Systematic analysis of the kaggle competition “20 questions”.

Laura Daniela Muñoz Ipus
Dept. of Computer Engineering
Universidad Distrital
Francisco José de Caldas
Email: ldmunozi@udistrital.edu.co

Esteban Bautista
Dept. of Computer Engineering
Universidad Distrital
Francisco José de Caldas
Email: eabautistas@udistrital.edu.co

Luisa Guerrero
Dept. of Computer Engineering
Universidad Distrital
Francisco José de Caldas
Email: lfguerreroo@udistrital.edu.co

***Abstract*—The Kaggle “20 Questions” competition presents a dynamic and interactive environment where two teams of language models (LLMs)—each composed of a questioner and an answerer—compete to guess a secret word through dichotomous (yes/no) questions. The game unfolds under strict time constraints, high uncertainty, and continuous feedback, making it an ideal case study for systems analysis. This paper presents a comprehensive examination of the underlying structure and behavior of the system, identifying its core components, information flows, control mechanisms, and performance dynamics. From a systems engineering perspective, a modular architecture is proposed to formalize the interactions between agents and support scalability, adaptability, and robustness. Additionally, the proposed design is validated through simulations based on real competition data, and a lightweight language model (Gemma 2B-IT) is integrated to automate question-answering tasks, demonstrating the system’s potential for interactive agent-based design and evaluation.**

***Index Terms*—System analysis, complex systems, feedback systems, adaptive architecture, multi-agent systems, modular design, Kaggle competition, artificial intelligence.**

1. Introduction

Understanding the behavior of interactive systems in competitive environments is a key challenge in the analysis of complex systems. One such scenario is the 20 Questions competition hosted on the Kaggle platform [1]. In this tournament, linguistic models (LLMs) compete in teams of two (a questioner and a responder) to guess a secret word in up to 20 rounds using dichotomous (yes/no) questions. Each match is played in a 2 vs 2 format, and the team that guesses the word first wins. If both guess correctly in the same round, the result is a tie. The questioner is allowed a maximum of 750 characters per question, and the responder answers only with “yes” or “no.” Each agent has 60 seconds per round and a total of 300 seconds of reserve time; exceeding these limits results in automatic loss. Performance is evaluated using a skill scoring system represented by a normal distribution $N(\mu, \sigma^2)$, where μ is the estimated skill and σ the uncertainty, both updated after each match. Bots start with an initial score $\mu_0 = 600$. and must validate themselves before entering the system. Participants may upload up to five bots per day, though only the three most recent remain active, and only the highest-scoring one appears on the leaderboard. The competition has fixed submission deadlines and a final evaluation, with cash prizes for the top five teams.

This competition tests key competencies such as deductive reasoning, strategic questioning, information-gathering efficiency, and inter-agent collaboration. These characteristics make it an ideal laboratory for systemic analysis, involving cybernetic feedback, sensitivity to initial conditions, emergent behavior, and chaotic dynamics.

This paper presents a systemic analysis and architectural design of the 20 Questions environment using tools from systems engineering. It identifies the structure, roles, and interactions of agents, mapping their feedback loops and dependencies on initial conditions. Based on this analysis, a modular and scalable architecture is proposed, designed to support robustness, adaptability, and traceability in uncertain and dynamic settings.

In the third phase of this work (Workshop 3), the proposed design was implemented through a simulation using real Kaggle datasets. The system was tested under various scenarios (balanced, chaotic, and skilled), revealing behavioral patterns, performance asymmetries, and limitations in system sensitivity. Finally, a lightweight language model (Gemma 2B-IT) was integrated into the architecture to automate question-answering tasks. The model was implemented using modular classes and prompt engineering techniques, allowing for evaluation of LLM-driven interaction in a controlled environment, while addressing the computational limitations of the Kaggle platform.

All supporting materials, including simulation code, analysis notebooks, and diagrams, are available in the GitHub repository listed in the bibliography [2].

2. Methods and materials

To approach the analysis and design of the Kaggle “20 Questions” competition system, a methodology divided into two complementary stages was adopted, corresponding to workshops 1 and 2 of the Systems Analysis and Design

course [1]. The first stage consisted of an in-depth analysis of the system from a functional, structural and dynamic perspective. The second stage was oriented to the architectural and technical design of a viable, adaptable and scalable solution. Both phases were developed under the principles of systems engineering, considering both the internal elements of the system and its interactions with the dynamic environment in which it operates.

2.1. First phase

In a first phase, a detailed analysis of the real system of the "20 Questions" competition hosted on the Kaggle platform was carried out. This competition proposes an environment in which two teams of bots participate, each composed of a guesser and an answerer, who must guess a secret word through a sequence of up to twenty rounds. From a systems engineering perspective, this environment is characterized by a complex structure, where multiple digital components interact under conditions of high uncertainty, variability and continuous feedback.

The analysis allowed us to identify the main functional elements of the system: the words dataset that feeds the game with the secret word, the tournament environment that organizes, validates and regulates the games, the teams of bots that face each other in each game, and the ranking system that adjusts the skills of the participants. Each component has defined functions, inputs, outputs and operational constraints: for example, bots must respect a limit of 60 seconds per turn, have a total of 300 seconds of reserve time per game, and must abide by length limits for questions (750 characters) and guesses (100 characters). These conditions are integrated into the logic engine of the tournament environment, which is also responsible for matching bots with similar skill levels and ensuring compliance with the technical rules of the competition.

The analysis also showed that the system is sensitive to the matching rules. The game matches bots randomly, but within a similar skill range. If this rule is altered, there could be a mismatch in the competition, where low level bots face highly trained bots, which would generate biases in the evaluation system and hinder the stability of the rankings, another factor that directly affects the sensitivity is the order of the questions. Since it is a cumulative deductive process, the same sequence of questions can be more or less effective depending on the order in which they are asked. Changing this order can completely alter the trajectory of the reasoning and, consequently, the final result. Chaotic behavior can also be evident in the system, since small variations in the initial questions or interpretations can trigger completely different trajectories. A poorly formulated question or an incorrect assumption at the start can lead to a wrong path that amplifies with each round, leading the bots away from the solution. The ambiguity of the answers, the lack of complete information, and the possibility of multiple valid sequences to arrive at the correct answer reflect the nonlinearity of the system.

This set of observations demonstrates that the system, although based on fixed, deterministic rules, exhibits highly sensitive global behavior, in which small variations in the inputs can cause effects on performance, score, and overall competitive balance. In addition, the system implements feedback mechanisms that make it dynamic and self-regulating. The result of each game is processed by the scoring system, which uses a normal distribution $N(\mu, \sigma^2)$ to adjust the skill of the bots, which μ represents the estimated skill, while σ expresses the level of uncertainty of that estimate. These parameters are constantly updated, allowing bots to progressively adapt to their accumulated performance. New bots start with high uncertainty and gain reliability as they participate in more games. This dynamic constitutes a continuous life cycle within the system, where agents go through phases of initialization, evaluation, adjustment and stabilization.

Taking into account the identified elements of the system, as well as their relationships and characteristics, the following diagram was created to show the above.

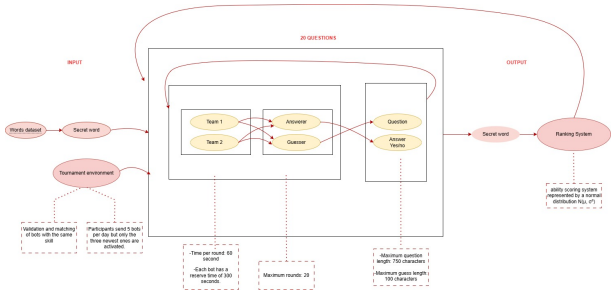

Figure 1. Relationship mapping of the system.

2.2. Second phase

In the second phase, an architectural representation was proposed based on principles of modularity, separation of responsibilities and scalability. This architecture does not seek to replicate the internal implementation of Kaggle, but rather to provide an organized view of its components as inferred from observed behavior. The architecture is composed of four functional layers. The first is the interaction layer, composed of the bots that send and receive messages; the second is the control layer, where the game engine is located, responsible for validating the question format, times and answers; the third is the evaluation layer, where the scoring system and performance metrics operate; and finally, the management layer, comprising the matchmaker, the game history module and the overall ranking system. This division allows each module to be analyzed as an autonomous subsystem with well-defined inputs and outputs, facilitating its traceability, maintenance and independent analysis. The modular architecture diagram can be placed after the explanation of this functional structure, as it

graphically synthesizes the elements and relationships that constitute the proposed system (see Figure 2).

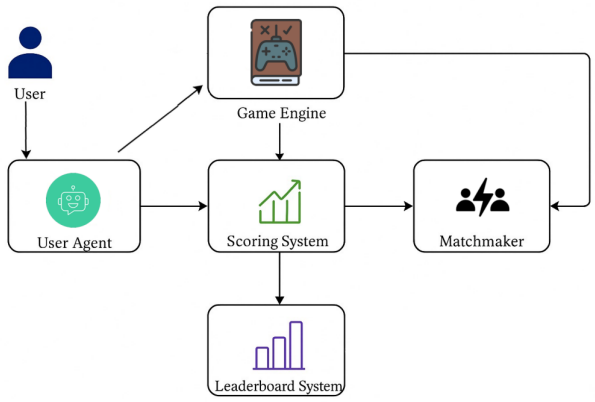

Figure 2. Architectural diagram describing the flow of data and interaction between components.

During design, technical decisions were made based on the need to maintain systemic coherence, adaptive flexibility and robustness in the face of uncertain conditions. For example, it was decided that the game engine should centralize validation to reduce ambiguities; that the scoring system should be based on aggregated statistics that evolve dynamically with player performance; and that the matchmaker should act as a balancing entity, assigning games according to similar skill levels. These decisions not only improve the fairness and traceability of the system, but also make it possible to better cope with the chaotic nature of the environment and guide the behavior of the bots toward more productive interaction.

Overall, this methodology, which implemented functional analysis, architectural modeling, and stability-oriented design decisions, provides a clearer understanding of the Kaggle competition system from a technical perspective. Although the project does not contemplate a functional implementation or code, the diagrams and proposed architecture provide a solid foundation for further studies, academic simulations, or developments that seek to explore similar automated competition systems in dynamic, high-uncertainty environments.

2.3. Third phase

Workshop 3 simulated the system designed for Kaggle's "20 Questions" competition, with the aim of validating its modular architecture using real data. Two key data sets were cleaned and processed: historical games and categorized keywords. The simulation included three different scenarios (balanced, chaotic, and specialized), allowing the system's behavior to be evaluated under different levels of strategy and ambiguity.

The results showed an unusually high success rate in all cases, even in the chaotic scenario, which showed low sensitivity to adverse conditions. In addition, an imbalance between roles was identified: questioners constantly improved while answerers lost skill. Based on this, improvements were proposed, such as fairer pairing, more complex metrics, contextual memory, and multilevel feedback to achieve a more balanced and adaptable system.

2.4. LLM implementation

To improve the capabilities of our system in the game "20 Questions," we integrated a large language model (LLM) called Gemma. This implementation allows the agent to generate questions and answers automatically, using previous examples (few-shot) to guide its behavior.

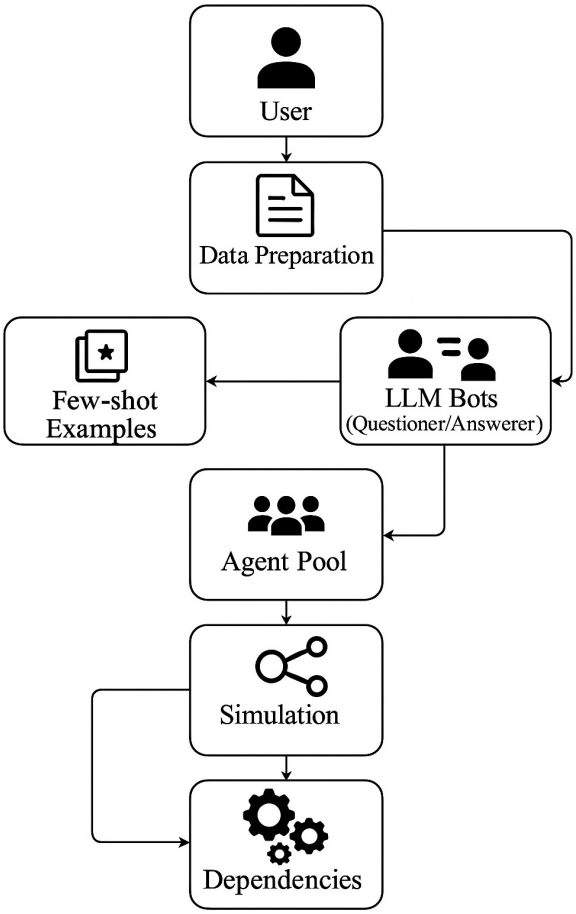

Figure 3. Architectural diagram describing the flow of data and interaction between components with llm implementation.

The configuration and preparation process for the Gemma 2B-IT model followed a structured methodology that ensured the correct initialization and operation of the system. Implementation began with the installation of critical dependencies, including immutabledict and sentencepiece, which are necessary for the tokenizer and the management of the model's immutable states. The initial configuration required cloning the gemma pytorch

repository from GitHub, providing access to Google's specific implementations for the Gemma model.
The integration of the Gemma 2B-IT model with the existing modular architecture was designed following principles of separation of responsibilities and loose coupling, allowing the LLM to interact seamlessly with the other components of the system. The integration was implemented through a class hierarchy that encapsulates the model's functionality while providing clear interfaces for the different roles in the game.
The base class GemmaAgent acts as the core of the integration, encapsulating model initialization, computational device management, and basic inference operations. This class implements the Template Method pattern, defining the general structure of interaction with the LLM while allowing derived classes (Questioner and Answerer) to implement specific behaviors.
The prompting strategies implemented in the system follow prompt engineering practices for large-scale language models, incorporating few-shot learning, prompt structuring, and role-based prompting techniques to maximize the model's effectiveness. The design of prompts is based on the premise that the quality of an LLM's output is directly related to the quality and structure of the input prompt.
The main strategy uses few-shot prompting, a technique that provides the model with specific examples of the task to guide its behavior. The examples include complete question-and-answer sequences that demonstrate the expected pattern of interaction. For example, the system includes examples such as "Is it a living thing?" $\rightarrow$ "yes" $\rightarrow$ "Is it a mammal?" $\rightarrow$ 'yes' $\rightarrow$ "Now guess the keyword." $\rightarrow$ "dog," which provide the model with a clear frame of reference for structuring its responses.

2.5. Resource and Performance Limitations Observed

During implementation, critical limitations related to the computational resources available in the Kaggle environment were identified. The LLM model experienced significant memory issues, both in system RAM and on the GPU when acceleration was attempted. These memory problems forced the decision to keep the model running on CPU, resulting in considerably longer processing times than expected.

The selection of the Gemma 2B-IT model was therefore dictated by these resource constraints, rather than by purely technical considerations. Larger and potentially more accurate models could not be used because the Kaggle environment ran out of memory during loading and initialization. This restriction had a direct impact on the quality of the generated responses, as lighter models are inherently less precise than their larger counterparts.

2.6. Analysis of Agent Behavior

The simulations revealed specific patterns in the agent's behavior, reflecting both the capabilities and limitations of the implemented model. Questions and answers varied significantly between different runs, even when using the same target keyword. This variability indicates an excessive dependence on the immediate context and a limited ability to maintain coherent strategies over the long term.

A particularly significant finding was that the model relied heavily on the provided few-shot examples. Instead of developing dynamic questioning strategies adapted to the specific context of each game, the agent tended to replicate patterns similar to the training examples, limiting its capacity for strategic innovation and adaptation to situations not covered by the examples.

Analysis of the generated question sequences showed a tendency toward repetition and a lack of logical progression on many occasions. Although the agent was able to formulate grammatically correct questions, it often failed to build upon previously obtained information to refine its search efficiently.

The quality of the guesses also showed significant inconsistencies. In some cases, the agent made logical assumptions based on the accumulated answers, while in other cases the guesses seemed disconnected from the context of the conversation.

2.7. Impact of Environment and Model Constraints

The restrictions of the Kaggle environment and the nature of the chosen model had a direct impact on the performance and conclusions of the project. Firstly, the exclusive use of CPU—driven by the lack of available GPU memory—meant that each inference with Gemma 2B-IT had a high response time. These prolonged response times limited the number of games that could be executed in a single simulation cycle and generated accumulated latency.

Secondly, the available RAM in the Kaggle environment prevented loading larger or more accurate model variants. Attempts to use higher-capacity versions—such as Gemma 7B or similar models from the Llama family—resulted in the system running out of memory and producing "out of memory" errors. This limitation forced the choice of Gemma 2B-IT.

Overall, these factors directly influenced the agent's ability to generalize, maintain coherent strategies, and deliver consistent performance across different simulation scenarios.

3. Results

A deep and structured understanding of the system that supports the "20 Questions" competency of the Kaggle platform was obtained by approaching it from

a systems engineering perspective. A detailed analysis of the system was carried out, recognizing that it is an adaptive, interactive environment with chaotic behavior. It was identified that the question and answer dynamics generate a high sensitivity to early decisions, since a single ambiguous or poorly formulated question can condition the trajectory of the bots' reasoning for the rest of the game. This property confirms that the system responds to principles of sensitivity to initial conditions, typical of nonlinear dynamic systems. It was also shown that there is continuous feedback: previous answers directly influence future questions, which generates a self-adjusting behavior of the questioning bot, configuring a cyclic process of adaptation.

The analysis of the system's life cycle allowed us to understand that it is not limited to the punctual interaction of a game, but integrates multiple chained phases: registration and validation of bots, matching by skill level, progressive interaction in turns, performance evaluation and updating of metrics. This structure defines a complete system, where each agent's experience accumulates and affects their future performance. The inclusion of a scoring system based on normal distributions with mean μ and standard deviation σ reinforces the notion of gradual learning, where more inexperienced agents have higher uncertainty (high σ) that decreases over time.

A conceptual architecture proposal was developed to clearly represent the interactions and responsibilities of each component of the system. This architecture was built from the elements observed in the previous analysis, highlighting six functional blocks: the participant bots (questioner and answerer), the game engine, the matching system, the format validator, the scoring system and the leaderboard. The functional connections between them were defined, organizing them in a modular model that facilitates their interpretation. Through this design, it was reflected how decisions flow from the agents to the control system, and how the results are fed back to the evaluation modules. This result makes it possible to clearly visualize a complex, distributed, but structured system, where each component has a specific and defined function.

Overall, the results obtained not only offer a comprehensive reading of the Kaggle system, but also provide an organized and technically sound proposal to represent its internal dynamics. The designed architecture functions as a conceptual tool that facilitates the study of adaptive competition systems, and could be extrapolated to other similar environments. The clarity in the separation of modules, the progressive evaluation logic, and the presence of control and feedback mechanisms are evidence of the structural value achieved through the analysis and design.

4. Conclusions

This work allowed a systemic analysis of Kaggle's "20 Questions" competition, applying systems engineering tools to understand and model its internal functioning. It was identified that the system presents characteristics of a complex system: a dynamic interaction between components, continuous feedback, sensitivity to chaos, adaptability to the environment and a life cycle influenced by the participants. A modular architecture was also proposed to represent the interaction between the main components of the system: the bots, the game engine, the validator, the matcher and the scoring system. This architecture allows to visualize the information flow in a structured way and to maintain a separation of responsibilities. Thus, a complex and adaptable system was clearly represented, demonstrating the effectiveness of the applied analysis approach. The work contributes not only to understand the dynamics of this competence, but also to offer a clear and structured vision of the internal functioning of the system, allowing to analyze its components and its information flows under a systemic approach.

The modular architecture implemented facilitated the development, debugging and adaptation of the system, allowing agile iterations and a robust base for future improvements. However, resource constraints in Kaggle forced the use of a lightweight model, which, together with the high dependency on few-shot examples, restricted the agent's generalization capability: it only managed to get it right when the keyword exactly matched the provided examples.

5. References

[1] Waechter, T. (2024). *LLM 20 Questions Games Dataset*. Retrieved from https://www.kaggle.com/datasets/waechter/llm-20-questions-games?select=EpisodeAgents.csv

[2] Muñoz, L. D. (2025). *Systems Analysis and Design*. Retrieved from https://github.com/LauraDanielaa/systems-analysis-and-design-

[3] Sierra, C. A. (2025). *Systems Analysis and Design – Course Materials*. Retrieved from https://github.com/EngAndres/ud-public/tree/main/courses/systems-analysis