Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

From Enterprise Institutional Documents to Editable Process Models: A Hybrid LLM-Rule-Engine Architecture for Enterprise Governance

Jia Tian

tianjia3@cnooc.com.cn

China University of Petroleum, Beijing https://orcid.org/0009-0009-4299-6516

Qingyang Li

China National Offshore Oil Corporation (China)

Tingrui Jiang

China National Offshore Oil Corporation (China)

Zhaonian Liu

CNOOC Research Institute Co., Ltd.

Xiaoyong Gao

China University of Petroleum, Beijing

Man Jiang

CNOOC Research Institute Co., Ltd.

Research Article

Keywords: Business process model, Large language model, Rule engine, Process automation, Enterprise governance, Design science research

Posted Date: November 1st, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11531565/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

Abstract

Enterprise institutional documents encapsulate substantial process-oriented management knowledge. Transforming them into structured editable process models constitutes a foundational task for business process management (BPM) and governance-risk-compliance (GRC) digitalization. However, manual process modeling is time-consuming, inconsistent across practitioners, and difficult to maintain, while existing automated approaches—whether rule-based methods grounded in traditional natural language processing (NLP) or recent large language model (LLM) generation methods—cannot simultaneously satisfy the dual requirements of semantic understanding accuracy and output format conformance, and typically lack deep integration with enterprise approval systems. Following the design-science-research (DSR) paradigm, this paper proposes, implements, and deploys a hybrid LLM and rule-engine architecture (LRHA) for automatically generating organization-compliant editable process models from enterprise institutional documents. The architecture decomposes the transformation task into six stages—text parsing, element extraction, logic modeling, rule matching, JSON generation, and draft output—with the LLM handling the first three stages that rely on semantic understanding and the rule engine handling the latter three stages that rely on deterministic constraints, thereby separating semantic interpretation from format enforcement. For Chinese enterprise institutional texts, we derive a set of "textual cue to process element" mapping rules and define a JointJS-oriented process model representation specification together with ten executable quality validation rules. Distinct from generic text-to-BPMN methods, LRHA provides native support for enterprise governance capabilities at the architectural level, including institutional-text ambiguity resolution, compliance constraint validation, asynchronous integration with existing approval OA systems, version traceability, and audit trail preservation. The system has been deployed in production within the risk control information system of a large energy enterprise group, achieving end-to-end automated integration with the group's OA system through asynchronous task interfaces (task submission, status query, multi-file download, and proactive callback), underpinned by HMAC signature verification and idempotency mechanisms. Case evaluation shows that for complex processes with over 40 nodes and edges, manual modeling work is reduced from several days to approximately 20 minutes, requiring only minor manual fine-tuning. Large-scale quantitative benchmark evaluation remains as future work. This paper makes four contributions: (1) it proposes and validates the division of labour and collaboration mechanism between LLMs and rule engines in automated process model generation; (2) it constructs a mapping rule system and executable quality validation specification tailored to Chinese enterprise institutional texts; (3) it natively implements enterprise-governance-oriented architectural capabilities for institutional-text ambiguity resolution, compliance constraint validation, OA-system integration, version traceability, and audit-trail preservation; and (4) it reports the complete deployment pathway and engineering experience of the hybrid architecture within an enterprise information system, offering a replicable reference paradigm for similar organizations.

1. Introduction

Business processes constitute the core carrier of enterprise operations, and process models—as formal representations of business processes—play an indispensable role in process analysis, optimization, execution, and compliance control. In enterprise governance practice, enterprise institutional documents represent the primary source of process knowledge, specifying in natural language responsibility division, approval workflows and management steps of process-oriented requirements. Aligning organizational processes with regulatory provisions is essential for regulatory implementation and constitutes a core foundation of GRC system construction [1].

However, this transformation has long been performed manually. Business personnel must read the full text of a regulation, comprehend its process logic, and draw flowcharts by hand, a workflow beset by three salient problems. First, it is inefficient: modeling a medium-length regulation typically requires several hours from reading to completion, and complex regulations take considerably longer. Second, it is inconsistent: different modelers produce flowcharts that vary in swimlane partitioning, shape formatting, and edge routing, lacking unified normative constraints and impeding standardized management. Third, it is difficult to maintain: after regulation revisions, corresponding flowcharts often lag behind due to insufficient manpower, resulting in text–diagram inconsistencies that undermine regulation enforcement and communication.

Automated process model generation has been studied extensively. Early work relied primarily on traditional NLP techniques, extracting activities and their control-flow relations from text through syntactic parsing and dependency analysis; representative contributions include the process model generation method of Friedrich et al. [2], the process discovery method of Epure et al. [3], and the formal-reasoning-based approach of Sánchez-Ferreres et al. [4]. While these methods achieve reasonable performance on constrained corpora, they exhibit limited robustness to complex natural language expressions—such as implicit temporal ordering, nested conditions, and non-local coreference—and are difficult to generalize to real-world enterprise institutional texts. In recent years, LLMs have substantially advanced process modeling through their semantic understanding and generation capabilities. Researchers have explored LLM applications in process model generation [5], process management tasks [6], and have systematically analyzed the associated opportunities and challenges [7]. Nevertheless, pure LLM methods suffer from two salient issues: hallucination risk, whereby the model may generate nodes, edges, or identifiers inconsistent with the source text [8, 9]; and unstable output formatting, whereby the JSON or graphical descriptions directly generated by LLMs often fail to conform to an organization's established modeling conventions (e.g., node numbering rules, swimlane layout, style parameters), necessitating substantial manual correction.

More critically, existing text-to-BPMN methods generally focus on the single step of "generating graphics from text," while overlooking the deeper requirements that enterprise governance scenarios impose on process models. In enterprise regulatory governance, a process model is not merely a graphical representation but a carrier of compliance control, approval integration, version management, and audit traceability. Specifically, enterprise-grade automated process model generation must satisfy capabilities that generic text-to-image methods lack: (1) **institutional-text ambiguity resolution**—enterprise

institutional texts often contain vague expressions, conflicting clauses, and overlapping responsibilities that must be identified and resolved during modeling; (2) **compliance constraint validation**—process models must conform to organizational internal control norms and regulatory requirements, necessitating the codification of compliance rules as executable constraints; (3) **approval OA integration**—generated process models must be directly embeddable into existing enterprise approval systems to close the loop from regulation to execution; (4) **version traceability**—after regulation revisions, process differences must be automatically identified and change descriptions generated; and (5) **audit trail preservation**—the modeling process must be traceable and auditable to satisfy enterprise internal control and regulatory requirements.

To address these challenges, this paper follows the design science research (DSR) paradigm [21, 22] and proposes, implements, and deploys a hybrid LLM and rule engine architecture (LRHA). The core idea is to delineate the division of labour between humans and machines based on task nature: the LLM handles the first three stages that depend on semantic understanding—text parsing, element extraction, and logic modeling—while the rule engine handles the latter three stages that depend on deterministic constraints—rule matching, JSON generation, and draft output. This design constitutes a domain-specific engineering instantiation of the neuro-symbolic paradigm [10, 11] for enterprise-institutional-text-to-process-model generation. It combines LLM semantic strengths with deterministic rule-system reasoning to guarantee accurate process-logic extraction together with format-compliant outputs. The system is deployed in a large energy enterprise group, achieving end-to-end automation from enterprise institutional documents to editable process models in JointJS format, and has been integrated with the group's OA risk control information system through asynchronous task interfaces.

The main contributions of this paper are as follows:

- **C1 (Architectural contribution)**: We propose and validate the division of labour and collaboration mechanism between LLMs and rule engines in automated process model generation, providing a complete design of the six-stage pipeline and a reusable paradigm for hybrid "semantic understanding + deterministic constraint" architectures in document-to-model transformation tasks. The design separates the semantic intermediate representation (IR) from the graphical presentation, thereby isolating the uncertainty of LLM output at the semantic layer.
- **C2 (Methodological contribution)**: We derive a set of "textual cue to process element" mapping rules tailored to Chinese enterprise institutional texts, covering swimlane identification, node identification, decision branch identification, condition labels, and subprocess references; we also define a JointJS-oriented process model representation specification and ten executable quality validation rules that codify organizational modeling standards as machine-checkable constraints.
- **C3 (Governance capability contribution)**: We provide native support for enterprise governance capabilities at the architectural level—including institutional-text ambiguity resolution, compliance constraint validation, approval OA integration, version traceability, and audit trail preservation—distinguishing LRHA from generic text-to-BPMN methods that focus solely on graphical generation.

- **C4 (Engineering contribution)**: We report the complete deployment pathway of the hybrid architecture within an enterprise information system, including asynchronous task interface design (submission, query, multi-file download, and proactive callback), HMAC signature verification and idempotency mechanisms, v1-to-v2 multi-file protocol evolution, and efficiency data and deployment experience from a real production environment.

The remainder of this paper is organized as follows. Section 2 reviews related work and presents a differentiated comparison with text-to-BPMN methods. Section 3 provides the research background and problem definition. Section 4 elaborates the system architecture and methodology. Section 5 describes system implementation and OA integration. Section 6 reports case evaluation results. Section 7 discusses the findings. Section 8 concludes the paper and outlines future work.

2. Related Work

2.1. Process Model Generation from Natural Language Text

Automated process model generation from text has been a long-standing research problem in the BPM community. Friedrich et al. [2] were among the first to systematically address this problem, proposing a method based on the Stanford parser to extract process models from use-case descriptions, identifying activities, resources, and ordering relations through part-of-speech tagging, dependency parsing, and semantic role labeling. Epure et al. [3] proposed an automatic process discovery method for methodological texts, employing shallow syntactic patterns to identify process elements. Sánchez-Ferreres et al. [4] proposed a framework for formal reasoning on natural language descriptions of processes, combining text annotation with formal verification to elicit a formal representation of textual process descriptions. Riehle et al. [12] developed the Text2BPMN tool to generate BPMN models from natural language text. In addition, Leopold et al. [13] investigated bidirectional transformation between process models and natural language for process model validation. Although these methods perform well on controlled corpora, they generally rely on language-specific syntactic features tailored for English and therefore exhibit limited adaptability to the characteristics of Chinese enterprise institutional texts—such as omitted subjects, nested long sentences, and clause-based expressions—and have difficulty integrating process logic distributed across different chapters.

2.2. Large Language Models in Business Process Management

With the rapid advancement of LLM capabilities, their application in BPM has attracted increasing research attention. Vidgof et al. [7] systematically reviewed the opportunities and challenges of LLMs in BPM, covering process modeling, process mining, and process execution. Kourani et al. [5] evaluated the performance of LLMs in process modeling tasks, finding that while LLMs can generate syntactically plausible process models, they remain deficient in semantic correctness and specification conformance.

Grohs et al. [6] demonstrated experimentally that LLMs can accomplish a variety of BPM tasks, but output quality is significantly influenced by prompt design. Berti et al. [14] explored abstractions and prompt definitions for process mining with LLMs. Overall, existing research indicates that LLMs are effective at understanding process semantics, yet their outputs fall short of enterprise-grade modeling standards in terms of format conformance, numbering consistency, and layout rationality, and are subject to hallucination risk [8, 9], necessitating additional constraint mechanisms for correction.

2.3. Neuro-Symbolic Hybrid Architectures

Neuro-symbolic AI, which combines neural networks with symbolic systems, has been identified as a major direction of third-generation AI [10, 11]. In engineering practice, hybrid architectures have been applied to structured output generation and knowledge graph construction. For instance, Geng et al. [15] proposed a grammar-constrained decoding method that constrains LLMs to generate syntactically valid structured outputs without fine-tuning. Within the process modeling domain, such hybrid designs follow a typical division of labour: LLMs undertake semantic understanding, while rule systems enforce format constraints. The LRHA architecture proposed in this paper is a concrete instantiation of this paradigm for the enterprise-institutional-text-to-process-model transformation task. Its distinctive feature lies in explicitly encoding an organization's established modeling conventions—including node numbering, swimlane layout, style parameters, and quality validation rules—as deterministic constraints within the rule engine. This design enables fully controllable output formatting while preserving the flexibility of LLM-driven semantic extraction.

2.4. Process Model Quality and Compliance

Process model quality is a classic topic in BPM research. Mendling et al. [16] proposed seven process modeling guidelines (7PMG) covering model size, structure, and labeling; Becker et al. [17] proposed guidelines of business process modeling (GoB). In the compliance domain, El Kharbili et al. [18] surveyed the state of the art in business process compliance checking. The quality validation mechanism in this paper draws on these frameworks, transforming them into executable checks for the JointJS representation, covering structural completeness, numbering consistency, layout conformance, and semantic completeness.

2.5. Differentiated Comparison with Existing Text-to-BPMN Work

Synthesizing the above literature, existing research exhibits the following gaps: (1) traditional NLP methods lack robustness to real-world enterprise institutional texts, particularly systematic studies targeting Chinese enterprise institutional texts; (2) pure LLM methods produce unstable output formats and high hallucination risk, making it difficult to directly satisfy enterprise-grade modeling conventions; (3) there is a lack of complete deployment cases spanning from algorithm design to enterprise information system integration (including interface protocols and security mechanisms); and (4) more critically, existing text-to-BPMN methods generally focus on the single step of "generating graphics from text," lacking systematic support for the deeper requirements of enterprise governance scenarios

(ambiguity resolution, compliance validation, OA integration, version traceability, and audit trail preservation).

To clearly present the differentiation between this work and existing approaches, Table 1 compares LRHA with representative text-to-BPMN methods across multiple dimensions.

Table 1
Differentiated comparison between LRHA and representative text-to-BPMN methods

Capability dimension	**Traditional NLP methods [2, 3, 12]**	**Pure LLM methods [5, 6]**	**LRHA (this work)**
Semantic understanding	Weak (relies on syntactic rules)	Strong	Strong (handled by LLM)
Output format conformance	Medium	Weak (format drift)	Strong (guaranteed by rule engine)
Institutional-text ambiguity resolution	Not supported	Limited (not designed for enterprise institutional compliance scenarios)	Natively supported (mapping rules + expert feedback)
Compliance constraint validation	Not supported	Limited (not designed for enterprise institutional compliance scenarios)	Natively supported (ten quality checks + compliance rules)
Approval OA integration	Not supported	Not supported	Natively supported (asynchronous task interfaces)
Version traceability	Not supported	Not supported	Natively supported (version metadata + difference identification)
Audit trail preservation	Not supported	Not supported	Natively supported (task logs + idempotency + signature verification)
Enterprise-grade deployment	None	None	Deployed in production (real system integration)

Table 1 shows that the differentiation of LRHA lies not merely in its ability to generate flowcharts, but in its comprehensive support for enterprise governance scenarios. Through the hybrid LLM and rule engine architecture, LRHA ensures semantic understanding and format conformance while providing native support for institutional-text ambiguity resolution, compliance constraint validation, approval OA integration, version traceability, and audit trail preservation—capabilities that generic text-to-image models do not possess.

3. Research Background and Problem Definition

3.1. Application Scenario: Enterprise institutional documents and Process Management

The regulatory system of a large energy enterprise group spans 17 business domains, including engineering technology, supply chain, and science and technology management, with a large and frequently revised corpus of enterprise institutional documents. Under the group's requirement for integrated regulation–process management, every process-oriented regulation must be accompanied by a corresponding flowchart for internal control management, risk prevention, and process execution. Enterprise institutional documents typically contain the following process-oriented content:

- **Division of responsibilities**: specifying the roles and responsibilities of functional departments and subordinate units in the process (e.g., "Department X is responsible for…" and "by Unit Y…");
- **Approval procedures**: specifying control steps such as review, approval, and filing (e.g., "report to the group's executive in charge for approval" and "after review by X…");
- **Management steps**: specifying the sequence and conditions of business activities (e.g., "first… second…" and "if… then… otherwise…").

Traditionally, flowcharts are drawn manually by business personnel through a workflow of reading the regulation, comprehending the logic, drawing the graphics, and manually proofreading. As noted above, this model suffers from three major problems: low efficiency, inconsistent standards, and difficult maintenance.

3.2. Problem Formalization

We formalize the problem of automated process model generation from enterprise institutional text as follows. Given an enterprise institutional document D (structured text comprising chapters and clauses), the objective is to generate a process model M such that:

1. **Semantic fidelity**: the activities, control flow, and decision conditions in M are consistent with the process logic described in D;
2. **Specification conformance**: M satisfies the organization's established modeling specification $\mathcal{S}$ (node numbering rules, graphical styles, swimlane layout, naming rules, etc.);
3. **Machine readability**: M is represented in a structured format (JointJS JSON in this paper) that can be directly imported into the target information system (the OA risk control system) for editing and use;
4. **Governance traceability**: the generation process of M is traceable and auditable, satisfying enterprise internal control and regulatory requirements.

Here, the modeling specification is defined by the organization and includes: business nodes numbered with a B prefix (B01, B02, …), control nodes numbered with a C prefix (C01, C02, …), decision conditions annotated on edges rather than as standalone nodes, swimlanes named after responsible departments, and output files named according to the "regulation code-P sequence number" rule.

3.3. Design Requirements

Based on the problem definition and organizational context, the system design must satisfy the following requirements:

- **R1 (Semantic understanding capability)**: the ability to handle complex expressions in Chinese enterprise institutional texts, including omitted subjects, clause-based long sentences, cross-chapter process integration, and implicit temporal and conditional relations;
- **R2 (Format determinism)**: the output must strictly conform to the modeling specification, including node numbering, style parameters, swimlane layout, and JSON structure, ensuring direct import into the OA system;
- **R3 (Quality verifiability)**: the generated results must undergo executable quality validation to automatically detect structural omissions, numbering errors, and layout conflicts;
- **R4 (System integrability)**: the system must be embeddable into enterprise information systems through standardized interfaces, supporting asynchronous tasks, multi-file delivery, secure authentication, and failure retry;
- **R5 (Human intervention)**: the generated results must be editable drafts that business personnel can fine-tune, forming a human-in-the-loop working model;
- **R6 (Governance traceability)**: the modeling process must support institutional-text ambiguity resolution, compliance constraint validation, version traceability, and audit trail preservation to satisfy enterprise governance requirements.

3.4. Design Science Research Paradigm

This paper follows the design science research (DSR) paradigm [21, 22]. The DSR paradigm emphasizes solving practical problems through the construction and evaluation of information technology artifacts, with a core process comprising six steps: problem identification and motivation, objectives of a solution, design and development, demonstration, evaluation, and communication. We define the regulation-to-process-model agent as a design artifact oriented toward regulatory governance and organize the research according to the six-step DSR framework:

- **Problem identification and motivation** (Sections 1 and 3): identifying the efficiency, standardization, and maintenance limitations of manual transformation from enterprise institutional text to process models, as well as the deficiencies of existing automated methods;
- **Objectives of a solution** (Section 3.3): defining the six design requirements R1–R6;
- **Design and development** (Section 4): designing and implementing the six-stage LRHA pipeline;
- **Demonstration** (Section 5): deploying and demonstrating the system in a real enterprise information system (the OA risk control system);
- **Evaluation** (Section 6): evaluating the system across efficiency, quality, human intervention, and user feedback dimensions;

- **Communication** (Section 8): summarizing the contributions and outlining future work.

The adoption of the DSR paradigm enables this paper to present not only the technical solution but also the complete research chain from problem to artifact to evaluation.

4. System Architecture and Methodology

4.1. Overall Architecture: A Six-Stage Pipeline

LRHA adopts a division of labour between the LLM and the rule engine, decomposing the transformation from enterprise institutional text to process model into six stages:

1. **Text parsing**: parsing the structure of the enterprise institutional document and extracting chapters, clauses, and metadata (regulation code, version, and name);
2. **Element extraction**: identifying process elements, including activities, control points, decision conditions, responsible departments, and temporal relations;
3. **Logic modeling**: organizing the extracted elements into a process logic structure, determining node ordering, branching, and merging relations;
4. **Rule matching**: matching graphical elements, numbering, and styles to the logic structure according to the mapping rules and modeling specification;
5. **JSON generation**: generating structured JSON files according to the JointJS data specification;
6. **Draft output**: executing quality validation and outputting editable flowchart drafts.

Stages 1–3 are handled by the LLM, relying on its semantic understanding capability, while stages 4–6 are handled by the rule engine, relying on its deterministic execution capability. This division is motivated by inherent differences between the two groups of tasks. Semantic-understanding work — identifying responsible actors, execution sequences and triggering conditions — is open-ended and naturally suited for LLMs. In contrast, format-generation tasks including numbering, styling, layout and JSON-schema enforcement represent closed, deterministic work best handled by a rule engine.

Figure 1. **Overall architecture of the six-stage LRHA pipeline**

4.2. Text Parsing and Element Extraction

The **text parsing** stage processes both the physical and logical structure of the enterprise institutional document. At the physical level, it parses the paragraphs, tables, and heading hierarchy of the docx document; at the logical level, it identifies the metadata table at the head of the regulation (regulation code, version, name, and issuing unit) and partitions the document into chapters and clauses. Metadata serves as the authoritative basis for subsequent process numbering and file naming and must be read from the document header rather than fabricated by the model.

The **element extraction** stage is performed by the LLM, whose input is the parsed enterprise institutional text and a prompt template and whose output is a structured set of process elements. Element types

include:

- **Activity**: a business action in the process, e.g., “formulating the annual international science and technology exchange work plan”;
- **Control point**: a control action such as review, approval, or filing, e.g., “project initiation report review”;
- **Decision condition**: the basis for process branching, e.g., “whether the review is passed”;
- **Responsible department**: the executing entity of an activity, used for swimlane partitioning;
- **Temporal relation**: the ordering of activities;
- **Conditional relation**: the logical relation between branching and merging.

4.3. “Textual Cue to Process Element” Mapping Rule System

Through the analysis and validation of multiple enterprise institutional documents, we derive a set of standard mapping rules from textual cues to process elements, serving as guidance for the LLM in extracting process elements. This rule system establishes correspondences between recurring linguistic patterns in enterprise institutional texts and process elements, as shown in Table 2.

Table 2
Textual cue to process element mapping rules (excerpt)

Textual cue (example)	Process element	Description
“Department X is responsible for” / “by Unit Y”	Swimlane	Responsible department, basis for swimlane naming
“first” / “then” / “finally”	Process node (ordering)	Temporal cues correspond to node execution order
“whether” / “passed” / “not passed” / “conforms” / “does not conform”	Decision branch	Decision cues correspond to branch determination
“if” / “then” / “otherwise”	Branch edge and condition label	Conditional cues correspond to branch logic
“implement by reference” / “see”	Reference/subprocess marker	Referential cues correspond to subprocess or reference annotation
“report to X for approval” / “after review by X”	Control node	Approval cues correspond to C-type control nodes
“filing” / “write-off” / “archiving”	Control node	Control cues correspond to C-type control nodes

The rule system serves two purposes. First, it is embedded as part of prompt engineering in the LLM's extraction instructions to improve the accuracy and completeness of element extraction. Second, it serves as a validation basis in the "rule matching" stage of the rule engine to perform consistency checks on the LLM's extraction results.

4.4. Logic Modeling

In the logic modeling stage, the extracted elements are organized into a process logic structure. Based on the process description in the enterprise institutional text, the LLM determines:

- **Node sequence**: the execution order of activities, forming the main process chain;
- **Branch structure**: the branch paths triggered by decision conditions, annotated with "yes/no" conditions;
- **Merge structure**: the merging points of branch paths;
- **Swimlane assignment**: the responsible-department swimlane to which each node belongs;
- **Control point embedding**: the positions of control nodes within the process chain.

The output of logic modeling is an intermediate representation (IR) comprising node lists, edge lists, and swimlane assignments, without any graphical style information. The separation of IR from graphical representation is a key design of the hybrid architecture: the LLM generates only the semantic-level IR, while all graphical-level decisions (styles, layout, and numbering) are handled uniformly by the rule engine, thereby avoiding uncertainty in the LLM's handling of format details.

4.5. Rule Engine: Styles, Layout, and Numbering

The rule engine performs the deterministic transformation from IR to graphical representation, with core components including:

(1) Node specification. According to the modeling specification, the elements in the IR are mapped to three types of graphical elements:

- **Business nodes**: rectangular boxes, numbered with a B prefix (B01, B02, …), carrying specific business actions;
- **Control nodes**: diamond-shaped boxes, numbered with a C prefix (C01, C02, …), corresponding to control activities such as review, approval, and filing;
- **Start/end nodes**: rounded rectangles, without numbering.

(2) Decision condition annotation. Decision branches do not create standalone D-type diamond nodes; instead, decision conditions are annotated directly on connecting edges (edge labels), formatted as "yes: X / no: Y". This design simplifies the graphical structure and conforms to the organization's established drawing conventions.

(3) Swimlane layout. Vertical swimlanes are adopted, with swimlane names corresponding to responsible departments; all swimlanes are of equal width and height (aligned at the bottom based on the deepest node plus margin); nodes are bound to their swimlanes via the parent attribute, with node x-coordinates falling within the corresponding swimlane interval; adjacent rows within the same swimlane are staggered to prevent overlap.

(4) Numbering and naming. Process numbering uses the regulation's internal control activity list (C01, C02, …) as the authoritative basis and the chapter clauses as the detailed basis; output files are named according to the "regulation code-P sequence number" rule (e.g., "regulation code-P01_process name.json"), with multiple processes numbered P01, P02, P03, … in sequence.

4.6. Quality Validation Mechanism

The rule engine executes ten quality validations before outputting the draft, transforming the modeling specification into machine-checkable constraints:

1. JSON is parseable and UTF-8 without BOM;
2. Swimlane heights are uniform;
3. Node parent/children bindings are consistent;
4. Node coordinates fall within the corresponding swimlane interval;
5. Nodes do not overlap;
6. Edge endpoints/ports exist and are valid;
7. Control nodes have C-prefixed numbering;
8. Business nodes have B-prefixed numbering;
9. Decision conditions exist on edge labels;
10. Text labels enable automatic line wrapping (textWrap).

The above validations cover four dimensions: structural completeness (1, 6), layout conformance (2, 4, 5), numbering consistency (7, 8, 9), and semantic completeness (3, 10). Any failure triggers the rule engine's correction process or a human intervention prompt.

4.7. Governance Capabilities for Enterprise Differentiation

This section elaborates the enterprise governance capabilities that distinguish LRHA from generic text-to-BPMN methods. These capabilities are core requirements of the regulatory governance scenario and embody the differentiated contributions of this paper.

(1) institutional-text ambiguity resolution. Enterprise institutional texts often contain vague expressions, conflicting clauses, and overlapping responsibilities. LRHA resolves ambiguity through a three-layer mechanism. First, the mapping rule system (Section 4.3) establishes deterministic correspondences between linguistic patterns in enterprise institutional texts and process elements, reducing semantic ambiguity. Second, the rule engine performs consistency checks after logic modeling to identify anomalies such as node conflicts and responsibility overlaps. Third, for ambiguities that cannot be

automatically resolved, the system outputs an ambiguity list for adjudication by business experts, forming a human-in-the-loop resolution loop.

(2) Compliance constraint validation. Process models must conform to organizational internal control norms and regulatory requirements. LRHA codifies compliance rules as executable constraints in the rule engine, including: control node completeness (control steps such as approval and filing must not be omitted), separation of duties (incompatible positions must not be assumed by the same entity), and numbering consistency (internal control activity numbers must be consistent with the authoritative basis in the regulation). The ten quality validations (Section 4.7) constitute the machine-based expression of compliance constraints; any output that violates compliance rules is automatically intercepted and flagged for correction.

(3) Approval OA integration. Generated process models are directly embedded into the enterprise's existing approval OA system through asynchronous task interfaces (Section 5.2), closing the loop from regulation to execution. Business personnel can directly reference and edit the generated process drafts within the OA system without switching between external tools and the OA system, reducing the technical barrier to process implementation.

(4) Version traceability. After regulation revisions, LRHA supports difference identification between old and new versions of processes through the association of version metadata (regulation code and version) with process numbering.

(5) Audit trail preservation. The entire modeling process is traceable and auditable. The system records complete logs of task submission, status transitions, and result delivery, and employs idempotency keys and HMAC signature verification to ensure the reliability and non-repudiation of tasks and callbacks, satisfying the audit requirements of enterprise internal control and regulation.

5. System Implementation and OA Integration

5.1. Agent Platform Implementation

The system is implemented on the general-purpose agent platform of a large energy enterprise group, which supports local processing of internal documents to satisfy enterprise data security requirements. The agent is deployed on the platform in the form of a "digital employee," accessible to business users through the platform's AI entry point, the group OA application navigation bar, or a dedicated URL. It supports batch import of enterprise institutional texts and generation triggered by simple prompts (e.g., "parse this regulation and output JointJS JSON process data in the fixed format"). The system implementation follows the architectural principle of "the LLM handles the first three steps and the rule engine handles the latter three steps," codifying the modeling specification as executable code and configuration to ensure the stability of output formatting.

Key engineering experience accumulated in practice includes:

- **Development of OA canvas editing functions**: We developed supporting functions on the corresponding canvas of the OA system, including “one-click repair,” “endpoint reset,” and a “text-to-diagram library,” accompanied by user manuals. Specifically, “one-click repair” automatically reorganizes scattered edges (primarily addressing edge detours and crossings), “endpoint reset” adjusts edge endpoint positions to make the graphical layout clear and standardized, and the “text-to-diagram library” centrally manages the process drafts generated by the agent and supports referencing and editing. Together with the generated process drafts, these functions constitute a complete “generate–repair–edit–reference” workflow loop, reducing the usage barrier for business personnel;
- **Edge routing optimization**: we adopt zero-waypoint orthogonal edge styling (orthogonalEdgeStyle), allowing the graphics library to perform native automatic routing, which both prevents edges from crossing shapes and maintains smooth dragging;
- **Node file association capability**: both control nodes and business nodes can be associated with control documents or output files, enabling the process model to carry not only activities and control flow but also governance elements such as regulatory bases, approval forms, and output documents, thereby providing file-level support for process execution and compliance traceability.

These engineering practices serve a broader objective: making the generated process model not merely a diagram but an executable carrier of enterprise risk and compliance governance. Through node file association, control nodes can be linked to the corresponding internal control activity bases, approval forms, and control requirements, while business nodes can be associated with output documents and deliverables, thereby explicitly mapping the compliance constraints, approval chains, and responsibility assignments in enterprise institutional texts onto the process model. Combined with the compliance constraint validation and audit trail preservation capabilities described in Section 4.7, the process models generated by LRHA can directly support the operation of the enterprise internal control system. The process model serves as a bridge between the internal control activity list (control nodes) and the regulatory clauses, supporting internal control effectiveness evaluation and risk prevention. In addition, node-level file association provides structured support for evidence retention and audit traceability during process execution, forming a complete “regulation–process–execution–audit” loop that aligns with the governance requirements of large enterprise groups for the digitalization of risk internal control.

5.2. Asynchronous Task Interface Design

To achieve end-to-end automated integration with the group’s OA risk control system (regulation upload → text parsing → result write-back), the system provides RESTful-style asynchronous task interfaces comprising four core operations:

(1) Task submission. A multipart/form-data form is used, containing the enterprise institutional document file, a business request ID, business fields (a JSON object including regulation code, regulation name, version, and organization code), and an optional protocol version.

(2) Status query. The response returns the task status (queued/launching/running/succeeded/failed), callback status (not_ready/pending/delivered/exhausted/disabled), timestamps, error information, and the result list.

(3) Result download. Multi-file download is supported, with the response including headers such as Content-Disposition (UTF-8 encoded file name) and ETag (file SHA-256 digest).

(4) Proactive callback. The platform proactively pushes results to the OA-side fixed callback address upon task success or failure, using multipart/form-data containing a manifest (JSON list) and the individual result file parts.

5.3. Security Mechanisms: Authentication, Signature Verification, and Idempotency

The system designs multiple layers of security mechanisms to ensure the reliability and security of interface calls:

(1) Machine authentication. Submission, query, and download all use an OA-specific interface key, passed through the Authorization: Bearer <INGEST_KEY> header; the interface access key and the callback signature key are distinct credentials.

(2) HMAC signature verification. Callback messages are signed using HMAC-SHA256. The message participating in the signature differs by protocol version: v2 uses UTF8("v2." + timestamp + "." + manifest raw text), while v1 uses UTF8(timestamp + "." + manifest raw text + "." + single-file sha256).

(3) Idempotency handling. On the submission side, repeated submission with the same idempotency key, request ID, business fields, file content, and protocol version returns the original job_id without restarting the task; the same key with different content returns 409. On the callback side, deduplication is performed using Idempotency-Key: oa-policy-flow-callback:<job_id>, and repeated callbacks that have been fully saved directly return 2xx.

(4) Error handling. The interface directly uses HTTP status codes to express error semantics (401 authentication failure, 404 task not found, 409 idempotency conflict, 410 result unreadable, 413/415 limit exceeded or unsupported type, 422 invalid parameter, 500 internal exception, 503 service unavailable), with bounded retry (up to 8 times) for network errors and 408/409/425/429/5xx statuses.

5.4. Protocol Evolution: v1-to-v2 Multi-File Delivery

The system interface has evolved from single-file (v1) to multi-file (v2) delivery to support scenarios in which one regulation generates multiple independent process files. The main changes include:

- **Version selection**: a result_protocol=v2 field is added to the request; omitting it or passing v1 follows the legacy protocol;

- **Result delivery**: v1 delivers a single JSON file, while v2 delivers a list of 1 to 10 independent JSON files (the results array, containing result ID, file name, byte count, and SHA-256 digest);
- **Download method**: v1 uses /{job_id}/result, while v2 uses /{job_id}/results/{result_id};
- **Callback content**: the v2 callback manifest carries the file parts corresponding to the list, no longer using the single-value result_file field;
- **Signature algorithm**: the v2 signature message adds a “v2.” prefix.

The protocol evolution follows a backward-compatibility principle: legacy calls that omit the version continue to be processed as v1, and the callback receiver is compatible with both v1 and v2. Engineering practice requires the OA side to traverse and download all files in v2 mode, associate them by job_id + result_id, save them idempotently, verify byte counts and SHA-256 against the raw bytes before parsing JSON, and load each file independently into JointJS, without displaying only the first file or concatenating multiple JSON texts.

6. Case Evaluation

6.1. Evaluation Framework and Metrics

To evaluate the effectiveness of LRHA in a real production environment, this paper follows the evaluation phase of the DSR paradigm and constructs an evaluation framework across four dimensions: efficiency, quality, human intervention, and user feedback.

- **Efficiency metrics**: the end-to-end time from enterprise institutional text input to editable draft output (), compared against the manual modeling baseline;
- **Quality metrics**: structural completeness (whether nodes, edges, and swimlanes are complete), specification conformance (whether numbering, styles, and layout conform to the modeling specification), and semantic fidelity (whether the process logic is consistent with the enterprise institutional text);
- **Human intervention metrics**: the fine-tuning workload of business personnel on the generated drafts (number of modified nodes, number of adjusted edges, and time);
- **User feedback metrics**: subjective evaluations of system usability, accuracy, and satisfaction collected from business personnel (risk control and regulation administrators) through expert scoring and user questionnaires.

Among these, structural completeness and specification conformance can be automatically determined by the quality validation mechanism in Section 4.7, while semantic fidelity, human intervention workload, and user feedback are manually evaluated by business experts and users.

6.2. Efficiency Evaluation Results

The system has undergone multiple rounds of validation on real enterprise institutional documents. Taking a complex process comprising more than 40 process nodes and 40 edges as an example (some regulations contain multiple independent processes), the end-to-end time for LRHA to generate an editable draft is approximately 20 minutes, whereas manual modeling of a process of equivalent complexity typically requires several days. Across multiple enterprise institutional documents, the process modeling cycle is reduced from several days of manual effort to approximately 20 minutes, an efficiency improvement of more than an order of magnitude. After the draft is generated, business personnel need only minute-level fine-tuning of certain details (e.g., adjusting individual node positions and correcting edge endpoints) before it can be put into use.

Table 3 summarizes the efficiency comparison across typical scenarios.

Table 3
Efficiency comparison across typical scenarios

Scenario	Manual modeling (baseline)	LRHA generation	Manual fine-tuning	Total time (LRHA path)
Medium-complexity regulation (single process)	Several hours	~ 10 minutes	Minute-level	~ 15 minutes
Complex regulation (40 + nodes, 40 + edges)	Several days	~ 20 minutes	Minute-level	~ 25 minutes
Multi-process regulation (multiple independent processes)	Several days	~ 20 minutes	Minute-level	~ 30 minutes
Note: The data in the table are typical values from multiple rounds of validation in the production environment; the specific time is mainly affected by computational resources, regulation length, and process complexity.				

6.3. Quality Evaluation Results

Structural completeness and specification conformance. The quality validation mechanism automatically checks the generated results, covering ten checks including JSON parseability, swimlane layout, node binding, numbering consistency, and decision condition annotation. In the completed validation cases, the drafts output after rule-engine correction all pass the structural completeness and specification conformance checks and can be directly imported into the OA risk control system for rendering.

Semantic fidelity. Business experts performed item-by-item comparison between the generated drafts and the enterprise institutional texts. The validation shows that for enterprise institutional documents with clear process logic and complete process elements, the node sequences, control point positions, and decision branches extracted by LRHA are highly consistent with the enterprise institutional texts. For enterprise institutional texts with implicit temporal ordering, cross-chapter references, or vague expressions, occasional element omissions or branch logic deviations occur, requiring manual fine-

tuning correction. This finding is consistent with the conclusion in the existing literature regarding the insufficiency of LLMs in semantic correctness for process modeling tasks [5, 7], but the hybrid architecture significantly reduces the error rate at the format level through the deterministic constraints of the rule engine.

6.4. Expert Scoring and User Questionnaire Evaluation

To systematically collect real feedback from business personnel, this paper designs an evaluation mechanism combining expert scoring and user questionnaires. This section presents the design of the evaluation instruments (sample forms); large-scale data collection and statistical analysis based on these instruments will be conducted as future work, with the current version of this paper focusing on system design and preliminary validation.

(1) Expert scoring. Experts from business domains such as risk control and regulation management are invited to score the generated process drafts across dimensions including semantic fidelity, specification conformance, usability, and completeness, using a five-point Likert scale (1 = very poor, 5 = very good). The expert scoring sample form is shown in Table 4.

Table 4
Expert scoring sample form (five-point Likert scale)

Evaluation dimension	Evaluation description	1	2	3	4	5
Semantic fidelity	Consistency between process logic and enterprise institutional text	☐	☐	☐	☐	☐
Specification conformance	Whether numbering, styles, and layout conform to the modeling specification	☐	☐	☐	☐	☐
Structural completeness	Whether nodes, edges, and swimlanes are complete	☐	☐	☐	☐	☐
Usability	Degree to which the draft can be directly edited and referenced	☐	☐	☐	☐	☐
Compliance	Whether compliance constraints such as control nodes and separation of duties are satisfied	☐	☐	☐	☐	☐

(2) User questionnaire. Questionnaires are distributed to business personnel who actually use the system (risk control and regulation administrators) to collect their subjective evaluations of system ease of use, efficiency improvement, result satisfaction, and improvement suggestions. The questionnaire is administered anonymously to ensure the authenticity of feedback. The user questionnaire sample form is shown in Table 5.

Table 5
User questionnaire sample form (five-point Likert scale)

Item	1	2	3	4	5
The system is easy to operate and learn	☐	☐	☐	☐	☐
Compared with manual modeling, it significantly improves work efficiency	☐	☐	☐	☐	☐
The quality of the generated process drafts meets business requirements	☐	☐	☐	☐	☐
Integration with the OA system is smooth	☐	☐	☐	☐	☐
Overall satisfaction	☐	☐	☐	☐	☐
(Note: Tables 4 and 5 are sample forms of the evaluation instruments. Large-scale data collection and statistical analysis (mean, standard deviation, sample size, etc.) based on these instruments will be conducted as future work; the current version of this paper does not include the relevant statistical results.)					

6.5. Deployment and Promotion

The system was deployed in production on October 15, 2026, and has been invoked in the background of the OA risk control system of a large energy enterprise group, establishing an end-to-end automated chain of regulation upload, text parsing, and result write-back. The system provides three invocation entry points: the “text-to-diagram” button on the regulation detail page, new invocation in the text-to-diagram library, and reference invocation on the process management page. The generated results support three reference methods (submission to the process management library, reference to the text-to-diagram result, and JSON file import), accompanied by editing functions such as one-click repair, endpoint reconnection, and node editing. A closed-loop management mechanism of “release–claim–review–approval–implementation–acceptance–promotion” has been established, providing organizational support for the continuous iteration and cross-domain replication of the system.

7. Discussion

7.1. Effectiveness of the Hybrid Architecture

The central claim of this paper is that, in the task of automated process model generation from enterprise institutional text, there exists a clear division of labour between LLMs and rule engines: the LLM handles open-ended semantic understanding, while the rule engine handles closed-form format generation. The case evaluation results support this claim: the hybrid architecture achieves fully controllable output formatting while preserving the flexibility of semantic extraction, enabling the generated results to be directly imported into enterprise information systems—a capability that pure LLM methods cannot reliably achieve.

From a broader perspective, LRHA embodies the engineering value of neuro-symbolic AI in document-to-model transformation tasks [10, 11]. Its key design lies in the **separation of the intermediate representation (IR) from the graphical representation**: the LLM's output is confined to the semantic-level IR, while all graphical-level decisions (numbering, styles, layout, and JSON structure) are completed uniformly by the rule engine. This design isolates the uncertainty of the LLM at the semantic layer, making the format layer fully deterministic, thereby achieving the goal of "semantically flexible, format-strict" at the system level.

7.2. Comparison with Pure LLM Methods

Compared with methods that directly require the LLM to output complete JSON or graphical descriptions, LRHA offers the following advantages:

- **Format stability**: the rule engine ensures that numbering, styles, and layout strictly conform to organizational specifications, eliminating the format drift of LLM outputs;
- **Verifiability**: the ten quality validations transform the specification into machine-checkable constraints, enabling any deviation to be automatically detected and corrected;
- **Maintainability**: changes to the modeling specification require only modification of the rule engine configuration, without retraining or adjusting prompts;
- **Explainability**: the separation of IR from graphical representation decouples process logic (IR) from presentation form (graphics), facilitating audit and traceability.

The trade-off is that the introduction of the rule engine increases system complexity, and the mapping rule system requires customized development for organizational specifications. For organizations whose modeling specifications are not yet standardized, or whose process expressions are highly unstructured, the benefits of the rule engine may be limited.

7.3. Differentiated Value for Enterprise Governance

Distinct from generic text-to-BPMN methods that focus solely on graphical generation, the value of LRHA lies in its comprehensive support for enterprise governance scenarios. Capabilities such as institutional-text ambiguity resolution, compliance constraint validation, approval OA integration, version traceability, and audit trail preservation position LRHA as a component of the digital infrastructure for enterprise regulatory governance, rather than a standalone diagram-generation tool. These capabilities originate from an in-depth understanding of the requirements of regulatory governance and are implemented at the system level through the hybrid architecture, rather than being appended post hoc.

7.4. Organizational Support and Promotion Mechanism

The successful deployment of the system depends not only on the technical architecture but also on organizational support mechanisms. The group has addressed the issues of resource support, expert participation, and cross-departmental collaboration in agent construction through mechanisms such as establishing a dedicated task force, building a closed-loop management of "release–claim–review–

approval–implementation–acceptance–promotion," and adopting a pilot-first, gradual-promotion approach. This experience indicates that the deployment of enterprise-grade AI systems is a process of co-evolution among technology, organization, and institutions, and that organizational mechanism design should advance in tandem with the technical architecture.

7.5. Limitations and Threats to Validity

This work has the following limitations:

- **Limited evaluation sample**: the current efficiency and quality data come from multiple rounds of validation in the production environment, with a limited sample size and without a large-scale, systematic benchmark; semantic fidelity evaluation relies on subjective judgment by business experts and lacks quantitative metrics (e.g., node-level precision/recall). In addition, the large-scale data collection and statistical analysis based on the expert scoring and user questionnaire instruments designed in Section 6.4 have yet to be conducted, and the relevant statistical results will be supplemented as future work;
- **Domain specificity**: the mapping rule system is derived from the linguistic characteristics of the enterprise institutional texts of a large energy enterprise group, and its cross-organization and cross-language generalizability remains to be validated;
- **LLM capability dependence**: the quality of element extraction and logic modeling is influenced by the underlying LLM capability, and performance differences across models have not been systematically evaluated;
- **Reinforcement learning tuning not yet completed**: reinforcement learning based on feedback from business domain experts is still ongoing, and the current version has not fully exploited expert feedback signals.

Furthermore, due to enterprise data security and confidentiality constraints, the original internal institutional document dataset cannot be publicly released. Nevertheless, all core design principles, mapping-rule specifications, quality-validation checks, and interface protocols of the LRHA artifact are fully documented within this paper, so that researchers and practitioners can replicate the hybrid architecture for comparable institutional-text-to-process-model tasks.

Consistent with the DSR philosophy, this paper emphasizes artifact design and real-world production-environment deployment. Large-scale benchmarking with quantitative node-level metrics and cross-organizational comparative experiments are reserved as future research work.

To address these limitations, future work will construct standardized evaluation datasets and quantitative evaluation metrics, conduct cross-model and cross-domain comparative experiments, and complete reinforcement learning tuning based on expert feedback.

8. Conclusion and Future Work

Following the design science research paradigm, this paper proposes and deploys a hybrid LLM and rule engine architecture (LRHA) for the automated generation of process models from enterprise institutional documents. The architecture decomposes the transformation task into six stages—text parsing, element extraction, logic modeling, rule matching, JSON generation, and draft output—with the LLM handling the first three stages that rely on semantic understanding and the rule engine handling the latter three stages that rely on deterministic constraints, thereby separating semantic interpretation from format enforcement. The system is deployed in a large energy enterprise group, achieving end-to-end automation from enterprise institutional documents to JointJS editable process models, and has been integrated with the group's OA risk control information system through asynchronous task interfaces.

The main contributions of this paper include: (1) proposing and validating the division of labour and collaboration mechanism between LLMs and rule engines in automated process model generation, providing a reusable hybrid architecture paradigm for document-to-model transformation tasks; (2) constructing a "textual cue to process element" mapping rule system and ten executable quality validation specifications tailored to Chinese enterprise institutional texts, codifying organizational modeling standards as machine-checkable constraints; (3) providing native support for enterprise governance capabilities at the architectural level—including institutional-text ambiguity resolution, compliance constraint validation, approval OA integration, version traceability, and audit trail preservation—distinguishing LRHA from generic text-to-BPMN methods that focus solely on graphical generation; and (4) reporting the complete deployment pathway of the hybrid architecture within an enterprise information system, including asynchronous task interfaces, HMAC signature verification and idempotency mechanisms, multi-file protocol evolution, and real production data. Case evaluation demonstrates that the generation time for an editable draft of a complex process is reduced from several days of manual effort to approximately 20 minutes, with only minute-level manual fine-tuning required, validating the effectiveness of the hybrid architecture in terms of both efficiency and quality.

Future work includes: (1) conducting large-scale data collection and statistical analysis (mean, standard deviation, sample size, etc.) based on the expert scoring and user questionnaire instruments designed in Section 6.4, and constructing standardized evaluation datasets and quantitative evaluation metrics (e.g., node-level precision/recall, process similarity) for large-scale systematic evaluation; (2) conducting reinforcement learning tuning based on feedback from business domain experts to continuously consolidate expert knowledge into the system; (3) exploring the cross-organization and cross-language generalization of the mapping rule system and evaluating its applicability to enterprise institutional texts in other industries; (4) extending the hybrid architecture to adjacent tasks such as regulatory review opinion extraction and regulatory review table generation, building a risk internal control domain agent ecosystem; and (5) researching process intelligent validation and version comparison management capabilities to achieve automatic identification of process differences and generation of change descriptions after regulation revisions.

Declarations

Funding statement:

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Declaration of Generative AI and AI-assisted technologies in the writing process

Generative-AI tools were employed solely for linguistic polishing and sentence-level rewriting during manuscript preparation. All problem formulation, artifact design, experimental observations, argumentation, analysis, and conclusions are human-authored. Full human oversight and revision have been applied throughout the manuscript.

References

1. Racz N, Weippl E, Seufert A (2010) A frame of reference for research of integrated governance, risk and compliance (GRC). In *IFIP International Conference on Communications and Multimedia Security* (pp. 106–117). Springer. https://doi.org/10.1007/978-3-642-13241-4_11
2. Friedrich F, Mendling J, Puhlmann F (2011) Process model generation from natural language text. In *International Conference on Advanced Information Systems Engineering (CAiSE 2011)* (pp. 482–496). Springer. https://doi.org/10.1007/978-3-642-21640-4_36
3. Epure EV, Martín-Rodríguez P, Hug C, Deneckère R, Salinesi C (2015) Automatic process model discovery from textual methodologies. In *2015 IEEE 9th International Conference on Research Challenges in Information Science (RCIS)* (pp. 19–30). IEEE. https://doi.org/10.1109/RCIS.2015.7128860
4. Sánchez-Ferreres J, Burattin A, Carmona J, Montali M, Padró L (2019) Formal reasoning on natural language descriptions of processes. In *Business Process Management (BPM 2019)*, LNCS 11675 (pp. 86–101). Springer. https://doi.org/10.1007/978-3-030-26619-6_8
5. Kourani H, Berti A, Schuster D, van der Aalst WMP (2024) Process modeling with large language models. In *International Conference on Advanced Information Systems Engineering (CAiSE 2024)* (pp. 229–244). Springer. http://dx.doi.org/10.1007/978-3-031-61007-3_18
6. Grohs M, Abb L, Elsayed N, Rehse J-R (2023) Large language models can accomplish business process management tasks. In *Business Process Management (BPM 2023)* (pp. 453–465). Springer. https://doi.org/10.1007/978-3-031-50974-2_34
7. Vidgof M, Bachhofner S, Mendling J (2023) Large language models for business process management: Opportunities and challenges. arXiv preprint arXiv:2304.04310.
8. Ji Z, Lee N, Frieske R, Yu T, Su D, Xu Y, Ishii E, Bang YJ, Madotto A, Fung P (2023) Survey of hallucination in natural language generation. ACM-CSUR 55(12):1–38. https://doi.org/10.48550/arXiv.2202.03629
9. Huang L, Yu W, Ma W, Zhong W, Feng Z, Wang H, Chen Q, Peng W, Feng X, Qin B, Liu T (2023) A survey on hallucination in large language models: Principles, taxonomy, challenges, and open

questions. *arXiv preprint* arXiv:2311.05232

10. Garcez A, d'A, Lamb LC (2023) Neurosymbolic AI: The 3rd wave. Artif Intell Rev 56(11):12387–12406. https://doi.org/10.1007/s10462-023-10448-w
11. Kautz H (2022) The third AI summer: AAAI Robert S. Engelmore Memorial Lecture. AI Magazine 43(1):105–125. https://doi.org/10.1002/aaai.12036
12. Riehle DM, Jirendra S, Freund M, Greiner T, Krcmar H, Matthes F (2017) Text2BPMN: A tool for generating BPMN models from text. In *Business Process Management (BPM 2017) (Demos)*. CEUR Workshop Proceedings
13. Leopold H, Mendling J, Polyvyanyy A (2014) Supporting process model validation through natural language generation. IEEE Trans Software Eng 40(8):818–840. http://dx.doi.org/10.1109/TSE.2014.2327044
14. Berti A, Schuster D, van der Aalst WMP (2023) Abstractions, scenarios, and prompt definitions for process mining with LLMs. In *Business Process Management (BPM 2023) (Demos)*. CEUR Workshop Proceedings
15. Geng S, Josifoski M, Peyrard M, West R (2023) Grammar-constrained decoding for structured NLP tasks without finetuning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP 2023)* (pp. 10932–10952). https://doi.org/10.48550/arXiv.2305.13971
16. Mendling J, Reijers HA, van der Aalst WMP (2010) Seven process modeling guidelines (7PMG). Inf Softw Technol 52(2):127–136. https://doi.org/10.1016/j.infsof.2009.08.004
17. Becker J, Rosemann M, von Uthmann C (2000) Guidelines of business process modeling. In *Business Process Management (LNCS 1806)* (pp. 30–49). Springer. http://dx.doi.org/10.1007/3-540-45594-9_3
18. El Kharbili M, de Medeiros AKA, Stein S, van der Aalst WMP (2008) Business process compliance checking: Current state and future challenges. In *Modellierung 2008* (pp. 107–121)
19. van der Aalst WMP (2016) *Process Mining: Data Science in Action* (2nd ed.). Springer. https://doi.org/10.1007/978-3-662-49851-4
20. Dumas M, La Rosa M, Mendling J, Reijers HA (2018) *Fundamentals of Business Process Management* (2nd ed.). Springer. https://doi.org/10.1007/978-3-662-56509-4
21. Hevner AR, March ST, Park J, Ram S (2004) Design science in information systems research. MIS Q 28(1):75–105. https://doi.org/10.2307/25148625
22. Peffers K, Tuunanen T, Rothenberger MA, Chatterjee S (2007) A design science research methodology for information systems research. J Manage Inform Syst 24(3):45–77. https://doi.org/10.2753/MIS0742-1222240302

Figures

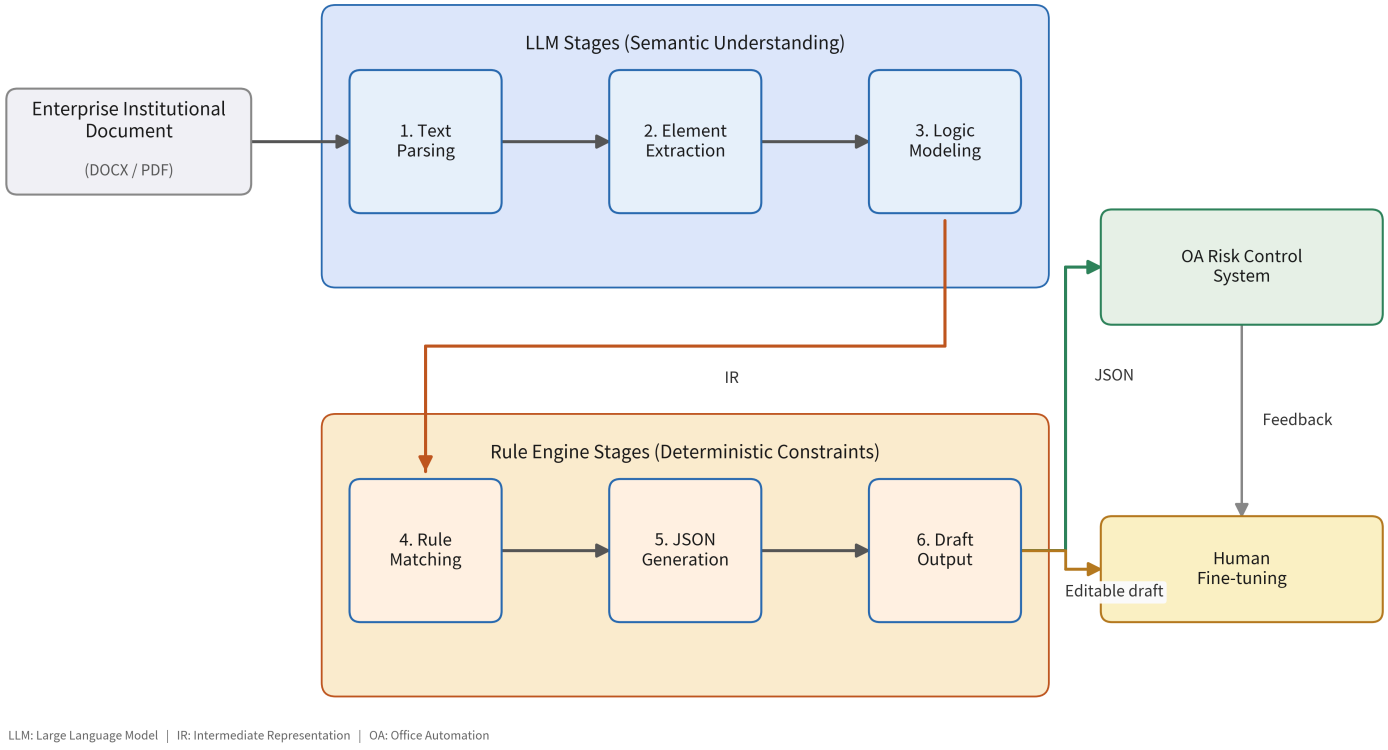


Figure 1

Overall architecture of the six-stage LRHA pipeline