FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Integrating Human Oversight in Automated Data Standardisation: the NOUS Project use case

Sérgio Peixoto

Mestrado em Engenharia de Software

Supervisor: Prof. Ademar Aguiar (FEUP)

Co-supervisor: Prof. Hugo Paredes (UTAD)

July 22, 2026

Integrating Human Oversight in Automated Data Standardisation: the NOUS Project use case

Sérgio Peixoto

Mestrado em Engenharia de Software

Approved in oral examination by the committee:

President: Prof. Nuno Flores

Referee: Prof. Ângelo Martins

Referee: Prof. Ademar Aguiar

July 22, 2026

Abstract

Europe's data spaces for energy depend on combining databases that were never designed to be combined, which requires mapping each proprietary schema onto a shared data standard. Performed by hand, this mapping does not scale, so automated standardisers carry it out, proposing for each source field the standard field it most likely corresponds to. Semantic matching produces matches that are probable, not certain, so a domain expert must review the uncertain cases, and for high-risk systems the EU AI Act makes that oversight a legal obligation. Existing human-in-the-loop architectures, across the peer-reviewed literature, the industrial data-stewardship platforms, and the closest prior work in the NOUS project, were not designed for the supervision a batch-oriented, confidence-scored standardisation pipeline needs: review organised around a whole standardisation job, and approved mappings governed as versioned artefacts.

This thesis identifies those requirements, designs a supervision architecture to meet them, implements the part that can be built ahead of the live standardiser, and evaluates it, taking the NOUS Auto-Standardiser as its use case. Following Design Science Research, a PRISMA systematic review and two practitioner interviews produce a catalogue of twenty supervision requirements and a structured gap analysis, of which only two are fully covered by existing architectures. The architecture provides a full specification for all twenty requirements; its routing, session, and governance components are implemented as a connected workflow over the batch standardisation job, with a mock Auto-Standardiser as the evaluation infrastructure. Evaluation runs on two tracks: against the mock standardiser all fourteen design objectives are satisfied, and a validation with two practitioners, one independent of the design, corroborates the implemented behaviour. The requirements catalogue and the supervision patterns are framed for confidence-scored, batch-oriented standardisation in general, so their transferability beyond the single energy use case is argued from that generality rather than demonstrated. The central contribution is the integration itself: the individual mechanisms are already known, but combining them into one workflow around the batch standardisation job and the versioned mapping profile is a design absent from the reviewed literature, the industrial platforms, and prior NOUS work, and one the evaluation finds coherent as a specification and satisfied on its implemented core.

Keywords: Human-in-the-loop, Data standardisation, Software architecture, Human oversight, Design Science Research

Resumo

Os espaços europeus de dados para a energia dependem da combinação de bases de dados que nunca foram concebidas para ser combinadas, o que exige mapear cada esquema proprietário para uma norma de dados comum. Feito manualmente, este mapeamento não é escalável, pelo que normalizadores automáticos o realizam, propondo, para cada campo de origem, o campo da norma a que mais provavelmente corresponde. A correspondência semântica produz correspondências prováveis, não certas, pelo que um especialista de domínio tem de rever os casos incertos, e, para sistemas de risco elevado, o Regulamento Europeu de IA torna essa supervisão uma obrigação legal e não uma conveniência. As arquiteturas *human-in-the-loop* existentes, na literatura científica, nas plataformas comerciais de gestão de dados e no trabalho anterior mais próximo do projeto NOUS, não foram concebidas para a supervisão de que necessita uma cadeia de normalização orientada a lotes e baseada em confiança: a revisão organizada em torno de um trabalho de normalização completo, e os mapeamentos aprovados geridos como artefactos versionados.

Esta dissertação identifica esses requisitos, concebe uma arquitetura de supervisão para os satisfazer, implementa a parte que pode ser construída antes do normalizador em produção e avalia-a, tomando o Auto-Standardiser do NOUS como caso de uso. Seguindo a metodologia Design Science Research, uma revisão sistemática segundo o PRISMA e duas entrevistas a profissionais produzem um catálogo de vinte requisitos de supervisão e uma análise de lacunas estruturada, dos quais apenas dois estão totalmente cobertos pelas arquiteturas existentes. A arquitetura fornece uma especificação completa para os vinte requisitos; os seus componentes de encaminhamento, sessão e governança são implementados como um fluxo de trabalho integrado sobre o trabalho de normalização em lote, juntamente com um Auto-Standardiser simulado que serve de infraestrutura de avaliação. A avaliação decorre em duas vias: face ao normalizador simulado, os catorze objetivos de design são satisfeitos, e uma validação com dois profissionais, um deles independente do design da solução, corrobora o comportamento implementado. O catálogo de requisitos e os padrões de supervisão são enquadrados para a normalização orientada a lotes e baseada em confiança em geral, pelo que a sua transferibilidade para além do único caso de uso de energia é argumentada a partir dessa generalidade e não demonstrada. O contributo central é a própria integração: os mecanismos individuais já são conhecidos, mas combiná-los num único fluxo de trabalho em torno do trabalho de normalização em lote e do perfil de mapeamento versionado é uma conceção ausente da literatura revista, das plataformas comerciais e do trabalho anterior no NOUS, que a avaliação considera coerente enquanto especificação e satisfeita no seu núcleo implementado.

Palavras-chave: Human-in-the-loop, Normalização de dados, Arquitetura de software, Supervisão humana, Design Science Research

UN Sustainable Development Goals

The United Nations Sustainable Development Goals (SDGs) provide a global framework of 17 goals[1] for a more sustainable future. This dissertation develops a human supervision architecture for automated data standardisation. It contributes most directly to SDG 9, as digital infrastructure and a technological innovation; supports clean energy integration under SDG 7 through its application to renewable energy data in the NOUS project; and advances accountable, transparent decision-making under SDG 16 by keeping automated decisions under human oversight.

SDG 7 Ensure access to affordable, reliable, sustainable and modern energy for all

SDG 9 Build resilient infrastructure, promote inclusive and sustainable industrialisation and foster innovation

SDG 16 Promote peaceful and inclusive societies for sustainable development, provide access to justice for all and build effective, accountable and inclusive institutions at all levels

SDG	Target	Contribution	Performance Indicators and Metrics
7	7.1	Supervised quality gates improve the consistency and trustworthiness of standardised energy data.	Mappings meeting the quality threshold; post-review error rate.
	7.2	Lowers the integration effort for onboarding new solar and wind data sources.	Renewable sources standardised; mapping effort per source.
9	9.1	An auditable oversight layer strengthens the resilience of the energy-data infrastructure.	Requirements coverage; profile versioning and rollback support.
	9.4	Confidence routing and batch review focus expert effort on uncertain cases.	Auto-resolved versus routed items; reviewer workload per batch.
	9.5	A novel, evaluated human-in-the-loop standardisation architecture.	Design objectives satisfied; requirements addressed beyond prior work.
16	16.6	Audit trails, governance approval, and explainable routing make automated decisions accountable.	Decisions captured in the audit trail; separation of duties enforced.
	16.7	Keeps reviewers in the decision loop rather than fully automating standardisation.	Oversight floor enforced; decisions subject to human review.

[1] https://sdgs.un.org/goals

Acknowledgements

Firstly, I would like to thank my two supervisors for their help and guidance throughout this year, despite the many difficulties I faced, and for always being there to help me.

Professor Ademar Aguiar, thank you for presenting me with the intriguing NOUS project, for sharing valuable tips, and for inspiring me with your contagious passion for software engineering. To Professor Hugo Paredes, thank you for accepting me into the NOUS team, for always making time to meet with me every week, for providing incredible guidance, and for answering all of my questions. To the brightest and hardest-working team: Carlos Duarte, Filipe Correia and Marco Oliveira, who played an important role in this journey, thank you for always being to help.

To my parents, who, although they did not always understand what my work was, still supported me through thick and thin. I thank them for providing me with all the opportunities I have had over the last five years.

To someone very special, Roks: thank you for never letting me give up, for believing in me and for encouraging me to push through boundaries and challenge myself every day.

Thank you to JuniFEUP, with whom I've shared four of the last five years. I can say with certainty that these were the best years of my life. I have met and worked with the most inspiring people I have ever known in that office. Special thanks go to the six little stars who made my last year the best of my life: Cacho, Clarinha, Maria, Miller, Pedrinho and Pipo. To my department, whom I had the honour of leading this year: you are all truly inspiring and amazing people to work with. I am sure you will all be successful and I wish you all the best. You have shaped me into the person I am today. You have encouraged me to challenge myself and step out of my comfort zone. I will certainly come back to see how things are going.

To my friends outside university, thank you for all your support throughout my life. I owe you all an apology for being so absent in the last couple of months.

And then, in a blink of an eye, here I am, saying goodbye to FEUP, to JuniFEUP, and to all the amazing people I have come across. Now, a new journey begins, and, while excited to see what the future holds for me, I can't forget my roots and surely can't forget all the people that made me who I am today and who inspired me to be a better version of myself.

Thank you all.

Sérgio Peixoto

"When small men begin to cast long shadows, it means the sun is setting"

Lin Yutang

Contents

List of Figures

List of Tables

List of Acronyms

AI	Artificial Intelligence
API	Application Programming Interface
AS	Auto-Standardiser
BERT	Bidirectional Encoder Representations from Transformers
CIM	Common Information Model
DO	Design Objective
DSR	Design Science Research
EU	European Union
FEUP	Faculdade de Engenharia da Universidade do Porto
GDPR	General Data Protection Regulation
HITL	Human-in-the-Loop
HTTP	HyperText Transfer Protocol
IEC	International Electrotechnical Commission
JSON	JavaScript Object Notation
LLM	Large Language Model
ML	Machine Learning
NLP	Natural Language Processing
ORM	Object-Relational Mapping
PPC	Public Power Corporation
PRISMA	Preferred Reporting Items for Systematic Reviews and Meta-Analyses
RBAC	Role-Based Access Control
REST	Representational State Transfer
SAREF	Smart Appliances REFerence ontology
SIEM	Security Information and Event Management
SME	Subject Matter Expert
SPA	Single-Page Application
SQL	Structured Query Language
SR	Supervision Requirement
UI	User Interface
UUID	Universally Unique Identifier
WCAG	Web Content Accessibility Guidelines

Chapter 1

Introduction

1.1 Context and Motivation

Europe's data spaces for energy, mobility, and the Green Deal depend on combining data that was never designed to be combined. Each contributing organisation maintains its own databases, with field names, units, and structural conventions accumulated over years of independent development. Bringing these sources together requires mapping every proprietary schema onto a shared data standard, so that a measurement recorded one way in one system is recognised as the same quantity everywhere else. Performed manually, this mapping does not scale: the fields are numerous, the domain knowledge required is specialised, and every new data source repeats the effort [34].

The NOUS project, an EU Horizon initiative federating cloud, edge, and high-performance computing resources, treats this standardisation as a core enabling capability [29]. Within NOUS, translating heterogeneous source schemas to target standards such as SAREF, CIM, and IEC 61850 [10, 18–20] is the task of an Auto-Standardiser (AS). The AS applies natural-language and semantic-similarity techniques to propose, for each source field, the standard field it most likely corresponds to. It automates the per-source mapping effort that does not scale by hand and applies the same matching criteria consistently across sources.

It does not remove the need for human judgement. Semantic matching produces matches that are probable, not certain. Synonymous terms, multilingual labels, and ambiguous field definitions leave a residue of cases the algorithm cannot resolve with confidence, and there an incorrect mapping silently corrupts the standardised output. The quality of the result depends on a person reviewing the uncertain matches, correcting them where necessary, and returning those corrections so the system improves. Supervision is not a fallback for when automation fails; it is specified as part of how the AS works [24].

That oversight is also a legal requirement, not only sound practice: the EU AI Act makes meaningful human oversight an obligation for high-risk AI systems, a category that covers automated decision-making on critical energy infrastructure [11]. Making that oversight practical rather than nominal is an engineering problem in itself: uncertain cases have to reach a suitable reviewer in good time, with enough context to support a sound decision.

The AS core, the translation algorithm and the component that learns from corrections to improve it, is specified and under development; the supervision layer around it is not [24]. Which uncertain matches should reach a human, how they should be presented for a reviewer to act on, how the resulting corrections should be captured and fed back, and how approved mappings should be governed once in use are questions that specification leaves open. The closest prior work within NOUS, Maldonado's reference architecture for human-centred AI systems, treated supervision in general terms and named its transfer to other sectors as an open question [25]. How the supervision layer should be specified for automated data standardisation, and which of its parts can be built and evaluated ahead of the live AS, is the problem this thesis takes up.

1.2 Problem Statement

Automated semantic matching is the practical alternative to manual schema mapping, but it carries a known limitation: similarity algorithms produce matches that are probable, not certain [34], and no algorithm resolves every case correctly. When that happens, a human must decide.

The AS, developed within the NOUS project, is built around this constraint. High-confidence matches proceed automatically; when confidence falls below an acceptable threshold, a domain expert must review the candidate mappings, choose or correct the best match, and the learning component records that correction to improve future results. This human-in-the-loop mechanism is not optional: it is part of the system's specification, which mandates that the system learns from manual corrections and detects when standardised output drifts from an approved mapping profile [24, 30].

That supervision layer is what this thesis investigates, and general-purpose HITL architectures are the natural starting point. They share a basic loop: a model makes predictions, a human reviews and corrects them, and those corrections return as labelled examples that retrain the model [1, 17, 28]. The AS keeps this loop in outline but diverges from it on five specific properties, some straining its assumptions and others with no counterpart in it. Its routing signal is the model's confidence, as in active learning's uncertainty sampling [28], but its cut-off cannot be generic: it must be calibrated to the mapping-quality level the domain requires. Its workflow is not a real-time stream but discrete jobs, each producing a batch of flagged fields a reviewer resolves as a session before committing an approved mapping profile. Its corrections are training data that feed a retraining loop [37], but making that loop dependable as a supervision mechanism means governing it: deciding when retraining is triggered, validating each retrained model before it is promoted, and keeping model versions tied to the profiles they produced. The use case's requirements add a governance lifecycle with no HITL equivalent: approved profiles must be versioned, available for rollback, and monitored for drift [30]. And because the AS maps to multiple target standards, a candidate's meaning depends on the standard, so a generic similarity score without standard-specific context underserves the reviewer.

These five properties (confidence-based routing, batch session workflow, correction-driven retraining, mapping profile governance, and multi-standard explainability) together establish AS

supervision as a structurally distinct domain; Chapter 4 develops them into formal requirements and sets them against existing architectures to determine which remain unaddressed.

The closest prior work within NOUS confirms the gap from inside the project. Maldonado's reference architecture for Human-Centered AI Systems, built from general HITL components and validated on a real-time crisis-management use case, was candid about where it stopped: "longer-term challenges like model drift and the generalizability of the architecture to other sectors remain open questions" [25]. The AS is precisely such a case, running as batch jobs scored by semantic similarity and coupled to a retraining loop rather than an event-driven stream, and applying those components to it without modification leaves the five mismatches unaddressed. Human oversight has already been taken into the same energy use case by a second NOUS thesis, but at the forecasting stage, supervising the models trained on already-standardised data while treating standardisation itself as an upstream input [12]; the standardisation layer this thesis addresses is the stage neither NOUS precedent supervises.

These gaps, in general HITL practice and in the closest NOUS precedent alike, are what this thesis addresses. The five properties are not separate problems to be solved in isolation: they turn on one standardisation job and the one body of approved mappings it yields, so assembling the oversight from separate tools, each addressing a property on its own, would leave the connections between them unmanaged. The response is therefore a single supervision architecture specified for all of them, with the part realisable ahead of the live AS implemented and evaluated.

1.3 Research Questions

Three questions guide this research, each building on the one before. The first does not presume that existing HITL work covers what automated data standardisation demands; it establishes the requirements and the coverage gap through systematic analysis of the system's technical and use-case specifications, practitioner interviews, and the literature. The second asks how the unmet requirements should be addressed through architecture and selective implementation; the third, how well the implemented result does so.

RQ1: What supervision requirements arise in automated data standardisation workflows, and to what extent are they addressed by existing human-in-the-loop architectures?

RQ2: How should the supervision requirements identified in RQ1 be addressed through architecture design and selective implementation within the NOUS Auto-Standardiser context?

RQ3: To what extent does the implemented supervision architecture address the requirements identified in RQ1, as assessed through technical evaluation against a mock Auto-Standardiser API and practitioner validation with domain experts?

1.4 Hypothesis

The problem statement assumes the supervision challenges are structural to automated data standardisation, not generic HITL problems in a new domain. This thesis tests whether that assumption is load-bearing: whether the components built for the realisable subset actually satisfy the requirements they target.

> *The supervision workflow components realisable without the live Auto-Standardiser ML pipeline can satisfy the functional and non-functional requirements they were built to address, as assessed through technical evaluation against a mock Auto-Standardiser API and validation with domain experts.*

The claim is one of both sufficiency, that components designed for these specific demands meet them rather than displacing the problem elsewhere, and compatibility, that the result extends rather than replaces the general HITL framework Maldonado established for NOUS.

It does not claim the implementation generalises beyond the energy use case, nor that it resolves every AS supervision requirement: the architecture is specified for all of them, but only the subset realisable without the live AS pipeline is built and evaluated here. Whether the remaining, architecture-specified components prove tractable in the same way is left open.

1.5 Contributions

This thesis makes three contributions, one for each research question; the objectives in Section 1.6 set out how each is produced.

C1: Requirements analysis. A domain-specific set of supervision requirements for automated data standardisation, set against existing HITL architectures (peer-reviewed patterns, industrial platforms, and Maldonado's NOUS framework [25]) to establish what is already handled, partially handled, or unaddressed. Its output is not a bare requirements list but a structured gap analysis that locates each requirement against what existing architectures already provide, and finds the primary gap to be the absence of an architecture integrating them for the standardisation domain, not the absence of the individual mechanisms.

C2: Architecture and implementation. A full supervision architecture specification for all twenty requirements from C1, whose contribution is the integration of the supervision concerns into a single workflow for confidence-scored, batch-oriented standardisation rather than any individual mechanism; a concrete implementation of the workflow components realisable without the live AS ML pipeline; and a mock Auto-Standardiser API that provides the technical evaluation infrastructure for C3.

C3: Empirical evaluation. An evidence-based answer to RQ3, from two complementary tracks: a technical evaluation against the mock Auto-Standardiser API, and a practitioner validation with domain experts.

1.6 Objectives

Four objectives operationalise the contributions above, each mapping to a stage of the research: deriving the requirements, characterising the gap, designing and building the artefact, and evaluating it.

O1: Derive supervision requirements. Draw them from the project's technical and use-case specifications [24, 30], practitioner interviews, and peer-reviewed HITL literature, as functional and non-functional requirements specified in full in Chapter 4.

O2: Characterise the gap. Survey that literature through a PRISMA-guided systematic review and compare it, with Maldonado's NOUS framework [25], against the O1 requirements to produce a structured coverage map.

O3: Design the architecture and implement the realisable subset. Specify an architecture addressing all O1 requirements, implement the components realisable without the live AS ML pipeline, and build a mock Auto-Standardiser API as the evaluation environment for O4; ground the design in peer-reviewed literature, independently of Maldonado.

O4: Evaluate empirically. Assess the O3 implementation through two tracks: a technical evaluation against the mock Auto-Standardiser API, and a practitioner validation with domain experts.

1.7 Research Methodology

This thesis adopts Design Science Research (DSR) as its methodological framework [16, 32]. DSR generates knowledge through the creation and evaluation of novel artefacts: designed solutions to real engineering problems, rather than through observation or explanation of existing phenomena.

Alternative paradigms like case study research or grounded theory are better suited to understanding phenomena than to constructing solutions, which is why DSR is the appropriate choice here.

The specific model followed is the six-stage process proposed by Peffers et al. [32], the same one Maldonado used for the prior NOUS HITL work [25]; using the same framework allows direct methodological comparison between the two theses without conflating their research questions or artefacts. Its stages, problem identification, definition of objectives, design and development, demonstration, evaluation, and communication, map onto the chapters that follow (Section 1.8), with this chapter covering the first two.

Four methodological choices warrant explanation. The requirements analysis in Chapter 4 draws on semi-structured interviews with practitioners as a primary data source alongside documentary sources (the system's technical and use-case specifications). Interviews are justified because technical specifications capture what the system is required to do; they do not capture practitioner mental models, operational priorities, or the tacit knowledge that determines whether a supervision

mechanism is usable in practice. The literature review follows PRISMA 2020 guidelines [31] for systematic, reproducible coverage of HITL architecture patterns, active learning, schema matching, and data governance. The energy use case is the domain context for both requirements elicitation and evaluation rather than a synthetic scenario, because it has real actors, requirements, and data standards. A synthetic scenario would be easier to satisfy, which is precisely why it would be less informative. The evaluation uses a mock Auto-Standardiser API rather than the live system. Because the supervision architecture is scoped to the components realisable around the AS through a stable interface, rather than the AS-internal ML pipeline, a mock implementing that interface is the appropriate evaluation environment: it exercises the supervision components against controlled, repeatable inputs that an evolving live system could not provide.

Five limitations bear on how the results should be read. Evaluation against a mock API cannot guarantee that every production characteristic of the live AS is captured; edge cases arising only at operational scale or from real data distributions may not appear under controlled conditions. The evaluation is confined to the energy use case: whether the solution transfers to Mobility or Green Deal standardisation contexts is not established here. The architecture addresses all twenty supervision requirements but the implementation is limited to the subset realisable without the live AS pipeline; the remaining components are architecture-specified and constitute directions for future implementation. The practitioner validation track includes a participant who informed the requirements analysis, whose shared background reduces the risk of misaligning requirements and evaluation criteria but introduces a confirmation-bias risk; this is countered by prioritising an independent reviewer with no part in the design as the primary validator, and the residual risk should be weighed when interpreting the Chapter 7 results. Finally, the closest prior work is grey literature: Maldonado's thesis has not been externally peer-reviewed, and all design decisions here are grounded in peer-reviewed literature independently of his work.

1.8 Thesis Structure

The remainder of this dissertation is organised into seven further chapters.

Chapter 2 "Background" establishes the technical and regulatory context: data standardisation and schema matching, the NOUS Auto-Standardiser, human-in-the-loop and interactive machine learning, and the EU AI Act oversight mandate.

Chapter 3 "Literature Review and State of the Art" surveys existing supervision architectures through a PRISMA 2020 systematic review and produces the coverage map that feeds the gap analysis.

Chapter 4 "Requirements Analysis" derives the supervision requirements from the project's technical and use-case specifications and two practitioner interviews, and sets them against that coverage map and Maldonado's work to establish what existing architectures leave unaddressed. This is contribution C1.

Chapter 5 "Design" specifies the architecture addressing all those requirements, with its design objectives, decisions, and component clusters.

Chapter 6 “Implementation” builds the components realisable without the live AS pipeline, together with the mock Auto-Standardiser API that also enables the demonstration. Chapters 5 and 6 are contribution C2.

Chapter 7 “Evaluation” assesses the implementation against the mock API and with domain experts. This is contribution C3 and the answer to RQ3.

Chapter 8 “Conclusion” revisits the research questions, states contributions and limitations, and sets out the future work the architecture specifies but leaves unimplemented.

Chapter 2

Background

2.1 Introduction

This chapter establishes the conceptual and project-level ground the rest of the thesis builds on. Section 2.2 introduces automated data standardisation and the schema matching problem the Auto-Standardiser addresses, including the three target standards relevant to the energy use case. Section 2.3 describes the NOUS project, the AS as a system, and the regulatory frame within which supervision obligations arise. Section 2.4 covers the human-in-the-loop and interactive machine learning foundations the rest of the thesis relies on. The systematic survey of supervision patterns in the data-integration literature is taken up in Chapter 3; the analysis of what the AS specifically requires follows in Chapter 4.

2.2 Automated Data Standardisation and Schema Matching

Schema matching is the task of identifying semantic correspondences between elements of two structured data representations [34]. In the data integration setting, one representation is a source schema (a database table, an XML document, a JSON structure) developed independently for an organisation's own purposes. The other is a target standard against which the source must be aligned for cross-organisational interoperability. The mapping product is structured: each source field is either bound to a standard field, paired with a confidence-scored candidate that a human must adjudicate, or flagged as unmatchable.

The literature distinguishes three granularity levels. Structural schema matching identifies correspondences between metadata elements such as columns, attributes, and keys; it is the operation most directly associated with the term. Ontology alignment operates between concepts in distinct ontologies, where the strict 1:1 field correspondence model gives way to subsumption, equivalence, and broader relational mappings [41]. Entity resolution operates at the instance level, identifying records that refer to the same real-world entity across heterogeneous sources [6]. The NOUS Auto-Standardiser operates predominantly at the schema and ontology levels. Entity resolution sits in the wider data lifecycle but lies outside the supervision layer this thesis addresses.

Algorithmic approaches divide along three broad lines. Linguistic matching uses string-similarity metrics over field names and labels: edit distance, token overlap, character n-gram comparison [34]. Semantic matching incorporates external knowledge through ontologies, thesauri, and curated vocabularies, resolving cases where surface form is misleading but underlying meaning aligns [34]. Embedding-based matching projects field names and accompanying context into a high-dimensional vector space, typically via transformer encoders, where cosine distance approximates semantic similarity even when lexical overlap is absent [5]. Production systems usually combine signals from several families. No single strategy reliably distinguishes correct from incorrect matches across the heterogeneity present in industrial data.

The defining property of automated matching, regardless of strategy, is that its output is uncertainty-qualified rather than categorical [41]. The system produces a confidence-scored candidate set: the top match may be correct, but the second or third candidate is sometimes the better choice, and occasionally the correct mapping is absent from the list [5]. Whether and how to surface this uncertainty to a human, rather than committing to the top candidate by default, is the structural premise behind the supervision layer this thesis specifies and analyses.

Three target standards are central to this thesis's energy use case. SAREF (Smart Appliances REFerence ontology) is an ETSI reference ontology for IoT-enabled smart appliance semantics, developed with the energy domain as a primary motivation and carrying a dedicated energy extension, SAREF4ENER [10]. The Common Information Model (CIM), defined in IEC 61968 and IEC 61970, is a data model for power system operations, transmission, and distribution [18, 20]. IEC 61850 standardises communication and information modelling for power utility automation, including distributed energy resources [19]. They occupy distinct layers of the power-systems data model: SAREF at device-level capability semantics, CIM at network topology and asset inventory, IEC 61850 at the operational substation. The same source field (for example, a voltage measurement from a solar park) may map plausibly into more than one of them under different target choices, and the contextual information a reviewer needs to adjudicate that choice differs from one standard to the next.

2.3 The NOUS Project, the Auto-Standardiser, and the Regulatory Frame

This section situates the Auto-Standardiser within the NOUS project and the energy use case it serves, describes it as a system, and sets out the regulatory frame that makes human supervision an obligation rather than a matter of good practice.

2.3.1 The NOUS Project, the Auto-Standardiser, and the Energy Use Case

NOUS (EuropeaN ClOUd ServiceS) is a Horizon Europe project running from January 2024 to December 2026, federating High-Performance Computing, edge nodes, and data-management infrastructure into a European-controlled cloud-edge-data continuum [29]. The broader motivation

is the development of European Data Spaces: federated ecosystems for sovereign, controlled data exchange across strategic sectors including energy, mobility, and the European Green Deal.

Within NOUS, Task 5.4 (T5.4) addresses the Auto-Standardiser, a microservice tasked with translating heterogeneous source databases into established target standards. The AS is the engineering responsibility of AETHON, an industry partner in the consortium; the human supervision architecture this thesis specifies, and partly implements, sits on top of it.

NOUS includes several validation use cases, of which Use Case 2 (UC2) is the validation context for this thesis. UC2 is conducted in partnership with the Public Power Corporation (PPC), Greece's principal electricity utility. PPC operates a heterogeneous portfolio of generation assets, including solar parks, wind farms, hydroelectric stations, and conventional thermal units. Each produces data streams whose schemas were developed independently by different engineering teams over many years. The use case targets improvements in PPC's energy-prediction pipeline across short-, medium-, and long-term horizons; achieving them requires harmonising the ingest streams into shared standards, primarily CIM and SAREF, before any forecasting model is trained.

Two actor roles are defined in the use case and recur throughout this thesis. The PPC Data Engineer is responsible for reviewing flagged mappings, capturing corrections, and proposing new mapping profiles. The PPC Data Governance Lead is responsible for approving or rejecting promotion of those proposed profiles, and for rolling profiles back when drift is detected. The separation between the two is not a process convention but an organisational and architectural requirement: a single user cannot legitimately discharge both functions for the same artefact.

2.3.2 The Auto-Standardiser as a System

The AS translates a complete source database to a target standard. A PPC Data Engineer initiates a standardisation job; the AS translates each source field into a confidence-scored ranked list of target standard candidates using its semantic similarity model; the job emits a set of per-field outputs covering the whole database. Fields above a configurable confidence threshold are intended to proceed automatically; fields below it are flagged for human review [24, 30].

Two architectural commitments in the task's development plan distinguish the AS from a purely automated translator. The first is the supervised ML component: human corrections captured during review feed back as training data, retraining the matching model on standard-partitioned correction batches [24]. The second is the mapping profile: each approved set of field mappings is persisted as a named, versioned artefact governing how a specific database is standardised going forward, with explicit rollback and drift-monitoring obligations defined in the use case's Story 1.1 (AC6 and AC7) [30].

One distinction in this arrangement conditions the design that follows. What the AS presents for human review is the suggested mapping between a source column and a target standard field, and what an approved session yields is a versioned mapping profile, not a committed transformation of the source data (Story 1.1, AC2 and AC5) [30]. Standardisation of the data is governed by whichever profile is active: rollback is defined over the active standardisation process and takes effect by reverting to a previously approved profile, not by undoing changes to the source database (AC6)

[30]. The source data is therefore never mutated irreversibly by the supervision workflow. The standardised output is a derivation from the source and the active profile version, and a superseded mapping is corrected by re-standardising under the new version, an operation internal to the AS.

The supervision layer that intermediates between the AS and the human reviewer is the architecture this thesis specifies (Chapter 5) and partly implements (Chapter 6): the routing of flagged items, the structure and pacing of the review session, the lifecycle of the resulting profile, the capture of corrections, and the alerts when output drifts from an approved profile. It is also the part of the AS most directly conditioned by the regulatory frame.

A separate NOUS thesis supervises the next stage of this pipeline (the forecasting models that consume standardised data), where the AS is an upstream input rather than the system under supervision [12].

2.3.3 Regulatory Frame

The EU AI Act (Regulation 2024/1689) places automated decision systems deployed in energy and other critical infrastructure under its high-risk category (Annex III) and imposes a meaningful human oversight obligation through Article 14. Operators are required to ensure that natural persons can monitor system behaviour, intervene to correct outputs, and bring affected decisions to a stop. Article 12 requires automatic recording of significant system events sufficient to reconstruct the decision history of any affected output.

The General Data Protection Regulation (Regulation 2016/679) overlaps with this frame in two relevant respects. Article 22 restricts automated decision-making that produces legal or similarly significant effects on a natural person. Article 32 mandates access controls and access-logging for processing of personal data. Although PPC's mapping data is not directly personal data, the supervision audit trail captures identifying information about the reviewers themselves and falls under Article 32 access-log obligations.

The European Accessibility Act (Directive 2019/882) and the harmonised standard EN 301 549 set accessibility requirements for digital services procured or deployed by EU public bodies. PPC's status as a regulated infrastructure operator brings the supervision interface within the scope of these obligations.

These instruments do not specify architectural patterns. They specify outcomes: oversight that works in practice, decisions that can be reconstructed after the fact, interfaces that do not exclude. The supervision requirements derived in Chapter 4 translate these outcome obligations into testable design constraints on the AS supervision layer. Several follow directly from this regulatory frame: SR-F-10 (audit trail), SR-NF-04 (retention and GDPR), SR-NF-06 (minimum oversight floor), and SR-NF-07 (accessibility).

2.4 Human-in-the-Loop and Interactive Machine Learning

Human-in-the-loop machine learning describes systems where human judgement is structurally embedded in the learning or decision process rather than treated as an external check after the

fact [17, 28]. The motivation is twofold. Where the problem cannot be fully specified in advance (open-vocabulary classification, ambiguous segmentation, schema mapping under heterogeneous source vocabularies), a model trained on a fixed objective drifts from what the user actually needs [1]. Where the cost of an incorrect output is significant, an automated system that cannot defer to a person leaves the operator no mechanism for catching predictable failure modes before they propagate [17].

A closely related thread is interactive machine learning, formalised by Amershi et al. [1], which emphasises rapid iterative interaction between users and learning systems through short feedback cycles, mixed-initiative control, and end-user model adjustment. Where classical HITL frames the human role primarily as a labeller [28], IML frames it as a co-designer: the user's actions shape the learned objective, not just the training set.

Active learning is the canonical mechanism for selecting which examples are surfaced to the human [37]. Rather than requesting labels on a random sample, the learner identifies examples whose labels would most improve the model and requests those preferentially. Uncertainty sampling, querying the examples on which the current model is least confident, is the simplest and most widely deployed variant. The active learning literature is predominantly concerned with training-time label acquisition: the learner is being built, and human input is the most expensive resource in the loop.

The distinction between training-time and inference-time supervision is load-bearing in the AS context, and is taken up systematically in Chapter 3. Training-time supervision selects training examples for a model that is not yet deployed; the loop terminates once the model converges. Inference-time supervision routes uncertain predictions from a deployed system to a human reviewer; the loop runs indefinitely as new inputs arrive. The two share an architectural primitive (uncertainty-based selection), but the operational concerns differ. Inference-time supervision must accommodate a fixed reviewer budget, an ongoing stream of decisions, and a feedback mechanism that closes back into the deployed system without disrupting it.

A related pattern is selective prediction, or learning with a reject option [13]. A classifier is augmented with an abstain action: when its confidence on an input falls below a threshold, it returns "I do not know" rather than committing to a class. The architectural question (what happens to the abstained inputs) is left open by the prediction model itself. In a fully automated pipeline, abstained inputs may be dropped or returned to the caller. In a HITL system, they are routed to a human, and the supervision layer specifies how that routing, review, and resulting correction are integrated. The AS is structurally a selective-prediction system at the per-field level: the confidence score over candidate mappings is the abstain signal, and the supervision layer is what closes the loop.

2.5 Summary

This chapter has established three reference points. First, automated data standardisation is structurally an uncertainty-bearing problem: the matching algorithm produces ranked confidence-scored candidates rather than categorical verdicts, and surfacing that uncertainty to a human is the architectural premise of the AS's design. Second, the NOUS AS is the concrete system in

scope, operating against PPC's heterogeneous energy data under a regulatory frame that requires meaningful human oversight, immutable audit, and accessibility-aware design. Third, the human-in-the-loop and interactive machine learning foundations distinguish training-time from inference-time supervision. Chapter 3 surveys what existing supervision architectures provide; Chapter 4 derives what the AS specifically demands.

Chapter 3

Literature Review & State of the Art

3.1 Introduction

This chapter answers the second part of RQ1 by synthesising the PRISMA corpus against AS-relevant dimensions: to what extent existing HITL architectures address supervision requirements for AS workflows. Chapter 4 answers the first from the operational context.

Most human-in-the-loop literature on data integration addresses training-time label acquisition: selecting which unlabelled pairs a human should annotate to improve a model before deployment. Inference-time supervision is a different problem: routing uncertain predictions to reviewers in a deployed system, handling uncertain cases safely, and using reviewer decisions to improve future performance. The corpus overwhelmingly addresses the first problem. This distinction between training-time and inference-time HITL organises the synthesis and determines Gap 1.

3.2 Literature Review Process

The review follows PRISMA 2020 guidelines [31] to provide a transparent, reproducible account of search design, filtering, and inclusion decisions.

Search Design

The search query uses a two-block Boolean structure. Block 1 covers the supervision dimension: HITL architectures, confidence-based routing, uncertainty sampling, correction capture, feedback governance, approval workflows, and human escalation mechanisms. Block 2 anchors the domain: data standardisation, schema matching, semantic interoperability, ontology alignment, and related field-mapping concepts. Together they restrict retrieved papers to human supervision mechanisms in data integration contexts specifically. The query was run against ACM Digital Library, IEEE Xplore, and Scopus with database-specific field-prefix syntax; searches were restricted to English-language articles and surveys published between 2018 and 2026.

("human-in-the-loop" OR "HITL" OR "human oversight" OR "human supervision" OR "human-centered AI" OR "human-centred AI" OR "interactive machine learning" OR "expert review" OR "review queue" OR "annotation queue" OR "human-AI collaboration" OR "oversight mechanism" OR "active annotation" OR "approval workflow" OR "human validation" OR "supervision requirements" OR "AI accountability" OR "audit trail" OR "auditability" OR "human control" OR "confidence-based routing" OR "confidence threshold" OR "uncertainty-based delegation" OR "selective automation" OR "selective prediction" OR "abstention" OR "reject option" OR "rejection option" OR "defer to human" OR "deferred prediction" OR "uncertainty sampling" OR "risk-based routing" OR "human escalation" OR "low confidence" OR "fallback to human" OR "correction capture" OR "human correction" OR "feedback-driven learning" OR "correction-driven" OR "retraining pipeline" OR "learning from corrections" OR "human feedback integration" OR "feedback governance" OR "human-in-the-loop retraining" OR "correction loop") AND ("data standardisation" OR "data standardization" OR "schema matching" OR "schema mapping" OR "semantic interoperability" OR "semantic matching" OR "automated mapping" OR "column mapping" OR "field mapping" OR "data integration" OR "ontology alignment")

Result Pipeline

Figure 3.1 shows the full screening process as a PRISMA 2020 flow diagram [15]. Database searches returned 2,052 records (ACM Digital Library: 1,419; IEEE Xplore: 364; Scopus: 269). Publication-type filtering reduced this to 873 records; deduplication in Rayyan removed 49, leaving 824 for title and abstract screening against the criteria in Table 3.1. Title and abstract screening excluded 795 records; full-text review of the remaining 29 excluded 13 further records, primarily for treating HITL as a training technique only (EC1), unrelated application domain (EC2), or short-form publication (EC5). Sixteen papers satisfied all inclusion criteria and form the analytical corpus for Section 3.3.

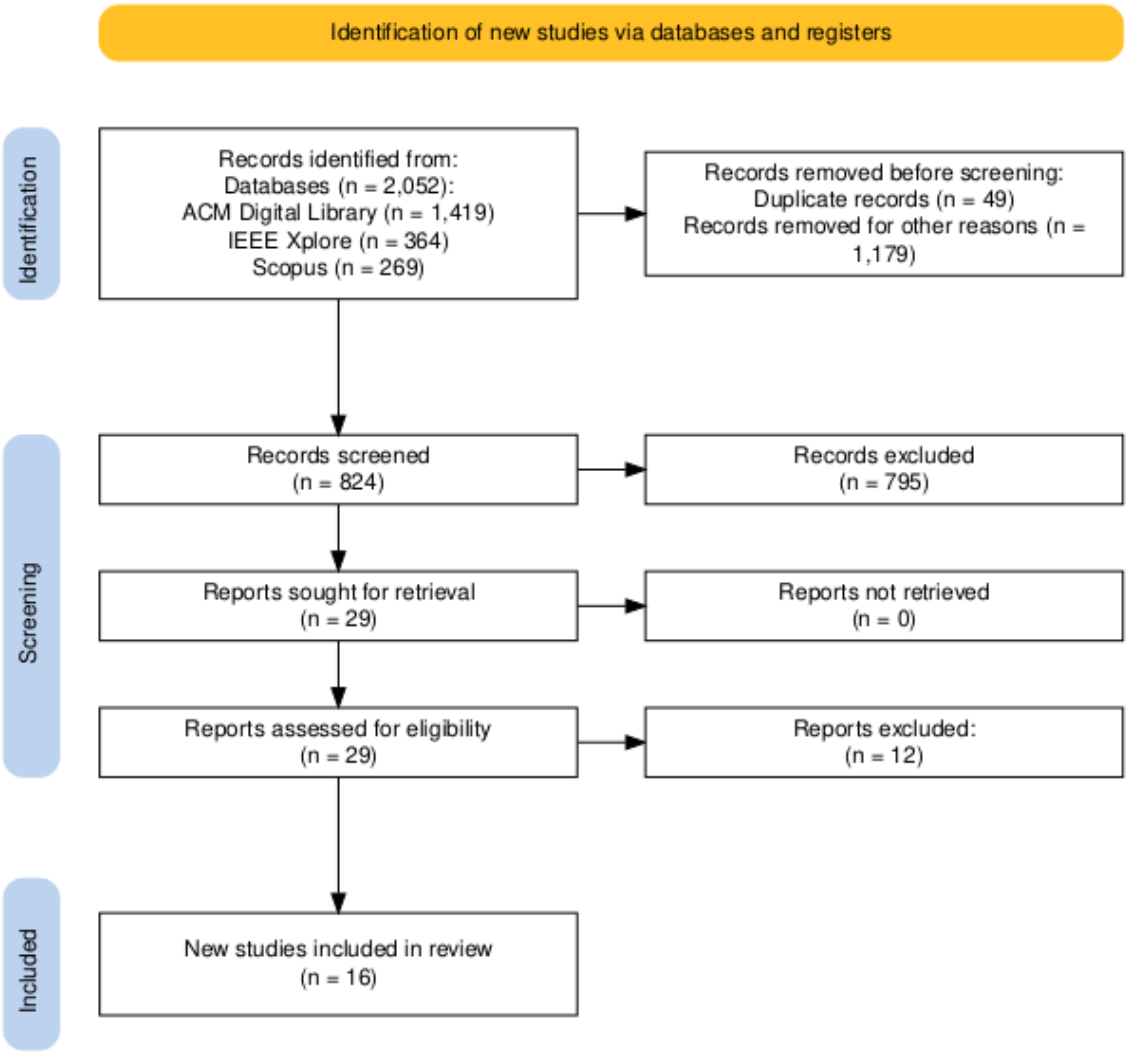

Figure 3.1: PRISMA 2020 flow diagram for the literature screening and selection process.

Inclusion and Exclusion Criteria

ID	Criterion
Inclusion criteria	
IC1	Addresses human supervision or oversight in an automated or AI system
IC2	Situated in a data integration, schema matching, semantic interoperability, or automated mapping context
IC3	Contains a design, architectural, or requirements-level contribution to human supervision mechanisms
IC4	Peer-reviewed publication (journal article, conference paper, or proceedings)
IC5	Published in English
IC6	Published between 2018 and 2026 (inclusive)
Exclusion criteria	
EC1	HITL mentioned only as a training technique; no discussion of supervision architecture or review workflow
EC2	Unrelated application domain (e.g. medical imaging, legal documents) despite matching query terms
EC3	Crowdsourcing or crowd-annotation platform — anonymous paid-task model incompatible with expert-review context
EC4	Purely algorithmic contribution with no discussion of how humans interact with the system architecturally
EC5	Short-form publication lacking sufficient depth: workshop papers, 2-page abstracts, extended abstracts, posters
EC6	Duplicate publication — retain most complete version
EC7	Retracted or not peer-reviewed

Table 3.1: Inclusion and exclusion criteria

Manually added foundational papers

Five foundational works were added outside the systematic review: Amershi et al. [1] on interactive ML, Holzinger [17] on HITL, Monarch [28] on HITL machine learning, Rahm and Bernstein [34] on schema matching, and Settles [37] on active learning. All pre-date the 2018 search window or are not peer-reviewed publications; they are excluded from the dimension coding but cited in Section 3.3 for conceptual grounding.

Thematic coding and cluster organisation

The coding scheme was derived inductively: iterative full-text reading of the corpus identified recurring supervision-relevant architectural properties, which were consolidated into the eight

dimensions listed in Table 3.2, covering the design concerns most relevant to deployed automated data standardisation systems. Each of the 16 included papers was coded against these dimensions as fully addressed (F), partially addressed (P), or absent (-). Dimension (7), empirical evaluation of supervision outcomes, characterises evidentiary grounding rather than a design capability and generates no gap candidate.

Paper	**(1)**	**(2)**	**(3)**	**(4)**	**(5)**	**(6)**	**(7)**	**(8)**
SIGMOD Survey	P	P	P	-	P	P	-	-
Batman & Robin	-	-	F	F	-	P	P	-
What Type	-	-	F	P	-	P	F	-
DataSpeck	F	F	F	F	P	-	F	P
Scrabble	F	F	P	P	F	P	F	P
DaME	P	P	F	P	P	P	F	P
Enhancing Construction Data	F	P	P	P	F	P	P	-
DIAL	P	F	-	-	P	-	F	P
Datalignment	-	-	F	F	P	-	-	-
Balancing Privacy	-	-	P	F	-	P	F	P
CopycHats	-	P	P	P	-	P	P	P
LLM Harmonized	P	-	P	P	F	-	P	-
MExI	-	-	P	-	-	F	F	-
RTS	F	-	F	P	-	P	F	-
ALFA	P	F	P	-	P	-	P	P
CPFilter	F	P	P	P	-	P	P	F
F count	**5**	**4**	**6**	**4**	**3**	**1**	**8**	**1**
P count	**5**	**5**	**9**	**8**	**6**	**10**	**6**	**7**
- count	**6**	**7**	**1**	**4**	**7**	**5**	**2**	**8**

Table 3.2: Eight-dimension coverage coding (F: fully addressed; P: partially addressed; -: absent). Columns: (1) Supervision trigger; (2) Case selection; (3) Reviewer task design; (4) Information shown; (5) Feedback loop; (6) Reviewer quality; (7) Supervision empirically evaluated; (8) Reviewer budget management.

The inductive coding that produced the eight dimensions also showed that they fall into four broader supervision concerns, and the synthesis is organised around these four clusters. The first gathers the dimensions that govern when and where supervision is invoked, covering supervision triggers, case selection within the uncertain set, and the budgeting of reviewer effort (Section 3.3.1). The second concerns the reviewer's working context: how the review task is designed and what information the interface provides (Section 3.3.2). The third follows a correction once it is made, through feedback integration and reuse (Section 3.3.3). The fourth concerns the reviewers themselves, and how their quality and expertise are assessed (Section 3.3.4). The remaining dimension, empirical evaluation of supervision outcomes, characterises evidentiary grounding rather than a design concern and forms no cluster. Each of the 16 papers was then assigned to the cluster matching its primary contribution; Table 3.3 shows that assignment.

Paper	Authors	Year	Venue	Primary cluster	Secondary cluster
SIGMOD Survey	Dong & Rekatsinas	2018	SIGMOD Tutorial	Domain background*	Trigger
DataSpeck	Rahman et al.	2026	ACM CHI	Trigger + Case Selection	Reviewer Task + Info
RTS	Chen et al.	2025	SIGMOD	Trigger	Reviewer Task
CPFilter	Guo et al.	2023	SIGMOD	Trigger + Budget	Info Shown
Scrabble	Koh et al.	2018	ACM BuildSys	Case Selection + Feedback	Trigger
ALFA	Meduri et al.	2024	VLDB Journal	Case Selection	Feedback
DIAL	Jain et al.	2022	PVLDB	Case Selection	—
CopycHats	Solomon et al.	2024	HILDA@SIGMOD	Reviewer Task + Info	Case Selection
Batman & Robin	Bossen & Pine	2023	ACM TOCHI	Reviewer Task + Info	—
Balancing Privacy	Ragan et al.	2018	ACM CHI	Info Shown	Reviewer Task
Datalignment	Deutch et al.	2019	PVLDB	Info Shown	Feedback
DaME	Asthana & Mahindru	2022	ACM ICAIF	Feedback	Trigger
LLM Harmonized	Singh et al.	2026	KBS	Feedback	Trigger
Enhancing Construction Data Integration	Martin et al.	2025	EC3/CIB W78	Feedback + Quality	Trigger
MExI	Shraga et al.	2021	IEEE ICDE	Reviewer Quality	—
What Type	Shraga et al.	2018	HILDA@SIGMOD	Reviewer Task	Reviewer Quality

Table 3.3: Included papers by thematic cluster. * Not assigned to any of the four synthesis clusters; discussed in the preamble to Section 3.3.

3.3 Thematic Synthesis

One paper is primarily domain background rather than a supervision contribution: the SIGMOD tutorial on data integration [9]. Its five P codings reflect survey-level acknowledgement; it frames supervision dimensions as open research problems rather than addressing them architecturally. It is cited where it establishes domain context. A BERT-based schema matching study [5] is also cited for domain context; it was excluded from the analytical corpus because it presents top-k match suggestions for 'automated or human-in-the-loop alignment' without specifying or studying that workflow, satisfying EC4. Figure 3.2 shows the distribution of papers across primary cluster assignments.

3.3.1 Supervision Triggers, Reviewer Budget, and Case Selection

Training time versus inference time

Most papers in the corpus address training-time label acquisition. ALFA [27], DIAL [22], and Scrabble [23] all build active learning loops that terminate once the model converges; none addresses

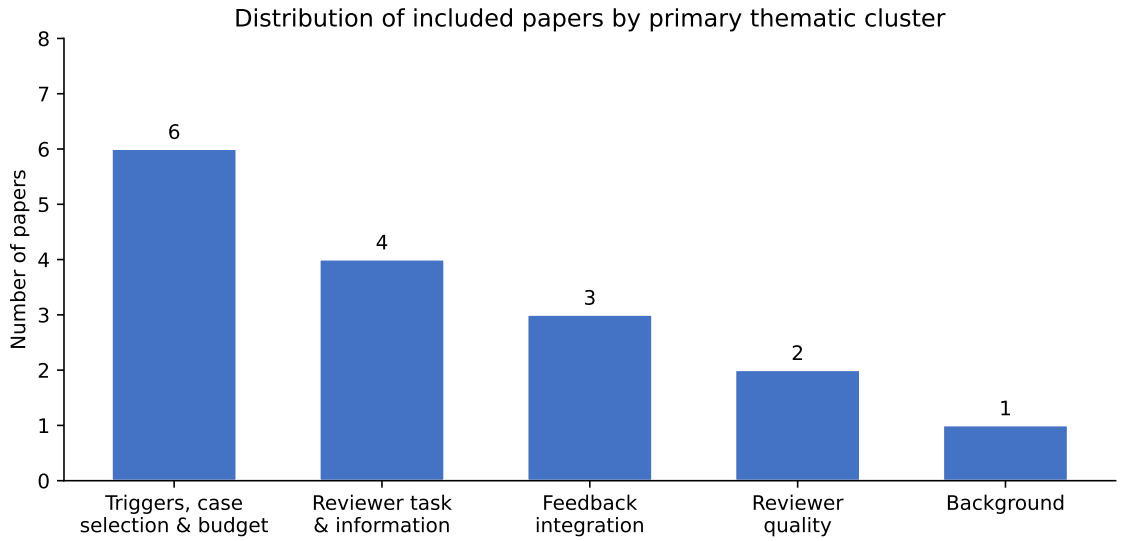

Figure 3.2: Distribution of papers across primary cluster assignments.

uncertain predictions in a deployed system. The SIGMOD Survey treats routing as a future direction rather than an addressable problem [9]. Only three papers in the corpus treat inference-time routing of uncertain decisions as an architectural concern: RTS [4], CPFilter [14], and DataSpeck [35].

Principled trigger design

The most rigorous trigger mechanism in the corpus is RTS's Binary Prediction Propagation (BPP), which uses conformal prediction on internal model states to guarantee that the correct answer is within the predicted confidence set with probability at least $1-2\alpha$, where α is the operator-configurable error level [4]. At $\alpha = 0.1$ on the BIRD benchmark, BPP achieves an AUC of 97–98% for table identification, with a true abstention rate of 19.10% and a false abstention rate of 12.77%. The statistical guarantee changes the trigger from an engineering tuning decision into a principled coverage commitment: the precision-recall trade-off is made explicit and bounded.

CPFilter takes a more operationally grounded approach: an explicit precision/recall/Workload Reduction Rate (WRR) trade-off curve allows threshold T to be set against a workload objective rather than a classification metric [14]. At threshold 0.53, it achieves precision 0.35, recall 0.84, and WRR 0.81 in benchmark evaluation; in production at Meltwater and Owler, WRR reaches 0.86. Multi-signal confidence aggregation (combining heterogeneous evidence types into a single routing score rather than relying on a single model output) is the pattern in deployed trigger systems: CPFilter, DaME [2], and Martin et al. [26] each aggregate three or four signal types; none found a single signal that reliably distinguishes reviewable from auto-executable cases.

DataSpeck extends binary threshold routing to a three-tier model: Confident (auto-execute), Assuming (request confirmation), and Insufficient (flag as non-derivable) [35]. Each tier carries a defined response protocol, which removes the ambiguity of a binary threshold about how to handle flagged cases. In isolated conditions, the Confidence Evaluator correctly assigned 93% of cases

to the Confident tier; this dropped to 86% in real-world conditions: 14% of cases were routed to auto-execution when they should have been flagged.

Reviewer Budget as design constraint

Trigger design and reviewer budget are coupled constraints: a threshold cannot be set appropriately without knowing the reviewer pool's processing capacity. CPFilter's WRR is the only operational metric in the corpus that explicitly models reviewer attention as a finite resource. It rewards threshold choices that eliminate more false positives while retaining true positives, and measures workload consequence rather than classification accuracy. No other paper in the corpus introduces an equivalent metric or treats session capacity as a primary design constraint.

Case selection: Prioritising within the uncertain set

Within the training-time framework, combining model uncertainty with structural diversity consistently outperforms single-signal strategies: DIAL, ALFA, and Scrabble each demonstrate 20–82% labelling cost reductions over plain uncertainty sampling [22, 23, 27]. CopycHats [43] establishes that presentation order matters independently of uncertainty scores: earlier decisions prime later ones in schema matching, so ordering is not neutral even when all flagged cases share the same uncertainty.

DataSpeck is the only inference-time case selection architecture in the corpus. Its Confidence Evaluator assigns every attribute to exactly one of three tiers and resolves the routing question for each case. DataSpeck does not, however, impose an ordering policy within the Assuming tier when session capacity is limited. No paper in the corpus addresses the prioritisation of a batch of uncertain field mappings at inference time under a finite reviewer budget.

3.3.2 Reviewer Task Design and Information Provision

The cognitive baseline: what human reviewers actually do

What Type deployed a controlled experiment with 100 human schema matchers and found that reviewers systematically over-report confidence: 49% of participants avoided low-confidence correspondences entirely, and 20 out of 100 produced F-1 scores at or below 0.2 [40]. Batman & Robin [3] provides complementary evidence from 48 hours of ethnographic observation of clinical coding specialists: in the absence of system-side routing, reviewers developed their own error taxonomies to compensate for the system's predictable failure modes. Reviewer quality is highly variable and undetectable from self-reported confidence alone; routing decisions cannot be delegated to the reviewers themselves.

What information schema matchers draw on

CopycHats [43] identifies five cue types that AI agents emulating human schema matchers consistently draw on: attribute name, data type, example instances, hierarchy position, and sibling

concepts. Hierarchy appeared most often; 73% of agent explanations referenced at least one of the five types. These cues have direct consequences for the AS reviewer interface: field name, data type, instance values, ontology tree position, and sibling concepts are the signals matchers demonstrably use.

A complication arises from reviewer defaults. What Type [40] establishes that syntactic name similarity is the most common coordination point, yet it is precisely the signal most likely to fail in flagged cases, where name similarity alone was insufficient to meet the confidence threshold. Explicit hierarchy and instance cues substitute for the heuristic that fails where review is most needed.

Provenance Display

Batman & Robin [3] found that a feature that let reviewers examine the text segments driving a code suggestion was essential for effective collaboration; without it, reviewers could not determine whether to trust the output. DataSpeck [35] reached the same conclusion from a different direction: step-by-step natural language descriptions of pipeline operations enabled 12-participant reviewers to verify correctness without reading raw code [35, Sec. 6]. Datalignment [8] treats provenance expansion trees as the primary interface rather than a supplementary display. RTS [4] and LLM Harmonized [42] both provide targeted evidence channels in more limited form. Across all five systems, evidence provision is a designed architectural feature, not a usability enhancement.

How much information is enough

Balancing Privacy [33] compared five conditions of information exposure in a record linkage task with 104 participants and found that 30% character exposure with structured markup produced accuracy statistically equivalent to full disclosure (both 84.1%). Only the low (8%) and fully masked conditions performed significantly worse ($p < 0.05$). Curated, targeted evidence is as useful as complete exposure. The same experiment found that structured markup significantly reduced reviewer overconfidence ($\chi^2(3) = 7.99,\ p = 0.005$), which is precisely the calibration improvement most absent from the What Type baseline. Batman & Robin adds that predictable error patterns contribute more to effective collaboration than higher but unpredictable accuracy: CDIS could identify recurring error categories and compensate for them, whereas opaque accuracy provides no such anchor.

Reviewer agency

Batman & Robin [3] found that reviewers had distinct working styles and that the system's value depended on supporting both rather than imposing a fixed review sequence. One CDIS stated explicitly: 'I have to make the decision. ... You do have to have that knowledge' [3, Sec. 4.2.3]. Reviewer agency (the ability to override suggestions, work in a self-determined order, and apply domain judgement without interface penalty) is a consistently observed property of effective supervision in knowledge work.

3.3.3 Feedback Integration and Correction Reuse

The structural absence of feedback loops

Most supervision architectures in the corpus treat corrections as terminal events. Batman & Robin [3] documents this in sustained production use since 2016: the system's error patterns remained fixed regardless of accumulated reviewer corrections. Datalignment [8] allows session-local corrections but has no mechanism to reduce the probability of the same error in subsequent independent runs. Seven of 16 papers provide no feedback mechanism at all, a pattern whose most natural explanation is the inherited assumption that learning and supervision are separate concerns.

Three feedback models with distinct properties

Three papers achieve full feedback loop coverage. Scrabble [23] implements an active learning loop: expert labels are incorporated after each round, the model is retrained, and the loop terminates when utilisation scores across all data points exceed a threshold. LLM Harmonized [42] uses a convergence-driven fine-tuning process that logs each step in a Transformation Matrix Database (original key, transformed key, validation outcome, retry count) and withdraws human involvement once the system reaches stable outputs. Martin et al. [26] integrate feedback directly into the confidence scoring function: S_{hitl} starts at 0.5 and converges toward the reviewer's correct-to-total validation ratio as decisions accumulate. DaME [2] provides the most complete correction taxonomy (six categories, each mapped to a distinct downstream remediation action) but does not specify how Incorrect-classified corrections are incorporated into model retraining.

Absence of Feedback governance

Even in papers that address feedback integration, the retraining cycle itself remains ungoverned. No paper in the corpus specifies minimum correction batch sizes before retraining is triggered, pre-promotion validation gates for the retrained model, or the partitioning of corrections by target standard to prevent cross-standard training contamination. The loop is present in parts; governed retraining (batch size thresholds, pre-promotion validation, standard-partitioned correction queues) is absent.

3.3.4 Reviewer Quality and Expertise Assessment

The quality distribution problem

The supervision architectures in Sections 3.3.1 and 3.3.3 treat reviewer competence as a design given. What Type [40] establishes that this assumption is wrong: 20 out of 100 matchers produced F-1 scores at or below 0.2 in controlled conditions. The standard pattern in the corpus is role-based competence designation: competence is declared by function rather than verified by performance, with DaME [2], Scrabble [23], and CPFilter [14] each treating "domain expert" as a sufficient designation. Balancing Privacy [33] is a partial exception: structured markup significantly reduced

reviewer overconfidence, but as a consequence of information design rather than a targeted quality measurement.

Self-reported confidence is the wrong proxy

MExI [39] empirically tests this proxy against four formally defined quality dimensions: precision, thoroughness, correlation, and calibration. Across 140 human matchers and 7,716 decisions, behavioural measurement outperforms self-reported confidence on all four metrics [39, Sec. IV-D]. The direction of failure is informative: 84% of under-confident matchers are precise, compared to 53% of the overall population. A system that routes uncertain cases to its most confident available reviewer will, on average, route them to a less reliable one.

A formal quality characterisation framework

MExI has the most comprehensive quality characterisation framework in the corpus. A model trained on the first 30 decisions of a single session already outperforms all baselines across all four quality dimensions [39, Fig. 11, Sec. IV-F]. A supervision system can therefore begin adjusting case routing on the basis of early reviewer behaviour, before the session ends. MExI's transferability finding extends this: a quality model trained on purchase order schema matching successfully characterises reviewer quality on OAEI ontology alignment tasks. This confirms cross-domain applicability within the AS context.

The most advanced quality integration in the corpus

Martin et al. [26] record each validation alongside a reviewer's domain expertise level and assign higher confidence to matches consistently endorsed by high-expertise users. However, expertise level is an unverified attribute, attached to each reviewer without a defined measurement procedure. The distinction between MExI (which derives reviewer quality from observed session behaviour) and Martin et al. (which applies differential weighting to an attribute of unknown provenance) is the difference between a measurement instrument and a design aspiration.

Operational consequences of quality-blind routing

No system in the corpus dynamically routes uncertain cases to reviewers based on measured quality. MExI identifies matching experts from behavioural profiles but does not propose a routing trigger or case-assignment architecture. Martin et al. weights incoming feedback by expertise but does not direct uncertain cases to reviewers selected by quality; assignment is user-initiated, not system-driven. In an AS context, quality-blind routing assigns uncertain mappings to imprecise reviewers without discrimination, and their incorrect approvals enter the training data silently. Martin et al. establish that reviewer accuracy is domain-specific: 95% in standardised domains, 74.1% in less uniform ones; a reviewer reliable for SAREF field mappings may not be reliable for IEC 61850.

3.4 Industrial and Commercial Platforms

The systematic protocol in Section 3.2 indexes peer-reviewed venues (IC4) and excludes commercial software by construction. Several industrial platforms nonetheless implement AS-relevant supervision mechanisms and belong in a state-of-the-art account. They are surveyed here from vendor and grey-literature sources for landscape positioning rather than as coded evidence, and are excluded from Table 3.2.

Two product categories overlap with the AS supervision problem. Machine-learning data-mastering and entity-resolution platforms route uncertain decisions to human curators and learn from the result: Tamr scores candidate mappings by attribute similarity and presents confidence-ranked suggestions to a curator whose accept and reject decisions retrain the model [45], while the open-source Zingg builds the same active-learning loop from a small labelled sample [47]. Data-quality and governance platforms implement the oversight and approval side: Informatica's CLAIRE engine lets data stewards review, approve, or override automated decisions [21], and Collibra routes governance artefacts through role-based approval workflows with task assignment, decision logging, and asset versioning [7].

These platforms implement the individual primitives, but none closes the three gaps, alone or in combination. Each resolves or masters records against an internal canonical schema; none supervises translation onto external formal standards such as SAREF, CIM, or IEC 61850, and none carries the standard-specific ontological context an AS reviewer needs. Profile versioning governs a stored artefact rather than a retraining cycle partitioned by target standard (Gap 1); stewardship runs as a continuous queue rather than a bounded review session with within-session prioritisation (Gap 2); approval routing is role-based rather than calibrated to measured reviewer quality (Gap 3). These are the commercial off-the-shelf solutions weighed by Test 3 of the gap filter in Section 3.5.2.

3.5 Coverage Map and Identified Gaps

3.5.1 Coverage hierarchy

The seven substantive dimensions in Table 3.2, ranked from most to least covered: **Reviewer task design and information provision** (6F/9P/1- and 4F/8P/4-) is the best-covered, but evidence is concentrated in clinical coding, data transformation and record linkage; schema-matching systems largely leave the reviewer interface unspecified. **Supervision triggers** (5F/5P/6-) are partially covered; only RTS, CPFilter, and DataSpeck address inference-time routing. **Case selection** (4F/5P/7-) follows the same pattern: three of four fully-covered papers are training-time systems; DataSpeck routes cases across tiers but imposes no within-tier ordering. **Reviewer quality** (1F/10P/5-) is weakly covered despite high partial-coding counts: nine papers use role-based designation; only Martin et al. introduces weighting, without measurement. **Feedback integration** (3F/6P/7-) is a major absence: none of the three fully-covered papers governs the retraining cycle as an architectural concern. **Reviewer budget management** (1F/7P/8-) is the most severely absent dimension; CPFilter's WRR is the sole operational budget model.

3.5.2 Three-test gap filter

The three tests applied to each candidate gap are: (1) the specific architectural capability is absent from the corpus; (2) it is specifically required by the AS operational context; (3) it cannot be addressed by combining existing off-the-shelf solutions, including the commercial platforms in Section 3.4.

Candidate Gap 1: Feedback integration and retraining governance.

Test 1: governed retraining (batch size thresholds, pre-promotion validation, standard-partitioned correction queues) is absent from all 16 papers; the feedback-loop dimension records seven absent codings. Test 2: in the NOUS AS context, reviewer corrections are directly training data for the field mapping model; standard partitioning of corrections is an AS-specific requirement not addressed anywhere in the corpus. Test 3: Scrabble, Martin et al., and LLM Harmonized address components of the loop; no paper governs the retraining cycle as an architectural concern requiring the combination of all four properties.

Candidate Gap 2: Inference-time batch prioritisation under a finite reviewer budget.

Test 1: principled inference-time batch prioritisation is absent from all 16 papers; the reviewer-budget dimension records eight absent codings. Test 2: every standardisation job produces a batch of flagged mappings that a PPC Data Engineer reviews in a finite session; the order and deferral policy for that batch are entirely unaddressed. Test 3: CPFilter supports threshold tuning to control queue size but is not a session manager; training-time strategies (ALFA, DIAL, Scrabble) optimise for model improvement per label; DataSpeck resolves tier assignment but imposes no ordering within the Assuming tier. None addresses within-session ordering of simultaneously flagged uncertain mappings at inference time.

Candidate Gap 3: Reviewer quality assessment integrated with dynamic routing

Test 1: dynamic quality-based routing is absent from all 16 papers; nine partially-coded papers address quality exclusively through role designation or self-reported confidence, which MExI empirically shows is inferior to behavioural measurement on all four quality dimensions. Test 2: PPC Data Engineers have domain expertise that varies across SAREF, CIM, and IEC 61850; routing uncertain mappings without considering ontology-specific competence risks incorrect approvals that degrade training data silently. Test 3: MExI provides quality characterisation but proposes no routing architecture; Martin et al. provides expertise weighting without measurement. Integrating quality measurement with dynamic case routing requires architectural work that existing components do not provide individually or in combination.

Table 3.4: Three-test gap filter results

Candidate gap	**Test 1**	**Test 2**	**Test 3**
Gap 1: Feedback integration and retraining governance	Met	Met	Met
Gap 2: Inference-time batch prioritisation	Met	Met	Met
Gap 3: Reviewer quality assessment and dynamic routing	Met	Met	Met

3.6 Summary

This chapter surveyed how existing supervision architectures address the design concerns of automated data standardisation. A PRISMA-guided review of sixteen papers, coded against eight supervision dimensions and synthesised through four thematic clusters, produced the coverage map and the three-test gap filter of Section 3.5.

Partial contributions exist in the corpus, but no complete architectural solution covers any of them in the AS context. In the literature, the second gap is labelled as a missing ordering policy; in the AS operational context it is a deeper absence: the session model itself does not exist in any reviewed architecture.

The three gaps all pass the filter (Table 3.4) and represent genuine structural absences relative to the AS context, but Gap 2 is architecturally foundational in a way the others are not: without a session model, neither the routing trigger from Gap 1 nor the quality-based assignment from Gap 3 has a unit of work to operate on. That interdependence is the review's deepest finding: the corpus offers no architecture that integrates supervision for confidence-scored, batch-oriented standardisation, so the three gaps are not detachable problems to be solved separately but facets of one design problem. Gap 2 is also the most immediately tractable. Gap 1 requires a live ML training pipeline to govern retraining cycles; Gap 3 requires a corpus of historical reviewer performance data from which to calibrate quality-based routing. Neither exists in the NOUS prototype context. Chapter 4 carries all three gaps forward into the requirements analysis, but the implementation boundary follows from these constraints: the session model and its derived requirements form the core implementation target of this thesis, while Gaps 1 and 3 are architecture-specified without implementation.

Chapter 4

Requirements Analysis

4.1 Introduction

This chapter completes Stages 1 and 2 of Peffers et al.'s [32] Design Science Research process (problem identification and definition of objectives) applied to the NOUS Auto-Standardiser context. Four sources of evidence inform the analysis: the task development plan and the use case's Story 1.1 acceptance criteria; two semi-structured practitioner interviews; the systematic literature review in Chapter 3; and Maldonado's NOUS HITL framework [25] as the intra-project baseline.

4.2 Requirements Derivation Approach

Task documentation and use-case acceptance criteria capture what the supervision layer is formally required to do. The task development plan specifies the machine learning architecture for the human-in-the-loop standardisation workflow [24]; the use case's Story 1.1 defines eight acceptance criteria (AC1–AC8) that the system must satisfy within the NOUS energy data standardisation use case [30]. Every clause bearing on routing conditions, reviewer actions, feedback capture, profile governance, audit obligations, and rollback capability was extracted and coded against the requirement categories from Chapter 3. Acceptance criteria governing AS-internal behaviour rather than the supervision layer were noted but not coded into supervision requirements; the clearest such case, the AC4 pre-validation of input data before mapping (SR-F-11, defined in Section 4.4), is excluded because it governs algorithm inputs rather than the supervision layer.

Two semi-structured practitioner interviews were conducted to capture knowledge that documentary sources cannot encode: how practitioners would work with a deployed AS review interface, what information they need under uncertainty, and what conventions govern escalation. Participant 1 is a NOUS project engineer with direct familiarity with the AS platform; Participant 2 is a researcher with expertise in annotation workflow design and multi-fidelity human–AI systems. Both interviews followed the interview protocol (Appendix A), a semi-structured procedure organised across five topic areas. Participant 2 requested anonymisation.

Chapter 3's systematic literature review contributes coverage evidence for the gap analysis rather than requirements: the PRISMA-guided review of sixteen papers established which supervision dimensions are addressed, partially addressed, or absent in the existing literature.

Maldonado's NOUS HITL framework [25] provides the intra-project baseline. It is grey literature (a FEUP master's dissertation, not peer-reviewed) and provides architectural context rather than scientific authority for design decisions.

Requirements derived from these sources are referenced by identifier throughout this chapter, with functional requirements prefixed SR-F and non-functional requirements prefixed SR-NF. Full specifications appear in Section 4.4.

4.3 Practitioner Perspectives

Thematic coding of both interview transcripts, conducted by a single researcher, identified six themes. The single-coder basis of this analysis is a reliability threat addressed among the threats to validity in Chapter 7.

Theme 1: Supervision intensity as a configurable cost–quality trade-off

Both participants treated cost awareness as a first-class design constraint. Participant 1 stated directly that 'having someone observing and evaluating is an expensive thing to do' and endorsed a transition to exception-based autonomy once a pipeline had established a performance record. Participant 2 framed the same constraint through a multi-fidelity model: human review is the highest-cost, highest-accuracy tier; automated surrogates are the lowest cost at a corresponding accuracy penalty. Both framings point to the same design requirement: supervision intensity must be configurable at the system level, embedded as a structural property rather than left as an operational convention (SR-NF-03, SR-NF-06).

Theme 2: Two models for reviewer authority and escalation

The participants proposed structurally different escalation models, each addressing a distinct level of the architecture.

The first model is horizontal: multiple reviewers assess the same mapping simultaneously, with each decision weighted by expertise and role authority (Participant 1). When the group cannot agree, disagreement re-triggers the mapping process rather than forcing a decision through. A designated tiebreaker role can resolve deadlocks in case-dependent situations.

The second model is sequential, grounded in inter-annotator agreement protocols from corpus linguistics (Participant 2): (1) if the algorithm's confidence exceeds a configurable threshold, auto-accept; (2) if below that threshold, route to the junior reviewer pool; (3) if inter-reviewer disagreement among juniors exceeds a second threshold, escalate to a domain expert. The budget rationale is explicit: scarce expert time should only be consumed when cheaper tiers cannot reach a decision.

The two models address different levels of the same architecture: the horizontal model governs authority within a tier; the sequential ladder governs how cases move between tiers. SR-F-12 combines both.

Theme 3: Review interface (priority sorting, comparative confidence, and information density)

Both participants, working independently, settled on a confidence-sorted priority list as the primary review interface. They diverged on candidate count: Participant 1 considered top-3 sufficient; Participant 2 preferred top-5, arguing that comparative confidence across candidates matters more than the absolute score of the primary match. Participant 2 also identified a distinction absent from T5.4/UC2: 'high confidence that a column should not be mapped' is categorically different from 'low confidence that a mapping exists'; the two require different reviewer responses and different supporting information. Both participants endorsed colour-based confidence tiers with text labels or icons to avoid excluding reviewers with colour vision deficiency, grounding SR-NF-07.

Theme 4: The never-fully-autonomous floor

This is the sharpest divergence. Participant 1 endorsed a mature exception-only mode: once a pipeline has performed reliably, it should proceed autonomously and surface only anomalies. Participant 2 explicitly rejected this: 'I never believe in a fully autonomous thing, even in a sampling manner… because sometimes the semantics of the schema might change in the real world. No one knows this because the field name remains the same.'

Semantic drift (where a field name remains stable while its real-world meaning changes) is not visible to confidence-score monitoring: the model has no signal that field meanings have shifted since training, because its confidence is computed over the unchanged field name. Within the supervision layer, periodic spot-checking of high-confidence outputs is the only available signal for this failure mode. The conservative position holds: the consequence of undetected semantic drift (silently degraded mappings propagating into AS training data) is more severe than the cost of maintaining a minimum sampling floor. This grounds SR-NF-06, which adopts the configurable-rate sampling pattern of RTS [4]. The requirement is sourced to a single participant; its design treatment and independent grounding are developed in Chapter 5.

Theme 5: Algorithm steering for unmapped columns

Participant 2 introduced a capability with no equivalent in any of the sixteen reviewed papers and not derivable from T5.4/UC2: for columns the system cannot map, the reviewer should be able to direct the algorithm to try a different matching approach. The motivating failure mode is domain-specific: column names in non-English languages (Greek abbreviations in PPC solar park datasets) cause matching failures that a human reviewer can diagnose but the algorithm cannot detect autonomously. This is a process directive, not an output correction: correction capture (SR-F-04) records the right answer to a wrong output; algorithm steering records the right approach

for a systematically unmatchable input type. SR-F-13 originated in the practitioner interviews; no reviewed paper contains an equivalent.

Theme 6: Deferred review for non-blocking uncertainty

Both participants endorsed routing uncertain mappings forward with a provisional flag rather than blocking the pipeline, provided the deferred decision resurfaces at the next review opportunity. Participant 1 framed the trade-off in cost terms: blocking the pipeline on every uncertain mapping is disproportionate when the downstream criticality is low. Participant 2 emphasised the safeguard side: a provisionally accepted mapping must never be silently promoted. SR-F-14 combines both elements through a provisional state with explicit confirmation gating.

4.4 Supervision Requirements Specification

Five challenge codes mark the structural property of the AS workflow that makes each requirement domain-specific, not reducible to a generic HITL concern.

Code	Name	Applies when
C-RT	Confidence-based Routing	Requirements driven by the numeric semantic similarity score per field pair
C-BW	Batch Workflow	Requirements arising from review sessions over complete job outputs
C-CR	Correction-driven Retraining	Reviewer corrections directly enter the ML training pipeline
C-PG	Mapping Profile Governance	Requirements concerning versioned mapping sets as organisational artifacts
C-ME	Multi-Standard Explainability	Requirements arising from contextualising suggestions differently across SAREF, CIM, and IEC 61850

Table 4.1: Challenge codes: AS-specific structural properties that make each requirement domain-specific, not reducible to a generic HITL concern.

Twenty requirements are specified: 13 functional (Section 4.4.1) and 7 non-functional (Section 4.4.2). SR-F-11 corresponds to UC2 Story 1.1 AC4, the automatic pre-validation check the AS performs on input data before mapping: data-type checking and unit normalisation, for example converting and labelling all measurements to a common unit such as kW [30]. It is excluded because it operates on algorithm inputs upstream of any mapping output, so no human-supervision decision attaches to it; it is a precondition for the supervision workflow rather than a property of it.

4.4.1 Functional Requirements

Confidence-Based Routing (C-RT)

SR-F-01 — Confidence-Threshold-Based Mapping Routing. The system must route individual field mappings whose confidence score falls below a configurable threshold to a human reviewer

for correction. High-confidence mappings must proceed automatically without review. The routing threshold must be calibrated against the domain-specific Mapping Quality Threshold defined in UC2 (column coverage ≥80% of required fields; correct mapping accuracy ≥80%), not against generic model performance metrics.

Source: T5.4 Sec. 2.3 (Machine Learning for Human-in-the-Loop Standardisation); UC2 Story 1.1 AC2.

AS routing is driven by a continuous numeric semantic similarity score over ontological field name pairs, not a policy-based trigger. Calibrating the threshold against the UC2 Mapping Quality Threshold rather than cross-domain benchmarks is an AS-specific constraint absent from the reviewed literature.

SR-F-12 — Tiered Escalation Routing (C-RT + C-BW). The system must support a configurable three-tier sequential escalation ladder: (1) mappings above a first configurable confidence threshold proceed automatically; (2) mappings below that threshold are routed to a junior reviewer pool; (3) when disagreement among junior reviewers exceeds a second configurable threshold, the case escalates to a domain expert. Each tier's threshold must be independently configurable.

Source: Participant 1, personal communication, 6 April 2026, Q8, Q9, Q27; Participant 2, personal communication, 21 April 2026, Q7, Q8.

SR-F-12 is a synthesised requirement rather than a single elicited proposal: it combines the two structurally different escalation models the participants advanced (Section 4.3, Theme 2). It adopts Participant 2's sequential ladder as its backbone and reflects Participant 1's pooled-reviewer model at the junior tier through the disagreement-triggered escalation in step (3). Participant 1's expertise-based weighting and designated tiebreaker mechanisms are not encoded in SR-F-12; they are treated as policy details internal to the junior tier rather than as routing-ladder properties. This vertical sequential structure differs from the horizontal consensus models that appear in existing HITL architectures.

Batch Workflow (C-BW)

SR-F-02 — Batch Review Session Management. The system must present all flagged mappings produced by a single standardisation job to the reviewer as a cohesive review session. The reviewer must be able to navigate the flagged set as a whole, maintain context across mappings within the batch, and submit the session outcome as a unit. The system must track session completion state, including the count of reviewed and unreviewed mappings within each session.

Source: T5.4 Sec. 3.4 (Supervised ML Component); UC2 Story 1.1 AC5.

A PPC Data Engineer reviewing mappings for a solar park database ingestion brings contextual knowledge that applies across all flagged mappings simultaneously. Individual-item review queues, the dominant model in existing HITL architectures, fragment this context by presenting each mapping in isolation. The session model has no structural equivalent in the reviewed literature, in Maldonado's framework, or in the commercial stewardship platforms surveyed in Section 3.4; the gap analysis in Section 4.5 develops this comparison.

SR-F-14 — Provisional Mapping Acceptance. The system must support a provisional mapping state: uncertain mappings may proceed downstream with a provisional flag without blocking the pipeline. The system must track all provisionally accepted mappings, surface them in the next available review session, and prevent them from being promoted to permanent status without explicit reviewer confirmation.

Source: Participant 1, personal communication, 6 April 2026, Q13; Participant 2, personal communication, 21 April 2026, Q14.

Both participants endorsed deferred review where blocking the pipeline would be disproportionate to a mapping's criticality (Section 4.3, Theme 6); the confirmation gate must hold regardless of session volume.

Multi-Standard Explainability (C-ME)

SR-F-03 — Standard-Aware Explainability for Mapping Suggestions. For each flagged mapping, the review interface must provide contextual information specific to the target standard: the standard field's formal definition, its position in the standard's ontological hierarchy, and the ranked alternative candidates that were considered but scored lower. The contextual information must differ per standard (SAREF, CIM, IEC 61850).

Source: T5.4 Sec. 2.1 (standards exploration); UC2 Story 1.1 AC3 (multi-standard support: SAREF, CIM, IEC 61850), AC5 (review interface).

UC2 AC3 and AC5 mandate multi-standard support and a review interface; the specific contextual fields above (ontological hierarchy position and ranked lower-scored alternatives) are an analyst elaboration of what an effective interface for that mandate requires, informed by the practitioner perspectives in Section 4.3. A reviewer comparing `voltage_phase_a` to `saref:Property` needs to know what that concept means in the SAREF hierarchy, whether `saref:Measurement` would be more appropriate, and how candidates ranked. Generic model explainability (SHAP values, counterfactual explanations) explains confidence scores, not ontological meaning. Multi-standard contextual display is qualitatively different from what any existing HITL architecture provides.

Correction-Driven Retraining (C-CR)

SR-F-04 — Structured Correction Capture. The system must capture human corrections as structured tuples preserving: (a) the source field identifier, (b) the target standard, (c) the algorithm-suggested mapping that was rejected, (d) the human-approved correct mapping, (e) the AS model version that produced the suggestion, (f) the reviewer identity and timestamp, and (g) the originating job identifier.

Source: T5.4 Sec. 3.4 (feedback capture during translation); UC2 Story 1.1 AC8 (system learns from manual corrections).

Corrections in AS are training examples for the semantic similarity model. Without the target standard, model version, and job identifier, a correction cannot be safely used for retraining. SAREF

corrections must not pool with CIM corrections; their training sets are standard-partitioned by design.

SR-F-05 — Standard-Partitioned ML Retraining Trigger. Accumulated corrections must trigger retraining of the AS translation model. Retraining must be partitioned by target standard: corrections for SAREF feed only the SAREF matching model; corrections for CIM feed only the CIM model. A minimum correction batch per standard must be reached before triggering retraining. The retrained model must be validated against a held-out test set before promotion to production.

Source: T5.4 Sec. 2.3, Sec. 3.4; UC2 Story 1.1 AC6, AC8.

Pooling corrections across standards degrades training signal quality. The minimum batch size prevents premature retraining from a small number of noisy corrections. The pre-promotion validation gate prevents silent degradation of the production model before errors become detectable from downstream output quality.

SR-F-13 — Algorithm Steering for Unmapped Columns (C-CR + C-ME). For columns the system cannot map, the reviewer must be able to provide a structured process directive to the matching algorithm, specifying not merely the correct output but the approach that should produce it. Examples include: apply translation or transliteration to the field name before re-matching; restrict candidate search to a specified ontology sub-tree; treat the field as a measurement concept and search accordingly. Logged directives must feed back as retraining signals.

Source: Participant 2, personal communication, 21 April 2026, Q10, Q19.

This requirement has no equivalent in the reviewed HITL literature and arises from an AS-domain-specific failure mode (Section 4.3, Theme 5): columns the AS fails to map automatically often have a diagnosable cause (non-English column names, non-standard unit encodings, vendor-specific naming conventions) that a human reviewer can identify but the algorithm cannot. Without a steering channel the column has no automated route to a mapping at all: correction capture (SR-F-04) records a one-off answer for a single column but leaves the systematic cause unaddressed, so the same class of column fails again on the next job. The current AS architecture provides no mechanism to receive structured process guidance of this kind.

Mapping Profile Governance (C-PG)

SR-F-06 — Mapping Profile Versioning. Each set of approved field mappings produced by a standardisation job must be stored as a named, versioned mapping profile. Each version must record: the profile identifier, version number, creation date, approving user identity, the AS model version that produced the mappings, and the source database schema version. Previous profile versions must remain queryable.

Source: UC2 Story 1.1 AC6 (stores and versions approved mapping profiles for future reuse and audit).

A mapping profile is a reusable organisational asset governing how a PPC database is standardised for ML training. Its lifecycle (creation, review, approval, deployment, drift monitoring, rollback) is specific to data standardisation. Crisis management decisions are ephemeral events

processed and closed individually; mapping profiles are persistent governance artifacts versioned and reused across standardisation runs.

SR-F-07 — Mapping Profile Rollback. The system must support rollback of the active mapping profile for a database to any previously approved version. Rollback must be triggerable by the PPC Data Engineer and must take effect without requiring re-standardisation of the source database.

Source: UC2 Story 1.1 AC6 (rollback to any previously approved mapping profile on detection of performance degradation).

Rollback is a safety mechanism for the data pipeline. Mapping profile rollback is distinct from ML model rollback: the same profile may be re-applied with a different model version, and both operations must be independently supported.

SR-F-08 — Mapping Output Drift Detection. The system must continuously monitor whether the standardised output from ongoing jobs conforms to the approved mapping profile. When the output distribution diverges from the approved profile, the PPC Data Engineer must be alerted.

Source: UC2 Story 1.1 AC7 (drift detection mechanism alerting the Data Engineer when output no longer adheres to the approved mapping profile).

Mapping output drift is detected by comparing outputs against the approved profile; no labelled ground truth is required. This differs from model performance drift detection, which requires labels.

SR-F-09 — Role-Based Governance Approval for Profile Promotion. Promotion of a new mapping profile version to production, and promotion of a retrained AS model to production, must require explicit authorisation from the PPC Data Governance Lead. The system must enforce a two-role workflow: the PPC Data Engineer conducts the review and initiates promotion; the PPC Data Governance Lead approves or rejects the promotion. No single user identity may perform both steps for the same asset.

Source: UC2 Story 1.1 AC6; UC2 Story 2.2 AC7 (dual control authorisation); UC2 Actor definitions (PPC Data Governance Lead responsibility for approving critical data decisions).

A mapping error propagating into ML training data could, in principle, degrade the energy forecasting models PPC trains on that data. The governance approval workflow is an explicit organisational requirement embedded in UC2's actor model. (See SR-F-12 for the architectural contrast with horizontal consensus models.)

Cross-Cutting

SR-F-10 — Immutable Audit Trail for Mapping Decisions and Corrections. Every significant event in the supervision workflow must be recorded in an immutable, retrievable audit trail: mapping job initiation and completion; individual routing decisions; reviewer correction submissions; profile version creation and promotion events; model retraining triggers; drift alerts and their acknowledgements.

Source: UC2 Story 2.5 (immutable transaction records for critical datasets); UC2 Story 2.3 AC4 (SIEM logging); EU AI Act (2024), Annex III (audit trail requirements for high-risk AI systems).

The audit trail serves regulatory compliance (GDPR Art. 32, EU AI Act human oversight obligations for high-risk AI in energy infrastructure), quality assurance (tracing a mapping error to the specific correction that degraded the model), and organisational accountability. SR-F-10 is cross-cutting because events from C-RT, C-CR, and C-PG all produce audit-relevant records.

4.4.2 Non-Functional Requirements

Confidence-Based Routing (C-RT)

SR-NF-01 — Mapping Quality Threshold: Column Coverage. The supervision architecture must track column coverage per standardisation job (the fraction of required standard fields receiving a confident automatic or human-approved mapping) and alert the reviewer when this fraction falls below 80%.

Source: UC2 Story 1.1 AC2 (Column Coverage: at least 80% for required fields).

Column coverage measures the proportion of required standard fields successfully mapped, not model classification accuracy. A standardisation output with high prediction accuracy but insufficient column coverage is unusable for ML training: required features are absent. The 80% threshold is a UC2 acceptance criterion, not a generic quality heuristic.

SR-NF-02 — Mapping Quality Threshold: Accuracy via Sampling. On first standardisation of a database, the supervision architecture must surface a sample of high-confidence automatic mappings for reviewer spot-check, verifying that the correct mapping accuracy criterion ($\geq$80%) is met before the full batch is accepted.

Source: UC2 Story 1.1 AC2 (Correct Mapping Accuracy: at least 80% verified via initial sampling).

Generic HITL queues flag items by confidence threshold alone and include no structured sampling step for high-confidence outputs. The sampling gate guards against a model that is well-calibrated on its training distribution but has not been validated on a specific database's schema conventions, a scenario that cannot be ruled out on first standardisation of an unseen database.

SR-NF-06 — Minimum Oversight Floor. Regardless of pipeline maturity, the system must always route a minimum configurable sample of high-confidence automatic mappings for spot-check review. Full autonomy is not an acceptable operational state. The minimum sample rate must be operator-configurable but must not be settable to zero.

Source: Participant 2, personal communication, 21 April 2026, Q5.

This requirement follows Participant 2's never-fully-autonomous position (Section 4.3, Theme 4): semantic drift is not visible to confidence-score monitoring, so within the supervision layer periodic spot-checking of high-confidence outputs is the only available safeguard against silently degraded mappings entering AS training data. The requirement derives from a single participant; the design adopts the configurable-rate sampling pattern of RTS [4] to discharge it (Chapter 5).

Batch Workflow (C-BW)

SR-NF-03 — Reviewer Workload Manageability. The supervision architecture must limit reviewer fatigue by: presenting flagged mappings in priority order within a session (lowest confidence first, or by standard field importance); providing session progress indicators showing reviewed and unreviewed counts; and supporting pause and resume of sessions, with per-session review load limits that are configurable by the operator.

Source: UC2 Story 1.1 AC5 (review interface usability); T5.4 Sec. 3.6 (usability as evaluation KPI); Participant 2, personal communication, 21 April 2026, Q19.

A PPC Data Engineer may process hundreds of flagged mappings for a large database ingestion. Without session-level workload management, review quality degrades as fatigue accumulates. AS review sessions are scheduled work at a reviewer's discretion, not real-time crisis responses requiring immediate decision under pressure.

SR-NF-07 — Accessibility of the Review Interface. The review interface must not rely on colour alone as an information channel. All colour-coded elements must be supplemented with text labels, icons, or other non-colour indicators. The interface must comply with WCAG 2.1 AA guidelines as a minimum standard.

Source: Participant 2, personal communication, 21 April 2026, Q19; European Accessibility Act (2019/882/EU); EN 301 549.

Colour vision deficiency affects approximately 8% of males and 0.5% of females [38]. EU public procurement and accessibility legislation, including the European Accessibility Act and EN 301 549, applies to professional tooling deployed in the PPC energy infrastructure context.

Governance and Compliance (C-PG)

SR-NF-04 — Audit Trail Retention and GDPR Compliance. Audit records must be retained for the regulatory retention period applicable to PPC's energy data operations. Records must not contain raw personal data; hashed identifiers must be used in place of direct identifiers. Records must be retrievable within a defined service level for compliance verification.

Source: UC2 Story 2.5 AC5 (GDPR, NIS2, Data Act compliance); UC2 Story 2.3 AC6 (GDPR Art. 32 access log mapping).

SR-NF-04 operationalises the audit trail mandated by SR-F-10 within a regulatory compliance frame. The GDPR and NIS2 constraints are imposed by PPC's status as a critical energy infrastructure operator in an EU member state.

SR-NF-05 — Separation of Duties. The system must enforce that no single user identity can both submit a mapping review or correction and approve the same mapping profile version for production. The role boundary between PPC Data Engineer and PPC Data Governance Lead must be enforced at the architecture level, not only as a process convention.

Source: UC2 Story 2.2 AC7 (dual control authorisation for critical data transactions); UC2 Actor definitions (PPC Data Engineer $\neq$ PPC Data Governance Lead).

Architectural enforcement is required because a deployed system cannot rely on a single user identity voluntarily restraining itself from performing both steps. (See SR-F-12.)

Summary Table

Table 4.2 consolidates all twenty supervision requirements.

ID	Short Description	Challenge	Primary Source	Derivation
SR-F-01	Confidence-threshold routing	C-RT	T5.4 Sec. 2.3; UC2 AC2	T5.4/UC2
SR-F-02	Batch review session management	C-BW	UC2 AC5	UC2
SR-F-03	Standard-aware explainability	C-ME	T5.4 Sec. 2.1; UC2 AC3, AC5	T5.4/UC2
SR-F-04	Structured correction capture	C-CR	T5.4 Sec. 3.4; UC2 AC8	T5.4/UC2
SR-F-05	Standard-partitioned retraining	C-CR	T5.4 Sec. 2.3, Sec. 3.4; UC2 AC6, AC8	T5.4/UC2
SR-F-06	Mapping profile versioning	C-PG	UC2 AC6	T5.4/UC2
SR-F-07	Mapping profile rollback	C-PG	UC2 AC6	T5.4/UC2
SR-F-08	Mapping output drift detection	C-PG	UC2 AC7	T5.4/UC2
SR-F-09	Role-based governance approval	C-PG	UC2 AC6, AC7; UC2 Actor defs	T5.4/UC2
SR-F-10	Immutable audit trail	Cross-cutting	UC2 Story 2.5, 2.3; EU AI Act	T5.4/UC2 + regulatory
SR-F-12	Tiered escalation routing	C-RT + C-BW	P1 Q8, Q9, Q27; P2 Q7, Q8	Both interviews
SR-F-13	Algorithm steering / hinting	C-CR + C-ME	P2 Q10, Q19	Participant 2 interview
SR-F-14	Provisional mapping acceptance	C-BW	P1 Q13; P2 Q14	Both interviews
SR-NF-01	Column coverage $\geq$80%	C-RT	UC2 AC2	T5.4/UC2
SR-NF-02	Accuracy $\geq$80% via sampling	C-RT	UC2 AC2	T5.4/UC2
SR-NF-03	Reviewer workload manageability	C-BW	UC2 AC5; T5.4 Sec. 3.4; P2 Q19	T5.4/UC2 + interview
SR-NF-04	Audit retention and GDPR	Cross-cutting	UC2 Story 2.5, 2.3	T5.4/UC2 + regulatory
SR-NF-05	Separation of duties	C-PG	UC2 AC7; UC2 Actor defs	T5.4/UC2
SR-NF-06	Minimum oversight floor	C-RT	P2 Q5	Participant 2 interview
SR-NF-07	Accessibility of review interface	C-BW	P2 Q19; EU Accessibility Act	Interview + regulatory

Table 4.2: Supervision requirements: challenge classification, primary source, and derivation. P1 = Participant 1 (Panagiotis Krokidas, 6 April 2026); P2 = Participant 2 (anonymous, 21 April 2026).

4.5 Gap Analysis

The twenty requirements from Section 4.4 are mapped against two reference points: the sixteen-paper PRISMA corpus from Chapter 3, and Maldonado's NOUS HITL framework [25].

4.5.1 Coverage Against the Literature

Trigger and Routing. Five papers fully address the trigger dimension, but effective inference-time coverage is narrower: only RTS [4], CPFilter [14], and DataSpeck [35] route uncertain predictions in deployed operational systems. SR-F-01 (confidence-threshold routing) is partially addressed by all three; the AS-specific gap is threshold calibration against the UC2 Mapping Quality Threshold, absent from all sixteen papers. SR-F-12 (tiered escalation routing) and SR-NF-06 (minimum oversight floor) are absent from the corpus: no paper implements a sequential multi-tier escalation ladder, and no paper formalises a non-negotiable minimum sampling obligation.

Reviewer Task, Information Provision, and Session Management. The reviewer task dimension has the strongest coverage (seven fully addressed, eight partial), concentrated in clinical coding, data transformation, and record linkage. SR-F-03 (standard-aware explainability) is partially addressed: CopycHats [43] identifies the five cue types reviewers draw on, Balancing Privacy [33] demonstrates that curated signals achieve full-exposure accuracy, Batman and Robin [3] establish that the evidence panel is load-bearing. None addresses the standard-specific ontological context AS requires. SR-F-02 (batch review session management) is absent: no paper models a review session as a cohesive unit of work for all outputs of a single job. SR-NF-07 (accessibility) is entirely absent from the corpus.

Feedback Integration. Three papers fully address the feedback dimension; seven are entirely absent. SR-F-04 (structured correction capture) and SR-F-05 (standard-partitioned retraining) are partially addressed: DaME [2] provides the most complete correction taxonomy; LLM Harmonized [42] provides the most complete audit trail template. Standard-partitioned correction queues, minimum batch size thresholds, and pre-promotion validation gates are absent from all sixteen papers. SR-F-13 (algorithm steering) is absent from the entire corpus.

Governance and Audit. Mapping profile governance (versioning, rollback, and lifecycle management of approved mapping sets as organisational artifacts) falls outside the eight supervision dimensions coded in Chapter 3 because governance artifact management is not addressed anywhere in the reviewed literature. SR-F-06 (profile versioning) and SR-F-07 (profile rollback) are absent from all sixteen papers. SR-F-08 (drift detection) is partially covered by LLM Harmonized's convergence monitoring and CPFilter's production WRR monitoring, but detecting divergence from an approved profile without labelled ground truth is absent. SR-F-10 (immutable audit trail) and SR-NF-04 (GDPR compliance) are partially covered; regulatory compliance properties have no peer-reviewed precedent in the corpus and are grounded by EU AI Act Annex III and GDPR Articles 22 and 32.

Reviewer Quality. One paper fully addresses the reviewer quality dimension; nine partially. MExI [39] provides a validated four-dimensional characterisation framework; EC3 [26] provides

expertise-weighted authority integration. Neither connects quality measurement to dynamic case routing. SR-F-12's tiered escalation requirement is absent primarily because this integration work does not exist in any reviewed paper.

4.5.2 Coverage Against Prior NOUS Work

Maldonado's NOUS HITL framework [25] adds five HITL containers to the NOUS node: a Decision Guard for confidence-gated routing, an Expert Review Queue for task management, an Explainability Service for context provision, a Feedback Manager for correction ingestion and retraining, and a Dashboard and APIs layer for operator interaction. Three cross-cutting stores underpin them: a Decision Store (append-only log), a Feedback Store (versioned SME labels), and an external DLT Ledger (tamper-proof provenance chain). Maldonado concluded in Section 5.4.6 that 'the same mechanisms can be reused, with minor policy tuning, for the other NOUS pilot use cases.'

What transfers. Two requirements are fully addressed. SR-F-10 (immutable audit trail) and SR-NF-04 (audit retention and GDPR compliance) are both covered by the DLT Ledger and Decision Store. Several partial transfers hold: the confidence-threshold routing pattern underlying SR-F-01 is present through the Decision Guard's adjustable threshold; the structured correction schema underlying SR-F-04 is present through the Feedback Store's versioned SME-label format, though without standard partitioning or AS-model-version linkage; the retraining trigger pattern underlying SR-F-05 is present through the Retraining Notifier; the drift-monitoring pattern underlying SR-F-08 is present through the Feedback Manager's retraining-cycle monitoring, though it tracks model-performance drift rather than profile-conformance drift; the operator/SME role split partially underlying SR-F-09 is present through the role definitions in the Dashboard and APIs layer, though without a vertical promotion-approval chain; the escalation-to-senior-reviewer pattern partially underlying SR-F-12 is present through the Consensus Engine's quorum-failure alternative flow; the configurable-prioritisation pattern underlying SR-NF-03 is present through the Expert Review Queue's prioritisation policy, though without session-level pause, resume, or per-session load limits; actor-level role differentiation underlying SR-NF-05 is present through the persona definitions in the Dashboard and APIs layer, though without architectural separation-of-duties enforcement; and the operator-facing presentation layer partially underlying SR-NF-07 is present through the Dashboard and APIs layer, though without explicit accessibility guidance. These are real contributions that reduce the design problem for their respective requirements.

Three structural mismatches. Three requirements reveal not a shortfall in parameter settings but a mismatch between the class of problem each component was designed to solve and the class of problem the AS presents.

The first mismatch is SR-F-02 (batch review session management). The Expert Review Queue [25] buffers individually flagged decisions and presents them to reviewers ordered by a configurable prioritisation policy. This model is appropriate for CRIMSON, where crisis events arrive asynchronously and are acted on independently. The AS workflow is structurally different: all flagged mappings for a standardisation job are produced simultaneously, and the reviewing PPC

Data Engineer brings contextual knowledge that applies across the entire batch. The Expert Review Queue has no concept of a session as a unit of work, no batch completion tracking, no job-level grouping, and no mechanism to surface session state to the reviewer. Chapter 3's three-test gap filter framed this as an ordering problem: which uncertain cases to process first under a finite session budget. The requirements derivation reveals that framing identifies a downstream symptom. No reviewed architecture and no commercial stewardship platform (surveyed in Section 3.4) provides the session container within which any ordering policy would operate.

The second mismatch is SR-F-03 (standard-aware explainability). The Explainability Service [25] generates SHAP values and counterfactual explanations. These outputs identify which input features most influenced a prediction score. They do not address what a PPC Data Engineer needs: the formal definition of the candidate standard field, its position in the ontological hierarchy, and the ranked alternatives considered by the algorithm. Standard-aware explainability requires access to the target ontology, not to model internals. It is a qualitatively different output class that the Explainability Service cannot produce by parameterisation.

The third mismatch is SR-F-06 and SR-F-07 (mapping profile versioning and rollback). Maldonado's Decision Store records approved individual decisions; the Feedback Store records SME labels. Neither models a mapping profile: a named, versioned, reusable artifact governing how a specific PPC database is standardised. A crisis management decision is ephemeral: made, recorded, and closed. A mapping profile is persistent: created through a review process, promoted through a governance approval chain, monitored for drift, and rolled back if quality degrades. This artifact class is absent from Maldonado's framework; the commercial governance platforms surveyed in Section 3.4 version and approve data assets, but none of them governs a source-to-standard mapping profile as a versioned, drift-monitored, reversible artifact.

One Maldonado functional requirement, FR.006 (divergent feedback reconciliation via Consensus Engine), was excluded from the SR set: Participant 1 assessed that a single experienced reviewer is sufficient for routine mapping approvals, making concurrent multi-reviewer quorum management a deployment-scaling concern rather than a baseline prototype requirement; the PPC Data Engineer to PPC Data Governance Lead promotion chain provides the required two-actor verification for governance decisions. This does not conflict with SR-F-12: FR.006 reconciles concurrent reviewers acting on the same routine decision, whereas SR-F-12's junior tier engages only as a sequential escalation path on the subset of cases a single reviewer cannot resolve. Routine approval remains single-reviewer; the multi-reviewer tier is an architecture-specified escalation route, not the baseline.

A second Maldonado functional requirement, FR.007 (real-time override and veto), does not appear in the gap table because the trigger event it relies on is absent in the current AS context. The Decision Guard operates as a synchronous gateway that intercepts live model predictions. Both practitioners confirmed that asynchronous batch review is the expected operational model: Participant 1 described evaluating an entire batch before getting back to the machine, and Participant 2 stated that data mapping 'in the typical case I would expect it to be delayed' and that real-time HITL is avoided unless there is an explicit system requirement; there is no active inference stream

to intercept or override. FR.007 is architecturally inapplicable to the current AS design rather than an unaddressed gap, though a future streaming variant of the AS would reopen this requirement.

Verdict. Across the twenty requirements, Maldonado's framework fully addresses 2, partially addresses 9, and leaves 9 unaddressed. The 'minor policy tuning' characterisation in Section 5.4.6 holds for requirements where the underlying pattern is present (audit logging, confidence routing, retraining triggering) but does not hold for SR-F-02, SR-F-03, SR-F-06, and SR-F-07, each of which requires a new architectural component. Maldonado's Section 6.2 future work implicitly acknowledges this: the recommendation to 're-enact the architecture walkthrough with at least two further NOUS demonstrators' presupposes the walkthrough would surface new findings.

A second NOUS precedent. A separate thesis within the project specifies a human-in-the-loop architecture for the same energy use case, applied to the federated-learning models that produce PPC's energy forecasts [12]. Its data pipeline standardises source data through the AS and only then trains on the result, so the AS enters as an upstream preprocessing input and its supervision falls outside that work's scope. The two architectures govern different stages of the UC2 pipeline: the forecasting-stage work oversees the models that learn from standardised data, while this thesis oversees the standardisation that produces it. The distinction is not merely topical. Where the two share generic human-in-the-loop primitives, such as confidence-based routing of uncertain cases and a governed approval step, those primitives act on different objects: model predictions and versions in the forecasting work, field mappings and mapping profiles here. The structural mismatches identified above persist regardless, since the forecasting-stage architecture provides no batch review session over a standardisation job's outputs (SR-F-02), no standard-aware explanation of a candidate mapping (SR-F-03), and no versioned mapping profile artifact (SR-F-06, SR-F-07). It confirms, from a second point inside NOUS, that the supervision the AS requires is not supplied by adapting an existing NOUS oversight design.

4.5.3 Gap Map Summary

Table 4.3 maps all twenty requirements across both coverage dimensions using five gap levels: **Critical gap** (absent from the PRISMA corpus and not fully addressed by Maldonado, with operationally blocking consequences); **Major gap** (partial coverage exists, but the AS-specific dimension is absent from both sources); **Partial gap** (meaningful partial coverage that reduces but does not resolve the design problem); **Gap** (addressable by adopting an established external standard without novel architectural design); **Addressed** (full coverage in at least one source).

Six requirements are classified as critical gaps: SR-F-02, SR-F-06, SR-F-07, SR-F-12, SR-F-13, and SR-NF-06. None can be resolved by parameterising or adapting existing components; each requires new architectural work. They are also not independent pieces of work. The batch session, profile governance, and the oversight floor turn on a shared unit of work (the standardisation job) and a shared artefact (the approved mapping profile), so addressing them as separate components would leave the dependencies between them unmanaged. This is the reason the design of Chapter 5 is a single integrated architecture rather than a set of standalone solutions.

ID	Short Description	Literature	Maldonado	Gap Status
SR-F-01	Confidence-threshold routing	Partial	Partial	Partial gap
SR-F-02	Batch review session management	Absent	Not Addressed	**Critical gap**
SR-F-03	Standard-aware explainability	Partial	Not Addressed	Major gap
SR-F-04	Structured correction capture	Partial	Partial	Major gap
SR-F-05	Standard-partitioned retraining	Partial	Partial	Major gap
SR-F-06	Mapping profile versioning	Absent	Not Addressed	**Critical gap**
SR-F-07	Mapping profile rollback	Absent	Not Addressed	**Critical gap**
SR-F-08	Mapping output drift detection	Partial	Partial	Partial gap
SR-F-09	Role-based governance approval	Partial	Partial	Major gap
SR-F-10	Immutable audit trail	Partial	Addressed	Addressed
SR-F-12	Tiered escalation routing	Absent	Partial	**Critical gap**
SR-F-13	Algorithm steering / hinting	Absent	Not Addressed	**Critical gap**
SR-F-14	Provisional mapping acceptance	Partial	Not Addressed	Partial gap
SR-NF-01	Column coverage $\geq$80%	Partial	Not Addressed	Gap
SR-NF-02	Accuracy $\geq$80% via sampling	Partial	Not Addressed	Gap
SR-NF-03	Reviewer workload manageability	Partial	Partial	Partial gap
SR-NF-04	Audit retention and GDPR	Partial	Addressed	Addressed
SR-NF-05	Separation of duties	Partial	Partial	Partial gap
SR-NF-06	Minimum oversight floor	Absent	Not Addressed	**Critical gap**
SR-NF-07	Accessibility of review interface	Absent	Partial	Gap

Table 4.3: Gap map: all twenty supervision requirements assessed against the PRISMA corpus and Maldonado's framework.

4.6 Implementation Feasibility Analysis

The question here is different: not which single requirement is most critical, but which requirements can be fully implemented within a thesis cycle, which are only partially tractable, and which can only be specified architecturally. Gap severity (Section 4.5) and implementation feasibility are distinct axes: a requirement can be a critical gap on the evidence and still fall into a lower implementation tier because the resources to build and test it are unavailable in the prototype context. A critical gap placed in Tier 3 is therefore deferred for feasibility, not deprioritised on importance. Four criteria from the DSR methodology determine tier placement: whether an absent

requirement blocks the workflow or merely degrades it; whether the gap requires genuinely new architectural work; whether design and implementation are realistic within a single master's thesis; and whether the result can be tested without ML pipeline access or historical reviewer data.

Tier 1 — Targeted for full implementation. Confidence-threshold routing (SR-F-01), batch review session management (SR-F-02), mapping profile versioning and rollback (SR-F-06, SR-F-07), drift detection (SR-F-08), role-based governance approval (SR-F-09), reviewer workload controls (SR-NF-03), separation of duties (SR-NF-05), the minimum oversight floor (SR-NF-06), and provisional mapping acceptance (SR-F-14). Each is blocking or governance-critical, absent from the reviewed literature and from Maldonado's framework, and fully testable against the mock API without ML pipeline access. These requirements fall into two clusters: the batch session cluster (SR-F-01, SR-F-02, SR-NF-03, SR-NF-06, SR-F-14) and the profile governance cluster (SR-F-06, SR-F-07, SR-F-08, SR-F-09, SR-NF-05). Together they constitute the implementation core of this thesis.

Tier 2 — Partially tractable. Some requirements split between what is tractable now and what depends on ML pipeline access or ontological resources unavailable in the mock environment. Standard-aware explainability (SR-F-03): the display layer will be implemented, surfacing confidence scores and candidate provenance; ontological lookup across SAREF, CIM, and IEC 61850 is architecture-specified. Structured correction capture (SR-F-04): the schema and API contract will be implemented; the retraining trigger requires a live model and is deferred.

Tier 3 — Architecture-specified only. Standard-partitioned ML retraining (SR-F-05) requires a live training pipeline and cannot produce meaningful evidence from the mock API. Tiered reviewer escalation (SR-F-12) is architecture-specified: the three-tier routing model is designed, but quality-based calibration requires historical reviewer performance data that does not exist in the prototype context. Algorithm steering and hinting (SR-F-13) is architecture-specified: the hint data contract is defined and a capture endpoint is included in the API specification, but AS-side execution cannot be evaluated without a live algorithm. Audit retention and GDPR compliance (SR-F-10, SR-NF-04) are addressed by established regulatory standards; they are architecture-specified but do not require novel design. Post-submission accuracy verification (SR-NF-02) is architecture-specified: accuracy is computable from submitted session data, but the calibration decisions required to set enforcement thresholds depend on production confidence distributions unavailable in the prototype context.

WCAG 2.1 AA accessibility (SR-NF-07) is not designed in this thesis; the prototype interface is functional but not validated against the standard. Chapter 7 acknowledges this as a limitation rather than as an architecture-specified component.

SR-NF-01 (column coverage $\geq$80%) is not assigned to any implementation tier. The supervision layer only receives the mapping candidates the AS chooses to submit; it has no ground truth of the total column count in the source database schema and therefore cannot detect missing columns. Enforcing a coverage threshold would require the full source schema as a separate input to the supervision system, an integration scope expansion not justified by any other requirement. Column coverage is a quality property of AS output, not of the supervision workflow, and its enforcement point sits upstream of the supervision layer boundary. It was analysed here because it arises from

the same UC2 acceptance criteria as the other requirements; that analysis concludes it falls outside supervision-architecture scope, and Chapter 5 excludes it accordingly.

4.7 Summary

This chapter delivered contribution C1: a specification of twenty supervision requirements, thirteen functional and seven non-functional, for the NOUS Auto-Standardiser context, completing Stages 1 and 2 of the Design Science Research process. The requirements were derived from the task development plan and use-case acceptance criteria, two semi-structured practitioner interviews, the systematic literature review of Chapter 3, and Maldonado's NOUS HITL framework as the intra-project baseline. The gap analysis of Section 4.5 assessed each requirement against both the literature and that baseline, and the feasibility analysis of Section 4.6 distributed the twenty requirements across three implementation tiers, identifying the batch review session cluster and the profile governance cluster as the implementation core of the thesis. Chapter 5 carries these requirements forward into the supervision architecture design.

Chapter 5

Supervision Architecture Design

5.1 Introduction

This chapter advances to Stage 3 of Peffers et al.'s [32] Design Science Research process: the design and development of an artifact that addresses the identified problem. Batch review session management (SR-F-02) anchors the implementation as the architectural precondition for the full supervision stack. Without a session model, the routing layer has nowhere to deliver flagged items, the profile governance service has no submitted session to read decisions from, and the minimum oversight floor has no enforcement point. The feasibility analysis in Section 4.6 distributed the twenty supervision requirements across three implementation tiers, with the SR-F-02 cluster forming the core of Tier 1.

This chapter specifies the supervision architecture for the twenty requirements identified in Chapter 4 and presents the implementation design for the three component clusters that form the thesis implementation scope: Confidence Routing (SR-F-01, SR-NF-06); Batch Review Session Manager (SR-F-02, SR-NF-03, SR-F-14); and Profile Governance (SR-F-06, SR-F-07, SR-F-08, SR-F-09, SR-NF-05). Two Tier 2 requirements, standard-aware explainability (SR-F-03) and structured correction capture (SR-F-04), are partially implemented within the session manager and governance clusters respectively. Five architecture-specified components address requirements that are not implemented: standard-partitioned ML retraining (SR-F-05), algorithm steering execution and hinting (SR-F-13), tiered reviewer escalation (SR-F-12), immutable audit trail with GDPR compliance (SR-F-10, SR-NF-04), and post-submission accuracy verification (SR-NF-02); these are specified in Section 5.7, alongside the deferred SR-F-03 ontology lookup service. WCAG 2.1 AA accessibility (SR-NF-07) is not addressed at the prototype stage; the interface is functional but not validated against the standard, and this is acknowledged as a limitation in Chapter 7.

Section 5.2 establishes the design principles, system scope, and component structure. The fourteen design objectives in Section 5.3 are the evaluation criteria for Chapter 7. The three service-level clusters are specified in Sections 5.4–5.6; Sections 5.7–5.9 address the non-implemented components, the API contract, and a summary of all design decisions.

5.2 Architecture Overview

Six design principles guided the architecture before any component was specified. They recur as justifications throughout Sections 5.4–5.6; a reader who contests a principle will contest the design decisions that follow from it.

Principle	Description
Batch-Scoped Oversight	The unit of supervision is the session (one AS job), not the individual item. Item-by-item queues are structurally mismatched with schema review, where related mappings share a source schema, naming convention, and domain vocabulary.
Non-Negotiable Minimum Floor	Human review cannot fall to zero regardless of model confidence. SR-NF-06 is enforced structurally at the API boundary (HTTP 422), not as a configuration option or advisory warning.
Single Source of Truth	`session_items` is the authoritative record of item state. The Routing Engine makes no database writes; routing decisions are captured as columns on session items to eliminate any second source that could diverge.
Pull-Based Integration	The supervision layer makes no outbound calls. Downstream consumers pull corrections and hints via `GET /corrections` and `GET /hints/{database_id}`. The integration contract sits entirely on the data side.
Separation of Review and Governance	The reviewer and governance-approver roles are structurally enforced via the Role Guard. A `DATA_ENGINEER` cannot approve their own mapping profile proposal.
Deferred but Complete	Review is asynchronous (sessions are scheduled work, not real-time interrupts) but produces complete, versioned, rollback-capable artefacts rather than ephemeral per-item decisions.

Table 5.1: Design principles for the NOUS AS Supervision Architecture.

5.2.1 System Scope

Three component clusters form the implementation core, each addressing a primary domain-specific challenge established in the requirements analysis: Confidence Routing (C-RT); Batch Review Session Manager (C-BW); and Profile Governance (C-PG). The remaining challenges, correction-driven retraining (C-CR) and multi-standard explainability (C-ME), are addressed by Tier 2 sub-components within these clusters rather than as standalone clusters.

The Confidence Routing cluster (SR-F-01, SR-NF-06) intercepts job outputs from the Auto-Standardiser and partitions items by confidence score before any review session is created. A configurable auto-acceptance threshold divides items into those that bypass review entirely and those that require it. A minimum sampling floor then pulls a mandatory spot-check from the auto-accepted pool into the session, ensuring human review never falls to zero regardless of how many items clear the threshold.

The Batch Review Session Manager cluster (SR-F-02, SR-NF-03, SR-F-14) groups the routed items into a bounded unit of work tied to one AS job. The session model enforces pause and resume, tracks progress counts, orders items by confidence, and prevents submission until all mandatory-sample items carry a non-null decision.

The Profile Governance cluster (SR-F-06, SR-F-07, SR-F-08, SR-F-09, SR-NF-05) manages the lifecycle of mapping profiles, versioned governance artefacts derived from submitted session decisions. The service governs the DRAFT to PROPOSED to APPROVED to SUPERSEDED transition chain, enforces role-based approval, detects when incoming AS job outputs diverge from an approved profile, and supports rollback to a prior approved version without creating new rows.

Two Tier 2 requirements are addressed within the cluster infrastructure rather than as standalone clusters. Standard-aware explainability (SR-F-03) is partially implemented: the display layer is realised through the Anti-Corruption Layer's candidate normalisation and the session item queue, which expose the data the display panel requires; the ontology lookup service that would populate hierarchy path and sibling concepts for each candidate is architecture-specified only (Section 5.7). Structured correction capture (SR-F-04) is implemented through the CORRECTED decision type and the `corrected_to` schema extension on session items.

Figure 5.1 shows the system in its operational context. Two human actors interact with it: the PPC Data Engineer, who reviews mappings and captures corrections; and the PPC Data Governance Lead, who approves and rolls back mapping profiles.

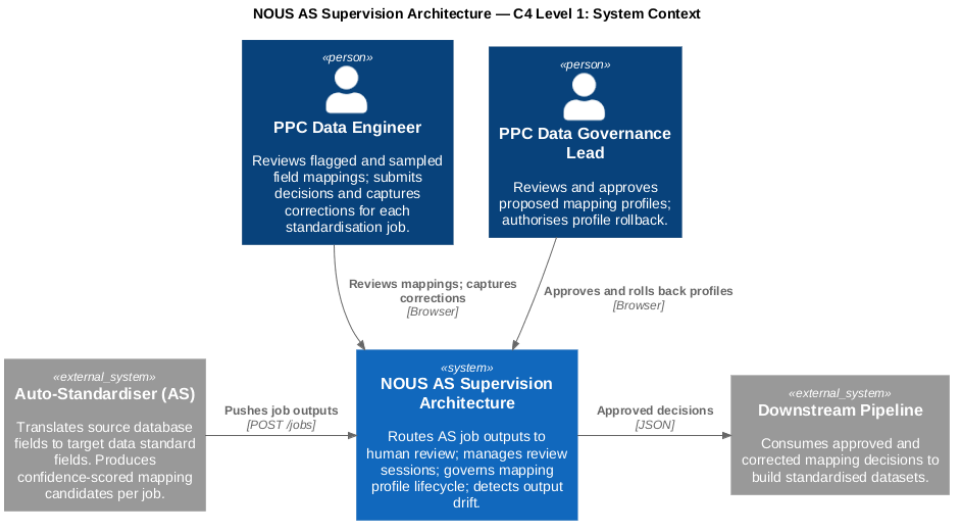

Figure 5.1: C4 Level 1 (Context): the NOUS AS Supervision Architecture and its external actors.

SR-NF-01 (column coverage $\geq$80%) is excluded from supervision architecture scope. The supervision layer only receives the mapping candidates the AS chooses to submit; it has no ground truth of the total column count in the source database schema and therefore cannot detect missing columns. Enforcing a coverage threshold would require the full source schema as a separate input, an integration scope expansion not justified by any other requirement. Column coverage is a quality property of AS output, not of the supervision workflow, and its enforcement point sits upstream of

the supervision layer boundary. SR-F-11 was excluded from requirements analysis entirely for the same upstream-boundary reason (Section 4.2); SR-NF-01 was analysed in Chapter 4 and excluded from architecture scope here once its enforcement point became clear.

5.2.2 Container Structure

The system is realised as three containers: a FastAPI backend, a two-surface browser interface, and a persistent supervision store.

The REST API boundary is the integration contract between the supervision architecture and any upstream system. The backend receives job outputs conforming to a documented JSON schema; it shares neither code nor a database with the AS. The contract is not coupled to AS-specific data formats; a batch classification system producing confidence-scored outputs could in principle integrate by conforming to this schema, though applicability beyond the NOUS UC2 context has not been evaluated.

The Supervision Interface is a React single-page application with two surfaces. The Review surface, accessed by Data Engineers, presents the session item queue, decision controls, and correction capture for each job. The Governance surface, accessed by Governance Leads, presents the profile lifecycle, drift alerts, and role-gated approval actions. Both surfaces communicate exclusively with the FastAPI backend via the REST API.

The Supervision Store persists all session state, item decisions, mapping profiles, profile entries, and drift alerts without requiring a separately deployed database service.

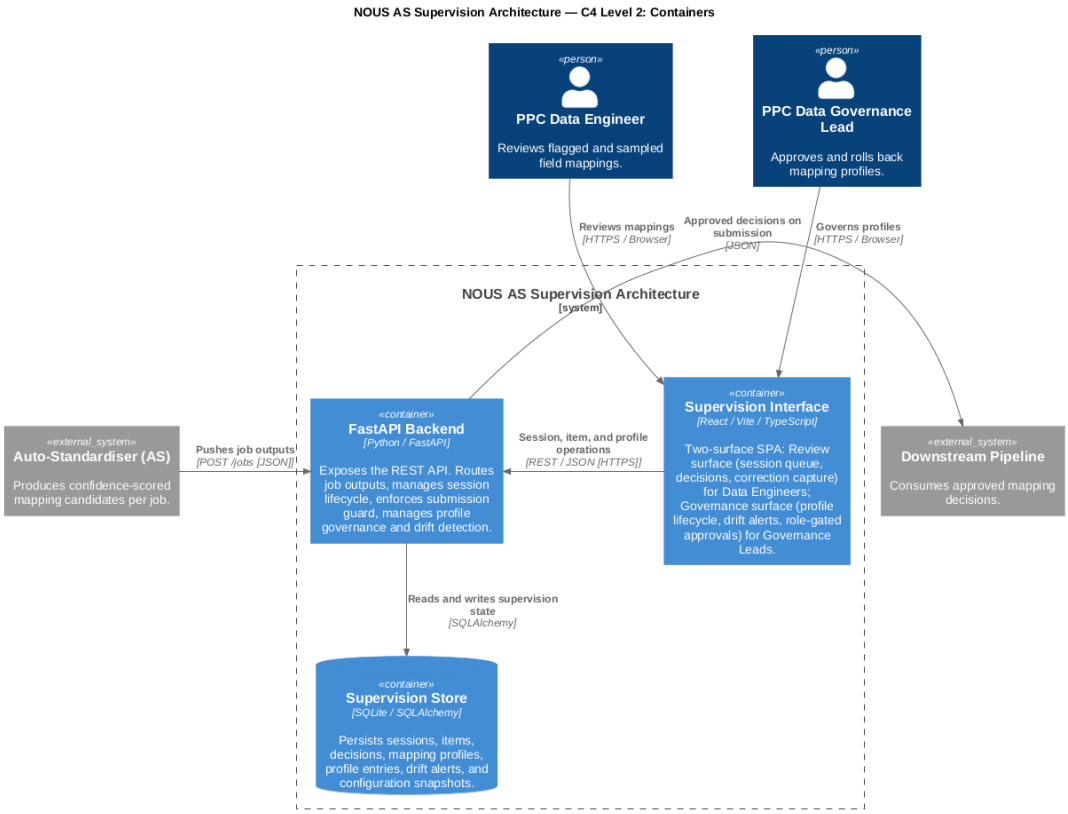

Figure 5.2: C4 Level 2 (Container): FastAPI backend, two-surface Supervision Interface, and Supervision Store.

All three clusters from Section 5.2.1 are realised as components within the FastAPI backend; the interface and store containers serve them but carry no cluster of their own.

The Supervision Store holds two logically distinct partitions: the Session Decision Record (`sessions` and `session_items`, effectively append-only once submitted) and the Profile Store (`profiles` and `profile_entries`, versioned and mutable through the governance lifecycle). These are co-located in the prototype.

5.2.3 Key Architectural Decisions and Assumptions

Four decisions were made when specifying the C4 structure; they are labeled here to distinguish what was decided from what was assumed or deferred.

D1: AS as External System. The Auto-Standardiser is modelled as an external system at C4 Level 1. The supervision layer shares neither code nor a database with the AS; integration is the HTTP boundary only (`POST /jobs` inbound; `GET /corrections` and `GET /hints/{database_id}` outbound pull). This keeps the supervision layer independently deployable and testable against a mock without the live ML pipeline.

D2: Supervision Store as Single Container. Session decisions and governance artefacts are co-located in one Supervision Store rather than split across separate services. Co-location removes the need for separate deployment infrastructure and connection configuration, which is appropriate

at prototype scale where the evaluation runs against a local mock API with modest data volumes. The two logical partitions described in Section 5.2.2 are architecturally independent; separating them in a production deployment is not expected to require changes to the API contract.

D3: Role Enforcement via Header, Not Auth Middleware. The `x-reviewer-role` header is a plain string accepted without token validation. The Role Guard uses it to enforce role-based access control structurally. This is expected to be replaceable with JWT claims without schema changes; the header-based approach is an intentional exclusion of authentication complexity from the prototype scope (Section 5.8).

A1: Mock AS for Evaluation. The integration boundary is exercised against a mock AS script rather than the live ML pipeline. The mock generates synthetic per-job field identifiers (`field_id`, the stable source-field key the drift detector relies on, defined in Section 5.5.4) rather than stable schema-derived values; the implications for drift detection evaluation are acknowledged in Chapter 7 as a threat to validity. The mock's full design (covering controlled confidence distributions, scenario setup, and the synthetic `field_id` limitation) is described in Chapter 6.

5.2.4 Artifact Form

In Hevner et al.'s [16] taxonomy of DSR artifacts, this work produces an *instantiation* (an implemented, operational system) rather than a construct, model, or method. Two properties of the supervision requirements make instantiation the appropriate form. The behavioral requirements (pause and resume with decision persistence, submission guard enforcement, role-gated approval, automatic drift detection at ingestion) specify properties that must be observed in operation; a specification alone cannot verify them. The system also occupies a specific integration position: it receives job outputs from the AS over HTTP and returns approved decisions to the downstream pipeline. A REST API with a browser-based interface is an appropriate artifact form for this integration position: it exposes the behavioral properties specified by the design objectives in a form that can be tested independently of the UI, and it produces a machine-readable OpenAPI contract as a secondary evaluation artifact, directly verifiable against the endpoint specifications in Section 5.8.

5.3 Design Objectives

The three component clusters described in Section 5.2 address twelve supervision requirements across the implemented tiers. The fourteen design objectives listed in Table 5.2 translate the ten Tier 1 requirements into specific, falsifiable conditions against which Chapter 7 evaluates the implemented components; the two Tier 2 requirements (SR-F-03, SR-F-04) are addressed structurally but assessed qualitatively in Track B (practitioner validation, Chapter 7).

Track A evaluates the implemented components against the mock Auto-Standardiser API using synthetic job outputs with controlled confidence distributions, so each DO can be tested against a known ground truth. Track B has practitioners operate the implemented components to assess whether they address the supervision needs identified during requirements analysis. A design

ID	Objective	Derived from
DO-1	A session must contain all flagged and mandatory-sample mappings from a single standardisation job, and no items from other jobs	SR-F-02
DO-2	A reviewer must be able to pause a session and resume it with all prior decisions preserved	SR-F-02
DO-3	The session must expose reviewed and unreviewed item counts at all times during an active session	SR-F-02, SR-NF-03
DO-4	Items must be presented in confidence-sorted order within each tier: provisional carry-forwards (Tier 0) before newly flagged items (Tier 1), and newly flagged before mandatory-sample items (Tier 2)	SR-NF-03
DO-5	Every session must include a minimum configurable sample of high-confidence items drawn from the auto-accepted pool	SR-NF-06
DO-6	Session submission must be rejected at the API boundary if any mandatory-sample item has not received a non-null, non-SKIPPED decision	SR-NF-06
DO-7	A session item marked PROVISIONAL in a SUBMITTED session must appear as a Tier 0 carry-forward in the next session created for the same `database_id`, and submission of that next session must be blocked until every Tier 0 carry-forward receives a non-null decision	SR-F-14
DO-8	Items at or above `routing_threshold_auto` must be excluded from the review session unless selected by the sampling floor (DO-9)	SR-F-01
DO-9	The minimum sampling floor, the maximum of a proportional rate and an absolute count, must be applied by the routing engine before session creation	SR-NF-06
DO-10	A DRAFT profile created from a SUBMITTED session must include entries for all APPROVED, CORRECTED, and AUTO_ACCEPTED items; PROVISIONAL and SKIPPED items must be excluded	SR-F-06
DO-11	The PROPOSED to APPROVED transition must be rejected with HTTP 403 for the DATA_ENGINEER role; only GOVERNANCE_LEAD may approve	SR-F-09, SR-NF-05
DO-12	Profile rollback must set the target version to APPROVED and the previously active version to SUPERSEDED without creating new rows	SR-F-07
DO-13	A DriftAlert must be created automatically at job ingestion when the fraction of diverging fields exceeds `drift_alert_threshold`	SR-F-08
DO-14	`JobResponse.drift_alert_id` must be non-null if and only if a DriftAlert was created for that job	SR-F-08

Table 5.2: Design objectives for the NOUS AS Supervision Architecture, derived from supervision requirements. DO-1 to DO-9 cover the session and routing clusters; DO-10 to DO-14 cover the profile governance cluster.

objective is considered satisfied when the system's behaviour under the corresponding test scenario matches the specified condition without manual intervention.

5.4 Confidence Routing Layer

Role: Pure function that partitions incoming AS job items into confidence tiers by applying configurable thresholds and a minimum sampling floor; returns a list of `RoutingDecision` objects to the Session Manager (Section 5.5) and makes no database writes.

Anatomy:

- *Anti-Corruption Layer*: translates AS `JobItem` Pydantic objects to internal dicts; normalises `candidates` to descending confidence order; sets `confidence = candidates[0]["confidence"]` so both threshold comparisons operate on the same value.
- *Threshold Evaluator*: compares `item.confidence` against `routing_threshold_auto`; assigns routing type AUTO_ACCEPTED or FLAGGED.
- *Sampling Selector*: draws a mandatory spot-check from the AUTO_ACCEPTED pool; the number of items to pull back is computed from a configurable proportional rate and an absolute minimum floor (Section 5.4.2); selected items have their `sampled` flag set to True before the list is returned, which the Session Manager subsequently maps to `SessionItem.routing=SAMPLED`.
- *RoutingDecision Publisher*: returns the completed list of `RoutingDecision` dataclasses (one per item: `routing_type`, `sampled`); no state is written and no further side effects occur.

The internal component structure is shown in Figure 5.3.

The end-to-end ingestion flow (from AS job submission through anti-corruption translation, routing, session creation, and drift detection) is shown in Figure 5.4.

5.4.1 Motivation

SR-F-01 (confidence-threshold routing) and SR-NF-06 (minimum oversight floor) are both resolved before a session is created. They determine which items enter a session and in what form.

SR-F-01 requires that items at or above a configurable confidence threshold be excluded from human review. The design question is where to enforce it. Embedding threshold logic inside the session manager would conflate two distinct concerns: session lifecycle management and routing policy. The routing policy is a business rule that belongs to the system as a whole, not to the component that manages review sessions. Separating it as an upstream pure function (one that receives items and a configuration, and returns routing decisions without touching the database)

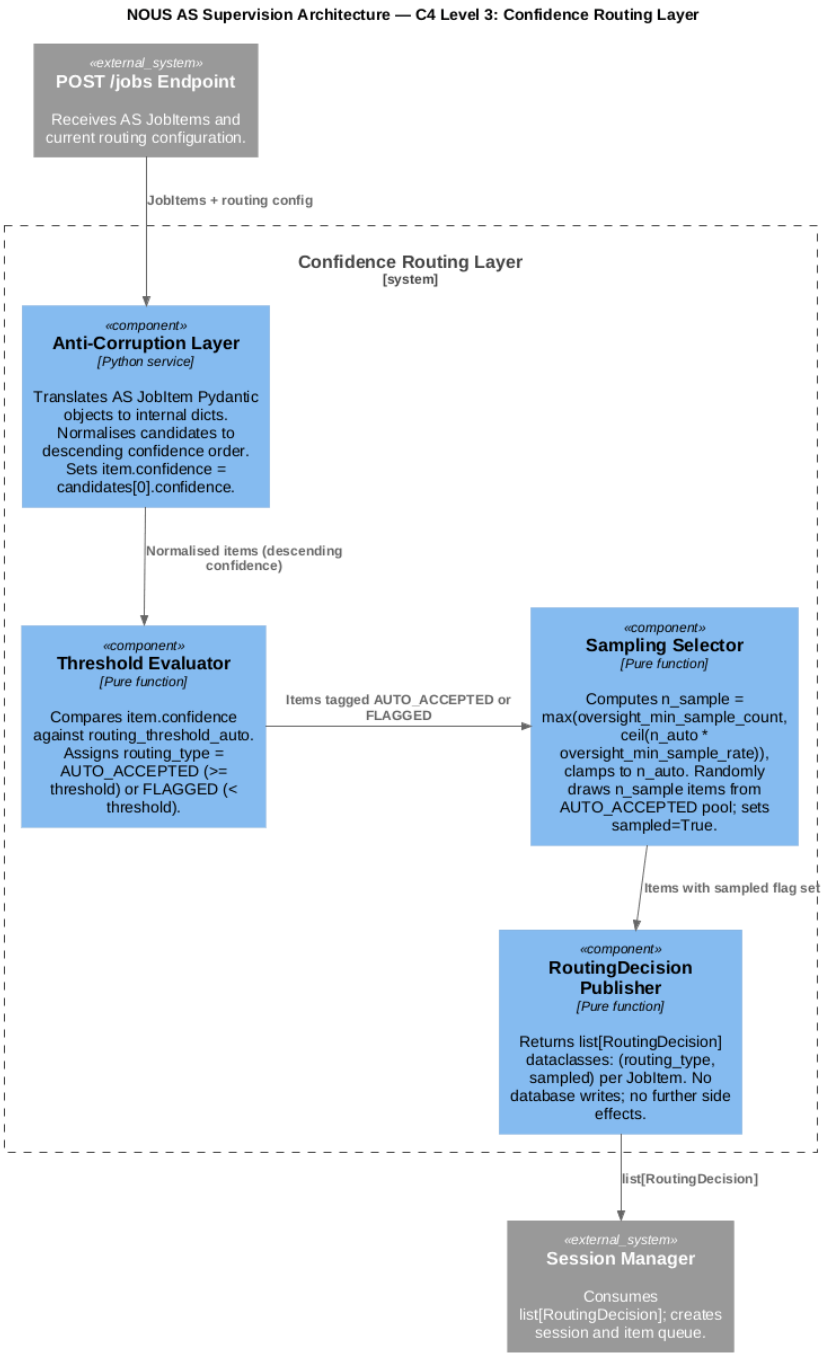

Figure 5.3: C4 Level 3 (Confidence Routing Layer).

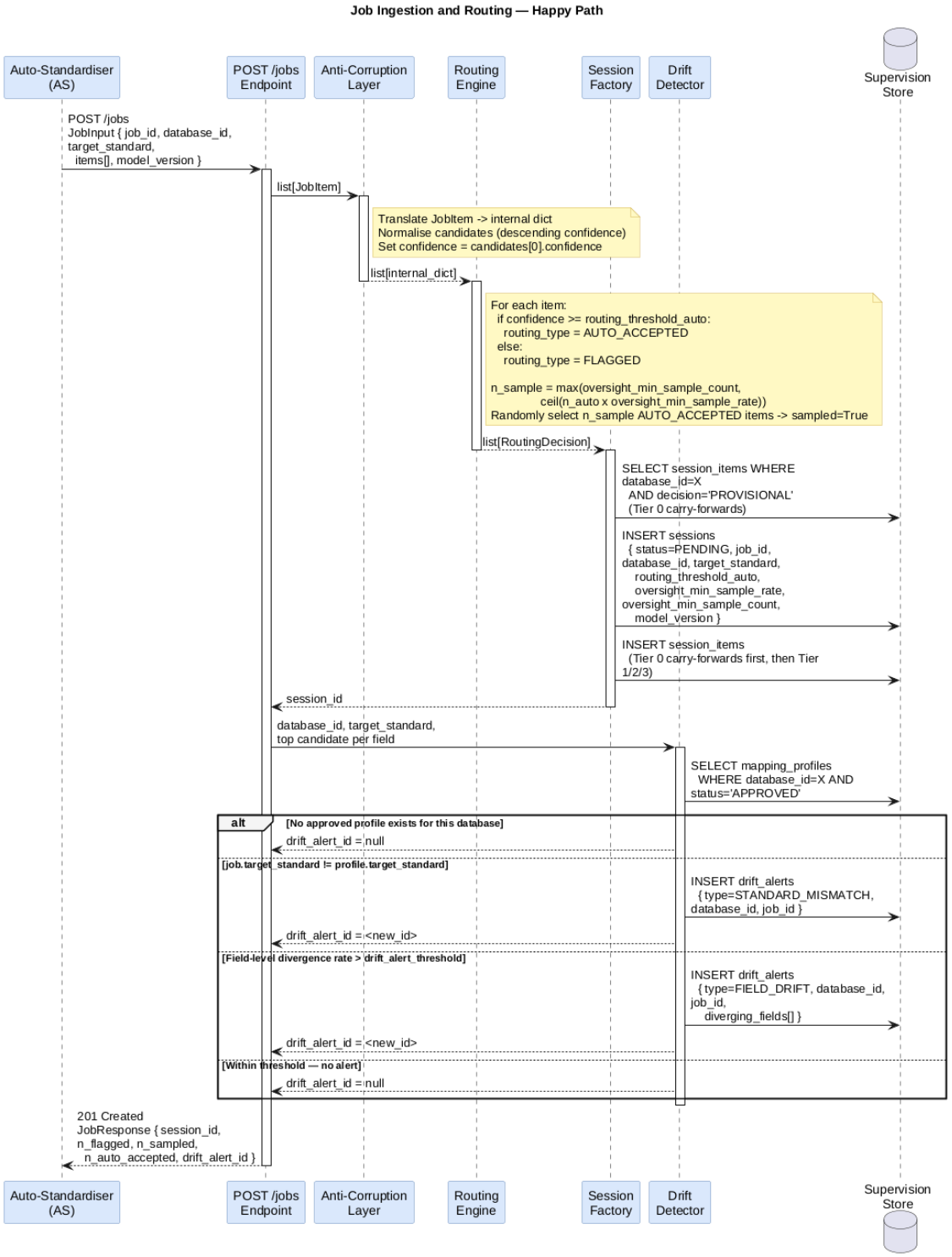

Figure 5.4: Job ingestion and routing sequence (happy path).

preserves this separation and allows the threshold model to change without modifying the session manager.

SR-F-12 (tiered escalation) extends this routing layer but is architecture-specified only; Section 5.7 gives the full specification, including why it is deferred and its substitution point.

SR-NF-06 requires a minimum sampling floor on high-confidence auto-accepted items. This is logically a routing decision: the floor determines which items from the auto-accepted pool are pulled back into the session for mandatory review. Assigning this responsibility to the routing engine rather than the session manager keeps the session manager's contract clean: it receives a pre-determined set of items and creates the session without applying any confidence logic.

5.4.2 Routing Algorithm

The routing engine is implemented as a pure function: `route(items, config)` → `list[RoutingDecision]`. It performs no database reads or writes. Each incoming `JobItem` carries a top-level `confidence` field equal to `candidates[0].confidence`, the AS's certainty score for its highest-ranked mapping candidate. The algorithm operates as follows:

```
for each item in items:
    if item.confidence >= routing_threshold_auto:
        item.routing_type <- AUTO_ACCEPTED
    else:
        item.routing_type <- FLAGGED

n_auto = |{item : item.routing_type = AUTO_ACCEPTED}|
n_sample = max(oversight_min_sample_count,
               ceil(n_auto x oversight_min_sample_rate))
n_sample = min(n_sample, n_auto)    {clamp: cannot exceed pool size}

randomly select n_sample items from AUTO_ACCEPTED pool:
    selected.routing_type <- FLAGGED  {out of auto pool}
    selected.sampled      <- True       {mandatory review}

return list[RoutingDecision] per item: routing_type, sampled
```

The function returns a `RoutingDecision` dataclass per item, a plain value type with no database identity. The Session Manager maps each decision to a `SessionItem` row: `sampled=True` → `routing=SAMPLED`, tier 2; `routing_type=FLAGGED`, `sampled=False` → `routing=FLAGGED`, tier 1; `routing_type=AUTO_ACCEPTED`, `sampled=False` → `routing=AUTO_ACCEPTED`, tier 3 (invisible to reviewer, persisted for profile creation). The routing engine makes no decisions about session structure; it classifies items and passes them on.

The Sampling Selector computes n_{sample}, the number of AUTO_ACCEPTED items to pull back into the session for mandatory review, as follows:

$$n_{\text{sample}} = \max\left(\texttt{oversight\_min\_sample\_count},\ \lceil n_{\text{auto}} \times \texttt{oversight\_min\_sample\_rate} \rceil\right)$$

where n_{auto} is the count of AUTO_ACCEPTED items in the current job; `oversight_min_sample_rate` is a configurable proportional rate (default 0.05, i.e. 5%); and `oversight_min_sample_count` is a configurable absolute minimum (default 1). The max ensures both constraints hold simultaneously: the proportional term scales with job size, while the absolute term prevents the floor from collapsing to zero on small jobs. For a job with 200 auto-accepted items at default parameters, $\lceil 200 \times 0.05 \rceil = 10$ items are pulled back for mandatory review. For a job with only 5 auto-accepted items, the proportional term yields $\lceil 5 \times 0.05 \rceil = 1$, so the absolute floor takes over and exactly 1 item is always sampled. When the AUTO_ACCEPTED pool is smaller than `oversight_min_sample_count`, for example a job with only 3 auto-accepted items at `oversight_min_sample_count = 5`, the clamp step ensures the sample size cannot exceed the pool, so all 3 are sampled and the floor degrades gracefully on small jobs; the case where `n_auto = 0` is handled by the same clamp, so the routing engine produces an empty sample without error. This follows the configurable-but-bounded principle: a system administrator sets the parameters, but the floor cannot drop below `oversight_min_sample_count` regardless of job size [4]. The default value of `routing_threshold_auto` is 0.85, chosen to produce a representative confidence distribution in the evaluation scenarios of Chapter 7. This default was not calibrated against real AS confidence distributions, which are unavailable in the prototype context; production use requires calibration against the observed distribution of the deployed AS. All three routing parameters are snapshotted onto the Session record at creation, so system-level configuration changes do not affect in-flight sessions.

5.4.3 Design Decisions

Decision	Requirement	Rationale
Pure function with no database writes	Architecture	Routing decisions are transient; their outcome is fully captured by `routing_type` on session items. Keeping the function side-effect-free simplifies testing and removes coupling between routing policy and persistence.
Sampling floor applied by routing engine, not session manager	SR-NF-06	The floor determines which items enter the session, making it logically a routing decision. Assigning it to the session manager would give that component responsibility for confidence-based logic it should not need to understand.

Table 5.3: Key design decisions for the routing and escalation layer.

5.5 Batch Review Session Manager

Role: Receives pre-routed items from the Routing Engine and creates a bounded, pausable unit of work tied to a single AS job; enforces the session lifecycle from PENDING through SUBMITTED to ARCHIVED.

Anatomy:

- *Session Factory*: creates the Session record with snapshotted configuration parameters; injects Tier 0 provisional carry-forwards from any prior submitted session for the same database before creating the item queue.
- *Lifecycle State Machine*: enforces PENDING → ACTIVE ↔ PAUSED → SUBMITTED → ARCHIVED transitions; the ACTIVE → SUBMITTED transition is the only guarded user-initiated transition.
- *Item Queue*: filters `WHERE tier < 3` to exclude AUTO_ACCEPTED items; orders by `tier ASC, confidence ASC`; exposes per-tier progress counts.
- *Submission Guard*: validates that all SAMPLED items carry a decision ≠ NULL and ≠ SKIPPED before permitting submission; returns HTTP 422 with the IDs of blocking items if the check fails (Section 5.5.5).

The internal component structure is shown in Figure 5.5.

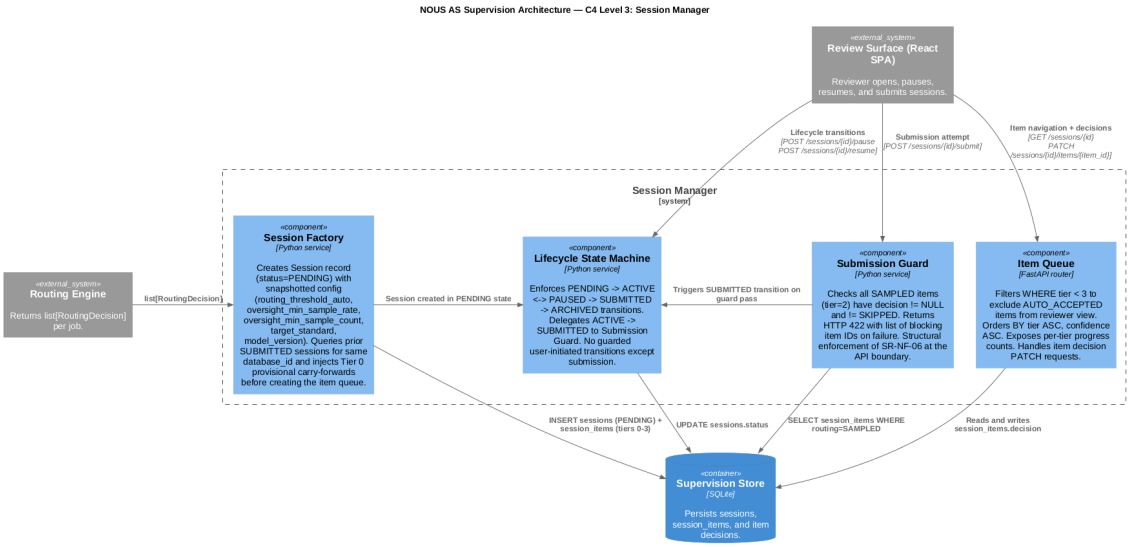

Figure 5.5: C4 Level 3 (Session Manager).

5.5.1 Session as Unit of Work

The session manager receives pre-routed items from the Routing Engine (Section 5.4) as a list of `RoutingDecision` objects; it does not apply confidence thresholds or sampling logic itself.

A standardisation job in the NOUS Auto-Standardiser produces all field mapping candidates for a source database before any human review begins. The T5.4 development plan describes the system as translating complete source databases to target data standards [24]. Participant 1 described the operational workflow that follows: reviewing all of them, evaluating them, then getting back to the machine. The engineer's role is to assess the resulting field matches across the database as a whole. The reviewer's task is therefore not a sequence of independent decisions but a coherent assessment of how a specific database schema has been interpreted against a target standard.

The PPC Data Engineer brings contextual knowledge that is intrinsically batch-scoped. Recognising that a source database encodes energy concepts using non-English abbreviations or vendor-specific naming conventions is knowledge that applies across every related field in the same job simultaneously. A reviewer who identifies this pattern in one flagged mapping immediately sees its implications for several others in the same batch. An interface that presents mappings one at a time, without exposing what else is pending in the same job, prevents the reviewer from applying this knowledge systematically.

Individual-item review queues are architecturally mismatched with this workflow. Maldonado's Expert Review Queue [25] buffers individually flagged decisions ordered by a configurable prioritisation policy, which suits CRIMSON, where crisis events arrive asynchronously and each is independent of the last. In the AS context, independence is the exception: mappings from the same job share a source schema, a naming convention, and a domain vocabulary. Every review interface among the 16 papers in the PRISMA corpus follows the same individual-item model, but all were designed for entity-level annotation rather than schema-level review. Applying an item-by-item queue to AS batch outputs eliminates the inter-mapping context that makes expert review both efficient and accurate.

A session groups all flagged and sampled mappings from a single standardisation job into a bounded unit of work. The reviewer opens the session, works through the full item set with access to the whole, and submits all decisions at once. The session boundary is the correct unit of provenance for the audit trail required by SR-NF-04: it records all decisions for a given job together, in the context in which they were made.

The session model is the design's novel response to a gap documented in Chapters 3 and 4: no reviewed architecture and no commercial stewardship platform (surveyed in Section 3.4) models review as a batch session tied to a single standardisation job. It is grounded not in a prior-art precedent, which does not exist, but in the elicited requirement SR-F-02 and the documented absence of any batch-scoped alternative.

5.5.2 Session Lifecycle

Sessions follow a five-state lifecycle that enforces the boundary between in-progress work and committed decisions.

PENDING: The session exists following job ingestion but has not yet been opened. Configuration parameters are snapshotted at this point.

ACTIVE: The reviewer has opened the session and is working through the item queue. Item decisions can be submitted one at a time; the progress counter updates with each decision.

PAUSED: The reviewer has temporarily suspended the session. All prior decisions are persisted; the session can be resumed from exactly the state at which it was paused.

SUBMITTED: The reviewer has submitted the complete session. The submission guard has verified that all mandatory-sample items have received a non-null, non-SKIPPED decision (Section 5.5.5). Once submitted, no further decisions can be recorded against this session.

ARCHIVED: The session record has passed the retention threshold. This transition is system-triggered and corresponds to the audit retention obligations of SR-NF-04; the cold-storage mechanism is out of prototype scope and is a downstream consequence of the audit-retention architecture described in Section 5.7.

Transition	Trigger	Guard
PENDING → ACTIVE	Reviewer opens	None
ACTIVE → PAUSED	Reviewer pauses	None
PAUSED → ACTIVE	Reviewer resumes	None
ACTIVE → SUBMITTED	Reviewer submits	All SAMPLED items must have decision ≠ NULL and ≠ SKIPPED
SUBMITTED → ARCHIVED	Retention elapsed	System-level only; no reviewer action

Table 5.4: Session lifecycle transitions, triggers, and guards.

The PENDING → ACTIVE and PAUSED → ACTIVE transitions have no guards because the reviewer should never be blocked from beginning or resuming a session. The ACTIVE → SUBMITTED transition is the only guarded user-initiated transition; the guard implements the minimum oversight floor of SR-NF-06 at the architectural boundary (Section 5.5.5).

5.5.3 Item Taxonomy

Items within a session are classified along two orthogonal dimensions: how the item entered the session (routing type) and what the reviewer decided about it (decision type).

Routing type is assigned at session creation and does not change:

FLAGGED: The item's confidence score falls below the configured threshold. Review is required. Tier 0 carry-forward items carry this routing type: they were flagged in a prior session, received a PROVISIONAL decision, and have been re-injected into the current session. The `tier` column (0 rather than 1) and a non-null `carried_from` foreign key distinguish them from newly flagged items.

SAMPLED: The item's confidence score meets or exceeds the threshold, but it has been randomly selected from the auto-accepted pool as a mandatory spot-check. It appears in the session because the minimum oversight floor requires it, not because the algorithm was uncertain.

AUTO_ACCEPTED: The item's confidence score meets or exceeds the threshold and it was not selected for sampling. It does not appear in the reviewer queue; no reviewer action is required or available. The item is persisted in the session at Tier 3 for profile creation (Section 5.6).

Decision type is assigned by the reviewer:

APPROVED: The reviewer accepts the algorithm's primary mapping suggestion as correct.

CORRECTED: The reviewer rejects the primary suggestion and specifies the correct mapping, captured in the `corrected_to` field for subsequent correction-driven retraining (SR-F-05).

PROVISIONAL: The reviewer accepts the mapping tentatively to avoid blocking the downstream pipeline but flags it for re-examination. Provisional items resurface in the next session created for the same source database.

SKIPPED: The reviewer defers the decision within the current session. Valid for FLAGGED items only; SAMPLED items cannot be skipped because the submission guard requires a decision on all sampled items.

5.5.4 Logical Data Model

The session model is realised as two tables: `sessions` and `session_items`. Full column definitions are listed in Appendix B.2; the discussion here addresses the design choices encoded in the schema.

Two columns on `sessions` deserve explicit comment. `database_id` is distinct from `job_id` because a source database may be standardised across multiple successive jobs; the database identifier is what links sessions for provisional carry-forward purposes (a session for job $N+1$ on database X must see the open provisional items from job N on database X). The three threshold and sample-rate columns (`routing_threshold_auto`, `oversight_min_sample_rate`, `oversight_min_sample_count`) snapshot the configuration at the moment of session creation: a system administrator can adjust system-level configuration without invalidating decisions already made in sessions that are in flight.

Three schema choices on `session_items` encode design decisions that would otherwise require application-layer enforcement. The `tier` integer (0 = PROVISIONAL carry-forward, 1 = FLAGGED, 2 = SAMPLED, 3 = AUTO_ACCEPTED) means the queue manager can filter with `WHERE tier < 3` and sort with `ORDER BY tier ASC, confidence ASC`; ordering is a property of the data rather than of the query. The `carried_from` foreign key links a provisional carry-forward item to the original item in the previous session, preserving the audit chain across sessions for the same database. The `corrected_to` nullable field captures the reviewer-specified mapping when a CORRECTED decision is recorded; it forms the correction tuple for SR-F-04.

5.5.5 Minimum Oversight Floor

Motivation

SR-NF-06 requires that the supervision layer maintain a minimum sampling floor on high-confidence auto-accepted mappings, regardless of overall pipeline performance. Two independent arguments support this requirement.

The first argument is empirical. DataSpeck's evaluation of real-world data transformation execution found that 14% of auto-executed transformations were incorrect despite meeting the

system's confidence threshold [35]. A 14% error rate on auto-executed transformations means that a high confidence score is a probability estimate, not a correctness guarantee. An auto-accepted mapping is a mapping the algorithm believes to be correct with high probability; it is not guaranteed to be correct. Treating high confidence as a proxy for correctness, and therefore as grounds for removing human oversight entirely, conflates the two.

The second argument is domain-specific. Participant 2 identified a failure mode that confidence-score monitoring cannot detect: semantic drift. In energy data standardisation, field names in source databases are stable: they are defined by equipment suppliers and rarely renamed across successive dataset versions. The real-world meaning of a field can, however, change without a corresponding name change: a measurement unit may be silently redefined, a sensor replaced with one operating on a different scale, or a convention reversed. The AS model has no signal that the semantics of a stable field name have changed; its confidence score for the mapping will remain high precisely because the field name is unchanged. Within the supervision layer, periodic spot-checking of high-confidence outputs is the only available signal for this failure mode before it propagates into the training data.

The two arguments are complementary. DataSpeck grounds the floor empirically: at least one evaluated system failed at a measurable rate despite meeting its confidence threshold. The interview grounds the floor architecturally: it points to a category of failure that the supervision layer's metric-based signals do not capture, leaving human inspection as the remaining check within the layer.

Configuration Model and Enforcement

The floor combines three environment variables (`routing_threshold_auto`, `oversight_min_sample_rate`, and `oversight_min_sample_count`) as specified in Section 5.4.2: the maximum of a proportional rate and an absolute minimum enforces both at once, so a system administrator sets the policy but cannot take the floor below its hard lower bound [4]. The reviewer has no override capability; only a system administrator with access to the deployment environment can change these values.

`oversight_min_sample_count` ≥ 1 is a hard constraint validated at system startup. A value of zero would nullify the absolute floor entirely.

When a reviewer attempts to submit a session via `POST /sessions/{id}/submit`, the submission guard checks two structural constraints. First, every SAMPLED item must carry a decision that is neither NULL nor SKIPPED, the SR-NF-06 oversight floor. Second, every Tier 0 carry-forward item must carry a non-null decision, so the carry-forward resolution chain (Section 5.5.6) cannot be silently broken by submitting a session that leaves a previously-deferred item unresolved. If either constraint is violated, the endpoint returns HTTP 422 with a response body that identifies the blocking items by ID. There is no flag or parameter that bypasses either check.

The submission guard interaction (both the HTTP 422 failure branch and the successful submission path) is shown in Figure 5.6.

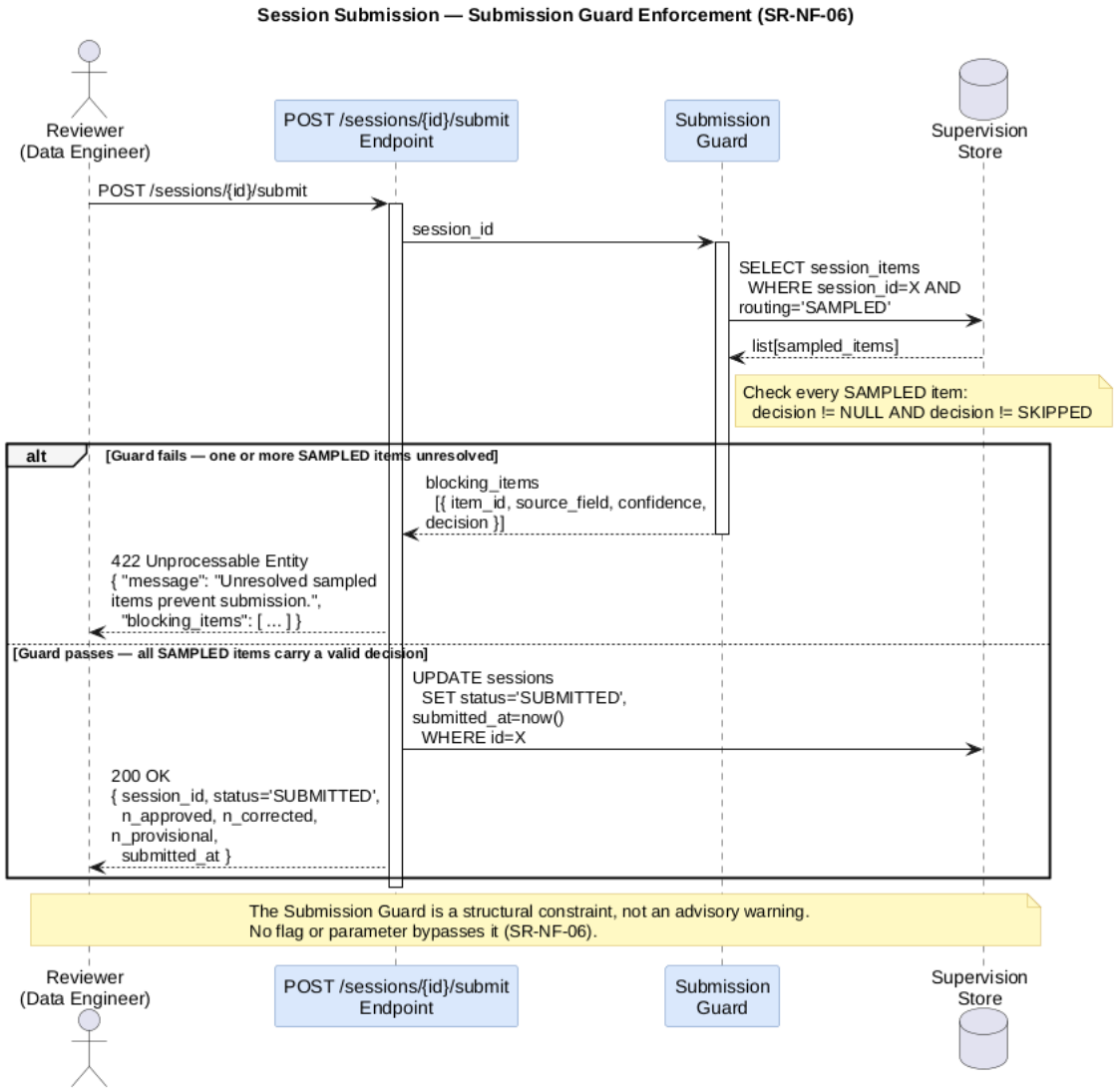

Figure 5.6: Submission guard enforcement on `POST /sessions/{id}/submit`.

5.5.6 Reviewer Interface Design

Queue Ordering

Items in a session are presented in a fixed three-tier order determined by two columns: `tier` (INT, set at session creation) and `confidence` (FLOAT, from the AS output). The database query `ORDER BY tier ASC, confidence ASC` produces the canonical queue order. The order is deterministic: the reviewer always opens a session to the same queue.

Tier 0, provisional carry-forwards (ascending by original confidence): items carried forward from a previous session for the same database appear first. Surfacing them at the top ensures the reviewer does not close a new session without addressing outstanding provisional items.

Tier 1, newly flagged items (ascending by confidence): items whose confidence fell below the threshold in the current job, ordered from most uncertain to least uncertain. Presenting the most uncertain cases first directs review effort to where the algorithm is least confident, the items on which reviewer judgement adds the most.

Tier 2, mandatory sampled items (ascending by confidence): high-confidence items drawn from the auto-accepted pool as mandatory spot-checks. Their role is coverage of the high-confidence pool rather than adjudication of uncertainty, and they are placed after the flagged items for two reasons: a spot-check of a confident suggestion is assessed against the schema-wide context the reviewer has accumulated while working the uncertain cases, and that context is what makes an implausible high-confidence mapping recognisable across the database as a whole.

CPFilter formalised reviewer attention as a finite resource and showed that workload reduction depends on which items absorb that attention first [14]; the three-tier order is a direct application of this principle, directing the scarcest attention to the most uncertain mappings.

Information Panel

Each item in the queue is accompanied by an information panel providing the evidence a reviewer needs to evaluate the algorithm's primary suggestion and its alternatives. The panel content follows the five cue types identified empirically by CopycHats as decision-relevant [43]: (1) attribute name, (2) data type, (3) instance values, (4) hierarchy position in the target standard ontology, and (5) sibling concepts adjacent to the candidate in the ontology. Cues (1)–(3) populate the source-field metadata panel (Figure 5.8, left); cues (4) and (5) accompany each candidate in the ranked list when the candidate JSON carries the `hierarchy_path` and `sibling_concepts` fields, which depend on the ontology lookup service architecture-specified in Section 5.7. For each candidate in the ranked list, the panel also displays the standard name, formal definition, and confidence score. The ranked list is presented in descending confidence order; CopycHats found that the order in which mapping candidates are shown to reviewers affects their decisions [43], so the panel fixes a deterministic candidate order rather than leaving it unspecified. Where the AS provides a `rationale` field, it is rendered beneath the confidence score as supplementary context.

Balancing Privacy established that curated per-signal metadata produces statistically equivalent review accuracy to showing the full schema while actively reducing reviewer overconfidence [33].

The panel therefore does not expose the full ontology or the raw source schema. Batman & Robin found that the evidence panel is load-bearing for reviewer performance [3], which is the design justification for treating the panel as a required component rather than an optional enhancement.

Provisional Carry-Forwards

Carry-forward items are visually distinguished by a **[⋆ PROVISIONAL]** tier badge and a reference to the originating session identifier. The badge communicates that the item was previously deferred, which is relevant context: a carry-forward that was marked PROVISIONAL because the reviewer was uncertain in the prior session warrants closer attention than a newly flagged item at the same confidence level.

Carry-forward items cannot receive a SKIPPED decision. A SKIPPED decision means "defer within the current session"; for a newly flagged item this is acceptable because the item will still be present in the same session. For a carry-forward, a SKIP at submission would remove it from the queue with no mechanism to re-surface it in a future session, silently breaking the carry-forward resolution chain. The reviewer must therefore resolve each carry-forward with APPROVED, CORRECTED, or a new PROVISIONAL decision before the session can be submitted. A new PROVISIONAL decision on a carry-forward will resurface it again in the next session for the same database.

Session Controls

Three persistent controls are available throughout an active session. The **progress indicator** displays the count of reviewed and total items, broken down by tier, allowing the reviewer to assess how much mandatory-sample work remains without navigating to the bottom of the queue. The **Pause** button triggers the ACTIVE $\rightarrow$ PAUSED transition immediately with all decisions persisted. The **Submit** button is enabled only when all SAMPLED items have a non-null, non-SKIPPED decision; if the Submission Guard finds blocking items, the interface displays their identifiers and the button remains disabled. On SAMPLED items the per-item Skip control is omitted, so the minimum oversight floor cannot be bypassed from the interface.

5.6 Profile Governance Service

Role: Derives versioned mapping profiles from submitted session decisions and manages their lifecycle through a role-gated approval chain; detects automatically when new AS job outputs diverge from an approved profile.

Anatomy:

- *Profile Builder*: reads all `session_items` (tiers 0–3) from a SUBMITTED session; creates one `ProfileEntry` per item with the approved or corrected mapping; excludes PROVISIONAL and SKIPPED decisions.

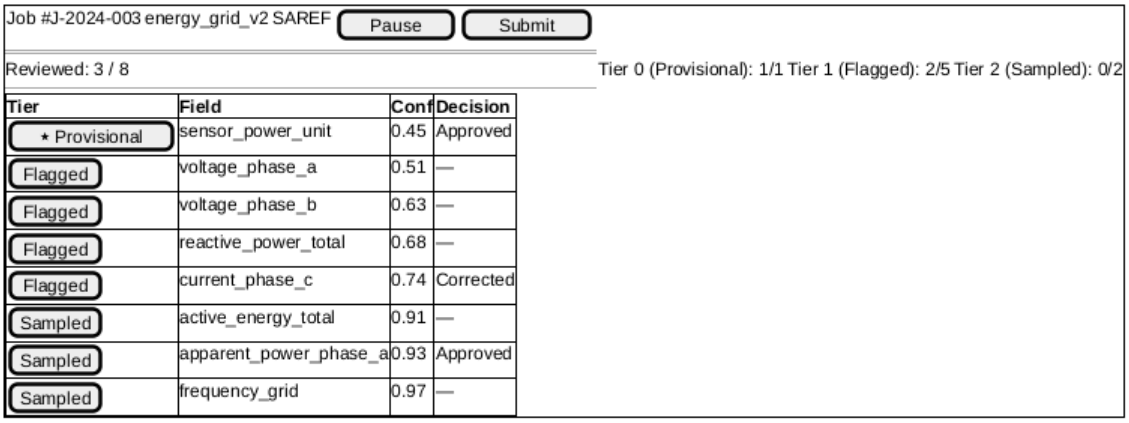

Figure 5.7: Reviewer interface: queue view.

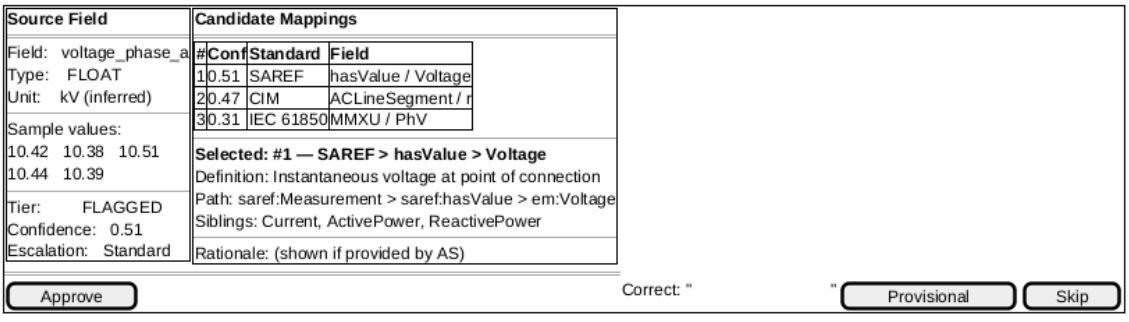

Figure 5.8: Reviewer interface: item detail view.

- *Lifecycle Transitions*: enforces DRAFT → PROPOSED → APPROVED → SUPERSEDED; returns HTTP 409 if a DRAFT or PROPOSED profile already exists for the database; sets previous APPROVED version to SUPERSEDED on new approval.
- *Role Guard*: FastAPI dependency injected on governance endpoints; `DATA_ENGINEER` required for `POST /profiles`, `POST /profiles/{id}/propose`, and `POST /drift-alerts/{id}/acknowledge`; `GOVERNANCE_LEAD` required for `POST /profiles/{id}/approve` and `POST /profiles/rollback`.
- *Drift Detector*: runs at ingestion; compares the top AS candidate per field against the corresponding `ProfileEntry`; raises `DriftAlert(STANDARD_MISMATCH)` if target standards differ; raises `DriftAlert(FIELD_DRIFT)` if the field-level divergence rate exceeds `drift_alert_threshold`.
- *Rollback Handler*: status update only, no new rows; moves target version from SUPERSEDED to APPROVED and current APPROVED to SUPERSEDED; version sequence remains monotonically increasing per database.

The internal component structure is shown in Figure 5.9.

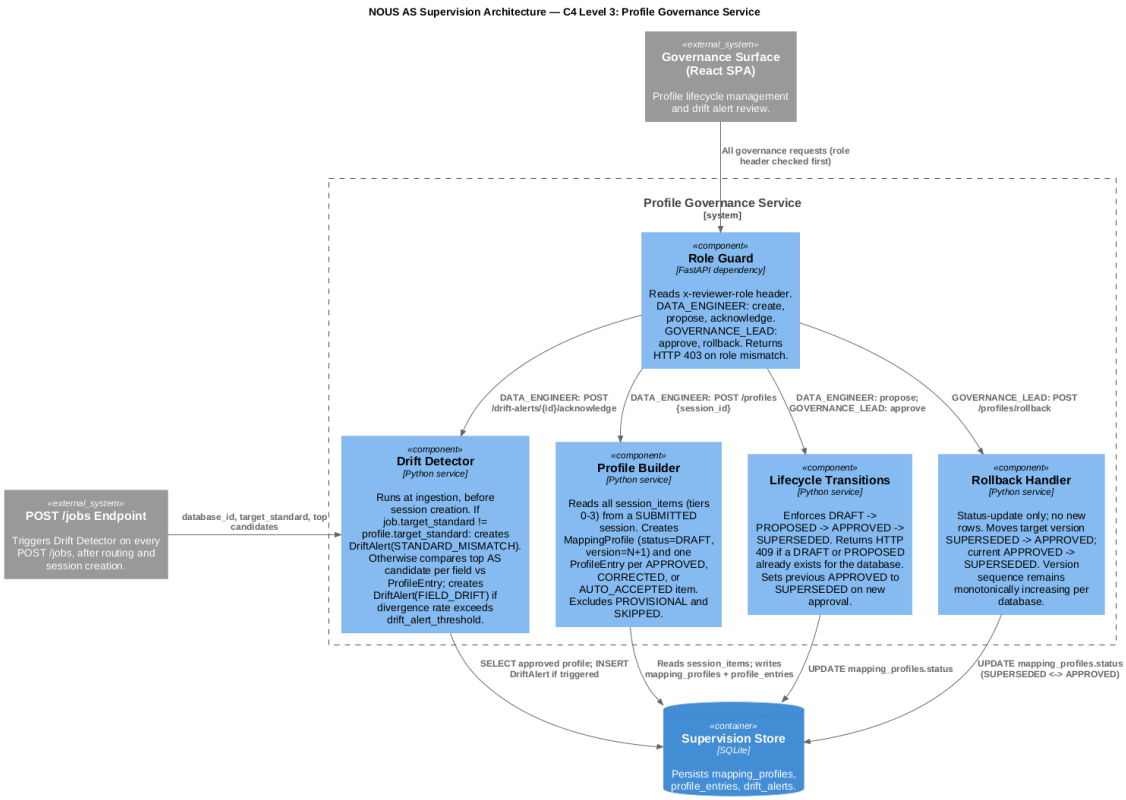

Figure 5.9: C4 Level 3 (Profile Governance Service).

5.6.1 Motivation

SR-F-06 and SR-F-07 address mapping profile versioning and rollback. The requirements analysis in Section 4.5 established that this is among the deepest structural mismatches between existing HITL architectures and the data standardisation domain: the concept of a versioned, governed mapping profile (a reusable artefact derived from session decisions and subject to a formal approval lifecycle) is absent from the entire sixteen-paper PRISMA corpus and from Maldonado's framework [25] (the commercial governance platforms surveyed in Section 3.4 manage versioned data assets, but none of them governs mapping profiles as governed artefacts). Approved decisions from a submitted session represent more than a completed review; they constitute a statement of correct field mappings for a specific source database, which downstream processes depend on and which must be revisable when conditions change. Treating these decisions as ephemeral outputs, rather than as a governed artefact with a version history, removes the audit trail and rollback capability that a production governance process requires.

SR-F-08 addresses mapping output drift. Once an approved profile exists for a database, it defines what the correct mappings look like. Drift detection is triggered automatically at job ingestion, before the session is created, because the job boundary is the earliest point at which divergence from a current approved profile becomes observable. Waiting for a reviewer to notice divergence during a session review is a later and less reliable signal.

SR-F-09 and SR-NF-05 address role-based governance approval. The engineer who reviews items and captures corrections should not be the same person who approves the resulting profile for production use. This separation of duties is a standard governance control [11]. The implementation enforces it architecturally via the `x-reviewer-role` request header and Role Guard dependency described in the anatomy above, without requiring authentication infrastructure not present in the prototype.

5.6.2 Profile Lifecycle

A mapping profile passes through four statuses: DRAFT, PROPOSED, APPROVED, and SUPERSEDED.

A DRAFT profile is created by a `DATA_ENGINEER` via `POST /profiles`, supplying the ID of a SUBMITTED session. The Profile Builder reads all SessionItems in that session with a decision of APPROVED, CORRECTED, or AUTO_ACCEPTED (Tier 3), creates a `MappingProfile` row (status=DRAFT, version=max_existing_version+1 for the database), and creates one `ProfileEntry` row per qualifying item. PROVISIONAL and SKIPPED items are excluded: provisional decisions are explicitly deferred, and skipped items carry no reviewer intent. AUTO_ACCEPTED items are included despite not receiving direct reviewer attention, because the profile records how the entire database was mapped in this job, not only the reviewed subset. Their inclusion reflects the confidence-gated routing decision rather than a reviewer endorsement; the stored confidence value carries the basis on which each entry was accepted. A profile that excluded

them would cover only the reviewed minority and could not serve as the complete mapping record the database's downstream consumers depend on.

The constraint that at most one DRAFT or PROPOSED profile may exist per database at any time is enforced at creation: if a DRAFT or PROPOSED profile already exists, the endpoint returns HTTP 409. Auto-superseding the existing draft was not chosen because it would silently discard work in progress; a Data Engineer must explicitly resolve the conflict.

A `DATA_ENGINEER` promotes a DRAFT to PROPOSED via `POST/profiles/{id}/propose`. A `GOVERNANCE_LEAD` approves it via `POST /profiles/{id}/approve`, which sets the profile to APPROVED and sets the previously APPROVED version for the same database to SUPERSEDED; a `DATA_ENGINEER` who attempts the approval step receives HTTP 403. Rollback is available to `GOVERNANCE_LEAD` via `POST /profiles/rollback`: the target version moves from SUPERSEDED to APPROVED; the currently APPROVED version moves to SUPERSEDED. No new rows are created; the full version history is preserved.

Promotion of a new version covers the case where a later job produces a revised mapping for a database that already has an approved profile, whether to correct an earlier decision or to reflect a changed source schema. Because standardisation is governed by whichever version is APPROVED (Story 1.1, AC6) [30], entries the AS already materialised under the now superseded version are not rewritten by the supervision system. They remain the output of the version that produced them and are reconciled only when the source is re-standardised under the new active version, an operation internal to the AS behind the pull boundary described in Section 5.8. The supervision system owns the version lifecycle and the audit record of each promotion; it does not itself re-materialise data. Stamping each standardised batch with the profile version that produced it is what keeps already-translated entries traceable to the mapping that governed them.

The complete profile creation and governance approval sequence (from `POST /profiles` through DRAFT, PROPOSED, and APPROVED, including the Role Guard enforcement at each step) is shown in Figure 5.10.

5.6.3 Drift Detection

Drift detection runs as part of the job ingestion flow, after routing but before session creation. The Drift Detector loads the current APPROVED profile for the incoming job's `database_id`. If no APPROVED profile exists for that database, the check returns immediately with no alert.

The detector then compares the job's `target_standard` against the profile's `target_standard`. If they differ, the job is standardising the database to a different standard than the one the approved profile was built for. Field-level comparison in this case would be semantically meaningless: a SAREF `hasValue/Voltage` profile entry and a CIM `ACLineSegment/r` candidate share no common identifier space, so divergence counts would be uninterpretable. In this case the detector creates a `DriftAlert` with `alert_type = STANDARD_MISMATCH` and returns without field-level comparison.

When the standards match, the detector compares the top AS candidate for each item, `candidates[0]` by confidence rank, against the `ProfileEntry` for the same `field_id`.

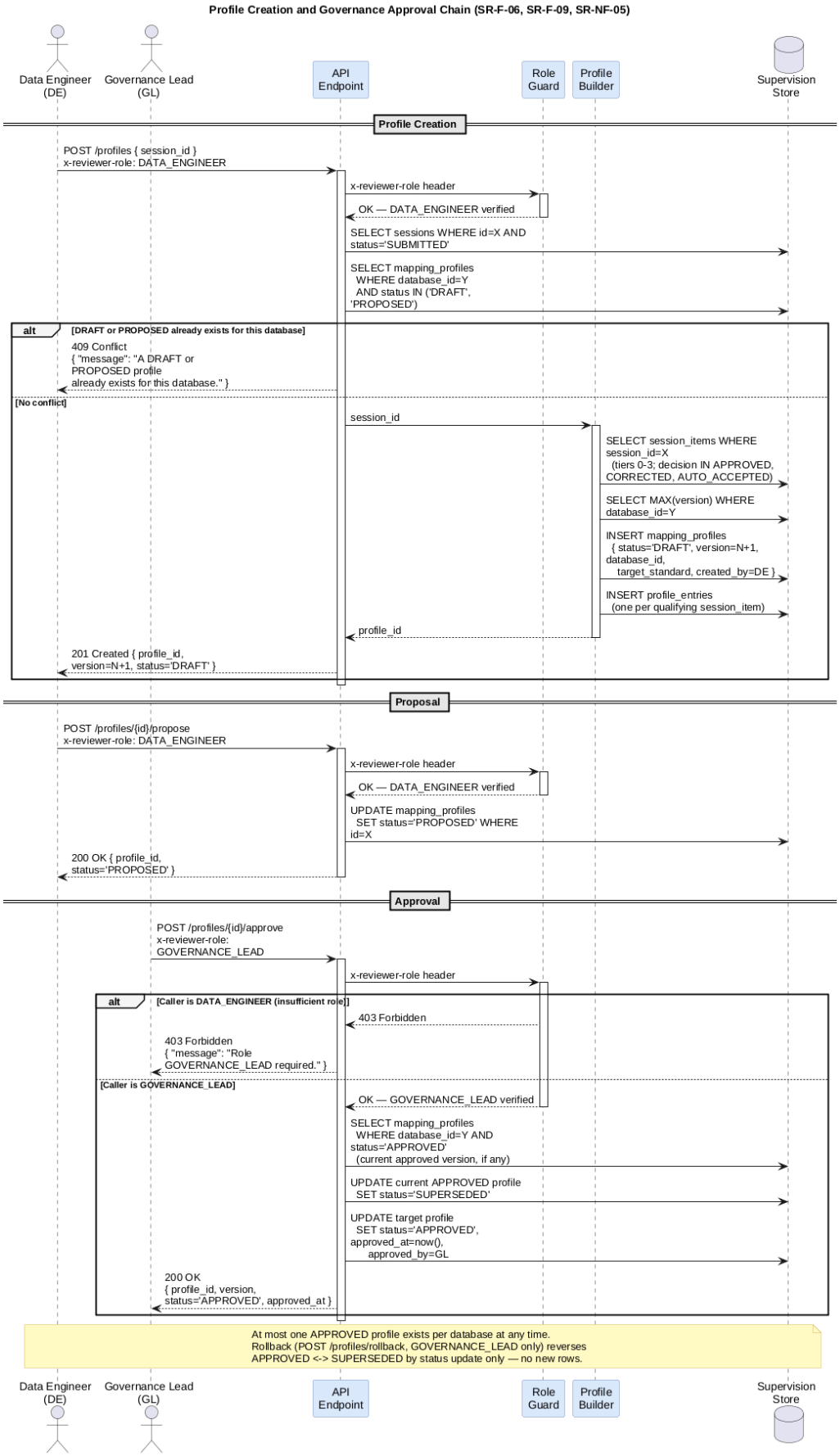

Figure 5.10: Profile creation and governance approval sequence.

A field is counted as diverging when the top candidate's `standard` and `field_name` differ from the profile entry's `approved_standard` and `approved_field_name`. If the fraction of diverging fields exceeds `drift_alert_threshold`, the detector creates a `DriftAlert` with `alert_type = FIELD_DRIFT` and returns its identifier in the `JobResponse`. The `JobResponse.drift_alert_id` field is non-null if and only if a DriftAlert was created for that job (DO-14).

`DATA_ENGINEER`s can retrieve drift alerts via `GET /drift-alerts` and acknowledge individual alerts via `POST /drift-alerts/{id}/acknowledge`. The Governance surface presents unacknowledged alerts alongside the profile detail view.

This mechanism operationalises the drift obligation of Story 1.1 AC7, which requires alerting when standardised output no longer adheres to the approved mapping profile [30]. Because adherence is defined against the profile, the check is a comparison of the incoming mapping against the approved mapping, and it needs only the job's candidates rather than the materialised output the supervision layer never receives. It therefore detects structural drift, where the AS re-maps a field or targets a different standard than the approved profile records. It does not detect semantic drift, where a field name and its mapping are unchanged but the meaning of the underlying values has shifted; that failure mode carries no signal in the mapping and is instead the responsibility of the minimum oversight sampling floor of Section 5.5.5, which spot-checks high-confidence outputs against the representative instance values the AS supplies with each field.

5.6.4 Data Model

Three tables support the profile governance service. `mapping_profiles` stores the profile header: `id`, `database_id`, `target_standard` (inherited from the session), `version` (integer, auto-incremented per database), `status` (DRAFT / PROPOSED / APPROVED / SUPERSEDED), `created_by`, `created_at`, `approved_by` (nullable), `approved_at` (nullable), and `parent_version_id` (nullable self-referencing foreign key). `profile_entries` stores one row per field mapping: `profile_id`, `field_id`, `source_field`, `approved_standard`, `approved_field_name`, and `confidence`. `drift_alerts` stores one row per detected drift event: `database_id`, `job_id`, `alert_type` (ENUM: `STANDARD_MISMATCH` or `FIELD_DRIFT`), `detected_at`, `diverging_fields` (JSON, populated for `FIELD_DRIFT` only), and `acknowledged` (boolean).

The `version` column is scoped to `database_id`: two different databases may each have a version 1. Rollback changes status only and does not reset version numbers. This preserves version history integrity: if version 3 is active and the Governance Lead rolls back to version 2, both records remain in the store and the sequence $1 \rightarrow 2 \rightarrow 3$ remains fully queryable for audit reconstruction. Resetting version numbers would either create gaps or reuse identifiers, making it impossible to reconstruct the approval sequence unambiguously.

5.6.5 Governance Interface Design

The Governance surface is the Governance Lead's counterpart to the reviewer interface (Section 5.5.6). Where the reviewer interface presents the session item queue, the Governance surface presents the two artefacts this service manages: the versioned mapping profiles and the drift alerts raised against them. It is part of the same single-page application and communicates exclusively with the FastAPI backend via the REST API.

The profile dashboard (Figure 5.11) lists the mapping profiles for the selected database with their version and lifecycle status, filterable by status and by database, so that the version history required by SR-F-06 and SR-F-07 is directly inspectable. Unacknowledged drift alerts (SR-F-08) are surfaced at the top of the dashboard, each carrying its type, the database and job that triggered it, and the diverging fields, with an acknowledgement control gated to the `DATA_ENGINEER` role.

The profile detail view (Figure 5.12) shows a single profile's position in the DRAFT → PROPOSED → APPROVED → SUPERSEDED lifecycle, its provenance (created and approved by whom, and when), and the profile entries with their source field, target mapping, and confidence. The lifecycle action valid for the current status is presented as a single control: **Propose for approval** on a DRAFT, **Approve** on a PROPOSED profile, and **Rollback** on an APPROVED one. The separation of duties required by SR-F-09 and SR-NF-05 is made observable rather than hidden: an action the current role may not perform is shown but disabled, with the required role indicated, so the governance control is visible in the interface itself rather than surfacing only as a rejected request.

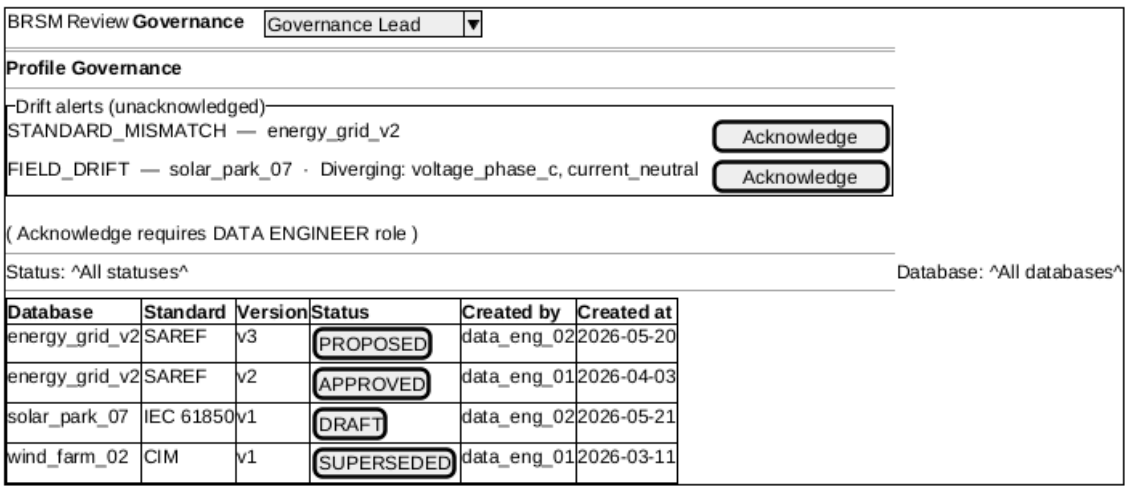

Figure 5.11: Governance interface: profile dashboard.

5.7 Architecture Specifications for Non-Implemented Components

The subsections below specify the architecture for components not implemented in this prototype. Five address Tier 3 supervision requirements (SR-F-05, SR-F-12, SR-F-13, SR-F-10/NF-04, SR-NF-02); a sixth covers the standard-aware ontology lookup service (SR-F-03), the deferred portion

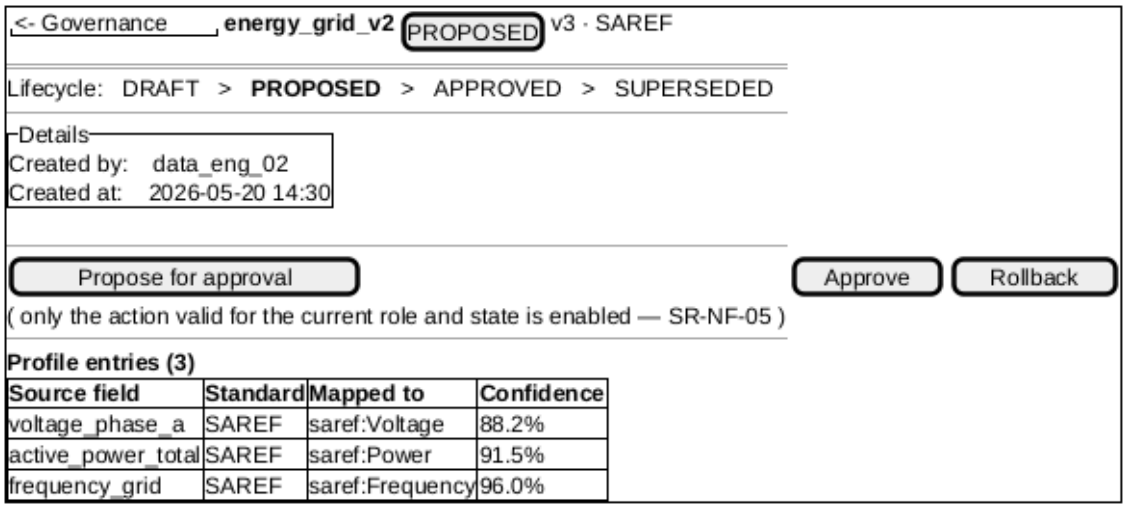

Figure 5.12: Governance interface: profile detail view.

of a Tier 2 partially-implemented requirement. Each subsection covers the component's role in the full architecture, its integration point with the implemented components, and the reason implementation was deferred.

5.7.1 Standard-Partitioned ML Retraining (SR-F-05)

The correction capture schema implemented in this thesis records reviewer-supplied corrections as a seven-field structured tuple assembled from the session and its items: `field_id`, `source_field`, `decision` (= CORRECTED), `corrected_to` (the reviewer's intended target mapping), and `reviewed_at` from the `session_items` row; plus `database_id` and `model_version` from the parent `sessions` row. The `model_version` field is populated from an optional field in `JobInput` and identifies the AS model version that produced the incorrect suggestion, enabling standard-partitioned retraining to attribute corrections to specific model releases. This field is the integration point for an ML retraining trigger. The architecture exposes corrections via a `GET /corrections` endpoint on the supervision system, filterable by `target_standard` and a `since` cursor (timestamp or session ID). The AS retraining pipeline queries this endpoint before each training run, pulling the corrections it has not yet consumed. The supervision system makes no outbound calls: it exposes corrections as data, and any consumer (the AS retraining pipeline, an audit tool, a monitoring system) can pull from the same endpoint without supervision system changes. Once pulled, corrections are partitioned by `target_standard` (SAREF, CIM, IEC 61850) and queued for batch model updates once a partition exceeds a configurable correction volume. A validation holdout drawn from each partition guards model promotion, so that a new model version is only deployed if its holdout accuracy exceeds the current model's on that standard's subset.

Implementation requires a live ML training pipeline that can consume structured correction tuples from the endpoint, retrain a standard-specific model, evaluate on a holdout set, and promote

a new model version without human intervention. None of these components exist in the NOUS prototype context; the mock Auto-Standardiser used in Chapter 7 produces synthetic confidence scores rather than a retrained model. The correction capture schema and the `GET /corrections` endpoint contract are in place so that this integration can be built without modifying the session data model or the supervision system's internal logic.

5.7.2 Algorithm Steering Execution (SR-F-13)

During a review session, a reviewer may supply one or more hints for an unmapped column, for example indicating that a field belongs to a particular SAREF concept class or IEC 61850 logical node, or that its source-column name needs translation before matching. A hint applies to a mapping profile rather than to a database alone, since a profile is a database and version pair and a later version may carry different hints; the specified data model therefore stores hints as a separate collection keyed to the profile and to the session item that produced them, each hint carrying its own identifier, type, parameters, and an `applied` flag set when the Auto-Standardiser consumes that individual hint. Because a single field may carry several hints, the `applied` state is recorded per hint rather than once per field. The prototype carries the simplified single-hint form of this model as the `hint_type`, `hint_parameters`, and `hint_applied` columns on session items: the columns and the integration endpoints are in place, but the reviewer-side capture control is not described in Section 5.5.6 and is deferred together with the rest of the SR-F-13 execution path. The architecture exposes outstanding hints via a `GET /hints/{database_id}` endpoint on the supervision system. Before generating candidates for a new job on a given database, the AS queries this endpoint, applies any outstanding hints to constrain its inference path or candidate ranking, then proceeds with `POST /jobs` as normal. After applying a hint, the AS calls `POST /hints/{id}/applied` with that hint's identifier to mark it consumed; the supervision system expires unconsumed hints after the first job processed for that database. As with corrections, the supervision system makes no outbound calls: it exposes hints as data and the AS is the active party in both directions.

Executing the hint requires AS-side implementation: the AS must query the hints endpoint before each job run, incorporate the hint payload into its inference path or post-processing step, and return updated candidates in the job output. Neither side of this integration can be completed without access to the AS source code and a deployed model. The hint capture schema and the `GET /hints/{database_id}` endpoint contract are the integration points; both are in place so that the AS-side implementation can proceed independently without modifying the supervision system.

5.7.3 Tiered Escalation Routing (SR-F-12)

SR-F-12 specifies a configurable three-tier sequential escalation ladder for items requiring human review: items at or above a first configurable confidence threshold (`routing_threshold_auto`) proceed to auto-acceptance; items below that threshold are routed to a junior reviewer pool for assessment within a session; when inter-reviewer disagreement on a flagged item exceeds a second

configurable threshold (`routing_threshold_expert`), the case escalates to a domain expert. The implemented Confidence Routing layer (Section 5.4) realises only the first tier boundary (the auto-accept threshold) and treats every below-threshold item as a single undifferentiated FLAGGED class. Tiers (2) and (3) are architecture-specified only. The full architecture follows a quality model pattern analogous to MExI [39]: each reviewer accumulates a quality score per standard domain, updated after each submitted session by comparing their decisions against a held-out expert reference. The routing engine would consult this quality model at ingestion to set per-reviewer escalation thresholds rather than a single system-wide value.

Three components absent from the prototype block implementation. First, a deployment in which multiple reviewers work the same session items, so inter-reviewer agreement on a flagged item is observable; the implemented session model assigns one reviewer per session. Second, a corpus of historical reviewer decisions sufficient to seed the quality model per reviewer per standard domain; the prototype operates in a zero-history context. Third, two schema extensions that the implemented data model does not carry: a `routing_threshold_expert` configuration value snapshotted onto each session record, and an `escalation_tier` column on `session_items` recording the tier assigned by the routing engine. With these additions, the routing engine's output interface extends with an `escalation_tier` field per `RoutingDecision`; the session manager and the item queue manager remain unchanged because tier assignment is purely a routing-side decision. The single-threshold engine in Section 5.4 is the substitution point for the full three-tier variant.

5.7.4 Audit Trail and Compliance (SR-F-10, SR-NF-04)

The supervision store persists all session state changes, item decisions, and profile lifecycle transitions as part of its normal operation, but this is not an audit log: rows can be updated, and there is no retention policy or access control log. A production audit trail would require an append-only log table with no UPDATE or DELETE operations permitted after initial insert, a configurable retention period aligned with the applicable data retention requirements, and an access log recording who queried which records and when.

The relevant legal grounding is established: EU AI Act Articles 9 and 12 mandate documentation and record-keeping for high-risk AI systems [11]; GDPR Articles 22 and 32 impose requirements on automated decision-making and appropriate technical safeguards. These requirements are well-covered by established audit log design patterns and do not constitute a structural design contribution of this thesis. Implementation was not prioritised because the distinctive architecture of this work lies elsewhere: in the batch session model, the minimum oversight floor, and the profile governance service. The integration point is the same persistence layer used by the implemented components; an audit log would wrap or shadow the existing SQLAlchemy session writes without changing any other component.

5.7.5 Post-Submission Accuracy Verification (SR-NF-02)

SR-NF-02 requires that the system verify an accuracy threshold of at least 80% through mandatory sampling before a batch's decisions are accepted downstream. After a session is submitted, all SAMPLED items carry reviewer decisions: APPROVED (the AS suggestion was correct) or CORRECTED (the suggestion was wrong). The accuracy metric for the session is therefore computable directly from the submitted session data as the fraction of SAMPLED items that received an APPROVED decision. No additional components are required; the submission guard already guarantees that all SAMPLED items have a non-null, non-SKIPPED decision at the point of submission.

The integration point is the session record: a post-submission service would read SAMPLED item decisions from the supervision store and write a computed `sample_accuracy` value to the session row, accessible via `GET /sessions/{id}`. Implementation was deferred for two reasons. First, with a minimum sample size of one item (`oversight_min_sample_count = 1`), a single-session accuracy estimate is statistically unreliable as a quality gate; the metric becomes meaningful only as a trend across multiple successive jobs for the same database. Second, the enforcement action when accuracy falls below 80% is undefined in the prototype context: whether to block downstream consumption, trigger additional sampling, alert a system administrator, or simply record the metric requires calibration decisions that depend on production data and domain expertise unavailable at prototype stage. The accuracy computation and storage are straightforward to add once those decisions are made, without modifying any other implemented component.

5.7.6 Standard-Aware Ontology Lookup Service (SR-F-03)

SR-F-03 requires that the review interface present standard-aware ontological context alongside each mapping candidate: specifically the hierarchy path of the candidate within the target standard and the sibling concepts adjacent to it in the ontology. The display layer implemented in Section 5.5 renders these fields when present in the `candidates` JSON structure on each SessionItem. The missing component is the service that populates them: given a field identifier and a target standard (SAREF, CIM, or IEC 61850), the ontology lookup service would query the appropriate ontology graph and return the canonical hierarchy path and a list of sibling concepts at the same level of the hierarchy.

The integration point is the `candidates` JSON column. At session creation, the Session Manager assembles the ranked candidate list from the AS job output; the ontology lookup service would enrich each candidate object with `hierarchy_path` and `sibling_concepts` fields before the list is written to the supervision store. The display layer already reads and renders these fields when present, treating them as optional in the current implementation for compatibility with AS outputs that do not include them. Implementation was deferred because no live ontology endpoint exists in the NOUS prototype context: SAREF, CIM, and IEC 61850 are available as published ontology graphs, but querying them programmatically requires either a deployed

SPARQL endpoint or a bundled local graph, neither of which is present in the T5.4 deployment environment used for evaluation.

5.8 Component API Specification

The supervision architecture specifies twenty-one REST endpoints. Eighteen are implemented and constitute the evaluation contract against which Chapter 7 assesses the design objectives. Three are integration-contract endpoints for SR-F-05 and SR-F-13 (correction export and algorithm steering hints): the endpoint shapes are implemented and reachable, but consuming them productively depends on AS-side work outside this prototype. The full endpoint table is given in Appendix B.1. Two worked examples illustrate the boundary conditions most relevant to the design objectives.

Submission Guard: Worked Example

The submission guard enforces SR-NF-06 at the API boundary (DO-6). The following example shows its behaviour when a reviewer attempts to submit a session with two unresolved mandatory-sample items.

Request:

```
POST /sessions/a3f1c2d4-.../submit
```

Response (HTTP 422 Unprocessable Entity):

```
{
  "detail": "Cannot submit: 2 SAMPLED items have no decision.",
  "blocking_items": [
    {
      "item_id": "7b2e4f1a-...",
      "source_field": "active_power_export",
      "confidence": 0.94,
      "routing": "SAMPLED",
      "decision": null
    },
    {
      "item_id": "c9d3a8e5-...",
      "source_field": "reactive_power_L1",
      "confidence": 0.91,
      "routing": "SAMPLED",
      "decision": "SKIPPED"
    }
  ]
}
```

A SKIPPED decision on a SAMPLED item is treated equivalently to a null decision: both are rejected. This is a structural constraint; no request parameter bypasses it.

Role Enforcement: Worked Example

The Role Guard enforces the `x-reviewer-role` header on governance endpoints (DO-11). The following example illustrates rejection of a profile approval request submitted with the DATA_ENGINEER role.

Request:

```
POST /profiles/d7e2b1f4-.../approve
x-reviewer-role: DATA_ENGINEER
```

Response (HTTP 403 Forbidden):

```
{
  "detail": "Role GOVERNANCE_LEAD required."
}
```

The same Role Guard dependency is applied to `POST /profiles/rollback` (GOVERNANCE_LEAD required) and to `POST /profiles`, `POST /profiles/{id}/propose`, and `POST /drift-alerts/{id}/acknowledge` (DATA_ENGINEER required).

JobInput Contract Requirements

Two fields in the `POST /jobs` payload carry stability invariants the supervision architecture relies on across multiple requests.

`database_id` must be a stable, unique identifier for the source database. The session store uses it as the grouping key for provisional carry-forwards, profile association, and drift detection. A caller that generates a new `database_id` for each job will produce orphaned sessions and will never trigger drift detection against an approved profile.

`field_id` must be a stable, unique identifier for a field within its source database. The profile store uses `field_id` as the join key between a session item and its corresponding profile entry during drift comparison. If the AS generates a new `field_id` for a field that was previously profiled under a different identifier, the Drift Detector will not recognise the field. `field_id` values should be derived from a stable schema property (e.g. column name, table-qualified column name) rather than a per-job UUID.

`target_standard` is normalised to uppercase on ingestion. The drift detector compares the normalised form against the profile's `target_standard`.

Intentional Exclusions

Authentication: The `x-reviewer-role` header is a plain string with no token validation. A production deployment would replace the header check with a JWT claim or RBAC middleware; this is not expected to affect any other component.

Rate limiting and deployment infrastructure: Out of scope for a prototype evaluated against a local mock API. These are operational concerns rather than architectural ones and do not affect the evaluability of the fourteen design objectives in Chapter 7.

5.9 Summary

Table 5.5 records the eighteen design decisions in the supervision architecture, each grounded in the requirements from Chapter 4 and in peer-reviewed literature. The six design principles from Section 5.2 shaped these decisions; they appear in the Rationale column to show the connection between domain values and specific choices.

Table 5.5: Supervision Architecture design decisions, their requirements, rationale, and grounding.

Decision	Requirement	Rationale	Grounding
REST API + two-surface React SPA as artifact form	DSR Stage 3	Contract evaluable independent of UI; governance surface adds role-gated operations without a second framework	[32]
Session as unit of work	SR-F-02	AS batch workflow; individual-item queues fragment inter-mapping context	Ch4 gap analysis (novel; no corpus or commercial precedent)
Routing engine as pure function, no DB writes	Architecture	Routing decisions are transient; `session_items` is the single source of truth; a separate routing log would create a second source that could diverge	Architectural decision; single source of truth principle
Sampling floor in routing engine, not session manager	SR-NF-06	The floor determines session composition before the session exists, a routing decision	SR-NF-06; [4] configurable-but-bounded
Five-state session lifecycle	SR-F-02	Pause/resume requires explicit persistent state; ARCHIVED defers to SR-NF-04 audit retention	SR-NF-04
PROVISIONAL decision type and carry-forward	SR-F-14	Deferred review without blocking the downstream pipeline; context preserved across jobs for same database	[35] Assuming tier analogue
Four-tier queue ordering (`tier < 3; ORDER BY tier ASC, confidence ASC`)	SR-NF-03	Finite attention applied to hardest cases first; carry-forwards surfaced before new items	[14]
Five-cue information panel	SR-F-03 (partial)	Empirically grounded content specification	[43]; [3]

(Continued on next page)

(Continued from previous page)

Decision	Requirement	Rationale	Grounding
Curated display, not full ontology	SR-F-03 (partial)	Curated metadata produces equivalent accuracy at lower cognitive load; reduces overconfidence	[33]
Configurable-but-bounded oversight floor	SR-NF-06	Operator sets thresholds and sample rate; floor cannot be zeroed; reviewer has no override	[4] configurable-α
HTTP 422 submission guard	SR-NF-06	Structural enforcement at the API boundary; no bypass parameter	SR-NF-06; [4] non-negotiable floor
AUTO_ACCEPTED items in profile at Tier 3	SR-F-06	SR-F-06 requires a governing artefact covering all decisions; auto-accepted items are the majority of the database's mappings; omitting them produces an incomplete record	SR-F-06; completeness requirement
Drift detection at ingestion (automatic)	SR-F-08	Job boundary is the earliest detection point; uncorrected drift reinforces incorrect mappings across successive sessions	SR-F-08; [35]
Single DRAFT or PROPOSED constraint per database	SR-F-06	Prevents concurrent governance conflicts; auto-superseding would silently discard work in progress	SR-F-06; explicit conflict resolution
Rollback as status update, no new rows	SR-F-07	Version sequence remains monotonically increasing; full history preserved without row duplication	SR-F-07; version history integrity
Role enforcement via `x-reviewer-role` header	SR-F-09, SR-NF-05	Separation of duties between reviewer and governance approver; prototype-appropriate; replaceable without schema changes	[11]
Pull model for correction export and algorithm steering hints	SR-F-05, SR-F-13	Supervision system makes no outbound calls; AS pulls from `GET /corrections` and `GET /hints/{database_id}`	Decoupling requirement; §5.7

(Continued on next page)

(Continued from previous page)

Decision	Requirement	Rationale	Grounding
Cross-standard job as STANDARD_MISMATCH, not field-level drift	SR-F-08	When `target_standard` differs from the approved profile's standard, field-level comparison is undefined; a distinct alert type surfaces the discrepancy without producing spurious drift counts	SR-F-08; profile `target_standard` invariant

The specification in this chapter covers two implementation tracks in Chapter 6: the routing engine and its refactor of the existing session factory, and the profile governance service, alongside the existing session management implementation. Chapter 6 builds to this contract and documents deviations from the specification that arise during implementation. Chapter 7 then evaluates the implemented components against all fourteen design objectives through two tracks: technical evaluation against the mock AS API, and practitioner validation with domain experts (Chapter 4). It frames these as the ex-post strand of a two-altitude evaluation and maps every requirement, implemented or specified-only, to the evidence it receives in Section 7.2.

Chapter 6

Implementation

6.1 Introduction

Chapter 5 specified the NOUS Auto-Standardiser supervision architecture: its component clusters, data model, lifecycle, and the fourteen design objectives the implemented components must meet. This chapter is the demonstration stage of the design-science process [32]: it builds that specification into a working artefact and shows it operating in its intended setting. It holds the build against the design of Chapter 5 section by section, shows that the engineering choices taken within the latitude the design leaves are reasonable, and establishes that the resulting system is a valid basis for the evaluation in Chapter 7. Where a passage would restate why the design is shaped as it is, it cites the section of Chapter 5 that settled the question rather than re-arguing it.

The implemented scope, fixed by the feasibility analysis of Chapter 4 and the design of Chapter 5, is three component clusters: Confidence Routing, the Batch Review Session Manager, and Profile Governance with its drift detector. Between them they realise the ten Tier 1 supervision requirements and, partially, the two Tier 2 requirements: standard-aware explainability (SR-F-03) and structured correction capture (SR-F-04). The components the thesis specifies but does not build are set out in Chapter 5 Section 5.7, and their exclusion from evaluation is detailed in Section 6.9.

The chapter then establishes the technology stack and persistence layer (Sections 6.2–6.3); walks the three backend clusters and their cross-cutting concerns, which form its core (Section 6.4); and describes the frontend, the mock Auto-Standardiser used in evaluation, and the testing strategy (Sections 6.5–6.7). Section 6.8 reports what building surfaced about the design, and Section 6.9 consolidates the build status against the fourteen objectives and hands the artefact to Chapter 7.

6.2 Implementation Approach

The supervision architecture specified in Chapter 5 is realised as a Python backend exposing a REST API, a TypeScript-based React single-page application, and a separate mock Auto-Standardiser service used in evaluation.

6.2.1 Technology stack

Table 6.1 lists the principal technologies and the architectural commitment in Chapter 5 that motivated each choice. The selection is deliberately conservative: each tool is in widespread use.

Layer	Technology	Architectural commitment from Ch. 5
REST surface	FastAPI	Dependency injection realises the Role Guard pattern (D3) without per-handler boilerplate
Boundary validation, settings	Pydantic v2	Separates the API request/response format from the ORM model; field validators enforce the SR-NF-06 hard floor at startup (Section 5.5.5)
ORM	SQLAlchemy 2.x	Typed `Mapped[...]` model definitions match the Pydantic schemas and reduce drift between the ORM and the request/response format
Persistence	SQLite	Single-file Supervision Store per D2; no external database service in the prototype
Schema versioning	Alembic	Profile governance treats schema lineage as a first-class artefact; the migration history is recorded evidence in its own right
Testing	pytest	Unit tests for pure functions and state machines; integration tests for objective-level scenarios through FastAPI's `TestClient`
Single-page application	React 18 + Vite + shadcn/ui	Two-surface SPA specified in Section 5.2.2; React Router separates the Review and Governance surfaces
API client integration	Shared `RoleContext` provider	Concentrates `x-reviewer-role` header injection in one module, consistent with D3

Table 6.1: Technology choices and the design commitment each one satisfies.

6.2.2 Code organisation

The implementation is split into three deployable units under `implementation/`: the FastAPI `backend/`, the React `frontend/`, and the standalone `mock_as/` used in evaluation. Within the backend, code is organised into four layers under `app/`, and the separation between them is deliberate. Route modules (`api/routes/`) are thin HTTP-boundary code: they parse the request, delegate to a service or a core function, and translate exceptions to HTTP status codes. Services (`services/`) hold orchestration logic and own the transaction scope through which an ingestion or governance action either commits as a whole or rolls back. Core modules (`core/`) hold the business rules, and split into two kinds. Some are pure functions over their inputs: the routing engine, the job translator, and the session state machine read and write no database at all, so their rules can be unit-tested without a fixture. The rest (the submission guard, the item queue manager, the drift detector, the profile builder, and the rollback handler) receive a database session as a parameter and query through it, but own no transaction: none of them commits, so the commit-or-rollback

boundary stays with the service caller. This split is a convention the modules follow rather than a constraint the language enforces, but it keeps the deterministic logic separable from transaction ownership. SQLAlchemy models (`db/models/`) and Pydantic schemas (`schemas/`) keep the persisted form separate from the API's request and response format, so model changes do not silently propagate to consumers.

6.2.3 Development workflow

Pure functions and state machines were written test-first: the routing engine, the session state machine, and the submission guard were each developed against unit tests for their boundary conditions before being integrated. Higher-level design objectives are exercised end-to-end through FastAPI's `TestClient`; Section 6.7 sets out the test layers and their per-objective coverage.

Schema changes are codified through Alembic, every migration committed alongside the model change that motivated it.

6.3 Persistence Layer and Migration History

The supervision system persists five tables across the two logical partitions identified in Section 5.2.2: the Session Decision Record (`sessions` and `session_items`) and the Profile Store (`mapping_profiles`, `profile_entries`, and `drift_alerts`). Both live in a single SQLite database in the prototype, consistent with design decision D2. Complete column definitions are in Appendix B.2; what follows is the migration history, where the sequence records the design questions the build resolved in code.

Eight Alembic migrations track the schema's evolution, in the three phases shown in Figure 6.1.

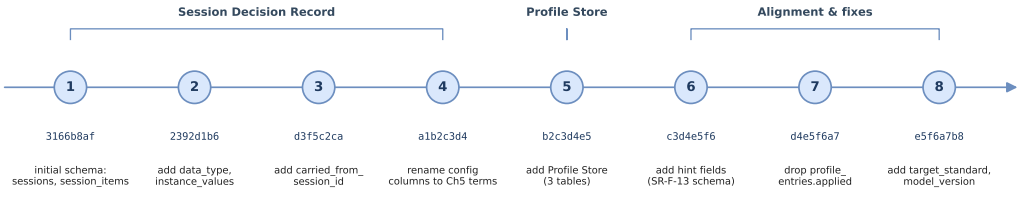

Figure 6.1: The eight Alembic migrations that track the schema's evolution.

6.3.1 Migration narrative

The initial schema (migration 1) created `sessions` and `session_items`, with the routing and decision enumerations, the snapshotted routing and sampling configuration, and the `carried_from` self-referencing foreign key that links a provisional item to its predecessor. Migrations 2 to 4 refined this record as the design settled: the `data_type` and `instance_values` cue columns once the information panel of Section 5.5.6 was fixed; an explicit `carried_from_session_id`

column to make the originating session directly queryable for audit reconstruction; and a rename of the configuration columns to the Chapter 5 vocabulary (`routing_threshold_auto`, `oversight_min_sample_rate`, `oversight_min_sample_count`). The rename is the one place SQLite's limitations surfaced: it has no `ALTER COLUMN RENAME`, so the migration rebuilds the table through Alembic's `batch_alter_table`, a pattern that recurs whenever a column is dropped or a non-nullable column added.

Migration 5 introduced the Profile Store as a unit, creating `mapping_profiles`, `profile_entries`, and `drift_alerts` together because the governance cluster references all three and they share the lifecycle and constraints of Section 5.6.2 and Section 5.6.3.

The final three migrations brought the schema into line with the specification. Two came from the findings of Section 6.8: the SR-F-13 hint columns (`hint_type`, `hint_parameters`, `hint_applied`) were added to `session_items`, and `profile_entries.applied` was dropped once `/hints` resolved to the algorithm-steering contract and left it with no producer or consumer. The last reconciled a different gap: the Session model declared `target_standard` and `model_version` as part of its configuration snapshot (Section 5.5.4), but no migration had created them, so every query projecting the full session row failed at runtime until the columns were added. It was the last migration applied before evaluation.

6.3.2 Modelling conventions

Two conventions run through the migrations. First, every status-bearing column uses a SQLAlchemy `Enum` rather than free-form text: `sessionstatus`, `routingtype`, `decisiontype`, `profilestatus`, and `alerttype` all sit in the database as named types, so the set of valid states is enforced at the persistence boundary rather than only at the application layer. Second, every primary key is a string UUID generated in Python at insert time. The choice keeps identifiers portable across SQLite and any production-grade database that the Supervision Store might later be moved to, and it avoids the implicit ordering coupling that an auto-incrementing integer key would create.

6.4 Backend Implementation

The three clusters are built on the layered structure of Section 6.2. The subsections below take them in the order of Chapter 5, from routing through session management to profile governance, closing with the cross-cutting concerns the three share (Section 6.4.4).

6.4.1 Confidence Routing Layer

The Confidence Routing layer is the supervision system's entry point: every job arriving at `POST /jobs` passes through it before any session row is created. It spans two files. `app/core/job_translator.py` is the Anti-Corruption Layer; it converts each AS-produced `JobItem` Pydantic object into an internal dictionary, sorts the candidate list in descending

confidence order, and recomputes the item-level confidence from the highest-ranked candidate so the rest of the system does not have to trust the AS to honour either invariant. `app/core/routing_engine.py` is the routing engine proper: a single pure function, `route(items, config)`, that takes the normalised dictionaries and a `RoutingConfig` value and returns a list of `RoutingDecision` dataclasses.

Keeping the engine a pure function follows Chapter 5's single-source-of-truth principle: routing decisions are captured as columns on session items by the Session Manager (Section 6.4.2), so the engine writes nothing itself. The side-effect-free shape also lets the unit tests cover threshold classification, the sampling floor, and the clamp behaviour without a database fixture, and lets the engine run at evaluation time against synthetic mock-AS inputs with no setup beyond constructing them.

The `RoutingConfig` and `RoutingDecision` types are Python dataclasses rather than Pydantic models. They are value types passed between modules in the same process and are never serialised to JSON for the API, so the validation cost of Pydantic would be pure overhead. `RoutingDecision` carries the field identifier, source field name, data type, instance values, confidence, candidate list, a `routing_type` of either `"FLAGGED"` or `"AUTO_ACCEPTED"`, and the `sampled` flag.

The algorithm follows the specification in Section 5.4.2. Each incoming item is classified as `FLAGGED` if its confidence falls below `routing_threshold_auto` and as `AUTO_ACCEPTED` otherwise. After the first pass, the size of the AUTO_ACCEPTED pool drives the sampling floor:

```
n_sample = max(
    config.oversight_min_sample_count,
    math.ceil(
        len(auto_accepted_raw)
        * config.oversight_min_sample_rate
    ),
)
sampled_items = random.sample(
    auto_accepted_raw, min(n_sample, len(auto_accepted_raw))
)
```

The inner `max(...)` reproduces the spec's two-term floor: a proportional rate that scales with job size and an absolute minimum count that prevents the floor from collapsing to zero on small jobs. The outer `min(n_sample, len(auto_accepted_raw))` is the clamp the spec adds explicitly for the edge case in which the AUTO_ACCEPTED pool is smaller than the floor would otherwise demand: instead of raising, the function samples every available high-confidence item, degrading gracefully on small jobs and producing an empty sample when no items clear the threshold at all.

Items selected by the sampling floor have their `routing_type` flipped to `"FLAGGED"` and their `sampled` flag set to `True`; non-selected items keep `routing_type = "AUTO_ACCEPTED"` and `sampled = False`. This two-field encoding matches the algorithm in Section 5.4.2; how the Session Manager maps it to a persisted tier is described in Section 6.4.2.

Randomness is concentrated in the single `random.sample(...)` call. Unit tests that need deterministic threshold behaviour disable sampling by setting `oversight_min_sample_rate = 0.0` and `oversight_min_sample_count = 0`, after which the function is fully deterministic; the test that confirms classification is stable across calls does exactly this.

6.4.2 Batch Review Session Manager

The Session Manager implements Section 5.5: it builds the session row and its item queue from the routing decisions, enforces the five-state lifecycle, gates submission through the dual-constraint guard, and serves the reviewer-visible queue in the order DO-4 specifies. The cluster spreads across five files in `app/core/` and one route module.

`app/core/session_state_machine.py` encodes the lifecycle as a data structure rather than a series of conditionals. A module-level dictionary maps each `SessionStatus` to the set of statuses it can transition to, and a single `transition()` function takes a session and a target status, validates the transition against the table, raises `InvalidTransitionError` on rejection, and sets the appropriate timestamps as a side effect:

```
VALID_TRANSITIONS: dict[SessionStatus, set[SessionStatus]] = {
    SessionStatus.PENDING:   {SessionStatus.ACTIVE},
    SessionStatus.ACTIVE:    {
        SessionStatus.PAUSED,
        SessionStatus.SUBMITTED,
    },
    SessionStatus.PAUSED:    {SessionStatus.ACTIVE},
    SessionStatus.SUBMITTED: {SessionStatus.ARCHIVED},
    SessionStatus.ARCHIVED:  set(),
}

def transition(session: Session, target: SessionStatus) -> Session:
    if target not in VALID_TRANSITIONS[session.status]:
        raise InvalidTransitionError(
            f"Cannot transition {session.status} -> {target}"
        )
    session.status = target
    if target == SessionStatus.ACTIVE and session.opened_at is None:
        session.opened_at = datetime.utcnow()
    if target == SessionStatus.SUBMITTED:
        session.submitted_at = datetime.utcnow()
    return session
```

The pattern has three useful properties. The transition table is the complete specification of valid moves and maps directly onto the lifecycle table in Section 5.5.2, so its correctness can be verified by inspection. Adding a state means editing one dictionary entry rather than several `if`/`elif` branches. And `transition()` is the only place that mutates session status, so the routes for pause, resume, submit, and archive each reduce to a single function call followed by a commit.

`app/core/session_manager.py` is responsible for building the Session row and the SessionItem rows from the routing decisions and the carry-forward list. The tier mapping checks the `sampled` flag first, then `routing_type`, matching the priority order specified in Section 5.5.2: any item with `sampled = True` becomes SAMPLED at tier 2 regardless of its routing type; otherwise, FLAGGED items go to tier 1 and AUTO_ACCEPTED items to tier 3. Carry-forward items are added in a separate loop that preserves the original item's `routing` value and sets `tier = 0`, satisfying DO-7. The Session Manager does not commit the transaction; its caller in `services/job_ingestion.py` is the transaction owner so that routing, drift detection, and session creation either succeed together or roll back together.

`app/core/item_queue_manager.py` implements DO-4 declaratively. A single SQL query (`WHERE tier < 3 ORDER BY tier ASC, confidence ASC`) produces the reviewer-visible queue. Tier 3 (AUTO_ACCEPTED items persisted for profile creation but invisible to reviewers) is filtered out at the persistence boundary rather than in application code, so a future view that exposes a different tier set does not require visiting application logic.

`app/core/submission_guard.py` enforces both halves of the guard described in Section 5.5.5: any SAMPLED item with a NULL or SKIPPED decision blocks submission, and any Tier 0 carry-forward with a NULL decision blocks submission. The two checks are combined into a single SQLAlchemy expression:

```
blocking = (
    db.query(SessionItem)
    .filter(
        SessionItem.session_id == session_id,
        or_(
            and_(
                SessionItem.routing == RoutingType.SAMPLED,
                or_(SessionItem.decision == None,
                    SessionItem.decision == DecisionType.SKIPPED),
            ),
            and_(SessionItem.tier == 0,
                 SessionItem.decision == None),
        ),
    )
    .all()
)
```

Composing the two checks in a single query keeps the guard atomic: the same database snapshot decides both arms, and the returned list of blocking items is the single source of the HTTP 422 response body. The guard raises `SubmissionBlockedError`, which the route handler in `app/api/routes/sessions.py` catches and translates to HTTP 422 along with the blocking-item details.

Finally, `app/api/routes/items.py` exposes `PATCH /sessions/{id}/items/{item_id}` and enforces two structural rejections at the boundary: SKIPPED is forbidden on SAMPLED items and Tier 0 carry-forwards (consistent with the chain-integrity rule in

Section 5.5.6), and CORRECTED is forbidden without a non-null `corrected_to`. Both return HTTP 422 with a short detail message. Decisions that pass the checks set the `decision`, `corrected_to`, and `reviewed_at` fields and commit immediately, because each item decision is the smallest reviewer-visible unit and the pause/resume contract specified in Section 5.5.2 requires that prior decisions be persisted even if the session is interrupted mid-review.

6.4.3 Profile Governance Service and Drift Detector

The Profile Governance cluster implements Section 5.6: it derives versioned mapping profiles from submitted session decisions, enforces a role-gated lifecycle, detects drift at job ingestion, and supports rollback to a prior approved version. The cluster spreads across four files in `app/core/` and `app/services/`, plus one route module.

`app/services/profile_service.py` orchestrates the three-stage governance lifecycle: `create_draft`, `propose`, and `approve`. The single-DRAFT-or-PROPOSED constraint is enforced as a pre-insert query: the service looks up any existing profile for the database whose status is either `DRAFT` or `PROPOSED`, and if one is found, raises `ProfileConflict`, which the route handler translates to HTTP 409. The same constraint is also enforced at the database level: a partial unique index over `database_id` (created with the governance tables) for profiles in DRAFT or PROPOSED status admits at most one open profile per database. The pre-insert query gives the common case a clean 409, while the index is the structural backstop for true concurrency, where two otherwise-simultaneous draft creations could each pass the application check; the creation that loses the race surfaces the index violation, which the service converts to the same `ProfileConflict`. The guarantee therefore does not rest on SQLite's serialised writes and carries to a concurrent production engine, since partial unique indexes are supported by both SQLite and PostgreSQL.

Approval is the most consequential of the three transitions and acts on two profile rows at once: the target profile moves from `PROPOSED` to `APPROVED`, and any current `APPROVED` profile for the same `database_id` is moved to `SUPERSEDED` in the same transaction. The atomic two-row update preserves the invariant that at most one APPROVED profile exists per database at any moment, an invariant the drift detector relies on.

`app/core/profile_builder.py` is the pure function that converts a SUBMITTED session into the list of entry dictionaries the service then persists. It iterates over every `SessionItem` in the session, excludes PROVISIONAL and SKIPPED decisions, includes APPROVED and CORRECTED items at tiers 0–2, and includes tier-3 AUTO_ACCEPTED items even though they carry no reviewer decision. The inclusion of AUTO_ACCEPTED items satisfies DO-10 directly: the resulting profile is a complete record of how the database was mapped, not only of the reviewed subset.

`app/core/rollback_handler.py` implements DO-12. The handler locates the target SUPERSEDED version by `database_id` and `version`, locates the currently APPROVED profile, and toggles their statuses in place without creating any new rows. The version sequence remains monotonically increasing per database and the full audit chain remains queryable.

`app/core/drift_detector.py` is the only governance-cluster component that runs at job ingestion rather than in response to a governance action. It is invoked by `services/job_ingestion.ingest_job` after routing but before session creation, consistent with Section 5.6.3, and returns the identifier of any drift alert it raised so the calling service can pass the identifier into the `JobResponse` and satisfy DO-14. The detector's logic is a two-arm decision tree:

```
if target_standard.upper() != approved.target_standard.upper():
    alert = DriftAlert(
        id=str(uuid.uuid4()),
        database_id=database_id, job_id=job_id,
        alert_type=AlertType.STANDARD_MISMATCH,
        detected_at=datetime.utcnow(),
        diverging_fields=[],
    )
    db.add(alert); db.flush(); return alert.id

# standards match -> field-level comparison
diverging_field_ids: list[str] = []
matched_count = 0
for item in items:
    entry = profile_entries.get(item.field_id)
    if entry is None:
        continue
    matched_count += 1
    if not item.candidates:
        continue
    top = item.candidates[0]
    if top.get("standard") != entry.approved_standard \
       or top.get("field_name") != entry.approved_field_name:
        diverging_field_ids.append(item.field_id)

if matched_count and (
    len(diverging_field_ids) / matched_count
    > settings.drift_alert_threshold
):
    alert = DriftAlert(
        ..., alert_type=AlertType.FIELD_DRIFT,
        diverging_fields=diverging_field_ids,
    )
    db.add(alert); db.flush(); return alert.id
```

The early return on STANDARD_MISMATCH is the implementation of an explicit design decision (Section 5.6.3): when the standards differ, the field-level comparison is undefined because a SAREF entry and a CIM candidate share no common identifier space, so the detector raises a distinct alert type with `diverging_fields = []` and never attempts the comparison loop. The detector flushes the alert but never commits; the ingestion service owns the surrounding transaction. Figure 6.2 shows the branching in the context of the full ingestion flow.

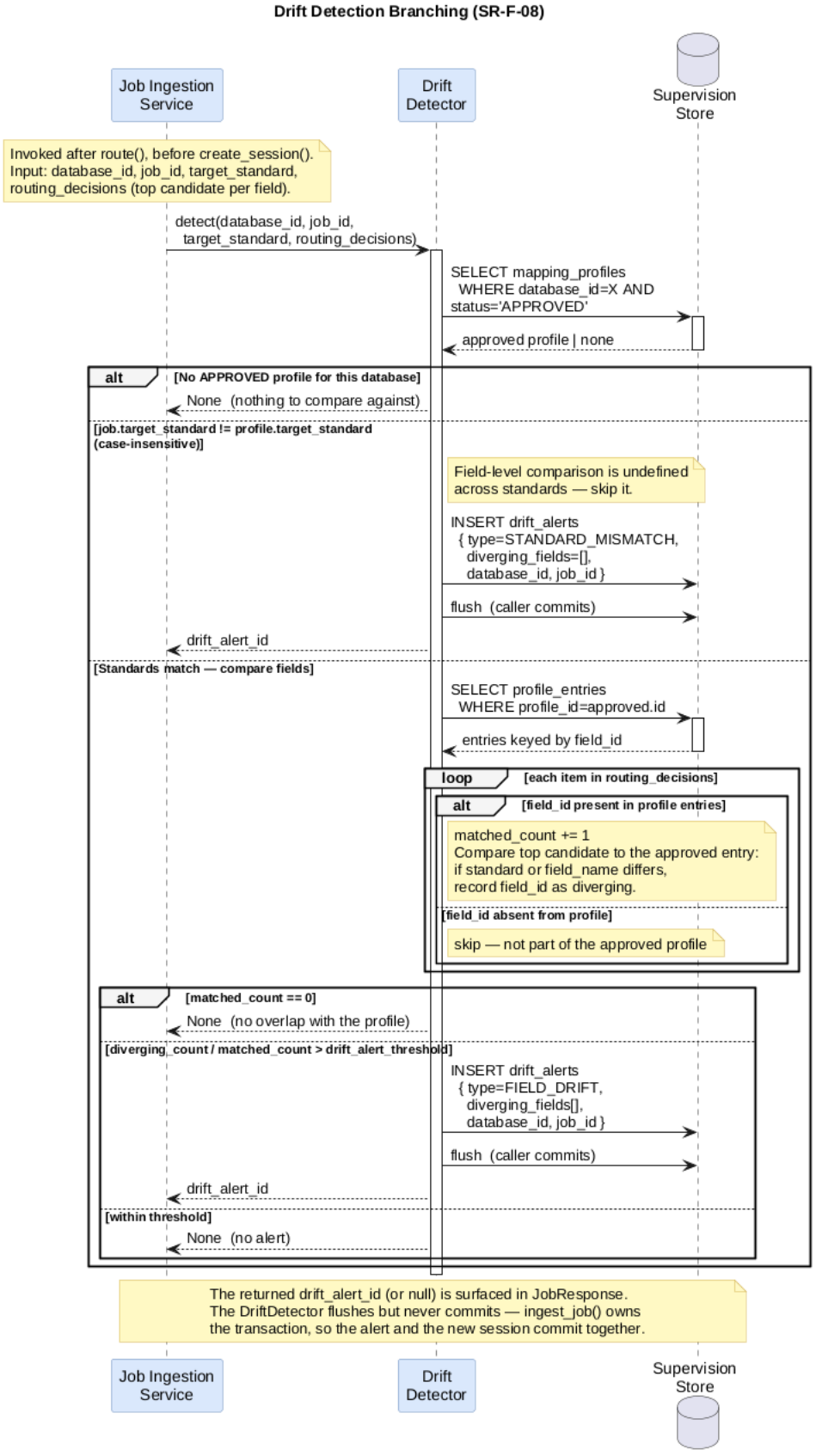

Figure 6.2: Drift detection branching at job ingestion.

`app/api/routes/profiles.py` is the route module for the governance cluster. Each transition endpoint is reduced to three lines: declare the role dependency, call the service, commit. The Role Guard described in Section 6.4.4 is injected per endpoint with `Depends(role_required("DATA_ENGINEER"))` or `Depends(role_required("GOVERNANCE_LEAD"))`, so the routes contain no role-checking code of their own. Business-logic exceptions raised by the services (`ProfileNotFound`, `ProfileConflict`, `ProfileTransitionError`, `RollbackError`) are caught at the boundary and translated to HTTP 404, 409, or 422 as appropriate, following the convention discussed in the next section.

6.4.4 Cross-cutting concerns

Three patterns recur across the route, service, and core layers: the Role Guard dependency factory, the configuration validators in Pydantic Settings, and the mapping of business-logic exceptions to HTTP responses.

Role Guard as a dependency factory

The Role Guard is implemented as a dependency factory in `app/api/dependencies/role_guard.py`. The factory takes a role string and returns a FastAPI dependency that reads the `x-reviewer-role` header on every request, compares it to the required role, and raises HTTP 403 on mismatch.

Two properties matter. Every protected route becomes a single line (`role: str = role_required("DATA_ENGINEER")` in the function signature) that concentrates the header-name decision in one place. The choice also gives effect to design decision D3, which deliberately defers authentication infrastructure: replacing the header check with a JWT-claim check requires editing only this file, with no impact on the routes or services that depend on it. The intentional exclusion of authentication complexity is recorded at the API-contract level in Appendix B.1 and at the implementation level here.

Configuration validators

The configuration layer in `app/config.py` uses Pydantic's `BaseSettings` with `field_validator` decorators that run at startup. Two validators are present and both are grounded in the design. The first cashes out Section 5.5.5's "hard constraint validated at system startup" claim by rejecting any `oversight_min_sample_count` value less than one, raising a `ValueError` that prevents the application from starting at all under a misconfigured environment. The second constrains `routing_threshold_auto`, `oversight_min_sample_rate`, and `drift_alert_threshold` to the unit interval $[0.0, 1.0]$. Both replace silent latent failures: an environment variable of `0` for the floor would zero out the SR-NF-06 oversight without warning, and an out-of-range threshold would produce nonsensical routing decisions without any error path indicating the cause.

Exception-to-HTTP mapping

The third recurring pattern is the convention by which cluster services raise typed exceptions (`ProfileNotFound`, `ProfileConflict`, `ProfileTransitionError`, `RollbackError`, `SubmissionBlockedError`, `InvalidTransitionError`) and route handlers translate them to the appropriate `HTTPException`. Business logic stays free of HTTP-specific concerns: services know nothing about status codes, and routes contain no business-logic conditionals. Where a response body needs detail, the exception carries it; `SubmissionBlockedError`, for example, takes the blocking-item list as a constructor argument so the route handler does not recompute it.

6.5 Frontend Implementation

The frontend is a React + Vite single-page application that exposes the two surfaces specified in Section 5.2.2: a Review surface used by Data Engineers to work through the session item queue, and a Governance surface used by Governance Leads to manage the profile lifecycle and acknowledge drift alerts. The frontend is the operator interface; the Chapter 7 evaluation tracks measure the supervision architecture at the REST contract layer, not at the UI.

The application shell is organised as one `BrowserRouter` with four routes (`App.tsx`). The Sessions page lists all sessions filterable by status and database; the Review page (`/review/:sessionId`) renders the session item queue alongside the item detail panel; the Governance page lists profiles and drift alerts; the Profile detail page renders a single profile and the governance actions available for it. A persistent navbar provides surface switching and exposes a role selector that toggles the active role between `DATA_ENGINEER` and `GOVERNANCE_LEAD`. The role selector is a prototype convenience: a production deployment would derive the role from an authenticated identity, as Appendix B.1 notes, but the surface-switching mechanism would be unchanged.

The active role is held in a `RoleContext` (`src/context/RoleContext.tsx`) that wraps the entire application. The API client in `src/services/api.ts` exposes two request helpers: `req` for routes the backend exposes to any caller, and `reqRole` for routes the backend's Role Guard protects. The latter reads the current role from the context and injects it as the `x-reviewer-role` header on each request. Concentrating header injection in one helper gives the frontend the same separation of concerns the backend Role Guard does: every component stays role-agnostic and the role contract sits in one module.

The components follow the same surface boundary. The Review surface is composed of the item queue (`ItemQueue`), the per-item detail panel (`ItemDetail`), and the per-candidate card (`CandidateCard`) that renders the five CopycHats cues of Section 5.5.6, alongside the session card and the persistent controls (Pause, Submit, and a progress indicator). These are built on shadcn/ui primitives, and two hooks, `useSession` and `useItemQueue`, wrap the API surface so the page components stay declarative.

Two omissions are deliberate and bear naming. The hint-capture control corresponding to the schema and endpoint layer of SR-F-13 is not implemented in the frontend, consistent with the framing in Section 5.7: the architecture-specified portion of SR-F-13 ends at the schema and the endpoint contract; the reviewer-side capture control is deferred. Accessibility against WCAG 2.1 AA (SR-NF-07) is similarly deferred and is acknowledged as a limitation in Chapter 7.

6.6 Mock Auto-Standardiser

A live Auto-Standardiser pipeline is not available in the NOUS prototype context. Chapter 5 makes this explicit in design decision D1 and assumption A1: the supervision system is built and evaluated against a mock service that exercises the same HTTP boundary the live AS would. The mock has two purposes. It supplies synthetic job payloads with realistic candidate metadata so the backend can be exercised end-to-end during local development; and it provides the controlled inputs that Chapter 7's evaluation scenarios depend on, where the confidence distribution and the relationship to existing profiles must be reproducible across runs.

6.6.1 Architecture

The mock is a separate FastAPI application in `mock_as/` (a top-level sibling of `backend/` under `implementation/`), deployable on a port distinct from the supervision backend. The two applications share no code and no database, exercising the contract boundary that D1 commits to. The mock holds two in-memory catalogues. The first, `ONTOLOGY`, lists realistic field entries for each of the three target standards (SAREF, CIM, IEC 61850), with each entry carrying the standard's canonical field name, a one-sentence definition, an ontological hierarchy position, and a list of sibling concepts, which is exactly the metadata the five-cue information panel in Section 5.5.6 expects on the candidate side. The second, `FIELD_CATALOGUE`, lists source fields realistic to the UC2 energy-data domain (power, voltage, and current channels, frequency, energy totals, and metering metadata) and pairs each with a data type and a sample of instance values for cues 2 and 3 on the source side.

The mock exposes two confidence modes for general job generation and three scenario endpoints for targeted evaluation. In `random` mode (the default), each candidate's confidence is drawn from a uniform distribution between 0.55 and 0.99. In `controlled` mode, the caller passes a `confidence_overrides` dictionary mapping source-field names to the desired top-candidate confidence, and the mock honours each override exactly. The two modes share the same item-construction code and differ only in how the top candidate's confidence is computed.

6.6.2 Scenario endpoints

Three scenario endpoints support the Chapter 7 Track A evaluation. `POST /mock/scenarios/baseline/{database_id}` generates a job whose confidence distribution places roughly 70% of items above the default `routing_threshold_auto` and 30% below, producing a known split

for evaluating DO-8 and DO-9. `POST /mock/scenarios/drift-job/{database_id}` reads the current APPROVED profile for the database from the supervision backend (first `GET /profiles?database_id=...&status=APPROVED` to locate the profile, then `GET /profiles/{id}` to fetch its entries) and generates a job whose candidates intentionally diverge from the approved mappings on the majority of fields, producing the FIELD_DRIFT alert tested by DO-13 and DO-14. `POST /mock/scenarios/standard-mismatch/{database_id}` reads the same APPROVED profile and emits a job whose `target_standard` differs from the profile's standard, exercising the STANDARD_MISMATCH branch of the drift detector. The drift-job scenario illustrates the round-trip pattern most directly:

```
@app.post("/mock/scenarios/drift-job/{database_id}")
def scenario_drift_job(database_id: str):
    list_resp = httpx.get(
        f"{BRSM_URL}/profiles",
        params={"database_id": database_id, "status": "APPROVED"},
    )
    if list_resp.status_code != 200 or not list_resp.json():
        raise HTTPException(
            422,
            f"No APPROVED profile for '{database_id}'.",
        )
    profile_id = list_resp.json()[0]["id"]
    profile = httpx.get(f"{BRSM_URL}/profiles/{profile_id}").json()
    approved_standard = profile["target_standard"]
    entry_by_field_id = {
        e["field_id"]: e for e in profile["entries"]
    }
    # ... build candidates that diverge on entries we recognise ...
```

The pattern keeps the mock free of any supervision-side schema knowledge: it only reads the API contract, treating the supervision backend's published endpoints the way a third-party AS would.

6.6.3 Limitations

Two limitations deserve explicit mention. The mock generates each source field's identifier as `f"{database_id}::{source_field}"`, which is stable across runs for the same database and source field but is a string template rather than a true schema-derived identifier from a source-database catalogue. The drift detector relies on this stability to match a profile entry against a new candidate for the same field, so Chapter 7 acknowledges the synthetic origin as a threat to validity. The mock's ontology is hand-curated, not drawn from the canonical SAREF, CIM, and IEC 61850 ontology graphs; it is sufficient to exercise the five-cue information panel and the standard-aware comparison the drift detector performs, but it does not stand in for the full ontological coverage a deployed AS would have. Both limitations are properties of the evaluation environment, not of the supervision architecture, and they bound the claims Chapter 7's Track A is permitted to make.

6.7 Testing Strategy

The test suite has two layers. Unit tests, under `backend/tests/unit/`, target pure functions and state machines in isolation; they run without a database fixture and without the FastAPI application context. Integration tests, under `backend/tests/integration/`, exercise the API surface end-to-end through FastAPI's `TestClient`, with each test starting from a clean SQLite database. The two layers are not redundant: the unit tests cover the boundary conditions of the algorithms (the threshold step, the sampling floor, the transition table, the guard predicate), and the integration tests cover the end-to-end realisation of each design objective through the routes that compose those algorithms.

Table 6.2 maps each test module to the design objectives it exercises.

Layer	Module	Design objectives covered
Unit	`test_routing_engine.py`	DO-8 (threshold classification), DO-9 (sampling floor with clamp)
Unit	`test_state_machine.py`	Session lifecycle invariants underlying DO-2
Unit	`test_submission_guard.py`	DO-6 (guard logic in isolation)
Integration	`test_session_integrity.py`	DO-1, DO-2, DO-3, DO-4, DO-5
Integration	`test_submission.py`	DO-6 (guard enforced at the API boundary; HTTP 422 with blocking-item payload)
Integration	`test_carry_forwards.py`	DO-7 (carry-forward chain and queue ordering)
Integration	`test_profile_governance.py`	DO-10, DO-11, DO-12
Integration	`test_drift_detection.py`	DO-13, DO-14

Table 6.2: Test module to design objective coverage.

The two layers cover DO-6 at different levels: the unit test confirms the guard logic on synthetic SessionItem mocks, and the integration test confirms that the same logic, invoked through the route, returns HTTP 422 with the blocking-item identifiers in the response body. The integration tests also confirm the persistence side-effects the design relies on, such as carry-forward visibility in the next session, profile entry composition from a SUBMITTED session, and drift alert creation at job ingestion. Each design objective is supported by at least one piece of test evidence, several at both layers.

The integration tests construct their inputs directly through the `TestClient` rather than going through the mock Auto-Standardiser. The choice keeps the tests deterministic: a fixture that constructs a job payload by hand can pin exact field identifiers, exact confidence values, and exact candidate metadata, whereas routing test inputs through the mock would introduce the mock's random-mode noise into the test environment. The mock is reserved for the seed script (`backend/seed.py`) and for the Chapter 7 evaluation runs, where reproducibility is achieved through the mock's controlled-confidence mode rather than through TestClient fixtures.

Three areas are intentionally not tested at this stage. The frontend has no automated test coverage; it was exercised manually against the running backend during development. Accessibility against WCAG 2.1 AA was not validated, consistent with SR-NF-07 being acknowledged as a limitation in Chapter 7. Performance was not measured: UC2's batch standardisation workflow is expected to involve modest volumes, but no load tests have been written to confirm how the system behaves under load. None of these omissions affects the design objectives the evaluation in Chapter 7 covers, but each is recorded honestly here so the testing claims the thesis makes remain within bounds.

6.8 Implementation Findings

Building an artefact from a specification is itself a test of the specification. Most of Chapter 5 survived construction without change: the cluster boundaries, the session lifecycle, the routing algorithm, and the governance state machine were realised as written. Three points did not, each a place where construction met something the design had left under-determined, mis-stated, or asserted in prose without a matching mechanism in code. The test suite alone does not expose them: a passing suite confirms the behaviour each test asserts, not that the build realises the whole specification, so a divergence outside that coverage can pass every test, including one that produces no observable difference at all. Holding the completed backend against the named specification of Chapter 5, section by section, is what brings such a divergence into view, and doing so favoured the design as the standard the build had to meet: two of the three points were closed by correcting the code to the specification, and the third by extending the specification itself, which had captured its own intent less completely than the code had. None of the three reached the evaluated artefact as a defect; each is recorded because resolving it sharpened either the design or its relationship to the build.

6.8.1 A named integration contract is not a free endpoint

The most consequential of the three concerned the `GET /hints/{database_id}` endpoint, where two different features answer to one signature. Algorithm steering (SR-F-13) needs an endpoint, keyed by database, that returns the ontological hints reviewers have captured, so the Auto-Standardiser can consume them on a subsequent job (Section 5.7). A profile-bootstrap mechanism (by which the AS would pull the entries of the current APPROVED profile as priors for the next job) would present identically: a GET keyed by database returning mapping guidance. The signature does not separate them, and read by signature alone `/hints` was first built as the profile-bootstrap pull, though only algorithm steering is the contract Chapter 5 names. What disambiguates is Section 5.7: read against it, `/hints` resolves to a single feature, returning the reviewer-captured hints held on session items, with a companion `POST /hints/{item_id}/applied` to mark a hint consumed. Realigning the endpoint to that reading cleared the other feature's footprint. The mock Auto-Standardiser's drift-job scenario, which had bootstrapped itself off `/hints`, now reads the approved profile through `GET /profiles` instead (Section 6.6); the

`ProfileEntry.applied` column the profile-bootstrap reading required became dead and was removed in migration `d4e5f6a7b8c9` (Section 6.3); and the hint-applied state now sits on the session item, where the algorithm-steering contract lives. What this case shows is that an integration contract named in the design is not interchangeable with a similarly shaped endpoint that happens to be convenient to build: a signature underdetermines the feature behind it, and the requirement the design attaches to the name is what has to be carried into the code.

6.8.2 Cross-cutting enforcement is under-described by default

The submission guard blocks the transition from ACTIVE to SUBMITTED when any item that must be decided has not been. Section 5.5.5 of Chapter 5 described this guard as checking the SAMPLED items only. Building it brought in a second condition that the design had placed elsewhere: a Tier 0 carry-forward item left undecided must also block submission, the chain-integrity rule of Section 5.5.6. The built guard was therefore more complete than the section that purported to define it, and the gap lay in the design's own internal consistency, one section describing half of an enforcement rule that a second section completed. Here the design gave way rather than the code: Section 5.5.5 and design objective DO-7 were brought into agreement with the chain-integrity rule that Section 5.5.6 already stated, consolidating an obligation the design had already committed to rather than importing a new criterion from the build. The carry-forward arm of DO-7 thus records a requirement that predated construction; what building supplied was the recognition that the two design sections had to be read together for the guard to be specified completely. The point here is specific to enforcement points that several requirements touch at once. A guard invoked on behalf of more than one requirement tends to be described partially by each of them, and no single description is guaranteed complete unless the requirements that share the guard are deliberately cross-checked against one another.

6.8.3 A hard constraint in prose is not a constraint until it is code

Section 5.5.5 of Chapter 5 calls `oversight_min_sample_count` of at least one "a hard constraint validated at system startup", the mechanism by which the never-fully-autonomous floor of SR-NF-06 is made non-negotiable. The phrasing states the property as a fact rather than prescribing the check that would make it one, and the configuration layer was built without that check. A deployment that then set the count to zero through an environment variable would have started without complaint and silently disabled the oversight floor, defeating the requirement the constraint exists to protect. Discharging the claim meant adding the Pydantic `field_validator` of Section 6.4.4, which rejects any value below one and stops the application from starting under that misconfiguration. The point is not specific to this one setting. A constraint asserted in prose is a claim about the system, not a property of it; until that claim is discharged in code that fails loudly when the constraint is violated, the requirement it protects is unguarded, and a "hard constraint validated at startup" that is validated nowhere is indistinguishable from a comment.

6.8.4 What the findings indicate about the design

The three findings share a structure characteristic of the design-science demonstration stage: none arose from a flaw in the architecture's concepts, but each arose at the seam between the specification and its realisation. Because each was traced and closed, Chapter 5 as presented is the version that has been held against a working build rather than an untested plan, and the artefact Chapter 7 assesses was checked against the design specification section by section. That check was carried out by the researcher who built the system rather than independently audited, a reliability limitation Chapter 7 records among its threats to validity.

6.9 Summary

The fourteen design objectives of Section 5.3 are the falsifiable conditions the implemented components must satisfy. Table 6.3 records, for each, that it is realised in the build and where Chapter 7 assesses it.

Every objective is built, and the clean match is by construction, not coincidence: Section 5.3 defined the set as the falsifiable conditions for exactly the components selected for implementation. The same boundary shows in the API contract. Of the twenty-one endpoints in Appendix B.1, eighteen serve the implemented clusters and the objectives above; the other three (`GET /corrections`, `GET /hints/{database_id}`, `POST /hints/{id}/applied`) are reachable stubs for the architecture-specified components the thesis does not build end to end, correction-driven retraining (SR-F-05) and algorithm steering (SR-F-13), whose consumers depend on the live Auto-Standardiser and lie out of scope (Section 5.7). Those three carry no design objective and fall outside the evaluation, together with tiered escalation (SR-F-12), which has no endpoint at all.

Chapter 7 evaluates the built artefact against the fourteen objectives on two tracks. Track A drives the components against the mock Auto-Standardiser of Section 6.6 with controlled-confidence synthetic jobs, checking each objective against a known ground truth; Track B has practitioners operate the prototype to test whether the session, routing, and governance behaviour meets the supervision needs identified in Chapter 4.

ID	Built	Track A test evidence	Condition in brief
DO-1	Yes	`test_session_integrity`	Session holds one job's flagged and sampled items only
DO-2	Yes	`test_state_machine`; `test_session_integrity`	Decisions survive a pause/resume cycle
DO-3	Yes	`test_session_integrity`	Reviewed and unreviewed counts exposed throughout
DO-4	Yes	`test_session_integrity`	Items ordered by tier ascending, then confidence ascending
DO-5	Yes	`test_session_integrity`	Every session carries a minimum high-confidence sample whenever auto-accepted items exist
DO-6	Yes	`test_submission_guard`; `test_submission`	Submission rejected while a mandatory sample is undecided
DO-7	Yes	`test_carry_forwards`	PROVISIONAL items carry forward and block the next submission until decided
DO-8	Yes	`test_routing_engine`	Items at or above the threshold excluded unless sampled
DO-9	Yes	`test_routing_engine`	Sampling floor is the clamped maximum of a rate and a count
DO-10	Yes	`test_profile_governance`	Profile includes APPROVED, CORRECTED, AUTO_ACCEPTED; excludes PROVISIONAL, SKIPPED
DO-11	Yes	`test_profile_governance`	Approval rejected for DATA_ENGINEER (HTTP 403); only GOVERNANCE_LEAD approves
DO-12	Yes	`test_profile_governance`	Rollback swaps statuses without creating new rows
DO-13	Yes	`test_drift_detection`	DriftAlert raised when the diverging fraction exceeds the threshold
DO-14	Yes	`test_drift_detection`	`drift_alert_id` non-null iff an alert was created

Table 6.3: Design objective build status and Chapter 7 evaluation locus. "Built" is a build fact; the test column names where Chapter 7's technical track assesses each objective, not a result.

Chapter 7

Evaluation

This chapter evaluates the NOUS AS Supervision Architecture. Section 7.1 sets out the evaluation approach, which operates at two altitudes, and Section 7.2 maps every supervision requirement to the evidence it receives. Section 7.3 demonstrates the full architecture against the canonical UC2 flow, the two empirical tracks then evaluate the implemented subset, Section 7.6 draws the consolidated verdict and residual risks, and Section 7.7 examines the threats to validity.

7.1 Evaluation Approach

This thesis produces an instantiation (Section 5.2.4), but its design contribution is wider than the running system. The architecture in Chapter 5 specifies supervision components for all twenty requirements, while the prototype implements the subset realisable without the live Auto-Standardiser ML pipeline. Evaluating only the running system would leave the specified-but-unbuilt part of that contribution unexamined. The evaluation therefore operates at two altitudes, following the distinction design science research draws between evaluating a design and evaluating a realised artifact. Sonnenberg and vom Brocke separate evaluation of a design before instantiation (their EVAL2) from evaluation of the instantiation in an artificial setting and then in use (EVAL3 and EVAL4) [44]; the framework for evaluation in design science research places these on a spectrum from artificial to naturalistic [46]. A related distinction appears in Peffers et al.'s separation of demonstration from evaluation [32] and in Hevner et al.'s treatment of an artifact as evaluable at the level of a model or method, not only an instantiation [16].

Ex-post implementation validation. The implemented components are evaluated empirically, which is what Research Question 3 asks: the extent to which the implemented supervision architecture addresses the requirements identified in Chapter 4. Two tracks supply the evidence. Track A is a technical evaluation against the mock Auto-Standardiser API, in which each of the fourteen design objectives in Table 5.2 is exercised against synthetic job outputs with controlled confidence distributions, so that each objective is met or not met under a known ground truth. Track B is a practitioner validation in which practitioners operate the deployed prototype through representative

scenarios and judge whether the implemented behaviour matches the supervision needs identified during requirements analysis. The full Track B protocol is reproduced in Appendix C.

Ex-ante design validation. The components specified but not implemented (Section 5.7) cannot be evaluated empirically, because there is no running behaviour to observe. They are evaluated as a design on four grounds. The first is an architectural demonstration: Section 7.3 traces the full architecture, implemented and specified components together, through the canonical UC2 flow, showing that the specified elements occupy defined integration points in an end-to-end workflow rather than standing alone. The second is requirements coverage: each of the twenty requirements maps to a specified design element, traced in Section 7.2. The third is internal consistency: every design decision is grounded in a stated requirement and in peer-reviewed literature or regulation, recorded in Table 5.5. The fourth is an expert design-credibility check: for tiered escalation (SR-F-12) and algorithm steering (SR-F-13), Topic 4 of the Track B protocol asks whether the specified design captures what the practitioners described, since both requirements originated in the requirements interviews and the specification synthesises that input without an independent check. In the event a single practitioner's session reached this block (Section 7.5.4), so this fourth ground rests on one reaction.

Ex-ante evidence is weaker than the ex-post evidence available for the implemented components, and its strength is not uniform across the specified-only set. Standard-partitioned retraining (SR-F-05), the audit trail and compliance requirements (SR-F-10, SR-NF-04), and post-submission accuracy verification (SR-NF-02) receive no practitioner reaction directed at their core behaviour. SR-F-05 is checked only for whether the correction schema carries the fields a retraining consumer would read, though a reviewer's question about how a correction propagates to earlier mappings (Section 7.5.3) touches it incidentally; SR-F-10 and SR-NF-04 rest on the EU AI Act [11] and GDPR rather than on any architectural novelty of this work; SR-NF-02 is a statistical-reliability property that practitioner reaction would not meaningfully test. For these four the available evidence is analytical or regulatory, and Section 7.2 records that limit rather than presenting them as validated to the same degree as the built components.

Research Question 3 is answered by the ex-post tracks alone, against the implemented subset. The ex-ante strand evaluates the wider architecture specification and is reported as a separate result, not as part of the answer to Research Question 3. This keeps the empirical claim restricted to what the prototype can demonstrate while still subjecting the full specification to the strongest scrutiny an unbuilt design admits.

7.2 Evaluation Scope and Requirement Traceability

Table 7.1 maps each of the twenty supervision requirements to the design element that realises it, its implementation tier (Section 4.6), the evaluation altitude that applies, and the method and pass criterion. Design elements marked arch-spec are specified only, in Section 5.7; the design objectives (DO-n) are defined in Table 5.2. The table is the single point at which the coverage claim for the full architecture can be read against the evidence each requirement actually receives.

Table 7.1: Requirement traceability: each supervision requirement mapped to its realising design element, implementation tier, evaluation altitude, and method.

SR	Realised by	Tier	Altitude	Method and pass criterion
SR-F-01	Confidence Routing (DO-8)	1	Ex post (RQ3)	Track A DO-8; Track B Vignette A
SR-F-02	Session Manager (DO-1/2/3)	1	Ex post (RQ3)	Track A DO-1/2/3; Track B Vignette A
SR-F-03	Display layer built; ontology lookup arch-spec	2	Ex post (display); ex ante (lookup)	Track B qualitative; lookup by coverage only
SR-F-04	CORRECTED type, `corrected_to` schema	2	Ex post	Track B sweep; correction-tuple completeness
SR-F-05	Correction export contract, arch-spec	3	Analytical	Track B schema-presence check; no live retraining pipeline
SR-F-06	Profile Governance (DO-10)	1	Ex post (RQ3)	Track A DO-10; Track B Vignette C
SR-F-07	Profile Governance (DO-12)	1	Ex post (RQ3)	Track A DO-12; Track B Vignette C
SR-F-08	Drift Detector (DO-13, DO-14)	1	Ex post (RQ3)	Track A DO-13/14; Track B Vignette C; mock `field_id` threat (A1)
SR-F-09	Role Guard (DO-11)	1	Ex post (RQ3)	Track A DO-11; Track B Vignette A
SR-F-10	Append-only audit log, arch-spec	3	Regulatory	EU AI Act Art. 9/12, GDPR; not practitioner-validated
SR-F-12	Three-tier routing, arch-spec	3	Ex ante (design)	Track B Topic 4, single-practitioner credibility check
SR-F-13	Hint contract and capture schema, arch-spec	3	Ex ante (design)	Track B Topic 4, single-practitioner check (spec revised in response)
SR-F-14	PROVISIONAL carry-forward (DO-7)	1	Ex post (RQ3)	Track A DO-7; Track B Vignette A

(Continued on next page)

(Continued from previous page)

SR	Realised by	Tier	Altitude	Method and pass criterion
SR-NF-02	Sample-accuracy metric, arch-spec	3	Analytical	Computable from SAMPLED decisions; unreliable at $n = 1$; not practitioner-validated
SR-NF-03	Item queue ordering (DO-3, DO-4)	1	Ex post (RQ3)	Track A DO-3/4; Track B sweep
SR-NF-04	Retention and access log, arch-spec	3	Regulatory	GDPR Art. 32, EU AI Act; not practitioner-validated
SR-NF-05	Role Guard separation (DO-11)	1	Ex post (RQ3)	Track A DO-11; Track B sweep
SR-NF-06	Oversight floor, submission guard (DO-5, DO-6, DO-9)	1	Ex post (RQ3)	Track A DO-5/6/9; Track B sweep
SR-NF-01	Not in scope (upstream property)	–	Out of scope	Upstream of supervision boundary (Section 5.2.1)
SR-NF-07	Interface built, not WCAG-validated	–	Limitation	Acknowledged limitation; not designed to WCAG 2.1 AA

Read by altitude, the table resolves into four bands. Twelve requirements are validated ex post against the prototype and form the evidence for Research Question 3: the routing, session, and governance behaviours, each carrying one or more design objectives testable on Track A and exercised again by practitioners on Track B. Two requirements, tiered escalation (SR-F-12) and algorithm steering (SR-F-13), received an ex-ante credibility check from the single practitioner whose session reached the Tier 3 block (Section 7.5.4); the reviewer endorsed both concepts, and for SR-F-13 the same check surfaced a correction to the hint data model that was adopted into the specification (Section 5.7) and not independently re-validated. Their realisation depends on reviewer-history data and AS-side execution absent from the prototype. Four requirements (SR-F-05, SR-F-10, SR-NF-04, SR-NF-02) carry only analytical or regulatory evidence, the weakest band, and are reported as such. The remaining two fall outside the evaluated set: column coverage (SR-NF-01) sits upstream of the supervision boundary (Section 5.2.1), and accessibility (SR-NF-07) is acknowledged as a limitation rather than a designed and validated property.

7.3 Architectural Demonstration via Use-Case Mapping

The traceability map in Section 7.2 asserts that every supervision requirement is realised by a design element, but a map is a claim about coverage, not a demonstration that the elements connect into a working supervision workflow. This section supplies the demonstration. It traces the canonical UC2 standardisation flow through the architecture step by step, mapping each step to the component that handles it and the requirement that component satisfies, and confirming that the flow reaches the end without a step that no component owns. This is the demonstration activity that design science research places between design and naturalistic evaluation [16, 32]: it exercises the whole architecture, including the components specified but not built, against one concrete scenario.

The demonstration covers the full architecture rather than the implemented subset, which is why it belongs to the ex-ante design-validation altitude (Section 7.1) and not to Research Question 3. The implemented components are reachable as running behaviour in the prototype; the Tier 3 components appear at their specified integration points, marked as such, so that the end-to-end flow can be read in full. Where a step depends on a specified-only component, the demonstration shows the integration point the implemented system already exposes, not running behaviour.

7.3.1 Scope

The scenario is a single standardisation job in UC2: a Data Engineer at the energy utility submits a source database of energy measurements to the Auto-Standardiser for translation to a target standard (SAREF), and the supervision architecture governs the human oversight of that job from ingestion to an approved mapping profile, with a later rollback after a downstream failure. The mapping aims to establish three properties of the architecture:

1. every step of the flow has a defined entry point into a supervision component, so no part of the job is unsupervised by omission;
2. every supervision-relevant decision or artefact the scenario produces is created or consumed by a named architecture component; and
3. each of the twenty supervision requirements is accounted for as the scenario unfolds: eighteen are exercised in the flow, twelve through running components and six at their specified integration points. The remaining two, column coverage (SR-NF-01) and accessibility (SR-NF-07), are boundary cases that are not steps in the supervision flow.

The scope is one standardisation-job lifecycle for one source database, together with the governance actions it triggers and the principal exception flows. Cross-database concerns, the AS translation algorithm itself, and the upstream pre-ingestion quality gate (SR-NF-01, Section 5.2.1) are outside this analysis.

7.3.2 Baseline Scenario

Without the supervision architecture, the AS handles the job as an unsupervised translation. The baseline flow has three steps: (B1) the Data Engineer submits the source database to the AS for translation to SAREF; (B2) the AS produces, for each source field, a ranked list of candidate target mappings, each carrying a confidence score; (B3) the AS writes the translated database to the target schema. In this baseline every mapping is accepted as produced. Low-confidence mappings are committed silently, no record distinguishes a reviewed mapping from an unreviewed one, there is no versioned statement of which mappings were approved for the database, and a later divergence from previous practice passes unnoticed. The supervision architecture inserts the human-oversight controls these gaps require around this baseline, without changing the AS translation step itself.

7.3.3 Step-to-Requirement Mapping

Table 7.2 traces the supervised flow. Each row is a step in the job lifecycle, the architecture component that owns it, the supervision requirements it satisfies, and whether that component is implemented or specified only. The flow order matches the implemented ingestion path, in which carry-forward injection, routing, and drift detection all complete before the session is created. Table 7.2 reads the architecture in flow order; the traceability map in Table 7.1 reads the same elements in requirement order, and the two are consistent.

Table 7.2: The canonical UC2 standardisation flow traced through the architecture: each step mapped to its owning component, the requirements it satisfies, and its implementation status.

Step	What happens	Component	Requirements	Status
S1	AS posts the completed job (per-field candidates, confidences, target standard, model version) to the supervision system	Job ingestion entry point (`POST/jobs`)	Entry point	Implemented
S2	Provisional items deferred from a prior session for the same database are pulled forward as Tier 0	Session Factory (carry-forward injection)	SR-F-14	Implemented
S3	Items are partitioned into auto-accepted and flagged by confidence threshold	Confidence Routing Layer	SR-F-01	Implemented

(Continued on next page)

(Continued from previous page)

Step	What happens	Component	Requirements	Status
S3′	Flagged items would route to a junior pool, escalating to an expert on disagreement	Tiered escalation routing	SR-F-12	Tier 3 (spec)
S4	A mandatory sample is drawn from the auto-accepted pool so no batch is fully unreviewed	Sampling Selector (oversight floor)	SR-NF-06	Implemented
S5	The job is compared against the current approved profile; a standard mismatch or field-drift alert is raised	Drift Detector	SR-F-08	Implemented
S6	A bounded, pausable review session is created over the Tier 0/1/2 items for the job	Batch Review Session Manager	SR-F-02	Implemented
S7	The reviewer works the confidence-sorted queue, with progress counts and pause/resume, recording approve/correct/skip/provisional decisions	Item Queue, Reviewer Interface	SR-F-02, SR-NF-03, SR-F-04	Implemented
S7′	Each candidate is shown with its hierarchy path and sibling concepts in the target standard	Standard-aware ontology lookup	SR-F-03	Display built; lookup spec
S7″	The reviewer supplies an ontological hint for an unmapped column	Hint capture and contract	SR-F-13	Tier 3 (spec)
S8	Submission is blocked until every sampled and carried-forward item carries a real decision	Submission Guard	SR-NF-06	Implemented

(Continued on next page)

(Continued from previous page)

Step	What happens	Component	Requirements	Status
S9	A draft mapping profile is derived from the submitted session's approved, corrected, and auto-accepted entries	Profile Builder	SR-F-06	Implemented
S10	The engineer proposes the profile; a separate governance lead approves it; the prior version is superseded	Role Guard, Lifecycle Transitions	SR-F-09, SR-NF-05	Implemented
S10′	Every state change is written to an append-only, retention-bound audit log	Audit trail service	SR-F-10, SR-NF-04	Tier 3 (spec)
S11	Captured corrections are exposed for the AS retraining pipeline to pull, partitioned by standard	Correction export contract (`GET/corrections`)	SR-F-05	Contract built; pipeline spec
S11′	Sample accuracy is computed from the reviewed sample and recorded on the session	Post-submission accuracy verification	SR-NF-02	Spec (analytical)
S12	After a downstream failure, the governance lead rolls back to the prior approved profile version	Rollback Handler	SR-F-07	Implemented

The flow reaches an approved, versioned mapping profile and a recoverable rollback point without a step that no component owns. Of the twenty requirements, the demonstration exercises eighteen along the main flow; the remaining two are the out-of-scope and limitation cases recorded in Section 7.2, column coverage (SR-NF-01, upstream of the boundary) and accessibility (SR-NF-07, a limitation), neither of which is a step in the supervision flow. Twelve of the eighteen are exercised by components reachable in the running prototype; the other six appear at integration points the implemented system already exposes: the correction and hint endpoints, the candidate JSON structure the display layer already renders, and the persistence layer an audit log would shadow.

7.3.4 End-to-End Sequence

The implemented spans of the flow are the subject of the sequence diagrams in Chapters 5 and 6, reused here as the dynamic view of the demonstration. Steps S1 to S6 are the job ingestion and routing sequence of Figure 5.4: the routing engine partitions the items and the sampling floor pulls back its mandatory sample, drift detection runs against the approved profile, and only then is the session created. Step S8, the submission guard, is the subject of Figure 5.6, which shows the guard rejecting a submission that leaves a sampled item undecided. Steps S9 and S10 are the profile creation and governance approval sequence of Figure 5.10, including the Role Guard rejecting an approval attempt by a Data Engineer. The drift path of step S5 and the rollback of step S12 are shown in Figure 6.2. The component internals each step touches are the three Level 3 views, Figures 5.3, 5.5, and 5.9. The specified-only steps (S3′, S7″, S10′, S11) have no running sequence; their place in the flow is the position shown in Table 7.2, and their integration contracts are specified in Section 5.7.

7.3.5 Alternative and Exception Flows

The main flow is the path on which the job proceeds without incident. The architecture is tested more by what it does when the scenario deviates. Six deviations are traced below.

Low-confidence burst. A job in which most fields fall below the auto-accept threshold produces a large flagged set. The routing engine still applies the oversight floor to the small auto-accepted pool, and the session manager's confidence-sorted queue, per-tier progress counts, and pause/resume keep the enlarged session workable (SR-NF-03, SR-NF-06). The burst changes the session's size, not the controls that govern it.

Drift on a known database. When the job targets a database that already has an approved profile and the new candidates diverge from it, the Drift Detector raises a field-drift or standard-mismatch alert at ingestion, before the session exists, and the governance surface presents it for acknowledgement (SR-F-08). Divergence is surfaced at the job boundary rather than waiting for a reviewer to notice it.

Unauthorised governance action. A Data Engineer who attempts the approval step is rejected by the Role Guard with HTTP 403; only a Governance Lead may approve (SR-F-09, SR-NF-05). The separation of duties is enforced at the API boundary, not by convention.

Downstream failure after promotion. When an approved profile causes a downstream failure, the Governance Lead rolls back to the prior approved version. The rollback changes status only and creates no new rows, so the full version sequence remains queryable for audit reconstruction (SR-F-07).

Conflicting reviewers on a flagged item. In the full architecture, an item on which reviewers disagree escalates from the junior pool to an expert tier (SR-F-12). This path is specified

only; the implemented single-reviewer session is its substitution point, and the routing engine is the component the escalation tier extends.

Unmapped column. A column the AS cannot map prompts the reviewer to supply an ontological hint for the next job on that database (SR-F-13). The hint schema and the hint endpoints are in place; the capture control and the AS-side consumption are specified only.

None of the six deviations bypasses the routing gate, the submission guard, or the role-gated approval chain; each is surfaced through the same session-and-governance workflow as the main flow, and the two specified-only deviations have defined substitution points in the implemented components. This is the property the architecture is meant to hold: the exceptions are handled by the same controls as the ordinary case.

7.3.6 Demonstration Verdict

The walkthrough demonstrates that the architecture forms a connected end-to-end supervision workflow over the canonical UC2 job: every step of the main flow and every principal exception maps to a named component, the flow terminates in a governed, versioned, recoverable mapping profile, and all twenty requirements are accounted for, eighteen as supervision steps along the flow and two as non-flow requirements (SR-NF-01, SR-NF-07) at the boundary. This is the evidence the coverage map cannot give: that the design elements interlock rather than merely each existing. It is analytical evidence about the design, not empirical evidence about behaviour. It does not show that the implemented components behave correctly under controlled inputs, which is Track A, nor that the implemented behaviour matches what practitioners need, which is Track B. The consolidated verdict across all three is drawn in Section 7.6.

7.4 Track A: Technical Evaluation against the Mock Auto-Standardiser

Track A is the ex-post technical evaluation of the implemented components against the mock Auto-Standardiser (Section 6.6). Each of the fourteen design objectives in Table 5.2 is exercised against synthetic job outputs whose confidence distribution is fixed in advance, so that the objective is either met or not met under a known ground truth.

7.4.1 Method

The evaluation runs the supervision backend and the mock Auto-Standardiser as two separate services communicating over HTTP, against a freshly migrated database. This is the integration position the deployed system occupies, with the supervision system receiving completed jobs from the Auto-Standardiser rather than constructing them internally. A scripted harness generates each scenario through the mock, posts the resulting job to the backend, drives the review and governance workflow through the public API, and checks each objective's condition against the values the mock

actually emitted. Two mock facilities supply the controlled inputs: a controlled-confidence mode, in which the confidence of each source field is pinned to a chosen value, and three scenario endpoints that produce a fixed high-versus-low confidence split and the two drift conditions (Section 6.6). The routing parameters in force during the run were the system defaults: an auto-accept threshold of 0.85, a proportional sampling rate of 0.05, and an absolute sampling floor of one item.

Ground truth is read from the job the mock emits rather than assumed: the harness reads back the confidence assigned to each field and the identifiers of the items created, and every assertion is made against those observed values. A design objective is recorded as satisfied when the system's behaviour matches its specified condition with no manual intervention. Because each check is evaluated against the concrete job produced for that run, the outcomes do not depend on the mock's random draws; the per-objective checks, the scenario inputs, and the recorded results are kept in the evaluation artifact (Section 7.7.4).

7.4.2 Results

All fourteen design objectives were satisfied. Table 7.3 records, for each objective, the scenario that exercised it, the observation that decided it, and the outcome.

Table 7.3: Track A results: each design objective exercised against the mock Auto-Standardiser, with the deciding observation under known ground truth.

DO	Requirement	Scenario	Observation	Result
DO-1	SR-F-02	Two jobs, all flagged	Each session holds exactly its own items; the two sets are disjoint	Pass
DO-2	SR-F-02	Pause then resume	Both decisions present after the PAUSED then ACTIVE cycle	Pass
DO-3	SR-F-02, SR-NF-03	Incremental decisions	`n_reviewed` starts at zero and advances by one per decision	Pass
DO-4	SR-NF-03	Carry-forward, flagged, sampled	Queue ordered by tier then ascending confidence; tiers 0, 1, 2 all present	Pass
DO-5	SR-NF-06	All high-confidence	Eleven items auto-accepted, one drawn as a mandatory sample	Pass
DO-6	SR-NF-06	Undecided sample	Submit rejected with HTTP 422 (one blocking item), then accepted once decided	Pass
DO-7	SR-F-14	Provisional to next session	Returns as a tier-0 carry-forward; blocks submission until decided	Pass

(Continued on next page)

(Continued from previous page)

DO	Requirement	Scenario	Observation	Result
DO-8	SR-F-01	Baseline split	Fourteen above and six below threshold: six flagged, thirteen auto-accepted, one sampled; no above-threshold item in the queue except the sample	Pass
DO-9	SR-NF-06	Baseline split	Sample count $= \max(1, \lceil 14 \times 0.05 \rceil) = 1$	Pass
DO-10	SR-F-06	Mixed decisions	Eighteen of twenty entries built; provisional and skipped excluded; correction carried into the entry	Pass
DO-11	SR-F-09, SR-NF-05	Approval by each role	Data Engineer approval rejected (HTTP 403); Governance Lead accepted	Pass
DO-12	SR-F-07	Rollback to prior version	Target version set to APPROVED, current to SUPERSEDED; no new rows	Pass
DO-13	SR-F-08	Drift and mismatch jobs	FIELD_DRIFT and STANDARD_MISMATCH alerts raised at ingestion	Pass
DO-14	SR-F-08	Alert and no-alert jobs	`drift_alert_id` non-null if and only if an alert was created	Pass

The routing and session objectives behaved as specified across the confidence range. On the baseline split of twenty fields, fourteen fell at or above the auto-accept threshold and six below it; the six below-threshold fields were all routed to review, and of the fourteen above-threshold fields, thirteen were auto-accepted and one was drawn by the sampling floor, which is exactly $\max(1, \lceil 14 \times 0.05 \rceil) = 1$ (DO-8, DO-9). No above-threshold field reached the review queue except through the mandatory sample, confirming that auto-acceptance is the default and the sample the only exception to it (DO-8). The session held only its own job's items (DO-1), preserved decisions across a pause and resume cycle (DO-2), reported reviewed counts that advanced by one per decision (DO-3), and ordered the queue by tier and then by ascending confidence with all three review tiers present (DO-4). The two non-negotiable controls held at the API boundary: a session carrying auto-accepted items always received at least one mandatory sample (DO-5), and submission was rejected while a sampled item or a carry-forward remained undecided, then accepted once every such item carried a decision (DO-6). A provisional decision resurfaced as a tier-0 carry-forward in the next session for the same database and blocked that session's submission until it was resolved (DO-7).

The governance objectives held in the same way. A profile built from a mixed-decision session contained exactly the approved, corrected, and auto-accepted entries and excluded the provisional

and skipped ones, with the correction carried into the entry's approved field name; for the evaluated session this was eighteen of the twenty fields (DO-10). Approval was rejected for the Data Engineer role and accepted for the Governance Lead (DO-11), and a rollback returned the prior version to approved and the current version to superseded without adding a row to the profile table (DO-12). At ingestion, a job whose candidates diverged from the approved profile and a job whose target standard differed from it each raised the corresponding alert (DO-13), and the job response carried a drift-alert identifier when and only when an alert was created, including the null case of a job for a database with no approved profile (DO-14).

7.4.3 Interpretation

Track A establishes that the implemented components satisfy all fourteen design objectives under controlled inputs with a known ground truth, subject to the construction-time reconciliation of DO-7 noted in Section 7.7.2. The result is bounded by what the mock can represent. The mock reproduces the Auto-Standardiser's interface contract, not the statistical behaviour of a trained model, so a passing objective shows that the supervision components behave correctly against the contract and against the confidence distributions the evaluation imposed, not that they would behave correctly against the distributions a deployed Auto-Standardiser produces. This is the construct-validity limit examined in Section 7.7.1 and carried as residual risk R2 in Section 7.6.4. Within that boundary every objective testable under a mock is met, which is the technical evidence Research Question 3 requires for the implemented subset; the practitioner half of that question is taken up in Track B.

7.5 Track B: Practitioner Validation

This section reports Track B, the practitioner validation. Two practitioners operated the deployed prototype through the scenario vignettes and requirement sweep of the protocol in Appendix C. One had contributed to the requirements interviews of Chapter 4 and is referred to here as the returning participant; the other is an independent reviewer with no involvement in the design of the architecture or the construction of the prototype. The independent reviewer is the control the protocol prioritises against designer-confirmation and participant-investment bias (Appendix C), and provides the only practitioner reaction to the Tier 3 specifications, since the returning participant's session did not reach that block. Both sessions were transcribed and coded against the requirement, outcome, and locus schemes of the post-interview procedure (Appendix C.7), and are reported here with the disconfirming and divergent findings ahead of the endorsements. Topic 4 of the protocol supplies the ex-ante design-credibility check for tiered escalation (SR-F-12) and algorithm steering (SR-F-13). Usability observations are reported at the end of the section and, accessibility aside (SR-NF-07), sit outside the requirement-satisfaction claim, as the protocol specifies.

7.5.1 Divergence Finding: Deliberate Non-Mapping as a Distinct Decision

When operating the flagged queue, the returning participant reached for the Skip control expecting it to record that a column should not be mapped at all, and was uncertain whether that was its effect, remarking that "if skipping means don't do it at all, I think it should be." The prototype defines a skipped decision as a within-session deferral (Section 5.5.3), and the profile builder excludes both skipped and provisional items from the generated mapping profile (Section 5.6). A skipped field is consequently absent from the resulting profile and left unmapped in the pipeline, yet the decision record carries no positive statement that the reviewer judged the column to have no valid target. Skip and deliberate exclusion reach the same downstream outcome through different recorded intent, and the prototype offers no decision that expresses the second.

This recovers a distinction the same participant had drawn during requirements elicitation, where he separated "high confidence that a column should not be mapped" from "low confidence that a mapping exists" and noted that the two are not the same state and call for different reviewer responses (Chapter 4). The requirements catalogue did not carry the distinction into a supervision requirement. The nearest existing requirements address adjacent but different states: SR-F-14 covers the deferred, not-yet-decided item, which is the opposite epistemic position, and SR-F-13 covers steering the Auto-Standardiser toward a mapping rather than recording a decision against one. The finding therefore identifies a capability that behaves exactly as specified yet not as the practitioner intends. This is the divergence class the verdict (Section 7.6.1) treats as the most informative, and it points to a missing decision type rather than a defect in an existing control.

The conflation also has an audit dimension. Because a deliberately excluded column and a column merely not yet reviewed are indistinguishable in the resulting profile, the provenance the audit trail requirement (SR-F-10) calls for is weakened: the profile cannot record why a column is unmapped. The implication is a first-class non-mapping decision rather than a relabelling of the existing control, taken up as future work in Chapter 8.

The independent reviewer, who took no part in the requirements elicitation, reached the same uncertainty when operating the Skip control. He read it as ambiguous between three intents: keeping the source data while recording no mapping, excluding the column from the output altogether, and keeping the column while deferring the mapping to a later session. He assigned the third reading to the provisional decision and was left without a control for the first, observing that the system needed to distinguish a column with no valid target from one whose mapping was merely unresolved. That an independent reviewer arrived at the same distinction the returning participant had raised during elicitation, without exposure to it, establishes that the ambiguity is not idiosyncratic to one participant; the case for a first-class non-mapping decision remains a design proposal carried into future work rather than an established requirement, but it now rests on two independent encounters rather than one.

7.5.2 Authentication and the Trust Boundary

The independent reviewer raised authentication as a question the prototype does not answer and argued that it falls within the supervision layer's responsibility rather than outside it. The prototype selects the acting role from a request header, a stand-in for authentication at the prototype stage. The reviewer observed that a header the client sets is a credential the client can forge: a request issued directly to the API with an arbitrary role header is accepted as that role, so the separation of duties the Role Guard enforces (SR-F-09, SR-NF-05) can be bypassed by any caller that reaches the API directly.

He pressed the scope question rather than treating it as out of band. Because the supervision layer is operated from within the Auto-Standardiser's environment rather than as a destination a user logs into separately, he characterised the two systems as integrated rather than merely interoperable, and concluded that the supervision layer must participate in the surrounding authentication rather than delegate the question away. On that reading the role header is an open concern, not a deferred one.

The prototype's standing on this is that it is a demonstrator of the supervision functionality against a mock Auto-Standardiser, not a production deployment; authentication is one of the production properties the design deliberately places outside the prototype scope (Section 5.8), a scope decision the architecture already records rather than one introduced here in response to the reviewer (Section 7.7.3). The reviewer's own recommendation matches the intended production design: the role header is replaced by an OAuth2 bearer token in the authorization header, issued by an external federated identity provider autonomous from the platform, with the supervision layer validating the token and deriving the role from it rather than trusting a client-supplied value. This is recorded as residual risk R8 (Section 7.6.4) and taken up as future work in Chapter 8.

7.5.3 Capability Gaps Surfaced by the Validation

Beyond the non-mapping divergence, the validation surfaced five capabilities the participants reached for and did not find, each pointing to a defined extension rather than a defect in an existing control.

The most prominent, raised without prompting, is a whole-dataset structural view. The independent reviewer noted that the interface presents per-item evidence but no overview of the source database's structure before standardisation or the target structure after it, and that a reviewer entering a session partway through the work has no way to orient against the dataset as a whole. He had named this need before the demo, when asked what such a system should provide, and returned to it at the close of both vignettes. The session model (SR-F-02) bounds review by job and the information panel (SR-F-03) presents per-candidate ontology context, but neither presents the dataset as a structure, which is the unit the reviewer wanted to reason over.

Two further findings concern decisions the workflow does not record. A governance lead can roll a profile back but cannot reject a proposed profile with a stated reason, and the reviewer argued that a rejection should be both explainable at the time and consultable on the next run, so that a

later session can take prior rejections into account when the same mappings reappear. This is the reject-side counterpart to the correction capture of SR-F-04 and was confirmed during the session as specified in the architecture but not implemented. Relatedly, the reviewer asked what effect a correction has on the current mapping and on earlier mappings of the same kind, and observed that the architecture leaves this unspecified; his position was that the expected behaviour should be stated even where its execution depends on the Auto-Standardiser, so that the design commits to an intended outcome rather than deferring the question entirely. Both bear on the correction and retraining path (SR-F-04, SR-F-05), which is specified but unbuilt.

The remaining two are narrower. The reviewer wanted the option to see candidate mappings beyond the ranked top three, on the grounds that the correct target may lie outside the set the Auto-Standardiser surfaces, which the present routing and display path does not expose. He also wanted to move directly from a governance rollback back into a review surface to rebuild the profile, where the implemented workflow instead requires a new job to flow through the pipeline first. Each is recorded as future work in Chapter 8.

7.5.4 Ex-Ante Credibility of the Tier 3 Specifications

The independent reviewer's session reached the Tier 3 block, the only one of the two to do so, and supplied the practitioner reaction the ex-ante design validation calls for on tiered escalation (SR-F-12) and algorithm steering (SR-F-13).

On tiered escalation he found the specified configuration sensible and, without being told why the requirement is deferred, identified the deferral's own reason: the platform cannot place a reviewer on an experience tier without a record of how that reviewer has performed on the standard in question, which the prototype does not hold. He proposed a form that does not need that record, in which a provisional or low-confidence decision is routed to a second reviewer for an independent pass, with experience-weighted tiering layered on later once performance data accrues. This endorses the specified design and clarifies a near-term form of it, and the second-reviewer step is recorded against SR-F-12 as an interim form of the escalation the full specification describes.

On algorithm steering he endorsed the hint concept and the pull-based integration it rests on, then corrected the hint data model. He read the two systems as autonomous peers that cooperate rather than one embedding the other, with the Auto-Standardiser calling the supervision layer's endpoints and needing no knowledge of its internals, which is the integration stance the design takes (Section 5.2). On the data shape he observed that a hint belongs to a mapping profile, which is a database and version pair rather than a database alone, so the stored hint should be keyed to the profile; that a single source field may carry more than one hint, so hints should form a collection with a stable identifier each; and that the applied flag should be recorded per hint rather than once per field, so that marking a hint consumed is unambiguous when several are present. The specification in Section 5.7 is revised accordingly. The correction tightens the contract the Auto-Standardiser would integrate against and leaves the pull boundary unchanged, a single endpoint keyed by database for the outstanding hints.

7.5.5 Endorsements, Calibration, and Usability

Within the implemented workflow the participants endorsed the controls the requirements treated as essential. The mandatory sample drawn on every session was understood and accepted as the mechanism that keeps a batch from passing without any human review (SR-NF-06), and the carry-forward of deferred items into the next session was read as intended (SR-F-14). In the governance cluster the separation of duties was endorsed as the prototype enforced it, the independent reviewer noting that a data engineer could view but not roll back or approve and that the approving role was required for promotion (SR-F-09, SR-NF-05). The rollback behaviour matched what he expected of it (SR-F-07), and drift acknowledgement worked, though he noted that an acknowledged alert disappears with no way to revisit it (SR-F-08).

The independent reviewer's numeric calibration for the governance vignette was four out of five, with the shortfall attributed to small improvements rather than to a capability the system lacked, which he stated directly when asked whether the functionality met his expectations. The onboarding vignette and the Tier 3 block were not scored, a deviation from the protocol's calibration step recorded in Section 7.7.4; the single governance score is reported here as an indicative signal alongside the returning participant's comparable score for the onboarding vignette, the sample being too small to aggregate.

Usability findings are reported separately and, accessibility aside (SR-NF-07), sit outside the requirement-satisfaction claim. The independent reviewer found the terminology for profiles, versions, and review sessions hard to follow, judged the decision controls easy to overlook in their current position, and suggested that the role selector be labelled as an impersonation control to signal that it stands in for authentication. He also encountered a defect, acknowledged during the session, that allowed a decided item to remain selectable after it was locked.

7.6 Discussion

This section draws the verdict across the three methods, sets out why the design is an integrated architecture rather than an assembly of point solutions and the trade-offs that integration makes, and records the residual risks the evaluation leaves open.

7.6.1 Verdict

At the design altitude the verdict is firm. The architecture specifies a supervision element for every one of the twenty requirements (Section 7.2), those elements connect into an end-to-end workflow over the canonical UC2 job and its principal exceptions (Section 7.3), and every design decision traces to a stated requirement and a peer-reviewed or regulatory source (Table 5.5). The architecture is therefore complete with respect to the elicited requirement set and internally consistent as a specification, with the qualification that this completeness is a property of the design relative to its own requirements catalogue (the construct-validity limit in Section 7.7.1), not proof that an unbuilt component would behave as specified.

At the implementation altitude the verdict rests on the two empirical tracks. Track A is complete: all fourteen design objectives pass against the mock Auto-Standardiser (Section 7.4), so the technical half of the verdict is established for both clusters, the routing and session cluster (SR-F-01/02, SR-NF-03/06, SR-F-14) and the governance cluster (SR-F-06/07/08/09, SR-NF-05), subject to the mock-versus-real-AS limit recorded as R2. The combined verdict is read per cluster: each is judged satisfied to the extent that its design objectives pass on Track A and practitioners confirm the implemented behaviour matches the need on Track B (Section 7.5).

On that reading both clusters are judged satisfied for their implemented behaviour, with the qualifications the Track B findings attach. For the routing and session cluster the practitioners endorsed the oversight floor and the carry-forward of deferred items, and the cluster's one substantive divergence is the deliberate-non-mapping case (Section 7.5.1), a control that behaves as specified but not as the practitioners intend and that points to a missing decision type rather than a defect; the structure-overview and additional-candidate requests are extensions the cluster does not yet offer rather than failures of what it does. For the governance cluster the separation of duties, rollback, drift detection, and approval workflow were endorsed, the governance vignette scored four out of five, and the open item is the unbuilt reject-with-reason path (Section 7.5.3). The practitioner half of the verdict rests on two participants, one of them independent, which is too small a sample to generalise and is read as corroborating rather than measuring satisfaction (Section 7.7.3).

The most informative cases are those where a technical pass coincides with a practitioner divergence, since they mark a capability that works as specified but not as intended; the Track B coding procedure is built to surface them (Appendix C.7). Research Question 3 is answered by this altitude alone, against the implemented subset.

7.6.2 Why an Integrated Architecture

The architecture's distinctive contribution is not any single supervision primitive. Confidence-thresholded routing, review queues, and governance approval each appear in the reviewed literature and in commercial data-stewardship platforms (Section 3.4); what is absent from both, and from Maldonado's framework (Chapter 3), is an architecture that integrates them for a confidence-scored, batch-oriented standardisation pipeline. The gap analysis records this as several component-level absences (the batch session, profile governance, standard-aware explanation, and the oversight floor), but they share a root: the supervision concerns of this domain are not separable. They turn on one unit of work (the standardisation job a reviewer resolves as a session) and one governed artefact (the versioned mapping profile that session produces). Routing acquires meaning only as delivery into a session; the session bounds review, tracks completion, and carries the reviewer's cross-mapping context; governance reads its decisions from a submitted session; the minimum oversight floor is enforced over that session at submission; and drift is measured against the profile the session promoted. Each capability presupposes the output of another, so the requirements cannot be met by detached tools that share no session and no profile.

This is why an integrated design improves on assembling point solutions for the same requirements. Put a confidence threshold in front of a stewardship queue, a governance tool behind it, and

an audit log beside them, and the seams between those tools are where the requirements go unmet. When one architecture owns both the session and the profile, every approved mapping keeps its lineage back to the review that produced it, so the audit trail (SR-F-10) is a property of the workflow and not a log reconciled after the fact. The minimum oversight floor sits at the boundary between routing and the session and can therefore guarantee that no batch is promoted without a mandatory human sample (SR-NF-06); a routing component or a governance tool acting alone cannot, because neither holds the whole job. Drift is reported against a specific promoted profile rather than a model's aggregate performance, so an alert names the profile it diverges from (SR-F-08). The two systems meet at a single pull boundary, corrections and hints exposed as data the Auto-Standardiser reads when it is ready, which keeps the supervision layer one coherent contract instead of several the standardiser must integrate against separately; the independent reviewer endorsed this boundary in Track B (Section 7.5.4). The regulatory obligation reinforces the point, since the EU AI Act frames meaningful human oversight as one duty over the system [11], which a piecemeal assembly discharges only as far as the weakest seam between its parts.

The claim is bounded. It is not that the individual mechanisms are new, which the systematic review and the industrial survey both show they are not, but that their integration around the batch job and the versioned profile, held to a non-negotiable oversight floor, is the design the standardisation domain requires and the one no reviewed architecture or surveyed platform provides. The use-case mapping (Section 7.3) is the evidence that this combination holds together as a workflow rather than a list of features: the canonical job runs end to end through the connected components and terminates in a governed, versioned, recoverable profile. What the mapping cannot show, that the handoffs behave correctly under real inputs, is what the two empirical tracks test.

7.6.3 Trade-offs

The integrated design carries trade-offs that the evaluation should state plainly. The minimum oversight floor trades reviewer effort for an assurance that no batch escapes review entirely: it adds load that scales with the auto-accepted pool, which the proportional-plus-absolute floor bounds but does not remove. The floor's coverage of high-confidence items is a sample rather than a census, so a confidently-wrong mapping outside the sample is not corrected within a session (R6 in Section 7.6.4); the assurance the floor offers is statistical, not exhaustive. The pull-based integration with the AS, in which the supervision system makes no outbound calls and exposes corrections and hints as data, keeps the coupling clean and lets the two sides evolve independently, but it makes the AS the active party, so corrections and hints are consumed only if the AS-side pipeline is built to pull them. The independent reviewer endorsed this boundary unprompted in Track B, describing the two as autonomous systems that cooperate as peers rather than one embedding the other (Section 7.5.4). The single-reviewer session model keeps the implemented workflow simple and is the reason tiered escalation remains specified only; it is also the substitution point at which the escalation tier attaches. None of these trade-offs undermines the architecture, but each marks a boundary of what the present design commits to.

7.6.4 Residual Risks

Eight risks remain open after the evaluation. Each is recorded with the mitigation or future work that addresses it.

R1: Uncalibrated routing threshold. The auto-accept threshold default was chosen to produce a representative confidence distribution in the evaluation scenarios, not calibrated against real AS confidence scores, which are unavailable in the prototype context. *Mitigation:* calibrate against the deployed AS's observed distribution before production use; the threshold is snapshotted per session, so recalibration does not disturb in-flight sessions.

R2: Mock-versus-real AS gap. Track A exercises the components against synthetic confidence distributions, so a technical pass does not by itself establish behaviour against a live AS. *Mitigation:* re-run the design-objective suite against real AS output once available; the anti-corruption layer isolates the AS contract, so the supervision components are insulated from AS-side change.

R3: Single-reviewer session model. The implemented session assigns one reviewer, so inter-reviewer disagreement and tiered escalation (SR-F-12) cannot operate. *Mitigation:* multi-reviewer assignment and per-reviewer history seeding are the prerequisites; the routing engine is the defined substitution point for the escalation tier (Section 5.7).

R4: Mutable persistence, not an audit trail. The supervision store persists state changes but permits updates and enforces no retention policy, so the audit and compliance requirements (SR-F-10, SR-NF-04) are specified, not built. *Mitigation:* shadow the existing persistence writes with an append-only, retention-bound log at the same integration point.

R5: Accessibility not validated. The review interface is functional but has not been validated against WCAG 2.1 AA (SR-NF-07). *Mitigation:* a conformance audit and remediation; this sits outside the requirement-validation claim and is acknowledged as a limitation in Chapter 8.

R6: Residual high-confidence false-accepts. The minimum oversight floor spot-checks only a sample of the auto-accepted pool, so a mapping the Auto-Standardiser scores above the threshold but maps incorrectly is corrected only when it is drawn into the mandatory sample; one outside the sample passes uncorrected within a session. This is intrinsic to confidence routing rather than a calibration artefact, and so distinct from R1. *Mitigation:* the floor guarantees the sampled coverage is non-zero and non-bypassable, and across successive jobs the sampling trend, drift detection (SR-F-08), and the specified post-submission accuracy verification (SR-NF-02) surface systematic over-confidence at the profile level; raising `oversight_min_sample_rate` trades reviewer effort for tighter per-batch coverage.

R7: Performance and scalability unevaluated. The prototype handles each job in a single in-memory pass with no pagination or batching, and no load testing was performed, so its

behaviour at production data volumes is unmeasured. *Mitigation:* load-test against representative UC2 job sizes before production deployment; UC2's batch workflow is expected to involve modest volumes, but this has not been measured.

R8: Client-supplied role header, not authenticated identity. The prototype derives the acting role from a request header rather than from an authenticated identity, so a caller that reaches the API directly can assert any role and bypass the separation of duties the Role Guard enforces (SR-F-09, SR-NF-05). This is a property of the prototype as a demonstrator against a mock Auto-Standardiser, not of the architecture: the role check sits at the API boundary and the header is the prototype's stand-in for a verified credential, an exclusion the design states as a scope decision (Section 5.8) rather than a gap the validation exposed. *Mitigation:* replace the header with an OAuth2 bearer token in the authorization header, issued by an external federated identity provider autonomous from the platform and validated server-side with the role derived from the token; the change is confined to the boundary that already reads the role. Raised in Track B (Section 7.5.2) and taken up as future work in Chapter 8.

7.7 Threats to Validity

The evaluation combines an analytical demonstration, a technical track, and a practitioner track, and each is open to threats that qualify the conclusions drawn from it. The threats are organised along the standard construct, internal, and external dimensions, with a closing note on reliability [36].

7.7.1 Construct Validity

Construct validity concerns whether the evaluation measures what it claims to measure. Three threats apply. First, the twenty requirements are the yardstick for the whole evaluation, and they were synthesised from the T7.5/UC2 deliverable, two requirements interviews, and the PRISMA corpus; a requirement the elicitation missed would not appear as a gap, because the same catalogue defines both the design and its evaluation. Second, Track A measures design-objective satisfaction, and the conclusion that an objective's passing implies its requirement is satisfied is only as good as the objective-to-requirement mapping in Table 5.2; an objective can be met while leaving part of its requirement unexercised. Third, and most consequential, Track A runs against a mock Auto-Standardiser producing synthetic confidence distributions. The mock reproduces the AS interface contract, not its statistical behaviour, so a pass establishes that the components behave correctly against the contract, not that they behave correctly against the confidence distributions a trained AS would produce. The use-case mapping reduces but does not remove this threat, since it too reasons about the contract rather than observed AS output. A further limit applies to the demonstration specifically: it shares an author with both the architecture it traces and the scenario it traces them through, so it can reveal only a flow step that no component owns, not whether the

handoffs between components are correct. Its evidence is therefore about coverage and connectivity, which Track A and Track B then test for behaviour.

7.7.2 Internal Validity

Internal validity concerns whether the evidence supports the conclusions without the researcher's involvement skewing them. A single researcher designed the architecture, built the prototype, and conducted the evaluation, so every measurement is taken on an artefact that one person produced. This shows up in three places. The Track A test scenarios and their ground-truth confidence distributions were authored by that same researcher, so they could unintentionally exercise the cases the implementation handles and avoid the ones it does not. The Track B sessions were run and coded by the same researcher, which the protocol counters with show-then-ask sequencing, pre-committed disconfirming prompts, and disconfirming-first reporting, but which it cannot remove (Appendix C). The qualitative coding is single-coder, with no second rater to establish inter-coder agreement. Two further consequences follow from the same person building and judging the artefact. The check that the build matches the Chapter 5 specification was itself internal, carried out by the same researcher by holding the implementation against the design section by section (Section 6.8) rather than by independent audit, so a divergence the researcher did not notice would not be caught. And because one design objective, DO-7, had its specification reconciled across two design sections during construction (Section 6.8), the fit between that objective and the built artefact is in part one the same researcher consolidated; the reconciliation closed an obligation the design already carried rather than bending the objective to the code, and the remaining thirteen objectives were fixed before construction. These are mitigated by procedure rather than eliminated, and the evaluation's conclusions should be read with that in mind.

7.7.3 External Validity

External validity concerns how far the findings generalise. The architecture is designed for and evaluated against one use case in one domain (UC2 energy-data standardisation in the NOUS project), so its fit to other standardisation settings is argued by the generality of the requirements, not demonstrated. The practitioner sample is two participants, one a domain expert who contributed to the requirements interviews and one an independent reviewer with no involvement in the design. This limits how far the Track B findings generalise. The returning participant carries a confirmation-bias risk, which the protocol addresses by prioritising an independent reviewer as the primary control; that reviewer was recruited and contributed the strongest disconfirming evidence in the set (Section 7.5.2), which reduces the risk without removing it at this sample size (Appendix C). The evaluated artefact is a prototype against a mock AS, not a production deployment, so properties that emerge only at production scale or against a live model are out of reach. Finally, the Tier 3 components are validated ex ante only, so any claim about them is a claim about a design, not about a system in use.

7.7.4 Reliability

Reliability concerns whether the evaluation would yield the same findings if repeated. Track A is the more reproducible of the two: the components are deterministic given a fixed configuration, the routing parameters are snapshotted per session, and each objective is checked against the job the mock actually emitted rather than against assumed values, so the design-objective outcomes are repeatable across runs even though the individual confidence draws within a scenario vary. The run records its per-objective checks, scenario inputs, and outcomes to a stored artifact, so the technical results can be regenerated and re-inspected. Track B is less reproducible, as qualitative interview findings depend on the participants and the session; the protocol provides for verbatim transcription of the interview recordings, member checking, and a fixed coding scheme applied uniformly across sessions (Appendix C.7), so that the coding, if not the raw reactions, can be re-examined.

7.8 Summary

This chapter evaluated the supervision architecture at two altitudes. Ex-ante design validation traced the full architecture for all twenty requirements through the use-case mapping of Section 7.3 and the coverage table of Section 7.2. Ex-post implementation validation combined Track A, which exercised the built components against the mock Auto-Standardiser and found all fourteen design objectives satisfied (Section 7.4), with Track B, a two-participant practitioner validation that endorsed the session, governance, and oversight design while surfacing an authentication gap and a deliberate-non-mapping divergence (Section 7.5). The architecture is complete with respect to the elicited requirement set, subject to the residual risks and threats to validity recorded in Sections 7.6.4 and 7.7. Chapter 8 draws together the contributions, revisits the research questions, and sets out future work.

Chapter 8

Conclusion

This chapter closes the dissertation. Section 8.1 recaps the problem and the work done in response, Section 8.2 revisits the three research questions, and Section 8.3 states the contributions as delivered. Section 8.4 considers what the work means beyond its immediate setting, Section 8.5 records the limitations that bound the claims, and Section 8.6 sets out the future work the architecture specifies but leaves unbuilt. Section 8.7 closes.

8.1 Overview

Automated semantic matching scales the schema-mapping work that manual effort cannot, but it produces matches that are probable, not certain, so a person must review the uncertain ones, correct them, and return those corrections for the system to improve [34]. Within the NOUS project this supervision is part of how the T5.4 Auto-Standardiser is specified to work, not a fallback for when it fails. The supervision layer around that core was left open. General human-in-the-loop architectures share the review-and-retrain loop the Auto-Standardiser keeps, but the Auto-Standardiser diverges from them on five specific properties: confidence-based routing calibrated to a required quality level, a batch session workflow rather than a real-time stream, correction-driven retraining that must be governed, a mapping-profile governance lifecycle with no human-in-the-loop equivalent, and multi-standard explainability. The closest prior work within NOUS, Maldonado's general framework, named its transfer to other sectors as an open question [25].

This thesis addressed that gap through design science research, in three contributions. It elicited the supervision requirements of automated data standardisation and set them against existing architectures to establish what they leave unaddressed. It specified a supervision architecture for the full set of requirements and implemented the components realisable without the live Auto-Standardiser machine-learning pipeline, together with a mock Auto-Standardiser that supplies the evaluation environment. And it evaluated the implemented components against that mock and with practitioners.

The outcome is twofold. As a specification the architecture is complete with respect to the elicited requirements and internally consistent. That completeness is measured against the thesis's

own requirements catalogue. As a built system the implemented subset satisfies the requirements it targets under technical evaluation, with the qualifications the practitioner validation attaches and the residual risks the evaluation records.

8.2 Revisiting the Research Questions

The three research questions are answered in turn, each drawing on the chapters that supply its evidence.

RQ1 asked what supervision requirements arise in automated data standardisation workflows, and how far existing human-in-the-loop architectures address them. Twenty requirements were elicited from the T5.4 development plan, the UC2 acceptance criteria, two practitioner interviews, and the human-in-the-loop literature, spanning the five challenge areas that distinguish the domain: confidence-based routing, batch session workflow, correction-driven retraining, mapping-profile governance, and multi-standard explainability (Chapter 4). Set against a baseline of the peer-reviewed corpus surveyed under PRISMA, the industrial data-stewardship platforms (Section 3.4), and Maldonado's NOUS framework, the coverage is uneven: two requirements are fully addressed, nine partially, and nine not at all. The deepest gaps are structural: the batch review session as the unit of work, standard-aware explainability, the mapping-profile governance artefact class, algorithm steering for unmapped columns, and a minimum oversight floor that never lets a batch pass unreviewed. None is closed by parameterising a general human-in-the-loop architecture or by composing the primitives that commercial stewardship platforms already provide. This coverage map is the answer to RQ1 and the foundation for what follows.

RQ2 asked how those requirements should be addressed through architecture design and selective implementation in the NOUS Auto-Standardiser context. Chapter 5 specifies the NOUS AS Supervision Architecture, which carries a design element for every one of the twenty requirements, with its design objectives, its decisions each traced to a requirement and a peer-reviewed or regulatory source, and its components grouped into the routing, session, and governance work the domain demands. The selective-implementation boundary follows feasibility: the components realisable without the live Auto-Standardiser pipeline were built (Chapter 6). The rest remain specified only (Section 5.7): retraining, tiered escalation, hint execution, and the audit and accuracy-verification requirements. The architecture is grounded in peer-reviewed literature independently of Maldonado, whose work is context for the problem, not authority for any design choice. The specified architecture and its implemented subset are the answer to RQ2.

RQ3 asked to what extent the implemented architecture addresses the requirements, assessed against a mock Auto-Standardiser API and with practitioners. The evaluation operated at two altitudes (Chapter 7). At the design altitude the verdict is firm. Eighteen of the twenty requirements are realised by a design element; the two boundary cases sit outside the supervision flow: column coverage upstream of it and accessibility as a limitation. Those elements connect into an end-to-end workflow over the canonical UC2 job and its exceptions (Section 7.3), and every design decision is grounded, with the qualification that this completeness is relative to the requirements catalogue the

thesis itself produced. At the implementation altitude the technical track is complete: all fourteen design objectives pass against the mock Auto-Standardiser (Section 7.4), so both the routing-and-session cluster and the governance cluster are satisfied for their implemented behaviour, subject to the mock-versus-real-AS limit. The practitioner track (two participants, one of them independent of the design) corroborates rather than measures. The oversight floor and the carry-forward of deferred items were endorsed; the independent reviewer, whose session reached the governance vignette, endorsed the separation of duties, rollback, drift detection, and the approval workflow. The track also surfaced one substantive divergence together with one scope gap. The divergence is the Skip control, which behaves as specified but not as reviewers intend and points to a missing deliberate-non-mapping decision (Section 7.5.1). The scope gap is the role header, a prototype stand-in for authentication that a direct caller could forge; the design already records authentication as a boundary placed outside the prototype (R8, Section 7.5.2). Both are carried as future work, not as defects in what the components do.

These answers bear on the hypothesis. The thesis held that the supervision components realisable without the live Auto-Standardiser pipeline can satisfy the requirements they were built to address, a claim of both sufficiency and compatibility. The evaluation supports the sufficiency half for the implemented subset: the components meet their requirements without displacing the problem, within the stated qualifications. The compatibility half is a property of the design, not an evaluated outcome. The architecture builds on the review-and-retrain loop general human-in-the-loop practice shares and adds the domain-specific supervision layer around it, so it extends the framework Maldonado established for NOUS instead of replacing it (Chapter 5). The claim was never that the implementation generalises beyond the UC2 energy domain or that it resolves every supervision requirement, and those parts remain open: by design for the architecture-specified components and by scope for the domain.

8.3 Contributions

The three contributions can now be stated as delivered.

C1: requirements analysis. A domain-specific catalogue of twenty supervision requirements for automated data standardisation and a structured gap analysis that locates each against what existing architectures already provide, across the peer-reviewed corpus, the industrial platforms, and Maldonado's NOUS work. The output is not a bare list but the coverage map that separates the partially and wholly unaddressed requirements from those general practice already handles (Chapter 4).

C2: architecture and implementation. A full supervision architecture specification for all twenty requirements, an implementation of the supervision workflow components realisable without the live Auto-Standardiser pipeline, and a mock Auto-Standardiser API that doubles as the evaluation infrastructure for C3 (Chapters 5 and 6). Its contribution is the integration itself rather than any single mechanism: the individual primitives are familiar from the literature and from commercial platforms, but no prior architecture assembles them into one supervision workflow for

confidence-scored, batch-oriented standardisation. The implemented system covers the routing, session, and governance components as a connected workflow over the batch standardisation job; the remaining components are specified at their integration points.

C3: empirical evaluation. An evidence-based answer to RQ3 from two complementary tracks: a technical evaluation in which all fourteen design objectives passed against the mock Auto-Standardiser, and a practitioner validation that tested whether the implemented behaviour matched the supervision needs identified during requirements analysis (Chapter 7).

8.4 Implications and Generalisability

The requirements catalogue and the supervision patterns the architecture introduces are framed around automated data standardisation in general, not around a single deployment. The batch review session as the unit of supervision, the minimum oversight floor, and the treatment of approved mappings as versioned, rollback-capable governance artefacts respond to properties any confidence-scored, batch-oriented standardisation pipeline shares, so they are candidates for reuse wherever such a pipeline needs human oversight. The regulatory pull is general in the same way: the EU AI Act makes meaningful human oversight an obligation for high-risk systems [11], and the GDPR constrains how the resulting records are kept, so the supervision layer is a compliance surface, not optional infrastructure.

The limit of this claim should be stated as plainly as the claim itself. Transferability is argued from the generality of the requirements, not demonstrated. The architecture is designed for and evaluated against one use case in one domain (UC2 energy-data standardisation), so whether it fits mobility, Green Deal, or non-energy standardisation contexts is an open question. Establishing that fit empirically is left to future work, and the external-validity limit recorded in Section 8.5 bounds how far the present findings should be read.

8.5 Limitations

The contributions hold within boundaries the evaluation makes explicit. Chapter 7 records them in full, as a residual-risk register (Section 7.6.4) and a threats-to-validity analysis (Section 7.7); the limitations that most qualify the thesis-level claims are drawn together here.

The first group bounds the evidence. Track A runs against a mock Auto-Standardiser that reproduces the interface contract but not the statistical behaviour of a trained model, so a technical pass establishes correctness against the contract and the imposed confidence distributions, not against live AS output (R2); the routing threshold was chosen for the evaluation, not calibrated to real scores (R1). A passing design objective is bounded by its mapping to the requirement it stands for, so objective satisfaction supports a requirement without fully establishing it. A single researcher designed the architecture, built the prototype, authored the Track A scenarios, and ran and coded the Track B sessions with no second coder, so the evidence is taken on an artefact one person produced, mitigated by procedure but not removed. The practitioner sample is two participants, too small

to support a general claim. One is an independent reviewer; the other is a returning requirements participant whose involvement carries a confirmation-bias risk the independent reviewer offsets but does not remove. The architecture is evaluated against a single domain, so its transfer beyond UC2 is unestablished.

The second group bounds the built system. The session model assigns a single reviewer, so inter-reviewer disagreement and tiered escalation cannot operate (R3). The supervision store persists state changes but permits updates and enforces no retention policy, so the audit-trail and compliance requirements are specified, not built (R4). The review interface is functional but has not been validated against WCAG 2.1 AA (R5). The minimum oversight floor spot-checks a sample, not the whole auto-accepted pool, so a confidently-wrong mapping outside the sample is not corrected within a session (R6). Performance at production data volumes is unmeasured (R7). And the prototype derives the acting role from a request header rather than an authenticated identity, so a direct caller could assert any role (R8). Each is a boundary of the prototype as a demonstrator, not a flaw in the architecture, and each has a defined route to closure recorded with the risk.

8.6 Future Work

The architecture specifies more than the prototype builds, and the evaluation surfaced needs the requirements catalogue did not carry. The future work falls into three groups: refinements the validation identified, completion of the specified-but-unbuilt architecture, and longer-term directions that close the evaluation's open risks.

8.6.1 Refinements Identified Through Validation

The practitioner validation (Chapter 7) produced a set of concrete extensions, each building on the existing components instead of requiring new ones.

A first-class non-mapping decision. The validation found that reviewers need to record a deliberate judgement that a source column has no valid target, distinct from deferring it and from its being unmapped by default (Section 7.5.1). The prototype expresses neither case: a skipped field and a deliberately excluded one both leave the profile without an entry, so the profile cannot tell a column a reviewer decided not to map from one not yet reviewed, which weakens the provenance the audit-trail requirement calls for. The extension adds an explicit non-mapping decision to the item vocabulary, recorded as positive intent and carried into the profile as an entry with a null target, so the exclusion is auditable and a later session does not re-surface a column already decided. Raised during requirements elicitation and independently corroborated by the external reviewer, it is a candidate supervision requirement in its own right, unimplemented only because it postdates the catalogue that fixed the build scope.

Federated authentication and authorisation. The prototype derives the acting role from a request header, which a client reaching the API directly can forge, so the separation of duties the governance design depends on is not enforceable as built (Section 7.5.2, residual risk R8). The extension replaces the header with token-based authentication: the supervision layer accepts an

OAuth2 bearer token issued by an external federated identity provider it does not own, validates it server-side, and derives the role from its claims. Because the supervision layer is operated from within the Auto-Standardiser's environment, federating identity lets a caller authenticate once against its own organisation's provider and reuse that identity across both systems. The change is confined to the boundary that already reads the role, leaving the routing, session, and governance components unaffected.

Whole-dataset structural overview. Both elicitation and validation pointed to a reviewer's need to see the source structure before standardisation and the target structure after it, not only the per-item evidence the queue presents (Section 7.5.3). A reviewer who joins a session partway through, or who wants to reason about coverage across the dataset, has no such view. The extension adds a structural overview to the session surface, derived from data already captured per item. It shows the input and target structures together and marks which fields are mapped, flagged, deferred, or excluded. It composes with the session model without altering it.

Reject-with-reason and rejection memory. The governance surface lets a lead approve or roll back a profile but not reject a proposed one with a stated reason, and the validation argued that a rejection should be both explainable at the time and visible on later runs, so a subsequent session can take prior rejections into account when the same mappings recur (Section 7.5.3). The extension records a rejection as a governance event carrying the rejecting role, a reason, and the mappings it concerns. It surfaces prior rejections when those mappings reappear. It extends correction capture to the reject side and strengthens the provenance the audit-trail requirement calls for.

8.6.2 Completing the Specified Architecture

Several requirements are specified at their integration points but left unbuilt, because they depend on the live Auto-Standardiser pipeline or on data the prototype does not hold (Section 5.7).

Standard-partitioned retraining (SR-F-05) consumes the corrections the supervision layer already exports; completing it requires the live retraining pipeline on the Auto-Standardiser side, and the validation added that the expected effect of a correction on later and related mappings should be specified even where its execution is AS-side, so the design commits to an outcome instead of deferring the question (Section 7.5.3). Tiered escalation (SR-F-12) needs reviewer-performance history the prototype does not record; the validation proposed an interim form that does not need it, routing a provisional or low-confidence decision to a second reviewer for an independent pass and layering experience-weighted tiering on once performance data accrues. This second-reviewer step exercises the multi-reviewer session path the full tier requires (Section 7.5.4).

Algorithm-steering execution (SR-F-13) has its endpoints and schema in place but not the reviewer-side capture control or the AS-side consumption; the validation also refined its data model, and the specification now reflects the result: a hint keyed to a mapping profile rather than to a database alone, several hints per field, and an applied flag recorded per hint (Section 7.5.4). The audit trail and retention requirements (SR-F-10, SR-NF-04) and post-submission accuracy verification (SR-NF-02) complete in the same manner, by shadowing the existing persistence at the integration points the architecture already defines.

8.6.3 Longer-Term Directions

Three directions extend beyond completing the present design. The first is to replace the mock Auto-Standardiser with the live system and re-run the design-objective suite against real output, which closes the mock-versus-real gap (R2) and lets the routing threshold be calibrated to observed confidence scores (R1); the anti-corruption layer that isolates the AS contract is the point at which this substitution attaches. A second is a conformance audit and remediation of the review interface against WCAG 2.1 AA (R5), which moves accessibility from an acknowledged limitation to a designed property. The widest in scope is to test the architecture's transfer beyond UC2, applying the requirements catalogue and the supervision components to a mobility or Green Deal standardisation context and measuring how much carries over. This is the empirical counterpart to the generalisability argued in Section 8.4.

8.7 Closing Remarks

As automated standardisation becomes the means by which heterogeneous data sources are combined across European data spaces, the human oversight of that automation moves from good practice to a regulated requirement [11]. The supervision a confidence-scored, batch-oriented standardisation pipeline needs is not the supervision a real-time stream needs, and treating the two as the same leaves the gaps this thesis documented. For the NOUS Auto-Standardiser, the work delivers a concrete step: a supervision architecture specified for its full requirement set and a prototype that satisfies the realisable part under evaluation. Beyond that, it delivers a requirement catalogue and design that frame the wider problem for standardisation pipelines. What it specifies but leaves unbuilt, and what the validation showed it should yet do, mark where that work continues.

References

[1] Saleema Amershi et al. “Power to the people: The role of humans in interactive machine learning”. In: *AI magazine* 35.4 (2014), pp. 105–120.

[2] Shubhi Asthana and Ruchi Mahindru. “Mapping of Financial Services datasets using Human-in-the-Loop”. In: *Proceedings of the Third ACM International Conference on AI in Finance*. 2022, pp. 183–191.

[3] Claus Bossen and Kathleen H Pine. “Batman and Robin in healthcare knowledge work: Human-AI collaboration by clinical documentation integrity specialists”. In: *ACM Transactions on Computer-Human Interaction* 30.2 (2023), pp. 1–29.

[4] Kaiwen Chen et al. “Reliable text-to-sql with adaptive abstention”. In: *Proceedings of the ACM on Management of Data* 3.1 (2025), pp. 1–30.

[5] Taeyoung Choe et al. “BERT-Based Schema Matching for Integrating Heterogeneous Flood Data: A Case Study in Korea”. In: *Systems* 14.3 (2026), p. 267.

[6] Peter Christen. *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Data-Centric Systems and Applications. Berlin, Heidelberg: Springer, 2012. ISBN: 978-3-642-31163-5. DOI: 10.1007/978-3-642-31164-2.

[7] Collibra. *Data Governance*. https://www.collibra.com/products/data-governance. Accessed 2026-06-18. 2024.

[8] Daniel Deutch, Evgeny Marants, and Yuval Moskovitch. “Datalignment: ontology schema alignment through datalog containment”. In: *Proceedings of the VLDB Endowment* 12.12 (2019), pp. 1870–1873.

[9] Xin Luna Dong and Theodoros Rekatsinas. “Data integration and machine learning: A natural synergy”. In: *Proceedings of the 2018 international conference on management of data*. 2018, pp. 1645–1650.

[10] ETSI. *SmartM2M; Smart Appliances; Reference Ontology and oneM2M Mapping*. Tech. rep. ETSI TS 103 264 V2.1.1. European Telecommunications Standards Institute, 2017.

[11] European Data Protection Supervisor. *AI Act Regulation (EU) 2024/1689 – Regulation (EU) 2024/1689 of the European Parliament and of the Council of 13 June 2024 laying down harmonised rules on artificial intelligence and amending Regulations (EC) No 300/2008, (EU) No 167/2013, (EU) No 168/2013, (EU) 2018/858, (EU) 2018/1139 and (EU) 2019/2144 and Directives 2014/90/EU, (EU) 2016/797 and (EU) 2020/1828 (Artificial Intelligence Act) (Text with EEA relevance)*. Publications Office of the European Union, 2025. DOI: doi/10.2804/4225375.

[12] António Oliveira Ferreira. “Architectural design for the integration of Federated Learning Strategies: the NOUS Project use case”. MA thesis. Universidade do Porto (Portugal), 2025.

[13] Yonatan Geifman and Ran El-Yaniv. "Selective classification for deep neural networks". In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017, pp. 4878–4887.

[14] Jinsong Guo et al. "When Automatic Filtering Comes to the Rescue: Pre-Computing Company Competitor Pairs in Owler". In: *Proceedings of the ACM on Management of Data* 1.2 (2023), pp. 1–23.

[15] Neal R. Haddaway et al. "PRISMA2020: An R package and Shiny app for producing PRISMA 2020-compliant flow diagrams, with interactivity for optimised digital transparency and Open Synthesis". In: *Campbell Systematic Reviews* 18.2 (June 2022), e1230. ISSN: 1891-1803. DOI: 10.1002/cl2.1230. URL: https://doi.org/10.1002/cl2.1230.

[16] Alan R Hevner et al. "Design science in information systems research1". In: *MIS quarterly* 28.1 (2004), pp. 75–106.

[17] Andreas Holzinger. "Interactive machine learning for health informatics: when do we need the human-in-the-loop?" In: *Brain informatics* 3.2 (2016), pp. 119–131.

[18] IEC. *Application Integration at Electric Utilities – System Interfaces for Distribution Management – Part 13: CIM RDF Model Exchange Format for Distribution*. Tech. rep. IEC 61968-13 Edition 1.0. International Electrotechnical Commission, 2008.

[19] IEC. *Communication Networks and Systems for Power Utility Automation – Part 7-420: Basic Communication Structure – Distributed Energy Resources Logical Nodes*. Tech. rep. IEC 61850-7-420 Edition 1.0. International Electrotechnical Commission, 2009.

[20] IEC. *Energy Management System Application Program Interface (EMS-API) – Part 301: Common Information Model (CIM) Base*. Tech. rep. IEC 61970-301 Edition 2.0. International Electrotechnical Commission, 2009.

[21] Informatica. *CLAIRE AI Engine*. https://www.informatica.com/platform/claire-ai.html. Accessed 2026-06-18. 2024.

[22] Arjit Jain, Sunita Sarawagi, and Prithviraj Sen. "Deep indexed active learning for matching heterogeneous entity representations". In: *arXiv preprint arXiv:2104.03986* (2021).

[23] Jason Koh et al. "Scrabble: transferrable semi-automated semantic metadata normalization using intermediate representation". In: *Proceedings of the 5th Conference on Systems for Built Environments*. 2018, pp. 11–20.

[24] Spyros Kolovos. *Development Plan of Task 5.4 of NOUS: The Auto-Standardiser Tool for Translation to Data Standards*. Tech. rep. Internal project deliverable, Task 5.4. AETHON Engineering, NOUS Horizon Europe Project, 2025.

[25] João Maldonado. "Architectural design for Human-Centered AI Systems: the NOUS Project use case". MA thesis. Universidade do Porto (Portugal), 2025.

[26] Jakob Martin et al. "Enhancing Construction Data Integration through Dynamic Ontology Alignment and Automated Attribute Mapping". In: *EC3 Conference 2025*. Vol. 6. European Council on Computing in Construction. 2025, pp. 0–0.

[27] Venkata Vamsikrishna Meduri et al. "Alfa: active learning for graph neural network-based semantic schema alignment". In: *The VLDB Journal* 33.4 (2024), pp. 981–1011.

[28] Robert Munro Monarch. *Human-in-the-Loop Machine Learning: Active learning and annotation for human-centered AI*. Simon and Schuster, 2021.

[29] Alberto Montero Fernández et al. "A catalyst for European cloud services in the era of data spaces, high-performance and edge computing: NOUS". In: *Proceedings of the 4th Eclipse Security, AI, Architecture and Modelling Conference on Data Space*. 2024, pp. 1–9.

[30] NOUS Consortium. *UC2 User Stories: Energy Prediction and Data Lifecycle Management*. Tech. rep. Internal project deliverable, version 2; Use Case 2 Story 1.1 acceptance criteria. NOUS Horizon Europe Project, 2025.

[31] Matthew J Page et al. "The PRISMA 2020 statement: an updated guideline for reporting systematic reviews". In: *BMJ* 372 (2021). DOI: 10.1136/bmj.n71. eprint: https://www.bmj.com/content/372/bmj.n71.full.pdf. URL: https://www.bmj.com/content/372/bmj.n71.

[32] Ken Peffers et al. "A design science research methodology for information systems research". In: *Journal of management information systems* 24.3 (2007), pp. 45–77.

[33] Eric D Ragan et al. "Balancing privacy and information disclosure in interactive record linkage with visual masking". In: *Proceedings of the 2018 CHI conference on human factors in computing systems*. 2018, pp. 1–12.

[34] Erhard Rahm and Philip A Bernstein. "A survey of approaches to automatic schema matching". In: *the VLDB Journal* 10.4 (2001), pp. 334–350.

[35] Adil Rahman, Koichiro Niinuma, and Aakar Gupta. "DataSpeck: An AI-Driven Human-in-the-Loop System for Automating Transformations in Data Conversion Workflows". In: *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems*. 2026, pp. 1–28.

[36] Per Runeson and Martin Höst. "Guidelines for conducting and reporting case study research in software engineering". In: *Empirical Software Engineering* 14.2 (2009), pp. 131–164. DOI: 10.1007/s10664-008-9102-8.

[37] Burr Settles. "Active learning literature survey". In: (2009).

[38] Lindsay T Sharpe et al. "Opsin genes, cone photopigments and color vision". In: *Color vision: From genes to perception* (1999), pp. 3–51.

[39] Roee Shraga, Ofra Amir, and Avigdor Gal. "Learning to characterize matching experts". In: *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE. 2021, pp. 1236–1247.

[40] Roee Shraga, Avigdor Gal, and Haggai Roitman. "What type of a matcher are you? coordination of human and algorithmic matchers". In: *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*. 2018, pp. 1–7.

[41] Pavel Shvaiko and Jérôme Euzenat. "Ontology matching: state of the art and future challenges". In: *IEEE Transactions on Knowledge and Data Engineering* 25.1 (2013), pp. 158–176.

[42] Parwinder Singh et al. "LLM-Based Harmonized Data Ingestion for Dataspace: A Novel System for Automating Data Ingestion Across Heterogeneous Data Sources". In: *Knowledge-Based Systems* (2026), p. 115800.

[43] Matan Solomon, Bar Genossar, and Avigdor Gal. "Copychats: Question sequencing with artificial agents". In: *Proceedings of the 2024 Workshop on Human-In-the-Loop Data Analytics*. 2024, pp. 1–7.

[44] Christian Sonnenberg and Jan vom Brocke. "Evaluations in the science of the artificial: reconsidering the build-evaluate pattern in design science research". In: *Design Science Research in Information Systems. Advances in Theory and Practice (DESRIST 2012)*. Vol. 7286. Lecture Notes in Computer Science. Springer, 2012, pp. 381–397.

[45] Tamr, Inc. *AI/ML-Based Data Mastering*. Product documentation, https://www.tamr.com/ai-ml-mastering. Accessed 2026-06-18. 2024.

[46] John Venable, Jan Pries-Heje, and Richard Baskerville. "FEDS: a framework for evaluation in design science research". In: *European Journal of Information Systems* 25.1 (2016), pp. 77–89.

[47] Zingg AI. *Zingg: Open-Source Entity Resolution and Master Data Management*. https://www.zingg.ai/. Accessed 2026-06-18. 2024.

Appendix A

Requirements Analysis: Supplementary Material

This appendix reproduces the full semi-structured interview protocol used to conduct the two practitioner interviews referenced in Chapter 4 (Sections 4.2 and 4.3). The protocol covers the interview's purpose and study context, interviewee and interviewer information including an explicit reflection on the interviewer's biases, ethical considerations and consent, the interview guide proper (the question set used in both sessions), the closing exchange, and the post-interview procedure that fed coded findings into the supervision requirements catalogue in Section 4.4.

A.1 Protocol Metadata

Thesis title	Integrating Human Oversight in Automated Data Standardisation: the NOUS Project use case
Method	Semi-structured qualitative interviews with practitioners
Interviewer	Sérgio Peixoto (MSc candidate, FEUP)
Date	[insert date]
Interviewee	[insert name and role]
Setting	[insert setting]
Duration	45–60 minutes

A.2 Purpose of the Interviews and Study Context

The interviews supported Research Question 1 of this dissertation: identifying the supervision requirements that arise in automated data standardisation workflows, and assessing the extent to which they are addressed by existing human-in-the-loop architectures. Formal specifications (the T5.4 development plan and the UC2 acceptance criteria) describe what a supervision layer must do; they do not record how practitioners would work with a deployed review interface, what

information they need under uncertainty, or what conventions govern escalation. The interviews captured this practitioner-side evidence to:

1. Surface supervision requirements not derivable from documentary sources.
2. Validate, refine, or contest the supervision dimensions identified by the systematic literature review in Chapter 3.
3. Provide grounding for design decisions that the predecessor NOUS HITL framework [25] treats only at the level of cross-use-case policy tuning.

A.3 Interviewee and Interviewer Information

Interviewee Selection and Background

The two participants were recruited to represent complementary vantage points. Participant 1 was selected on the basis of direct operational familiarity with the AS platform and its planned deployment in the UC2 energy domain. Participant 2 was selected on the basis of established methodological expertise in human-annotation workflow design and multi-fidelity human–AI systems, sourced from a discipline (corpus linguistics) with a systematic literature on escalation protocols. The first vantage point reflects operational practice within the AS context; the second brings external methodology against which AS-specific assumptions can be checked. At session start the following items were captured for each participant: role and focus area; years of relevant professional or research experience; and typical responsibilities pertinent to data standardisation, annotation, or human oversight workflows.

Interviewer and Reflection on Biases

The interviewer is Sérgio Peixoto, MSc candidate at FEUP and author of this dissertation. Two structural biases were identified before data collection and explicitly accounted for in the conduct of each session.

Designer-confirmation bias. I am concurrently developing the supervision architecture this dissertation specifies. Although the interviews were intended to elicit requirements rather than to validate a pre-existing design, the parallel development creates a natural tendency to interpret responses in ways that confirm assumptions already shaped by the architecture in progress. To mitigate this bias I committed to: (i) probing for disagreement when responses aligned with my working architectural assumptions; (ii) recording contradictory or unexpected statements verbatim in the transcript and carrying them forward into thematic coding rather than discarding them; (iii) attributing each requirement to the participant's exact wording where possible, so that downstream readers can audit the chain from response to requirement.

Project-insider bias. When a participant is a colleague within the NOUS project who shares my institutional context, the shared background reduces friction in domain-specific discussion

but also encourages shortcuts — unspoken assumptions, in-group framings — that can obscure positions a non-insider would have probed further. To mitigate this bias I included at least one participant external to the NOUS project, so that domain-specific terminology in insider responses could be cross-checked against an outsider's framing of the same problem areas.

A.4 Ethical Considerations and Consent

Both interviews followed the ethical guidelines below, which were communicated to each participant before the session began.

- **Voluntary participation.** Participation is voluntary; participants may decline any question or withdraw at any time without consequence.
- **Confidentiality and anonymisation.** Role-only anonymisation is offered as the default, with stronger anonymisation available on request.
- **Recording.** Audio is recorded solely for transcription accuracy and is deleted after the thesis is graded.
- **Right to review.** Each participant receives the transcript for member checking, with a five working-day window to request corrections or redactions before any quoted material is incorporated into the dissertation.
- **Data usage.** The data are used exclusively for this Master's dissertation.
- **Time commitment.** The interview is expected to last between 45 and 60 minutes.

A.5 Interview Guide

The guide is thematic rather than scripted. Each topic area names what the section is intended to surface, opens with one or two prompts to begin the discussion, and lists sub-topics that I aimed to cover before moving on. Wording and order were adapted to the participant's role; sub-topics raised spontaneously by the participant were not re-asked. The scenarios in the final block functioned as discussion vignettes rather than question sets, used when time permitted to ground earlier abstract responses in a concrete case.

Topic 1 — Role and Current Practice (~10 min)

Purpose: establish the participant's role, day-to-day relationship with data, and any first-hand experience of standardisation problems and their downstream consequences.

Opening prompt. How would you describe your role and the way you work with data day-to-day?

Sub-topics to explore:

- Direct experience with data standardisation, field mapping, or schema integration.
- The shape of typical onboarding for a new dataset: manual mapping or reformatting steps, time cost, and the main friction points.
- Past incidents involving a mislabelled field, an incorrect standard mapping, or wrong units — how the error was detected, and the impact on downstream consumers such as ML training or reporting.

Topic 2 — Supervision Needs and Authority (~15 min)

Purpose: surface views on reviewer authority, the set of actions a reviewer should be able to take, the conditions under which supervision intensity may change, and how disagreement between reviewers should be resolved.

Opening prompt. If an automated system produced a complete set of suggested mappings for a new dataset, who should have the final say on whether they are correct?

Sub-topics to explore:

- Single-reviewer authority versus multi-reviewer involvement; dependence on confidence or on the criticality of the data.
- The reviewer's action set: batch approval, per-field approval, correction, rejection, pause; which actions are essential versus nice-to-have.
- Whether mature pipelines should shift to exception-based autonomy, and what signal would trigger a return to full review (confidence drop, drift alert, scheduled spot-check).
- Configurability of supervision intensity as a function of data importance or available resources.
- Resolution of inter-reviewer disagreement: weighting by expertise, disagreement-triggered re-processing, and the existence of a tie-breaking role.

Topic 3 — Information Needs for Review (~10 min)

Purpose: identify what information the reviewer needs to make a decision under uncertainty, and how that information requirement varies with the target standard.

Opening prompt. When you are reviewing an uncertain mapping, what information do you need in front of you to decide?

Sub-topics to explore:

- Sufficiency of a similarity score plus a top-k candidate list, and the appropriate value of k.
- The role of ontological context: the candidate field's definition, its position in the standard's hierarchy, and the neighbouring concepts that ranked just below.

- Whether the information requirement differs across target standards (SAREF, CIM, IEC 61850).
- Reviewer behaviour for columns the system could not map at all: expected forms of system assistance (partial matches, semantic hints, analogues from similar datasets) versus turning to a colleague.

Topic 4 — Quality, Monitoring, and Alerts (~10 min)

Purpose: capture views on when the system should alert a reviewer, when uncertain mappings may flow downstream provisionally, and what behaviour is acceptable when no reviewer is available.

Opening prompt. In what situations should the system actively alert you rather than handling the case silently?

Sub-topics to explore:

- Trigger conditions for alerts: confidence drop, shift in mapping quality, drift, or other signals.
- Pipeline behaviour on alert: blocking and waiting for the reviewer versus continuing and flagging for later; dependence on data type or downstream consumer.
- Acceptability of provisional mappings: cases where the mapping flows downstream marked unconfirmed until a human validates it.
- Behaviour in the absence of a reviewer: proceeding automatically with previously approved high-confidence mappings while queueing uncertain ones, and whether this should be opt-in per dataset.

Topic 5 — Corrections, Learning, and Governance (~10 min)

Purpose: surface views on how reviewer corrections should feed back into the model, how downstream errors should be rolled back, what audit trail is expected, and what authorisation is required to promote mappings into production.

Opening prompt. When you correct a mapping, what should happen to that correction afterwards?

Sub-topics to explore:

- Automatic versus reviewer-gated retraining; whether corrections should be batched and approved before they reach the training pipeline.
- Rollback granularity when an approved mapping turns out to be wrong: full rewind versus restoration to the last valid state while preserving correct intermediate work.
- Scope of the audit trail (who approved what, when, with what confidence), its retention period, and access controls.
- The sign-off conditions for promoting mappings to the ML training pipeline: what evidence the responsible person needs to see, and whether a second senior approver is expected.

Topic 6 — Discussion Vignettes (select 1–2 based on role and time)

Purpose: ground the abstract responses above in concrete operational cases. Each vignette is used as a discussion opener; the bullet points below are threads to draw out rather than scripted questions.

Vignette A — First-time dataset onboarding. *A new PV park sends a dataset with 180 columns using Greek abbreviations. The system produces: 145 high-confidence (green), 20 medium-confidence (amber), 15 unmapped (red).*

Threads to explore: where the reviewer starts; treatment of high-confidence mappings (accept-as-is or spot-check); what information makes an amber decision possible (top-k candidates with similarity and target-field definitions); handling of the unmapped columns; the steps between session completion and data flowing downstream.

Vignette B — Drift after months of autonomous operation. *The same PV park has been sending weekly data for 6 months, processed automatically. One week the system raises an alert: 5 columns have shifted (a unit change from kW to MW, a temperature sensor switching from Celsius to Fahrenheit).*

Threads to explore: the urgency of the response and whether the pipeline should stop immediately; what should happen to the current week's data while the alert is open; the follow-up once drift is confirmed (profile update, new profile version, reprocessing of affected batches).

Vignette C — Rollback after a downstream failure. *A mapping profile was approved last month and used for 4 weekly batches. The ML model trained on this data is performing poorly. The cause is traced to* `wind_direction`*, which was mapped to the wrong standard field — a subtle error the confidence score did not catch.*

Threads to explore: the corrective action (field-level correction, profile rollback, reprocessing of affected batches); what information would have helped catch the error during the original review; the notification scope when a rollback affects data already consumed downstream.

Wrap-up (~5 min)

Purpose: surface anything the structured topics did not reach, and identify the single feature the participant considers most important.

Prompts:

- Anything about supervising an automated data standardisation system that we have not discussed and that you consider important.
- If the participant could design the supervision interface themselves, the single most important thing it should have.

Role Focus Notes

When interview time was constrained, topics were prioritised by participant role as shown in Table A.1. Within each prioritised topic the interviewer followed the conversation naturally rather than asking each sub-topic in turn.

Role	Priority topics
PPC Data Engineer	Topics 1, 2, 3; Vignette A or B
PPC Data Governance Lead	Topics 1, 2, 5; Vignette C
Central Data Analyst / ML practitioner	Topics 2, 4, 5; Vignette A or C

Table A.1: Role-based topic prioritisation applied when interview time was constrained.

A.6 Closing the Interview

At the end of each session I thank the participant, reiterate the confidentiality and data-usage commitments, and confirm that they will receive the transcript for member checking. Participants are offered a summary of findings once thematic analysis is complete.

A.7 Post-Interview Procedure

1. Transcribe audio verbatim (auto-transcription with manual review against the recording).
2. Send the transcript to the participant for member checking, with a five working-day window for corrections or redactions.
3. Code responses against the requirement categories used in this thesis: routing and confidence, batch workflow, correction and retraining, profile governance, explainability, monitoring and alerting, and audit and accountability.
4. Maintain chain of evidence: each requirement claim is traceable to a verbatim quotation or attributed paraphrase in the transcript.
5. Cross-reference findings with the supervision requirements in Section 4.4 (SR-F-01 to SR-F-10, SR-F-12 to SR-F-14, SR-NF-01 to SR-NF-07) to identify confirmations, enrichments, and new requirements absent from the UC2 documentation. SR-F-11 (pre-validation of input data) was excluded as a pipeline prerequisite rather than a supervision requirement (see Section 4.2).

Appendix B

Design: Supplementary Material

B.1 Component API Specification

Table B.1 lists all twenty-one endpoints in the supervision architecture. Eighteen are implemented and constitute the evaluation contract against which Chapter 7 assesses the design objectives. Three are integration-contract endpoints for SR-F-05 and SR-F-13 (correction export and algorithm steering hints): the endpoint shapes are implemented and reachable, but consuming them productively depends on AS-side work outside this prototype, so they are not evaluated in Chapter 7.

B.2 Session Data Model

Tables B.2 and B.3 list the full column definitions for the `sessions` and `session_items` tables. Design rationale for the schema choices is discussed in Section 5.5.4.

Method	Path	Role	Key behaviour
Job ingestion and routing			
POST	`/jobs`	open	Routes items via routing engine; creates session and drift check; returns `session_id`, item counts, and `drift_alert_id` (non-null if DriftAlert created); payload requires `target_standard` (e.g. `SAREF`, `CIM`, `IEC61850`) identifying the target data standard for the job; accepts optional `model_version` field (snapshotted on session for SR-F-04 correction tuples)
GET	`/routing/config`	open	Returns current threshold values: `routing_threshold_auto`, `oversight_min_sample_rate`, `oversight_min_sample_count`
Session management			
GET	`/sessions`	open	List sessions; filterable by `status` and `database_id`
GET	`/sessions/{id}`	open	Session detail and ordered item queue (Tier $<$ 3, `ORDER BY tier ASC, confidence ASC`); includes per-tier progress counts
PATCH	`/sessions/{id}/items/{item_id}`	open	Submit item decision; updates `decision` and `reviewed_at`; returns updated session progress
POST	`/sessions/{id}/open`	open	PENDING $\rightarrow$ ACTIVE; reviewer-initiated lifecycle entry (Section 5.5.2)
POST	`/sessions/{id}/pause`	open	ACTIVE $\rightarrow$ PAUSED; all decisions persisted
POST	`/sessions/{id}/resume`	open	PAUSED $\rightarrow$ ACTIVE; session state unchanged
POST	`/sessions/{id}/submit`	open	ACTIVE $\rightarrow$ SUBMITTED if submission guard passes; HTTP 422 with blocking item IDs if any SAMPLED item has decision = NULL or SKIPPED
POST	`/sessions/{id}/archive`	open*	SUBMITTED $\rightarrow$ ARCHIVED; system-triggered after retention threshold (Section 5.5.2), not reviewer-initiated. * Open in the prototype for manual evaluation; a production deployment would restrict this to a retention-scheduler service token without changing the route shape
Profile governance			
POST	`/profiles`	DE	Create DRAFT profile from a SUBMITTED session; HTTP 409 if a DRAFT or PROPOSED profile already exists for the database
GET	`/profiles`	open	List profiles; filterable by `database_id` and `status`

Column	Type	Notes
`id`	UUID PK	
`job_id`	TEXT	Identifier of the upstream standardisation job
`database_id`	TEXT	Source database identifier; used for provisional carry-forward
`target_standard`	TEXT	Target data standard for this job (e.g. `SAREF`, `CIM`, `IEC61850`); all candidates in the job payload belong to this standard
`status`	ENUM	PENDING, ACTIVE, PAUSED, SUBMITTED, ARCHIVED
`reviewer_id`	TEXT	Plain string in prototype; auth layer is future work
`created_at`	TIMESTAMP	
`opened_at`	TIMESTAMP	Set on first PENDING → ACTIVE transition
`submitted_at`	TIMESTAMP	Set on ACTIVE → SUBMITTED transition
`routing_threshold_auto`	FLOAT	Snapshot of `routing_threshold_auto` at session creation
`oversight_min_sample_rate`	FLOAT	Snapshot of `oversight_min_sample_rate` at session creation
`oversight_min_sample_count`	INT	Snapshot of `oversight_min_sample_count` at session creation
`model_version`	TEXT	Nullable; AS model version from `JobInput`; used when assembling correction tuples for SR-F-05

Table B.2: Full column definitions for the `sessions` table.

Column	Type	Notes
`id`	UUID PK	
`session_id`	UUID FK	→ `sessions.id`
`field_id`	TEXT	Source field identifier from AS
`source_field`	TEXT	Human-readable field name
`routing`	ENUM	FLAGGED, SAMPLED, AUTO_ACCEPTED; Tier 0 carry-forward items inherit the original item's routing value — `tier=0` and `carried_from` are the carry-forward markers
`confidence`	FLOAT	Algorithm's confidence score for the primary candidate
`data_type`	TEXT	Nullable; inferred source-database type (CopycHats cue 2 — e.g. `float64`, `string`)
`instance_values`	JSON	Nullable; representative sample values from the source field (CopycHats cue 3)
`candidates`	JSON	Sorted descending by confidence; `candidates[0]` is the primary candidate; list guaranteed by the anti-corruption layer at ingestion. Each entry: `{standard, field_name, definition, hierarchy_position, sibling_concepts, confidence, rationale?}`
`tier`	INT	0 = PROVISIONAL carry-forward, 1 = FLAGGED, 2 = SAMPLED, 3 = AUTO_ACCEPTED
`decision`	ENUM	NULL, APPROVED, CORRECTED, PROVISIONAL, SKIPPED
`corrected_to`	TEXT	Populated when decision = CORRECTED
`hint_type`	TEXT	Nullable; reviewer-supplied ontology hint type (SR-F-13, Section 5.7)
`hint_parameters`	JSON	Nullable; structured hint payload for AS re-ranking on next job run
`hint_applied`	BOOL	False by default; set to True via `POST /hints/{id}/applied` when the AS consumes the hint
`carried_from`	UUID FK	→ `session_items.id`; set when item is a provisional carry-forward
`carried_from_session_id`	UUID	Optional back-reference to the originating session for audit traceability
`reviewed_at`	TIMESTAMP	Set when reviewer submits item decision

Table B.3: Full column definitions for the `session_items` table.

Appendix C

Evaluation: Supplementary Material

This appendix reproduces the full semi-structured protocol used to conduct the practitioner validation interviews referenced in Chapter 7. These interviews form Track B of the evaluation: a qualitative practitioner check on whether the implemented supervision architecture addresses the requirements identified in Chapter 4 as practitioners understand them, complementing Track A's technical evaluation against the mock Auto-Standardiser API.

C.1 Protocol Metadata

Thesis title	Integrating Human Oversight in Automated Data Standardisation: the NOUS Project use case
Method	Semi-structured qualitative validation interview with hands-on prototype walkthrough
Interviewer	Sérgio Peixoto (MSc candidate, FEUP)
Date	[insert date]
Interviewee	[insert name and role]
Setting	[insert setting; requires screen sharing with mouse-and-keyboard control for the participant, or in-person session with shared screen]
Duration	75 minutes
Prototype access	Live deployed instance; participant operates the interface with the interviewer co-piloting; no pre-interview access

C.2 Purpose of the Interviews and Study Context

The interviews support Research Question 3 of this dissertation: the extent to which the implemented supervision architecture addresses the requirements identified in RQ1, as assessed through technical evaluation against a mock Auto-Standardiser API and practitioner validation with domain experts. Track A (covered separately in Chapter 7) evaluates the implemented capabilities against Design Objective satisfaction using the mock AS API. Track B, the strand this protocol covers, captures evidence that the mock-API evaluation cannot:

1. Whether the implemented capabilities address the supervision requirements identified in Chapter 4 as practitioners with relevant domain experience understand them, surfacing divergences between specified intent and realised behaviour.
2. Workflow friction visible only when a practitioner operates the prototype end-to-end through a representative scenario, rather than via isolated API contract tests.
3. Whether the architecture-specified-but-not-implemented capabilities (Tier 3 in Chapter 5: SR-F-12 tiered escalation and SR-F-13 algorithm steering) credibly capture practitioner intent, since both originated in the requirements interviews and the architecture spec is a synthesis of practitioner contributions that has not been independently validated.

C.3 Interviewee and Interviewer Information

Interviewee Selection and Background

The validation interviews target practitioners with relevant domain experience in automated data standardisation, annotation workflows, or human oversight in machine-learning pipelines. The primary recruitment target is practitioners with no involvement in the design of the architecture being validated: an independent reviewer is the cleanest control against the designer-confirmation and participant-investment biases discussed below, and is treated here as the default rather than a contingency. Returning participants from the requirements interviews in Chapter 4 (Participant 1, NOUS project engineer with direct familiarity with the AS platform; Participant 2, researcher with expertise in annotation workflow design and multi-fidelity human–AI systems) may be included as a secondary cohort, with their responses read against the heightened participant-investment bias their prior contribution carries (Section 7.7). The protocol is valid regardless of whether a participant contributed to the requirements interviews: the supervision requirements being validated were not articulated by any single participant, but synthesised from the T7.5/UC2 deliverable, two requirements interviews, and the PRISMA literature corpus. A uniform *pre-demo expectation capture*, in which the participant articulates before each scenario demo what a supervision system handling the scenario should provide, anchors the post-demo calibration for every participant regardless of prior involvement.

The protocol is sized to the cohort that recruitment actually yields. With independent reviewers, or with more than two participants, the full guide applies and the numeric calibration and capability-versus-interface scores are reported as measures. If the cohort reduces to one or two returning participants, those scores are still recorded during sessions but are reported as indicative signals rather than aggregated as measurements, since the sample cannot support that precision; the validation then leans on the qualitative devices that retain their weight at that size: show-then-ask sequencing, the required disconfirming prompts, member checking, and disconfirming-first reporting. At session start the following items are captured for each participant: role and focus area; years of relevant professional or research experience; and (for any participant new to the dissertation) typical responsibilities pertinent to data standardisation, annotation, or human oversight workflows.

Interviewer and Reflection on Biases

The interviewer is Sérgio Peixoto, MSc candidate at FEUP and author of this dissertation. Three structural biases were identified before data collection and explicitly accounted for in the conduct of each session. The first two carry forward from the requirements interviews in Chapter 4 but apply with greater force at validation time, because a concrete artifact now exists to evaluate; the third is new to the validation stage.

Designer-confirmation bias. I designed the supervision architecture this dissertation specifies and built the prototype the participant is being asked to react to. At the requirements stage this bias was conceptual; at the validation stage it is direct, because every response is an evaluation of an artifact I produced. To mitigate this bias the protocol embeds three procedural rules: a *show-then-ask* sequencing rule that prohibits explaining design intent before getting the participant's reaction (Section C.5); a list of *pre-committed disconfirming prompts* that I am required to ask at specific points in each vignette, regardless of whether the participant has volunteered criticism; and a post-interview commitment to tag and prominently report disconfirming evidence during transcript coding (Section C.7).

Participant-investment bias. Where a participant contributed to the requirements catalogue the prototype implements, they hold a partial co-authorship relationship to the design they are now being asked to validate, which creates social pressure to endorse what they see. The primary control is again the recruitment priority above: an independent reviewer did not articulate the requirements being validated, so the investment relationship is absent. For any returning C1 participant in the secondary cohort, two structural devices back this up: a *pre-demo expectation capture* asked before each demo block, in which the participant articulates what a supervision system handling the scenario should provide *before* seeing the implementation, so that endorsement of what is shown can be compared against their own pre-demo position; and a *numeric calibration score* at vignette boundaries that forces a commitment harder to inflate than open verbal approval and surfaces gaps between high verbal endorsement and lower numeric agreement, read as an indicative signal at the small samples this validation expects (see the sizing rule above).

Project-insider bias. As at the requirements stage, when a participant is a colleague within the NOUS project who shares my institutional context, the shared background reduces friction in domain-specific discussion but also encourages shortcuts, such as unspoken assumptions and in-group framings, that can obscure positions a non-insider would have probed further. The mitigation is again the recruitment priority above: an external reviewer does not share this institutional context.

C.4 Ethical Considerations and Consent

Both interviews follow the ethical guidelines below, which are communicated to each participant before the session begins.

- **Voluntary participation.** Participation is voluntary; participants may decline any question, decline any prototype interaction, or withdraw at any time without consequence.

- **Confidentiality and anonymisation.** Role-only anonymisation is offered as the default, with stronger anonymisation available on request. For returning C1 participants, anonymisation preferences from the requirements interviews are carried forward unless the participant requests otherwise; for participants new to the dissertation, anonymisation preference is captured at session start.
- **Recording.** Audio is recorded for transcription accuracy, and the prototype session is screen-recorded so that participant actions can be re-examined in conjunction with their spoken reactions. Both recordings are deleted after the thesis is graded.
- **Prototype data.** The deployed prototype operates against a mock Auto-Standardiser; nothing the participant submits affects any production system, and no real personal data is processed during the session. The prototype state may be reset between sessions.
- **Right to review.** Each participant receives the transcript for member checking, with a five working-day window to request corrections or redactions before any quoted material is incorporated into the dissertation. Member checking covers both verbal responses and any quoted descriptions of participant actions during the hands-on segments.
- **Data usage.** The data are used exclusively for this Master's dissertation.
- **Time commitment.** The interview is expected to last approximately 75 minutes.

C.5 Interview Guide

The guide structures a 75-minute session across six topic blocks. The substantive blocks combine scenario-driven validation (Topic 2) with a complementary requirement sweep (Topic 3) and a focused architecture-spec block (Topic 4); this hybrid is intended to surface friction that a pure scenario walk-through would hide while still preserving the naturalistic signal that an exhaustive requirement-by-requirement walk-through would destroy. Three phrasing rules apply throughout, baked into the protocol as procedural commitments rather than recommended technique:

- **Show-then-ask.** Demonstrate the surface or open the screen before stating any design intent; the participant reacts first, the interviewer explains only if asked.
- **Disconfirming prompts are required.** At the specified points in each block at least two of the disconfirming prompts listed below must be asked, regardless of whether the participant has already volunteered criticism.
- **Silence after answers.** Pause after a response; the interviewer does not fill silence with reassurance or rephrasing, since participants frequently qualify or extend a position in the second sentence if not interrupted.

Topic 1 — Orientation (~5 min)

Purpose: brief the participant on the prototype scope and on the Tier 1/2/3 framing, and confirm the session format.

Opening setup. Confirm consent and recording. Summarise that the prototype implements a defined subset of the supervision requirements identified during requirements analysis (Tier 1 and Tier 2 partial), and that two of the architecture-specified-only requirements (SR-F-12 and SR-F-13) will be discussed without implementation. Confirm that the participant will operate the prototype and that the interviewer will only intervene when navigation help is needed.

Topic 2 — Scenario Vignettes (~35 min)

Purpose: surface workflow fit, requirement realisation, and friction by replaying two scenarios from the requirements interview vignettes using the deployed prototype.

Two vignettes are used: *Vignette A — First-Time Dataset Onboarding* (covering the session cluster: SR-F-01/02, SR-NF-03/06, SR-F-14, SR-F-03 display, SR-F-04 schema, and triggering profile promotion via SR-F-09) and *Vignette C — Rollback After a Downstream Failure* (covering the governance cluster: SR-F-06/07/08 and SR-NF-05). Each vignette follows a fixed five-step structure.

Per-vignette structure.

1. **Pre-demo expectation capture.** Before opening the prototype, ask: *"Before we look at the prototype, take a moment and describe what a supervision system handling this scenario should provide. What is the key thing you would want it to get right?"* Record the unprompted response verbatim; do not correct, prompt, or comment. This pre-demo statement serves as the comparison anchor for the numeric calibration that follows the demo.

2. **Vignette restatement.** Restate the vignette context as it appeared in the requirements interview, to refresh the scenario without leading.

3. **Hands-on walkthrough.** Open the prototype to the relevant entry point and invite the participant to drive: *"Walk me through how you would handle this scenario, and as you go, please verbalise what you're seeing and what you'd try next."* The interviewer remains silent except for navigation help.

4. **Required disconfirming prompts.** At a natural pause mid-walkthrough and again at completion, the interviewer asks at least two of: *"What frustrates you here?", "What in this would make you abandon it in production?", "What does this not let you do that you'd want to do?", "Where did you expect something different?"*

5. **Numeric calibration and cap-vs-UI probe.** Ask: *"On a 1–5 scale, how well does what you just used match what you described before we opened the prototype?"* followed by *"What would have made it a 5?"* If the score is below 5, follow with the disambiguating probe

"Of what kept it below 5, how much was the system not doing what you wanted versus the interface not letting you do it easily?" Record the score, the gap statement verbatim, and the participant's allocation between capability and interface concerns. The probe is the live mechanism for separating requirement-fit evidence from UI-usability evidence within the numeric calibration; the post-interview coding (see Post-Interview Procedure) applies the same separation to all other findings.

Topic 3 — Requirement Sweep (~10 min)

Purpose: catch the Tier 1 and Tier 2-partial requirements that the vignettes did not exercise. Coverage map between vignettes and requirements is determined per session before the interview, against the prototype state at that point.

For each requirement not exercised by Vignette A or C, the interviewer briefly demonstrates the surface or the configuration point that realises it and asks one targeted question: *"Does this match what you needed when you described [requirement topic]?"* followed by one required disconfirming prompt from the list above. Sweep coverage typically includes the workload controls in the review interface (SR-NF-03 pause-and-resume, progress indicators), the configurable oversight floor and SAMPLED-item handling (SR-NF-06), and the dual-control governance approval flow (SR-NF-05) if it was not exercised during vignette completion.

Topic 4 — Architecture-Spec Validation (~10 min)

Purpose: validate that the two architecture-specified-only requirements whose design synthesises practitioner input (SR-F-12 tiered escalation and SR-F-13 algorithm steering) credibly capture what the participants described in the requirements interviews. This block is the ex-ante design-validation strand of the evaluation approach (Section 7.1): it assesses two specified but unimplemented designs, not running behaviour.

For each of the two SRs the interviewer shares a single slide containing three items: a one-line description of what the architecture specifies; the data shape (the threshold structure for SR-F-12; the hint contract JSON keys for SR-F-13); and a one-line statement of what is not implemented and why. The accompanying question is targeted to the synthesis being validated:

- **SR-F-12 — Tiered escalation.** *"The architecture combines a sequential reviewer ladder raised during requirements elicitation with a pooled-reviewer model from the literature into a single three-tier routing engine. Does this capture how you would expect tiered escalation to work, or would you have it work differently?"*
- **SR-F-13 — Algorithm steering.** *"Take the Greek-abbreviation case from requirements elicitation, where reviewers flag untranslated source-column names so the standardiser can retry with a translation hint. Would this hint contract let the AS do what is needed? Is anything missing from the data shape?"*

At the end of the block, ask one numeric calibration question covering both SRs: *"On a 1–5 scale, how credibly does the architecture spec for these two capture what you'd want a real implementation to do?"* followed by *"What would push it to a 5?"*

SR-F-05 (standard-partitioned retraining), SR-F-10 / SR-NF-04 (audit trail and GDPR compliance), and SR-NF-02 (post-submission accuracy verification) are not included in this block: they are validated analytically or by regulatory grounding rather than by practitioner reaction, and are recorded as such in the traceability map (Section 7.2). SR-F-05 still receives a brief check during the SR sweep that the correction-tuple fields a retraining consumer would need are present in the captured schema.

Topic 5 — Surprises and Absences (~5 min)

Purpose: solicit findings the structured blocks did not surface, with explicit framing toward the disconfirming side.

Prompts:

- Anything in the prototype you did not expect to be there, and would not want?
- Anything missing that you reached for during the session, or that you would reach for in production but did not see?
- Anything about how a supervision system for this kind of workflow should work that you would state differently now that you have used the prototype?

Topic 6 — Wrap-up (~5 min)

Purpose: surface anything the structured topics did not reach, and identify the single most important improvement.

Prompts:

- Anything about supervising an automated data standardisation system, or about the prototype, that we have not discussed and that you consider important.
- If you could change one thing about what you used today, what would it be?

C.6 Closing the Interview

At the end of each session I thank the participant, reiterate the confidentiality and data-usage commitments, confirm that they will receive the transcript and the screen recording for member checking, and remind them of the five working-day window for corrections or redactions. Participants are offered a summary of findings once thematic analysis is complete.

C.7 Post-Interview Procedure

1. Transcribe audio verbatim (auto-transcription with manual review against the recording). Annotate the transcript with timestamps from the screen recording at points where a participant action triggered a verbal reaction, so that the reaction can be re-read in conjunction with what was on screen at the time.

2. Send the transcript and the annotated screen recording to the participant for member checking, with a five working-day window for corrections or redactions.

3. Code responses against three coding schemes in parallel. The first is the requirement coding scheme from the requirements interviews (routing and confidence, batch workflow, correction and retraining, profile governance, explainability, monitoring and alerting, audit and accountability), used to map findings back to specific SRs. The second is an evaluation outcome coding scheme: `ENDORSE` for statements confirming the implementation matches the participant's intent; `DIVERGE` for statements identifying a substantive difference between intent and realised behaviour; `DISCONFIRM` for statements critical of, surprised by, or contradicting the implementation; `ABSENT` for capabilities the participant reached for but did not find. The third disambiguates the locus of the finding: `[CAP]` where the implemented behaviour does not match the requirement as the practitioner understands it; `[UI]` where the capability is present but the interface obscures or impedes its use; `[BOTH]` where the participant's reaction blends the two and cannot be cleanly separated. The vignette-end cap-vs-UI probe supplies the participant's own allocation for the numeric calibration; the same axis is applied during coding to all `DISCONFIRM`, `DIVERGE`, and `ABSENT` findings. Capability-fit findings constitute the primary Track B evidence for RQ3, since the validation claim is requirement satisfaction; usability findings are reported separately as a secondary contribution and explicitly acknowledged as outside the requirement-validation claim, since interface usability beyond SR-NF-07 accessibility (documented as a limitation) is not part of the supervision-requirement catalogue.

4. Record per-vignette and per-block numeric calibration scores alongside the verbatim gap statements (*"what would have made it a 5?"*) and, for the vignette calibrations, the participant's cap-vs-UI allocation. Discrepancies between high verbal endorsement and low numeric scores are flagged for prominent reporting; cap-vs-UI allocations where the participant attributes the gap predominantly to interface friction are kept separate from those attributing it to capability fit, since the two affect the requirement-satisfaction conclusion differently.

5. Report disconfirming and divergent findings before endorsing findings in the Chapter 7 write-up. Counts of `DISCONFIRM` and `DIVERGE` quotes per SR are reported alongside the endorsements, and at least one representative quote from each category is included where present, so that the evaluation cannot collapse into selective citation of confirmatory responses.

6. Cross-reference Track B findings with Track A's Design Objective satisfaction analysis to identify cases where a technical pass on Track A coincides with a practitioner divergence on Track B; these are reported as the most informative findings of the evaluation, since they identify capabilities that work as specified but not as intended.