Workshop No. 3 — Robust System Design and Project Management

Team #9

Juan Esteban Ávila Trujillo – 20251020054
Juan José León Gómez – 20212020055
Juan Pablo Díaz Ricaurte – 20222020076
Miguel Ángel Hernández Medina – 20222020035

Universidad Distrital Francisco José de Caldas
Systems Engineering Program

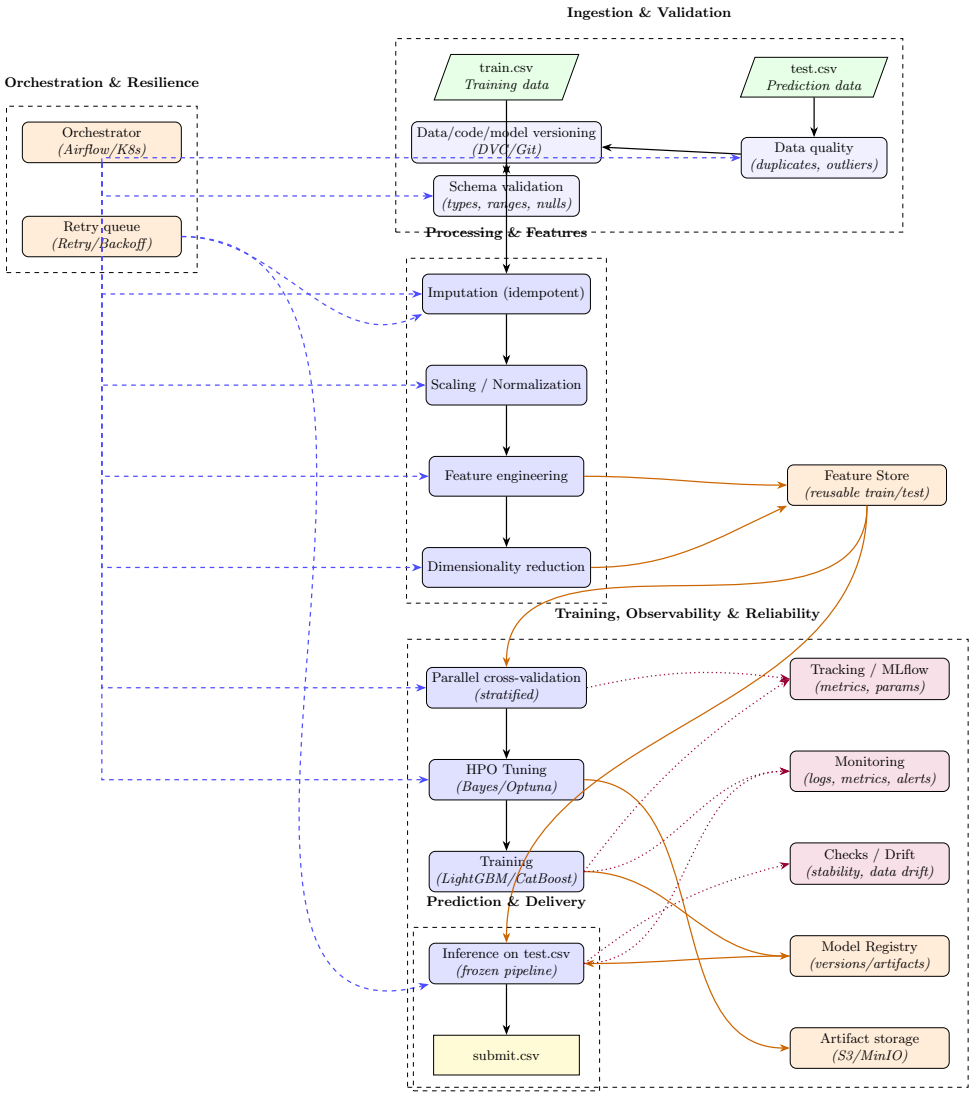
November 8, 2025

Introduction

In Workshop No. 3, we focused on strengthening the system design by applying robust engineering principles and introducing project management strategies to ensure the solution is viable and sustainable. The central objective was to refine the system architecture, address quality and risk, and define how the project will be managed on its path to implementation. We focused on three key areas: Robust Design, Risk and Quality Management, and the definition of a Project Management Plan.

Contents

1 Review and Refine System Architecture



1

Figure 1: Analysis

2 Quality and Risk Analysis

2.1 Overview

The **Loan Default Prediction System** operates in a sensitive financial environment that demands high levels of **reliability**, **accuracy**, and **data security**.
Because it handles large, anonymized datasets and relies on continuous feedback for model retraining, the system must prevent data degradation, ensure traceability, and maintain operational continuity even under unpredictable conditions.

To achieve this, the project integrates **robust engineering principles** inspired by ISO 9000 (quality management), **CMMI Level 3** (process maturity), and **Six Sigma** (error reduction and continuous improvement). These frameworks guide both the design and monitoring of software quality and risk mitigation across all modules.

2.2 Identified Risks and Mitigation Strategies

Table 1: Identified Risks and Mitigation Strategies

Risk ID	Description	Potential Impact	Mitigation Strategy	Monitoring / Response Plan
R1: Data Loss or Corruption	During ingestion or preprocessing, missing or corrupted data files could disrupt the training process or compromise the dataset integrity.	High – Leads to inaccurate predictions and unreliable outputs.	Implement automated data validation scripts, daily backup, and integrity checks (hash comparisons). Use version-controlled datasets with rollback capability (Git + DVC).	Continuous data integrity logging. Alerts triggered by checksum mismatches or abnormal missing-value rates.
R2: Model Drift / Performance Degradation	Over time, economic or behavioral patterns may change, reducing model accuracy (concept drift).	High – Causes unreliable predictions and loss of business trust.	Deploy a monitoring module to track metrics like F1-score and MAE across time. When drift is detected, trigger automated retraining and hyperparameter optimization.	Weekly performance dashboards with threshold-based alerts. Historical metric logs for trend analysis.
R3: Security Breach / Unauthorized Access	Exposure of confidential financial data due to poor access control or misconfigured APIs.	Critical – Legal and ethical implications; data leakage.	Apply role-based access control (RBAC), encryption (AES-256 at rest, HTTPS/TLS in transit), and secure API tokens. Conduct periodic penetration testing.	Monthly security audits, credential rotation, and alerting via centralized logging tools.
R4: Pipeline Failure / Downtime	A failure in one module (e.g., preprocessing or model training) halts the entire pipeline, delaying deliverables.	Medium – Reduces reliability and scalability.	Implement modular fault-tolerance: independent services with restart mechanisms. Use Docker containers with retry logic and logging.	Automated system monitoring (e.g., Prometheus + Grafana). Restart failed containers and notify team via Slack/Email.
R5: Bias or Inconsistent Predictions	Imbalanced or non-representative data may introduce bias, leading to unfair or unstable predictions.	Medium – Ethical and reputational risk.	Include data balance checks and SHAP/feature importance analysis for interpretability. Apply stratified sampling and fairness metrics during validation.	Model fairness monitoring integrated into evaluation reports; regular stakeholder reviews.

2.3 Quality Assurance Approach

To ensure continuous quality improvement and early risk detection, the project follows these core quality assurance (QA) principles:

1. **Preventive Quality Control**: Automated validation and code review pipelines detect inconsistencies early (aligned with *CMMI Level 3* process maturity).
2. **Continuous Integration & Testing**: CI/CD workflows execute unit, integration, and regression tests at each commit to maintain reliability.
3. **Traceability and Documentation**: All experiments, data versions, and model configurations are logged and traceable through GitHub and MLflow.
4. **Quantitative Quality Metrics**: Quality performance is tracked using the following indicators:
   - Mean Absolute Error (MAE) ≤ 0.15
   - Uptime $\geq 99\%$
   - Response Time ≤ 2 s per prediction request

- Data Validation Success Rate $\geq 98\%$

5. **Feedback-Driven Improvement**: Post-deployment monitoring identifies process deviations, which feed into retraining and architecture refinement cycles.

2.4 Monitoring and Response Framework

To operationalize quality control and risk mitigation, the following mechanisms are in place:

- **Automated Monitoring Dashboards:** Built using Grafana and Streamlit for real-time system status visualization.
- **Incident Response Workflow:** Incidents are categorized (Critical / High / Low), logged in GitHub Issues, and resolved within predefined SLAs.
- **Quality Review Meetings:** Weekly team meetings evaluate KPIs and update mitigation strategies.
- **Documentation Updates:** Every system modification triggers an update to the technical documentation and traceability matrix.

2.5 Summary

By embedding robust design principles and applying standardized frameworks such as ISO 9000, CMMI, and Six Sigma, the system ensures predictable behavior, resilience, and long-term sustainability. The combination of automated quality control, continuous monitoring, and proactive risk mitigation minimizes downtime, preserves data integrity, and maintains stakeholder confidence in both the technical and ethical performance of the predictive system.

3 Project Management Plan: Roles and Milestones

Team Roles and Responsibilities

Table 2: Team Roles and Specific Responsibilities

Role	Assigned Team Member	Key Responsibilities
Project Leader	Juan Esteban Avila Trujillo	• Oversees the entire project lifecycle, ensuring alignment with workshop objectives. • Coordinates all team activities, manages communication, and resolves conflicts. • Defines and tracks the project timeline, milestones, and deliverables. • Manages the project scope and maintains the risk register.
Data Analyst	Juan Jose León Gomez	• Leads the Exploratory Data Analysis (EDA) to identify data quality issues and feature interactions. • Designs and implements the data preprocessing and feature engineering pipeline. • Selects, trains, and validates the core predictive models (e.g., LightGBM, XGBoost).
DevOps Engineer (QA)	Juan Pablo Diaz Ricaurte	• Manages the GitHub repository, enforces version control, and manages code merges. • Implements the robust system architecture, focusing on modularity, scalability, and reproducibility. • Ensures the `scikit-learn` pipeline is consistent and reusable (*Transformation Reuse*). • Implements CI/CD (Continuous Integration) checks to validate pipeline integrity.

Table 2: Team Roles and Specific Responsibilities

Role	Assigned Team Member	Key Responsibilities
Tester	Miguel Angel Hernandez Medina	• Develops the master test plan, including unit tests, integration tests, and regression tests. • Writes and executes test cases for each part of the data pipeline (e.g., data ingestion, imputation, normalization). • Validates the logical integrity and accuracy of the model's outputs (`submission.csv`). • Documents all bugs, tracks their resolution, and verifies fixes.

Key Milestones and Deliverables

Table 3: Key Milestones and Project Deliverables

Milestone	Key Deliverables	Status
M1: System Analysis (Workshop 1)	• Systemic Analysis Report (identifying chaos and sensitivity). • High-level system boundary definition.	Completed
M2: High-Level Design (Workshop 2)	• High-Level Architecture Diagram (Modular). • Technical Stack (Python, LightGBM, Pipelines). • Initial Scope and Limitations Definition.	Completed
M3: Robust Design & Project Plan (Workshop 3)	• Refined Robust Architecture Diagram. • Quality and Risk Management Plan. • Project Management Plan (Roles & Milestones). • Incremental Improvement Summary.	In Progress
M4: Pipeline Implementation	• Version 1.0 of the reproducible preprocessing pipeline (Code). • Trained baseline model (LightGBM). • Initial `submission.csv` file generated.	Pending
M5: Final Project Delivery	• Final, optimized predictive model and code. • Final Technical Report documenting all phases. • Complete and documented GitHub repository.	Pending

Project Management Methodology and Tools

To ensure the system's development is managed effectively, iteratively, and transparently, the team adopted the **SCRUM** methodology, supported by specialized tools for collaboration and version control.

SCRUM Methodology and JIRA

SCRUM is an agile framework selected for its strength in managing complex, adaptive problems—a necessity given the chaotic nature identified in the system analysis (M1). The workflow focuses on short iterative cycles (Sprints) to facilitate constant inspection and adaptation. The JIRA platform is used to manage the **Product Backlog**, define **Sprint Goals**, track tasks, and visualize progress using a Kanban board, ensuring transparency among team members.

Advantages of Using SCRUM for this Project:

- **Managing Complexity and Change:** SCRUM's iterative nature allows the team to frequently review the results of model experiments (e.g., changes in feature engineering or model ensemble techniques) and pivot rapidly without significant overhead.
- **Improved Team Transparency:** Daily stand-ups and JIRA's dashboards ensure that the Project Leader and all team members are aware of progress, bottlenecks, and the status of deliverables (M3).
- **Risk Reduction:** By delivering working increments at the end of each Sprint, potential integration issues or major flaws in the predictive pipeline are identified and mitigated early, reducing the risk of catastrophic failure near the final deadline (M5).
- **Focus on MAE Optimization:** The sprint cycle naturally aligns with the need for continuous testing and refinement required to optimize performance against the competition's primary metric (MAE).

Version Control with GitHub

GitHub is the primary tool for collaborative development and ensuring the **reproducibility** of the entire predictive system, aligning directly with the core principles of robust system design (M3). The Quality/DevOps Engineer (QA) manages the central repository, enforcing standardized branching and merging policies.

Advantages of Using GitHub for this Project:

- **Reproducibility (Robustness):** GitHub acts as the single source of truth for the codebase, including the final `scikit-learn` pipeline, model configuration, and scripts. This ensures that any team member (or external reviewer) can reproduce the exact environment and results, directly mitigating the "chaos effect" risk.
- **Collaboration and Traceability:** It enables asynchronous collaboration among the Data Analyst, Tester, and QA Engineer by facilitating parallel work on different features (e.g., imputation vs. ensemble stacking) through feature branches. Every commit provides a traceable history of who made which change and why.
- **Disaster Recovery:** The decentralized nature of GitHub (local and cloud copies) provides redundancy for the codebase, protecting against data loss of the vital pipeline components.
- **Code Quality (QA):** Pull Requests (PRs) allow the Tester and QA Engineer to review code before merging it into the main branch, enforcing quality standards and detecting potential bugs early.

3.0.1 Workflow Implementation (JIRA Kanban)

The team will utilize a Kanban workflow within JIRA to visualize progress and manage the state of each task across the pipeline development lifecycle. This workflow ensures that every code change undergoes strict review and testing before integration, aligning with the project's robustness requirement (M3).

Table 4: Project Workflow States (Kanban Board)

Workflow State	Activity and Quality Gate
To Do	Prioritized list of all required features, bug fixes, and system improvements. Tasks are defined based on the Sprint Goal and the main objective of optimizing the MAE metric.

Table 4: Project Workflow States (Kanban Board)

Workflow State	Activity and Quality Gate
In Progress	A team member is actively working on the task (e.g., coding an imputation strategy or training a model ensemble). The associated GitHub feature branch is opened.
In Progress	The task is complete and the code is submitted for a Pull Request (PR) on GitHub. The **Quality/DevOps Engineer** reviews the code for robustness, modularity, and adherence to coding standards.
Testing	The code has passed review and is deployed to a staging environment. The **Tester** executes unit and integration tests to verify functional requirements and data integrity.
Done	The task has passed all tests and reviews. The code is merged into the main branch, and the Project Leader updates the milestone progress (M3, M4).

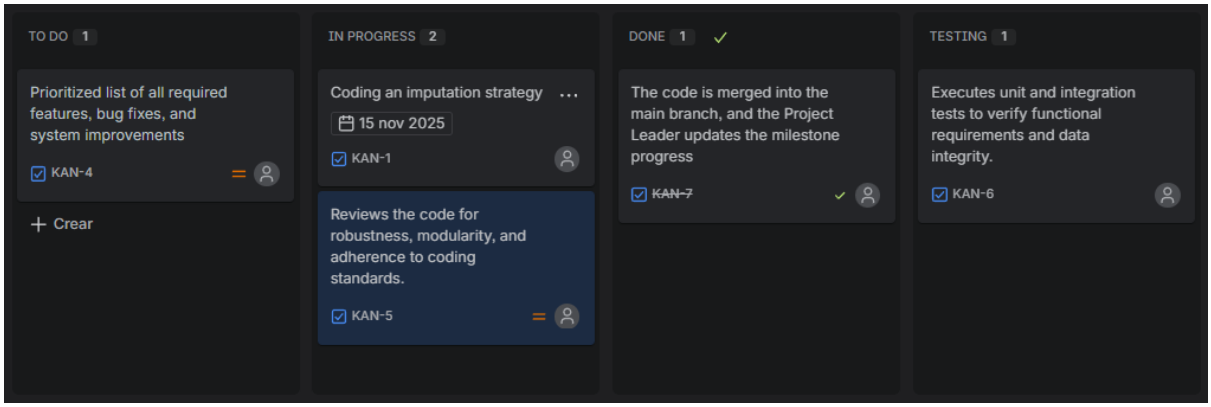


Figure 2: Workflow

4 Incremental Improvements

During the previous workshops, the design of the Loan Default Prediction system has evolved from an exploratory and disorganized approach to a structured, traceable, scalable, and measurable model, following a clear path toward a high-value solution.

Workshop Lessons #1

In the initial analysis process, we understood that the system was not simply about providing a closed solution through a mathematical function, but rather a dynamic ecosystem where:

- The quality of the input data directly influences the accuracy of the model.
- The **F1-Score** metric is fundamental due to the inherent imbalance between paid and defaulted loans.
- The system does not operate with a simple "yes or no" answer, but rather with **probabilities** and expected losses.
- The model variables are interdependent, requiring models capable of capturing nonlinear relationships and consequently increasing the system's complexity.
- The system must function as a **loop**: the output feeds back into the preprocessing stage and the model.

These observations allowed us to structure the complete system chain:

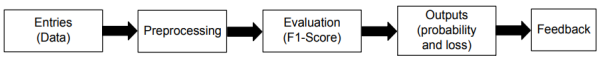


Figure 3: Diagram

They also allowed us to identify current limitations:

- Dependence on internal data.
- Submission limits.
- Poor data quality.

Furthermore, we discussed how implementing the system in a real-world context could lead to financial speculation, affecting data behavior. This workshop gave us a deep understanding of the system, but we still did not have a way to solve it.

Workshop No. 2 Lessons

In the second phase, we moved from understanding the system to designing it correctly:

- A high-level, modular, and scalable architecture was defined.
- Chaotic points were identified (data variability, environment changes), and strategies were established to control them.
- Contingency plans were incorporated (retraining, continuous monitoring).
- Non-functional quality criteria were integrated: transparency, maintainability, scalability, usability, and reliability.
- Different model types were evaluated (LightGBM, XGBoost, Neural Networks), and their selection was justified using experimental evidence.
- The technical implementation was defined based on efficiency and clarity of the pipeline.

This workshop provided us with control, execution capability, and a clear path to follow.

Conclusion

Throughout the workshops, the Loan Default Prediction system evolved from a loosely structured idea to a well-defined, robust, and scalable solution framework. The team progressed from understanding the complexity of the problem to actively managing it through a modular architecture, continuous monitoring, risk management practices, and data-driven model selection.

By incorporating feedback loops, quality assurance strategies, and contingency planning, the system is now able to adapt to data variability, maintain reliability over time, and ensure transparency in both its predictions and its operation. This transformation reflects not only a technical improvement but also the development of a systematic mindset for creating resilient, data-driven solutions.