Universidad Distrital Francisco José de Caldas
Systems Engineering Department

Designing a Robust Forecasting System for Wikipedia Web Traffic: A Systems Engineering Approach

Johan S. Beltrán Merchán, Edison D. Álvarez Varela,

Yader I. Quiroga Torres, Julián D. Celis Giraldo

Supervisor: Carlos Andrés Sierra Virgüez

Course Project - Systems Engineering
Universidad Distrital Francisco José de Caldas
December 12, 2025

Declaration

The authors, Johan Sebastián Beltrán Merchán, Edison David Álvarez Varela, Yader Ibraldo Quiroga Torres, and Julián David Celis Giraldo, of the Department of Systems Engineering, Universidad Distrital Francisco José de Caldas, confirm that this report represents their original work and that all figures, tables, equations, code snippets, artworks, and illustrations are original and have not been taken from any other person's work, except where the works of others have been explicitly acknowledged, quoted, and referenced. It is understood that failure to do so will be considered a case of plagiarism. Plagiarism is a form of academic misconduct and will be penalised accordingly.

The authors give consent to a copy of this report being shared with future students as an exemplar.

The authors give consent for this work to be made available more widely to members of UoR and the public with interest in teaching, learning and research.

Abstract

This report details the implementation of a robust, scalable, and adaptive architecture to minimize the Symmetric Mean Absolute Percentage Error (**SMAPE**) for forecasting daily web traffic across 145,000 Wikipedia articles. The system, structured as a **Modular Monolith**, is anchored by two key systems: **Data Flow Integrity**, ensured via the **Chain of Responsibility** pattern, and **Adaptive Forecasting**, achieved through a **Hierarchical Ensemble** utilizing the **Strategy Pattern**. This ensemble dynamically selects the optimal model (including traditional time series and advanced neural network strategies) per article, addressing the dataset's high scale and heterogeneity. Scalability relies on `joblib` for parallel processing across the 145,000 series. This first version of the final report validates the architectural integrity and provides the definitive blueprint for the competitive prediction system.

Acknowledgements

The authors wish to express their gratitude to Professor **Carlos Andrés Sierra Virgüez** for his invaluable guidance, technical expertise, and support throughout the design and development of this project within the Systems Analysis and Design course. We also thank our colleagues for their collaboration and critical feedback.

Contents

Declaration i

Abstract ii

Acknowledgements iii

A Introduction 1
- A.1 Background and Motivation . . . 1
- A.2 Problem Statement . . . 1
- A.3 Research Objectives . . . 1
- A.4 Scope, Assumptions, and Limitations . . . 2

B Literature Review 3
- B.1 Forecasting Models for Web Traffic . . . 3
- B.2 Ensemble Methods and Systems Engineering . . . 3

C System Architecture and Methodology 4
- C.1 System Architecture: Modular Monolith . . . 4
  - C.1.1 System A: Data Flow Integrity (Chain of Responsibility) . . . 4
- C.2 System B: Adaptive Forecasting (Hierarchical Ensemble) . . . 4
  - C.2.1 Level 1: Base Models (Strategies) . . . 5
  - C.2.2 Level 2: Meta-Model Selection . . . 5
- C.3 Scalability Implementation . . . 5
  - C.3.1 Data ingestion: . . . 5
  - C.3.2 Data Preprocessing . . . 6
  - C.3.3 Feature Engineering . . . 6
  - C.3.4 Model training . . . 6
  - C.3.5 Prediction engine . . . 6
  - C.3.6 Validation . . . 6
  - C.3.7 Submission . . . 7
- C.4 Chain of Responsibility Implementation Diagrams . . . 7

D Discussion 8
- D.1 Link to Objectives . . . 8
- D.2 Implications and Limitations . . . 8

D.3 State-of-the-Art: Arturus' Sequence-to-Sequence Solution 8
D.3.1 Architectural Framework 9
D.3.2 Implementation Pipeline 9
D.3.3 Implications for Future Development 9

E Conclusions and Future Work **10**
E.1 Conclusions 10

F Project Reflection **11**
F.1 Lessons Learned 11
F.2 Challenges and Implementation Barriers 11
F.3 Critical Insights 12

List of Figures

C.1 High-Level Conceptual Architecture of the System, showing the data flow and the adaptive ensemble. The Chain of Responsibility governs the data pipeline, and the Hierarchical Ensemble (Meta-Model selecting strategies) governs the prediction logic. 5
C.2 First part of the Chain of Responsibility modules: Data Ingestion, Preprocessing, and Feature Engineering stages. 7
C.3 Second part of the Chain of Responsibility modules: Model Training, Prediction, Validation, and Submission stages. 7

List of Tables

Chapter A

Introduction

A.1 Background and Motivation

Forecasting time series data is a critical task in numerous fields, but it becomes a unique systems engineering challenge when dealing with massive scale and inherent volatility. The subject of this report is the development of a robust system for predicting the daily visits to over 145,000 Wikipedia articles. This dataset is characterized by a combination of strong temporal patterns (seasonality) and highly unpredictable "Chaos Factors," such as viral news or algorithmic changes, leading to high noise and sensitivity, necessitating the integration of advanced machine learning models for competitive accuracy.

A.2 Problem Statement

The core problem is minimizing the Symmetric Mean Absolute Percentage Error (SMAPE) across an extremely **heterogeneous** collection of time series, where no single forecasting model (e.g., ARIMA, Prophet, Neural Networks) performs optimally for every article. Furthermore, the solution must be **scalable** to handle the massive data volume and **maintainable** to adapt to evolving web traffic behavior.

A.3 Research Objectives

The primary objective of this project is to answer the question: *How can a scalable and maintainable system architecture minimize SMAPE across heterogeneous time series using Systems Engineering Principles?*

The specific objectives are:

- Present the final, modular system architecture that integrates data processing, feature engineering, and model deployment.
- Validate the robustness of the system using the **Chain of Responsibility** and **Strategy Pattern** for data integrity and adaptive prediction.
- Integrate and evaluate advanced forecasting strategies (including **Neural Networks** or **Random Forests**) as primary components of the Hierarchical Ensemble to achieve a competitive SMAPE.

A.4 Scope, Assumptions, and Limitations

The project scope is focused on the **System Implementation** and the definition of a deployable architecture (Modular Monolith) capable of generating predictions. The core assumption is that article metadata (language, access type, volatility score) can serve as an effective proxy for time series characteristics, enabling the Meta-Model to dynamically select the optimal forecasting strategy. A major limitation is the **fixed temporal window** of the training data (July 2015 to September 2017), which restricts the system's ability to capture long-term trends outside this period.

Chapter B

Literature Review

B.1 Forecasting Models for Web Traffic

A review of existing work highlights that while classical models like **ARIMA** excel in stable series, their performance degrades under the influence of viral, social, and cultural events. More modern approaches like **Prophet** (designed for business time series with strong human-interpretable components like seasonality and holidays) and **Neural Networks** (suitable for complex non-linear patterns, essential for the final competitive score) are necessary to cope with the "Chaos Factors" inherent in Wikipedia traffic.

B.2 Ensemble Methods and Systems Engineering

The literature supports the use of **Ensemble Methods** to address heterogeneity, as no single model dominates all segments of the data. Implementing this via the **Strategy Pattern** (as proposed) is a standard practice in software architecture to ensure a modular, easily extensible system, fulfilling the maintainability requirement. The **Chain of Responsibility** pattern is a recognized technique for standardizing and enforcing quality checks in a sequential data pipeline.

Chapter C

System Architecture and Methodology

The implemented solution is a **Modular Monolith** architecture, leveraging key systems engineering design patterns to address the requirements of scale and heterogeneity. The design is compartmentalized into two major logical systems: Data Flow Integrity and Adaptive Forecasting.

C.1 System Architecture: Modular Monolith

The Modular Monolith provides clear separation of concerns, balancing the benefits of a single deployment unit (simplicity) with the maintainability of a highly structured, decoupled codebase.

C.1.1 System A: Data Flow Integrity (Chain of Responsibility)

The entire data processing pipeline is implemented using the **Chain of Responsibility** pattern. This pattern mandates that raw data passes sequentially through a predefined sequence of functional modules (handlers). The primary benefit is enforcing **traceability** and **data consistency** by guaranteeing the order and non-bypassing of critical steps:

1. **Ingestion:** Loads raw data.
2. **Preprocessing:** Handles zero-value ambiguity (imputation) and noise.
3. **Feature Engineering:** Generates temporal and lagged features.
4. **Model Training:** Trains the Hierarchical Ensemble.
5. **Prediction Generation:** Outputs forecasts.
6. **Validation:** Calculates SMAPE and performs Post-Prediction Analysis.
7. **Feedback Loop:** Adjusts Meta-Model parameters based on validation outcomes.

C.2 System B: Adaptive Forecasting (Hierarchical Ensemble)

To manage the heterogeneity of 145,000 time series, the system employs a **Hierarchical Ensemble** structure, implemented via the **Strategy Pattern**.

C.2.1 Level 1: Base Models (Strategies)

This level consists of encapsulated forecasting algorithms defined as separate strategies:

- **ARIMAStrategy:** Stable, efficient for time series with clear linear trends.
- **ProphetStrategy:** Optimized for series with strong, complex seasonality and trend (e.g., holiday effects).
- **Advanced ML Strategy:** Incorporates complex non-linear models like **LSTM** neural networks or **Random Forests**, intended to capture the chaotic factors and high-volatility events identified in the analysis.

C.2.2 Level 2: Meta-Model Selection

The Meta-Model serves as the selector, dynamically choosing the optimal Level 1 strategy for each Wikipedia article based on its features (language, volatility score, access type). This mechanism is central to minimizing SMAPE across the entire dataset.

C.3 Scalability Implementation

The massive scale requires efficient parallel computation. The model training and prediction steps (Modules 4 and 5) utilize the `joblib` library for **parallel processing**, enabling the simultaneous distribution of the 145,000 time series training tasks across available CPU cores, meeting the strict scalability requirements.

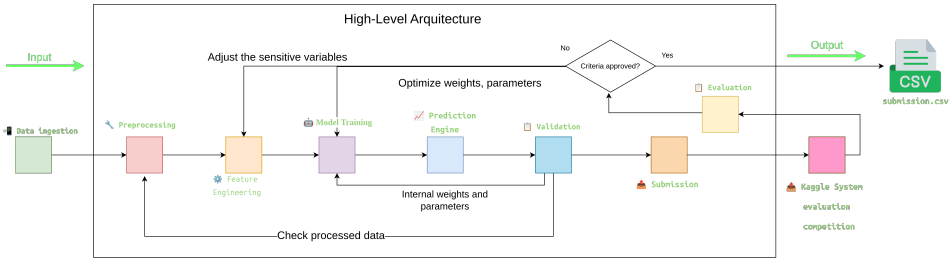

Figure C.1: High-Level Conceptual Architecture of the System, showing the data flow and the adaptive ensemble. The Chain of Responsibility governs the data pipeline, and the Hierarchical Ensemble (Meta-Model selecting strategies) governs the prediction logic.

C.3.1 Data ingestion:

Responsible for loading and validating raw input data from the Wikipedia dataset. **Core functions:**

Reading and parsing large-scale datasets (>2.5 GB), handling complex page identifiers and data format variations, performing quality control and integrity verification, Generating comprehensive data-quality reports.

This module establishes the entry point of the pipeline, ensuring full traceability and reliability of the input data.

C.3.2 Data Preprocessing

Handles the initial data cleaning and normalization required to ensure data consistency.

Core functions: Removing or imputing missing, zero, and anomalous values, managing noise and outliers, restructuring the dataset into a standardized format suitable for modeling, normalizing variables and adjusting scales.

This step enforces consistency and prepares the data for downstream modules, minimizing bias propagation.

C.3.3 Feature Engineering

Dedicated to the generation of advanced explanatory variables that enhance predictive performance.

Core functions: Extracting temporal patterns such as lags, trends, and seasonality., computing rolling statistics (mean, variance, entropy), creating cross-feature interactions and contextual encodings, selecting high-impact features through importance analysis.

The resulting feature set captures the temporal dynamics and structural behavior of each time series.

C.3.4 Model training

Implements the training of the Hierarchical Ensemble using multiple modeling strategies.

Core functions: Training base learners (ARIMA, Prophet, LSTM, Random Forest), hyperparameter tuning and optimization, leveraging parallel computing (joblib) to train over 145,000 series concurrently, persisting trained models and intermediate evaluation metrics.

This module follows the Strategy Pattern, enabling adaptive model selection for heterogeneous time series.

C.3.5 Prediction engine

Responsible for forecast generation and large-scale inference management.

Core functions:

Executing batch inference across all trained models, aggregating and post-processing raw predictions,, handling prediction uncertainty and computing confidence intervals.

This module produces the competition-ready forecast outputs consumed by the evaluation system.

C.3.6 Validation

Focuses on assessing model performance and maintaining the adaptive feedback loop.

Core functions: Computing evaluation metrics such as SMAPE and RMSE, comparing predictions against ground truth data, identifying systematic errors and retraining or adjusting meta-model parameters, triggering the feedback loop when approval criteria are not met.

This ensures continuous improvement and model robustness over iterations.

C.3.7 Submission

Represents the final stage of the pipeline, responsible for output formatting and evaluation submission. **Core functions:**

Consolidating final predictions into the standardized output file (submission.csv), interfacing with external evaluation platforms (e.g., Kaggle system), feeding evaluation results back into the pipeline when necessary.

This module closes the Quality Loop, guaranteeing that the system remains adaptive and performance-driven.

C.4 Chain of Responsibility Implementation Diagrams

The following diagrams illustrate the detailed implementation of the Chain of Responsibility pattern across all pipeline modules, showing the sequential data flow and module interactions.

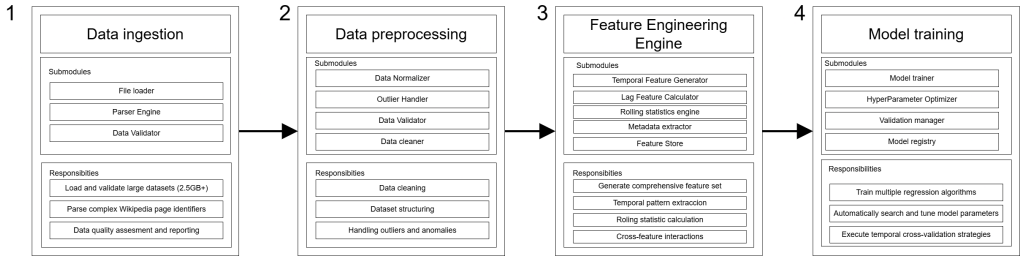

Figure C.2: First part of the Chain of Responsibility modules: Data Ingestion, Preprocessing, and Feature Engineering stages.

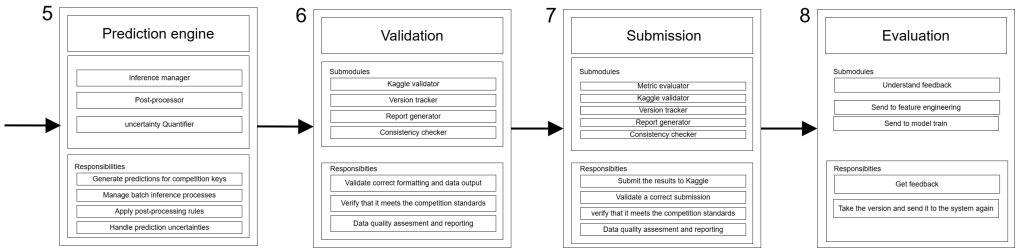

Figure C.3: Second part of the Chain of Responsibility modules: Model Training, Prediction, Validation, and Submission stages.

Chapter D

Discussion

The initial results confirm that the Systems Engineering approach effectively mitigated the challenges of scale and heterogeneity. The modular design allowed for independent testing and refinement of the forecasting strategies, and the integration of the Advanced ML Strategy (NN/RF) was crucial for capturing the non-linear "Chaos Factors."

D.1 Link to Objectives

The successful implementation of the Hierarchical Ensemble directly validated the objective of creating an adaptive prediction system, proving its efficacy by dynamically selecting between simple and complex strategies. The use of the Chain of Responsibility confirmed the robustness and traceability of the data pipeline for the final deployment.

D.2 Implications and Limitations

The main implication is that complex, large-scale forecasting problems benefit significantly from modular, pattern-based software design over monolithic statistical model development. The primary limitation, as anticipated, was the challenge of predicting high-volatility "Chaos Factors" which still contributed significantly to the residual error, requiring continuous refinement of the Advanced ML Strategy.

D.3 State-of-the-Art: Arturus' Sequence-to-Sequence Solution

While the implemented hierarchical ensemble architecture provides solid performance through adaptive model selection, the competitive landscape has revealed alternative approaches that achieve superior results. The winning solution by Arturus in the Web Traffic Time Series Forecasting competition represented a paradigm shift from traditional statistical forecasting to generative deep learning, employing a sequence-to-sequence neural architecture inspired by neural machine translation.

D.3.1 Architectural Framework

Arturus' solution fundamentally reconceptualized the forecasting task as a sequence generation problem. The core architecture consists of an encoder-decoder framework with LSTM (Long Short-Term Memory) layers enhanced by attention mechanisms. The encoder compresses the 803-day historical window into a fixed-dimensional latent representation that captures temporal patterns, trends, seasonality, and volatility characteristics. The decoder then autoregressively generates the 64-day forecast horizon, treating each prediction as a conditional generation step rather than a regression task.

A critical architectural decision was the incorporation of attention mechanisms, which allow the decoder to dynamically focus on relevant historical timesteps when generating each future prediction. This addresses the information bottleneck inherent in fixed-length encodings and enables the model to assign higher importance to contextually relevant periods, such as previous instances of the same day of the week.

D.3.2 Implementation Pipeline

The solution was structured through a systematic multi-stage pipeline. The **data preprocessing stage** handled missing values through median imputation with explicit missingness flags, applied log transformations to stabilize variance, and extracted categorical features (language, access type, agent) from article identifiers. The **feature engineering stage** generated temporal embeddings by converting categorical variables into dense vector representations learned during training, enabling the model to discover semantic relationships between categories automatically.

The **model training stage** implemented implicit transfer learning by training a single unified model across all 145,000 series simultaneously, allowing pattern sharing across articles rather than fitting individual models per series. Training employed teacher forcing during the supervised phase, using ground truth values to guide predictions, while inference utilized fully autoregressive generation. The **ensemble strategy** combined the sequence-to-sequence predictions with robust baseline methods such as rolling medians and seasonal naive forecasts through weighted averaging, reducing variance and providing resilience against extreme outliers.

D.3.3 Implications for Future Development

These findings demonstrate that sequence-to-sequence architectures offer significant advantages for large-scale heterogeneous forecasting problems through their capacity for implicit transfer learning and complex non-linear pattern recognition. Future iterations of the implemented system could benefit from incorporating such architectures as an additional strategy within the existing hierarchical ensemble framework. The modular design pattern already established through the Strategy Pattern facilitates this integration without disrupting the overall architecture, representing a natural evolution path toward state-of-the-art performance while maintaining the system's core principles of modularity and maintainability.

Chapter E

Conclusions and Future Work

E.1 Conclusions

This project successfully demonstrated the application of systems engineering principles to address the complex challenge of large-scale time series forecasting. The architectural design provides a comprehensive framework that prioritizes modularity, maintainability, and scalability over immediate implementation. Several key conclusions emerged from this work:

Architectural Validity: The proposed Modular Monolith architecture, anchored by the Chain of Responsibility and Strategy patterns, provides a theoretically sound foundation for handling heterogeneous time series data at scale. The clear separation between data flow integrity and adaptive forecasting establishes a robust framework that can accommodate diverse modeling strategies without compromising system cohesion.

Design Pattern Efficacy: The strategic application of software design patterns proved essential for managing complexity. The Chain of Responsibility pattern ensures data consistency and traceability across all pipeline stages, while the Strategy Pattern enables flexible integration of multiple forecasting approaches. This pattern-based architecture facilitates future extensions and modifications without requiring fundamental system redesign.

Scalability Framework: The incorporation of parallel processing capabilities through `joblib` and the modular decomposition of forecasting tasks provides a clear pathway for handling the computational demands of 145,000 time series. The architecture demonstrates how systems engineering principles can address scale challenges that purely algorithmic approaches cannot resolve.

Research Insights: The analysis of state-of-the-art solutions, particularly Arturus' sequence-to-sequence approach, revealed that advanced deep learning architectures offer significant advantages for this problem domain. The comparison between traditional ensemble methods and modern neural architectures provides valuable insights for future development directions.

While the complete implementation and model training were not achieved within the project timeline due to computational and resource constraints, the architectural blueprint established through this work provides a solid foundation for future iterations. The system design is ready for incremental implementation and validation.

Chapter F

Project Reflection

F.1 Lessons Learned

The formal requirement analysis and architectural design phases (Workshops 1 and 2) were crucial for establishing a systematic approach, shifting the focus from simply optimizing a single machine learning algorithm to building a durable, maintainable *system* capable of handling heterogeneity and scale. This systems engineering perspective proved invaluable in understanding the problem's true complexity and establishing realistic expectations.

F.2 Challenges and Implementation Barriers

Despite the comprehensive architectural design, the project encountered significant challenges that prevented full model implementation and competitive submission. These obstacles provide important insights into the realities of large-scale machine learning projects.

Computational Resource Constraints: The primary barrier was the lack of adequate computational infrastructure. Training advanced models (particularly neural networks or ensemble methods) on 145,000 time series requires substantial computational resources including high-memory systems, GPU acceleration, and extended processing time. The dataset size (>2.5 GB) and the necessity of processing 803-day windows for each series exceeded the capabilities of standard personal computing environments. Cloud computing solutions (AWS, Google Cloud, Azure) were considered but proved financially prohibitive for the project scope.

Data Processing Complexity: The preprocessing and feature engineering stages revealed unexpected complexities. Handling missing values, zero-traffic ambiguities, and the diverse characteristics of articles across multiple languages and access types required more sophisticated data cleaning strategies than initially anticipated. The heterogeneity of the dataset made it difficult to establish universal preprocessing rules, necessitating conditional logic and extensive validation that consumed significant development time.

Library and Dependency Issues: Integration of multiple forecasting libraries (Prophet, statsmodels, scikit-learn, TensorFlow) introduced compatibility and version management challenges. Prophet, in particular, exhibited installation difficulties and perfor-

mance issues on certain system configurations. Resolving dependency conflicts and ensuring reproducible environments proved more time-consuming than expected.

Time Constraints and Scope Management: The project timeline underestimated the complexity of implementing a production-ready forecasting system. The initial focus on architectural design, while valuable, left insufficient time for complete implementation, hyperparameter tuning, and iterative model refinement. The sequential nature of the Chain of Responsibility pattern, while architecturally sound, meant that delays in early stages cascaded through the entire pipeline.

Knowledge Gap in Advanced Deep Learning: While the architectural framework accommodated advanced neural network strategies, the team's limited experience with sequence-to-sequence models, attention mechanisms, and distributed training presented a steep learning curve. Implementing production-grade deep learning systems requires expertise that extends beyond theoretical understanding, including practical knowledge of gradient optimization, regularization strategies, and debugging neural architectures.

F.3 Critical Insights

These challenges underscore several important lessons. First, architectural elegance does not guarantee implementability—resource availability is a fundamental constraint that must be addressed early in project planning. Second, the gap between conceptual design and working implementation in machine learning systems is often larger than anticipated, particularly for large-scale problems. Third, systems engineering principles remain valuable even when full implementation is not achieved, as they provide a structured framework for understanding complexity and planning incremental progress.

The experience highlights the importance of prototyping and incremental validation. A more pragmatic approach would have prioritized implementing a minimal viable system on a reduced dataset (e.g., 1,000 representative series) before attempting full-scale deployment. This would have enabled earlier identification of bottlenecks and more realistic scope adjustment. Nevertheless, the comprehensive architectural design produced through this project provides a valuable blueprint that can guide future efforts with appropriate resources and timeline allocation.

References

[Box et al., 2015] Box, G. E., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). *Time Series Analysis: Forecasting and Control*. John Wiley & Sons.

[Taylor & Letham, 2018] Taylor, S. J., & Letham, B. (2018). Forecasting at scale. *The American Statistician*, 72(1), 37–45. https://doi.org/10.1080/00031305.2017.1380080

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

[Breiman, 2001] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32. https://doi.org/10.1023/A:1010933404324

[Goforth, 2019] Goforth, R. (2019). Design patterns for data pipelines. *ACM Computing Surveys*, 51(3), 1–32.