE), a two-stage empirical approach was adopted, consisting of exploratory semi-structured interviews with industry experts followed by quantitative survey validation.

The initial pool of indicators was refined through expert consensus and statistical screening, ensuring that only the most relevant and impactful factors were retained. This process yielded 22 specific indicators that represent the primary inputs for the study's mathematical and AI models. These indicators are categorized into four dimensions as shown in Table 2.

These 22 indicators serve as the "Independent Variables" (xi) in the IDCE equation. The weights (wi) assigned to each indicator were derived from the Relative Importance Index (RII), which was used to rank and quantify the relative significance of each factor based on practitioner responses (N=111). This ensures that the equation is not merely a theoretical construct but is firmly grounded in empirical evidence (Table 2).

Table 2. Categorized delay indicators used as IDCE equation inputs

Category	Indicator No.	Indicators Used as Inputs for the IDCE Equation
Technical factors	1–6	1. Code quality 2. Requirement complexity 3. Testing tool maturity 4. System compatibility 5. Design flaws 6. Technical documentation accuracy
Administrative factors	7–12	7. Approval procedure complexity 8. Management experience 9. Planning accuracy 10. Schedule pressure 11. Resource allocation efficiency 12. Bureaucratic hurdles
Client-related factors	13–17	13. Scope creep / frequent changes 14. Client response time 15. Vision clarity 16. Stakeholder communication 17. Sudden requirement shifts
Organizational factors	18–22	18. Work environment quality 19. Executive support 20. Methodology maturity, including Agile or Waterfall practices 21. Team skill gap 22. Automation tool utilization

5.3Systematic Application and Data Collection

To ensure the reliability of the IDCE coefficients, a multi-source data collection methodology was employed, focusing on four primary levels:

5.3.1Documentary Auditing

Analysis of official timelines, financial documents, and change logs for the 52 projects to extract objective "Ground Truth" data.

5.3.2Semi-Structured Interviews

Engaging key stakeholders to capture the "Why" behind the delays factors often missed by automated logging.

5.3.3Field Observation

Direct assessment of communication and decision-making workflows within the project teams.

5.3.4Systematic Questionnaires (N=111)

Quantitative data collection from practitioners to determine the weighting factors (wi) for the delay drivers. The data collection process followed a structured multi-stage procedure to ensure accuracy, reliability, and completeness. Initially, data were gathered from multiple sources, including online repositories, official project records, and institutional databases. A total of diverse data points were extracted to represent project timelines, cost deviations, and delay occurrences. In addition, expert-based data collection was conducted through structured engagement with more than 100 domain experts. These experts contributed to validating the identified indicators and refining their relative importance. The integration of multi-source data ensured both breadth and depth in capturing the delay-cost phenomenon. Selected source-code excerpts are provided in the supplementary file (File S3) for transparency.

5.4Advanced Data Processing

Before analysis, all data underwent a rigorous three-stage processing cycle:

5.4.1Stage 1

Cleaning & Normalization: This stage involved rigorous preprocessing procedures, including the removal of duplicate records, handling of missing values, and detection of outliers using statistical thresholds. Data consistency was further ensured through normalization techniques, allowing integration across heterogeneous data sources.

5.4.2Stage 2

Systematic Coding: Converting qualitative interview insights into quantitative parameters for the AI models.

5.4.3Stage 3

Integration & Synthesis: Combining SPSS-based statistical results with the predictive outputs of the AI models to create a comprehensive narrative of the delay-cost phenomenon.

5.5Model Validation and Reliability Framework

To ensure empirical rigor, the study defines specific target thresholds as benchmarks for evaluating the performance and practical readiness of the proposed AI-GQM framework:

5.5.1Predictive Accuracy (RMSE)

Target < 0.15 to ensure high precision in delay estimation.

5.5.2Classification Quality (F1-Score)

Target > 0.85 to ensure reliable project risk categorization.

5.5.3Explanatory Strength (Adjusted R^2)

Target > 0.80 to confirm that the IDCE captures a substantial proportion of the variance in cost-overrun outcomes.

5.5.4Operational Efficiency

Target response time < 5 seconds for the DIA tool to ensure practical usability in decision-support environments.

5.5.5Field Readiness (Acceptance Rate)

Target > 80% based on practitioner feedback (N=155), indicating sufficient acceptance of the framework's usability, clarity, and managerial relevance.

In addition, the research addresses potential validity threats through a proactive validation strategy:

5.5.6Internal Validity and Analytical Rigor

To reduce bias and improve the reliability of the findings, quantitative survey results were triangulated with expert feedback and project-level archival evidence. K-fold cross-validation and bootstrap procedures were employed to reduce the influence of outliers, assess model stability, and improve the consistency of the validation results. Data leakage was also minimized by separating the global training dataset from the unseen 52-project testing sample.

5.5.7External Validity and Generalizability

While the 52-project Saudi context-specific testing sample provides practical grounding, the three-source empirical strategy strengthens the broader applicability of the framework. The 10,000-project Zenodo dataset was used for machine learning training, the 52-project sample was reserved for unseen testing and validation, and the 111 survey responses were used for GQM-D indicator validation and IDCE weighting. This design supports both general pattern learning and context-sensitive evaluation, while allowing the framework to be recalibrated for different organizational and regulatory environments.

5.5.8Construct and Content Validity

The GQM-D indicators and IDCE components were reviewed and validated through expert input to ensure that they accurately represent the constructs of project delay, cost impact, risk severity, and delay-related financial consequences. This process helped reduce measurement bias and ensured that the selected indicators were theoretically meaningful and practically relevant.

5.5.9Criterion Validity

The predictive outputs of the AI-GQM framework were compared against actual historical project outcomes in the unseen 52-project testing sample. This comparison assessed the framework's ability to classify project risk, estimate delay behavior, and quantify financial impact in a way that reflects real project performance.

6.RESULTS

6.1Descriptive Analysis of the Saudi Context-Specific Testing Sample

To establish the empirical baseline for evaluating the proposed AI-GQM framework, the 52-project Saudi context-specific testing sample was analyzed in terms of sectoral distribution, project duration, cost profile, and delay-related indicators. This sample was not used for machine learning training or hyperparameter tuning; rather, it was reserved for unseen testing, contextual validation, and practical evaluation against real completed project records.

Table 3 presents the sectoral distribution and primary project characteristics of the Saudi context-specific testing sample (N=52). The governmental sector accounts for the largest share of projects (59.6%), with an average implementation period of 18.5 months and an average cost of 8.2 million SAR. This relatively higher duration and cost may reflect the complexity of projects in this sector, which are often characterized by multiple stakeholders, formal approval procedures, extensive documentation, and governance requirements. In contrast, commercial projects constituted 30.8% of the sample, with a shorter average duration of 12.3 months and an average cost of 4.7 million SAR, suggesting greater flexibility in decision-making and faster implementation. Educational projects, despite their smaller representation in the sample (9.6%), recorded the lowest average duration and cost.

Table 3. Sectoral Distribution of the Saudi Context-Specific Testing Sample (N=52).

Sector	Number of Projects	Percentage	Average Duration (Months)	Average Cost (Million SAR)
Governmental	31	59.6%	18.5	8.2
Commercial	16	30.8%	12.3	4.7
Educational	5	9.6%	9.8	2.1
Total	52	100%	15.8	6.5

Building on the sectoral baseline, Table 4 summarizes the descriptive statistics for the main delay and cost indicators within the Saudi context-specific testing sample (N=52), quantifying the magnitude of temporal and financial deviations. The average delay duration was (87.3) days with a relatively high standard deviation (45.2), indicating significant variance among projects in terms of delay intensity, and delay duration ranged from a minimum of (15) days to a maximum of (240) days, reflecting substantial differences in the projects' ability to adhere to time schedules. The average delay percentage was (28.7%), a relatively concerning indicator reflecting a temporal deviation exceeding a quarter of the

original planned schedule on average. The average cost increase was also high, reaching (35.2%), with some projects experiencing increases exceeding (120%), emphasizing the strong correlation between temporal delays and additional costs. Furthermore, the average number of changes (47.8) points to an unstable project environment, particularly at the requirements level, which helps explain the results of subsequent models.

Table 4. Descriptive Statistics of Main Delay Indicators in the Saudi Testing Sample (N = 52)

Indicator	Mean	Standard Deviation	Minimum	Maximum	Interquartile Range (IQR)
Delay duration (days)	87.3	45.2	15	240	62–105
Delay percentage (%)	28.7	18.5	5.2	85.3	15.8–38.4
Cost overrun (%)	35.2	22.1	8.5	120.5	20.3–45.8
Number of changes	47.8	32.5	5	150	25–65

[Note: IQR = interquartile range. Delay duration is reported in days, whereas delay percentage and cost overrun are reported as percentages. The number of changes represents the total recorded change events per project.]

The systemic nature of these deviations was examined based on practitioner insights (N=111), where it was found that there are strong, statistically significant positive relationships between delay duration, delay percentage, cost increase, and the number of changes.

6.2IDCE Application

The empirical application of the IDCE equation to the Saudi context-specific testing sample (N=52) reveals distinct risk clusters as shown in Table 5, which displays the classification of projects according to the IDCE risk categories, distinguishing between Excellent, Good, Moderate, Poor, and Critical levels, illustrating how projects are distributed across the delay-cost risk spectrum. The largest proportion of projects fall into the "Good" (34.6%) and "Moderate" (28.8%) categories, indicating that most projects experience significant delays, though not to a critical level. Conversely, (21.2%) of projects fall into the "Poor" and "Critical" categories, exhibiting very high average delays and cost increases.

Table 5. Distribution of Projects by IDCE Category and Associated Delay–Cost Outcomes

IDCE Category	IDCE Value Range	Number of Projects	Percentage of Projects	Average Delay (Days)	Average Cost Overrun (%)
Excellent	0.0–1.0	8	15.4%	23.0	10.8
Good	1.0–2.0	18	34.6%	62.5	24.4
Moderate	2.0–3.0	15	28.8%	95.7	36.8
Poor	3.0–4.0	7	13.5%	144.5	59.2
Critical	>4.0	4	7.7%	195.9	84.6
Total	—	52	100.0%	87.3	35.2

The IDCE system evaluation and prototype are detailed in the supplementary file (File S2).

6.3AI Model Performance Analysis

Using the global dataset (N=10,000) for training and the unseen 52-project Saudi context-specific testing sample for validation, Table 6 compares the predictive accuracy and error rates of the Hybrid Model against standard AI algorithms, where the hybrid model achieved the highest predictive accuracy (92.5%) and the lowest RMSE and MAE values compared to the rest of the models (Table 6).

Table 6. Comparative evaluation of predictive models used in the AI-GQM framework

Model	Accuracy (%)	RMSE	MAE	R^2	Training Time (s)	Prediction Time (ms)
Random Forest	87.3	0.142	0.098	0.834	12.5	45
LSTM	89.1	0.135	0.092	0.851	325.8	120
XGBoost	90.2	0.128	0.087	0.867	28.7	65
Hybrid Model	92.5	0.112	0.075	0.892	58.3	85

The reliability of the hybrid approach was assessed by examining the model's performance consistency across different data scenarios and feature constraints, including the original dataset, a dataset without outliers, and a reduced-feature dataset. The results showed that the hybrid model remained relatively stable under these different conditions, reflecting its ability to generalize and adapt to data variations. This stability provides an important indication of the model's reliability and applicability in diverse practical environments.

To provide model transparency, Table 7 ranks the most influential delay and cost drivers according to their SHAP values, identifying the primary factors shaping the model's predictions. The SHAP analysis shows that delay duration, number of changes, and requirements variability are among the most influential factors on the model's predictions. This indicates

that delay is not the result of a single factor, but rather the product of a complex interaction between managerial, technical, and organizational decisions. The impact of team size and contract value also emerges as secondary but influential factors, while team experience played a protective role, as higher experience was associated with a lower probability of delay.

Table 7. SHAP-Based Feature Importance and Direction of Impact

Rank	Feature	Mean Absolute SHAP Value	Positive Impact on Delay/Cost Risk	Negative Impact on Delay/Cost Risk
1	Delay duration	0.45	Long delays	Short delays
2	Number of changes	0.38	Many changes	Few changes
3	Requirements volatility	0.32	High volatility	High stability
4	Team size	0.28	Large teams	Small teams
5	Contract value	0.25	Large projects	Small projects
6	Approval complexity	0.22	High complexity	Simplified procedures
7	Design defects	0.20	Many defects	High quality
8	Integration problems	0.18	Complex integration	Smooth integration
9	Testing delay	0.15	Repeated tests	Effective testing
10	Team experience	0.12	Low experience	High experience

[2]Note: Higher accuracy and R^2 values indicate better predictive performance, whereas lower RMSE and MAE values indicate lower prediction error. The hybrid model achieved the best overall performance, with the highest accuracy and explanatory power and the lowest error values, although it required more training time than Random Forest and XGBoost.

Moreover, the regression analysis between IDCE and cost overrun was performed, and showed a strong, statistically significant positive relationship between the IDCE index and the increase in cost, with a value of R^2 (0.862), which means that more than (86%) of the change in cost can be explained by the IDCE, as the slope coefficient (12.45) indicates that every one-unit increase in the IDCE corresponds to a large increase in the cost percentage, which confirms the explanatory and practical power of the proposed equation.

Beyond statistical correlation, the IDCE index was translated into concrete financial terms that project managers and stakeholders can readily interpret and act upon (Table 8), which demonstrates the sharp escalation in financial impact as projects move from the "Excellent" category (10.8% cost increase) to the "Critical" category (84.6% cost increase), providing a clear business case for early intervention and proactive delay management.

Table 8. Estimated Financial Losses Due to Software Project Delay

Project Category	Average Base Cost (Million SAR)	Average IDCE Value	Estimated Additional Cost (Million SAR)	Percentage Increase in Cost
Excellent	4.8	0.65	0.52	10.8%
Good	6.2	1.45	1.51	24.4%
Moderate	7.5	2.35	2.76	36.8%
Poor	9.3	3.55	5.51	59.2%
Critical	12.1	4.85	10.24	84.6%

6.4Comparison between AI-GQM model and traditional models

To position the proposed AI-GQM hybrid model relative to established approaches for delay and cost prediction, this section compares its performance with traditional GQM, classical statistical models, and standalone AI models. Table 9 summarizes this comparison across key evaluation criteria, including prediction accuracy, prediction speed, interpretability, data adaptability, and computational cost. By contrasting these dimensions, the table highlights how the hybrid model achieves a favorable balance between accuracy and transparency while maintaining reasonable computational requirements, making it particularly suitable for practical deployment in project management environments. Results indicate the superiority of the hybrid model in terms of predictive accuracy and data adaptability, while maintaining a high level of interpretability compared to models based solely on artificial intelligence, and this superiority reflects the successful balance between methodology and artificial intelligence.

Table 9. Performance Comparison of the Proposed AI-GQM Hybrid Model with Alternative Modelling Approaches

Criterion	Hybrid Model (AI-GQM)	Traditional GQM	Statistical Models	AI-Only Models
Prediction accuracy	92.5%	74.8%	81.2%	88.7%
Prediction speed	85 ms	120 ms	95 ms	70 ms
Interpretability	High	High	Medium	Low
Data adaptability	Excellent	Good	Medium	Excellent
Computational cost	Medium	Low	Low	High
Implementation ease	Medium	High	High	Low

While descriptive indicators and performance metrics provide an initial picture of model behavior, it is essential to verify that the observed differences between models are statistically significant and not due to random variation. To this end, a series of formal hypothesis tests was conducted, including a t-test comparing the hybrid model with traditional GQM, one-way ANOVA across all models and a non-parametric Kruskal–Wallis test. Table 10 summarizes the results of these significance tests, reporting the test statistics, degrees of freedom, p-values, and decisions. Together, these results provide robust statistical evidence that the AI-GQM hybrid model significantly outperforms alternative approaches. These statistically significant differences support the reliability of the results and strengthen the conclusion that the proposed hybrid model offers superior predictive performance compared with traditional GQM, classical statistical models, and standalone AI models.

Table 10. Results of Statistical Significance Tests.

Test	Value	Degrees of Freedom	p-value	Decision
t-test (Hybrid vs. GQM)	5.82	50	<0.0001	Statistically significant difference
ANOVA (All Models)	8.47	3, 200	0.0001	Statistically significant differences
Kruskal-Wallis	15.23	3	0.0016	Statistically significant differences

7. Discussion

Software project delays remain a persistent challenge in software engineering because they affect delivery time, financial control, stakeholder confidence, and organizational value. Prior research has shown that delays are rarely caused by a single factor; rather, they emerge from the interaction of technical, managerial, organizational, and client-related conditions (Binu, 2025; Kula et al., 2023). However, much of the existing literature has remained descriptive, focusing on identifying delay factors without translating them into predictive or financially interpretable models. The present study addresses this limitation by developing and validating the AI-GQM framework, which integrates GQM-D, IDCE, machine learning prediction, and the DIA tool to support proactive delay and cost assessment.

The first major finding is that software project delays in the Saudi context were substantial and financially consequential. The 52-project testing sample showed a mean delay duration of 87.3 days and a mean cost overrun of 35.2%, indicating a clear association between schedule deviation and financial escalation. This finding is consistent with Choetkiertikul et al. (2015), who emphasized the predictability of software project delays using project-level features, and with Briciu et al. (2016), who highlighted the limitations of conventional software cost-estimation methods under uncertain project conditions (Briciu et al., 2016). However, the present study extends this literature by converting delay-related variables into a composite delay-cost index rather than treating delay and cost overrun as separate outcomes.

The second important finding concerns the multidimensional nature of delay. SHAP analysis identified delay duration, number of changes, requirements volatility, team size, contract value, approval complexity, and design defects as the most influential predictors. These results support previous studies showing that requirement uncertainty and organizational complexity strongly affect software project performance (Haleem et al., 2021; Lebedeva & Guseva, 2019). They also align with Saudi-context studies reporting that planning weaknesses, approval procedures, coordination issues, and limited tool adoption contribute to software project failure (Alenezi, 2022; Alghamdi et al., 2024; Alotaibi et al., 2016). The contribution of the current study lies in moving beyond factor identification by ranking these drivers according to predictive influence and embedding them within a structured delay-cost framework.

The results also demonstrate the value of extending the traditional GQM paradigm into a dynamic measurement structure. Earlier GQM-based studies have shown the usefulness of goal-oriented measurement in software quality, agile monitoring, process assessment, and decision support (Aldahmash & Gravell, 2018; Boerman et al., 2015; Farina et al., 2022). Nevertheless, these applications have generally treated GQM as a static measurement approach, with limited ability to adapt to evolving project conditions. The proposed GQM-D model addresses this limitation by allowing delay indicators and their weights to change across the project lifecycle. In this respect, the study advances the GQM literature by linking structured measurement to predictive analytics and financial interpretation.

The IDCE represents a central contribution because it formalizes the relationship between delay indicators and financial consequences. Previous studies have acknowledged that software delays increase cost and reduce project performance, but few have provided a mathematical mechanism for estimating this impact in an interpretable manner (Garg & Rastogi, 2022; Ibraigheeth & Fadzli, 2019). In the present study, the IDCE showed strong explanatory power for cost-overrun outcomes, indicating that delay-related financial exposure can be systematically modeled. This finding is particularly important for project managers and project management offices because it converts abstract delay severity into actionable financial information.

The performance comparison further supports the value of the hybrid approach. The AI-GQM model achieved higher predictive accuracy than traditional GQM, classical statistical models, and standalone AI models.

This result suggests that combining structured measurement with machine learning provides advantages over isolated approaches. Traditional GQM offers interpretability but lacks predictive capacity, while standalone AI models may provide strong prediction but are often limited by transparency concerns. This is consistent with broader concerns in AI-based estimation research, where predictive accuracy must be balanced with explainability and managerial usability (Alhumam, 2025; Alturki & E-AMIN, 2025). By integrating SHAP-based interpretation, the proposed framework strengthens transparency and supports trust in model outputs.

The Saudi context-specific results also highlight the importance of contextual validation. Governmental projects represented the largest share of the sample and showed higher average duration and cost values, likely reflecting more complex governance structures, documentation requirements, and approval processes. This finding is consistent with Alotaibi et al. (2016), who identified project management tool use and procedural complexity as important issues in Saudi project environments (Alotaibi et al., 2016). By training the model on a large global dataset and validating it on Saudi project records, the study balances general pattern learning with local contextual relevance.

Overall, the findings show that software project delays can be measured, predicted, and financially quantified through an integrated framework. The study contributes theoretically by extending GQM into a dynamic AI-supported model, methodologically by separating training, testing, and survey-based validation, and practically by providing the DIA tool for delay-risk and cost-impact assessment. These contributions position the AI-GQM framework as a useful approach for improving schedule reliability, financial control, and risk-informed decision-making in software project management.

7.1 Strengths and Limitations

This study has several strengths, including the use of three distinct empirical sources, separation of training and testing data, integration of practitioner-informed indicator weighting, and validation of the AI-GQM framework using an unseen Saudi context-specific project sample. The combination of GQM-D, IDCE, machine learning, and DIA also strengthens the practical relevance and interpretability of the proposed approach. However, the study has limitations. The external validation sample was limited to 52 Saudi projects, which may affect generalizability across other sectors and countries. In addition, some contextual and organizational variables may not have been fully captured in the available data.

8. Conclusion

This study developed and validated an AI-GQM framework for proactively measuring, predicting, and financially quantifying software project delays. By integrating GQM-D, IDCE, machine learning models, and the DIA tool, the framework addresses the limitations of static cost-estimation and project control methods. The findings show that delays arise from interacting technical, managerial, organizational, and client-related factors and are strongly associated with cost escalation. The hybrid model achieved high predictive performance, while the IDCE provided an interpretable measure of delay-related financial exposure. Overall, the framework offers a practical and data-driven approach for improving schedule reliability, financial control, and risk-informed decision-making.

Declarations

Author Contributions

The authors contributed to the conceptualization, methodology, framework development, data analysis, validation, interpretation of results, writing, review, and editing of the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding

The project was funded by KAU Endowment (WAQF) at King Abdulaziz University, Jeddah, Saudi Arabia. The authors, therefore, acknowledge with thanks WAQF and the Deanship of Scientific Research (DSR) for technical and financial support.

Data Availability

The 10,000-project training dataset is publicly available through Zenodo (Zenodo, 2026). The Saudi context-specific project records and survey data are anonymized and may be made available from the corresponding author upon reasonable request, subject to confidentiality and ethical restrictions.

Conflicts of Interest

The authors declare no conflict of interest.

References

1. Abdalkareem, R., Mujahid, S., Shihab, E., & Rilling, J. (2021). Which Commits Can Be CI Skipped? IEEE Transactions on Software Engineering, 47(3), 448-463. https://doi.org/10.1109/TSE.2019.2897300
2. Addakhil, M. H., Yaqin, M. A., Fadilah, M. F. F., & Wulandari, C. (2021). Metode AHP dan GQM Untuk Pengembangan Metrik Skala dan Kompleksitas Proyek. ILKOMNIKA, 3(3), 344-352. https://doi.org/10.28926/ilkomnika.v3i3.384
3. Aldahmash, A. M., & Gravell, A. (2018). Measuring success in agile software development projects: A GQM approach. ICSEA 2018: The Thirteenth International Conference on Software Engineering Advances, Nice, France.
4. Alenezi, M. (2022). Factors Hindering the Adoption of DevOps in the Saudi Software Industry. arXiv preprint arXiv:2204.09638.
5. Alghamdi, A. A. N., Al Jedaibi, W., & Alghamdi, A. S. A. (2025). Integrating the GQM Model to Evaluate the Effect of Project Delay on Software Industry Project Cost Estimation. Journal of New Materials for Electrochemical Systems, 28(4), 470-486.
6. Alghamdi, A. A. N., Al Jedaibib, W., & Alghamdi, A. S. A. (2024). Factors affecting success or failure in software projects in the

Kingdom of Saudi Arabia. CAHIERS MAGELLANES-NS, 6(1), 783-803.
7. Alhumam, A. (2025). Software cost estimation using TabNet and Harris Hawks Optimization. Scientific Reports.
8. Alotaibi, N. O., Sutrisna, M., & Chong, H.-Y. (2016). Guidelines of using project management tools and techniques to mitigate factors causing delays in public construction projects in kingdom of Saudi Arabia. Journal of Engineering, Project, and Production Management, 6(2), 90.
9. Alshammari, F. H. (2022). Cost estimate in scrum project with the decision-based effort estimation technique. Soft Computing, 26(20), 10993-11005.
10. Alturki, F., & E-AMIN, F. (2025). Comprehensive Analysis of Software Effort Estimation Techniques: Evolving Trends, Key Challenges, and Prospective Directions. International Journal of Computer Applications, 186(68), 42-48.
11. Askarbekuly, N., Sadovykh, A., & Mazzara, M. (2020). Combining two modelling approaches: Gqm and kaos in an open source project. IFIP International Conference on Open Source Systems,
12. Ba'abbad, I. M., & Qureshi, M. R. J. (2021). Quality Extended Use Case Point (QUCP): An Improved Cost Estimation Method. International Journal of Computer Science and Mobile Computing, 10(6), 1-9.
13. Basili, V., Heidrich, J., Münch, J., Regardie, M., Rombach, D., Seaman, C., & Trendowicz, A. (2007). GQM+Strategies: A Comprehensive Methodology for Aligning Business Strategies with Software Measurement. DASMA Software Metric Congress (MetriKon 2007): Magdeburger Schriften zum Empirischen Software Engineering, Kaiserslautern, Germany.
14. Binu, A. (2025). Delays in software projects: Identifying contributing factors [Master's thesis, Jönköping University].
15. Boerman, M. P., Lubsen, Z., Tamburri, D. A., & Visser, J. (2015). Measuring and monitoring agile development status. 2015 IEEE/ACM 6th International Workshop on Emerging Trends in Software Metrics,
16. Briciu, C., Filip, I., & Indries, I. (2016). Methods for cost estimation in software project management. IOP Conference Series: Materials Science and Engineering,
17. Calvo, M., & Beltrán, M. (2024). Applying the Goal, Question, Metric method to derive tailored dynamic cyber risk metrics. Information & Computer Security, 32(2), 133-158. https://doi.org/10.1108/ICS-03-2023-0043
18. Chang, B.-M., Cassandra Son, J., & Choi, K. (2020). A GQM approach to evaluation of the quality of smartthings applications using static analysis. KSII Transactions on Internet & Information Systems, 14(6). http://doi.org/10.3837/tiis.2020.06.003
19. Choetkiertikul, M., Dam, H. K., Tran, T., & Ghose, A. (2015). Predicting delays in software projects using networked classification (t). 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE),
20. Farina, M., Gorb, A., Kruglov, A., & Succi, G. (2022). Technologies for GQM-based metrics recommender systems: a systematic literature review. IEEE Access, 10, 23098-23111.
21. Garg, P., & Rastogi, R. (2022). Statistical Analysis of COCOMO for Automated CRM: Application of SE for Customer Retention in 21 st century Life Style and Smart Cities. 2022 4th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N),
22. Haleem, M., Farooqui, M. F., & Faisal, M. (2021). Cognitive impact validation of requirement uncertainty in software project development. International Journal of Cognitive Computing in Engineering, 2, 1-11.
23. Ibraigheeth, M., & Fadzli, S. A. (2019). Core factors for software projects success. JOIV: International Journal on Informatics Visualization, 3(1), 69-74.
24. Kārkliņa, K., & Pirta, R. (2018). Quality metrics in agile software development projects. Information Technology & Management Science (RTU Publishing House), 21, 54-59. https://doi.org/10.7250/itms-2018-0008
25. Khan, J. A., Khan, S. U. R., Iqbal, J., & Rehman, I. U. (2021). Empirical investigation about the factors affecting the cost estimation in global software development context. IEEE Access, 9, 22274-22294.
26. Kula, E., Greuter, E., Deursen, A. v., & Gousios, G. (2022). Factors Affecting On-Time Delivery in Large-Scale Agile Software Development. IEEE Transactions on Software Engineering, 48(9), 3573-3592. https://doi.org/10.1109/TSE.2021.3101192
27. Kula, E., Greuter, E., van Deursen, A., & Gousios, G. (2023). Dynamic prediction of delays in software projects using delay patterns and bayesian modeling. Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering,
28. Lebedeva, A. V., & Guseva, A. I. (2019). Cognitive maps for risk estimation in software development projects. Biologically Inspired Cognitive Architectures Meeting,
29. Lehrig, S., Eikerling, H., & Becker, S. (2015). Scalability, elasticity, and efficiency in cloud computing: A systematic literature review of definitions and metrics. Proceedings of the 11th international ACM SIGSOFT conference on quality of software architectures,
30. Leme, N. P., Arriel, J., Garcia, A., Azevedo, J., Sousa, T., Bueno, L., Godinho, J., & Pereira, J.

A. (2024). Mental Health and Productivity in Software Development: A Study with the Bravo Central Platform. Proceedings of the 20th Brazilian Symposium on Information Systems,
31. Liu, N. Y.-C., Miao, K. C.-J., & Ying, D. C. Y. (2022). A metric instrument for the games with cultural heritage. International Journal of Serious Games, 9(4), 89–136. https://doi.org/10.17083/ijsg.v9i4.529
32. Mansoor, F., Alim, M. A., Jilani, M. T., Monsoor Alam, M., & Su'ud, M. M. (2024). Enhancing Software Cost Estimation Using Feature Selection and Machine Learning Techniques. Computers, Materials & Continua, 81(3), 4603. https://doi.org/10.32604/cmc.2024.057979
33. Monden, A., Matsumura, T., Barker, M., Torii, K., & Basili, V. R. (2012). Customizing gqm models for software project monitoring. IEICE TRANSACTIONS on Information, E95-D(9), 2169-2182. https://doi.org/10.1587/transinf.E95.D.2169
34. Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. Advances in Neural Information Processing Systems, 30, 4765–4774.
35. Nakai, H., Honda, K., Washizaki, H., Fukazawa, Y., Asoh, K., Takahashi, K., Ogawa, K., Mori, M., Hino, T., & Hayakawa, Y. (2014). Continuous product-focused project monitoring with trend patterns and GQM. 2014 21st Asia-Pacific Software Engineering Conference,
36. Saraiva, R., Medeiros, A., Perkusich, M., Valadares, D., Gorgonio, K. C., Perkusich, A., & Almeida, H. (2020). A Bayesian networks-based method to analyze the validity of the data of software measurement programs. IEEE Access, 8, 198801-198821. https://doi.org/10.1109/ACCESS.2020.3035217
37. Shojaeshafiei, M. (2020). Analytic hierarchy process-based fuzzy measurement to quantify vulnerabilities of web applications. International Journal of Computer Networks & Communications (IJCNC) Vol, 12.
38. Takai, T., Shintani, K., Andoh, H., & Washizaki, H. (2020). Continuous modeling supports from business analysis to systems engineering in IoT development. 2020 9th International Congress on Advanced Applied Informatics (IIAI-AAI),
39. Timoshchuk, E. V. (2020). Assessing the quality of the requirements specification by applying GQM approach and using NLP tools. Труды института системного программирования РАН, 32(2), 15-28. https://doi.org/10.15514/ISPRAS-2020-32(2)-2
40. Zenodo. (2026). Dataset [Data set]. Zenodo. https://doi.org/10.5281/zenodo.19069527

Supplementary Files

File S1. The Delay Impact Analyzer (DIA) Tool: Design, Validation, and Implementation.
File S2. IDCE System Evaluation and Prototype.
File S3. Selected Source-Code Excerpts for Transparency.