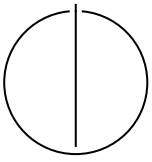
SCHOOL OF COMPUTATION,
INFORMATION AND TECHNOLOGY —
INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Reliable Multi-Agent Systems for Automated Technical Data Acquisition and Validation

Yange Zheng



SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

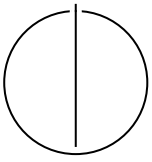
TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Reliable Multi-Agent Systems for Automated Technical Data Acquisition and Validation

Zuverlässige Multi-Agenten-Systeme für die automatisierte technische Datenerfassung und -validierung

Author:	Yange Zheng
Examiner:	Prof. Dr. Viktor Leis
Advisor:	Dr. Alexander Schiffmacher
Supervisor:	Joe Yu
Submission Date:	01.07.2025



I confirm that this master's thesis is my own work and I have documented all sources and material used.

Munich, 01.07.2025 Yange Zheng

Acknowledgments

I owe special thanks to **Dr. Alexander Schiffmacher** (BMW AG), whose guides thesis from start to finish, with invaluable insights and support throughout the process.

I would also like to thank **Joe Yu** (BMW AG & TUM) for providing timely feedback and encouragement on key milestones.

Many thanks to **Stefan Merbele** (BMW AG) for his generous support with data access and infrastructure.

...

...

Finally, I am grateful to my family and friends for their unwavering support and encouragement during this journey.

Munich, July 2025 Yange Zheng

Abstract

In industrial procurement, engineers often work with **Quotation Analysis Forms (QAFs)** provided by suppliers—semi-structured documents containing part descriptions, electrical specifications, and inconsistent identifiers. While Manufacturer Part Numbers (MPNs) are typically present, they are often embedded in noisy text alongside other attributes, making reliable extraction and interpretation difficult. Automating the full pipeline—from QAF to datasheet to validated component specifications—requires coordinating diverse tools: language models, web search, document parsing, and schema validation. Rigid scripts and monolithic workflows tend to break in the face of data variation or missing fields. This thesis addresses that gap by designing a **modular multi-agent system** where each agent specializes in one task (e.g., MPN extraction, datasheet retrieval, VLM parsing), enabling flexible, fault-tolerant, and interpretable orchestration across real-world data acquisition scenarios.

Contents

1 Introduction

Example QAF Line (synthetic)

```
10UF 25V X5R 0603 C1608X5R1E106K080AC RoHS Reel 4000pcs $0.073
```

Figure 1.1: Typical supplier-submitted QAF line (synthetic example).

1.1 Background ✌

The digitalisation of automotive supply chains has accelerated in recent years, yet many data-critical workflows still depend on semi-structured documents and manual interpretation. A prominent example at BMW is the *Quotation Analysis Form* (QAF), a file exchanged during supplier negotiations that lists part designations, preliminary prices, delivery conditions, and essential technical details. Although QAFs frequently include an identifier resembling a Manufacturer Part Number (MPN), suppliers often embed the MPN within longer descriptive strings—alongside electrical characteristics and ordering information—or omit it altogether. As shown in Figure 1.1, QAF lines typically appear as unstructured, single-line entries that conflate multiple attributes in free-form text. Consequently, engineers must manually inspect each line, identify or correct the MPN, locate the corresponding datasheet, and extract key specifications—such as maximum voltage or operating temperature—for input into BMW's internal cost and qualification systems. With several thousand parts assessed per program phase, this *ad hoc* process consumes hundreds of engineer-hours and remains susceptible to transcription errors.

1.2 Problem Statement ✌

Previous attempts to automate QAF processing primarily relied on regular expressions to extract Manufacturer Part Numbers (MPNs) from raw supplier text. While effective in narrowly defined cases, these approaches were brittle—minor variations in formatting, inconsistent delimiters, or embedded product descriptions frequently caused extraction to fail. A further limitation was ambiguity: when the regular expression returned a

null result, it was unclear whether no MPN was present or if the pattern simply failed to match a valid one. To address these issues, large language models (LLMs) offer a more robust alternative. By reasoning over free-form text and recognizing contextual cues, LLMs can identify MPNs even when they are embedded in descriptive strings or surrounded by supplier-specific notations.

1.3 Research Objectives ✌

The central objective of this thesis is to design, implement, and evaluate a **reliable multi-agent system** that transforms noisy QAF inputs into validated, structured component specifications. To operationalise this goal, the work addresses three research questions:

RQ1. How accurately can a language-model agent extract correct MPNs from semi-structured QAF strings?

RQ2. Can cooperative retrieval agents—one using a distributor API, the other web search—achieve near-complete datasheet coverage across diverse suppliers and part formats??

RQ3. What precision and recall can a vision-language parsing agent attain on key electrical specifications?

1.4 Proposed Approach ✌

To answer these questions, the thesis proposes a modular pipeline whose agents each address a single sub-task:

1. An **MPN Extraction Agent** uses GPT-4 with prompt engineering and regex to isolate MPN from QAF text.
2. Two **Retrieval Agents** operate in parallel:
   (i) An *API Agent* queries the Octopart distributor API to obtain structured part metadata and datasheet links.
   (ii) A *Datasheet Agent* performs a Google-based search to locate datasheet PDFs directly, prioritising high-confidence domains (e.g., manufacturer or distributor sites). And leverages Vision Language Models (VLMs) to extract the key electrical specifications from the retrieved datasheets.
3. A **Validation Module** enforces JSON-Schema constraints and physical range rules and flags low-confidence extractions for manual review.

4. Finally, a **Cost-Prediction Model** is trained on the extracted specifications, demonstrating the economic value of the pipeline's output.

1.5 Contribution ✌

The thesis offers four concrete contributions:

1. A fault-tolerant multi-agent framework that integrates LLMs, web search, distributor APIs, and VLM parsing into a single orchestrated workflow.
2. Empirical benchmarks reporting MPN extraction accuracy, datasheet retrieval success, field-level precision/recall, and end-to-end latency.
3. A downstream cost-prediction experiment showing that structured specs extracted by the system improve price-estimation accuracy, thus underlining direct business impact.

1.6 Thesis Structure ✌

The remainder of this document is organised as follows: Chapter 2 surveys literature on language-model agents, vision-language models, and document AI in procurement. Chapter 3 formalises the challenges inherent to QAF processing. Chapter 4 details the proposed multi-agent architecture, while Chapter 5 describes implementation specifics, including prompts, API wrappers, and validation logic. Chapter 6 presents the experimental setup and quantitative results. Chapter 7 demonstrates the downstream cost-modelling application. Chapter 8 reflects on limitations and industrial implications, and Chapterion 9 concludes the thesis with avenues for future work.

2 Related Work

This chapter surveys prior research relevant to the proposed multi-agent pipeline, grouped into four areas: (i) tool-using language-model agents, (ii) vision-language models for document understanding, (iii) multi-agent orchestration frameworks, and (iv) document-AI applications in procurement and engineering.

2.1 Tool-Using Language-Model Agents

Large language models have been extended with *tool use* capabilities to overcome context-length limits and provide up-to-date information. ReAct [**yao2023react**] interleaves chain-of-thought reasoning with executable actions, while the OpenAI function-calling interface popularised structured tool invocation in production systems. LangChain[1] offers an open-source abstraction layer for wrapping external tools (APIs, SQL, web search) behind LLM calls. AutoGPT and Voyager [**wang2023voyager**] push autonomy further but lack guarantees on reliability—highlighting the need for validation mechanisms such as those proposed in this thesis.

2.2 Vision-Language Models for Document Understanding

Document AI has progressed from OCR-centric pipelines to transformer-based models that combine layout and text. LayoutLMv3 [**xuhuang2022layoutlmv3**], DocPrompting [**li2023docprompt**], and StrucTexT [**li2021structext**] incorporate visual and spatial cues for table extraction. Recent work extends general VLMs (BLIP-2, GPT-4 Vision) to technical documents, but evaluation on complex engineering datasheets remains limited. Our parsing agent adopts GPT-4 Vision and focuses on reliability and schema-validated output rather than raw captioning.

2.3 Multi-Agent Orchestration Frameworks

Early agent systems in NLP used finite-state controllers; recent research revisits *LLM-driven* agents. MetaGPT [**hong2023metagpt**] assigns specialised roles (product manager,

[1] `https://www.langchain.com`

engineer) to different LLM instances. CrewAI[2] and OpenAI's "Assistants" API provide abstractions for role-based coordination. Most studies focus on qualitative demos; quantitative benchmarks on noisy industrial tasks, such as QAF processing, are scarce. Our work contributes empirical results on orchestration efficacy (success, latency) in a production-grade setting.

2.4 Document-AI in Procurement and Engineering

Prior industry case studies address invoice OCR, purchase-order matching, and part catalogue normalisation [**chung2020invoiceai**; **koch2022bom**]. Datasheet processing is typically rule-based or limited to text segments [**rahman2021datasheet**]. This thesis fills that gap and releases a benchmark on 1 000 QAFs to foster reproducible research.

2.5 Research Gap

- Existing LLM agents excel in controlled benchmarks but lack validation and provenance when applied to safety-critical engineering data.
- VLMs demonstrate impressive captioning accuracy yet are rarely evaluated on fine-grained numeric specs inside technical PDFs.
- No prior work jointly optimises extraction *and* downstream business value (e.g. cost prediction) within an end-to-end industrial pipeline.

These gaps motivate the multi-agent system and evaluation strategy proposed in the following chapters.

[2] `https://github.com/joaomdmoura/crewAI`

3 Problem Setting

3.1 Characteristics of Quotation Analysis Forms (QAFs)

A Quotation Analysis Form (QAF) is a semi-structured document exchanged during early supplier negotiations. At BMW, QAFs appear in multiple digital formats—CSV exports, Excel sheets, and supplier-specific PDF templates. Typical fields include:

- **Part Designation:** free-text description (e. g., "STM32F103C8 MCU 64 KB Flash TQFP48"),
- **Technical Specs:** one or more numeric attributes (voltage, current, temperature limits),
- **Price and MOQ:** preliminary unit price, minimum order quantity,
- **Manufacturer Part Number (MPN):** often concatenated with package or tolerance, occasionally missing or misspelled.

The absence of a universal schema means column order, header names, and punctuation vary widely across suppliers. Furthermore, multilingual notations ("Gehäuse", "Temp. Bereich") complicate rule-based parsing.

3.2 Core Challenges

C1: Inconsistent Identifier Extraction An MPN may be surrounded by package codes ("`-TR`"), voltage values, or marketing text, making pattern-matching unreliable. Example:

```
PI3USB9281ZLE_EX 20V Vbus Switch, TQFN-14, -40-85°C
```

C2: Heterogeneous Datasheet Sources Even after a correct MPN is found, the corresponding datasheet might be located on:

1. manufacturer domains (trusted),

2. distributor portals (moderately trusted),
3. third-party PDF mirrors (low trust, risk of 404 or captcha).

Selecting the "best" source requires ranking by reliability, download cost, and historical success.

C3: Complex Document Layouts Technical PDFs mix tables, figures, and footnotes. Numeric specs can appear in summary tables or deep in application sections. Standard OCR engines often split rows or mis-align columns, reducing extraction accuracy.

C4: Lack of Validation and Provenance Classic table-extraction pipelines output values without units or context. For safety-critical components, engineers must know *where* a value came from (page and bounding box) and whether it respects physical constraints (e. g., voltage > 0).

3.3 Formal Problem Definition

Let $q \in \mathcal{Q}$ denote a single QAF entry containing noisy text tokens. The task is to compute a validated specification record

$$\mathbf{s} = \left(\text{MPN},\ \text{Voltage}_{\text{max}},\ \text{Current}_{\text{max}},\ \text{TempRange}, \ldots\right)$$

such that:

1. **Correctness:** each numeric field equals the value printed in the authoritative datasheet.
2. **Completeness:** mandatory fields are present; optional fields are `null` if not in the document.
3. **Provenance:** every field carries evidence page, bbox.
4. **Latency:** average end-to-end processing time < 15 s per QAF.

3.4 Automation Requirements

Derived from the challenges and formal criteria, the target system must:

1. **Robust MPN Extraction** – tolerate noise, multi-language descriptors, and embedded package codes.

2. **Adaptive Retrieval** – combine distributor APIs and web search to maximise datasheet coverage while minimising failed downloads.
3. **Multi-Modal Parsing** – handle tables, prose, and regulatory footnotes using a vision-language model.
4. **Schema Validation** – enforce type, unit, and range checks; flag anomalies.
5. **Traceability** – store page coordinates and raw snippets for audit.
6. **Scalability** – process at least 3 000 QAF lines per day on commodity cloud hardware.

3.5 Mapping Challenges to Research Questions

Challenge	Requirement	Mapped RQ
C1	R1	RQ1
C2	R2	RQ2
C3	R3	RQ3
C4	R4, R5	RQ3 (validation aspect)
Performance	R6	All RQs (latency+throughput)

This mapping clarifies how each experimental evaluation in later chapters directly addresses an identified pain point in BMW's procurement workflow.

4 System Architecture

This chapter describes the overall design of the proposed multi-agent pipeline, the responsibilities of each agent, their communication strategy, and the supporting infrastructure for storage, logging, and fault-recovery.

4.1 High-Level Overview

Figure 4.1 illustrates the data flow from a raw QAF line to a validated specification record and a downstream cost prediction. Each rectangular node represents an autonomous agent implemented as a function-calling wrapper around a large language model (LLM) or vision-language model (VLM). Solid arrows indicate the primary execution path, while dashed arrows represent feedback loops used for retries and validation.

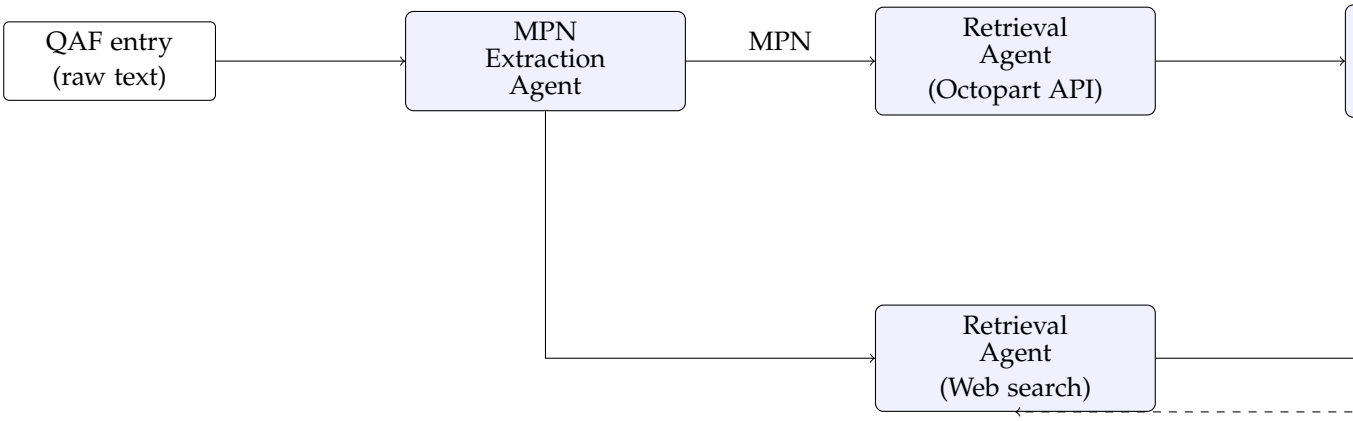


Figure 4.1: End-to-end architecture of the multi-agent pipeline.

4.2 Agent Responsibilities

MPN Extraction Agent Receives a raw QAF string and applies a few-shot GPT-4 prompt that asks the model to return a JSON object with the most likely manufacturer

part number. A lightweight post-processor removes package suffixes (e. g. "-TR") and checks against a regex whitelist of vendor prefixes.

Retrieval Agent (API) Queries the Octopart REST API with the extracted MPN. If the API returns `datasheets` with a direct PDF link and the file size is within 50 kB–20 MB, the link is forwarded to the Parsing Agent.

Retrieval Agent (Web) Uses Playwright to perform a Google search of the form `{MPN} datasheet filetype:pdf`. The top-3 PDFs are downloaded for validation.

Parsing Agent Given a PDF path, calls GPT-4 Vision with a chain-of-thought template: first summarise the datasheet sections, then extract `max_voltage`, `max_current`, `power_rating`, `temp_range`, and return JSON plus page/bbox evidence.

Validation Module Executes three layers of checks:

1. JSON-Schema validation (types + units),
2. range rules (e. g. $0 < V_{max} < 1000V$),
3. duplicate consistency (table vs. paragraph vs. Octopart value).

Failures trigger a dashed feedback edge to the Web Retrieval Agent for retry.

Cost Model Trains an XGBoost regressor on validated specs + historical SAP prices to demonstrate the economic relevance of the structured output.

4.3 Agent Coordination and Memory

Agents communicate via the LangChain executor, which offers:

- **Function-calling interface**: each agent exposes a JSON schema; GPT-4 picks the correct function based on the task.
- **Shared Memory**: RedisJSON stores intermediate results (`mpn`, `pdf_path`, validation flags) accessible to all agents.
- **Retry Logic**: a decorator intercepts exceptions, waits exponentially (1 s, 2 s, 4 s), and reroutes control to the fallback agent if attempts > 3.

4.4 Source Ranking Heuristic

For every candidate PDF URL u, the cost score is computed as

$$\text{cost}(u) = 0.2\,\text{domain}(u) + 0.4\,\text{pdfWeight}(u) + 0.4\,\text{failRate}(u), \tag{4.1}$$

where domain $= 0$ for manufacturer sites, 1 for trusted distributors, 2 otherwise; pdfWeight $= 0$ if the URL ends with `.pdf`, else 1; and failRate is an exponentially decayed average of previous download failures from the same host.

4.5 Data Storage and Logging

Validated JSON records are written to a PostgreSQL table with columns for `mpn`, `spec_json` (JSONB), `provenance` (JSONB), and an Azure Blob link to the PDF. OpenTelemetry traces (agent name, latency, token usage) are exported to Grafana for live monitoring.

4.6 Fault Tolerance

If both retrieval agents fail, the system stores a `status = "unresolved"` entry for manual triage. Parsing errors with low OCR confidence (< 0.7) raise a warning but still save partially validated fields, ensuring *at-least-once* data capture.

4.7 Security and Cost Considerations

All external calls pass through BMW's outbound proxy; API secrets are managed via Azure Key Vault. Estimated operating cost on an Azure D8s v5 VM plus one A10 GPU is ≈ 0.90 per 1 000 processed QAF lines, dominated by GPT-4 Vision tokens.

4.8 Mapping to Requirements

Table 4.1 summarises how architectural components satisfy the automation requirements defined in Chapter **??**.

Together, these design choices create a fault-tolerant, interpretable, and extensible architecture capable of meeting BMW's data-quality and throughput requirements.

Table 4.1: Requirements coverage by architectural component.

Requirement	Fulfilled by
R1 Robust MPN extraction	MPN Extraction Agent (LLM + regex)
R2 Adaptive retrieval	API + Web Retrieval Agents, heuristic ranking
R3 Multi-modal parsing	VLM Parsing Agent (GPT-4 Vision)
R4 Schema validation	Validation Module (JSON Schema, range rules)
R5 Traceability	Provenance storage (page, bbox)
R6 Scalability	Redis memory, async Playwright, GPU offload

5 Implementation

6 Evaluation

7 Cost Model

8 Discussion

9 Conclusion

Abbreviations

List of Figures

List of Tables