Universidad Distrital Francisco Jose de Caldas

Workshop 3

Systems Analysis: Global Energy Forecasting Competition 2012
Load Forecasting

Autores:

David Santiago Téllez Melo – 20242020107
Ana Karina Roa Mora – 20232020118
Daniela Bustamante Guerra – 20241020131
Andrés Felipe Correa Méndez – 20221020141

Fecha: 8 de noviembre de 2025

1. System Architecture and Robust Design

1.1 Review and Update of the System Architecture

The electric load forecasting system was designed following a modular and layered structure, which allows maintaining a clear and easy-to-update workflow. The new version of the diagram reflects five main modules that communicate with each other to ensure an organized and reliable process.

Data ingestion and validation: receives information sources (weather, historical load, and calendar). It is responsible for checking data quality, correcting errors, and maintaining temporal consistency. This avoids failures in the following stages and improves system stability.

Processing and feature generation: cleans the data and creates new predictive variables (such as day of the week, seasons, or temperature lags) that help the model recognize patterns.

Training and validation: here the MLP and LSTM models are applied. The MLP handles tabular data, and the LSTM handles temporal sequences. Both are trained with rolling-origin validation, which simulates real forecasting conditions.

Forecast generation: produces predictions by zone and for the total system, ensuring hierarchical consistency between both.

Evaluation and monitoring: measures performance with metrics such as RMSE and MAPE, and activates alerts or retraining when a drop in accuracy is detected.

This modular design allows each part to be improved or replaced without affecting the rest, ensuring greater reliability and easier maintenance.

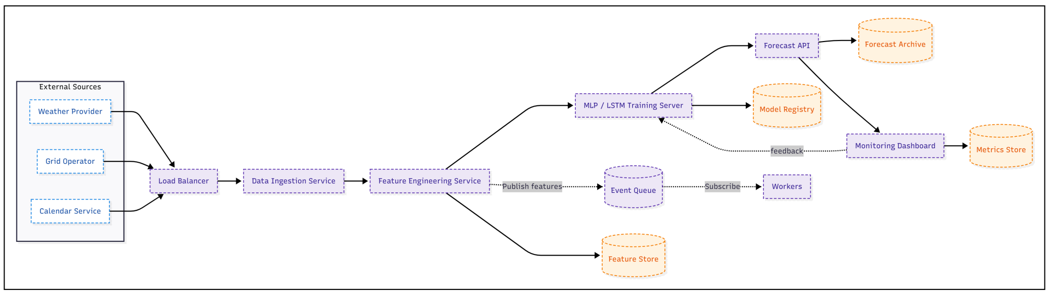

Figura 1: Figure 1: Updated System Architecture Diagram

1.2 Reliability, Scalability, and Maintainability

Each component of the architecture directly contributes to the robustness of the system.

Reliability: achieved through continuous data validation, automatic backups, and version control of the models. This ensures that the system can recover from errors or defective information.

Scalability: the layered structure allows handling more data, zones, or models without modifying the general flow. In addition, tasks can be executed in parallel, speeding up processes.

Maintainability: thanks to the separation of functions (data, features, models, and evaluation), any change or update can be made without affecting other parts. Internal documentation and clear naming make collaborative work easier.

Usability: process automation (from data loading to evaluation) reduces human errors and makes the system easier to use and understand, even for new team members.

Together, these aspects ensure that the system is stable, adaptable, and ready for future growth.

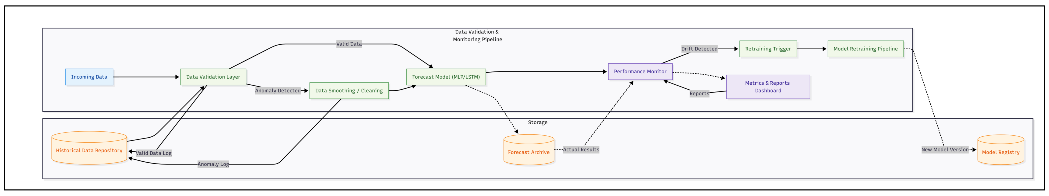

Figura 2: Figure 2: Robustness, Scalability and Quality Diagram

1.3 Good Practices and Quality Standards

During the redesign, engineering best practices were applied to strengthen system robustness.

Modularity: each component can be tested and replaced without affecting the others.

Fault tolerance: automatic validations and redundant storage are applied to prevent information loss.

Redundancy: important data and results are saved in multiple copies to ensure continuity in case of failures.

These practices are aligned with recognized quality standards such as ISO 9001 (process management), ISO 25010 (software quality), CMMI (maturity and continuous improvement), and Six Sigma (error and variability reduction).

Applying these principles improves system stability, facilitates testing, and maintains forecast accuracy even when external conditions change or the amount of data increases.

In summary, this version of the system combines a clean architecture with good engineering practices, achieving a reliable, maintainable, and adaptable solution, ideal for both academic and real-world scenarios in the energy industry.

2. Quality and Risk Analysis

2.1 Risk Identification and Mitigation

Based on the comprehensive system architecture and technical stack defined in our previous workshops for the GEFCom2012 electricity load forecasting system, we have identified three critical risks that align with the sensitive and chaotic nature of energy forecasting systems.

1. Data Integrity and Availability Risk

Given the hierarchical load forecasting system's dependence on continuous historical load data (20 zones + aggregated system) and real-time weather information from multiple stations, any data corruption, loss, or unavailability would severely compromise forecast accuracy. This risk is particularly critical given the nonlinear temperature-load relationships and hierarchical consistency requirements identified in Workshop #1.

Mitigation Strategies:

- Implement automated data validation checks at ingestion points with rolling statistical analysis
- Establish regular encrypted backups for historical datasets using version control
- Create data quality monitoring with alert thresholds for missing values and anomalies
- Develop fallback mechanisms using cached seasonal patterns when real-time sources fail
- Implement data versioning to track changes and enable rollbacks to stable states

2. Model Performance Degradation Risk

The MLP/LSTM models may experience performance decay due to changing electricity consumption patterns, evolving weather conditions, or the chaotic elements identified in our systemic analysis. This risk directly impacts the RMSE and MAPE metrics critical for competition success.

Mitigation Strategies:

- Implement continuous model performance monitoring with automated retraining triggers

- Establish A/B testing framework for model updates using rolling-origin cross-validation
- Maintain model versioning with rollback capabilities in the model registry
- Set up ensemble methods combining multiple model predictions to smooth variability
- Implement drift detection algorithms for early warning of changing patterns

3. System Integration and Dependency Risk

The Clean ML architecture relies on PyTorch, pandas, scikit-learn, and external weather APIs. Version conflicts, API changes, or service disruptions could cause system failures, especially given the temporal complexity and sensitivity constraints identified in our analysis.

Mitigation Strategies:

- Use dependency pinning and containerized virtual environments
- Implement comprehensive unit and integration testing for all pipeline stages
- Create mock services for external API dependencies during development
- Establish dependency update protocols with testing requirements
- Maintain compatibility matrices for library versions across the technical stack

2.2 Risk Analysis Table

Cuadro 1: Risk Analysis and Mitigation Plan for Electricity Load Forecasting System

Risk Name	Impact	Probability	Mitigation Strategies	Control Owner
Data Integrity & Availability	**High**	**Medium**	• Automated data validation • Regular encrypted backups • Quality monitoring with alerts • Fallback mechanisms • Data versioning	Data Engineer
Model Performance Degradation	**High**	**High**	• Performance monitoring • Automated retraining • Model versioning • Ensemble methods • Drift detection	ML Engineer
System Integration & Dependencies	**Medium**	**Medium**	• Dependency pinning • Comprehensive testing • Mock services • Update protocols • Compatibility matrices	DevOps Engineer

2.3 Incident Monitoring and Response Framework

The team will implement a comprehensive incident management process aligned with the modular pipeline architecture and systems engineering principles established in Workshop #2:

Real-time Monitoring Framework:

- Continuous monitoring of system metrics (CPU, memory, disk I/O) across all pipeline stages
- Automated alerting for forecast performance degradation (RMSE/MAPE increases beyond 15 % threshold)
- Data quality dashboards with anomaly detection for load and temperature inputs
- Model performance tracking with statistical control limits and temporal validation
- Hierarchical consistency monitoring to detect aggregation errors between zones

Incident Response Protocol:

- Establish severity classification (P0-P3) based on forecast accuracy impact
- Define escalation procedures and on-call rotations for data engineers and ML engineers
- Maintain incident runbooks for common failure scenarios in the forecasting pipeline
- Conduct post-incident reviews with root cause analysis focusing on systemic sensitivities
- Implement continuous improvement based on lessons learned from chaotic patterns

Quality Assurance Integration:

- Regular model validation against GEFCom2012 benchmark forecasts
- Cross-validation using temporal splits that respect the competition's evaluation methodology
- Statistical testing for data distribution changes in load and temperature inputs
- User acceptance testing for new features in the educational interface
- Compliance with ISO 9000 quality guidelines for documentation and process standardization

The incident response team will follow ITIL-based practices integrated with our Clean ML architecture, ensuring systematic handling of issues from detection through resolution while maintaining the hierarchical consistency and temporal accuracy required for competitive forecasting performance.

3. Project Management Plan

3.1 Team Roles and Responsibilities

The project team is composed of four members, each assigned to complementary roles that reflect the interdisciplinary nature of the *Educational Electricity Load Forecasting System* inspired by the GEFCom2012 competition. The allocation of roles is random but balanced, ensuring every key aspect of system design and implementation is covered:

- **David Santiago Téllez Melo – Project Manager & Analyst:** Responsible for planning, scheduling, and coordinating activities. Ensures the system meets educational and technical objectives while maintaining alignment with the Clean ML philosophy and course deadlines.
- **Ana Karina Roa Mora – Developer:** Leads system implementation in Python and PyTorch. Integrates forecasting models (MLP/LSTM) and manages data pipelines, ensuring robustness and reproducibility.
- **Daniela Bustamante Guerra – Tester & Documentation Lead:** Conducts testing, verification, and report preparation. Ensures all deliverables comply with system requirements and user-centered design guidelines.
- **Andrés Felipe Correa Méndez – Data Engineer & Integration Specialist:** Handles data ingestion, preprocessing, and feature engineering. Maintains dataset consistency and supports hierarchical reconciliation mechanisms.

These roles are necessary to maintain quality and coherence between analysis, development, testing, and integration tasks. They mirror real-world load forecasting team dynamics, where domain, modeling, and validation responsibilities must be distributed.

3.2 Methodology and Management Approach

Given the academic timeframe and the iterative nature of model design, a **Scrum–Kanban hybrid** methodology was selected. This approach combines the adaptability of Scrum with the visual control and flow optimization of Kanban.

Justification:

- Scrum allows short and well-defined sprints (weekly sessions aligned with class meetings).
- Kanban supports visual monitoring of parallel tasks (data cleaning, modeling, documentation).
- The hybrid approach minimizes overhead and improves transparency within an academic setting.

Each sprint corresponds to one week, with milestones reviewed during Monday and

Saturday sessions. Tasks are managed using a shared Trello board, integrating GitHub commits and documentation updates.

3.3 Milestones and Deliverables

Milestone	**Deliverable / Description**	**Responsible**	**Deadline**
System Architecture Refinement	Update diagrams with robust design (ISO-based).	Team	Nov 11, 2025
Quality and Risk Report	Risk matrix, mitigation strategies.	Daniela	Nov 15, 2025
Integration of Feedback Loops	Implement model validation and retraining triggers.	Andrés	Nov 18, 2025
Testing and Documentation	Error metrics, testing logs, final PDF draft.	Ana & Daniela	Nov 25, 2025
Final Presentation Preparation	Slides, GitHub update, and demo run.	Santiago	Dec 6, 2025

Cuadro 2: Key milestones and deliverables for Workshop 3.

3.4 Schedule and Class Timeline

The project schedule follows the university's academic sessions, with team meetings every Monday and Saturday until December 6th, 2025. National holidays in November (November 4 and November 11) are considered non-working days.

Week	Dates	Main Tasks	Output
1	Oct 28 – Nov 2	Define roles, setup tools, backlog creation	Trello and GitHub ready
2	Nov 3 – Nov 9	Architecture updates, Scrum board setup (Nov 4 holiday)	Updated system diagrams
3	Nov 10 – Nov 16	Risk management plan (Nov 11 holiday)	Risk report
4	Nov 17 – Nov 23	Model validation and testing	Evaluation metrics
5	Nov 24 – Nov 30	Documentation and review	Final report draft
6	Dec 1 – Dec 6	Presentation and submission	Final PDF & GitHub update

Cuadro 3: Weekly schedule including holidays and deliverables.

3.5 Tools and Collaboration Environment

- **Trello:** Task management and sprint board (Scrum–Kanban hybrid visualization).
- **GitHub:** Version control, collaboration, and issue tracking for the Clean ML codebase.
- **Google Colab / Jupyter:** Development and model experimentation in Python and PyTorch.
- **Overleaf:** Collaborative writing and LaTeX report editing.
- **Google Meet / WhatsApp:** Communication and coordination between Monday and Saturday sessions.

These tools ensure traceability, transparency, and version consistency throughout the development cycle, supporting both technical and educational goals.

3.6 Workflow Diagram

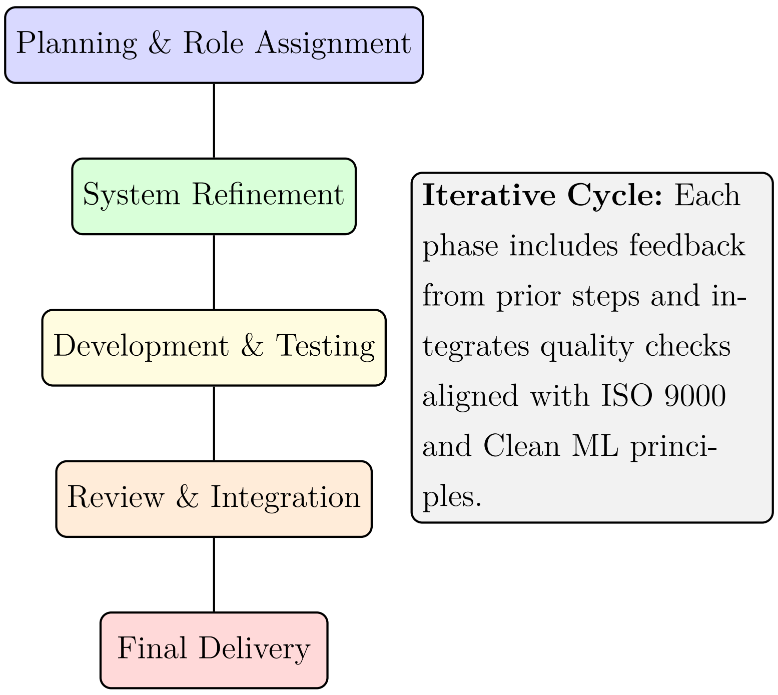
Planning & Role Assignment
System Refinement
Development & Testing
Review & Integration
Final Delivery
Iterative Cycle: Each phase includes feedback from prior steps and integrates quality checks aligned with ISO 9000 and Clean ML principles.

Figura 3: Project management workflow.

3.7 Method Justification

The selected methodology ensures alignment between academic constraints and professional practices. It provides:

- Clear accountability for each member.
- Continuous monitoring of progress via Trello and GitHub.
- Adaptability for unforeseen delays or data issues.
- Iterative improvement and real-time risk control.

The hybrid Scrum–Kanban model fits the educational context by fostering collaboration and maintaining consistent project velocity within the limited workshop timeframe.

4. Feedback and System Evolution

The first two workshops provided essential feedback that guided the refinement of the system's architecture and management plan. Three main lessons were identified:

1. **Need for clear modularity:** The initial prototype lacked separation between data, modeling, and evaluation layers. The current version adopts a *Clean ML* modular design with five independent modules—data ingestion, feature generation, training, forecasting, and monitoring—allowing easy updates and testing.

2. **Reproducibility and robustness:** Earlier versions had limited version control and validation. The new pipeline integrates automatic data validation, dependency management, and versioning to ensure consistency and error recovery.
3. **Project organization:** Previous workshops used informal coordination. A *Scrum–Kanban* hybrid method was introduced, defining clear roles, weekly milestones, and collaborative tools (Trello, GitHub, Overleaf) to improve transparency and continuous improvement.

Overall, the system evolved from a conceptual framework into a reliable, scalable, and maintainable forecasting platform, integrating quality standards (ISO 9001, CMMI) and systematic feedback loops that strengthen both the technical and management aspects of the project.