

AI-Enabling Requirements: An Industrial Experience Report from a Data-Driven Recruitment System

Sylwia Kopczyńska 

Poznan University of Technology, Poznań, Poland

Mirosław Ochodek 

Poznan University of Technology, Poznań, Poland

Eryk Kosmala 

Poznan University of Technology, Poznań, Poland

Jędrzej Musiał 

Poznan University of Technology, Poznań, Poland

Rafał Mroziewski 

MONDI sp. z o.o., Poznań, Poland

Dominik Nitychoruk 

MONDI sp. z o.o., Poznań, Poland

Abstract

Requirements Engineering commonly distinguishes between functional and non-functional requirements. In AI-based systems, an orthogonal perspective might also be worth considering: whether requirements concern AI components or not.

This paper reports on a 2.5-year R&D project building MONEKTO, a holistic data-driven recruitment system for a Polish temporary employment agency. The system integrates three data streams: recruitment process events, candidate and job order profiles, and employment histories – to support both candidate-order matching and employee retention prediction.

Drawing on project logs covering June 2023 to January 2026, we identified three categories of requirements in AI-based systems: Non-AI-based requirements, AI-based requirements, and AI-enabling requirements.

We show how these categories differ in their lifecycle: while traditional requirements stabilize after implementation, AI-enabling requirements emerge incrementally and require continuous investment throughout the system lifetime. We report five observations on the emergent nature of AI-enabling requirements and derive seven practitioner lessons for RE in AI-based system projects.

2012 ACM Subject Classification Software and its engineering → Requirements analysis; Computing methodologies → Machine learning

Keywords and phrases requirements engineering, AI-based systems, AI-enabling requirements, industry case study, data-driven recruitment

Digital Object Identifier 10.4230/LIPIcs.ESEM.2026.92

Category Software Engineering in Practice Track Paper

Funding This work was supported by the European Funds for a Modern Economy 2021-2027 (FENG) under project no. FENG.01.01-IP.02-0857/23: “MONEKTO – System wspomagający proces rekrutacji kandydatów oraz monitorowania pracy pracowników wykorzystujący sztuczną inteligencję”.



© Sylwia Kopczyńska, Mirosław Ochodek, Eryk Kosmala, Jędrzej Musiał, Rafał Mroziewski, and Dominik Nitychoruk;

licensed under Creative Commons License CC-BY 4.0

20th International Symposium on Empirical Software Engineering and Measurement (ESEM 2026).

Editors: Robert Feldt, Maria Paasivaara, Daniel Mendez, Stefan Wagner, and Marvin Muñoz Barón;

Article No. 92; pp. 92:1–92:13



Leibniz International Proceedings in Informatics

LIPIcs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

The interest in building AI-based software systems has grown substantially over recent years. However, as practitioners frequently discover, embedding artificial intelligence (AI)/machine learning (ML) components into real software systems is significantly more complex than it initially appears. A well-documented challenge is *hidden technical debt* [20]: the real code in an ML system is often the smallest part of the effort; data pipelines, serving infrastructure, monitoring, and feature engineering dwarf the model code itself.

Requirements Engineering (RE) for AI-based systems has similarly attracted increasing attention [4, 7, 21]. Much of this work focuses on quality attributes specific to AI – fairness, transparency, explainability, robustness – and how to specify them [17]. Less attention has been paid to a more fundamental challenge: the category of requirements that must be implemented not to provide user-valued features, but simply to *make the AI work at all*. We call these **AI-enabling requirements**.

Examples include: CV parsing pipelines, expert labeling tools, feedback mechanisms for model correction, and explainability components mandated by regulation. None of these deliver features that a user would request, yet without them the AI components cannot function.

The concept of AI-enabling requirements emerged from our experience building MON-EKTO, a data-driven recruitment system for a Polish temporary employment agency, over a 2.5-year industrial research and development (R&D) project. MONEKTO integrates three complementary data streams – recruitment process events, candidate and job-order profiles, and employment histories – supporting two AI components: a candidate-order recommender and a turnover predictor.

As the project progressed, we noticed that a recurring class of requirements resisted the usual RE categories. This prompted our research question: *What categories of requirements emerge when building an AI-based industrial system, and how does their lifecycle differ from traditional software requirements?*

To answer this question, this paper makes two contributions:

1. A **taxonomy** of requirements for AI-based systems with three categories: Non-AI-based requirements, AI-based requirements, and AI-enabling requirements (Section 4). The subjects of AI-enabling requirements are individually studied elsewhere in the literature and many are commonly known; our contribution is identifying them as a requirements category and characterizing their lifecycle, which differs fundamentally from both Non-AI-based and AI-based requirements;
2. **Empirically grounded observations and lessons learned** characterizing the lifecycle of AI-enabling requirements and their implications for project planning (Sections 5–6).

Our taxonomy offers practitioners a language to name and track AI-enabling requirements explicitly, rather than letting them surface as unplanned technical debt. Evidence from the MONEKTO project suggests that most of these requirements cannot be identified upfront, with direct consequences for effort estimation and project planning in AI-based system development.

2 Related Work

Organizational AI Readiness. Research on AI readiness identifies organizational prerequisites for AI adoption [1, 2, 9, 16], spanning technology infrastructure, data quality, regulatory compliance, and organizational culture. A common finding is that readiness is a *precondition for adoption* – yet this literature operates at the strategic level and does not address how readiness factors translate into concrete software requirements in a specific project.

Requirements Engineering for AI-Based Systems. The RE community has recognized distinct challenges for AI-based systems: difficulty specifying model quality upfront, a mismatch between iterative AI development and traditional RE processes [7, 21], and the need for ML specification frameworks [4, 17]. These studies focus on what AI systems *must do*; our work shifts attention to what must be *built around* them to function at all.

ML Systems as Surrounding Infrastructure. Sculley et al. [20] show model code is a small fraction of total ML system effort, with data pipelines, serving, and monitoring the primary source of hidden technical debt. Sambasivan et al. [19] find data work systematically undervalued, and Nahar et al. [15] trace ML failures to gaps at the model-system interface. MLOps frameworks [12, 13] translate these concerns into architectural patterns, but none treats the surrounding infrastructure as a category of software requirements with a distinct lifecycle.

The literature thus addresses AI enablers at the *organizational* (readiness) and *architectural* (MLOps) levels but not at the *requirements* level – what a team must specify and maintain to make AI work in a specific system. This gap is what our work addresses. We do not claim that the artifacts themselves – data pipelines, annotation tools, monitoring, human oversight, or compliance infrastructure – are newly identified; each is well documented in the MLOps and RE literature cited above. Our contribution is to frame them as a coherent category of software requirements with a distinct lifecycle.

3 Research Approach

3.1 Study Design

This industrial experience report is structured as a single-site, longitudinal, practitioner-embedded, participant-observer case study [18, 22], spanning 2.5 years of development (June 2023–January 2026). Our observations draw on: (i) contemporaneous project logs – monthly team meeting reports produced throughout the project lifetime; (ii) requirements and architecture artifacts generated at four project milestones; and (iii) milestone validation reports documenting quantitative system outcomes.

The underlying register was reconstructed retrospectively from these contemporaneous sources rather than tracked prospectively. The taxonomy itself was derived inductively from this retrospective analysis: requirements were iteratively classified based on their relationship to AI components, with categories crystallizing from recurring patterns in that classification. We counted as a single requirement each separately specified and implemented register item linked to one of the system’s two core AI-based use cases – e.g., for requirements regarding ML explainability in these use cases, SHAP computation and its UI presentation are counted as two items. We exhaustively cataloged requirements linked to the two AI use cases (8 AI-based, 28 AI-enabling); non-AI-based requirements, numbering in the hundreds, were not exhaustively cataloged and are not the subject of quantitative claims in this paper. A requirement was classified as *planned* if present in the original project scope (grant application or initial requirements elicitation) and as *emergent* if its first documented mention in the monthly logs appears after that point together with an identifiable trigger (a data-quality, model-accuracy, regulatory, or operational problem); emergent requirements were dated by first log mention, with the month estimated from project-phase context for 8 of the 18 items. One co-author first reviewed the register independently and proposed an initial taxonomy structure; the resulting categories and the classification of individual requirements were then

discussed jointly among the co-authors until consensus was reached. Single-site case studies are an established practice in software engineering research [3, 20], with their value lying in depth rather than generalizability.

3.2 Industrial Context

MONDI (<https://mondi.pl>) is a Polish temporary employment agency that recruits blue-collar workers (manufacturing, logistics, skilled trades) for German industrial clients. The agency’s core business process spans candidate matching, worker deployment to German employers, and ongoing employment monitoring. A critical business challenge is *qualitative turnover*: the rate at which recruited workers terminate their employment prematurely for reasons potentially attributable to a poor recruitment decision (e.g., skills mismatch, language barriers, location issues).

Before the project, the company managed its recruitment-to-employment workflow across multiple disconnected systems and tools. The MONEKTO development project (06. 2023–01. 2026) aimed to design a holistic recruitment and workforce monitoring process and to build the system supporting it. MONEKTO covers the full workflow – candidate registration, job order management, contract administration, and mobile communication with deployed workers – and integrates an AI subsystem (MONDAI) providing data-driven candidate-order matching and retention risk assessment.

The system integrates three complementary data streams: **S1 – Recruitment Process Events** (discrete hiring-process events, e.g., CV submission, language test result, modeled via BPMN); **S2 – Candidate and Job Order Profiles** (structured features extracted from unstructured CVs and job descriptions via NLP and domain ontology matching); and **S3 – Employment History** (time-varying payroll features: absences, disciplinary warnings, overtime, and contract milestones).

These three streams feed two AI components: the **Recommender**, which ranks candidates for a given job order, and the **Turnover Predictor**, which estimates the probability of qualitative turnover. Critically, the system is not limited to the recruitment decision point: Stream S3 continuously monitors active contracts, creating a *feedback loop* that links post-deployment outcomes back to the original candidate-order matching. This enables the system to be retrained on outcome-labeled data, improving recommendation quality over time as real employment results accumulate.

The system architecture separates: MONEKTO (handling business workflows) and MONDAI (the AI subsystem), communicating via REST API with asynchronous event-driven feature recalculation. The architecture was designed using template-based NFR elicitation [11] (38 requirements per ISO/IEC 25010:2023 [8]), 20 documented architectural decisions, and an ATAM workshop [10] generating 57 evaluation scenarios.

4 A Taxonomy of Requirements in AI-Based Systems

We observed three qualitatively distinct requirement categories in our AI-based system. Type A and Type B are mutually exclusive; Type C is not and can co-occur with either.

Type A: Non-AI-based Requirements (Non-AI)

Non-AI-based requirements are simply conventional software requirements – functional and non-functional alike – whose specification and realization do not involve AI components. We use the explicit “Non-AI” label to keep the three categories terminologically parallel and to

emphasize that the taxonomy is orthogonal to the classical FR/NFR distinction: each of the three types can contain both. They are frequently elicited through standard techniques (interviews, use case modeling, user stories). In MONEKTO, examples include: “*A recruiter can view a report on the number of candidates for each job order.*” or “*A recruiter can comment on a candidate’s profile.*”

Their lifecycle is well understood: they are elicited, specified, implemented, accepted, and then maintained with minor corrections.

Type B: AI-based Requirements (AI-based)

AI-based requirements describe user-visible behaviors, constraints and non-functional requirements which are based on AI components. In MONEKTO, examples include: (1) “*The system recommends candidates for a given job order*”; (2) “*The system displays a turnover risk indicator for each active employment contract*”; (3) “*The system extracts structured profile data from a candidate’s uploaded CV document*” (a user-valued feature also labeled Type C1 – see the classification rule below).

Type C: AI-enabling Requirements (AI-enabling)

AI-enabling requirements are a documented prerequisite for a Type B requirement to function, regardless of whether they also qualify as Type A or Type B. When a requirement serves only this role, with no independent business justification, it is typically *not requested by users*: they might not know it is needed, and its absence is not observable as a defect from their point of view. Such purely-enabling requirements are identified by the project team – some anticipated early, during initial NFR analysis, but most discovered during development, typically when encountering data quality problems, model accuracy shortfalls, regulatory obligations, or operational issues with deployed models.

Table 1 summarizes eight AI-enabling sub-categories identified in MONEKTO. We do not claim that this list is exhaustive across all AI-based systems.

Classification rule

We classify a requirement along two independent dimensions. The first asks whether it has genuine value on its own: would a user or the business still want it if every AI component in the system worked perfectly? If yes, it is labeled Type A (realized without AI) or Type B (realized using AI); the two are mutually exclusive. If it fails this test (if it exists only so that some AI component can work, improve, or stay compliant), it earns no A/B label at all.

The second dimension asks whether the requirement is a documented prerequisite for a Type B requirement’s AI component to function correctly. If yes, it is also Type C, independently of the answer to the first question. A requirement can therefore be Type A and Type C at once (CRUD(L) operations for candidates and job orders – needed for ordinary business processing, but also feeding the recommender’s feature vector), Type B and Type C at once (CV parsing – wanted by recruiters to complement candidate profile data regardless of the recommender, but also feeding it), or purely Type C with no A/B label (SHAP explanations – needed only to satisfy a legal obligation). Labels can also change over time – a purely-C requirement can later gain a B label once it acquires independent value, without losing its C label. This is also why the taxonomy is orthogonal to the FR/NFR distinction: the A/B label depends on why a requirement exists, not on whether it is functional; the C label depends on a separate question – what else depends on it.

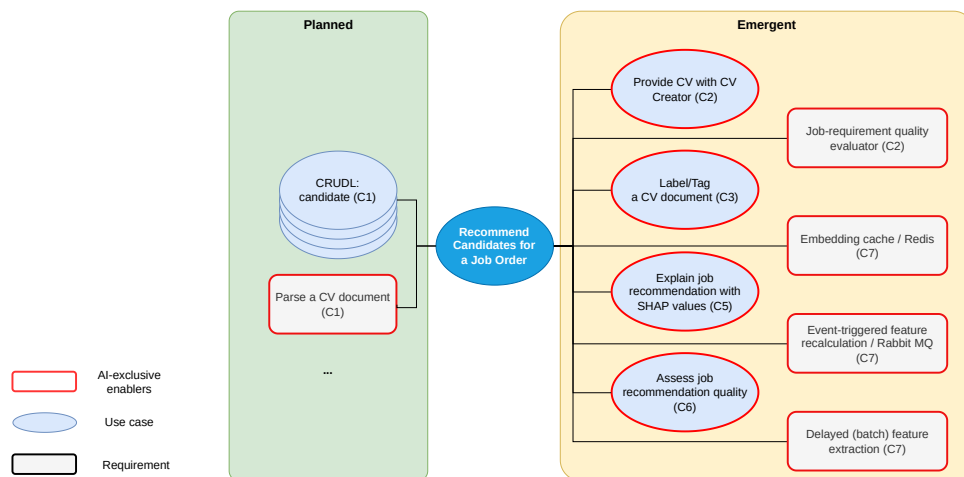
■ **Table 1** Sub-categories of AI-enabling requirements (Type C) with MONEKTO examples.

Sub-category	Purpose	MONEKTO examples
C1 Data Collection	Acquire and structure data for training & inference	CV parsing (PDF/DOCX); NER extraction of skills/ work history; OCR of warning documents, payroll record import
C2 Data Quality	Ensure input data meets quality standards for reliable inference	LLM-based job description assistant; structured CV creator; data input completeness validation
C3 Model Evaluation	Assess model quality during development	Expert annotation tool (Dataset Tagger) for labeling of candidate-order pairs for use as ground truth; validation dataset protocols
C4 Business Outcome Measurement	Link AI decisions to observable business outcomes serving as the signal of whether the system delivers its intended value	Recommendation feedback by recruiters; qualitative turnover rate computation
C5 Explainability	Make model outputs interpretable to decision-makers	Displaying SHAP values per recommendation; explanations in recruiter UI; full candidate list policy
C6 Human-in-the-Loop	Enable human oversight and correction of model behavior	Expert labeling workflow; recruiter override; periodic model review
C7 Data Freshness	Ensure features reflect current state at inference time	Event-triggered recalculation (RabbitMQ); embedding cache (Redis)
C8 Privacy & Security	Protect personal data; secure AI-specific interfaces	CV anonymization post-NER; GDPR-compliant training data; MONDAI network isolation

5 AI-Enabling Requirements: Key Observations

► **Observation 1** (Most, but not all, AI-enabling requirements owe their existence to AI). When examining the AI-enabling requirements in MONEKTO, we observe two distinct classes. *Shared foundations* have business value independently of AI but also provide essential AI input: for example, CRUD(L) operations for candidates, job orders, and job applications are needed to run recruitment and to provide structured data for the recommender. *AI-exclusive enablers* exist only because an AI-based requirement exists: a job-description quality evaluator, CV labeling tool, recommendation feedback endpoint, SHAP explanations, or feature cache would not appear in a non-AI project scope. Shared foundations tend to be identified during ordinary business analysis; AI-exclusive enablers emerge later as the team encounters data, accuracy, regulatory, or operational constraints. Figure 1 illustrates examples of both classes for the *Recommend Candidates for a Job Order* use case.

► **Observation 2** (The deceptive stability of AI-based requirements). At the use-case level, AI-based requirements appeared stable throughout the project: *Recommend Candidates for a Job Order* and *Predict Turnover Risk* could have been written similarly on the first and last day. This stability was deceptive. Beneath each use case, AI-enabling requirements kept growing. For the Recommender, after the initial learn-to-rank architecture [5] and Precision@5 = 61% by May 2024, later sprints added requirements-matching features,



■ **Figure 1** An example of AI-enablers associated with the AI-based use case *Recommend Candidates for a Job Order*.

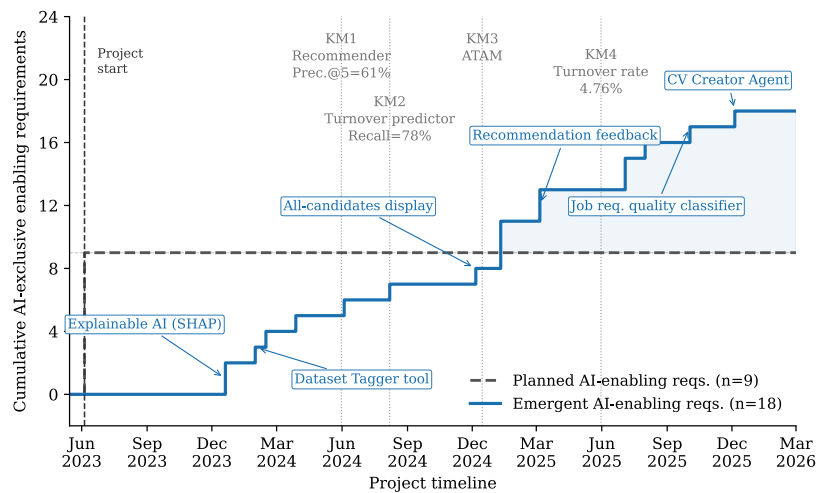
expert correction rules, LLM-based semantic matching, two-phase filtering, and retraining. For the Turnover Predictor, the validated August 2024 model was followed by feature-vector extensions, sentiment analysis of coordinator notes, and real-time updates. AI-based requirements are therefore like the tip of an iceberg: the visible specification remains stable while the enabling infrastructure grows.

► **Observation 3** (Most AI-enabling requirements cannot be fully identified upfront). AI-exclusive enabling requirements are particularly difficult to anticipate at project inception. While some can be foreseen – a data ingestion pipeline, for instance, is a predictable need whenever historical records must feed a model – most emerge only when the team encounters concrete problems during development or after deployment.

Five MONEKTO examples illustrate this. The job description quality generator (C2) was added only after early recommendation failures were traced to vague, incomplete job orders that made feature extraction unreliable. Candidate data quality was weaker than assumed: among 169,293 unique candidates, only 22,630 (13%) initially had CV documents; integration work raised coverage to 92,068 (54%), motivating the CV Creator (C2). The embedding cache (C7) became necessary once production measurements showed that recomputing semantic similarity embeddings per request exceeded acceptable response-time thresholds. A recruiter-facing feedback mechanism (C4) followed once operational experience showed that offline metrics such as Precision@5, adequate during development, could not monitor recommendation quality in production; embedding a lightweight agree/disagree signal in the recruiter tool created a feedback stream neither the project scope nor any user interview had anticipated. Finally, building a reliable training dataset required far more effort than expected: historical records yielded only 1,369 ground-truth candidate-order pairings, a fraction of the roughly 100,000 pairings in the validation set (100 orders $\times$ 1,000 candidates); the remainder came from structured expert labeling sessions using a dedicated annotation tool (C3, MONDAI Dataset Tagger) – not foreseen in the project scope.

These requirements were triggered by operational evidence rather than user interviews, echoing hidden technical debt dynamics [20].

Figure 2 shows the cumulative growth of AI-exclusive enabling requirements tracked in our project logs for the two core AI-based use cases, excluding shared foundations such as CRUDL operations. The planned requirements (dashed line) were established at project



■ **Figure 2** Cumulative growth of planned vs. emergent AI-exclusive enabling requirements in the MONEKTO project (Jun 2023–Jan 2026). Vertical dotted lines mark project milestones; the dashed horizontal line marks the planned ceiling of 9 requirements.

start; emergent requirements accumulated progressively, with notable clusters around the first model validation (Dec 2023), the architecture phase (mid-2024), EU AI Act compliance work (late 2024–early 2025), and operational monitoring needs in the post-deployment period. Of the 27 requirements captured in the figure, only 9 were anticipated in the original project scope; the remaining 18 emerged during development in response to concrete problems with data quality, model accuracy, regulatory obligations, or operational constraints. We treat this fraction as an illustrative order of magnitude for a single project rather than a statistically precise estimate; it is sensitive to classification choices and the completeness of the documented set. This ratio likely underestimates the true proportion of emergent requirements: the documented set is not exhaustive, and several planned requirements were substantially expanded in scope during implementation.

► **Observation 4** (The AI Act as a Source of AI-Enabling Requirements). MONEKTO falls within the scope of the EU AI Act [6] as a high-risk AI system¹ used in employment decision-making. This regulatory context generated the following cluster of AI-enabling requirements that were not driven by user needs but by legal obligations.

Transparency (C5). The obligation to provide meaningful explanations drove the implementation of SHAP values [14] and their display in the recruiter UI – not a user-requested feature but a compliance necessity. **Human oversight (C6):** The non-automation requirement drove the policy of displaying all candidates rather than a pre-filtered shortlist, ensuring the recruiter retains full decision authority. **Auditability (C3):** The obligation to log AI-assisted decisions drove the implementation of audit logging.

These requirements emerged as the team’s understanding of the regulatory landscape deepened. Their implementation consumed significant development effort and influenced architectural decisions (e.g., the audit logging infrastructure, the SHAP computation pipeline).

¹ Annex III, point 4: AI systems used for recruitment, selection, or evaluation of candidates in employment contexts.

► **Observation 5** (The AI-Enabler-to-Feature Trajectory). An interesting pattern observed in our project is that some AI-enabling requirements eventually *became* user-valued features in their own right. We term this the *AI-enabler-to-feature trajectory*.

The CV extraction pipeline (C1) began as an internal mechanism for parsing PDF/DOCX files into structured fields. It evolved into the user-facing *CV Creator*, letting candidates enter structured data directly or upload a CV file that is parsed by an LLM – adding direct user value while improving training data quality, since structured inputs reduce noise in the feature vector.

Similarly, job-requirement quality tooling (C2) began as a model assessing requirement importance and verifiability, in direct response to low-quality, unverifiable requirements entered by sales staff that introduced noise into the recommender’s feature vector. It evolved into a more ambitious capability: an LLM-based generator producing structured job requirements from a brief client description – addressing data quality at the source rather than downstream, and improving business data quality system-wide.

In both cases, an AI enabler improved training data quality and later added business value, suggesting such requirements should be treated as design artifacts, not only cost items.

6 Guidelines and Lessons Learned

LL 1. Plan for AI-enabling requirements as a permanent budget item. AI-based requirements are never “done” in the traditional sense: each improvement to an AI model – to correct errors, add features, or adapt to data drift – generates new AI-enabling requirements. Project plans should include a standing allocation for this work rather than treating it as exceptional, releasing a minimum viable version alongside the AI feature and planning for incremental elaboration throughout the project lifetime. *Source: Observations 2, 3.*

LL 2. Data quality problems emerge late and are expensive. In AI-based systems, data quality requirements cannot be fully identified before the team has hands-on experience with actual data. Low-quality inputs are frequently discovered to be the primary bottleneck for model accuracy only after early validation attempts fail. Practitioners should build in explicit data quality discovery phases and budget for data improvement tooling as a likely emergent need rather than an optional refinement. *Source: Observations 2, 3.*

LL 3. Regulatory compliance is a source of AI-enabling requirements. Regulatory frameworks such as the EU AI Act can generate a substantial cluster of AI-enabling requirements – covering transparency, human oversight, and auditability – that are entirely invisible during standard user requirements elicitation. Organizations building AI systems for regulated domains should analyze applicable regulations during initial requirements elicitation and treat compliance as a requirement driver alongside user needs. *Source: Observation 4.*

LL 4. The AI-enabler-to-feature trajectory is a design opportunity. AI-enabling requirements can evolve into user-facing features over time. Designing these components with eventual user exposure in mind – clean APIs, extensible data models, usable interfaces – can yield business value while reducing future rework. This trajectory also improves training data quality, creating a virtuous cycle between infrastructure investment and model performance. *Source: Observation 5.*

LL 5. Human-in-the-loop mechanisms are architectural requirements. Standard offline metrics such as Precision@5 are insufficient as the sole quality signal in a deployed AI system. As Observation 3 illustrates, detecting quality degradation in a live recommender required embedding a feedback mechanism directly in the recruiter’s tool; building a reliable training

set required sustained expert annotation effort. Both are forms of human-in-the-loop design that were not initially planned and had to be introduced reactively. Treating human feedback and correction mechanisms as architectural requirements from the outset – rather than as late additions – reduces rework and improves data quality throughout the system lifecycle. *Source: Observations 3, 4.*

LL 6. User involvement is an AI-enabling requirement, not a soft process concern. Building and validating AI-based systems requires sustained, structured user involvement that extends well beyond tool usage. As Observation 3 illustrates, accurate training data for the recommender required recruitment experts to produce tens of thousands of labeled pairings in organized annotation sessions. User feedback loops, annotation tasks, and expert review workflows are AI-enabling requirements in their own right: they must be planned, toolled, and resourced explicitly. *Source: Observation 3.*

LL 7. Define acceptance criteria at three levels: business, AI-based, and AI-enabling. Acceptance criteria for AI-based systems must be defined at three distinct levels. At the *AI-enabling level*, criteria should specify data quality benchmarks, evaluation protocols, and compliance checklists. At the *AI-based level*, they should cover model performance thresholds alongside functional correctness. At the *business level*, the metric may be structurally decoupled from AI-component metrics – a business KPI such as turnover reduction may require additional AI-enabling requirements (e.g., monitoring infrastructure) before it can even be evaluated. Aligning these levels early reveals hidden enablers that would otherwise remain hidden until late in the project. *Source: Observations 1, 2, 3.*

A natural follow-up question is whether AI-enabling requirements call for different specification practices. We did not find this to be the case: individual Type C requirements in MONEKTO were documented with the same techniques as their Non-AI and AI-based counterparts – use cases and user stories where the requirement was functional in nature, template-based specification [11] where it was not. The practical difference lies not in how these requirements are written, but in how they are identified and managed over the project lifecycle: recognizing a requirement as AI-enabling is what triggers the planning, budgeting, and acceptance-criteria practices captured in the lessons learned above, rather than letting it surface unplanned. This suggests a concrete direction for future work: a closer, NLP-supported analysis of project artifacts – specifications, backlog items, and meeting logs – to identify recurring keywords and linguistic patterns that could support earlier elicitation and identification of AI-enabling requirements in future projects, before they resurface as unplanned technical debt.

7 Threats to Validity

Internal validity. The taxonomy emerged from a single project and classification involved judgment. The research team was also the development team, which may introduce confirmation bias – a recognized threat in practitioner-embedded case studies [18]. We mitigated this through co-authorship spanning academic researchers and industry practitioners, contemporaneous monthly logs, and consensus discussion, with any disagreements resolved until consensus was reached.

Construct validity. The concept of “AI-enabling requirements” is our own construct. It may overlap with, or be subsumed by, existing concepts such as “data requirements” [21] or operational ML requirements [3]. We position our contribution as a practitioner-oriented synthesis that makes explicit a category of work that is often implicitly understood but rarely

named or systematically managed. The distinguishing contribution is not the identification of these artifacts per se, but the characterization of their lifecycle as structurally different from both Non-AI-based and AI-based requirements (a more specific discussion in Section 2).

External validity. This is a single-company, single-domain case study; the recruitment domain's high data volume, regulatory context, and unstructured inputs may amplify AI-enabling requirements relative to other domains. One alternative interpretation is that emergence simply reflects the uncertainty inherent in any R&D project – yet within the same project, Non-AI-based requirements followed the expected stable lifecycle while the emergent pattern was specific to the AI-enabling layer. This differential behavior suggests the dynamics are structural rather than merely context-specific, though replication in non-R&D settings is needed to confirm this.

We expect the taxonomy's three categories and Observations 1–3 (shared vs. exclusive enablers, deceptive stability, incremental emergence) to generalize beyond recruitment: they describe how ML components behave once embedded in a production system, a structural property rather than a domain-specific one. Observation 4 should likewise hold for any system classified as high-risk under the EU AI Act [6], a category spanning employment, education, credit, and law enforcement (Annex III). One element is more clearly shaped by our specific project: the 67% emergent ratio, which we offer as a reference point for future replications rather than a general benchmark. Observation 5 (the enabler-to-feature trajectory) is grounded in two documented instances in MONEKTO; we see it as a promising pattern worth testing in other projects.

Conclusion validity. The planned/emergent ratio is based on a tracked subset of requirements and is sensitive to classification choices, including granularity – what counts as one requirement (Section 3). It should be read as an illustrative order of magnitude rather than a statistical estimate. Research in other projects and domains is needed to assess the generalizability of this ratio.

8 Conclusions

This paper reported on 2.5 years of building MONEKTO, a data-driven recruitment system, and proposed a taxonomy distinguishing three requirement types in AI-based systems: Non-AI-based, AI-based, and AI-enabling. Drawing on project artifacts, we derived five observations and seven practitioner lessons on the lifecycle of AI-enabling requirements.

Our main finding is that AI-based requirements have a different lifecycle from Non-AI ones: the visible AI feature may remain stable, but the enabling requirements around it emerge and grow in response to data quality, accuracy, regulatory, and operational challenges.

In the MONEKTO project, approximately two thirds of the tracked AI-exclusive enabling requirements were emergent rather than planned, with notable clusters appearing around model validation, regulatory compliance work, and post-deployment operational needs. This ratio has direct implications for project planning: AI-based system projects that budget only for the visible AI features and neglect the enabling infrastructure will systematically underestimate cost and schedule.

A further observation is that AI-enabling requirements are not purely cost items: some evolve into user-facing features, creating a virtuous cycle between infrastructure investment and model performance. Naming and tracking this category, rather than letting it surface as unplanned debt, is the taxonomy's practical contribution.

Future work includes operationalizing the taxonomy, replicating it elsewhere, and building tool support for early requirement identification.

References

- 1 Wajid Ali and Abdul Zahid Khan. Factors influencing readiness for artificial intelligence: A systematic literature review. *Data Science and Management*, 8, 2024.
- 2 Sulaiman Alsheibani, Yen Cheung, and Chris Messom. Artificial intelligence adoption: AI-readiness at firm-level. In *Proceedings of the 22nd Pacific Asia Conference on Information Systems (PACIS)*, 2018.
- 3 Saleema Amershi, Andrew Begel, Christian Bird, Robert DeLine, Harald Gall, Ece Kamar, Nachiappan Nagappan, Besmira Nushi, and Thomas Zimmermann. Software engineering for machine learning: A case study. In *Proc. of the 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pages 291–300. IEEE, 2019. doi:10.1109/ICSE-SEIP.2019.00042.
- 4 Hrvoje Belani, Marin Vukovic, and Željka Car. Requirements engineering challenges in building ai-based complex systems. In *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*, pages 252–255, 2019.
- 5 Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: From pairwise approach to listwise approach. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, pages 129–136, 2007. doi:10.1145/1273496.1273513.
- 6 European Parliament and Council of the European Union. Regulation (EU) 2024/1689 laying down harmonised rules on artificial intelligence (artificial intelligence act). Official Journal of the European Union, 2024.
- 7 Hans-Martin Heyn, Eric Knauss, Amna Pir Muhammad, Olof Eriksson, Jennifer Linder, Padmini Subbiah, Shameer Kumar Pradhan, and Sagar Tungal. Requirement engineering challenges for ai-intense systems development. *2021 IEEE/ACM 1st Workshop on AI Engineering - Software Engineering for AI (WAIN)*, pages 89–96, 2021.
- 8 ISO/IEC. ISO/IEC 25010:2023 – systems and software engineering – systems and software quality requirements and evaluation (SQuARE) – product quality model. International Organization for Standardization, 2023.
- 9 Jan Jöhnk, Malte Weißert, and Katrin Wyrski. Ready or not, AI comes — an interview study of organizational AI readiness factors. *Business & Information Systems Engineering*, 63(1):5–20, 2021. doi:10.1007/S12599-020-00676-7.
- 10 Rick Kazman, Mark Klein, Mario Barbacci, Tom Longstaff, Howard Lipson, and Jeromy Carriere. The architecture tradeoff analysis method. In *Proc. of the Fourth IEEE Internat. Conference on Engineering of Complex Computer Systems*, pages 68–78. IEEE, 1998.
- 11 Sylwia Kopczyńska, Jerzy Nawrocki, and Mirosław Ochodek. An empirical study on catalog of non-functional requirement templates: Usefulness and maintenance issues. *Information and Software Technology*, 103:75–91, 2018. doi:10.1016/J.INFSOF.2018.06.009.
- 12 Dominik Kreuzberger, Niklas Kühl, and Sebastian Hirschl. Machine learning operations (MLOps): Overview, definition, and architecture. *IEEE Access*, 11:31866–31879, 2023. doi:10.1109/ACCESS.2023.3262138.
- 13 Jay Lee, Hanqi Su, Dai-Yan Ji, and Takanobu Minami. Engineering artificial intelligence: Framework, challenges, and future direction. arXiv preprint arXiv:2504.02269, 2025.
- 14 Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Neural Information Processing Systems*, volume 30, 2017.
- 15 Nadia Nahar, Shurui Zhou, Grace Lewis, and Christian Kästner. Collaboration challenges in building ML-enabled systems: Communication, documentation, engineering, and process. In *Proc. of the 44th Intl Conference on Software Engineering (ICSE)*. ACM, 2022. doi:10.1145/3510003.3510209.

- 16 Luisa Pumplun, Christoph Tauchert, and Margareta Heidt. A new organizational chassis for artificial intelligence: Exploring organizational readiness factors. In *Proceedings of the 27th European Conference on Information Systems (ECIS)*, 2019.
- 17 Mona Rahimi and Jane Cleland-Huang. Toward requirements specification for machine-learned components. In *Proceedings of the 27th International Requirements Engineering Conference Workshops (REW)*, pages 241–244. IEEE, 2019. doi:10.1109/REW.2019.00049.
- 18 Per Runeson and Martin Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14(2):131–164, 2009. doi:10.1007/S10664-008-9102-8.
- 19 Nithya Sambasivan, Shivani Kapania, Hannah Highfill, Diana Akrong, Praveen Paritosh, and Lora M. Aroyo. "everyone wants to do the model work, not the data work": Data cascades in high-stakes AI. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–15. ACM, 2021. doi:10.1145/3411764.3445518.
- 20 D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-François Crespo, and Dan Dennison. Hidden technical debt in machine learning systems. In *Advances in Neural Information Processing Systems*, 2015.
- 21 Andreas Vogelsang and Markus Borg. Requirements engineering for machine learning: Perspectives from data scientists. In *Proc. of the 6th International Workshop on Artificial Intelligence for Requirements Engineering (AIRE)*, pages 1–8. IEEE, 2019.
- 22 Robert K. Yin. *Case Study Research and Applications: Design and Methods*. SAGE Publications, 6th edition, 2018.