ISSN: 1692-7257 - Volume 2 – Number 48 - 2026

RCTA
Revista Colombiana de Tecnologías de Avanzada
UNIPAMPLONA

Digital Object Identifier: 10.24054/rcta.v2i48.4320

Multi-agent system for raw material order validation based on a vision- language model

Sistema multiagente para validación de pedidos de materia prima basado en un modelo visión-lenguaje

MSc. Anny Astrid Espitia Cubillos[1], PhD. Robinson Jiménez-Moreno[2], Ing. Mateo Pulido Aponte[3]

[1]***Universidad Militar Nueva Granada,*** *Profesora asociada, Facultad de ingeniería, Programa de Ingeniería Industrial, Bogotá, Colombia*
[2]***Universidad Militar Nueva Granada,*** *Profesor asociado, Facultad de ingeniería, Programa de Ingeniería Mecatrónica, Bogotá, Colombia.*
[3]***Universidad Militar Nueva Granada,*** *Asistente de investigación, Facultad de ingeniería, Programa de Ingeniería Industrial, Bogotá, Colombia.*

Correspondence: *anny.espitia@unimilitar.edu.co*

Received: *march 24, 2026.* ***Accepted:*** *june 12, 2026.* ***Published:*** *july 06, 2026.*

How to cite: *D. A. Landinez Quintero, F. E. Moreno García, and W. Palacios Alvarado, "Multi-agent system for raw material order validation based on a vision- language model", RCTA, vol. 2, n.º. 48, pp. 57–68, jul. 2026. Recovered from https://ojs.unipamplona.edu.co/index.php/rcta/article/view/4320*

Abstract: This document presents the results of integrating the Deepseek OCR model with a vision-to-language model and a multi-agent system to develop an automated order validation system. This system operates by interpreting and comparing documents related to a raw material order, such as invoices, purchase orders, and delivery notes. It is integrated with a web interface for uploading and processing documents and viewing results. This allows for obtaining accuracy metrics for the overall system, facilitating the analysis of multiple documents, and identifying inconsistencies during order receipt and management. For validation, 35 tests were conducted using real data from a company that manufactures cleaning products. The information interpretation and comparison workflow were successfully executed in 82.353% of the cases.

Keywords: Deepseek OCR model, multi-agent system, document verification, orders, raw materials, vision-language model.

Resumen: En este documento se presentan los resultados de la integración del modelo Deepseek OCR mediante un modelo de visión a lenguaje y un sistema multi agente con el objetivo de desarrollar un sistema automático de validación de pedidos, el cual opera al interpretar y comparar documentos referentes a un pedido de materia prima como son la factura, la orden de compra y la remisión, integrado a una interfaz web para la carga y procesamiento de los documentos y visualización de resultados. Permitiendo obtener métricas de similitud y completitud del sistema conjunto para facilitar el análisis de múltiples documentos y encontrar inconsistencias en los mismos, en el momento de la recepción y gestión de un pedido. Para la validación se realizaron 35 pruebas usando información real de una empresa dedicada a la fabricación de productos de aseo obteniendo una ejecución exitosa del flujo de interpretación y comparación de información en el

82,353% de los casos.

Palabras clave: modelo Deepseek OCR, sistema multiagente, verificación de documentos, pedidos, materia prima, modelo de visión a lenguaje.

1. INTRODUCTION

The acquisition of raw materials and production input requires verifying orders upon receipt. This activity is crucial for ensuring consistency between key documents such as purchase orders, requisitions, delivery notes, and supplier invoices. This process is commonly performed manually, leading to human error, which can result in inconsistencies and rework, especially when validating electronic and physical documents in various formats simultaneously. Therefore, this article presents the development of a multi-agent solution using artificial intelligence to digitally verify documents, improving accuracy and efficiency.

An artificial intelligence (AI) agent can perceive, planning, decomposing, acting, and reflecting [1]. A Multi-Agent System (MAS) can be understood as a network of autonomous intelligent agents that interact to achieve a common goal. They are useful for tackling complex problems by breaking them down into simpler subproblems, enabling the development of resilient, adaptive, and scalable systems that allow the integration of artificial intelligence techniques to improve their computational and decision-making capabilities [2]. Despite their potential, they face hardware limitations. The use of multi-agent systems constitutes an efficient automation option for complex problems in logistics areas, thanks to decentralized decision-making and cooperation among agents, which allows for responses that adapt to operational changes [3]. To review network template planning documents, see [link to relevant documentation]. [1] It uses a multi-agent architecture based on open-source language models, which improves its processing and analysis.

Multi-agent systems using reinforcement learning (MARL) have made significant contributions to optimal inventory management [4], even under complex conditions such as multi-level networks [5]. Using MARL with graph neural networks to model the relationships between supply chain links leads to more robust decisions that promote collaborative work [6]. MARL has also been employed to improve inventory management in scenarios with random uncertainty, achieving efficient responses like or better than those obtained using traditional heuristics [7] and effectively validated through simulations under real-world business conditions. However, these models face challenges related to stability and scalability, particularly in probabilistic environments.

For text analysis in documents, Optical Character Recognition (OCR) has shown promising performance, but with disadvantages associated with high computational costs, a lack of flexibility in languages or document types, and the propagation of OCR errors to subsequent processing [8]. Therefore, [9] and [10] are turning to deep learning with generative elements, achieving advantages in accuracy and computational efficiency compared to traditional OCR. OCR and object detection (OD) technologies have also been integrated for recognizing textual and non-textual information on printed labels for industrial applications [11]; their advantage lies in the extraction of diverse data. However, errors propagate through each phase. OCR vision models only focus on transcribing documents and not on understanding them.

Currently, more advanced models, such as vision-to-language models (VLM), integrate image and text information into a single process, revolutionizing the automatic interpretation of files. These models improve the understanding of images with text in various formats, such as forms, tables, and scanned documents [12].

Several multi-agent systems have applications in the manufacturing sector. One paper [13] proposes managing increased production using a multi-agent system (MAS) that evaluates strategies in multivariate production contexts and illustrates its application in the furniture manufacturing sector, noting that MAS coordinates complex industrial tasks. Another [14] paper incorporates long language models (LLM) into a multi-agent system to manage manufacturing resources, a system that can be implemented in different smart production plants, replacing heuristics with natural language negotiation processes. Along the same lines, and given that supply chain management requires routine consensus, another [15] paper used LLM

agents for automation, where the authors validated their effectiveness through an inventory management case study. In this study, LLM agents reduced the bullwhip effect (the amplification of demand variations throughout the supply chain) more effectively than centralized replenishment policies and demand approaches.

Despite the afore mentioned advances, gaps remain in the literature regarding solutions that integrate multi-agent systems with VLM (Virtual Material Readiness) that function effectively in real-world scenarios for interpreting heterogeneous documents related to raw material order receipts and automatically validating the logical consistency of the information they contain in real time. It is within this context that the work presented in this article proposes a solution through the design and validation, using real data from a micro-industrial enterprise, of a multi-agent system for reviewing raw material orders. This system is based on a VLM that integrates OCR (Optical Character Recognition) and intelligent agents to not only extract data from multiple sources but also to classify, interpret, and compare information within the documents. Document validation is performed using textual and numerical similarity metrics and includes a web user interface for report visualization and management. The system reduces the need for human intervention and eliminates the propagation of errors in raw material order receipts.

2. METHODOLOGY

In the design and development of the multi-agent system for validating raw material orders, an applied research orientation was chosen, of an experimental type with a focus on design, application and validation, for which five phases were established.

In the first phase, the architecture of the multi-agent system is designed, which integrates an OCR system, where an agent is used that interprets and compares the information through a graphical user interface.

In the second phase, the OCR system is configured and executed in a local CUDA-supported environment, with specific graphics card compatibility (RTX 5070). This requires the prior installation of the Transformers and PyTorch libraries with support for the Blackwell – sm_120 architecture, which allows for the correct installation of the DeepSeek VLM model used to segment the documents. Meta's SAM and a visual decoder (DeepSeek encoder) are used to understand visual tokens, optimizing the processing of scanned documents. Its operation is validated in two stages: the first with predefined pipelines and the second with an optimized pipeline to initialize the segmentation and vision models under controlled conditions, thus reducing memory usage and computational requirements.

In the third phase, the multi-agent system architecture is designed using Google's Agent Developer Kit (ADK) [16], with a structure of both sequential and looping agents. This integrates OpenAI's large language model (GPR-4.1-mini) to classify documents, interpret data, and validate information consistency across documents [17]. Process control is achieved using callbacks, which allow monitoring of each step, preservation of intermediate results, updating of instructions, and process termination. A structure is designed to ensure consistent interpretation output across the documents: invoice, purchase order, and delivery note.

In the fourth phase, the web interface for user interaction between the multi-agent system and Streamlit is designed. Streamlit allows for the implementation of simple, powerful, and asynchronous interfaces, which is useful for integrating multiple systems. The components are integrated there, facilitating document capture, image-to-text conversion, and the presentation of information comparing raw material orders.

In the fifth phase, the complete system undergoes experimental validation. This involves 35 tests using real data from a company that manufactures personal care products. Match and difference indicators were calculated for the documents required for each order, as well as text similarity using the Ratcliff-Obershelp algorithm [18] and numerical similarity using a normalized function based on the relative difference between values. This phase allows for the identification of inconsistencies, false positives, and limitations through objective metrics of the designed system's performance.

Figure 1 presents the diagram that explains each of the phases and activities of the methodology, showing the integrations of the tools used, which facilitate replicating the process in similar scenarios.

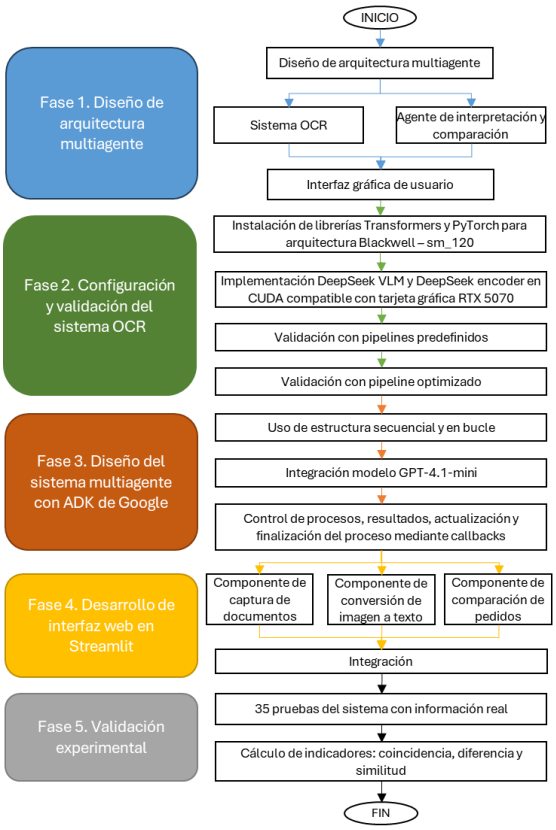

Fig. 1. *Detailed methodology*

3. RESULTS

3.1. Multi-agent system architecture

The problem of cross-validating documents is subdivided into four computational subproblems: the first corresponds to text extraction, the second to the standardization and mapping of plain text in natural language to JSON schemas, the third to semantic comparison and logical cross-validation between the standardized information, and the fourth to visual and functional communication between the user and the system. This allows for the selection of the appropriate technology for each subproblem. Thus, the design of the multi-agent system architecture is structured, and it is presented In Fig. 2, where Deepseek is used to interpret the texts in the images of the documents related to the process of purchasing raw materials and supplies (order, delivery note, invoice, purchase order) to be analyzed, the multi-agent system is made using Google ADK, which is useful to build the architecture to control and monitor the flow of the interpretation and comparison process, and finally Streamlit is used [19] for the web interface that will allow the uploading of files, displaying the results of DeepSeek OCR and the multi-agent system.

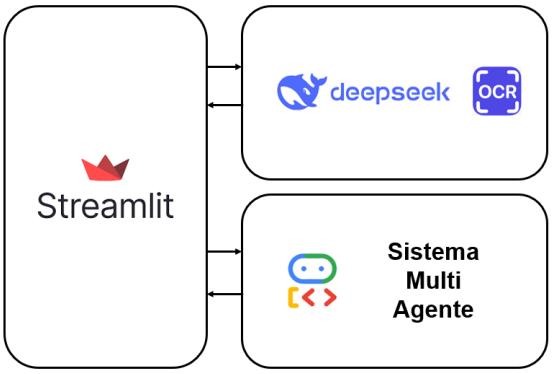

Fig. 2. *Architecture of the multi-agent* ***system***

Five criteria were considered for the design. First, the strict separation of perception and reasoning responsibilities to mitigate performance degradation. Deepseek-OCR is used exclusively for text digitization, as it is optimized for sequential inference interpretation in local environments with hardware acceleration (GPU), requiring video memory (VRAM). Inferential capabilities were assigned to the GPT-4.1-mini language model, which operates on the previously extracted text and therefore does not require massive visual capabilities but does require advanced symbolic processing to interact with structured JSON schemas. Second, orchestration was achieved through a specialized multi-agent pipeline to manage both data and task flows. This pipeline is supported by ADK, which was selected after comparing it with traditional architectures such as monolithic scripts, simple linear chains, and centralized single-agent systems. Third, a generator agent and a validator agent were implemented to prevent errors such as omitted sections or premature process termination, ensuring data validation during the generation phase. The fourth step involves decoupling through an external parametric document schema in Excel to facilitate future maintenance and operational flexibility of the solution, allowing a process analyst to modify variables, data types, and default values without compiling the system. The fifth step is the flow of information between agents using the ADK session state as a centralized data conduit, eliminating direct message exchange between agents.

The interaction mechanisms between the agents prevent direct communication between them; everything is handled centrally with the session state. File control is done using a counter so that the pipeline can select the organized documents.

3.2. OCR System

The DeepSeek model was initially configured [20] locally, and its functionality was subsequently validated using predefined pipelines. Based on this, a custom pipeline was developed for initializing the models (SAM and VLM), optimizing processing and memory costs for each inference.

3.3. Multi-agent system

The multi-agent system was built using the agent flow concept referenced in the Google ADK documentation [16], which consists of a base agent that can be extended by means of an LLM (Local Learning Management) agent, workflow agents (sequential, parallel, looped, and/or with consumption logic). Based on this, the following structured agents are used:

- ADK Interpreter Agent: It is assigned a role exclusively for the transcription of semantic data, without making assumptions about missing data, it relates the plain text of Deepseek-OCR with the Excel file loaded in real time forcing the output to valid JSON.
- LLM-based agent: It uses Large Language Models (LLM) as its main engine to understand natural language, reason, plan, generate responses, and dynamically decide how to proceed or which tools to use, making it ideal for flexible, language-centric tasks. It is configured with the role of logical auditor and decision-maker, based on the instruction to verify the correspondence between the tax identification number, rounding tolerances of 0.01 monetary units, and semantic consistency in the description of raw materials. The output is a JSON-encoded report with the approved or rejected concept and a difference matrix.
- Sequential Agent: Executes its sub-agents in the order specified in the list.
- Loop Agent: Executes its sub-agents in a loop (i.e., iteratively). It repeatedly executes a sequence of agents for a specified number of iterations or until a termination condition is met.

For the operation of these agents, the process flow illustrated in Fig. 3 was defined, integrating the output of Deepseek OCR.

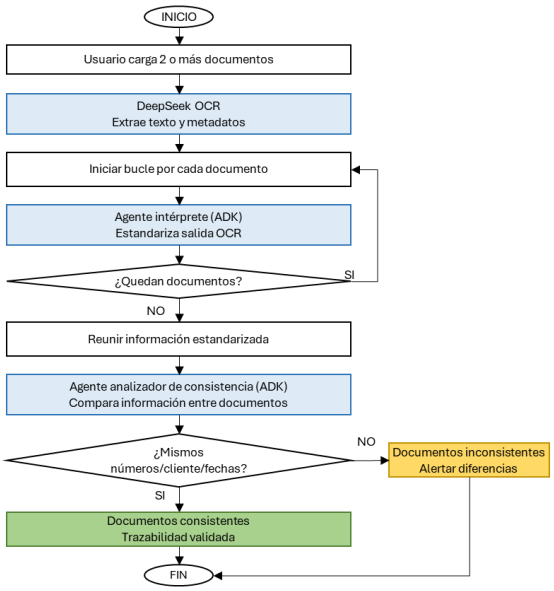

***Fig. 3. Multi** -agent system flowchart*

For the control and monitoring of the process, because it is a flow of agents, “Callbacks” are used as operational control and synchronization mechanisms, which in the framework there are three types focused on the executions of the flow and each type has a control option before and after an execution.

These methods allowed the process to be controlled in the following aspects:

- Agent Callbacks:
  - Counter for monitoring.
  - Storage of results.
  - Monitoring of results.
- Tools Callbacks:
  - Control for ending iterations in the loop.
- Model Callbacks:
  - Dynamic prompt updates.

Additionally, this framework addresses important concepts in the operational flow:

- Session: Regarding the current conversation thread
  - It represents unique and continuous interaction between a user and their agent system.
  - It contains the chronological sequence of messages and actions performed by the agent (referred events) during that specific interaction.
  - A session can also contain temporary data (states) relevant only during this conversation.

- States: Data within the current conversation
  - Data stored within a session.
  - It is used to manage information relevant only to the current active conversation thread.

These concepts are useful because both agents and callbacks make use of states that are stored in the current processing session. Therefore, the following states are defined for the agent flow to function:

- Number of files (deterministic integer counter used as a control index for document selection).
- List of texts extracted from each document by DeepSeek OCR (collection sequentially indexed with the plain text strings).
- List of processed file names (text list).
- Counter of files processed by the agent flow (initially integer 0).
- List of texts interpreted by the agent flow (list of texts that is initially empty).
- Interpretation scheme data (parametric reference structure with mandatory fields and types) required by the organization.

According to the process flow initially proposed, the files uploaded by the user and the text output generated by the DeepSeek model are available. The "Interpretation Scheme" state is a data structure obtained from a parametric table in Excel that allows defining the variables, the type of variable to extract and interpret, and contains the spaces shown in Table 1. This table allows parameterizing and controlling the responses of the interpreting agent, which must both classify the type of document and generate the predefined structured output.

Table 1: *Predefined Table*

Space	Description
Document_type	Invoice, Delivery Note, Purchase Order
ID_variable	Sequential identifier
Hierarchy	1=document, 2=object/array template, 3=atomic field
Variable	Exact name in the output JSON
variable_type	string, integer, array, object
Example	Guiding value for the LLM
Default_value	Value if the field is not found in the OCR

Table 2 presents a concrete example for constructing the scheme for an order.

Table 2: *Example of a predefined table*

Hierarchy	Variable	Guy	Default
1	supplier_name	string	""
1	raw materials	array	[]
2	raw_material	object	{name:"", quantity:0, presentation:"", price:0}
3	raw_material_name	string	""
3	quantity_of_raw_material	integer	0
3	raw_material_presentation	string	""
3	raw material price	integer	0

To ensure the validation of the information in JSON format during the verification phase, the validator agent checks for syntactic conformity, schema integrity, and adherence to the data type in each field. If errors are identified, the structure is rejected, the error is detailed, and the generation loop is restarted a maximum of three times to prevent infinite loops. Table 3 shows representative examples of the process's input and output. The sequential agent takes the structured data and dynamically concatenates it at the interpreter agent's prompt, delimiting the text search. The interpreter then searches for patterns that correspond to the table's constraints. The generator agent outputs the intermediate JSON structure, which the consistency analyzer compares. Finally, the interpreter agent describes the audit results.

Table 3: *Examples of process inputs and outputs*

Type	Details
Entry to the pipeline (post-OCR)	data = { "file_0": [text_ocr_order, "path/order.png"], "file_1": [text_ocr_invoice, "path/invoice.png"], } run_pipeline_sync(data, folder_session_id="session_001")
Interpreter agent exit	RAW MATERIAL: Blue Fresh Fragrance | QUANTITY: 12 Kg | PRICE: $64,000
Standardized JSON output	{ "supplier_name": "ASYS AND PROCESSES SAS", "raw_materials": [{ "raw_material_name": "Blue Fresh Fragrance", "raw_material_quantity": 12, "raw_material_presentation": "12 Kg", "raw_material_price": 600 }]
Analyzer output - Success story	{ "corresponden": true, "analisis": "File_1 was identified as an INVOICE. Suppliers ANDLER SAS and HANDLER SAS are equivalent. Quantities and products match.", "documentos_processados": ["invoice", "delivery note", "order"], "documentos_diferentes": [], "diferencias_detectadas": [] }
Analyzer output - Case with inconsistencies	{ "match": false, "analysis": "Invoice vs order was compared. Differences in supplier and quantity.", "documents_processed": ["invoice", "order"], "different_documents": ["file_1_order"], "differences_detected": ["comment":

	"file_1_order", "different_field": "supplier", "invoice_value": "CONQUIMICA SAS", "compared_value": "Industrias Novaquim SAS" }, { "compared_document": "file_1_order", "different_field": "item_size", "_item": "_0", "subfield": "quantity", "invoice_value": 200, "compared_value": 215 }] }
Enriched output	{ "corresponden": false, "detalle_comparacion_metricas": { "file_1_pedido":[{"campo": "proveedor", "similitud": 0.372, "similitud_numerica": null}, {"campo": "item_detalle", "subcampo": "cantidad", "similitud": 0.333, "similitud_numerica": 0.930}] } }

3.4. Web interaction interface

A web application is developed to interact with the file interpretation and comparison system, as shown in Fig. 4 shows the initial screen where the images of the documents associated with each purchase are uploaded and sent to the system.

Fig. 4. Initial *screen of the multi-agent system*

Fig. 5 shows a preview of the document load (2 files) and the execution of the system's DeepSeek OCR where each colored box demarcates the fields to be compared.

Fig. 5. Screenshot *of document loading and execution of DeepSeek OCR of the multi-agent system*

Finally, for this example case, the response generated by the multi-agent system is: *"There are only two documents to compare: the invoice (file_0) and the purchase order (file_1). The supplier on the invoice is CONQUIMICA SAS, while on the purchase order it appears as Industrias Novaquim SAS, which is the customer, so the investment was corrected. A difference in the supplier is detected after the correction. In addition, the quantity of linear sulfonic acid differs in the items: 200 on the invoice and 215 on the purchase order, and the unit price also differs (8263 vs. 9038). The ethyl alcohol matches in quantity and price. There is no delivery note to compare."*

3.5. Experimental validation

For experimental validation, real data from a company dedicated to the manufacture and marketing of liquid cleaning products were used. A total of 35 tests were executed, one of which did not contain documents, and 34 tests in which between 1 and 3 reference files were loaded per case (invoices, orders, purchase orders, and/or delivery notes). 28 tests with documents were executed correctly, corresponding to an 82.353% success rate. The failures occurred in the response generated by the "consistency analyzer agent" because they were not structured responses or did not comply with the parameters established in the instructions.

According to the information available in the company, the distribution of the number of documents studied in each case, in the successful tests, can be observed on Fig. 6. A test was done with a single document (3.57%) to verify that the identification of the correspondence was false, a result that coincided with what was expected, 11

tests were executed (39.29%) comparing 2 documents and 16 (57.14%) comparing 3 files.

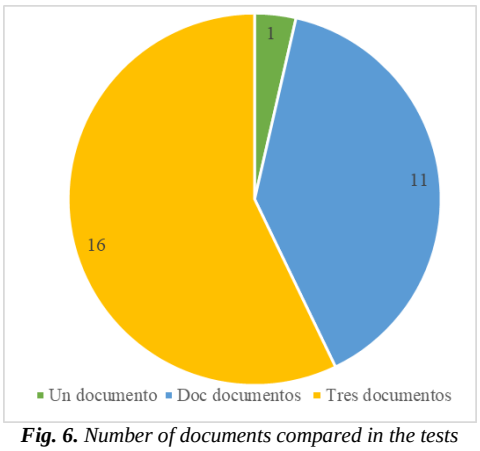

Fig. 6. *Number of documents compared in the tests*

The use of similarity metrics, both textual and numerical, allows for a more precise analysis of the differences detected, providing useful and real-time information for those responsible for managing raw material orders.

The comparison of information in the documents is performed for the following fields: supplier, quantity, unit of measure, product description, and unit price. Therefore, similarity metrics are calculated based on qualitative (supplier and product) and quantitative (price, quantity) values to refine the system analysis, using the Ratcliff–Obershelp algorithm for text (1) and for quantity (2), respectively.

$$Similitud(A,B) = \frac{2*M}{|A+B|} \quad (1)$$

Where M= number of matching characters in the longest common substrings, A = length of A, and B = length of B.

$$Similitud(A,B) = 1 - \frac{A-B}{\max(A,B)} \quad (2)$$

Where A = value of A and B = value of B

These metrics are only calculated when the system reports inconsistencies, allowing us to detect which of the fields (supplier, quantity, unit of measure, product description and unit price) have differences to promptly inform those responsible for raw material purchasing management. Figure 7 shows the results of these metrics applied by the system in the tests at a general level. Numerical similarities are observed with extreme values that correspond to little or much coincidence. Verbal similarity has a more uniform behavior, but with a greater concentration in low data that corresponds to low similarity.

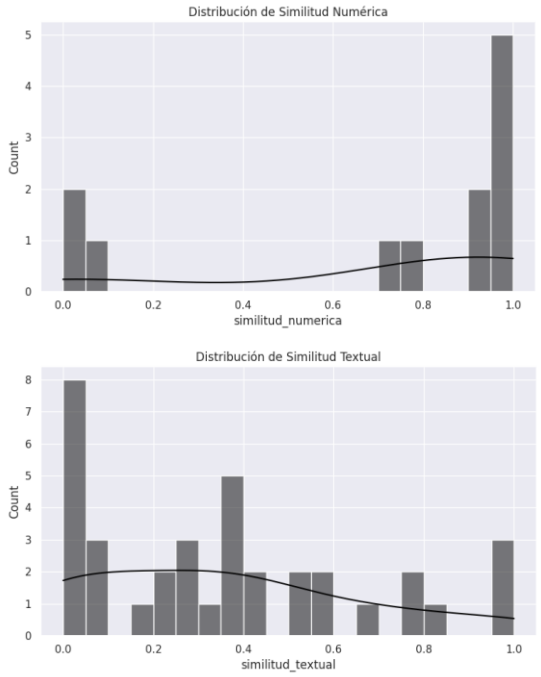

Fig. 7. *Results of numerical and textual similarity tests*

It was evident that even when the system finds empty fields, it performs comparisons and therefore reports zero similarity values, but it also yields similarity values of 1 corresponding to identical fields, making it necessary to review the parameterization defined for interpretations at these two extreme points.

For the performance evaluation of the prototype, experimental validation was designed to measure two main capabilities of the system: the quality of information extraction from different types of documents and the ability of the analyzing agent to identify documentary consistency or inconsistency with respect to a reference criterion.

The validation was performed in 35 cases, which together grouped 89 processed documents (including invoices, orders and delivery notes) using the following analysis methods:

- Document extraction evaluation: The automatically extracted information was compared against the variable schema, calculating the percentage of completeness (3) for each case between the ratio of correctly extracted fields (CEC) to the expected fields (according to the schema) (CE). This metric

allows quantifying the ability to extract information regardless of the result.

$$Completeness = \frac{CEC}{CE} * 100 \qquad (3)$$

- Analyzer agent evaluation: Subsequently, the analysis agent received only the structured output from the pipeline and generated a binary verdict of Consistent (True): No relevant differences were identified between documents. Inconsistent (False): There are differences that require review. No verdict: The system failed to produce a valid or structured output.

The evaluation showed that the system achieved an average extraction completeness of 78.3%, with a median of 81.3%, indicating that in most cases it recovered more than 80% of the expected information. The results are summarized in Table 4.

Table 4: *Validation results*

Metrics	Result
Cases evaluated	35
Processed documents	89
Average completeness	78.3%
Median completeness	81.3%
Expert verdicts available	34

Based on the system's behavior, it can be defined that the prototype presents a stable and reliable documentary interpretation, although information losses persist associated with structural variability between documents and documents with low quality.

Additionally, an analysis was performed for each type of document to identify the impact of each one on the performance of the system, the results of which are related in Table 5.

Table 5: *Results by document type*

Guy	Bill	Order	Remission
Documents	35	36	18
Extracted	299	424	80
Expected	389	501	111
Global completeness	76.9%	84.6%	72.1%
Average per case	74.3%	84.6%	72.8%

Order document type had the best extraction rate, achieving 84.6% completeness, while *the Delivery Note* had the lowest performance (72.1%), due to greater complexity or less standardized format. Furthermore, when the behavior of the analyzer agent was evaluated (without considering the manual reference), the decision distribution obtained is tabulated in Table 6.

Table 6: *Distribution of verdicts*

Verdict	Cases	Stake
Inconsistent (False)	20	57.1%
Consistent (True)	9	25.7%
No verdict	6	17.1%

There is a trend towards the detection of inconsistencies, desirable behavior in document reception and validation scenarios.

To measure system reliability, the system's automatically generated suggestion was compared against the manual review performed for each case. The detection of document inconsistency (False) was defined as a positive result, which allowed for the creation of the matrix in Table 7.

Table 7: *Results Matrix*

	Expert: Inconsistent	Expert: Consistent
Agent: Inconsistent	16	4
Agent: Consistent	4	4

From this matrix, the related ranking metrics in Table 8 were calculated.

Table 8: *Ranking Metrics*

Metrics	Worth
Evaluable cases	28
Non-evaluable cases	7
Precision	0.800
Accuracy	0.714

The results show that the system achieves a balance between detection capability and stability, defining a consistent response both for identifying real inconsistencies and for limiting false positives. However, the 71.4% accuracy indicates that there is still room for improvement related to extraction errors that affect later stages of system analysis and definition, instances where the agent fails to generate a valid output.

Factors that contribute to success include high-quality OCR, clear supplier identification on the invoice, correspondence between reference codes on the purchase order and invoice, and the availability of all relevant documentation for the process. Unsuccessful cases arise from interpretation errors, not analysis errors, as fields that are not understood are simply marked as null. The primary cause of these errors is confusion between the supplier and the customer. Although the analysis agent detects the issue, it merely reports the error without correcting it.

This document comparison procedure is currently performed manually by a company employee, an activity that takes approximately 7 minutes to compare two documents (invoice versus purchase order), and about 10 minutes if a delivery note is included. On several occasions, the comparison is not accurate due to human error. In contrast, the multi-agent system takes less than 1 minute (57.2 seconds) and 2 minutes (80.85 seconds) for two and three documents, respectively, also contributing to the elimination of human error. It's worth noting that the code does not use timers; these times were calculated from the timestamps of tests performed on a system with an NVIDIA GeForce RTX 5070 GPU (12 GB VRAM), an AMD Ryzen 9 9950X 16-core CPU, 16 GB of RAM (WSL2), Windows 11 + WSL2 operating system, and NVIDIA driver 591.86.

The system's practical limitations are related to its reliance on specialized hardware, external services, reproducibility, and scalability. First, the sequential operation of the OCR is not CPU-based but requires a GPU with more than 16 GB of VRAM. Second, the LLM (GPT-4.1-mini) connects via API, making latency dependent on network and API availability. A zero temperature is used to reduce variability, but the service may update the model. Alternatively, Gemini 2.0 Flash could be used (not yet evaluated). Regarding reproducibility, only cases without manual annotations are available, preventing the calculation of general-purpose metrics for this type of application. Therefore, it is proposed to add the corresponding annotations, and since the maximum number of iterations is 3, this could be increased proportionally if more documents are required. As for scalability, the sequential nature of the OCR makes processing large batches of documents slow.

4. CONCLUSIONS

The methodology phases were crucial for the multi-agent system's functionality. The first two phases established an optimized OCR pipeline to reduce memory usage and computational requirements, thanks to the processing of visual tokens with Meta's SAM and DeepSeek's decoder, ensuring efficient data capture. In the system design phase, the use of sequential and looping agents allowed for the validation to be divided into classification, interpretation, and comparison tasks, using the GPT-4.1-mini model to achieve consistent interpretation. The development of the web interface facilitates document uploading and report viewing, making the system user-friendly for purchasing managers. Finally, experimental validation enabled the identification of errors, the calculation of similarity metrics, and the detection of limitations, allowing the transition from a theoretical design to a tool validated with real-world data.

The design, implementation, and verification of the multi-agent system for validating raw material orders, using an architecture that integrates the DeepSeek OCR model with a Multi-Agent system based on the ADK framework and a web interface created in Streamlit, enabled the successful execution of the information interpretation and comparison flow in 82.353% of the tests performed. This demonstrated functional capability in real-world operating scenarios for analyzing heterogeneous documents related to the raw material purchasing process, such as invoices, purchase orders, requisitions, and delivery notes. The parametric table in Excel is a significant advantage, functioning as a dynamic schema that can be modified by non-expert users thanks to its three callback injections. Separating OCR and reasoning allow for the integration of local OCR operation with agent operation via API, resulting in independently evaluable layers. Finally, the calculation of quantitative textual and numerical similarity metrics per field was validated with field analytics measurements.

Two types of limitations were identified. The first relates to the generation of unstructured responses in 17.647% of cases, due to the lack of defined output schemes. The second is associated with the presence of extreme values (0 and 1) in the similarity metrics results, indicating the need to adjust the system. The bottleneck is interpretation, not analysis. Pipeline failures arise from poor classification or incomplete extraction by the interpreter, which functions perfectly when the JSON data is valid.

To optimize system accuracy and mitigate detected inconsistencies, a technical adjustment to the agent parameters and prompts is necessary, focusing primarily on disambiguation rules so that the interpreter agent correctly distinguishes between provider and client, thus eliminating erroneous comparisons against null values. Additionally, the output structure of the consistency analyzer agent needs strengthening to ensure strict compliance with the established JSON parameters and prevent execution errors. Furthermore, the acceptance thresholds of the Ratcliff-Obershelp algorithm for textual variables must be recalibrated to improve the

identification of qualitative differences; this is proposed as future work.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the support of the Nueva Granada Military University, where they are full-time faculty members. This work is derived from the research project entitled “Strengthening Order Reception and Raw Material Inventory Control Processes with the Support of Industry 4.0” INV-ING-4150, funded by the Vice-Rectorate for Research of the Nueva Granada Military University, 2025.

REFERENCES

[1] W. Chu, S. Yin, L. Huang, L. Lin, X. Wang, Z. Zhang and H. Li, "Verify-Agent: Large Language Model Multi-Agent for Intelligent Verification," in *2024 International Conference on Ubiquitous Computing and Communications (IUCC)*, Chengdu, China, 2024, doi:10.1109/IUCC65928.2024.00072.

[2] P. H. Luzolo, Z. Elrawashdeh, I. Tchappi, S. Galland and F. Outay, "Combining multi-agent systems and artificial intelligence of things: Technical challenges and gains," *Internet of Things*, vol. 28, p. 101364, 2024, doi:10.1016/j.iot.2024.101364.

[3] M. Z. R. Farazi, "Enhancing supply chain resilience with multi-agent systems and machine learning: a framework for adaptive decision-making," *The American Journal of Engineering and Technology*, vol. 7, no. 3, pp. 6-20, 2025, doi:10.37547/tajet/Volume07Issue03-02.

[4] M. Mousa, D. Van de Berg, N. Kotecha, E. A. del Rio Chanona and M. Mowbray, "An analysis of multi-agent reinforcement learning for decentralized inventory control systems," *Computers & Chemical Engineering*, vol. 188, p. 108783, 2024, doi:10.1016/j.compchemeng.2024.108783.

[5] G. Ziegner, M. Choi, H. M. C. Le, S. Sakhuja and A. Sarmadi, "Iterative Multi-Agent Reinforcement Learning: A Novel Approach Toward Real-World Multi-Echelon Inventory Optimization," *arXiv*, 2025, doi:10.48550/arXiv.2503.18201.

[6] N. Kotecha and A. del Rio Chanona, "Leveraging graph neural networks and multi-agent reinforcement learning for inventory control in supply chains," *Computers & Chemical Engineering*, vol. 199, p. 109111, 2025, do:10.1016/j.compchemeng.2025.109111.

[7] J. Hu, L. Xia, T. Huang and H. Wu, "A multi-agent deep reinforcement learning approach for multi-echelon inventory optimization and its application to the beer game," *Transportation Research Part E: Logistics and Transportation Review*, vol. 203, p. 104367, 2025, doi:10.1016/j.tre.2025.104367.

[8] G. Kim, T. Hong, M. Yim, J. Nam, J. Park, J. Yim, W. Hwang, S. Yun, D. Han and S. Park, "OCR-Free Document Understanding Transformer," in *Computer Vision – ECCV 2022. ECCV 2022. Lecture Notes in Computer Science*, vol. 13688, S. Avidan, G. Brostow, M. Cissé, G. Farinella and T. Hassner, Eds., Springer, 2022, doi:10.1007/978-3-031-19815-1_29.

[9] S. Chen, X. Guo, Y. Li, T. Zhang, M. Lin, D. Kuang, Y. Zhang, Z. Zhou and W. Chen, "Ocean-OCR: Towards general OCR application via a vision-language model," *arXiv*, 2025, doi:10.48550/arXiv.2501.15558.

[10] L. Hamdi, A. Tamasna, P. Boisson and T. Paquet, "VISTA-OCR: Towards generative and interactive end to end OCR models," *arXiv*, 2025, doi:10.48550/arXiv.2504.03621.

[11] G. Monteiro, L. Camelo, G. Aquino, R. Fernandes, R. Gomes, A. Printes, I. Torné, H. Silva, J. Oliveira and C. Figueiredo, "A Comprehensive Framework for Industrial Sticker Information Recognition Using Advanced OCR and Object Detection Techniques," *Applied Sciences*, vol. 13, no. 12, p. 7320, 2023, doi:10.3390/app13127320.

[12] R. Sapkota and M. Karkee, "Object detection with multimodal large vision-language models: An in-depth review," *Information Fusion*, vol. 126, no. Part A, p. 103575, 2026, doi:10.1016/j.inffus.2025.103575.

[13] K. Medini, D. Romero and T. Wuest, "Developing a multi-agent system to support multi-variant production ramp-up management," *Smart and Sustainable Manufacturing Systems*, vol. 5, no. 1, pp. 129-147, 2021, doi:10.1520/SSMS20200082.

[14] Z. Zhao, D. Tang, C. Liu, L. Wang, Z. Zhang, H. Zhu, K. Chen, Q. Nie and Y. Ji, "A Large language model-based multi-agent manufacturing system for intelligent shopfloors," *Advanced Engineering Informatics*, vol. 69, no. Part A, p. 103888, 2026, doi: 10.1016/j.aei.2025.103888.

[15] V. Jannelli, S. Schöpf, M. Bickel, T. Netland and A. Brintrup, "Agentic LLMs in the supply chain: towards autonomous multi-agent consensus-seeking," *International Journal of Production Research*, p. 1–31, 2025, doi:10.1080/00207543.2025.2604311.

[16] Google, "Agent Development Kit," 2025. [Online]. Available: https://google.github.io/adk-docs/agents/. [Accessed 28 Enero 2026].

[17] A. Bandi, B. Kongari, R. Naguru, S. Pasnoor and S. Vilipala, "The Rise of Agentic AI: A Review of Definitions, Frameworks, Architectures, Applications, Evaluation Metrics, and Challenges," *Future Internet*, vol. 17, no. 9, p. 404, 2025, doi:10.3390/fi17090404.

[18] H. Guo, Y. Li, Y. Liu and W. Li, "Evaluation of string comparators for record linkage in Chinese environment," *International Journal of Wavelets, Multiresolution and Information Processing*, vol. 20, no. 6, p. 2250023, 2022, doi:10.1142/S0219691322500230.

[19] C. Nantasenamat, A. Biswas, J. Nápoles-Duarte, M. I. Parker and R. L. Dunbrack, "Chapter 27 - Building bioinformatics web applications with Streamlit," in *Cheminformatics, QSAR and Machine Learning Applications for Novel Drug Development*, K. Roy, Ed., Academic Press, 2023, pp. 679-699, doi:10.1016/B978-0-443-18638-7.00001-3.

[20] H. Wei, Y. Sun and Y. Li, "Deepseek-OCR: Contexts optical compression.," *arXiv* , 2025. https://doi.org/10.48550/arXiv.2510.18234.