# Technical Report
# Systems Analysis for the Kaggle Competition "Drawing with LLMs"

Edward Julian García Gaitán (20212020136)
Jaider Camilo Carvajal Marin (20212020120)
Nelson David Posso Suárez (20212020132)

Course: Systems Analysis and Design
Professor: Eng. Carlos Andrés Sierra, M.Sc.

Program: Systems Engineering
Faculty of Engineering
Universidad Distrital Francisco José de Caldas

Bogotá D.C.

May 16, 2025

**Abstract**

This document presents the systems analysis for the Kaggle competition "Drawing with LLMs." It includes a summary of Workshop No. 1, system requirements, an architecture proposal, strategies for managing sensitivity and chaos, the recommended technological stack, an implementation plan, and references. The objective is to propose a software architecture and implementation methodology that proactively address the challenges identified in generating SVG images from textual descriptions using Large Language Models (LLMs).

# Contents

# 1 Introduction

The "Drawing with LLMs" competition [1], promoted by the Kaggle platform, poses a contemporary challenge at the forefront of artificial intelligence: generating images in Scalable Vector Graphics (SVG) format from textual descriptions (prompts) using Large Language Models (LLMs) [2]. This task lies at the confluence of Natural Language Understanding (NLU), automatic code generation, and computer graphics synthesis, inherently introducing notable systemic complexity.

This technical report is derived from Workshop No. 2 of the Systems Analysis and Design course, which, in turn, was based on a prior analysis (Workshop No. 1) of the competition from a systems engineering perspective. This initial analysis unveiled the system's structure, identifying crucial elements and their sensitive interactions, as well as anticipating manifestations of chaos theory [6] in the generation process.

The main problem addressed is the design of a system capable of efficiently translating textual prompts into visually coherent and syntactically valid SVGs, considering the variability and sometimes unpredictable nature of LLMs. This report's primary objective is to propose a detailed software architecture and implementation methodology that directly and proactively respond to the identified challenges and requirements.

# 2 Literature Review

The proposed solution is based on several key fields and technologies:

- **Large Language Models (LLMs):** LLMs, such as those available through the Hugging Face Transformers library [2], are deep learning models trained on vast amounts of textual data. They have demonstrated remarkable capabilities in text understanding and generation, and their application extends to intermodal translation tasks, such as generating code from natural language.

- **Scalable Vector Graphics (SVG):** SVG is an XML-based vector image format for two-dimensional graphics with support for interactivity and animation [4]. Its textual and structured nature makes it a plausible target for generation by LLMs, although its formal grammar demands syntactic precision.

- **Systems Engineering and Systems Thinking:** The design of complex systems, especially those with AI components, greatly benefits from systems engineering principles. Systems thinking, as described by Meadows [5], helps to understand interconnections, feedback loops, and emergent behaviors, crucial for managing the sensitivity and chaos inherent in systems like the one proposed.

- **Chaos Theory:** Sensitive dependence on initial conditions ("butterfly effect") and non-linear behaviors are characteristic of chaotic systems [6]. In the context of SVG generation with LLMs, small variations in the prompt or model parameters can lead to drastically different results, which needs to be managed.

- **Image Quality and Evaluation:** Although the competition may have its own metrics, concepts like Structural Similarity (SSIM) [8] are relevant in evaluating the quality of generated images compared to references, if applicable. For SVG, syntactic validation and semantic coherence are paramount.

This review underscores the multidisciplinary nature of the problem and the need to integrate knowledge from AI, computer graphics, and robust software engineering.

# 3 Background

The "Drawing with LLMs" competition by Kaggle is part of the growing interest in the generative capabilities of AI. The specific challenge of converting textual descriptions into structured graphical representations like SVG involves several sub-problems:

- **Prompt Interpretation:** The system must understand the semantics of the input text, identifying objects, attributes, spatial relationships, styles, etc.
- **Intermodal Translation:** A transformation from the semantic representation of natural language to a formal representation of SVG code is required.
- **Valid Code Generation:** The resulting SVG must strictly adhere to the format specification to be renderable.
- **Handling Ambiguity and Variability:** Natural language is inherently ambiguous, and LLMs can generate multiple outputs for the same input.

Workshop No. 1 identified the system's sensitivity to prompt quality, LLM configuration (e.g., temperature, top-p), and the robustness of SVG parsing and rendering components. This background directly informs the requirements and architecture of the system proposed in this document.

# 4 Proposed System Objectives

The macro-objectives for the SVG generation system, derived from the guidelines of Workshop No. 2 and best practices in software engineering, are:

1. **Functional Effectiveness:** The system must be capable of generating SVG representations that are semantically coherent with the input prompt and syntactically valid according to the SVG format specification.
2. **Robustness and Reliability:** The system must exhibit predictable behavior with valid inputs, be tolerant to failures in individual components, and appropriately handle anomalous or malformed inputs.
3. **Operational Efficiency:** The system must meet the performance constraints imposed by the competition, particularly in terms of generation latency and processing capacity.
4. **Scalability and Maintainability:** The architecture must facilitate adaptation to increasing volumes of data or requests and allow for system evolution (e.g., incorporating new LLMs, adjusting modules) with reasonable maintenance effort.

# 5 Scope

The scope of the proposed system focuses on:

- The ingestion of textual prompts in the format specified by the Kaggle competition.
- The preprocessing and advanced semantic interpretation of these prompts.
- The generation of SVG 1.1 source code (or the relevant specified version) using LLMs.
- The syntactic validation and postprocessing (heuristic correction, sanitization) of the generated SVG.
- The flexible configuration of generation parameters.
- The persistence of results (SVGs and metadata) for analysis and submission.

The system will not initially address (unless extended as future work):

- End-user interfaces for interactive prompt creation.
- Large-scale training or fine-tuning of LLMs from scratch, although fine-tuning pre-trained models is considered.
- Generation of graphical formats other than SVG.

# 6 Assumptions

The system design is based on the following assumptions:

- Availability of pre-trained LLMs (either via APIs or downloadable models) capable of generating code or structured text.
- The input/output specifications and formats provided by Kaggle for the competition are clear and stable.
- The Kaggle execution environment (or the development environment) will allow the use of the technologies selected in the stack.
- It is possible to define metrics (or use those from the competition) to evaluate the quality and coherence of the generated SVGs.

# 7 Limitations

The proposed system and the problem itself have certain limitations:

- **Ambiguity of Natural Language:** Despite advances in NLU, perfectly unambiguous interpretation of complex or subjective prompts remains a challenge.
- **Nature of LLMs:** LLMs can "hallucinate" or generate unintended outputs. Their complete control is difficult, requiring mitigation strategies.

- **Performance (Latency):** The NFR1 objective (generation latency ¡ 1s) is ambitious, and its feasibility will depend on the complexity of the selected LLM, the complexity of the prompt and the resulting SVG, and hardware capabilities.
- **Subjective Visual Quality:** The "quality" or "aesthetics" of an image can be subjective, and the system will focus on semantic coherence and technical validity.
- **Dependency on External Tools:** The system will depend on the stability and APIs of libraries like Hugging Face Transformers or the LLMs themselves.

# 8 Methodology

The methodology for the system design and development is based on systems engineering principles and an iterative development approach.

## 8.1 System Architecture

A modular, layered architecture with a well-defined data processing flow is proposed.

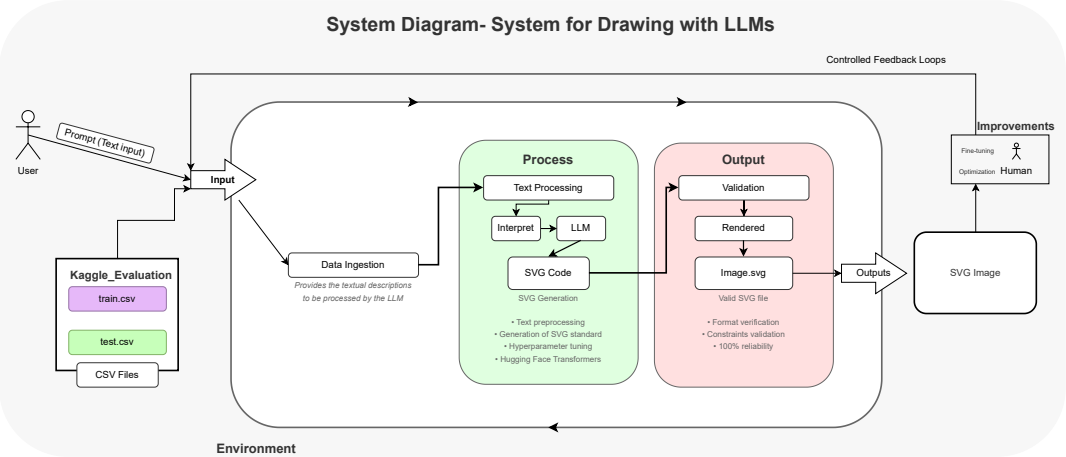


Figure 1: Conceptual System Diagram for the "Drawing with LLMs" System (Textual description of Figure 1 from the Workshop document with corrections).

## 8.2 Strategies for Sensitivity Management and Chaos Mitigation

To address the system's high sensitivity to prompt formulation and the unpredictable nature of LLM outputs, the following strategies are proposed:

1. **Robust and Adaptive Prompt Engineering (in Module M2 - Preprocessing):**
    - Normalization and Standardization of prompts.

- Use of Prompt Templating.
- Dynamic Few-Shot Learning techniques.

2. **Control and Stabilization of LLM Generation (in Module M3 - LLM Core):**
   - Fine-tuning of LLM Inference Parameters.
   - Multiple Sampling and Selection strategies for SVG candidates.
   - Intelligent Retry Mechanisms.
   - Definition of Stop Sequences and Length Limits.
3. **Rigorous Validation and Defensive Postprocessing (in Module M4 - SVG Postprocessing):**
   - Multi-level Validation (XML syntax, SVG profiles, attributes).
   - Heuristic Correction and Sanitization Rules for SVG.
   - Complexity Limits (Circuit Breakers) for the generated SVG.
4. **Continuous Monitoring, Detailed Logging, and Error Analysis (in M0 and specific modules):**
   - Structured Logging (e.g., JSON) at each stage.
   - Monitoring of Process and Quality Metrics.
   - Root Cause Analysis for systematic failures.
5. **Controlled Feedback Loops (in the development process):**
   - Human-in-the-Loop Evaluation.
   - Long-term consideration of LLM Fine-tuning.

## 8.3 Phased Implementation Plan

An iterative and incremental development approach is proposed, divided into five phases over 12 weeks:

**Phase 1: Initialization and Core Prototyping (Week 11):** Environment setup and development of a minimal "end-to-end" prototype. Objective: Validate technical feasibility.

**Phase 2: Development of Central Modules (Weeks 12):** Robust implementation of modules M1 (Ingestion), M2 (Basic Preprocessing), M3 (LLM Core), and M4 (SVG Postprocessing). Objective: Main processing chain functional and tested.

**Phase 3: Implementation of Robustness and Quality Strategies (Weeks 13-14):** Refinement of M2, implementation of sensitivity management in M3 and M4, development of M5 (Quality Assessment). Objective: Significantly improve quality, reliability, and predictability.

**Phase 4: System Optimization and Documentation (Weeks 15-16):** Performance optimization, scalability testing, thorough documentation. Objective: Functionally complete and optimized system.

**Phase 5: Final Testing, Refinement, and Packaging (Weeks 17-18):** Execution of a comprehensive test plan, bug fixing, preparation for Kaggle submission. Objective: Maximum quality and compliance.

# 9 Expected Design Outcomes

As this is a design document, the "outcomes" refer to the deliverables and characteristics of the proposed system, as well as the criteria for its evaluation once implemented.

- **Defined Architecture:** A modular and layered architecture as described in Figure 1.
- **Clear Mitigation Strategies:** The detailed strategies for managing sensitivity and chaos.
- **Structured Implementation Plan:** The phased plan that will guide development.
- **Requirements Compliance:** The implemented system is expected to meet the defined functional (FR1-FR5) and non-functional (NFR1-NFR6) requirements, especially:
  - NFR1 (Performance): SVG generation ¡ 1s (ambitious goal).
  - NFR2 (Reliability and Accuracy): 100% syntactically valid SVGs and high semantic coherence.
  - NFR3 (Scalability): Ability to process up to 10,000 prompts.
  - NFR6 (Robustness): Controlled handling of errors and anomalous inputs.

# 10 Discussion

The proposed architecture and strategies seek a balance between the innovative generative capabilities of LLMs and the discipline of software engineering to build a robust and reliable system.

**Modularity** is key to managing complexity. It allows each component (ingestion, preprocessing, LLM core, postprocessing, etc.) to be developed, tested, and maintained relatively independently. The use of **well-defined interfaces** between modules is crucial to minimize coupling and facilitate future modifications, such as changing a specific LLM or improving a validation algorithm.

The **strategies for managing sensitivity and chaos** are fundamental given the nature of LLMs. Prompt engineering, control of inference parameters, and especially defensive postprocessing (multi-level validation, heuristic correction) act as control mechanisms to increase the predictability and quality of outputs. Continuous logging and monitoring will allow for iterative improvement based on evidence of system behavior.

The selected **technology stack** (Python, Hugging Face, etc.) offers a rich and mature ecosystem for developing this type of application. Community familiarity with these tools and the abundance of documentation and support are significant advantages.

Nevertheless, challenges are recognized. **Generation latency** (NFR1) is a demanding objective that will depend on implementation efficiency and the chosen LLM's capabilities. The **interpretation of intent** in very ambiguous or abstract prompts will remain an area where perfection is difficult to achieve. Dependency on LLM models, especially if they are third-party services, introduces an external risk factor.

The **phased implementation plan** allows these challenges to be addressed incrementally, validating assumptions and adjusting course as progress is made. Early phases focus on establishing core functionality, while later phases are dedicated to robustness, quality, and optimization.

# 11 Conclusion

In summary, as demonstrated throughout this report, substantial progress has been achieved in establishing the foundational elements of the project. This groundwork has effectively prepared us to transition into the implementation phase, now that the core conceptual framework, system architecture, and strategic approaches are clearly defined. The iterative process, particularly the insights gained from Workshop 1 and Workshop 2, has been instrumental. These workshops have not only crystallized the functional and non-functional requirements but have also been pivotal in formulating robust strategies to manage and mitigate the anticipated chaotic behaviors and sensitivities inherent in a system leveraging Large Language Models for creative generation tasks.

The critical importance of adopting a systemic perspective to address the complexities of the "Drawing with LLMs" competition has become evident. Systems engineering principles have provided an invaluable lens through which to structure the problem, design modular components, and anticipate interactions. This structured approach is particularly beneficial for AI-driven projects, which often involve intricate, non-linear dynamics. Furthermore, an understanding of chaos theory has enriched our comprehension of the system's potential behavior, particularly how slight variations in input prompts or model parameters can lead to diverse outputs—a characteristic intrinsic to the generative nature of LLMs and their implementation. This awareness has directly informed the design of defensive mechanisms and adaptive strategies within the proposed architecture.

The successful development of this comprehensive design document, which outlines a clear path from conceptualization to a deployable solution, is a testament to a rich learning environment. The knowledge imparted during lectures, the valuable academic resources and articles shared via the course's communication channels (e.g., Slack), and the insightful guidance provided by the professor have all been crucial contributions. This collaborative and well-resourced academic setting has been fundamental in shaping the analysis and fostering the development of the robust system design presented herein. We are now well-positioned to embark on the practical implementation, confident in the thoroughness of the preparatory design and analysis.

# References


[1] Kaggle. (n.d.). *Drawing with LLMs*. Retrieved from https://www.kaggle.com/competitions/drawing-with-llms/overview

[2] Hugging Face. (n.d.). *Transformers Documentation*. Retrieved from https://huggingface.co/docs/transformers/

[3] The svgwrite Core Developers. (n.d.). *svgwrite Documentation*. Retrieved from https://svgwrite.readthedocs.io/

[4] W3C. (2011). *Scalable Vector Graphics (SVG) 1.1 (Second Edition)*. Retrieved from https://www.w3.org/TR/SVG11/

[5] Meadows, D. H. (2008). *Thinking in Systems: A Primer*. Chelsea Green Publishing.

[6] Gleick, J. (1987). *Chaos: Making a New Science*. Penguin Books.

[7] VanderPlas, J. (2016). *Python Data Science Handbook*. O'Reilly Media.

[8] Wang, Z., Bovik, A. C., Sheikh, H. R., & Simoncelli, E. P. (2004). Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing, 13*(4), 600-612.

[9] lxml.de. (n.d.). *lxml – Processing XML and HTML with Python*. Retrieved from https://lxml.de/


# Glossary of Terms

**LLM (Large Language Model):** A type of artificial intelligence model trained with large amounts of textual data to understand and generate natural language.

**SVG (Scalable Vector Graphics):** An XML-based file format for describing two-dimensional vector graphics.

**NLU (Natural Language Understanding):** A subfield of AI focused on machine interpretation of human language.

**FR (Functional Requirement):** Describes what the system must do.

**NFR (Non-Functional Requirement):** Describes how the system must behave or the qualities it must possess (e.g., performance, reliability).

**SDD (System Design Document):** A document that describes the architecture and design of a system.

**API (Application Programming Interface):** A set of rules and protocols for building and interacting with software applications.

**IDE (Integrated Development Environment):** A software application that provides comprehensive facilities to computer programmers for software development.

**CSV (Comma-Separated Values):** A file format for tabular data.

**JSON (JavaScript Object Notation):** A lightweight data-interchange format.

**XML (Extensible Markup Language):** A markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.