

Republic of Tunisia  
Ministry of Higher Education  
and Scientific Research

University of Sfax

The National School of Electronics and  
Telecommunications of Sfax



**Diploma in:**  
Data Engineering and Decision Systems

**Academic Year:**  
2025-2026

# Internship Report

*Presented at*

**The National School of Electronics and Telecommunications of  
Sfax**

**National Engineering Diploma in:  
*Data Engineering and Decision Systems***

*by*

**RAMI AYOUNI**

---

---

**Agentic AI for Automated CBAM Compliance  
Declarations in Odoo**

---

---

Supervisor : Mr Fahmi BEDBABIS

Project carried out at





---

# Acknowledgement

I would like to express my sincere gratitude to my supervisor, Mr. Fahmi BEDBABIS, for his guidance and availability throughout this project. Beyond the technical side, his advice on the project's business orientation was just as valuable, and his trust in a more ambitious technical approach — a genuine agentic tool-use loop rather than a simpler scripted pipeline — shaped this work considerably, from the early architectural choices through to the final evaluation methodology.

I would also like to thank **ninety. Startup House**, an accelerator and incubator in Sfax, for hosting this project and providing the working environment, resources, and startup-oriented atmosphere in which it was carried out. Working alongside a community of entrepreneurs and builders, rather than in isolation, gave this project a practical grounding it would otherwise have lacked.

I am equally grateful to the entire teaching and administrative staff of The National School of Electronics and Telecommunications of Sfax for the training and support provided throughout my studies, which laid the groundwork this project builds on.

Finally, I want to thank my family and friends for their unwavering encouragement and support, not only throughout this project but throughout my entire academic journey. Their patience and confidence in me made this work possible.



---

# Abstract

This internship, carried out at ninety. Startup House (Sfax), builds a system that prepares EU Carbon Border Adjustment Mechanism (CBAM) declarations for steel and aluminium imports inside Odoo 17. The design separates two concerns. A deterministic, unit-tested engine computes every compliance figure, while an agentic tool-use loop, driven by a locally hosted open-weight language model, decides which extraction, lookup and validation tools to call when reading supplier documents. Three safeguards protect the numeric output from model hallucination: facts already known to the system override the model's arguments, tool results are remembered across a job, and a measured emissions value is only written with a verbatim quote from the source document. The system runs on two machines linked by a private VPN. On 50 synthetic supplier documents, accuracy on the hardest field rose from 7% with the first model tested to 81% (95% CI 66–91%) with the final configuration. The evaluation also showed that a standard tool-calling benchmark wrongly ruled out a usable model, so model choice must be validated against the real execution path.

**Keywords:** agentic AI, large language models, tool use, CBAM, regulatory compliance, Odoo, hallucination mitigation.

## Résumé

Ce stage, réalisé chez ninety. Startup House (Sfax), développe un système qui prépare les déclarations du Mécanisme d'ajustement carbone aux frontières de l'UE (MACF/CBAM) pour les importations d'acier et d'aluminium dans Odoo 17. La conception sépare deux préoccupations. Un moteur de calcul déterministe et testé produit chaque valeur de conformité, tandis qu'une boucle agentique, pilotée par un modèle de langage open-weight hébergé localement, choisit les outils d'extraction, de recherche et de validation à appeler pour lire les documents des fournisseurs. Trois garde-fous protègent le résultat numérique contre les hallucinations du modèle : les faits déjà connus remplacent les arguments du modèle, les résultats d'outils sont mémorisés au cours d'une tâche, et une émission mesurée n'est enregistrée qu'accompagnée d'une citation exacte du document source. Le système fonctionne sur deux machines reliées par un VPN privé. Sur 50 documents synthétiques, la précision sur le champ le plus difficile passe de 7 % avec le premier modèle testé à 81 % (IC à 95 % : 66–91 %) avec la configuration finale. L'évaluation montre aussi qu'un benchmark standard d'appel d'outils avait écarté à tort un modèle utilisable : le choix du modèle doit être validé sur le chemin d'exécution réel.

**Mots-clés :** IA agentique, grands modèles de langage, utilisation d'outils, MACF, conformité réglementaire, Odoo, atténuation des hallucinations.



---

# Table of Contents

|                                                                        |             |
|------------------------------------------------------------------------|-------------|
| <b>List of Figures</b>                                                 | <b>VI</b>   |
| <b>List of Tables</b>                                                  | <b>VII</b>  |
| <b>Acronyms</b>                                                        | <b>VIII</b> |
| <b>General Introduction</b>                                            | <b>1</b>    |
| <b>I Project Context and State of the Art</b>                          | <b>3</b>    |
| I.1 Introduction . . . . .                                             | 4           |
| I.2 Host Organisation . . . . .                                        | 4           |
| I.3 State of the Art . . . . .                                         | 5           |
| I.3.1 EU CBAM as a Moving Regulatory Target . . . . .                  | 5           |
| I.3.2 Tool-Using LLMs and the ReAct Paradigm . . . . .                 | 6           |
| I.3.3 Hallucination Risk in High-Stakes Numeric Extraction . . . . .   | 8           |
| I.4 Critique of the State of the Art . . . . .                         | 9           |
| I.5 Problem Statement . . . . .                                        | 10          |
| I.6 Existing Methodologies for This Problem and Their Limits . . . . . | 12          |
| I.7 Proposed Solution . . . . .                                        | 13          |
| I.8 Conclusion . . . . .                                               | 15          |
| <b>II System Analysis and Design</b>                                   | <b>16</b>   |
| II.1 Introduction . . . . .                                            | 17          |
| II.2 Requirements Specification . . . . .                              | 17          |
| II.2.1 Functional Requirements . . . . .                               | 17          |
| II.2.2 Non-Functional Requirements . . . . .                           | 18          |
| II.3 Actors and Use Cases . . . . .                                    | 20          |
| II.4 Dynamic View: Sequence Diagram . . . . .                          | 22          |
| II.5 Static View: Data Model . . . . .                                 | 24          |
| II.6 Design Philosophy . . . . .                                       | 26          |

|                                                      |           |
|------------------------------------------------------|-----------|
| II.7 System Architecture . . . . .                   | 27        |
| II.8 Conclusion . . . . .                            | 28        |
| <b>III Implementation and Results</b>                | <b>29</b> |
| III.1 Introduction . . . . .                         | 30        |
| III.2 Model Selection Methodology . . . . .          | 30        |
| III.3 The Deterministic Calculation Engine . . . . . | 31        |
| III.4 The Agentic Pipeline . . . . .                 | 32        |
| III.5 Results . . . . .                              | 34        |
| III.6 Limitations and Threats to Validity . . . . .  | 38        |
| III.7 Conclusion . . . . .                           | 39        |
| <b>General Conclusion</b>                            | <b>41</b> |
| <b>Bibliography</b>                                  | <b>46</b> |



---

# List of Figures

|   |                                                                                                           |    |
|---|-----------------------------------------------------------------------------------------------------------|----|
| 1 | ninety. Startup House logo. . . . .                                                                       | 4  |
| 2 | The ReAct-style agent loop [1] as implemented for CBAM declaration processing. . . . .                    | 7  |
| 3 | Use case diagram of the CBAM Declaration Automation System. . . . .                                       | 21 |
| 4 | Sequence diagram for <i>Trigger Declaration Processing</i> . . . . .                                      | 23 |
| 5 | Class diagram of the seven core cbam models. . . . .                                                      | 25 |
| 6 | System architecture: Hardworker (LLM host) and Softworker (Odoo host), meshed over a private VPN. . . . . | 27 |
| 7 | The three-stage model-selection process. . . . .                                                          | 30 |
| 8 | Accuracy on the two hardest fields across evaluation stages. . . . .                                      | 34 |



## List of Tables

|    |                                                                                                              |    |
|----|--------------------------------------------------------------------------------------------------------------|----|
| 1  | Original ReAct versus this project's ReAct-style loop. . . . .                                               | 8  |
| 2  | Existing compliance-automation approaches vs. this project's design. . . . .                                 | 13 |
| 3  | Functional requirements. . . . .                                                                             | 18 |
| 4  | Non-functional requirements. . . . .                                                                         | 19 |
| 5  | Use case descriptions. . . . .                                                                               | 22 |
| 6  | The seven agent tools. . . . .                                                                               | 32 |
| 7  | Raw tool-calling benchmark (95% CI in brackets). . . . .                                                     | 35 |
| 8  | Real-harness comparison on the same 10 records. . . . .                                                      | 35 |
| 9  | Evaluation stages (95% Wilson CI in brackets; comprehension is direct / indirect where they differ). . . . . | 36 |
| 10 | Causes of the lower full-50 score. . . . .                                                                   | 37 |
| 11 | Project summary metrics. . . . .                                                                             | 38 |
| 12 | Limitations and threats to validity. . . . .                                                                 | 39 |
| 13 | Objectives and research questions: outcomes. . . . .                                                         | 42 |



---

# Acronyms

**AI** Artificial Intelligence

**API** Application Programming Interface

**CBAM** Carbon Border Adjustment Mechanism

**CI** Confidence Interval

**ETS** Emissions Trading System

**EU** European Union

**GPU** Graphics Processing Unit

**HTTP** HyperText Transfer Protocol

**JRC** Joint Research Centre

**JSON** JavaScript Object Notation

**LLM** Large Language Model

**OCR** Optical Character Recognition

**ORM** Object-Relational Mapping

**ReAct** Reasoning and Acting (agent framework)

**RPA** Robotic Process Automation

**RPC** Remote Procedure Call

**SDK** Software Development Kit

**SQL** Structured Query Language

**UI** User Interface

**VPN** Virtual Private Network



---

# General Introduction

**R**EGULATORY compliance in international trade has traditionally relied on manual document review: an analyst reads a supplier's mill certificate or carbon-intensity report, cross-references it against a country's published default emission values, and manually computes a liability. As the European Union's Carbon Border Adjustment Mechanism (CBAM) enters its definitive, annual-declaration phase in 2026, this manual burden falls on every EU importer of steel and aluminium and, symmetrically, creates new reporting pressure on the non-EU exporters whose shipment documents now feed directly into a foreign compliance calculation. Large Language Models (LLMs) offer an obvious lever to automate the reading and reconciliation of such documents, but a compliance number is exactly the kind of output where an LLM's well-documented tendency to hallucinate plausible-looking values [2] cannot be tolerated.

This project resolves that tension architecturally, rather than by simply hoping the model behaves. A **deterministic, LLM-free calculation engine** computes every compliance number. Around it sits a **genuinely agentic tool-use loop**: a locally hosted, open-weight LLM that decides for itself which extraction, lookup, and validation tool to call, in what order, until it is confident enough to write a result or escalates to a human instead. The two halves run on separate machines, communicating over a private network, and are integrated into a working Odoo 17 module — so what comes out the other end is a declaration workflow a real compliance team could open and use, not a research prototype left in isolation.

This report is organised into three chapters. Chapter I establishes the regulatory context, states the problem, and reviews the literature on agentic AI, tool use, and hallucination risk that motivates the system's design. Chapter II presents the system analysis and design: requirements, actors and use cases, the sequence and class diagrams, the design philosophy, and

the system architecture. Chapter III covers the implementation — the calculation engine, the agentic pipeline, and the empirical evaluation results — and closes with a candid discussion of the system’s limitations. A general conclusion summarises the contribution and outlines future work.

---

# Project Context and State of the Art

## Contents

---

|            |                                                                 |           |
|------------|-----------------------------------------------------------------|-----------|
| <b>I.1</b> | <b>Introduction</b>                                             | <b>4</b>  |
| <b>I.2</b> | <b>Host Organisation</b>                                        | <b>4</b>  |
| <b>I.3</b> | <b>State of the Art</b>                                         | <b>5</b>  |
| I.3.1      | EU CBAM as a Moving Regulatory Target                           | 5         |
| I.3.2      | Tool-Using LLMs and the ReAct Paradigm                          | 6         |
| I.3.3      | Hallucination Risk in High-Stakes Numeric Extraction            | 8         |
| <b>I.4</b> | <b>Critique of the State of the Art</b>                         | <b>9</b>  |
| <b>I.5</b> | <b>Problem Statement</b>                                        | <b>10</b> |
| <b>I.6</b> | <b>Existing Methodologies for This Problem and Their Limits</b> | <b>12</b> |
| <b>I.7</b> | <b>Proposed Solution</b>                                        | <b>13</b> |
| <b>I.8</b> | <b>Conclusion</b>                                               | <b>15</b> |

---

## I.1 Introduction

This chapter first presents the host organisation in which this project was carried out, then reviews the state of the art in EU CBAM regulation and in agentic, tool-using language models, critiques that state of the art against the specific demands of high-stakes numeric compliance work, and formulates the resulting problem. It then reviews the existing methodologies used for compliance-document automation and their limits, and closes by presenting this project's proposed solution objectives, research questions, and scope.

## I.2 Host Organisation

This project was carried out at **ninety. Startup House**, an accelerator and incubator based in Sfax, Tunisia (Rte Mahdia KM 1.5, El Bosten).



**Figure 1: ninety. Startup House logo.**

### Mission and Vision

ninety.'s stated mission is to assist innovative SMEs and startups throughout their entrepreneurial journey, by offering a dynamic working environment, personalised support, and the resources needed to turn an idea into a functioning business. Its longer-term vision is to become a primary innovation catalyst for the Sfax region, building an ecosystem in which startups can not only survive but grow, and in doing so inspire further entrepreneurship locally.

### Core Values

The organisation identifies five values as central to how it operates:

- **Innovation** fostering creativity and originality in the startups it hosts.
- **Collaboration** encouraging teamwork between startups, mentors, and investors.
- **Community** building a supportive network oriented toward shared success.
- **Excellence** pursuing continuous improvement and quality.
- **Ethics** emphasising transparency, accountability, and mutual respect.

## Services

Alongside its acceleration and incubation programmes, ninety. offers coworking desks, private offices, dedicated training and study spaces, meeting rooms for presentations and workshops, and business domiciliation (virtual office) services for early-stage companies that do not yet need a physical office of their own.

It is within this startup-support environment surrounding company building and entrepreneurship that the present project's own subject matter — automating a specific, high-stakes regulatory compliance task with agentic AI — was defined and carried out.

## I.3 State of the Art

### I.3.1 EU CBAM as a Moving Regulatory Target

Regulation (EU) 2023/956 establishes the Carbon Border Adjustment Mechanism and its Annex I/Annex II scope [3]; Directive 2003/87/EC, Art. 10a(1a), defines the phase-out schedule for the free-allocation benchmark deduction (2.5% phased out in 2026, rising to 100% by 2034) [4]; Commission Implementing Regulations 2025/2621 and 2026/1412 publish the yearly country/route default emission values, markup schedule, and EU ETS free-allocation benchmarks used in this project's reference data [5; 6]. The Joint Research Centre's JRC134682 report [7] is the underlying scientific basis for these values. CBAM entered its **definitive phase**

**on 1 January 2026:** declarations became **annual** rather than quarterly (the first annual declaration, covering 2026 imports, is due 30 September 2027), and a binding **50-tonne-per-year exemption** applies below which no certificate obligation arises [3]. What distinguishes this regulatory domain from a typical integration target is that the “constants” a naive implementation would reach for are neither constant nor available in one place: default emission values are published per exporting country, carry a yearly markup that itself changes, and are periodically revised by implementing acts [5] — a requirement that shaped the data model (Section III.3) more than any purely technical consideration did. A system that hardcodes these figures in application code is wrong the moment a new implementing regulation is published; the data must be modelled as a **versioned reference table**, not a constant.

### I.3.2 Tool-Using LLMs and the ReAct Paradigm

LLMs capable of multi-step reasoning and external tool use have enabled agents that autonomously decompose complex goals into executable steps [8; 9]. Toolformer showed that a model can learn when and how to call external APIs [10], and Gorilla that API-call accuracy depends heavily on how the tool interface is presented to the model [11]; AgentBench found that open-weight models trail commercial ones most sharply on exactly this kind of multi-turn, instruction-following work [12] — a gap directly relevant to a project restricted to locally hosted models. The **ReAct** framework [1] grounds LLM reasoning in external reality by interleaving reasoning traces with action steps — calls to external tools or APIs. At each step the agent produces a *thought*, selects an *action* (a tool call with structured arguments), and receives an *observation* (the tool’s result). This cycle repeats until sufficient information has been gathered, or the agent escalates. Figure 2 illustrates the loop as implemented in this project’s orchestrator.

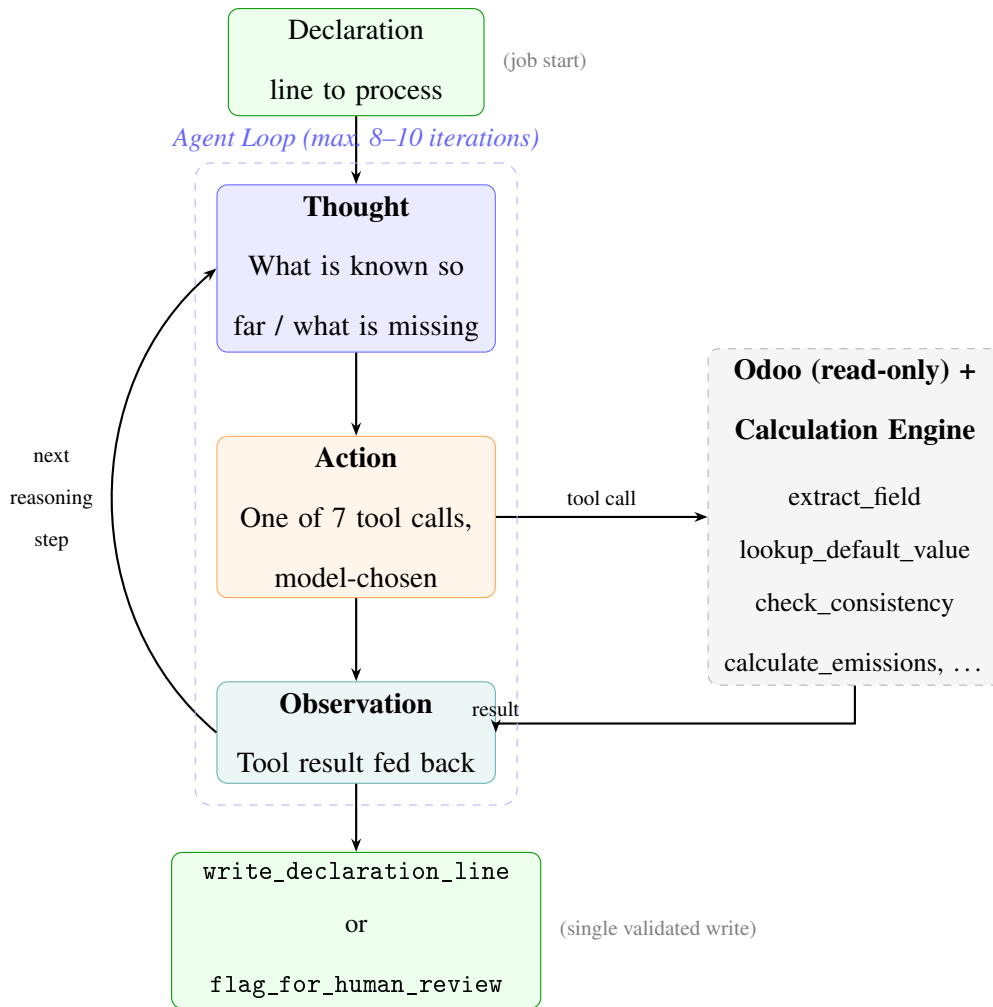


**Figure 2: The ReAct-style agent loop [1] as implemented for CBAM declaration processing.**

The distinguishing property of this execution model is that the loop itself carries no knowledge of which tool “should” come next — that decision is made by the model at every turn, from the current conversation state, not by a step counter or a branch on a previous tool’s name. A fixed pipeline that always calls “extract quantity, then emissions, then calculate” will, on a document where the emissions figure is genuinely absent, either fail outright or silently proceed with a placeholder; an agent that decides its own order can instead request missing data, retry a different extraction strategy, or escalate.

#### a) What “ReAct-style” means precisely here.

This project borrows ReAct’s interleaved reasoning-and-acting *pattern*, not its original *mechanism*. Table 1 sets out the difference.

| Aspect                      | Original ReAct [1]                                      | This project                                                      |
|-----------------------------|---------------------------------------------------------|-------------------------------------------------------------------|
| Action mechanism            | Harness parses literal<br>Thought:/Action: text markers | Native function calling: the model returns a structured tool call |
| Reasoning step              | Part of the parsed output format                        | Prompted, never enforced or parsed                                |
| Turn skipping reasoning     | Malformed output                                        | Accepted unchanged                                                |
| Role of the reasoning trace | Drives control flow                                     | Counted for observability only                                    |

**Table 1: Original ReAct versus this project’s ReAct-style loop.**

Not enforcing the reasoning step is deliberate: checking a turn’s *content* is one step away from the code deciding what a turn should contain, which would reintroduce the orchestrator-side ordering logic this design rules out. Instead, the loop counts every tool-calling turn and, separately, those whose content stated reasoning first, and reports both through the job status endpoint. That lets the report say, from evidence rather than assumption, how often the model actually behaves in the ReAct pattern.

### I.3.3 Hallucination Risk in High-Stakes Numeric Extraction

Hallucination surveys [2; 13] document the general tendency of generative language models to produce fluent, plausible content unsupported by their input — a risk that becomes a compliance-integrity problem the moment the fabricated content is a number that determines a certificate liability. Mallen et al. [14] additionally show that LLM recall degrades specifically for rare, sparsely represented facts relative to popular ones — directly analogous, in this project’s domain, to a regulatory default value for an infrequently-traded country/route combination being exactly the kind of fact a general-purpose model is least reliable at recalling from parametric memory alone. This motivates routing every default value through a deterministic database lookup rather than ever trusting the model to recall or compute one itself.

## I.4 Critique of the State of the Art

Taken individually, none of the three bodies of work reviewed above was designed for the exact situation this project sits in, and each leaves a gap that matters once the output is a compliance number rather than a chatbot answer.

**The ReAct paradigm assumes, but cannot verify, that reasoning actually happens.** Its central claim is that interleaving explicit reasoning with action improves grounding and reduces error-propagation, but that claim rests entirely on the model choosing to produce the reasoning in the first place. Nothing in the paradigm itself, and nothing in the native function-calling interfaces that have since replaced its original text-parsing mechanism, gives an orchestrator a reliable way to confirm that a “thought” actually preceded an “action” rather than being an after-the-fact rationalisation. The only two ways to check would be to constrain the model’s output format, which defeats the point of letting it call tools freely, or to simply trust it. In short, the paradigm is a prompting discipline, not a verifiable guarantee.

**General hallucination-mitigation techniques target the wrong granularity for this problem.** Retrieval augmentation [15], self-consistency voting across multiple samples [16], and learned fact-checking classifiers — the mitigations most commonly proposed against hallucination [13] — are all built for open-domain factual question answering, where the model has to recall or retrieve a fact from a large, unbounded knowledge base. This project’s extraction problem is narrower and more tractable than that: the value to be grounded is already sitting, in full, inside one specific document the model has already been shown in its own context window. The general literature does not distinguish this “verify against a document already in context” case from the harder open-domain case it was designed for, which leaves the specific, cheaper check this project actually relies on (Chapter III) unaddressed by name.

**The rare-fact-recall finding motivates a fix the literature does not itself supply.** Mallen et al.’s own proposed remedy is adaptive retrieval, which assumes an underlying corpus large and general enough to retrieve from. CBAM’s regulatory default values do not fit that setting: they form a small, precisely-scoped, versioned table with a handful of dimensions (sector,

route, country, validity period), not an open corpus. A lookup this constrained does not need retrieval-augmented generation at all — a deterministic query answers it exactly, every time, with no retrieval-relevance uncertainty — but the literature reviewed above never separates this bounded-lookup case from the general recall problem it studies.

None of these three gaps rules out using a tool-using LLM for this task; each just means the general-purpose techniques in the literature need adapting, or replacing with a narrower, purpose-built mechanism, rather than applied as published. What remains is to state precisely where an agentic, tool-using LLM can be trusted, and where it cannot, for a task whose output is a real financial liability.

## I.5 Problem Statement

This regulatory design has an asymmetric commercial consequence for exporters outside the EU. Tunisia exports steel, aluminium, and phosphate products to the European Union, and is directly affected by CBAM reporting requirements that originate on the EU-importer side: a Tunisian mill's own paperwork becomes an input to a foreign compliance pipeline, without any change on the exporter's part. That is why the extraction layer is built against realistic, heterogeneous document formats rather than a single clean template — the documents an importer actually receives from a diverse supplier base are never uniform.

### The Compliance-Automation Dilemma

A CBAM declaration line requires combining several pieces of information that are rarely all present, in a consistent format, in a single document: a quantity, a production route, an exporting country, and either a measured emissions figure or a default value looked up from the versioned reference table described above. Reading and reconciling this information from real supplier paperwork is exactly the kind of unstructured-text task large language models are good at. The problem is what happens next — the number that comes out the other end determines a *real certificate liability*. An LLM that hallucinates a plausible-looking emissions figure, or

silently drops a required field under multi-argument tool-call construction (both observed in this project's own evaluation, Section III.5), produces a wrong compliance number that looks exactly like a correct one. Any system that lets a language model touch the arithmetic directly inherits that risk unconditionally.

## The Workflow-vs-Agent Gap

Most existing document-automation tooling — scripted extraction pipelines with an LLM call bolted onto one step — encodes a predetermined execution path: extract field A, then field B, then compute. This is brittle against the real variability of supplier documents, and it conflates two concerns that should really be kept separate: deciding what to do next given what has been read so far (a reasoning problem, suited to an LLM), and computing a number from validated inputs (a deterministic problem that must never depend on a model's judgment). Left unresolved, this conflation exposes a business to three compounding risks:

- **Silent numeric hallucination:** a fabricated emissions value is indistinguishable, in the output, from a genuinely extracted one, unless the system is explicitly designed to demand and verify provenance.
- **Brittleness to document variability:** a fixed step sequence has no recovery strategy when an expected field is genuinely absent from a given supplier's document, beyond failing the whole run.
- **Silent drift against regulatory data:** default emission values and benchmarks that are hardcoded, rather than versioned and cited, become quietly wrong the moment an implementing act revises them.

A system that wants to be robust to document variability, without also inheriting an LLM's arithmetic unreliability, has to separate these concerns architecturally rather than merely prompt around the risk. That is the precise problem this project is designed to solve.

## I.6 Existing Methodologies for This Problem and Their Limits

Three families of existing approaches apply to this problem.

**Robotic process automation (RPA).** RPA tools operate on the user interface of other systems the way a human operator would, replaying a recorded procedure [17]. They are fast and predictable on the exact layout they were built against, but the RPA literature itself lists brittleness to process and interface change among the field’s open challenges [18]. Applied to supplier documents, a supplier reformatting a mill certificate can silently break every downstream field mapping, and the robot cannot tell “this field is absent” from “this field moved.”

**Learned document-understanding models.** Layout-aware models such as LayoutLM [19] and OCR-free models such as Donut [20] extract fields from forms and receipts far more robustly than templates. They are, however, trained extractors: they require labelled training data for each document type, produce a single pass of predictions with no ability to ask for missing information, and have no notion of a downstream calculation they must stay separate from. No labelled CBAM corpus exists to train them on.

**Generative LLM extraction.** Generative LLMs can extract structured fields from free text with little or no task-specific training, and this is now an active research area in its own right [21]. Most such pipelines, however, apply one extraction call per document, with no iterative reasoning and no deterministic boundary between reading a value and computing with it; when a downstream figure is needed, it is often computed inside the same model call that did the reading. Tool-use benchmarks also show that open-weight models are markedly less reliable at multi-turn, instruction-following agent work than commercial ones [12], so published results obtained with large hosted models do not transfer directly to the small, locally hosted models a data-sovereign deployment would use. Finally, documents fed to an LLM are untrusted input: indirect prompt injection through processed content is a demonstrated attack on LLM-integrated applications [22], which matters when the output is a regulatory figure.

| Aspect                | Existing Approaches                                      | This Project                                                                                 |
|-----------------------|----------------------------------------------------------|----------------------------------------------------------------------------------------------|
| Execution model       | Fixed pipeline or single-shot LLM call                   | Autonomous ReAct-style loop; model decides tool order/count at runtime                       |
| Compliance arithmetic | Often computed inline by the LLM or rigid template logic | Deterministic Python, zero LLM involvement, independently unit-tested                        |
| Regulatory data       | Frequently hardcoded or embedded in prompts              | Versioned reference table, never hardcoded, every value cited to a primary source            |
| Hallucination defense | Rarely addressed explicitly                              | Multi-layer: plausibility bounds, untrusted-content framing, verbatim source-quote grounding |
| Model selection       | Assumed from vendor/model-card claims                    | Empirically benchmarked, with a documented self-correction of an initial, biased ranking     |

**Table 2: Existing compliance-automation approaches vs. this project’s design.**

Table 2 positions this project’s own choices against these limits directly. No widely available compliance-document tool separates extraction (agentic) from calculation (deterministic) as a first-class constraint; few publish a harness-matched benchmark of which locally-hostable model is reliable enough before committing to one; and CBAM’s versioned, country/route-dependent default-value regime combined with an agentic extraction loop has, to the author’s knowledge, no existing open implementation to compare against. This is the space the proposed solution below occupies.

## I.7 Proposed Solution

This project resolves the gaps identified above architecturally, rather than by prompting around them. A deterministic, LLM-free calculation engine is kept categorically separate from a genuinely agentic tool-use loop for document extraction, and candidate models are benchmarked empirically against the project’s own production validation logic — rather than trusted on the strength of a published card, or a generic tool-calling benchmark whose scoring rule may not match how the real system actually consumes a model’s output. Concretely, the project aims to design and implement a genuinely agentic tool-use system for CBAM declaration automation,

paired with a deterministic compliance-calculation core. Three objectives guide its development:

- **O1.** Implement a deterministic, unit-tested calculation engine for CBAM embedded emissions and certificate liability, with zero involvement from any language model in producing the compliance number itself.
- **O2.** Design and implement a genuine agentic tool-use loop — the model decides its own tool-call order and count, with no hardcoded step sequence in the orchestrating code — for supplier document extraction and reconciliation.
- **O3.** Empirically benchmark candidate locally-hosted language models for tool-calling reliability and end-to-end field-extraction accuracy, rather than assuming capability from a model’s published card.

Two research questions guide the investigation:

- **RQ1.** How reliably can a small, locally-hosted language model execute a self-directed, multi-step tool-use loop for structured regulatory data extraction, and does a raw tool-calling benchmark actually predict that reliability?
- **RQ2.** What architectural and prompting safeguards are needed to prevent a tool-using language model from corrupting a compliance-critical numeric output, given that document text fed to the model is untrusted external input?

The system covers **steel and aluminium imports only** (Annex II’s direct-emissions-only sectors), uses **synthetic supplier documents** (no real company data was available), and targets a **two-machine deployment** meshed over a private VPN. Real-time monitoring, sectors beyond steel/aluminium, and a production-grade multi-tenant deployment fall outside the scope of this project. As Table 2 above summarises, each of these objectives answers a specific limitation of the existing approaches already reviewed, rather than a generic ambition to “add AI” to compliance work.

## **I.8 Conclusion**

This chapter has situated the project at ninety. Startup House, reviewed CBAM’s shift toward annual, versioned-data declarations together with the agentic-AI and hallucination literature it depends on, and shown where that literature falls short of the specific demands of a high-stakes numeric compliance task. It then formulated the resulting compliance-automation dilemma, reviewed the RPA-based and single-shot LLM approaches currently used against it and their shared limits, and presented this project’s own objectives, research questions, and scope as a direct response to those limits. These constraints — a deterministic core, a genuinely autonomous extraction loop, and an empirical model choice — directly shape the design presented in the following chapter.

---

## System Analysis and Design

### Contents

---

|                                                      |           |
|------------------------------------------------------|-----------|
| <b>II.1 Introduction . . . . .</b>                   | <b>17</b> |
| <b>II.2 Requirements Specification . . . . .</b>     | <b>17</b> |
| II.2.1 Functional Requirements . . . . .             | 17        |
| II.2.2 Non-Functional Requirements . . . . .         | 18        |
| <b>II.3 Actors and Use Cases . . . . .</b>           | <b>20</b> |
| <b>II.4 Dynamic View: Sequence Diagram . . . . .</b> | <b>22</b> |
| <b>II.5 Static View: Data Model . . . . .</b>        | <b>24</b> |
| <b>II.6 Design Philosophy . . . . .</b>              | <b>26</b> |
| <b>II.7 System Architecture . . . . .</b>            | <b>27</b> |
| <b>II.8 Conclusion . . . . .</b>                     | <b>28</b> |

---

## II.1 Introduction

This chapter turns the problem and objectives stated in Chapter I into a concrete design: it specifies the functional and non-functional requirements the system must satisfy, identifies its actors and the use cases they exercise, presents the dynamic view of the system as a sequence diagram of its end-to-end flow, presents the static view as a class diagram of its Odoo data model, and closes with the design philosophy and system architecture that these requirements led to. Chapter III then covers how this design was actually implemented and evaluated.

## II.2 Requirements Specification

### II.2.1 Functional Requirements

| ID         | Requirement                                                                                                                                                                                                                        |
|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>FR1</i> | Read a named field's value from an uploaded supplier document, using a deterministic regex pre-parser for known templated fields, with a fallback to LLM-grounded reading of the raw document text when no pattern matches.        |
| <i>FR2</i> | Look up a versioned default emission factor for a given sector, production route, and country as of the declaration date, rejecting an ambiguous match (zero or more than one candidate record) rather than resolving it silently. |
| <i>FR3</i> | Deterministically compute embedded emissions and certificate liability for a declaration line from validated inputs, with zero language-model involvement in the arithmetic itself.                                                |

| ID         | Requirement                                                                                                                                                                                                                  |
|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>FR4</i> | Run an advisory self-consistency check on extracted data before the final write, including grounding any actual-sourced emissions value against document text genuinely extracted earlier in the same job.                   |
| <i>FR5</i> | Escalate a declaration line to human review, either explicitly (model-initiated) or automatically once the iteration cap is reached without a completed write.                                                               |
| <i>FR6</i> | Persist at most one validated write per declaration line per job, together with the full tool-call trace as an audit record.                                                                                                 |
| <i>FR7</i> | Distinguish, on the declaration line, a genuinely exempt line from one whose certificate figure is zero only because reference data has not yet been seeded two states that would otherwise be visually identical to a user. |
| <i>FR8</i> | Scope declarant users to their own declarations, declaration lines, and source documents; grant managers unrestricted access across all declarants.                                                                          |

Table 3: Functional requirements.

## II.2.2 Non-Functional Requirements

| ID          | Requirement                                                                                                                                                  |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>NFR1</i> | Compliance arithmetic must never depend on a language model's output; the agent loop may only decide <i>when</i> to trigger a calculation, never perform it. |

| ID          | Requirement                                                                                                                                                                                                                               |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>NFR2</i> | Every agent job is bounded to at most 8–10 tool-calling iterations, with an automatic escalation fallback if the cap is reached without a terminal state.                                                                                 |
| <i>NFR3</i> | Numeric values submitted for a declaration write are bounded by hard plausibility limits (quantity, emission-factor magnitude), enforced independently of the advisory consistency check.                                                 |
| <i>NFR4</i> | A measured (“actual”) emissions figure may only be written if accompanied by a verbatim quote that is a genuine substring of document text the system itself has already extracted during that job not merely a plausible-looking string. |
| <i>NFR5</i> | Regulatory reference data (emission factors, benchmarks, year parameters) is held in versioned, citation-bearing database tables; no compliance constant may be hardcoded in application code.                                            |
| <i>NFR6</i> | Uploaded document attachments are size-capped, with the limit enforced independently at both upload time and read time.                                                                                                                   |
| <i>NFR7</i> | The orchestrator’s job API fails closed: an unconfigured API key rejects every request rather than disabling authentication, and key comparison is constant-time.                                                                         |
| <i>NFR8</i> | The deployment is data-sovereign: the only external network dependency is the private VPN mesh between the two project-owned machines; neither host makes a third-party API call.                                                         |

**Table 4: Non-functional requirements.**

## II.3 Actors and Use Cases

The system's own access-control configuration distinguishes exactly two human-facing actors. The **Declarant** is row-scoped to their own declarations, declaration lines, and source documents, while the **Manager** has unrestricted access across every declarant — a generalisation relationship, since a Manager can do everything a Declarant can, plus manage shared reference data.

One further point is worth making precise here, since it shapes how the loop itself should be read as a security diagram: the **Agent Loop (LLM)** is modelled as a *secondary, system* actor, not an independently privileged one. The orchestrator authenticates to Odoo as a single privileged service account for every job, so the agent loop is never itself subject to Odoo's row-level security. It is invoked by a Declarant's own trigger action, and is instead scoped to the correct declaration and declaration line entirely in application code (Section I.6 already explained why no LLM-facing argument is trusted for this scoping).

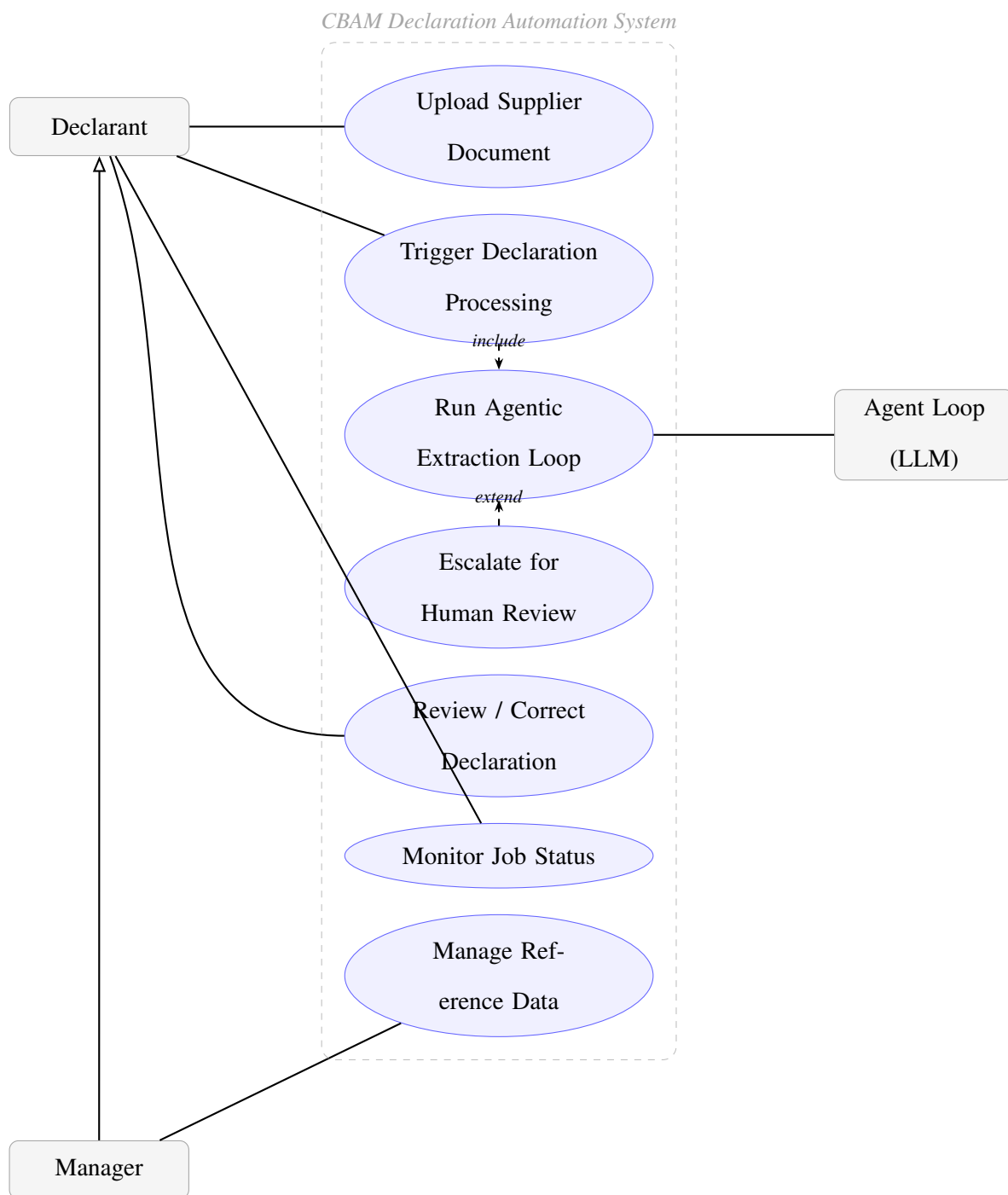


**Figure 3: Use case diagram of the CBAM Declaration Automation System.**

Table 5 summarises each use case's actor(s) and intent.

| Use Case                       | Actor(s)           | Description                                                                                                           |
|--------------------------------|--------------------|-----------------------------------------------------------------------------------------------------------------------|
| Upload Supplier Document       | Declarant          | Attach a document to a declaration line as a <code>cbam.source.document</code> record.                                |
| Trigger Declaration Processing | Declarant          | Start an orchestrator job for one declaration line; includes <i>Run Agentic Extraction Loop</i> .                     |
| Run Agentic Extraction Loop    | Agent Loop         | The ReAct-style loop from Chapter I, executing the seven tools described in Chapter III until it writes or escalates. |
| Escalate for Human Review      | Agent Loop         | Extends the loop: triggered explicitly by the model, or automatically once the iteration cap is reached.              |
| Review / Correct Declaration   | Declarant, Manager | Inspect a line's extracted values, certificate status, and audit trail after a job completes.                         |
| Monitor Job Status             | Declarant, Manager | Poll a job's status, iteration count, and tool-call/reasoning-turn counters.                                          |
| Manage Reference Data          | Manager            | Maintain emission factors, benchmarks, year parameters, goods, installations, and production routes.                  |

**Table 5: Use case descriptions.**

## II.4 Dynamic View: Sequence Diagram

Figure 4 traces *Trigger Declaration Processing* end to end, across the four participants introduced above: the Declarant, Odoo (the Softworker host), the Orchestrator (the FastAPI agent-loop service, also on the Softworker host), and the LLM (served by Ollama on the Hardworker host). Only the Orchestrator–LLM exchange crosses the Tailscale VPN between the two ma-

chines; every Orchestrator–Odoo exchange is an RPC call over the Softworker’s own internal Docker network.

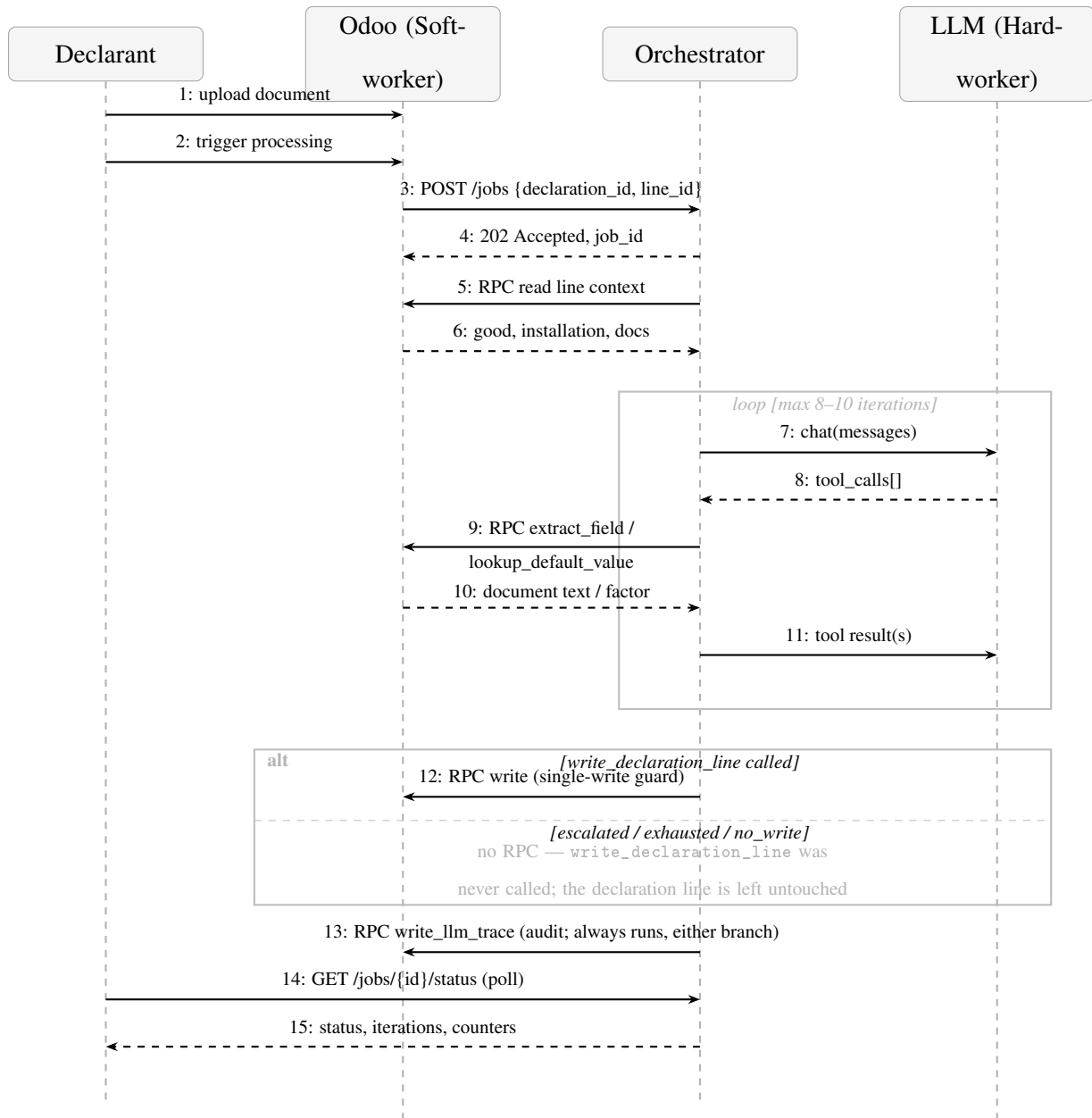


**Figure 4: Sequence diagram for *Trigger Declaration Processing*.**

Solid arrows above are calls and dashed arrows are returns. Steps 7–11 repeat inside the bounded loop; step 9 is a stand-in for whichever read-only tool the model calls (`calculate_emissions` and `check_consistency` are pure computation and never leave the Orchestrator). The diagram deliberately does not show only the happy path: step

12 runs, at most once per job, only in the branch where the model actually called `write_declaration_line` before the loop ended. In every other outcome — an explicit escalation, the iteration cap being exhausted, or the loop ending without a write — no declaration-line RPC happens at all, and the job produces nothing for a human to review beyond the audit trace. Step 13, by contrast, is unconditional: `write_llm_trace` runs on every job regardless of which branch was taken, recording `extraction_status` as `extracted` only in the first case and `needs_review` in every other one (`softworker/orchestrator/main.py`'s `_EXTRACTION_STATUS_BY_LOOP_STATUS` mapping).

## II.5 Static View: Data Model

The domain is modelled as nine Odoo models, all in the custom `cbam` addon. Figure 5 shows the seven models most central to a declaration line's own data; `cbam.good.benchmark` and `cbam.year.parameter` are described in the accompanying text rather than drawn, to keep the figure legible.

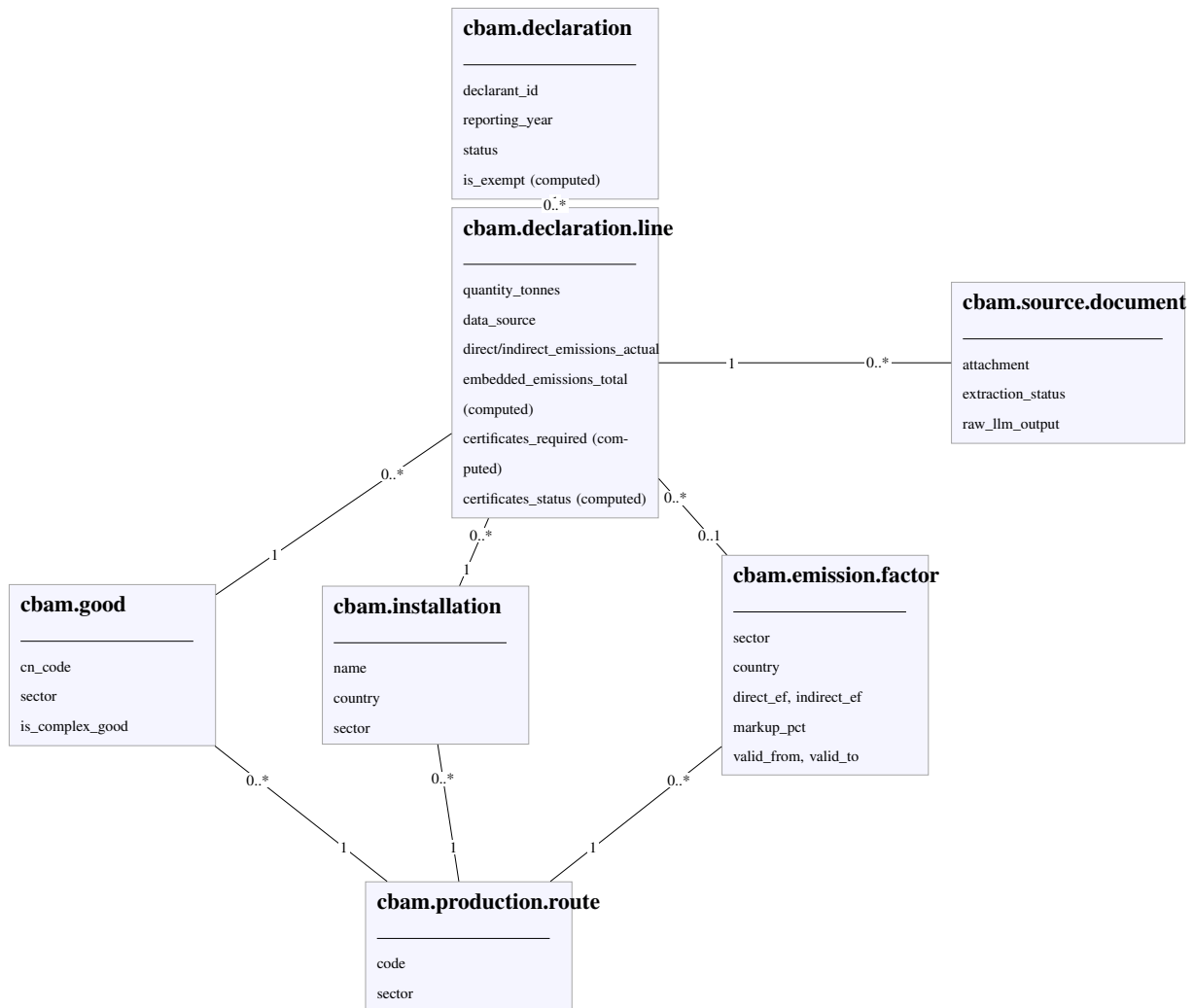


**Figure 5: Class diagram of the seven core cbam models.**

Beyond the associations shown, two reference tables complete the model: `cbam.good.benchmark` (a per-good, per-route free-allocation benchmark, unique per `good_id/production_route_id` pair, itself referencing `cbam.production.route`) and `cbam.year.parameter` (the yearly phase-out factor and reference certificate price, unique per year); `cbam.declaration.line` holds a computed reference to each. Several constraints keep this model internally consistent rather than merely populated: a Python constraint on `cbam.declaration.line` rejects a good/installation/route combination whose sectors disagree; a Python constraint on `cbam.emission.factor` rejects a validity range that overlaps an existing one for the same sector/route/country, so a lookup can never silently match more than one record; SQL unique constraints enforce one `(cn_code, sector)` per good, one

(code, sector) per production route, and one row per year in `cbam.year.parameter`; and a size constraint on `cbam.source.document.attachment` caps every uploaded document, enforced again independently at read time when the orchestrator fetches it.

## II.6 Design Philosophy

Three principles governed every architectural decision in this project.

### a) **Determinism over convenience:**

It would be simpler to let the language model read a document and directly state a certificate liability. This was ruled out categorically: the compliance number is computed by a standalone, Odoo-independent Python module, unit-tested against hand-calculated examples before any LLM code existed, and the agent never has write access to that computation — it only decides *when* to call it.

### b) **Autonomy over prescription:**

The extraction layer makes its own decisions about which tool to call, in what order, and how many times, without a human- or code-authored step sequence. This ruled out a fixed extraction pipeline in favour of a loop that reasons about the current conversation state and selects the next action itself, every turn.

### c) **Verifiability over assumed capability:**

No model was chosen because of published benchmark scores or model-card claims. Every candidate was run against the project's own production validation logic, and — as Section III.5 shows — an initial ranking obtained this way was itself later found to be a measurement-methodology artifact and corrected, rather than left unexamined once a plausible answer had been found.

## II.7 System Architecture

The system is deployed across two roles, meshed over a private VPN, as shown in Figure 6.

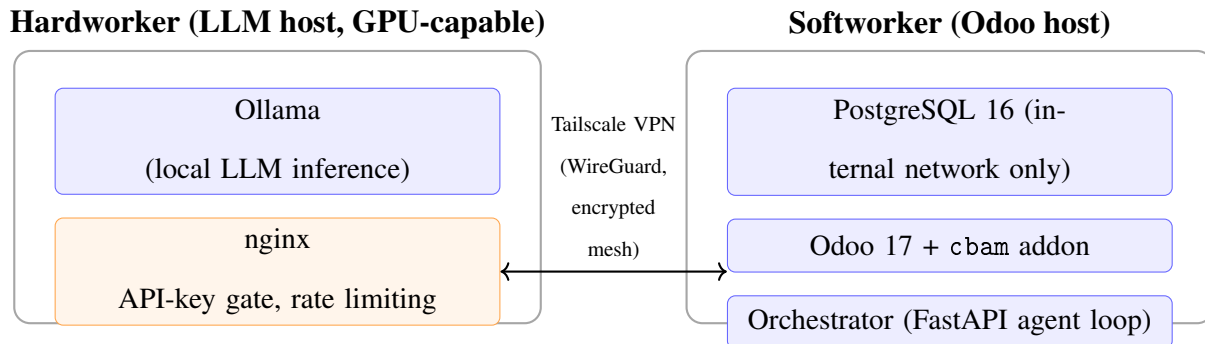


**Figure 6: System architecture: Hardworker (LLM host) and Softworker (Odoo host), meshed over a private VPN.**

Only the Orchestrator–Ollama exchange crosses the VPN shown above. The originally specified per-role Tailscale sidecar identity had to be replaced, on **both** hosts, by each host’s own already-authenticated VPN connection, after a reproducible key-generation fault turned up in the VPN provider’s own console. The fix was applied first to the Softworker host during live deployment, then pre-emptively mirrored onto the Hardworker host once the same root cause was recognised — a documented deviation, not one silently absorbed into the design.

A user uploads a supplier document in the Odoo UI and triggers processing; Odoo calls the orchestrator’s job endpoint and polls its status while the agent loop runs against the LLM host over the VPN; on completion the orchestrator performs a single validated write back to Odoo, and the full tool-call trace is persisted as an audit trail (Figure 4 above gives the precise sequence).

The stack uses Odoo 17 Community, PostgreSQL 16, Ollama for local inference, FastAPI for the orchestrator, Pydantic for argument validation, and the plain `openai` client SDK pointed at Ollama’s OpenAI-compatible endpoint. Deliberately absent is **any agent framework** (LangChain, AutoGen, CrewAI): for a fixed, seven-tool schema, a framework would only

add indirection without benefit, and would make the deterministic-calculation boundary harder to audit.

## **II.8 Conclusion**

This chapter has turned Chapter I's problem statement into a concrete design: eight functional and eight non-functional requirements, two human actors plus the agent loop modelled correctly as a secondary system actor, a traced end-to-end sequence, and a nine-model data model built around three deliberate design principles and a two-machine architecture. Chapter III now covers how this design was implemented — the calculation engine, the agentic pipeline built on top of it, and the empirical evaluation of both.



---

## Implementation and Results

### Contents

---

|                                                      |    |
|------------------------------------------------------|----|
| III.1 Introduction . . . . .                         | 30 |
| III.2 Model Selection Methodology . . . . .          | 30 |
| III.3 The Deterministic Calculation Engine . . . . . | 31 |
| III.4 The Agentic Pipeline . . . . .                 | 32 |
| III.5 Results . . . . .                              | 34 |
| III.6 Limitations and Threats to Validity . . . . .  | 38 |
| III.7 Conclusion . . . . .                           | 39 |

---

## III.1 Introduction

This chapter covers how the design presented in Chapter II was actually built and evaluated: the methodology used to select a candidate language model, the deterministic calculation engine, the agentic pipeline built around it, the empirical evaluation results, and a candid discussion of the system’s limitations.

## III.2 Model Selection Methodology

Four locally hosted models were candidates for the default: llama3.1:8b, qwen2.5:7b-instruct, qwen3:8b and qwen2.5:14b-instruct. Selection followed the three stages in Figure 7, treating each benchmark result as a hypothesis to check rather than a verdict.

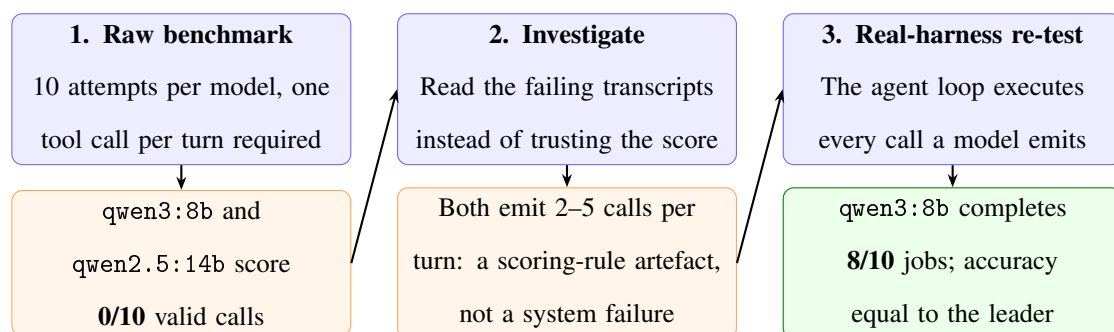


**Figure 7: The three-stage model-selection process.**

The raw benchmark (Table 7) scored two models at 0% valid tool calls. Their transcripts showed why: both routinely emitted several tool calls in one turn, which the benchmark’s scoring rule counted as a failure. The real agent loop has no such rule; it executes every call a model emits, in the order the model chose. Re-run on the real harness with the same ten records, qwen3:8b completed 8/10 jobs against the benchmark’s prediction of none, with field accuracy equal to the leading model’s (Table 8). The benchmark had measured a real habit but drawn the wrong conclusion from it, because its scoring rule did not match how the system consumes a model’s output. From then on, every raw benchmark in this project was treated as a hypothesis to re-test on the real harness. The same reasoning later led to testing the larger

qwen2.5:14b-instruct, which the benchmark had dismissed for the same reason and which became the default model (Section III.5).

### III.3 The Deterministic Calculation Engine

The calculation engine is an Odoo-independent module made up of four pure functions, and the language model never calls any of them with an unvalidated argument: an exemption check (the 50 t/year threshold), an embedded-emissions calculation, a certificate-liability calculation implementing Regulation (EU) 2023/956 Art. 9/31 [3], and a versioned emission-factor lookup that raises an error on zero or more than one match, rather than resolving an ambiguity silently.

Indirect emissions are deliberately excluded from the embedded-emissions sum, and not merely expected to come out as zero: steel and aluminium are Annex II, direct-emissions-only sectors, so a non-zero indirect figure recorded for statistical purposes can never leak into the compliance number. The certificate calculation also floors every deduction and adjustment at zero, so no combination of inputs — including a negative or erroneous carbon price — can invert the formula into a surcharge. An earlier version of this formula actually had a sign-inversion bug (the phase-out factor was applied directly rather than as its complement, which would have understated every early-year certificate requirement); it was found and fixed with a dedicated regression test, documented here as a caught defect rather than left out of the record. Twenty unit tests cover the engine in total, all written and confirmed failing before the corresponding implementation existed.

The Odoo data model (Section II.5) wraps this engine with stored, dependency-tracked computed fields and a status field distinguishing a genuinely exempt line from one that is zero only because reference data has not yet been seeded — two states that would otherwise be visually identical to a user. Row-level security scopes each declarant to their own declarations.

### III.4 The Agentic Pipeline

The model is exposed to seven fixed tools, defined once and shared, unmodified, between the model-facing schema and the orchestrator's own argument validation (Table 6).

| Tool                                | Kind                        | Purpose                                                                                                                                          |
|-------------------------------------|-----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>extract_field</code>          | Hybrid<br>regex/LLM<br>read | Deterministic regex pre-parse for known templated fields; falls back to the model reading a supplier document's raw text when no pattern matches |
| <code>lookup_default_value</code>   | Deterministic<br>read       | Looks up a versioned default emission factor                                                                                                     |
| <code>check_consistency</code>      | Deterministic<br>validation | Flags implausible, missing, or ungrounded extracted values                                                                                       |
| <code>request_missing_data</code>   | Soft escalation             | Flags a gap back for human/Odoo follow-up                                                                                                        |
| <code>flag_for_human_review</code>  | Hard<br>escalation          | Terminates the loop, requests human review                                                                                                       |
| <code>calculate_emissions</code>    | Deterministic<br>calc.      | The model decides <i>when</i> to call it, never computes the number itself                                                                       |
| <code>write_declaration_line</code> | Single side<br>effect       | The only tool with write access; enforced to run at most once per job                                                                            |

**Table 6: The seven agent tools.**

`extract_field` is worth a closer look, since its mechanism changed materially over the course of the project, from a pure LLM read to a **hybrid** one. Because the synthetic supplier documents used for evaluation (Section III.5) are generated from a fixed set of templates, a deterministic regex pre-parser now attempts to read a templated numeric field — and its own verbatim `source_quote` — directly from the document text, before the model ever sees it, falling back to LLM-grounded reading only when no known pattern matches. This removes the model's own reading reliability as a source of error for exactly the fields where a fixed template makes deterministic parsing possible, while leaving genuinely free-text fields to the model.

At each turn, the model receives the full conversation history and either calls one or more tools or produces a final answer. The orchestrating code's only responsibility is to execute whatever the model named and feed the result back, subject to a hard iteration cap (8–10 loops) that auto-escalates if the model has not converged by then. That is a documented, expected

failure mode for small local models, something the design accounts for rather than treats as an edge case.

Executing whatever the model requests is not the same as trusting it blindly, and three safeguards answer RQ2 directly.

First, facts the orchestrator already knows with certainty — the good, installation, quantity, and data source already set on the line — are injected directly into tool calls, overriding whatever the model supplied. This responds to a measured failure mode where small local models reliably drop a required field under multi-argument JSON construction.

Second, the dispatcher remembers a model’s own prior tool results within one job, so a correctly extracted figure is not silently lost if the model fails to restate it several turns later. This is a failure mode that was observed live during development (Section III.6).

Third, and most directly targeting the weakest measured fields (Section III.5), the final write is rejected for measured (“actual”) emissions data unless the model supplies an exact, verbatim quote — a real substring of text an extraction tool actually returned during that job, not merely a plausible-looking string. Hard numeric plausibility bounds and a single-write guard sit underneath this as a structural backstop, regardless of whether the model itself is fooled or a document attempts prompt injection. `check_consistency` runs a related, advisory version of this same grounding check earlier in the loop, on the raw numeric value rather than the quote, so a fabrication can be caught — and corrected by the model itself — an iteration before the harder, final gate would otherwise reject it.

Beyond these three safeguards, the loop is also instrumented for **observability**, which is kept deliberately separate from the safeguards above. Every real model turn that makes a tool call is counted, and the subset of those turns whose message content was non-empty — where the model stated reasoning before acting, per the ReAct-style prompting discussed in Section I.3.2 — is counted separately and surfaced through the job status endpoint. This measures how often the model’s behaviour actually matches the ReAct pattern it is asked to follow, without rejecting or altering any turn that does not. It is a metric, not a fourth safeguard, and it answers no part of RQ2 on its own.

## III.5 Results

This section reports every evaluation stage in the order it was run, each one acting on evidence the previous stage produced. Every number is given at the sample size it was measured on, and regressions are reported next to the fixes that followed them.

### a) How to read the numbers.

The evaluation set contains 50 synthetic supplier documents (37 with measured, “actual” emissions and 13 using default values), and several stages use only its first 10 or 20 records. With samples this small, a single record moves a percentage by 2 to 10 points, so each headline proportion is given with a 95% Wilson score interval [23]. Where two intervals overlap, the difference is reported as *observed on this sample*, not as established.

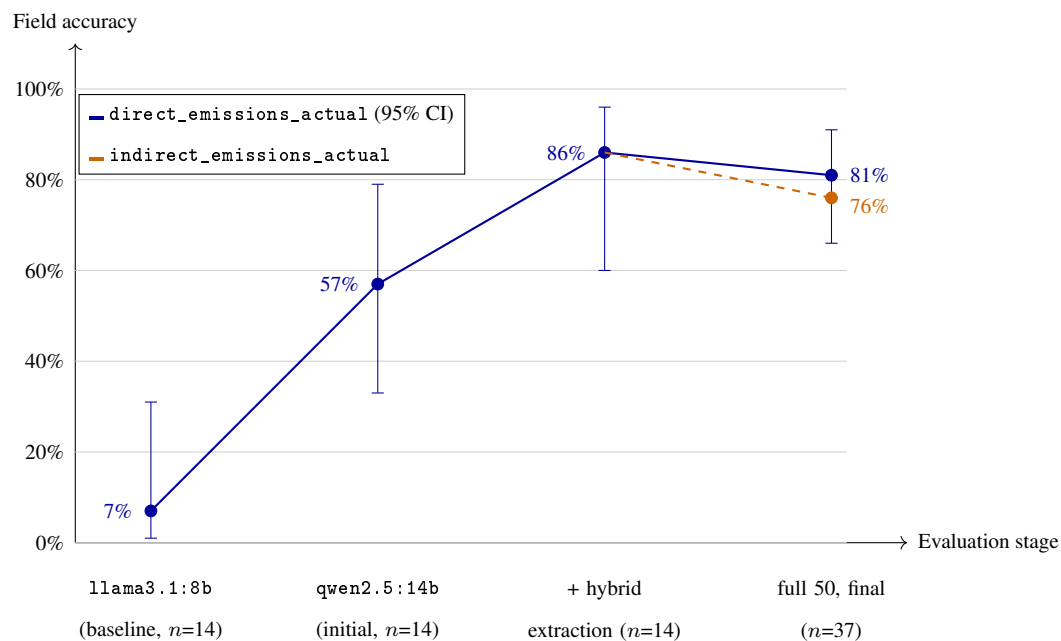


**Figure 8: Accuracy on the two hardest fields across evaluation stages.**

Figure 8 summarises the trajectory on the two fields that require genuine document comprehension. The two fields gave identical results until the final full-50 run. The baseline and final intervals ([1–31%] and [66–91%]) do not overlap, so the overall improvement is not a small-sample artefact; the step between the last two stages is within noise.

## Raw Benchmark and Model Comparison

The raw benchmark (Table 7) ran ten attempts per model against the production argument-validation logic, scoring one tool call per turn. Its two 0% results were re-tested on the real harness, on the same ten records (Table 8).

| Model                | Valid calls   | Latency | Dominant failure mode          |
|----------------------|---------------|---------|--------------------------------|
| llama3.1:8b          | 5/10 [24–76%] | 1.7 s   | Occasional invalid enum values |
| qwen2.5:7b-instruct  | 4/10 [17–69%] | 2.2 s   | Wrong argument types           |
| qwen3:8b             | 0/10 [0–28%]  | 14.1 s  | Several tool calls per turn    |
| qwen2.5:14b-instruct | 0/10 [0–28%]  | 5.5 s   | Same multi-call pattern        |

**Table 7: Raw tool-calling benchmark (95% CI in brackets).**

| Model               | Completed     | Escalated | Exhausted | Field accuracy |
|---------------------|---------------|-----------|-----------|----------------|
| llama3.1:8b         | 4/10 [17–69%] | 5/10      | 1/10      | 12–30%         |
| qwen2.5:7b-instruct | 0/10          | 0/10      | 10/10     | 0%             |
| qwen3:8b            | 8/10 [49–94%] | 0/10      | 2/10      | 12–30%         |

**Table 8: Real-harness comparison on the same 10 records.**

The benchmark predicted that qwen3:8b could not complete a job (0/10, interval [0–28%]); on the real harness it completed 8/10 ([49–94%]). These intervals do not overlap, so the benchmark’s verdict was wrong for this system, not just unlucky. The comparison between qwen3:8b and llama3.1:8b is weaker: 8/10 against 4/10 completed, with field accuracy identical, but the completion intervals overlap, so ten records cannot separate the two models. The phase also showed that prompt and architecture fixes closed the *completion* gap without moving *accuracy* until the harness’s own bugs were fixed: finishing the loop and extracting the correct value have different causes and different fixes.

## From First Model to Final Configuration

Table 9 consolidates every later stage. “Plumbing fields” are quantity\_tonnes, data\_source and precursor\_emissions, which scored identically at every stage; “comprehension fields”

are `direct_emissions_actual` and `indirect_emissions_actual`, scored only on actual-sourced records.

| Stage                               | <i>n</i> | Done /<br>Esc. / Exh. | Plumbing<br>fields           | Carbon<br>price              | Comprehension                                        |
|-------------------------------------|----------|-----------------------|------------------------------|------------------------------|------------------------------------------------------|
| 1. llama3.1:8b, fixed harness       | 20       | 5 / 13 / 2            | 20%<br>[8–42]                | 15%                          | 7%<br>[1–31]                                         |
| 2. qwen2.5:14b-instruct             | 20       | 14 / 0 / 6            | 70%<br>[48–85]               | 60%<br>[39–78]               | 57%<br>[33–79]                                       |
| 3. + hybrid regex/LLM<br>extraction | 20       | 17 / 3 / 0            | 90%<br>[70–97]               | 70%<br>[48–85]               | 86%<br>[60–96]                                       |
| 4. Full 50 records                  | 50       | 34 / 14 / 2           | 68%<br>[54–79]               | 50%<br>[37–63]               | 73% / 68%                                            |
| 5. + fixture coverage (final)       | 50       | 41 / 7 / 2            | <b>86%</b><br><b>[74–93]</b> | <b>64%</b><br><b>[50–76]</b> | <b>81%</b><br><b>[66–91] / 76%</b><br><b>[60–87]</b> |

**Table 9: Evaluation stages (95% Wilson CI in brackets; comprehension is direct / indirect where they differ).**

**Stage 1** ran after fixing several infrastructure bugs (missing document context, an ORM constraint that made the default-value path uncallable, an unset inference temperature, a value-carryover bug). The comprehension fields barely moved, which motivated the source-quote grounding gate. **Stage 2** switched to a larger model in the same open-weight tier: every field improved, and comprehension rose from 7% to 57%, the first evidence that the small-model ceiling was not architectural. A re-run of qwen3:8b with the full safeguard set (16/20 completed, 25–35% on plumbing fields, 7% on comprehension) confirmed the gap, and also exposed an instrumentation bug: the reasoning-turn counter read zero because the model returned its reasoning in a separate `message.reasoning` field. Once fixed, reasoning appeared on effectively every turn. qwen2.5:14b-instruct became the default model.

**Stage 3** added the regex pre-parser to `extract_field`. Two regressions were caught before any aggregate was trusted: a generic result shape that the model copied into the write call, silently zeroing every value, and a null-coercion fix that zeroed a correctly extracted sibling value. This is the strongest result measured, but it targets the dataset’s own templates; a document in an unfamiliar format falls back to plain LLM reading.

**Stage 4** ran the full 50 records for the first time. Records 20–49, never used during development, scored well below records 0–19 (plumbing 53% against 90%). Table 10 lists the three causes, each traced live on individual records.

| Cause                           | Evidence                                                                                                                                                                        | Status                 |
|---------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|
| Test-fixture gap                | Default-factor fixtures covered 6 of the 13 (sector, route, country) combinations; 7 of 11 escalations in records 20–49 needed a missing one, and the model correctly escalated | Fixed (stage 5)        |
| Single write vs. grounding gate | The gate rejected a fabricated direct value three times; the model escalated, and the correctly extracted indirect value was discarded with it                                  | Design trade-off, open |
| Run-to-run variance             | Two records scored zero in the batch but were fully correct when re-run on the same code (temperature 0.1)                                                                      | Model property         |

**Table 10: Causes of the lower full-50 score.**

**Stage 5** extended the fixtures to all 13 combinations. Escalations in records 20–49 fell from 11 to 4, exactly the 7 records the diagnosis predicted. The plumbing-field gain (68% to 86%) is at the edge of the intervals; the comprehension gains (73% to 81%, 68% to 76%) are within noise and are reported as observed only.

## Self-Consistency (Rejected) and Two Targeted Fixes

Self-consistency, sampling several answers and taking the majority, is a standard way to reduce random reasoning errors [16]. It was tested on 15 records with two independent runs each: 13 were correct both times, 2 were wrong both times with the same failure, and none disagreed. Majority voting only helps when a second draw can avoid an error, and every failure observed here was systematic, so it would have doubled cost for no measured benefit. Zero disagreements in 15 still leaves a true disagreement rate of up to 20% plausible, so the rejection is a judgement for this scale, not a proof. The two systematic failures were instead traced and fixed. First, the model sometimes never asked for `carbon_price_paid_origin`; the dispatcher now parses that field opportunistically whenever `extract_field` runs. Second, on a customs document with no emissions data, the model spent all nine iterations retrying a write; a prompt rule now makes it escalate, which it did in 3 iterations. On records 0–19, carbon price rose from 70% to

80% (14/20 to 16/20, intervals overlapping) and no other field changed. These two fixes have not yet been re-run on all 50 records.

## Summary Metrics

| Metric                                     | Value                                   |
|--------------------------------------------|-----------------------------------------|
| Development window                         | 2026-07-28 → 2026-09-18                 |
| Commits / automated tests                  | 66 / 175 test functions in 14 files     |
| Python lines of code                       | ≈7,600 incl. tests (≈4,400 application) |
| Odoo models / agent tools / candidate LLMs | 9 / 7 / 4                               |
| Default model                              | qwen2.5:14b-instruct                    |
| Plumbing fields, full 50 (final)           | 64–86% (lowest field 64% [50–76])       |
| Comprehension fields, full 50 (final)      | 76–81% ([60–87], [66–91])               |

**Table 11: Project summary metrics.**

## III.6 Limitations and Threats to Validity

Table 12 lists what the results above do and do not support. The first four rows concern how far the numbers generalise; the rest are open engineering items.

| Limitation                      | Consequence                                                                                                                                                | What would close it           |
|---------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------|
| Small sample (10–50 records)    | Intervals are wide; only the baseline-to-final gain and the benchmark reversal are clearly outside noise                                                   | A larger evaluation set       |
| Synthetic data only             | Documents come from templates the regex pre-parser was built for, so accuracy on real supplier paperwork is unmeasured                                     | Even 10–20 real certificates  |
| Narrow model range              | Four open-weight 7–14B models (Llama, Qwen), all served through Ollama; larger or hosted models were not tested                                            | Broader model sweep           |
| No single-shot baseline         | The agentic loop was not compared with one extraction call per document on the same harness, so its benefit is argued, not measured                        | Ablation run on the LLM host  |
| Not production-grade            | Final run: 7/50 escalate, 2/50 exhaust the cap, no field reaches 100%                                                                                      | Further fixes, human review   |
| Single write vs. grounding gate | A rejected fabrication discards a correctly extracted sibling value on the same record                                                                     | Partial writes, design change |
| Two fixes on 20 records only    | Carbon-price recovery and the customs prompt rule are unvalidated at full scale (full-50 figure: 64%)                                                      | Full-50 re-run                |
| Value carry-over                | The model once replaced a looked-up factor with invented numbers; dispatcher memory mitigates the symptom, not the behaviour                               | Model or prompt change        |
| Deployment unverified           | The two-machine topology runs on a VPN workaround and has not been verified end to end; the LLM host has not been run against the physical Odoo deployment | Live two-machine test         |
| Reasoning vs. correctness       | Reasoning-turn counts are now measured but not yet correlated with correct extractions                                                                     | Analysis over a full run      |

**Table 12: Limitations and threats to validity.**

## III.7 Conclusion

This chapter has shown the three design principles of Chapter II realised concretely: an independently tested calculation engine that no language model ever writes to directly, a tool-use loop with no hardcoded ordering logic, and a benchmarking methodology rigorous enough to catch and correct its own initial ranking error — twice over, first correcting a wrongly-dismissed qwen3:8b, then discovering and adopting the stronger qwen2.5:14b-instruct. Applied to synthetic CBAM declarations for steel and aluminium, and measured honestly at full dataset

scale rather than on a flattering subsample, the system raised measured field accuracy from a 0% baseline to 64–86% on plumbing-reachable fields and 76–81% on document-comprehension fields. Those were the fields this project’s evaluation history had treated as a hard model-capability ceiling, until a larger model and a hybrid deterministic/LLM extraction path together showed that ceiling was not architectural after all. What remains open is not whether the architecture works, but how far its two still-unresolved tradeoffs — the single-write/grounding-gate interaction, and two fixes not yet re-validated at full scale — can be closed without reintroducing the risks Chapters I and II were designed to rule out.



---

## General Conclusion

THIS project set out to address a concrete tension in applying large language models to regulatory compliance: the same unstructured-document reasoning that makes an LLM attractive for this task is exactly what makes it untrustworthy once it is allowed anywhere near a number that determines a real financial liability. Rather than avoiding LLMs altogether, this project kept the two concerns architecturally separate: a deterministic, independently verifiable calculation engine on one side, and a genuinely autonomous, ReAct-style tool-use agent on the other, wired together through a two-machine, VPN-meshed deployment and integrated into a working Odoo module.

The evaluation turned up a finding more nuanced than a single accuracy number. The dominant constraint on agentic compliance extraction turned out to be a small local model's own tool-calling discipline, not the orchestration architecture built around it — and, just as importantly, a model's apparent reliability depends heavily on how it is measured, rather than being some fixed property of the model itself. The project's own raw benchmark initially and wrongly dismissed a model that, once re-evaluated against the real execution harness, completed 8 of 10 jobs where the benchmark had predicted none, with field accuracy equal to the presumed leader's. **This methodology-sensitivity of model ranking is one of this work's central findings:** a benchmark that does not match the real system's execution semantics can actively mislead an architectural decision, and only empirical re-verification against the production path — not a one-off benchmark score — earns the right to trust a model choice. Pushing that same discipline further, at full and honest dataset scale rather than on a flattering subsample, the project went on to find a larger candidate model and a hybrid deterministic/regex-plus-LLM extraction path. Together they raised measured field accuracy from a 0% baseline to 64–86% on plumbing-reachable fields and 76–81% on the two fields requiring genuine document com-

prehension — fields that, for most of this project’s evaluation history, looked like a hard ceiling on what the model could do.

Table 13 returns to the three objectives and two research questions set in Chapter I and states, for each, what this work achieved and where the evidence is.

| Item       | Outcome                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | Evidence                          |
|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------|
| <b>O1</b>  | <b>Met.</b> Four pure calculation functions with no language-model involvement; the model only decides <i>when</i> to call them. Twenty unit tests, written before the code, including a regression test for a caught sign-inversion bug.                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | Section III.3                     |
| <b>O2</b>  | <b>Met.</b> The loop executes whatever tool the model names; its only name-based branch is the model’s own escalation signal. <i>Not shown:</i> that this outperforms a fixed pipeline, since no single-shot baseline was run.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | Section III.4, Figure 2, Table 12 |
| <b>O3</b>  | <b>Met, within limits.</b> Four models evaluated on a raw benchmark and on the real harness; the initial ranking was corrected. The model range (7–14B, one serving stack) and sample sizes are narrow.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | Section III.2, Tables 7 and 8     |
| <b>RQ1</b> | <b>Usefully, but not yet at production level; and no, a raw benchmark does not predict it.</b> The final 14B configuration completed 41/50 jobs with a write (82%, 95% CI 69–90%), with 64–86% accuracy on plumbing fields and 76–81% on comprehension fields; the 7–8B models did markedly worse (5/20 completed, 7% comprehension). The raw benchmark scored qwen3:8b at 0/10 valid calls, yet it completed 8/10 jobs on the real harness, with non-overlapping intervals.                                                                                                                                                                                                                                     | Tables 8 and 9                    |
| <b>RQ2</b> | <b>Architectural separation first, then three safeguards.</b> The calculation must never depend on the model. On top of that: facts the system already knows override the model’s arguments; tool results are remembered across a job; and a measured emissions value is written only with a verbatim quote from extracted text, under hard plausibility bounds and a single-write guard. An advisory consistency check flags ungrounded values earlier in the loop. The quote gate was observed rejecting a live fabrication. Two costs remain: the gate can discard a correctly extracted sibling value, and resistance to deliberate prompt injection is structural only, not tested against crafted attacks. | Section III.4, Tables 10 and 12   |

**Table 13: Objectives and research questions: outcomes.**

Looking ahead, the two most important next steps are measuring the agentic loop against a single-shot extraction baseline on the same harness, and testing on real supplier documents; both would turn the current results from strong evidence on synthetic data into general claims. Beyond those, this work leaves a clear path forward: re-validating the two most recent targeted

fixes at the same full dataset scale the rest of the evaluation insists on; resolving the identified interaction between the single-write design and the source-quote grounding gate; benchmarking additional and fine-tuned models against the same harness-matched methodology; extending the audit trail into a human-review interface that surfaces grounding confidence per field; and restoring the originally specified per-role network topology once its external blocker is resolved. Because deterministic calculation and agentic extraction were kept architecturally separate from the start, each of these is an incremental extension rather than a rebuild. The foundation is in place; what remains is to make the extraction layer as trustworthy as the calculation layer already is.

# Bibliography

- [1] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023. URL <https://arxiv.org/abs/2210.03629>.
- [2] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation, 2023. URL <https://arxiv.org/abs/2202.03629>.
- [3] European Union. Regulation (EU) 2023/956 of the European Parliament and of the Council Establishing a Carbon Border Adjustment Mechanism, 2023. Official Journal of the European Union.
- [4] European Union. Directive 2003/87/EC Establishing a Scheme for Greenhouse Gas Emission Allowance Trading, as Amended, Art. 10a(1a), 2003. Official Journal of the European Union.
- [5] European Commission. Commission Implementing Regulation (EU) 2025/2621 (Default Values and Markup Schedule), 2025. Official Journal of the European Union.
- [6] European Commission. Commission Implementing Regulation (EU) 2026/1412 (EU ETS Free-Allocation Benchmarks 2026–2030), 2026. Official Journal of the European Union.
- [7] Marijana Vidovic, Sarah Marmier, Neza Zore, and José Antonio Moya. Greenhouse gas emission intensities of the steel, fertilisers, aluminium and cement industries. Technical Report JRC134682, European Commission Joint Research Centre, 2023.
- [8] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, 4th edition, 2020. URL <https://aima.cs.berkeley.edu/>.
- [9] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents, 2023. URL <https://arxiv.org/abs/2308.11432>.
- [10] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools, 2023. URL <https://arxiv.org/abs/2302.04761>.
- [11] Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive APIs, 2023. URL <https://arxiv.org/abs/2305.15334>.

- [12] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. AgentBench: Evaluating LLMs as agents. In *International Conference on Learning Representations (ICLR)*, 2024. URL <https://arxiv.org/abs/2308.03688>.
- [13] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 2025. URL <https://arxiv.org/abs/2311.05232>.
- [14] Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, 2023. URL <https://arxiv.org/abs/2212.10511>.
- [15] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. URL <https://arxiv.org/abs/2005.11401>.
- [16] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations (ICLR)*, 2023. URL <https://arxiv.org/abs/2203.11171>.
- [17] Wil M. P. van der Aalst, Martin Bichler, and Armin Heinzl. Robotic process automation. *Business & Information Systems Engineering*, 60:269–272, 2018. doi: 10.1007/s12599-018-0542-4.
- [18] Rehan Syed, Suriadi Suriadi, Michael Adams, Wasana Bandara, Sander J. J. Leemans, Chun Ouyang, Arthur H. M. ter Hofstede, Inge van de Weerd, and Moe Thandar Wynn. Robotic process automation: Contemporary themes and challenges. *Computers in Industry*, 115:103162, 2020. doi: 10.1016/j.compind.2019.103162.
- [19] Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. LayoutLM: Pre-training of text and layout for document image understanding. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2020. URL <https://arxiv.org/abs/1912.13318>.
- [20] Geewook Kim, Teakgyu Hong, Moonbin Yim, Jeongyeon Nam, Jinyoung Park, Jinyeong Yim, Wonseok Hwang, Sangdoo Yun, Dongyoon Han, and Seunghyun Park. OCR-free document understanding transformer. In *European Conference on Computer Vision (ECCV)*, 2022. URL <https://arxiv.org/abs/2111.15664>.
- [21] Derong Xu, Wei Chen, Wenjun Peng, Chao Zhang, Tong Xu, Xiangyu Zhao, Xian Wu, Yefeng Zheng, Yang Wang, and Enhong Chen. Large language models for generative information extraction: A survey. *Frontiers of Computer Science*, 2024. URL <https://arxiv.org/abs/2312.17617>.

- [22] Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection, 2023. URL <https://arxiv.org/abs/2302.12173>.
- [23] Edwin B. Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927. doi: 10.1080/01621459.1927.10502953.