Universidad Distrital Francisco José de Caldas

Systems analysis and desing GR020-85

Workshop 3
Robust System Design and Project Management

Universidad Distrital Francisco José de Caldas

Systems analysis and desing GR020-85

Workshop 3

Competition: Web Traffic Time Series Forecasting

Members:

Johan Sebastián Beltrán Merchán - 20222020019
Edison David Alvarez Varela - 20222020043
Yader Ibraldo Quiroga Torres - 20222020034
Julián David Celis Giraldo - 20222020041

Lecturer:

Carlos Andrés Sierra Virgüez

Bogotá, Colombia
November 6, 2025

Chapter 1

Robust System Design and Architectural Refinement

This section refines the **Modular Monolith** architecture proposed in Workshop 2, emphasizing **robust engineering principles** to meet quality standards (e.g., ISO 9000, Six Sigma).

1.1 Refinement for Quality and Scalability

The system relies on two design patterns: the **Chain of Responsibility (CoR)** for data integrity (**NFR3: Reliability**) and the **Hierarchical Ensemble** (**Strategy Pattern**) for adaptive prediction (**NFR4: Accuracy**).

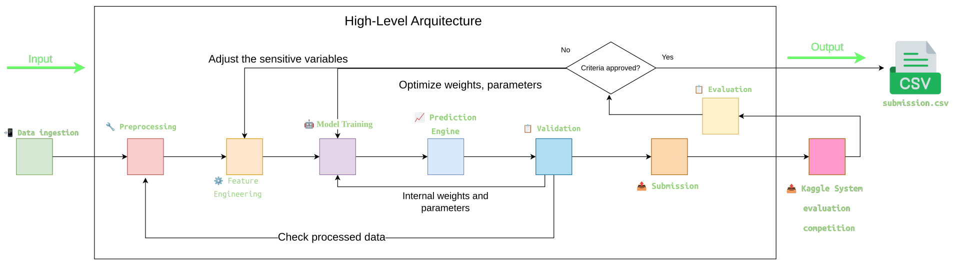

Figure 1.1: Robust System Architecture.

1.1.1 Principle 1: Modularity and Maintainability (ISO 9000)

The architecture aligns with **ISO 9000** by standardizing modules, ensuring that components can be individually maintained and tested.

- **Strategy Encapsulation:** Each forecasting strategy (**ARIMAStrategy, ProphetStrategy, Advanced ML Strategy**) is interchangeable and uses a consistent interface, allowing easy model rotation or upgrading without affecting the core prediction logic (**FR2**).

- **Decoupling:** The **Meta-Model** is decoupled from the actual forecasting implementation, focusing solely on the selection logic based on contextual metadata (language, volatility).

1.1.2 Principle 2: Fault-Tolerance and Data Integrity

Fault-tolerance is essential to combat the identified **High Sensitivity** and **Zero-Value Ambiguity** risks.

- **Chain of Responsibility (CoR) Enforcement:** The sequential nature of the CoR ensures that critical preprocessing steps (e.g., bot traffic filtering, zero-value imputation) are **mandatory** and executed sequentially, acting as a quality gate (**NFR3**).
- **Cascading Fallback:** Within the **Prediction Engine**, if a complex strategy (e.g., LSTM) fails to converge or times out (addressing **NFR2: Scalability** constraint), the system must immediately invoke a simpler, guaranteed strategy (e.g., Naive or Seasonal Naive) to prevent prediction failure.

Chapter 2

Quality and Risk Analysis

2.1 Risk Identification and Mitigation

We analyze three high-priority risks that threaten the system's ability to maintain a competitive **SMAPE** score and operational availability.

2.1.1 Risk 1: Model Drift (Dynamic Chaos Invalidation)

- **Description:** The underlying patterns of Wikipedia traffic shift (e.g., a major change in Google's search algorithm or a long-term global event), causing the Meta-Model's selection logic to become obsolete and predictions to fail against **Dynamic Chaos**.
- **Impact: High**. Direct degradation of the primary metric (**SMAPE**).
- **Mitigation Strategy:** Implement continuous **Drift Monitoring** via the **Stratified Post-Prediction Analysis (FR4)**, if the stratified SMAPE for a specific subgroup (e.g., 'es.wikipedia.org_mobile') exceeds a **15%** threshold, the system triggers an alert for mandatory re-training and recalibration of the Meta-Model's weights for that specific segment.

2.1.2 Risk 2: Resource Overload in Parallel Training

- **Description:** The parallel execution of **145,000** time series jobs using **Joblib** (**NFR2**) leads to resource contention, memory errors, or excessive runtimes when deploying memory-intensive models like **LSTM/Random Forest**.
- **Impact: Medium-High**. Can cause partial system failure or missed submission deadlines.
- **Mitigation Strategy:** Implement **Resource Throttling and Timeouts**. All training jobs must be capped by CPU and Memory limits. If a complex job exceeds a predefined time limit (e.g., 5 minutes), the process is forcefully terminated, logged, and the series prediction falls back to the simpler, pre-calculated ARIMA model

(Fault-Tolerance). The usage of **Dask** over pure Joblib will be explored for better distributed resource management.

2.1.3 Risk 3: Feature Leakage during Feature Engineering

- **Description:** Information from the future (the target prediction period) inadvertently leaks into the features used for training, leading to unrealistically low SMAPE during internal validation but catastrophic failure upon external submission.
- **Impact: High**. False confidence leading to total project failure in the final Kaggle submission.
- **Mitigation Strategy:** Rigorous adherence to **Temporal Validation**. The **Feature Engineering** module (**FR3**) must be unit-tested to ensure that all lag and temporal features are calculated using data strictly $\mathbf{t-1}$ or earlier. This requires code review and specialized integration testing to validate the **Chain of Responsibility** sequence for time-dependent data.

Chapter 3

Project Management Plan

3.1 Team Roles and Responsibilities

The team is structured to cover the full lifecycle, from architecture to testing, ensuring clear accountability.

- **Project Manager (Julián D. Celis Giraldo):** Timeline, resource allocation, risk tracking, and external communication.
- **System Architect (Johan S. Beltrán Merchán):** CoR integrity, design pattern adherence, NFR compliance (**Scalability, Reliability**).
- **Data Scientist/Model Developer (Edison D. Álvarez Varela):** Implementation of Base Strategies (ARIMA, Prophet, LSTM), Feature Engineering logic, and Meta-Model weights tuning.
- **Quality Assurance / Tester (Yader I. Quiroga Torres):** Development of the **Validation Module (FR4)**, SMAPE analysis, Unit/Integration testing, and execution of the Post-Prediction Analysis.

3.2 Methodology and Timeline

We adopt a **Scrum/Kanban Hybrid** methodology. **Scrum** (2-week sprints) drives the model implementation (**FR2**), while **Kanban** manages the issue backlog (bugs, stratified errors).

M1. **M1: Data Pipeline Hardening (CoR Completion):** Finalize **FR1** (Data Management) and **FR3** (Feature Engineering), including Joblib parallelization setup and all integrity checks.

M2. **M2: Hierarchical Ensemble Core:** Complete implementation of all three Strategies (Level 1) and the Meta-Model selection logic (Level 2).

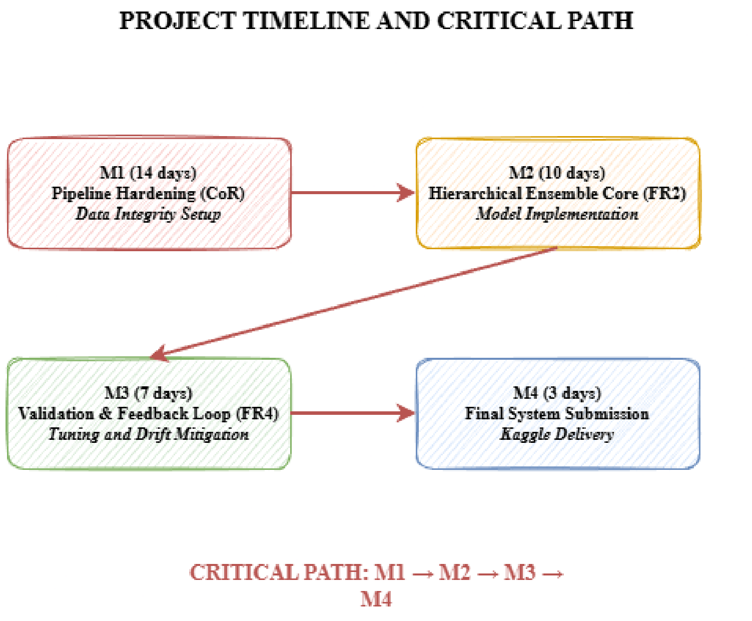
PROJECT TIMELINE AND CRITICAL PATH
M1 (14 days)
Pipeline Hardening (CoR)
Data Integrity Setup
M2 (10 days)
Hierarchical Ensemble Core (FR2)
Model Implementation
M3 (7 days)
Validation & Feedback Loop (FR4)
Tuning and Drift Mitigation
M4 (3 days)
Final System Submission
Kaggle Delivery
CRITICAL PATH: M1 → M2 → M3 → M4

Figure 3.1: Project Timeline and Milestone Dependencies

M3. **M3: Validation and Feedback Loop:** Full implementation and successful execution of the **Validation Module (FR4)** and the **Stratified Post-Prediction Analysis**.

M4. **M4: Final System Submission:** Integration testing, final hyperparameter optimization, and submission of the optimized forecast predictions to the Kaggle competition.

Chapter 4

Incremental Improvements and Evolution

This section documents the evolution of the system in response to feedback and lessons learned.

4.1 Evolution based on Workshop Feedback

The system's design choices were consistently validated by previous workshop findings, particularly regarding the inherent complexity of the data.

- **From W1 (Constraint):** The identification of **Massive Scale** and **Structural Complexity** led to the architectural decision to adopt parallel processing (**Joblib/Dask**) and the adaptive **Hierarchical Ensemble**, respectively.
- **From W2 (Design):** The design of the system's core feedback mechanism was refined. Instead of a simple pass/fail, the **Stratified Post-Prediction Analysis (FR4)** was formalized. This ensures that the Meta-Model learns not just from overall error, but from specific segment failures (e.g., errors isolated to 'mobile-spanish' articles), making the system's adaptation much more targeted and effective.

4.2 System Management Evolution

The management approach evolved to explicitly address quality and risk:

- **Reliability Integration:** The theoretical design of the pipeline was hardened by explicitly naming the **Chain of Responsibility** as the mechanism for **NFR3** compliance. This makes the system's reliability testable (via integration tests on the CoR).
- **Proactive Risk Management:** The identification of **Model Drift** (**Risk 1**) led to the integration of continuous monitoring thresholds (**NFR3** alignment with Six

Sigma standards for deviation control) into the **Validation Module (FR4)**, transforming the module into a proactive risk-mitigation tool rather than just a final evaluation step.

4.3 Improvements

- In previous workshops, we understood that system design is a complex process that demands rigorous analysis, critical reflection, and a deep understanding of the problem to be solved. This process is not limited to defining a technical structure; it also involves understanding the environment, the factors that affect it, and the interrelationships between the various components that make up the system. Thanks to the knowledge acquired, the team has been able to identify the sensitive elements, inputs, processes, outputs, and feedback mechanisms that characterize the system. This has allowed us to develop a coherent conceptual architecture that not only synthesizes the solution but also seeks to faithfully represent the real dynamics of the problem. In this sense, the design has been oriented toward a modular, scalable, and adaptable structure, capable of responding to changing scenarios and integrating future improvements without compromising the overall stability of the system.

- The design process has evolved significantly along with its management plan, strengthening the team's collective understanding of all the elements involved in the project. This progress is reflected in the definition of a comprehensive architecture that considers as many variables as possible to ensure the system accurately and efficiently replicates reality. The management plan, in turn, establishes clearly defined roles, assigned based on each team member's strengths, as well as a work methodology aligned with the project's pace and demands. Appropriate technical and organizational tools have also been established to facilitate communication, version control, results validation, and continuous improvement. Together, these elements form a robust framework that not only guides system development but also fosters collaboration, quality, and project sustainability throughout its various stages.