Workshop N°3 – Robust System Design and Project Management

GEFCom2012 Wind Forecasting System

Tomás Cárdenas Benítez – 20221020021
Diego Angelo Ruano Vergara – 20242020278
Sebastián David Trujillo Vargas – 20242020217
Arlo Nicolás Ocampo Gallego – 20221020104

Universidad Distrital Francisco José de Caldas
Facultad de Ingeniería – Ingeniería de Sistemas

October 2025

1 Review and Refine – System Architecture

1.1 Overview

We refine the layered forecasting architecture from Workshops 1–2 into a robust, fault-tolerant, and observable system that meets the project's non-functional requirements (99% uptime, reproducibility, adaptivity). The refined architecture enforces clear separation of concerns, versioned data and model artifacts, and automated monitoring + failover to ensure continuous operation and safe retraining. This refinement builds on the previously proposed pipeline (data ingestion → feature store → modeling → serving → monitoring) while adding concrete fault domains, scaling points, and traceability controls.

1.2 Design Principles Applied

- **Modularity:** Each layer is independently deployable and testable.

- **Fault isolation:** Failures are confined to components with well-defined fallbacks.
- **Observability:** Metrics, logs and traces are exported from every layer for rapid diagnosis.
- **Reproducibility:** All data transformations, feature versions and model checkpoints are versioned.
- **Security by design:** Encryption at rest/in transit, RBAC for services and audit logging.

These choices align with the course requirements for robustness and standards-aware design.

1.3 Refined Layered Architecture

1. Raw Sources (Input Layer) Historical normalized power (wp1–wp7) and meteorological forecasts (u, v, ws, wd). *Responsibility:* canonical source of truth; maintain immutable raw files and ingestion manifests.

2. Ingestion & Validation Pulls raw files, validates schema/timestamps, applies strict causal-time checks (no future leakage). Produces ingestion manifest with checksums, row counts and integrity flags. *Failure mode:* broken input $\rightarrow$ mark manifest, trigger alert, use last-clean snapshot for short-term serving.

3. Raw Data Lake / Time-series Store Stores raw and cleaned time series; supports snapshots and object versioning (e.g., S3). *Responsibility:* durable, recoverable storage.

4. Processing & Alignment (ETL) Cleaning, imputation, alignment of horizons, generation of deterministic feature transformations (lags, rolling stats, sin/cos(wd)). All transformations are codified in pipelines with DAG + unit tests (e.g., Prefect/Airflow) and produce transformation metadata (version id). Produces reproducible artifacts stored in Feature Store.

5. Feature Store (versioned) Central registry for feature vectors and their versions; supports online (serving) and offline (training) access. *Responsibility:* prevent feature drift by binding training and inference to the same feature version.

6. Model Training / Experimentation Experiment tracking (MLflow or similar), dataset $\rightarrow$ training config $\rightarrow$ model artifact (with metadata:

training data version, seed, hyperparams). Uses rolling-origin cross-validation, checkpointing and automated evaluation (RMSE per farm and horizon).

7. Model Registry & Canary Deployment Registry holds candidate models, baseline persistence model, and production model. Canary testing compares new model vs current production on a small slice before full rollout.

8. Serving / API Layer Exposes forecasts and model metadata via authenticated API endpoints; supports batch and near-real-time requests. Has a configurable fallback to persistence baseline when model or data health fails.

9. Monitoring, Alerting & Retraining Control Observability stack (metrics + logs + traces). Key metrics: RMSE (per horizon/site), data latency, missing features, model inference errors. Automated retraining controller: if RMSE degrades beyond threshold OR data drift detected, trigger retrain pipeline with human-in-the-loop approval for major changes.

10. Dashboard & Audit Interactive dashboard for stakeholders showing forecasts, error heatmaps by horizon/site, feature importance and retraining events. Audit logs for data ingestion, model changes, and user access (security requirement).

11. Operations & Backup Scheduled backups, multi-region replicas of critical data, and documented runbooks for failover to baseline model.

1.4 Fault Tolerance and Scaling Strategies

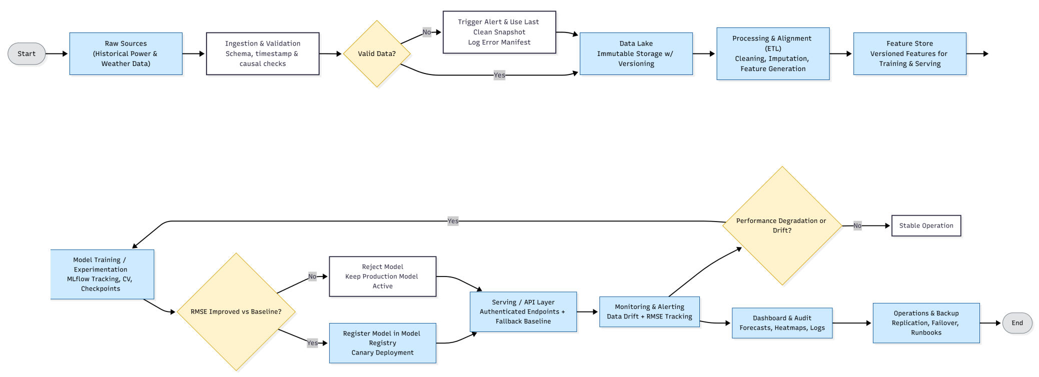
- **Data loss/corruption:** Use object-storage versioning + immutable ingestion manifests; nightly integrity checks and restore from last verified snapshot.
- **Model degradation / drift:** Continuous evaluation (rolling RMSE) with automated alerting; ensemble + weighted blending limits single-model failures.
- **Service outage:** Architecture supports horizontal scaling (stateless API + autoscaled inference workers) and immediate fallback to persistence baseline model for uninterrupted forecasts.

1.5 Mapping to Standards and Quality Controls

- **Reliability & Maintainability:** ISO/IEC 25010 — trackability (versioning), fault-tolerance and recoverability.

- **Security & Compliance:** ISO/IEC 27001 — encryption, RBAC, audit logs for data access and pipeline executions.
- **Process maturity:** Agile + CI/CD flows with code reviews, branch protection, and reproducible artifacts (aligns with CMMI practices described in course materials).

1.6 Architecture Diagram


Start
Raw Sources (Historical Power & Weather Data)
Ingestion & Validation Schema, timestamp & causal checks
Valid Data?
No
Trigger Alert & Use Last Clean Snapshot Log Error Manifest
Yes
Data Lake Immutable Storage w/ Versioning
Processing & Alignment (ETL) Cleaning, Imputation, Feature Generation
Feature Store Versioned Features for Training & Serving
Model Training / Experimentation MLflow Tracking, CV, Checkpoints
RMSE Improved vs Baseline?
No
Reject Model Keep Production Model Active
Yes
Register Model in Model Registry Canary Deployment
Serving / API Layer Authenticated Endpoints + Fallback Baseline
Monitoring & Alerting Data Drift + RMSE Tracking
Performance Degradation or Drift?
Yes
No
Stable Operation
Dashboard & Audit Forecasts, Heatmaps, Logs
Operations & Backup Replication, Failover, Runbooks
End

1.7 Alignment with Quality and Process Standards

The refined architecture is guided by recognized international standards and continuous-improvement frameworks that ensure both technical robustness and organizational maturity:

- **ISO 9000 (Quality Management Systems):** Promotes a process-oriented approach to ensure consistent data and model quality. In the system, each pipeline stage (data ingestion, feature engineering, model training, and deployment) is documented, version-controlled, and validated against explicit acceptance criteria—mirroring ISO 9000's emphasis on quality assurance and traceability.
- **CMMI (Capability Maturity Model Integration):** The architectural refinement follows CMMI Level 3 principles—defined and standardized processes across the project lifecycle. CI/CD pipelines, change-control workflows, and automated testing ensure process discipline,

repeatability, and continuous improvement, aligning with the CMMI model for software process maturity.

- **Six Sigma:** Applied to reduce variation and optimize predictive performance. Model evaluation metrics (e.g., RMSE, MAE) are continuously monitored to detect performance drift and trigger corrective actions, reflecting the Six Sigma DMAIC cycle (Define, Measure, Analyze, Improve, Control).
- **ISO/IEC 25010 and ISO/IEC 27001:** Complement the above by defining measurable quality attributes (reliability, maintainability, security) and enforcing data protection and auditability within the system's architecture.

Together, these standards provide a formal foundation for the architecture's quality, reliability, and continuous improvement, ensuring that technical design decisions are auditable, measurable, and aligned with internationally recognized best practices.

1.8 Concise Summary

The refined architecture organizes the forecasting system into clear, versioned layers: ingestion & validation, persistent raw store, deterministic ETL, a versioned feature store, experiment-driven model training with a model registry and canary deployment, and a monitored serving layer exposing authenticated APIs. Observability (RMSE per horizon/site, data health), reproducibility (feature + model versioning), and secure operations (encryption, RBAC, audit logs) are first-class concerns. The design provides automatic fallback to a persistence baseline, automated retraining triggers under drift, and multi-region backups to guarantee reliability and business continuity.

2 Risk and Quality Management

Risk and quality management in the wind power forecasting system is essential to guarantee the reliability, security, and adaptability of the predictive framework. Given the complex and data-driven nature of the system, several potential risks and failure points have been identified, each requiring mitigation strategies aligned with established standards such as ISO/IEC 25010, ISO/IEC 27001, and NIST SP 800-30.

One major risk concerns **data loss or corruption** within the historical or meteorological datasets. Since the predictive model depends on continuous and temporally coherent data, any inconsistency or missing information could severely affect the quality of forecasts. To mitigate this risk, the system enforces redundant storage solutions, including cloud-based backups and local replicas, ensuring data persistence and recoverability. Additionally, data integrity checks are embedded within the ETL process to detect anomalies, while the feature store maintains version-controlled datasets to prevent feature drift and ensure reproducibility. These practices align with the *Reliability* and *Recoverability* principles of the ISO/IEC 25010 quality model.

A second critical risk involves **model degradation and performance drift**. Due to the dynamic nature of meteorological conditions, the forecasting model may lose accuracy over time as the underlying data distribution evolves. To address this issue, the system integrates continuous performance monitoring through real-time RMSE tracking and automatic retraining triggers when predefined thresholds are exceeded. Ensemble modeling techniques, such as Gradient Boosted Trees and Random Forests, further enhance robustness against random noise and overfitting. The risk management approach follows the NIST SP 800-30 framework, emphasizing continuous evaluation and mitigation of drift-related risks.

The third potential risk arises from **system downtime or security breaches**. As the forecasting architecture operates continuously, any interruption or unauthorized access could compromise service availability and data confidentiality. To prevent this, the system guarantees a 99% uptime through automatic failover to a baseline persistence model during outages. Furthermore, all stored and transmitted data are encrypted, and access is regulated through role-based authentication and detailed activity logging. Periodic security audits and penetration testing are conducted in accordance with ISO/IEC 27001 guidelines, ensuring operational continuity and compliance with data protection standards.

During both development and operation, the system employs a proactive monitoring and response plan. In the development phase, automated testing pipelines validate data integrity, model accuracy, and integration stability before deployment. In the operational phase, the monitoring layer tracks RMSE evolution, data drift, and system performance in real time. Alerts are generated automatically when anomalies are detected, and retraining or failover procedures are executed to maintain forecast reliability. This continuous feedback loop transforms uncertainty into adaptive control, reinforcing both the scientific and operational robustness of the system.

In summary, the proposed risk and quality management framework ensures that the forecasting system remains resilient, reliable, and secure under varying environmental and operational conditions. Through redundant data management, adaptive learning mechanisms, and rigorous adherence to international standards, the system achieves long-term sustainability and dependable predictive performance.

3 Project Management Plan

3.1 Team Roles and Responsibilities

Role	**Member**	**Responsibilities**
Project Manager / Scrum Master	Sebastián David Trujillo Vargas	Oversees coordination, monitors progress, and ensures milestone completion. Facilitates communication and documentation.
Data Analyst / System Architect	Diego Angelo Ruano Vergara	Designs and maintains the data pipeline, ensuring preprocessing integrity and traceability.
Machine Learning Developer	Tomás Cárdenas Benítez	Implements, trains, and tunes predictive models (e.g., XGBoost, RNNs) and evaluates RMSE performance.
DevOps Engineer / Tester	Arlo Nicolás Ocampo Gallego	Manages GitHub repository, version control, deployment, and continuous monitoring of model performance.

Table 1: Team roles and responsibilities.

3.2 Key Milestones and Deliverables

Milestone	**Objective**	**Deliverable**	**Date**
M1 – Data Ingestion & Processing	Clean and align raw wind and weather data.	Preprocessed dataset and ETL scripts	Nov 5, 2025
M2 – Feature Engineering & Modeling	Build predictive models and optimize RMSE vs. persistence baseline.	Trained model and feature store	Nov 6, 2025
M3 – Evaluation & Backtesting	Validate model performance using rolling-origin testing.	Evaluation report	Nov 7, 2025
M4 – Serving, Monitoring & Feedback	Deploy model, implement monitoring and retraining feedback loop.	Final deployed system and monitoring logs	Nov 8, 2025

Table 2: Project milestones and deliverables.

3.3 Project Management Methodology and Tools

The team follows an **Agile Scrum** methodology, emphasizing iteration, collaboration, and **rapid feedback**. This approach is fundamental for implementing continuous monitoring and *retraining cycles* required to manage the chaotic and sensitive nature of wind forecasting.

The main project management tools and practices include:

- **GitHub Projects (Kanban Board):** Used to visualize the modular workflow across the MLOps layers (Data Ingestion, Modeling, Evaluation) and ensure proper task tracking.
- **Google Sheets (Gantt Chart):** Enables visual scheduling and tracking of key milestones (M1 to M4) within the defined project timeline (Nov 5–Nov 8, 2025).
- **Git + Branch Protection:** Essential for **Reproducibility**. Ensures consistent version control of code and the *Feature Store*, preventing *feature drift* [**?**, **?**].
- **Discord / WhatsApp:** Used for internal coordination and rapid communication among team members.

- **Google Docs:** Facilitates collaborative documentation and collective editing of the final report.

3.4 Project Timeline and Workflow Diagram

The project workflow, shown in Figure 1, represents the complete process from data ingestion to deployment and monitoring. Each milestone (M1–M4) corresponds to a project phase. Model performance is continuously evaluated using RMSE against the persistence baseline; if it falls below the 10% improvement threshold, retraining is automatically triggered, creating a self-correcting feedback loop that ensures robustness and quality.

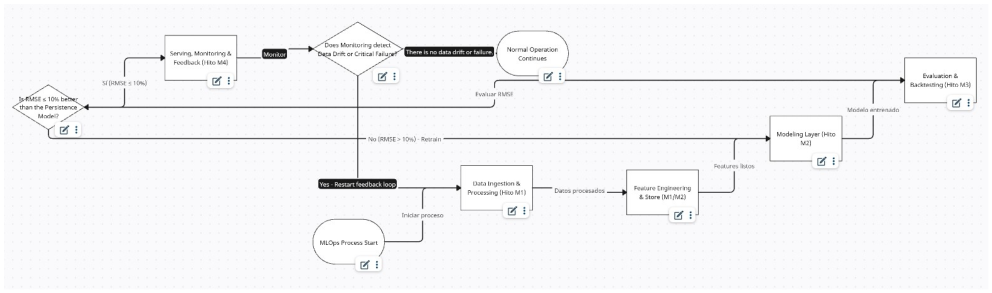

Serving, Monitoring & Feedback (Hito M4)
Monitor
Does Monitoring detect Data Drift or Critical Failure?
There is no data drift or failure.
Normal Operation Continues
Evaluation & Backtesting (Hito M3)
Sí (RMSE ≤ 10%)
Is RMSE ≤ 10% better than the Persistence Model?
Evaluar RMSE
Modelo entrenado
Modeling Layer (Hito M2)
No (RMSE > 10%) - Retrain
Features listos
Yes - Restart feedback loop
Data Ingestion & Processing (Hito M1)
Datos procesados
Feature Engineering & Store (M1/M2)
Iniciar proceso
MLOps Process Start

Figure 1: Project workflow integrating milestones M1–M4 and feedback monitoring.

4 Incremental Improvements

4.1 Lessons Learned from Workshops 1 and 2

In the first workshop, the team learned that wind forecasting is very complex and unpredictable. They found that even tiny changes in wind speed or direction could cause big, different results in the power forecast. This showed they needed stronger methods to prepare their data and use multiple models together to handle all the uncertainty and "noise." They also mapped out the system's key parts (like wind farms, weather data, and models) and figured out how they all affect each other. Building on these insights, Workshop 2 translated the analytical findings into a modular system architecture. This version formalized the pipeline into distinct layers: data ingestion, processing and alignment, feature storage, modeling, postprocessing, evaluation, and monitoring. Each layer was assigned a well-defined role to ensure scalability, resilience, and feedback control, turning sensitivity analysis into actionable design principles. The team also established functional and non-functional requirements, emphasizing adaptivity, interpretability, and system uptime, and introduced feedback-driven retraining as a direct response to chaotic and stochastic conditions.

4.2 Evolution of System Design and Management Plan

Because of feedback from these workshops, the team changed its approach. They moved from using fixed models to building a system that can learn and adapt on its own. The new design uses its own mistakes to get better. When a prediction is wrong, the system automatically uses that error to retrain and update itself. This changed the system from just dealing with errors to fixing itself, which is a key part of any smart, independent system. The management plan was also refined in response to the lessons learned. Early project stages lacked explicit coordination between analytical, design, and implementation tasks. Workshop 2 introduced structured version control, modular task distribution, and milestone tracking, aligning the workflow with system layers and ensuring traceability across iterations. This iterative management approach—supported by documentation and simulation results—allowed the team to implement continuous improvement cycles similar to agile and feedback-based development.