



BDMA Master Thesis

SEC BPMN-GPT: A HYBRID LLM & RULE-BASED FRAMEWORK FOR AUTOMATING SECURITY ANNOTATION IN BUSINESS PROCESS MODELS

Md Kamrul ISLAM

GitHub Link ¹

Advisor: Tiphaine HENRY – CEA LIST tiphaine.henry@cea.fr
Sami SOUIHI – CEA-LIST / UPEC Sami.SOUIHI@cea.fr

¹ *GitHub Repository:* github.com/tiphainehenry/BPM-cybersec-extension-generation

Abstract

Secure-by-design business process modeling requires the explicit integration of security requirements into process models, yet this remains largely manual. The Security Business Process Model and Notation (SecBPMN) standard provides a formal means of expressing such constraints but lacks scalable automation methods. This thesis presents an end-to-end hybrid framework that combines rule-based and large language models (LLMs) to automate the extraction and generation of SecBPMN annotations from natural-language process descriptions. The proposed pipeline integrates process normalization, LLM-driven constraint extraction, LLM-based schema mapping, and reconstruction into valid SecBPMN XML models, effectively bridging the gap between unstructured text and formal security representations.

The framework was evaluated on 27 text–SecBPMN pairs using three annotation strategies such as Prompt Engineering, Retrieval-Augmented Generation (RAG), and STS-ML guidance extraction across GPT-4.1-mini and Mistral-small models. Post-mapping schema enforcement improved both accuracy and structural validity, increasing average F1-scores by 8–12% and reducing spurious annotations by nearly 50%. GPT-4.1-mini achieved the highest accuracy ($F1 = 0.715$ for simple processes), while Mistral-small demonstrated superior efficiency and lower latency. Correlation analysis between objective metrics and LLM-as-Judge evaluations revealed strong alignment, indicating that language models can act as consistent evaluators of annotation quality. Compared with human experts, the SecBPMN-GPT framework achieved higher annotation accuracy ($F1 = 0.516$) while reducing annotation time by 95%.

Overall, the results demonstrate that the proposed hybrid LLM– and rule-based framework advances the automation of security annotation in business process models, offering a scalable foundation for secure-by-design process engineering and opening pathways toward intelligent, regulation-aware workflow generation in future systems.

Keywords: Security Business Process Modeling, SecBPMN, Large Language Models, Information Extraction, Schema Mapping, Hybrid Reasoning, Secure-by-Design

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my supervisors, Dr. Tiphaine Henry and Prof. Sami Souihi, for granting me the opportunity to conduct research on this fascinating topic and for their invaluable guidance throughout my internship. I am profoundly indebted to Dr. Tiphaine Henry for her exceptional mentorship, patience, and continuous encouragement. Her insightful feedback and meticulous attention to detail helped me refine my ideas and transform them into a concrete and meaningful research contribution. She taught me how to approach research methodically and to communicate my findings effectively. It has been a true privilege and honor to work and learn under her supervision. I am especially thankful for her tireless support and commitment in helping me bring this work to a production ready state. My sincere thanks also go to LISSI, UPEC, for providing the necessary funding and research environment, and to my lab colleagues for their motivation and constant inspiration. I am also grateful to the domain expert who participated as a human baseline in the user experience study, and contributed through open discussions and research exchanges. My heartfelt thanks extend to Prof. Mattia Salnitri and Prof. Julius Köpke for providing the real-world dataset, valuable insights, and enthusiasm that were instrumental in shaping this work and driving its innovative direction. I am deeply grateful to the BDMA program and all the professors for offering me the opportunity to pursue my master's studies across different universities around the world, an experience that has profoundly shaped both my academic and personal perspectives. I would like to extend my heartfelt appreciation to Prof. Nacera Seghouani, for her valuable advice, encouragement, and continuous support during my M2 studies. I owe my deepest gratitude to my parents for their unconditional love, emotional support, and countless sacrifices that have enabled me to follow my dreams. I am also thankful to my sister for her affection, kindness, and constant motivation. Finally, I would like to thank all my friends, classmates, and acquaintances who have supported me directly or indirectly throughout the BDMA journey. Your friendship and encouragement have made this experience truly memorable.

Contents

Abstract	i
Acknowledgments	ii
1 Introduction	1
1.1 Research Problem and Motivation	2
1.2 Objectives and Scope	4
1.3 Main Contributions of the Work	5
1.4 Organization and Structure	7
2 Background	8
2.1 Business Process Management and BPMN 2.0	8
2.1.1 Core BPMN Elements	9
2.2 BPMN Security Extensions	12
2.2.1 SecBPMN-ml: A Modeling Language for Secure Business Processes	13
2.3 Information Extraction and Security Requirement Extraction	22
2.3.1 Overview of Information Extraction	22
2.3.2 Information Extraction to Structured Output with LLMs	23
2.3.3 Retrieval-Augmented Generation (RAG)	26
3 Related Work	29
3.1 Security-Aware Extensions of BPMN	29
3.2 Large Language Models for BPMN Generation and Analysis	31
3.3 Requirement Extraction and Evaluation	34
4 Proposed Approach	37
4.1 Overall Approach	38
4.2 Pipeline Stages	40
4.2.1 Step 1: BPMN XML to JSON Normalization	40
4.2.2 Step 2: LLM-based Cybersecurity Label Extraction	42
4.2.3 Step 3: Security Label Mapping to BPMN Elements	46

4.2.4	Step 4: SecBPMN JSON to XML Reconstruction	49
5	Implementation and Evaluation	52
5.1	Prototype Implementation	53
5.2	Experimental Setup	56
5.2.1	Dataset	56
5.2.2	Human Annotation Baseline	62
5.2.3	Security Label Extraction Paradigms and Prompting Strategies	66
5.2.4	Model Families	70
5.2.5	Evaluation Methods	71
5.2.6	Experimental Design	71
5.2.7	Evaluation Metrics	73
5.2.8	Efficiency Benchmark	76
5.3	Results	77
5.3.1	Classification Performance	77
5.3.2	LLM-as-Judge Evaluation	84
5.3.3	Efficiency Benchmark: Token Cost and Latency Analysis	87
5.3.4	Expert vs. SecBPMN-GPT Comparison	89
6	Conclusion and Future Work	90
	Bibliography	93

List of Figures

2.1	An example of a simple Business Process Model White (2004)	11
2.2	RBT negotiation process model annotated with security constraints Salnitri et al. (2017)	20
4.1	Overview of the proposed end-to-end pipeline	39
4.2	Pipeline of the LLM-based cybersecurity label extractor.	43
4.3	LLM-based security labels mapper with BPMN elements	46
5.1	SecBPMN-GPT prototype architecture	54
5.2	Itinerary Details Transmission process model enriched with automatically generated cybersecurity annotations by SecBPMN-GPT.	55
5.3	Domain coverage of the curated dataset.	57
5.4	Pre-mapping confusion matrix for GPT-4.1-mini across methods and process complexities. Each cell represents the average counts of True Positives (TP), False Positives (FP), and False Negatives (FN).	82
5.5	Post-mapping confusion matrix for GPT-4.1-mini across methods and process complexities.	82
5.6	Pre-mapping confusion matrix for Mistral-small across methods and process complexities.	83
5.7	Post-mapping confusion matrix for Mistral-small across methods and process complexities.	84
5.8	Correlation between LLM-as-Judge metrics and objective metrics for GPT-4.1-mini. Left: Pearson correlation coefficients; Right: Spearman rank correlations.	86
5.9	Correlation between LLM-as-Judge metrics and objective evaluation metrics for Mistral-small. Left: Pearson correlation coefficients; Right: Spearman rank correlations.	86
5.10	Average total token consumption by method and model.	87
5.11	Average total latency by method and model. Latency is measured as total end-to-end response time (milliseconds) per evaluation run.	88

List of Tables

1	Comprehensive list of acronyms used throughout the thesis.	ix
1.1	Overview of the Thesis Structure	7
2.1	Overview of security goals in SecBPMN, with corresponding icons and short descriptions.	15
2.2	Formal predicates associated with each security goal, grouped by the annotated BPMN element type.. . . .	18
2.3	Mapping of SecBPMN security goals to applicable BPMN element types. (✓: applicable, ✗: not applicable).	19
3.1	Comparison of BPMN and SecBPMN modeling approaches.	35
4.1	Excerpt of BPMN XML to JSON transformation for the <i>Send Itinerary Details</i> task in the iternery details transmission process.	42
5.1	Comprehensive structural and security overview of all curated SecBPMN process models. Simple models are shown in green, Medium in orange, and Complex in red. The final five columns detail the distribution of security targets across activities (ACT), data objects (DO), message flows (MF), gateways (GW), and organizational constraints (ORG). . .	59
5.2	Annotation effort distribution across domain experts.	63
5.3	Inter-annotator agreement metrics across overlapping SecBPMN process models. T1, T2, T3 denote Tester1, Tester2, and Tester3 respectively. .	65
5.4	Classification performance (F1, Precision, Recall) by method, model, and process complexity (Pre-Mapping).	78
5.5	Classification performance (F1, Precision, Recall) by method, model, and process complexity (Post-Mapping).	79
5.6	F1-score before vs. after post-mapping by method, model, and process complexity (absolute Δ). Upward arrows (\uparrow) denote improvement, downward arrows (\downarrow) indicate decline, and horizontal arrows (\rightarrow) represent no change.	80

5.7	Structural quality metrics before vs. after post-mapping (Jaccard, Completeness, Spurious Rate) for GPT-4.1-mini.	81
5.8	Structural quality metrics before vs. after post-mapping (Jaccard, Completeness, Spurious Rate) for Mistral-small.	81
5.9	LLM-as-Judge evaluation of SecBPMN generation quality using three assessment dimensions: Accuracy, Completeness, and Correctness by method, model, and process complexity.	85
5.10	Comparison between human expert annotators and the automated SecBPMN-GPT chatbot. N denotes the number of annotated models. .	89

List of Equations

5.1 Jaccard Index for measuring inter-annotator overlap.	64
5.2 Cohen’s Kappa coefficient for inter-annotator agreement.	64
5.3 Definition of Precision.	73
5.4 Definition of Recall.	73
5.5 Computation of F1-score.	73
5.6 Definition of Jaccard Index.	74
5.7 Definition of Completeness.	74
5.8 Computation of Schema Validity.	74
5.9 Definition of Spurious Rate.	74
5.10 Definition of Redundancy Rate.	74
5.11 Average Task Similarity using cosine similarity of BERT embeddings.	74
5.12 Pearson correlation coefficient measuring linear association.	76
5.13 Spearman rank correlation coefficient measuring monotonic association.	76

Acronym	Definition
API	Application Programming Interface
BPM	Business Process Management
BPMN	Business Process Model and Notation
CI	Confidence Interval
F1	Harmonic Mean of Precision and Recall
GDPR	General Data Protection Regulation
HIPAA	Health Insurance Portability and Accountability Act
JSON	JavaScript Object Notation
LLM	Large Language Model
PE-BPMN	Privacy-Enhanced Business Process Model and Notation
POWL	Partially Ordered Workflow Language
RAG	Retrieval-Augmented Generation
SecBPMN	Security Business Process Model and Notation
SecBPMN-Q	Query Language for Security Policies in BPMN
STS-ml	Socio-Technical Security Modeling Language
XML	Extensible Markup Language

Table 1: Comprehensive list of acronyms used throughout the thesis.

Chapter 1

Introduction

Business Process Management (BPM) has become a cornerstone of digital transformation initiatives, enabling organizations to systematically design, execute, and optimize their operational workflows [White \(2004\)](#). Through process modeling, execution monitoring, and continuous improvement, BPM provides the foundation for aligning business operations with organizational objectives and regulatory obligations. The Business Process Model and Notation (BPMN 2.0) has emerged as the de facto standard for representing business processes, offering a visual and executable language that bridges the gap between business analysts and technical implementers [Chinosi and Trombetta \(2012\)](#). Its standardized notation facilitates interoperability and automation across heterogeneous systems, making it an indispensable tool for process-driven organizations.

However, as organizations increasingly digitalize their operations and handle sensitive information, security modeling has become an essential component of process management [Rodríguez et al. \(2007\)](#). Business processes frequently involve confidential data exchanges, access control mechanisms, and compliance requirements that must be explicitly captured at the modeling stage. Despite BPMN's expressive power for describing control flow and resource interactions, it lacks native constructs for specifying and verifying security properties such as confidentiality, integrity, availability, and accountability [Salnitri and Giorgini \(2014\)](#). This limitation hinders the integration of security considerations into the early stages of process design, resulting in ad hoc or post-hoc enforcement of security policies.

To address this shortcoming, researchers have introduced security-annotated BPMN, an extension of BPMN that supports the explicit representation of security requirements and constraints throughout the process lifecycle [Brucker et al. \(2012\)](#) [Salnitri et al. \(2015b\)](#) [Pullonen et al. \(2019\)](#). Through these annotations, process designers can specify, verify, and adapt security requirements such as confidentiality, non-repudiation,

and separation of duties alongside functional workflows. This integration ensures that security objectives are not treated as afterthoughts but are systematically embedded into process structures from the outset.

The importance of security modeling extends across multiple industry domains. In the healthcare sector, for instance, hospitals rely on SecBPMN to formalize access control rules, data-handling procedures, and auditability mechanisms, ensuring that only authorized medical personnel can access patient information and that all activities comply with privacy regulations such as HIPAA and the General Data Protection Regulation (GDPR) [Ramadan et al. \(2020\)](#). In banking and financial systems, SecBPMN has been employed to define separation-of-duty constraints and traceability requirements for ATM transaction workflows, thereby preventing insider fraud and ensuring that every financial operation is auditable [Salnitri and Giorgini \(2014\)](#). Similarly, in e-commerce, enterprises use SecBPMN to model secure payment and refund processes by enforcing role-based access, fraud detection policies, and Payment Card Industry (PCI) compliance for credit-card data handling [Salnitri et al. \(2015a\)](#). Across these sectors, SecBPMN provides a structured, model-driven means to prevent unauthorized access, ensure accountability, and maintain regulatory compliance in complex, distributed environments.

However, creating and maintaining SecBPMN annotations remains largely manual and expert-driven, requiring significant knowledge of both security engineering and process modeling. As process complexity increases, the manual specification of security constraints becomes time-consuming, error-prone, and difficult to scale across large organizations.

1.1 Research Problem and Motivation

Designing secure business processes requires the explicit representation of how confidentiality, integrity, accountability, and related protection goals are maintained throughout a workflow. Although the Business Process Model and Notation (BPMN 2.0) offers a standardized language for depicting control flow, resources, and data interactions, it lacks native constructs for security. Consequently, modelers must rely on ad hoc annotations or external documentation to express access rules and compliance obligations. This manual approach is time-consuming, error-prone, and difficult to maintain as processes evolve, which ultimately limits the systematic adoption of security-by-design practices.

To address these limitations, several security-aware BPMN extensions have been developed. Frameworks such as SecureBPMN [Brucker et al. \(2012\)](#), SecBPMN [Salnitri et al. \(2017\)](#), and PE-BPMN [Pullonen et al. \(2019\)](#) extend BPMN with constructs that

specify and verify security properties. These approaches have enhanced the expressiveness and formality of process models by enabling the definition of authorization, trust, and privacy requirements directly within the notation. Nevertheless, they still depend on the manual interpretation of textual specifications where modelers must extract security requirements from unstructured organizational documents and encode them by hand. This dependence on expert intervention prevents scalability and impedes alignment between evolving security policies and their formal representations.

Recent advances in Large Language Models (LLMs) have introduced new opportunities to automate process modeling tasks. Pioneering works on automatic BPMN generation [Kourani et al. \(2024b\)](#); [Köpke and Safan \(2024\)](#); [Nour Eldin et al. \(2024\)](#) demonstrated that natural-language process descriptions can be transformed into formal BPMN structures from unstructured input. These studies highlight the potential of LLMs to understand domain context, reason over procedural narratives, and generate syntactically valid BPMN models. However, current approaches focus primarily on control-flow reconstruction and do not capture security semantics such as role-based access control, confidentiality constraints, or separation-of-duty relations.

This gap between formal modeling and natural-language understanding reflects a broader challenge in security requirements engineering. Most organizational policies, compliance rules, and procedural documents are written in unstructured natural language, which makes it difficult to translate them into formal, verifiable constraints. Traditional rule-based or ontology-driven approaches struggle to process such text effectively due to their dependence on predefined vocabularies and limited contextual reasoning capabilities. As a result, enterprises face increasing difficulty in ensuring that their process models remain aligned with evolving regulatory and security requirements.

Large Language Models offer a promising means to bridge this gap. Their capability to infer semantic relations and generate structured representations (e.g., JSON or XML) from unstructured text enables the automatic translation of security requirements into machine-readable annotations. When combined with advanced techniques such as prompt engineering, chain-of-thought reasoning, and retrieval-augmented generation (RAG), LLMs can interpret diverse formulations of security constraints without relying on rigid grammars. Yet applying LLMs to security-aware process modeling introduces new challenges: the extracted annotations must be mapped correctly to BPMN elements, comply with the syntactic rules of SecBPMN, and maintain logical consistency across the model. Achieving this balance between semantic accuracy and structural validity calls for a hybrid approach that integrates LLM-based reasoning with deterministic, rule-based validation and schema enforcement.

Despite notable progress in both BPM and natural-language processing, no existing framework automates the extraction and mapping of security requirements from textual

process descriptions into formally valid SecBPMN models. Addressing this gap is essential for scalable, reliable, and verifiable security-by-design modeling in modern enterprises.

To address these research problems, this thesis is guided by the following questions, which operationalize the analytical focus of this study:

- RQ1:** How effectively can LLMs extract and formalize security requirements from unstructured natural-language process descriptions into structured SecBPMN annotations?
- RQ2:** How does the integration of rule-based and LLM reasoning improve the consistency, structure, and overall reliability of the generated SecBPMN models?
- RQ3:** What changes occur in accuracy and error behavior when schema-based mapping is applied after initial generation, and how does this affect the balance between precision and recall?
- RQ4:** To what extent can an LLM assume the role of a *judge*—not merely reproducing objective metrics, but aligning with human-like notions of completeness and correctness in evaluating generated annotations?
- RQ5:** How does efficiency measured through processing latency and token usage relate to the quality of generated annotations across different models and strategies?
- RQ6:** In what ways do human experts differ from automated systems in accuracy, completeness, and time efficiency, and what do these differences reveal about the complementary roles of humans and LLMs in secure process modeling?

1.2 Objectives and Scope

Building upon the research questions outlined in the section 1.1, this study defines a set of overarching objectives that guide the methodological and analytical design of the thesis. The principal aim is to advance the automation of security-aware business process modeling by establishing how Large Language Models (LLMs) can facilitate the interpretation, formalization, and validation of security requirements within the SecBPMN framework. The research is driven by the recognition that, although prior extensions of BPMN have enriched its expressiveness with security constructs, their practical deployment remains constrained by manual annotation and limited scalability. This thesis therefore seeks to bridge the gap between unstructured, natural-language security specifications and syntactically valid, semantically coherent SecBPMN models

through an integrative, hybrid approach that combines LLM-based reasoning with rule-based transformation.

To fulfill this aim, the research pursues three interrelated objectives:

1. **Conceptual Objective:** Establish a theoretical foundation for translating unstructured natural-language security requirements into structured, machine-interpretable annotations that can be seamlessly incorporated into the SecBPMN framework.
2. **Methodological Objective:** Investigate the role of LLMs as reasoning agents for automating the generation of security annotations, focusing on prompt design, semantic grounding, and schema-constrained validation to minimize human intervention while preserving accuracy and consistency.
3. **Analytical Objective:** Develop a rigorous evaluation framework that integrates quantitative metrics with qualitative, model-based assessments to evaluate the scalability, reproducibility, and reliability of LLM-assisted SecBPMN modeling.

The scope of this research is delimited to the automation of security annotation generation within existing BPMN process models, rather than the synthesis or execution of complete workflows. The investigation focuses on the primary security dimensions formalized in SecBPMN, including confidentiality, integrity, availability, authenticity, accountability, auditability, non-repudiation, non-delegation, and privacy. Broader topics such as runtime policy enforcement, process optimization, and executable workflow deployment fall outside the present scope. Empirical evaluation is conducted on a curated benchmark dataset spanning diverse domains to ensure representational diversity and reproducibility of findings.

Through these objectives and boundaries, the thesis positions itself at the intersection of business process management, security requirements engineering, and natural language processing, aiming to establish a reproducible and generalizable pathway toward scalable, secure-by-design process modeling.

1.3 Main Contributions of the Work

This thesis advances the scientific foundations of security-aware business process modeling by introducing a framework for automatic annotation of security constraints in existing Business Process Model and Notation (BPMN) diagrams. This process generates semantically enriched, security-annotated BPMN (SecBPMN) models. The research reframes secure process modeling as a structured information extraction and mapping task. In this context, Large Language Models (LLMs) serve as semantic intermediaries

between unstructured textual security requirements and formal BPMN representations. This theoretical approach introduces a hybrid reasoning paradigm that combines the contextual understanding of LLMs with the determinism of rule-based synthesis. As a result, security labels such as confidentiality, integrity, accountability, and privacy are accurately assigned to corresponding BPMN elements. This approach clarifies how foundation models can bridge the gap between natural-language policy interpretation and formal security annotation in model-driven engineering.

This thesis presents a modular pipeline that automates the transformation of standard BPMN models into SecBPMN models with systematic security annotation. The framework combines LLM-based extraction of security semantics with rule-based mapping to maintain syntactic validity and semantic coherence. This integration demonstrates that LLM reasoning can be anchored in deterministic validation, providing a scalable approach for automation in secure modeling. The research further benchmarks LLM annotation by evaluating GPT, Mistral, and Gemini models using various prompting methods. This evaluation identifies trade-offs in contextual completeness, accuracy, and computational efficiency.

To support scientific reproducibility, the thesis introduces a benchmark dataset that aligns textual process descriptions with their corresponding SecBPMN-annotated models. This dataset includes twenty-three process instances from domains such as health-care, finance, and e-commerce, providing a standardized basis for evaluating LLM-based security annotation methods. In addition, a multidimensional evaluation framework is proposed. This framework integrates quantitative metrics, including precision, recall, F1-score, Jaccard similarity, and schema validity, with qualitative assessments from rubric-based LLM-as-a-Judge evaluations. Correlation analysis between these evaluations offers methodological insights into the alignment of automated, model-based judgment with objective measures of annotation quality and completeness. This work contributes to research on evaluating structured generation and annotation tasks. The thesis demonstrates the framework's feasibility through the SecBPMN Chatbot, a prototype that allows users to input BPMN models and process descriptions for automatic, security-enriched SecBPMN outputs. The prototype validates the workflow and shows how AI-assisted modeling fits into design environments. Overall, these contributions establish a solid foundation for AI-driven secure process modeling, coupling LLM generalization with rule-based rigor for scalable, interpretable automation.

1.4 Organization and Structure

This thesis is systematically structured into chapters, each dedicated to specific aspect of the research, as outline below:

Chapter	Description
Chapter 1	<i>Introduction</i> — Establishes the research context, defines the problem of manual security annotation in BPMN, outlines the motivation, objectives, and research questions, and summarizes the main contributions.
Chapter 2	<i>Background</i> — Reviews core concepts of Business Process Management (BPM), BPMN 2.0, and SecBPMN extensions; introduces security dimensions such as confidentiality, integrity, and accountability; and discusses Large Language Models (LLMs) and related NLP concepts relevant to this study.
Chapter 3	<i>Related Work</i> — Surveys the state of the art in secure process modeling, existing BPMN security extensions, and recent LLM-based approaches for information extraction and structured generation, highlighting research gaps motivating the proposed solution.
Chapter 4	<i>Proposed Framework</i> — Describes the overall design of the hybrid LLM + rule-based pipeline, including its key components (extraction, mapping, and generation) and the strategies used for prompting, reasoning, and retrieval-augmented generation (RAG).
Chapter 5	<i>Implementation and Evaluation</i> — Details the benchmark dataset, experimental setup, and evaluation metrics; compares model families and prompting strategies; analyzes quantitative and qualitative results; and discusses insights regarding performance, validity, and computational efficiency.
Chapter 6	<i>Conclusion Limitations, and Future Work</i> — Summarizes the research outcomes, theoretical and practical contributions, limitations, and directions for future exploration of AI-assisted security-aware process modeling.

Table 1.1: Overview of the Thesis Structure

Chapter 2

Background

This chapter presents the core concepts, standards, and methodologies relevant to the research. It first outlines the fundamentals of Business Process Management and the Business Process Model and Notation (BPMN 2.0), which is the standard for modeling organizational workflows. Emphasis is placed on constructs that are most pertinent for extending models with security semantics. The discussion then shifts to security modeling in business processes, including established extensions such as SecBPMN and the principal security properties they represent. Subsequently, the chapter introduces approaches to requirement and information extraction, demonstrating how structured constraints can be derived from unstructured text. The chapter concludes with an overview of large language models, detailing their capabilities, limitations, and previous applications in process modeling.

After establishing the general need for secure and automated process modeling, the discussion turns to foundational concepts. A comprehensive understanding of Business Process Management (BPM) and its standard notation, BPMN 2.0, is essential because these elements provide the basis for subsequent extensions and automation techniques.

2.1 Business Process Management and BPMN 2.0

Business Process Management (BPM) is a discipline concerned with representing, analyzing, and improving organizational processes in a systematic and iterative way. It provides a structured approach to coordinating people, systems, and information flows to achieve business objectives more efficiently and consistently. BPM is not merely about diagramming workflows but encompasses governance, continuous improvement, compliance, and alignment of processes with organizational strategy [Chinosi and Trombetta \(2012\)](#).

To support these goals, the Business Process Model and Notation (BPMN) was intro-





duced as a unified, flowchart-based graphical notation for modeling business processes. Conceived within the Business Process Management Initiative and standardized by the OMG, BPMN was designed to be understandable by business analysts, technical developers, and end-users alike [White \(2004\)](#). Its central aim is to bridge the gap between business-level modeling and IT-level execution by providing diagrams that are both intuitive and semantically precise. Later enhancements in BPMN 2.0 further strengthened its role by formalizing execution semantics, supporting collaborations and choreographies, and offering a native XML serialization [Aagesen and Krogstie \(2014\)](#). BPMN relies on a set of core graphical categories: Flow Objects (events, activities, gateways), Connecting Objects (sequence flows, message flows, associations), Swimlanes (pools and lanes), and Artifacts (data objects, groups, annotations). These constructs allow modelers to represent not only the control-flow of activities but also responsibilities, interactions, and supporting information within a process [White \(2004\)](#); [Chinosi and Trombetta \(2012\)](#). The balance between simplicity and expressiveness explains BPMN's suitability for tasks ranging from descriptive documentation to simulation, validation, and execution across industries [Aagesen and Krogstie \(2014\)](#). BPMN has emerged as the de facto global standard for business process modeling because it provides a common language across stakeholders, ensures syntactic and semantic rigor, and enables interoperability with execution platforms such as BPEL. Its widespread adoption and tool support confirm its central role in both academic research and industrial practice [Chinosi and Trombetta \(2012\)](#); [Aagesen and Krogstie \(2014\)](#). Having established the role of BPMN as a global standard for process modeling, it is useful to examine its core elements in detail, since these building blocks provide the foundation for later extensions with security semantics.

2.1.1 Core BPMN Elements

BPMN defines four categories of elements: **Flow Objects**, **Connecting Objects**, **Swimlanes**, and **Artifacts** [White \(2004\)](#). Each category provides graphical symbols that together form the basis of process modeling notation. Below, we summarize these elements along with notation symbols (e.g., T , E , G) that will be used consistently throughout this thesis.




Flow Objects

Flow Objects define the fundamental behavior of a process, including events, activities, subprocesses, and gateways.

Event (<i>E</i>)	A state or occurrence that affects the process flow. Events can start, interrupt, or end a process and are categorized as start, intermediate, or end events.	
Task / Activity (<i>T</i>)	An atomic unit of work performed by a human, system, or service.	
Sub-Process (<i>SP</i>)	A compound activity that encapsulates a group of tasks.	
Gateway (<i>G</i>)	A decision or synchronization node that controls branching and merging of flows.	

Connecting Objects

Connecting Objects specify how flow objects are linked and how control or information moves across the process.

Sequence Flow (<i>S</i>)	Defines the execution order of activities within a process.	
Message Flow (<i>M</i>)	Represents the exchange of information between different process participants (across pools).	
Association (<i>A</i>)	Connects artifacts (e.g., data objects or annotations) to flow objects without affecting control flow.	

Swimlanes

Swimlanes partition a process model to reflect organizational or role-based responsibilities.

Pool (P) Defines an independent process participant, such as an organization or system, and establishes process boundaries.



Lane (L) A subdivision of a pool that represents roles, departments, or subsystems responsible for specific activities.



Artifacts

Artifacts provide supplemental information that clarifies a process without altering control flow.

Data Object (D) Represents data consumed or produced by activities, illustrating the information flow within the process.



Annotation (AN) Provides explanatory notes to clarify assumptions, constraints, or other model details for human readers.



To illustrate these elements in practice, Figure 2.1 shows a simple Business Process Model. The diagram captures interactions between activities, gateways, events, and annotations.

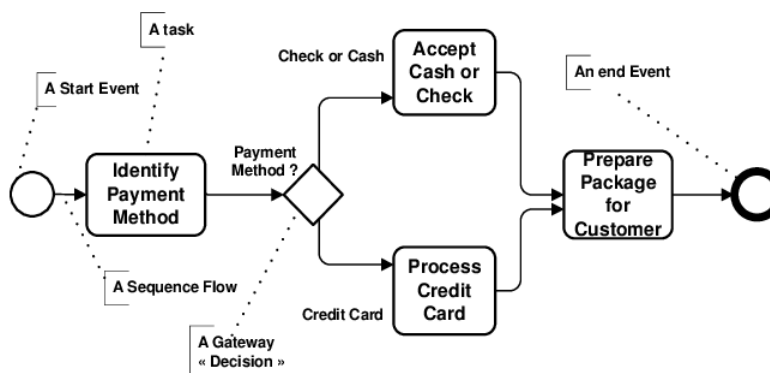


Figure 2.1: An example of a simple Business Process Model [White \(2004\)](#).

While BPMN offers a powerful and widely adopted notation for capturing control flow and organizational interactions, it deliberately omits constructs for security. This limitation makes it difficult to model protections such as confidentiality, integrity, or accountability directly within process diagrams. To overcome this gap, several extensions have been proposed, most notably SecBPMN, which introduces explicit security semantics into the BPMN framework.

2.2 BPMN Security Extensions

Business process models are increasingly applied in socio-technical systems where humans, organizations, and technical components interact. In such settings, neglecting security can have severe consequences, as processes often handle confidential information, sensitive data, and mission-critical operations. Although the standard Business Process Model and Notation (BPMN) is highly effective for modeling control flow and organizational interactions, it lacks native constructs for representing security requirements. As a result, essential concerns such as confidentiality, integrity, availability, or accountability are typically recorded informally in accompanying documentation. This practice hampers systematic verification and increases the risk of inconsistencies or incomplete protection [Salnitri et al. \(2017\)](#).

To address these limitations, the *SecBPMN* family of extensions has been developed. SecBPMN enriches BPMN with explicit security constructs that enable requirements to be directly attached to process elements such as tasks, data objects, and message flows.

It comprises two complementary components:

- **SecBPMN-ml** — a modeling language that introduces structured annotations for specifying security goals (e.g., confidentiality, integrity, privacy, non-repudiation) and associating them with BPMN elements. This ensures that process models encode not only functional behavior but also the guarantees required for secure and trustworthy execution.
- **SecBPMN-Q** — a query language that extends BPMN-Q with security-specific constructs, enabling automated compliance checking of annotated models against organizational policies and regulatory requirements. While SecBPMN-ml provides the annotation framework, SecBPMN-Q allows analysts to verify that the attached constraints collectively satisfy higher-level security objectives.

Together, SecBPMN-ml and SecBPMN-Q elevate security from an informal concern to a first-class modeling dimension. They provide a unified representation that is

both understandable to business analysts and formally analyzable by security experts. Since the focus of this thesis is on the automatic generation of security annotations, the following subsection concentrates on *SecBPMN-ml*, which serves as the foundation for embedding structured security constraints into BPMN models.


2.2.1 SecBPMN-ml: A Modeling Language for Secure Business Processes

SecBPMN-ml is the central extension of the SecBPMN framework. It introduces structured annotations that integrate security requirements directly into BPMN models. Instead of recording protections informally in accompanying documentation, modelers can attach explicit constraints to activities, data objects, and message flows. This elevates security from an external concern to a first-class modeling property [Salnitri et al. \(2017\)](#).










Security Goals

Business processes often manipulate sensitive data and mission-critical operations, making them prime targets for misuse, manipulation, or disruption. To ensure trustworthiness, SecBPMN-ml defines a standardized vocabulary of *security goals* that specify what protections must be guaranteed, independent of how they are enforced. Linking these goals directly to BPMN elements enables systematic specification, reasoning, and verification.

In this work, we adopt the comprehensive set of security aspects proposed by the Reference Model for Information Assurance and Security (RMIAS) [Cherdantseva and Hilton \(2013\)](#). This extends the classical CIA triad (confidentiality, integrity, availability) with additional goals such as accountability, auditability, authenticity, non-repudiation, privacy, and non-delegation [Salnitri et al. \(2017, 2014a\)](#). Table 2.1 summarizes these goals and their semantics.

Graphical Syntax	Security Goal	Description
	Accountability	Ensures that system users are held responsible for their actions, e.g., misuse of information.

Continued on next page

Graphical Syntax	Security Goal	Description
	Auditability	Provides persistent, non-bypassable monitoring of human or machine actions across activities, data, or message flows.
	Authenticity	Guarantees verification of identity and trust in third parties or exchanged information.
	Availability	Ensures that resources and components remain operational and accessible to authorized users when required.
	Confidentiality	Restricts access to information, ensuring only authorized users can read or modify it.
	Integrity	Maintains completeness and correctness of system components, preventing unauthorized modifications.
	Non-repudiation	Provides legally valid proof of execution or non-execution of events or actions.
	Privacy	Ensures compliance with privacy regulations and protects personal or sensitive information.
	Non-delegation	Ensures tasks are executed only by explicitly authorized users, prohibiting delegation.
	Binding of Duties	The ability of the system to require the same person to be responsible for the completion of a set of related tasks.

Continued on next page







Graphical Syntax	Security Goal	Description
	Separation of Duties	The ability of the system to force two or more different people to be responsible for the completion of a task or set of related tasks.
	Privity – Public	Allows unrestricted read access; data are publicly available to all participants or the public, ensuring maximum transparency but no confidentiality.
	Privity – Strong Dynamic	Restricts read access only to participants who are certain to execute subsequent tasks using the data, ensuring context-aware confidentiality during execution.
	Enforceability – Public	Ensures that the correctness of control flow and decisions is publicly verifiable by a broad set of blockchain nodes, typically on a public blockchain. Provides maximum transparency but limited confidentiality.
	Enforceability – Private	Restricts verification of process decisions to authorized process participants within a private or consortium blockchain, achieving confidentiality and controlled trust.
	Enforceability – User defined	Specifies a custom subset of participants responsible for verifying decisions or control flow correctness, enabling fine-grained, context-aware enforceability through endorsement or voting mechanisms.

Table 2.1: Overview of security goals in SecBPMN, with corresponding icons and short descriptions.

Formalization and Predicates

To capture security requirements rigorously, each SecBPMN annotation is represented as a tuple $\alpha = (e, g, p)$, where $e \in \mathcal{E}$ denotes a BPMN element, $g \in \mathcal{G}$ a security goal,

and $p \in \text{Params}(g)$ a goal-specific parameter set. BPMN elements are partitioned into five disjoint subsets:

$$\mathcal{E} = \mathcal{E}_{\text{Act}} \cup \mathcal{E}_{\text{DO}} \cup \mathcal{E}_{\text{MF}} \cup \mathcal{E}_{\text{GW}} \cup \mathcal{E}_{\text{ORG}},$$

representing respectively activities, data objects, message flows, gateways, and organizational associations (e.g., pools or participant relations).

A goal-element compatibility predicate $\text{compat}(g, \text{type}(e)) \in \{\text{true}, \text{false}\}$ encodes whether a given security goal g may be attached to an element type $\text{type}(e)$. The compatibility relation follows the attachment matrix defined in Table 2.3. An annotation α is schema-valid iff $\text{compat}(g, \text{type}(e))$ holds and p satisfies the well-formedness constraints defined by the corresponding predicate schema.

At the model level, a SecBPMN process is annotated through a mapping

$$\mathcal{A} : \mathcal{E} \rightarrow 2^{\mathcal{G} \times \text{Params}},$$

assigning each BPMN element the set of security goals and their parameter instances. Concretely, these tuples are operationalized by formal predicate definitions that specify how each security goal applies to its respective BPMN element type and what parameter values it requires (e.g., authorized users, enforcement mechanisms, validation frequency, or contextual conditions).

Table 2.2 lists the canonical predicate forms aligned with the SecBPMN JSON schema. These include not only traditional activity-, data-, and message-level goals such as *Accountability*, *Confidentiality*, and *Integrity*, but also gateway-level enforceability constraints (*Enforceability-Public*, *-Private*, *-User-defined*) and organizational-level separation and binding constraints (*Separation of Duties*, *Binding of Duties*). Together, these extensions enable a unified, schema-based formalization of both information-centric and structural security properties in BPMN processes.

For example, the statement “only tower and pilot may read the flight message” can be expressed as

$$\begin{aligned} \alpha &= (e_{\text{msg}}, \text{CONFIDENTIALITY}, p), \\ p &= \{\text{type} = \text{MF}, \text{readers} = \{\text{tower}, \text{pilot}\}, \text{writers} = \{\text{tower}\}\}, \end{aligned}$$

which corresponds to the predicate **ConfidentialityMF** in Table 2.2. This unified formalization makes constraints machine-readable, enabling automated reasoning and verification.

Security Constraints	Predicates / Parameters
Accountability	sec:AccountabilityAct(enfBy:String[], monitored:String[])
Auditability	sec:AuditabilityAct(enfBy:String[], frequency:String) sec:AuditabilityDO(enfBy:String[], frequency:String) sec:AuditabilityMF(enfBy:String[], frequency:String)
Authenticity	sec:AuthenticityAct(enfBy:String[], ident:Boolean, auth:Boolean, trustValue:Real) sec:AuthenticityDO(enfBy:String[])
Availability	sec:AvailabilityAct(enfBy:String[], level:Real) sec:AvailabilityDO(enfBy:String[], authUsers:String[], level:Real) sec:AvailabilityMF(enfBy:String[], level:Real)
Confidentiality	sec:ConfidentialityDO(enfBy:String[], readers:String[], writers:String[]) sec:ConfidentialityMF(enfBy:String[], readers:String[], writers:String[])
Integrity	sec:IntegrityAct(enfBy:String[], personnel:Boolean, hardware:Boolean, software:Boolean) sec:IntegrityDO(enfBy:String[]) sec:IntegrityMF(enfBy:String[])
Non-repudiation	sec:NonRepudAct(enfBy:String[], execution:Boolean) sec:NonRepudMF(enfBy:String[], execution:Boolean)
Privacy	sec:PrivacyAct(enfBy:String[], sensitiveInfo:String[]) sec:PrivacyDO(enfBy:String[], sensitiveInfo:String[])
Non-delegation	sec:NonDelegationAct(enfBy:String[])

Continued on next page

Security Constraints	Con-	Predicates / Parameters
Enforceability Public	–	sec:EnforcePublicGW(enfBy:String[], verifiable:Boolean, conditionType:String)
Enforceability Private	–	sec:EnforcePrivateGW(enfBy:String[], verifiable:Boolean, conditionType:String)
Enforceability User-defined	–	sec:EnforceUserDefinedGW(enfBy:String[], verifiable:Boolean, conditionType:String)
Privity – Public		sec:PrivityPublicDO(enfBy:String[], authorizedRole:String[]) sec:PrivityPublicMF(enfBy:String[], authorizedRoles:String[])
Privity – Strong Dynamic		sec:PrivityStrongDynamicDO(enfBy:String[], authorizedRole:String[]) sec:PrivityStrongDynamicMF(enfBy:String[], authorizedRoles:String[])
Separation of Duties (ORG)	of	sec:SeparationOfDutiesORG(fromPool:String, toPool:String, enfBy:String[], reason:String, strict:Boolean)
Binding of Duties (ORG)		sec:BindingOfDutiesORG(fromPool:String, toPool:String, enfBy:String[], context:String, strict:Boolean)

Table 2.2: Formal predicates associated with each security goal, grouped by the annotated BPMN element type..

Attachment to BPMN Elements

Finally, Table 2.3 summarizes the mapping between security goals and BPMN element types. It shows at a glance which goals apply to activities, data objects, or message flows. For example, confidentiality is only meaningful for data and communication, whereas accountability and non-delegation apply exclusively to activities.

Security Goal	Activity	Data Object	Message Flow	Gateway	Org. Association
Accountability	✓	✗	✗	✗	✗
Auditability	✓	✓	✓	✗	✗
Authenticity	✓	✓	✗	✗	✗
Availability	✓	✓	✓	✗	✗
Confidentiality	✗	✓	✓	✗	✗
Integrity	✓	✓	✓	✗	✗
Non-repudiation	✓	✗	✓	✗	✗
Privacy	✓	✓	✗	✗	✗
Non-delegation	✓	✗	✗	✗	✗
Enforceability – Public	✗	✗	✗	✓	✗
Enforceability – Private	✗	✗	✗	✓	✗
Enforceability – User-defined	✗	✗	✗	✓	✗
Privity – Public	✗	✓	✓	✗	✗
Privity – Strong Dynamic	✗	✓	✓	✗	✗
Binding of Duties	✗	✗	✗	✗	✓
Separation of Duties	✗	✗	✗	✗	✓

Table 2.3: Mapping of SecBPMN security goals to applicable BPMN element types. (✓: applicable, ✗: not applicable).

To better understand how security goals and their formal predicates operate in practice, we turn to the *Route Based Trajectory (RBT) negotiation process model* introduced in [Salnitri et al. \(2017\)](#). This model represents the interaction between pilots and the control tower to propose, evaluate, and finalize flight trajectories. Because this process manipulates highly sensitive information (such as flight plans and personal data), it provides an ideal case study for demonstrating why and how security annotations must be integrated into BPMN.

The figure shows how each stage of the negotiation is enriched with explicit security constraints, attached as SecBPMN predicates to activities, data objects, and message flows. Below, we explain each security goal in the context of this model.

- **Accountability** — applied to approval tasks such as *Approve destination* and *Accept RBT*. Predicate: `AccountabilityAct(a, enfBy:{SecMechanisms}, monitored:{Users})`. Here, the model specifies that junior pilots must be monitored when carrying out these tasks, with RBAC used as the enforcement mechanism.

This ensures that if misuse occurs (e.g., approving a wrong destination), the

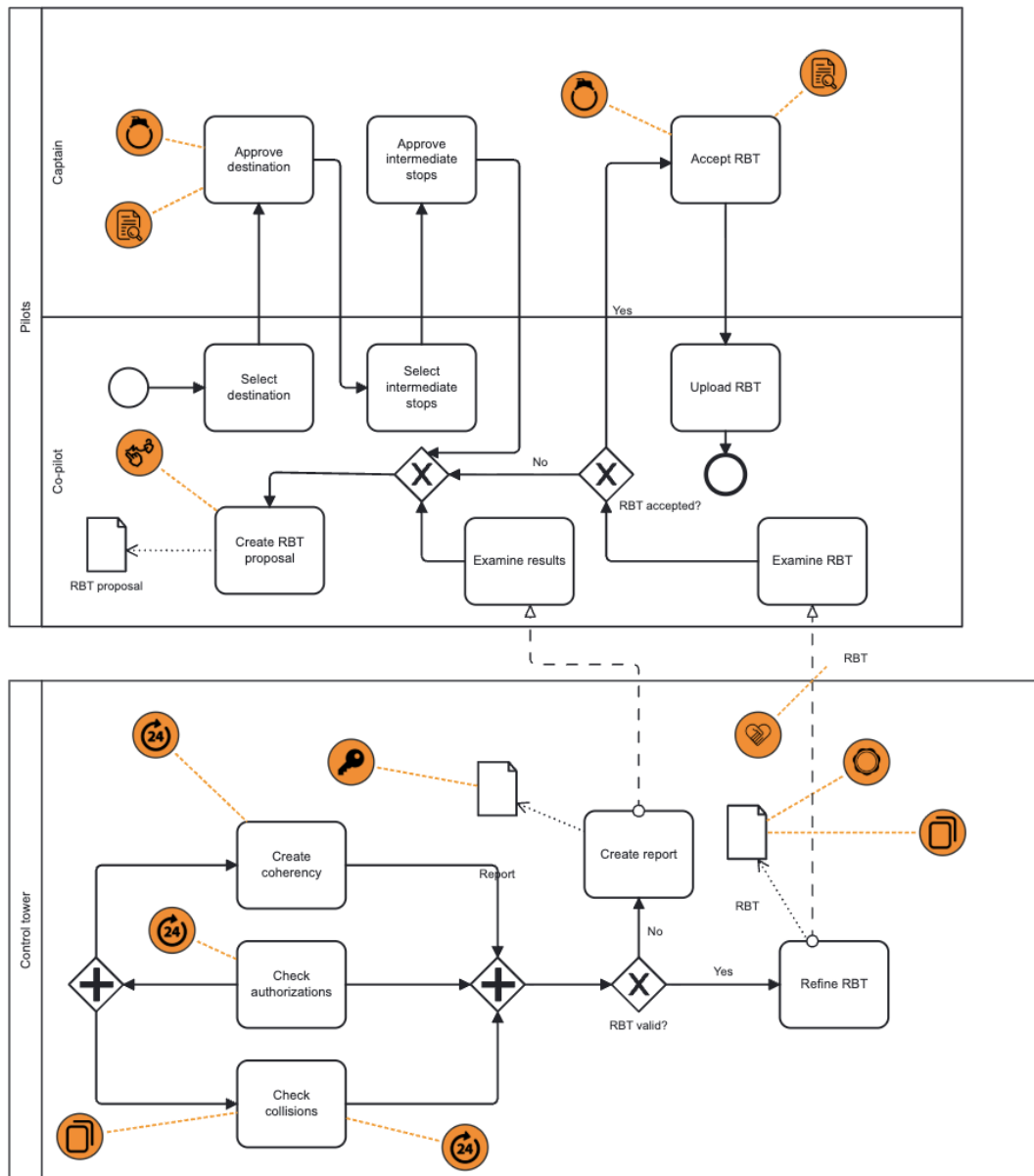


Figure 2.2: RBT negotiation process model annotated with security constraints [Salnitri et al. \(2017\)](#).

responsible individual can be traced. Without this annotation, responsibility

could remain ambiguous.

- **Auditability** — also attached to approval of destination and acceptance of RBT. For example, `AuditabilityAct(..., frequency:10d)` records actions at regular intervals (every 10 days in this case).

This makes it possible to verify later who performed which operations, or detect unusual behavior in communication and data handling. The model thus ensures that manipulations of the RBT proposal are never invisible or bypassed.

- **Authenticity** — attached to the *RBT* data object exchanged between pilots and the control tower. Predicate: `AuthenticityDO(do, enfBy:{TLS, X.509})`.

This guarantees that the trajectory file comes from a legitimate source and has not been replaced with a forged version. In the context of air traffic management, this is vital: an unauthenticated RBT could endanger safety if introduced by an attacker. For activities, the related predicate `AuthenticityAct` ensures that only properly authenticated pilots can perform negotiation steps.

- **Availability** — attached to activities such as *Create coherency*, *Check authorizations*, and *Check collisions*. Predicate: `AvailabilityAct(a, ..., level:99.5)`.

By requiring 99.5% uptime, the model ensures these critical operations are not disrupted. Imagine if the *Check collisions* step were unavailable during peak hours: flight safety would be compromised. Availability annotations ensure reliability under operational conditions.

- **Confidentiality** — attached to the message flow from *Refine RBT* to *Examine RBT*. Predicate: `ConfidentialityMF(mf, enfBy:{TLS, RBAC}, readers:{towerControl, controlAuthority, RBTowner}, writers:{towerControl, RBTowner})`.

This restricts who can send and receive the trajectory information. Without this constraint, unauthorized actors might eavesdrop on or tamper with flight plans.

- **Integrity** — applied to the *RBT* data object and the *Check collisions* activity. Predicates: `IntegrityDO`, `IntegrityAct`, `IntegrityMF`. For instance, `IntegrityAct(..., hardware:true, software:true)` protects against corruption at both hardware and software levels, ensuring that collision detection results cannot be falsified. On data objects, integrity ensures that any modification of the RBT proposal is legitimate and attributable.

- **Non-repudiation** — applied to the task *Create RBT proposal*. Predicate: `NonRepudAct(a, enfBy:{SecMechanisms}, execution:false)`.

This annotation ensures that proof of non-execution is kept. Pilots cannot later deny that they failed to create a trajectory proposal. This is crucial for accountability in case of disputes or investigations.

- **Privacy** — applied to the *Report* data object produced by the control tower. Predicate: `PrivacyDO(do, enfBy:{SecMechanisms}, sensitiveInfo:{name, surname, dateOfBirth})`.

This protects personal information contained in reports. In aviation, personal details about pilots or passengers must remain private and under the control of the individuals concerned.

The RBT negotiation model highlights why adding security predicates to BPMN is essential. By encoding security as predicates with parameters on concrete BPMN elements, the model turns informal expectations (“keep RBT confidential”) into machine-checkable rules (e.g., `ConfidentialityMF(..., readers, writers, enfBy)`). This lets engineers verify that the RBT exchange is restricted, the RBT object is authentic and intact, critical analyses remain available, actors are accountable/auditable, and personal data in reports is protected—in the model, not only in prose. This alignment with SecBPMN-Q supports automated compliance checking between the business process and policy models. Without these annotations, BPMN diagrams would remain limited to functional logic, unable to capture the protections that real-world processes require. By embedding security directly in the process model, SecBPMN makes workflows not just executable but also auditable, trustworthy, and compliant with organizational and regulatory requirements.

2.3 Information Extraction and Security Requirement Extraction

2.3.1 Overview of Information Extraction

Information Extraction (IE) in natural language processing refers to the automatic transformation of unstructured text into structured representations that machines can process and analyze. Classical information extraction typically produces three complementary outputs: *entities*, which denote mentions of objects such as people, organizations, or process elements; *relations*, which capture semantic links between entities (e.g., that a task requires confidentiality of a data object); and *events*, which describe

more complex interactions involving multiple entities and relations anchored in time or context. Together, these components form the basis for building knowledge bases, ontologies, or executable models from natural language [Singh \(2018\)](#).

Most information extraction systems follow a pipeline architecture. Preprocessing prepares the text through tokenization, sentence segmentation, and shallow linguistic analysis. Extraction then identifies entities, classifies relations, or detects event triggers using rules, statistical models, or neural architectures. Finally, a structuring step converts the extracted items into machine-readable forms such as schemas, knowledge graphs, or JSON/XML annotations. This modular design has proven effective across domains: in clinical NLP, it extracts diagnoses and treatments from health records; in legal NLP, it identifies obligations and rights in contracts; and in cybersecurity, it highlights threats, vulnerabilities, or compliance requirements [Liu et al. \(2022\)](#). Each application relies on domain-specific vocabularies and ontologies to ensure semantic precision and interpretability.

Despite its successes, this architecture exhibits well-known limitations. Rule-based and statistical components are brittle when faced with noisy or heterogeneous text, and cross-sentence reasoning remains challenging [Zheng et al. \(2023\)](#). Moreover, adapting pipelines to new domains requires extensive manual engineering of schemas and rules, which hinders scalability. These limitations have motivated a shift toward more flexible approaches with large language models (LLMs). Rather than treating extraction and structuring as separate stages, modern methods increasingly generate structured annotations directly from text, collapsing the pipeline into a single generative step [Xu et al. \(2024\)](#). This transition provides the foundation for structured output generation with LLMs, where free-text requirements can be mapped into machine-readable constraints in one pass.

2.3.2 Information Extraction to Structured Output with LLMs

Traditional information extraction approaches separate extraction from structuring: first, entities, relations, or events are detected, and only afterward are they formatted into structured representations. Large language models (LLMs) enable a different paradigm. Through careful prompting, an LLM can generate structured outputs directly in formats such as JSON, XML, or tabular schemas. This collapses the pipeline into a single step, reducing the need for post-processing and allowing greater flexibility when the target schema is not fixed in advance.

The practical importance of structured outputs is widely recognized. A recent survey of industry practitioners found that developers consistently need ways to constrain model outputs to ensure reliability. Tools such as *ConstraintMaker* demonstrate this demand by allowing users to specify structural requirements—ranging from JSON keys to list

formats and regular expressions—that guide LLMs toward valid outputs Liu et al. (2024b). Structured generation is therefore not merely a theoretical advance but a prerequisite for deploying LLMs in safety- or compliance-critical settings.

Benchmark studies further highlight both the potential and the challenges of this paradigm. The *SoEval* benchmark systematically assessed the ability of LLMs to produce structured outputs across multiple formats, finding frequent issues with schema fidelity such as missing fields or malformed structures Liu et al. (2024a). The more recent *StructEval* framework expanded this analysis to twenty-one formats and forty-four task types, combining syntax checks with visual quality assurance to measure output correctness Yang et al. (2025). These results demonstrate that while LLMs are capable of producing complex structured representations, achieving consistent adherence to strict schemas remains difficult.

For this thesis, these findings are highly relevant. Direct generation of structured annotations opens the possibility of producing SecBPMN-style security constraints from natural language descriptions. At the same time, the challenges of schema fidelity, hallucination, and faithfulness underline the need for careful design. Addressing these issues requires complementary strategies such as prompt engineering and retrieval-augmented generation, which are examined in the following subsections.

2.3.2.1 Prompt-based Strategies for Structured Extraction

Structured output generation with large language models (LLMs) requires more than simply requesting an answer: the model must be guided to respect a schema, map free-text descriptions to formal labels, and perform subtle reasoning steps. To achieve this, researchers employ prompting strategies, which frame the task in different ways to balance generalization, guidance, and reasoning. The central motivation behind these strategies is to steer models away from open-ended text generation and toward outputs that are syntactically valid and semantically consistent with a target schema such as SecBPMN. A useful starting point is *role-based prompting*, where the model is instructed to act as a domain specialist (e.g., “You are a security labels extractor”). This role specification narrows the model’s scope, signals the required expertise, and improves adherence to schema-constrained outputs Kong et al. (2023). Building on this, different prompting strategies provide varying levels of task guidance.

Zero-shot prompting: The model receives only an instruction describing the task, without examples. This setting illustrates the ability of large models to generalize from pretraining Radford et al. (2019); Brown et al. (2020). For structured extraction, zero-shot prompting evaluates whether the model can map descriptions into schema-compliant annotations using only instructions and role framing.

Instruction: Extract security constraints and return valid JSON.

Input: "Only authorized staff may approve the request."

Output: { "type": "nonDelegationAct" }

One-shot prompting: A single worked example precedes the task, showing the mapping from text to structured output [Brown et al. \(2020\)](#). This anchors the schema and output format, enabling the model to imitate the example in new cases.

Example:

Input: "Actions must be logged."

Output: { "type": "auditabilityAct" }

Task: Extract from new description.

Input: "Manager signs the document."

Output: { "type": "nonRepudAct" }

Few-shot prompting: Several examples are provided, covering diverse constraint types. This enables the model to better infer the schema and apply it consistently [Brown et al. \(2020\)](#). Few-shot prompting is particularly useful for SecBPMN, where heterogeneous security goals (e.g., confidentiality, availability, auditability) must be expressed within a unified schema.

Example 1: "Data must remain confidential."

→ { "type": "confidentialityDO" }

Example 2: "Server must be available."

→ { "type": "availabilityAct" }

New input: "All approvals must be logged."

Output: { "type": "auditabilityAct" }

Chain-of-Thought (CoT) prompting: Prompts include explicit reasoning steps, encouraging the model to explain its decision process before producing the final output [Wei et al. \(2022\)](#). For structured extraction, CoT improves reliability when selecting the appropriate BPMN element and determining the correct security type.

Input: "Messages must be encrypted so only pilot and tower can read them."

Reasoning: Confidentiality \rightarrow applies to Message Flow
 \rightarrow type = confidentialityMF.

Output: { "type": "confidentialityMF" }

Hybrid prompting: This strategy combines the breadth of few-shot examples with the reasoning scaffolding of CoT. Each example shows both the reasoning and the final structured output [Ma et al. \(2023\)](#). Hybrid prompting improves both schema fidelity and reasoning accuracy, particularly for complex annotations that involve multiple security properties.

Example:

Input: "Only authorized clerks may check availability."

Reasoning: Activity \rightarrow requires non-delegation.

Output: { "type": "nonDelegationAct" }

New input: "Pilot must prove task execution."

Output: { "type": "nonRepudAct" }

While prompt-based strategies enhance schema adherence and reasoning quality, they also have clear limitations. LLMs may still hallucinate fields, misclassify element types, or misapply labels, especially when domain-specific terminology is involved. Furthermore, prompting relies entirely on the model's internal knowledge, which may not cover specialized concepts or regulatory vocabularies. These shortcomings highlight the need for integrating external knowledge sources. This motivates the adoption of retrieval-augmented generation (RAG), which enriches LLM outputs with domain-specific examples and references to reduce hallucinations and improve accuracy.

2.3.3 Retrieval-Augmented Generation (RAG)

Prompt-based strategies improve the ability of large language models (LLMs) to produce schema-compliant outputs, yet they remain fundamentally constrained by the knowledge encoded in their parameters. This often leads to hallucinations, schema violations, or omissions when the required information lies outside the model's pretraining distribution. Retrieval-Augmented Generation (RAG) addresses these shortcomings by coupling external knowledge retrieval with generative reasoning. In RAG, the model retrieves relevant passages, examples, or structured records from a curated corpus and integrates them into its prompt context [Lewis et al. \(2020\)](#). Conditioning generation on this domain-specific evidence improves both factual accuracy and schema fidelity.

For structured extraction tasks, this mechanism is particularly valuable. By grounding generation in repositories such as annotated BPMN models or security taxonomies, RAG reduces hallucinations and ensures adherence to predefined schemas. Empirical studies confirm that retrieval improves the reliability of structured outputs by aligning LLM predictions with verified external knowledge bases, significantly lowering the risk of invalid or incomplete annotations [Béchar and Ayala \(2024\)](#). This makes RAG especially well-suited for translating unstructured security requirements into SecBPMN-style annotations, where both correctness and faithfulness are critical.

Consider a process description stating:

“Only the control tower and pilot may read the flight message.”

In a prompt-only setting, the model may correctly detect that this expresses a confidentiality requirement but generate an incomplete or inconsistently formatted output:

```
{
  "type": "confidentiality",
  "actors": ["control tower","pilot"]
}
```

Here the intent is captured, but the schema is violated: the annotation type should be tied to a specific BPMN element (e.g., `confidentialityMF`), and the field must be expressed as `readers[]` rather than a generic `actors[]` list.

With retrieval, schema documentation is injected into the context, reminding the model of the proper attachment site and required fields:

```
- confidentialityMF applies to Message Flow
- required fields: readers[], writers[]
```

Conditioning on this evidence, the model produces a grounded and schema-compliant output:

```
{
  "type": "confidentialityMF",
  "readers": ["control tower","pilot"],
  "writers": ["control tower"]
}
```

This example illustrates how RAG does not invent new information but instead normalizes outputs against domain knowledge, ensuring consistency with the target schema. By integrating retrieval with generation, RAG provides a more reliable foundation for structured extraction in specialized domains. Its ability to combine generative flexibility with external knowledge grounding makes it an essential technique for scaling secure process modeling to realistic enterprise settings.

Having established how prompting and retrieval strategies enable LLMs to generate structured annotations, the natural next step is to examine whether these models can go further namely, producing complete BPMN diagrams directly from text. Recent research in this direction offers important context for the contributions of this thesis.

The motivation for our proposed methodology is clear, manual process modeling remains labor-intensive, requires specialized expertise, and is often prone to inconsistencies across analysts. By contrast, LLMs offer the potential to bridge the gap between informal textual descriptions and formal process models, producing diagrams that are both machine-readable and understandable to non-experts. If effective, such systems could democratize process modeling by lowering the technical barriers for stakeholders while accelerating the design of executable models.

The current state of the art demonstrates both the feasibility of automated process modeling and the existing gaps. BPMN is a widely adopted modeling notation, but does not include native constructs for representing security requirements. SecBPMN extends BPMN with security constructs; however, manual annotation using SecBPMN is complex and prone to errors. LLM-based structured generation offers potential for automating process modeling, but existing approaches have not addressed the integration of security features. These identified gaps provide the rationale for the pipeline introduced in the following chapter, which automates the generation of SecBPMN annotations from natural-language process descriptions.

Chapter 3

Related Work

This chapter reviews four complementary strands of prior research that together define the foundation of this work. The first examines security-aware extensions of BPMN, which introduce formal constructs for modeling security and privacy concerns within process diagrams. The second focuses on lifecycle and compliance methodologies, which aim to ensure traceability and alignment between organizational requirements and process implementations. The third explores LLM-based approaches for BPMN generation and analysis, where generative models automate process discovery and structural synthesis from natural language descriptions. The fourth covers requirement extraction and structured-output evaluation, encompassing methods that transform textual policies or domain narratives into machine-readable representations with verifiable structure. While each of these lines of research contributes valuable insights, none provides an integrated framework that automatically derives, represents, and validates security annotations from natural language. Bridging this gap, the present thesis introduces an AI-assisted pipeline that combines LLM-based extraction, ontology-grounded representation, and compliance validation to advance secure and intelligent process modeling.

3.1 Security-Aware Extensions of BPMN

White introduced BPMN as a unified graphical language for business processes, emphasizing readability for both analysts and developers [White \(2004\)](#). While foundational, the language provided no constructs for non-functional requirements. Chinosi and Trombetta offered one of the first comprehensive assessments of BPMN 2.0. They clarified its semantics but also highlighted the lack of support for cross-cutting concerns such as security [Chinosi and Trombetta \(2012\)](#). Aagesen and Krogstie further contextualized BPMN 2.0 within the broader BPM landscape, showing that standardization ensured adoption but also reinforced the absence of explicit mechanisms for

data protection or compliance [Aagesen and Krogstie \(2014\)](#).

To address these shortcomings, researchers proposed extensions targeting specific aspects of security. Brucker et al. introduced SecureBPMN, which integrates access control into process models and supports explicit representation of separation and binding of duties [Brucker et al. \(2012\)](#). However, SecureBPMN focuses narrowly on role-based authorization and requires manual specification of protected tasks and actors. Pullonen et al. proposed Privacy-Enhanced BPMN (PE-BPMN), incorporating constructs for personal-data handling and PII leakage analysis, thereby enabling privacy-by-design assessments [Pullonen et al. \(2019\)](#). Yet PE-BPMN is privacy-specific and does not generalize to broader security requirements such as integrity or availability. Awad introduced BPMN-Q, a query language for detecting structural and semantic patterns in BPMN models, enabling pattern-based verification against design rules [Awad \(2007\)](#). Awad and Sakr subsequently optimized BPMN-Q evaluation for efficiency, scaling analysis to larger model repositories [Awad and Sakr \(2012\)](#). BPMN-Q supports verification but presumes that annotated models already exist. A more comprehensive framework emerged with SecBPMN. Salnitri et al. demonstrated a model-driven transformation from socio-technical requirements in STS-ml into enforceable SecBPMN annotations, thereby bridging organizational goals with process-level semantics [Salnitri et al. \(2014a\)](#). SecBPMN2 extended this work by introducing a modeling language (SecBPMN-ml), which enriches BPMN with constructs for confidentiality, availability, accountability, and duty constraints, and a query language (SecBPMN-Q) for compliance validation [Salnitri et al. \(2017\)](#). SecBPMN-Q operates by matching query graphs to process graphs, enabling designers to check whether annotated processes satisfy intended security policies at design time. Validation in safety-critical domains such as air-traffic management showed how SecBPMN queries could reveal violations in annotated models [Salnitri and Giorgini \(2014\)](#). This demonstrates effectiveness once annotations exist, but does not automate their derivation from natural-language descriptions.

Conceptual frameworks further systematized BPMN security research. Maines et al. proposed a cybersecurity ontology mapping fundamental properties—confidentiality, integrity, access control—to BPMN elements, thereby improving terminological consistency across extensions [Maines et al. \(2015\)](#). Nake developed a taxonomy of 18 BPMN security extensions, classifying them by supported goals, extended elements, and risk orientation, and exposing fragmentation across approaches [Nake \(2023\)](#). Cherdantseva and Hilton introduced a reference model of information assurance and security, systematizing relationships between security properties, functions, and assurance processes to provide a neutral vocabulary for aligning with BPMN security goals [Cherdantseva and Hilton \(2013\)](#). These contributions improved clarity but remained descriptive: they standardized terminology without offering methods for automated annotation.

Lifecycle-oriented works focused on compliance preservation. Salnitri et al. coupled STS-ml with SecBPMN to ensure consistency between organizational requirements and process models before and after deployment, providing iterative conformance analysis [Salnitri et al. \(2014b\)](#). The approach, however, presupposes that both requirement models and process annotations already exist. Ramadan et al. extended SecBPMN with data-minimization constructs such as anonymity and unlinkability, and encoded anti-patterns to detect conflicts like accountability versus anonymity in healthcare settings [Ramadan et al. \(2018\)](#). In later work, they incorporated fairness requirements and validated reusable conflict patterns via user studies [Ramadan et al. \(2020\)](#). While effective for design-time checks, these methods are applied after annotation and do not help produce the annotations themselves.

Security-aware BPMN has also been connected to specialized analysis tools. Hacks et al. mapped BPMN constructs to a meta-attack language (MAL/coreLang), enabling automated attack simulations that identified potential vulnerabilities without altering original models [Hacks et al. \(2021\)](#). This mapping supports risk analysis but depends on the availability and correctness of underlying annotated models. Meroni et al. presented SecBPMN2BC, an online editor that integrates blockchain-related concepts such as enforceability and privacy, offering real-time guidance on which process parts suit distributed execution [Meroni et al. \(2025\)](#). The tool improves design-time decision support for distributed execution, yet it does not address how security annotations are generated.

Collectively, these security-oriented extensions diversify BPMN’s capacity to capture non-functional requirements along three principal dimensions: (i) the range of supported protection goals (e.g., confidentiality, integrity, privacy, availability), (ii) the formality of specification and verification mechanisms, and (iii) the degree of automation in generating security annotations. Despite substantial progress in extending BPMN with formal security constructs and verification methods, existing frameworks continue to rely on manually authored annotations and presuppose that security requirements are already modeled. The present research addresses this gap by introducing an AI-assisted approach that automatically extracts and maps security constraints from natural-language process descriptions onto existing BPMN models, thereby transforming them into SecBPMN-compliant representations.

3.2 Large Language Models for BPMN Generation and Analysis

Although this thesis does not focus on the generation of BPMN models, research on automating process modeling provides essential context for understanding how Large

Language Models (LLMs) can facilitate structured reasoning in the Business Process Management (BPM) domain. These studies demonstrate the technical feasibility of using natural-language input to produce structured process representations, establishing a foundation for subsequent advances in compliance and security-aware modeling. By reviewing this line of work, this section clarifies both the capabilities and the limitations of existing LLM-based approaches, motivating the need to extend automation beyond control-flow generation toward the semantic enrichment of process models with security annotations.

Early attempts to automate BPMN construction relied on natural-language processing (NLP) techniques to reduce human involvement in model creation. Friedrich et al. developed one of the first end-to-end pipelines that combined syntactic parsing, semantic role labeling, and rule-based control-flow extraction to generate BPMN models from textual process descriptions [Friedrich et al. \(2011\)](#). While their system demonstrated feasibility, it relied on handcrafted linguistic rules and struggled to generalize across domains. Ivanchikj et al. later introduced *BPMN Sketch Miner*, an interactive web-based tool that transformed semi-structured textual input into BPMN diagrams in real time, enabling non-experts to prototype processes during workshops [Ivanchikj et al. \(2020\)](#). The tool achieved strong usability but was constrained by a simplified grammar and limited expressiveness for complex workflows. More recently, Licardo et al. proposed a hybrid NLP pipeline combining coreference resolution, BERT-based task extraction, and LLM prompting to map sentences into hierarchical BPMN constructs, achieving improved structural accuracy while still depending on well-formed text and limited BPMN coverage [Licardo et al. \(2024\)](#).

These studies collectively illustrate the evolution from rule-based NLP systems to LLM-assisted automation, highlighting progress in process synthesis but also persistent limitations in linguistic ambiguity, domain generalization, and semantic completeness. Most importantly, they underscore a key boundary: existing methods primarily reconstruct *control-flow* structures and rarely address *non-functional semantics* such as security, privacy, or compliance. This limitation motivates the broader research question of how LLMs can be adapted beyond structural generation to handle the semantic and regulatory dimensions of process modeling.

Vidgof et al. conceptualized LLMs as general-purpose enablers across the BPM lifecycle, spanning process discovery, compliance analysis, and documentation assistance [Vidgof et al. \(2023\)](#). Their work identified schema adherence and output evaluation as open challenges but remained primarily conceptual. Kourani et al. introduced one of the first operational LLM-based pipelines that translated textual descriptions into intermediate process representations (POWL) before exporting them to BPMN or Petri nets [Kourani et al. \(2024a\)](#). This two-stage architecture ensured structural correctness and demonstrated, through extensive benchmarking on sixteen models, that prompt

engineering and self-correction strategies improve fidelity. A subsequent study benchmarked LLMs for BPMN tasks and found that iterative refinement enhanced syntactic accuracy but still failed to address domain-specific semantics such as security [Kourani et al. \(2025\)](#). To date, no study has shown how LLM pipelines can jointly perform control-flow generation and the derivation of security annotations aligned with SecBPMN, precisely the conceptual gap this thesis seeks to address.

Complementary studies have emphasized usability and interactivity over semantic depth. Köpke and Safan proposed a conversational framework where an LLM incrementally refines BPMN models through user dialogue, improving accessibility but omitting compliance considerations [Köpke and Safan \(2024\)](#). Nour Eldin et al. developed *Nala2BPMN*, an end-to-end generator that produces XML-compliant BPMN diagrams directly from natural language, reducing modeling effort but offering limited semantic validation and no support for security constraints [Nour Eldin et al. \(2024\)](#). Wenger et al. improved stylistic consistency in generated models, while Hörner introduced *BPMNGen* to enable bidirectional text-to-model and model-to-text transformations, broadening accessibility but remaining focused on syntax and readability [Wenger et al. \(2024\)](#); [Hörner \(2025\)](#). Collectively, these works demonstrate that while LLMs have improved the usability and efficiency of BPMN modeling, they remain largely disconnected from compliance or security reasoning.

Beyond model generation, several studies have expanded the application of LLMs to process analysis and related automation tasks. Toxtli and Li evaluated GPT-4o for robotic process automation workflows, reporting strong performance for simple processes but scalability limitations for complex delegation [Toxtli and Li \(2024\)](#). Nousias et al. applied LLMs to decouple decision logic from gateways, enhancing maintainability while overlooking auditability and policy compliance [Nousias et al. \(2025\)](#). Voelter et al. combined multimodal reasoning and process mining, leveraging LLMs and computer vision to extract process elements from textual and diagrammatic sources, though domain-specific tuning remained necessary [Voelter et al. \(2024\)](#). De Michele et al. used structured prompting for value-added analysis, outperforming zero-shot baselines but revealing subjectivity in evaluation labels [De Michele et al. \(2025\)](#). Stiehle et al. linked BPMN to blockchain execution through smart-contract generation, assuming that non-functional requirements had already been modeled [Stiehle et al. \(2025\)](#).

It demonstrates that LLMs have significantly expanded the automation frontier in BPMN modeling, improving structural accuracy, usability, and multimodal reasoning. However, these advances remain confined to the syntactic layer of process representation and do not address the semantic enrichment required for compliance or security-aware modeling. This limitation defines the methodological boundary that the present thesis seeks to cross. We therefore examines how techniques for requirement extraction and structured-output evaluation, largely developed in domains such as healthcare,

compliance, and policy modeling can inform the automated derivation and validation of security annotations in BPMN.

3.3 Requirement Extraction and Evaluation

Automating secure BPMN requires not only generating process structures but also extracting structured, security-relevant information from unstructured sources and evaluating outputs with precision. This dual challenge: requirement extraction and structured-output validation, has been explored across domains such as clinical NLP, healthcare compliance, and regulatory modeling.

Van Woensel and Motie conducted a systematic review of process extraction methods, highlighting enduring obstacles: the lack of publicly available benchmarks, incomplete coverage of real-world process complexity, and the difficulty of synthesizing executable BPMN models directly from text [Van Woensel and Motie \(2024\)](#). Their findings underscore that despite decades of NLP and rule-based efforts, robust, reproducible benchmarks for requirement-to-model pipelines remain scarce.

Domain-specific pipelines have begun to address these gaps. Wiest et al. introduced LLM-AIx, an open-source, privacy-preserving pipeline for extracting structured entities from unstructured medical reports [Wiest et al. \(2024\)](#). The framework runs locally within hospital infrastructures, avoiding data transfer to external servers, and applies a four-stage process: data preparation, preprocessing, LLM-driven entity extraction, and automated evaluation. Demonstrations on pathology reports and emergency records showed reliable extraction of entities such as TNM cancer staging and symptoms, while maintaining compliance with strict privacy constraints. However, the pipeline remains domain-specific and its evaluation focuses on entity-level accuracy rather than higher-order process semantics.

Parallel efforts in text generation illustrate complementary insights. Boulanger et al. compared encoder-decoder and decoder-only models for the controlled generation of synthetic clinical cases conditioned on patient demographics and features [Boulanger et al. \(2024\)](#). Their study demonstrated that encoder-decoder models offer superior controllability at higher computational cost, while decoder-only models scale more efficiently but require more careful prompt engineering. Although the goal was narrative quality rather than structured requirement extraction, the results highlight the tension between controllability and efficiency an issue also central to SecBPMN-oriented annotation.

At the intersection of regulation and automation, Siavvas et al. applied LLMs to machine-readable compliance, extracting requirements from standards such as ISO/IEC 27001 and NIST SP 800-53 into structured policies [Siavvas et al. \(2024\)](#).

Their work illustrates the feasibility of “compliance-as-code,” where unstructured legal text is formalized into security controls. Yet, these controls are not linked to process-level artifacts such as tasks, gateways, or data objects, leaving a gap between high-level policies and actionable SecBPMN annotations.

Evaluation frameworks are equally critical. Liu et al. benchmarked LLM outputs across structured formats such as JSON, YAML, and HTML, revealing large performance variance and demonstrating that schema-constrained prompting significantly improves reliability Liu et al. (2024b). Gu et al. surveyed LLM-as-a-judge techniques, finding that automated evaluation offers scalability but risks bias if not combined with rule-based or statistical metrics Gu et al. (2024). Together, these studies emphasize that reliable structured-output pipelines require both syntactic enforcement (schemas, validators) and semantic grounding (domain-specific queries or constraints).

Paper (Year)	Input Format	Method Type	BPMN Gen.	Req. Extract.	Sec. Annot.
SecBPMN2BC Salnitri et al. (2017)	BPMN + manual constraints	Rule-based	✗	✓ (manual)	✓
Process Modeling with LLMs Kourani et al. (2024a)	Natural-language text (NL)	LLM-only	✓	✗	✗
Efficient Conversational Modeling Köpke and Safan (2024)	User dialogue text	LLM-only	✓	✗	✗
Nala2BPMN Nour Eldin et al. (2024)	Natural-language text (NL)	LLM + Rule-based	✓	✓ (basic)	✗
Multimodal BPMN Voelter et al. (2024)	PDFs with text + diagrams	Multimodal LLM	✓	✗	✗
LLM-Assisted SecBPMN Annotation (This Thesis)	NL process + legal text + BPMN model	Hybrid (Rule + LLM)	✓	✓	✓ (automated)

Table 3.1: Comparison of BPMN and SecBPMN modeling approaches.

As summarized in Table 3.1, prior research has explored diverse directions in automating process modeling, from rule-based security frameworks such as SecBPMN2BC to recent LLM-driven BPMN generation systems like Nala2BPMN and multimodal approaches leveraging GPT-4V. While these advances have significantly improved usability and automation, they remain limited in their ability to incorporate explicit security and compliance semantics. Existing SecureBPMN variants, including SecureBPMN, PE-BPMN, and SecBPMN2, provide expressive formalizations but continue to depend on expert-provided annotations.

To our knowledge, no existing approach yet integrates security-aware semantics into BPMN workflows through automated extraction from natural-language sources. This thesis addresses this gap by introducing an end-to-end framework that derives SecBPMN-style annotations directly from process descriptions and regulatory texts. By aligning extracted constraints with established ontologies and validating them via structured-output evaluation, the proposed method bridges the gap between manual security modeling and security-agnostic automation to advance secure, AI-assisted security annotated business process modeling. The following chapter introduces the proposed methodology in detail.

Chapter 4

Proposed Approach

This chapter presents the automated data processing pipeline developed to annotate security constraints within existing Business Process Model and Notation (BPMN) diagrams, transforming them into Security Business Process Model and Notation (SecBPMN)–compliant representations. The pipeline integrates deterministic rule-based transformations with the semantic reasoning capabilities of Large Language Models (LLMs), ensuring both structural validity and contextual adaptability. By combining formal schema enforcement with advanced language understanding, the system enables the consistent and reproducible enrichment of business process models with security annotations derived from natural-language descriptions and security requirements.

The proposed pipeline operates as an end-to-end workflow that converts unstructured process descriptions into formally enriched SecBPMN models. It comprises four main stages: (i) rule-based normalization of BPMN XML into a simplified JSON representation; (ii) schema-guided LLM for extracting security-relevant information from textual inputs; (iii) LLM-assisted mapping of extracted constraints to their corresponding BPMN elements; and (iv) deterministic reconstruction of the final SecBPMN XML. Together, these stages establish a reproducible and modular pathway for translating unstructured security requirements into machine-readable process annotations.

The remainder of this chapter is organized as follows. Section 4.1 presents the overall pipeline architecture and its guiding methodological principles. Section 4.2 describes each component of the pipeline in detail, covering rule-based transformation, LLM-driven extraction and mapping, and schema-compliant reconstruction. The chapter concludes by summarizing the design considerations that ensure the reproducibility, robustness, and adaptability of the proposed framework.

4.1 Overall Approach

The proposed methodology is implemented through an end-to-end pipeline that converts unstructured textual process descriptions into security-annotated business process models compliant with the Security Business Process Model and Notation (SecBPMN) standard. Figure 4.1 provides an overview of this pipeline. The primary objective of the pipeline is to translate natural language specifications of security requirements into formal representations within Business Process Model and Notation (BPMN) models, thereby enabling automated generation of secure, machine-readable process diagrams. The pipeline employs a hybrid approach that integrates rule-based transformations with semantic enrichment driven by large language models (LLMs). The rule-based modules ensure structural validity. These modules also ensure conformance with both BPMN and SecBPMN schemas. In contrast, the LLM-based modules contribute flexibility and domain expertise necessary for extracting and interpreting security requirements from natural language. This hybrid design balances the determinism and robustness of rule-based methods with the adaptability of generative models. The selection of data formats and augmentation strategies in the pipeline follows both technical and cognitive considerations. JSON was chosen as the intermediate representation instead of XML or YAML due to its flat key-value hierarchy, explicit typing, and ease of parsing by large language models, which struggle with deeply nested XML syntax. Unlike POWL-style graph encodings that emphasize control-flow soundness, the JSON schema allows direct indexing of BPMN entities such as tasks, data objects, and message flows, thereby facilitating precise anchoring of security annotations.

Similarly, the adoption of a retrieval-augmented generation (RAG) mechanism over pure few-shot, chain of thought and hybrid prompting mitigates hallucination and enhances factual grounding. By injecting domain knowledge from SecBPMN specifications [STS-Tool Development Team \(2015a\)](#), SecBPMN2BC [Köpke et al. \(2023\)](#) and STS-ML tutorials [STS-Tool Development Team \(2015b\)](#), RAG provides contextual priors that improve recall of less frequent constraints (e.g., non-delegation, auditability) without sacrificing schema validity. These design choices balance interpretability, scalability, and semantic accuracy across both rule-based and LLM-driven stages of the pipeline.

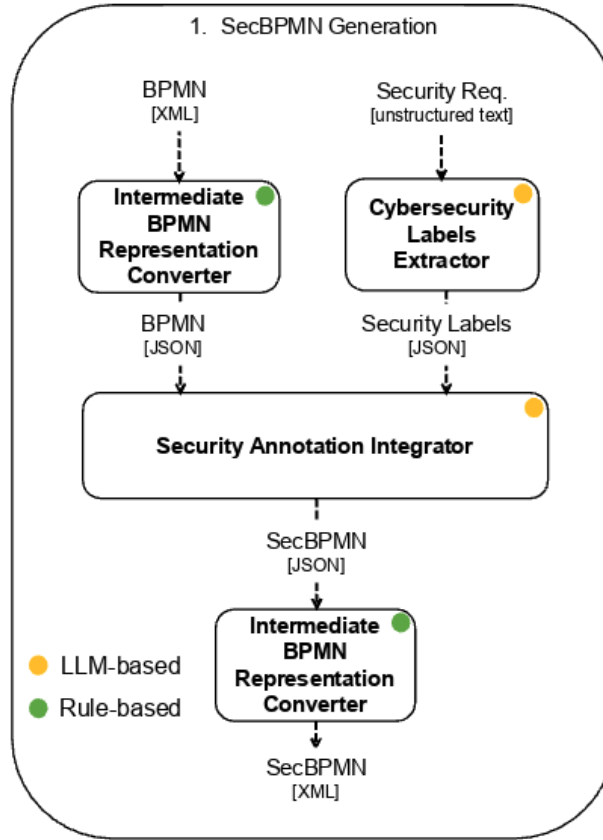


Figure 4.1: Overview of the proposed end-to-end pipeline

Each component of the pipeline serves a specific function. The initial rule-based converter transforms BPMN XML into a simplified JSON format to facilitate parsing and manipulation by LLMs. The LLM-based extractor processes unstructured text, such as requirements documents or process descriptions, to identify relevant cybersecurity labels and outputs them in structured JSON format. These labels are integrated into the process model using a dedicated LLM-based mapper that aligns them with the appropriate BPMN elements. A rule-based generator then converts the enriched SecBPMN JSON into SecBPMN XML, producing a complete diagram suitable for visualization and exploration in the chatbot interface. Collectively, these modules establish a comprehensive workflow for automating security-aware business process modeling.

4.2 Pipeline Stages

4.2.1 Step 1: BPMN XML to JSON Normalization

The initial stage of the proposed pipeline deterministically transforms Business Process Model and Notation (BPMN) 2.0 XML models into a normalized JavaScript Object Notation (JSON) representation. While BPMN XML adheres to its schema, its verbosity and deep nesting reduce its suitability for automated reasoning by large language models (LLMs). In contrast, JSON offers a compact and explicit schema that enables direct element access, supports enrichment with security annotations, and allows subsequent conversion into valid Security-annotated BPMN (SecBPMN) XML. This rule-based conversion process ensures reproducibility and structural fidelity for all process elements.

Algorithm 1 formalizes this transformation. The extraction process begins with participants (**participant**) and lanes (**lane**) to preserve the collaboration hierarchy of the process. Flow nodes, such as **task**, **event**, and **gateway** elements, are represented by their normalized types, textual labels, and sets of incoming and outgoing connections. Control-flow edges (**sequenceFlow**) capture directed dependencies between nodes and store associated conditional or default expressions. Message flows (**messageFlow**) maintain cross-pool communication links and their corresponding message references. Data-related elements (**dataObject**, **dataObjectReference**, and data associations) are retained to preserve information dependencies between activities and artifacts. Global **message** and **property** definitions provide completeness at both the process and collaboration levels.

Algorithm 1 Rule-based BPMN XML \rightarrow JSON Conversion

```

1: Input: BPMN XML  $X$    Output: JSON  $J$ 
2: Parse  $X$  into DOM  $T$ ; initialize  $J$  with empty arrays for pools, lanes,
   processes, sequenceFlows, messageFlows, dataObjects, dataObjectRefs,
   dataAssociations, messages, properties.
3: Build parent-child relationship map  $P$  and task-property associations
4: for all  $e \in T$  do
5:    $\tau \leftarrow \text{type}(e)$ ;  $id \leftarrow \text{id}(e)$ ;  $(in, out) \leftarrow (\text{incoming}(e), \text{outgoing}(e))$  as arrays
6:   if  $\tau = \text{participant}$  then
7:     append  $\{id, \text{name}(e), \text{processRef}(e)\}$  to  $J.\text{pools}$ 
8:   if  $\tau = \text{lane}$  then
9:     append  $\{id, \text{name}(e), \text{flowNodeRefs}(e), P.\text{getParent}(id)\}$  to  $J.\text{lanes}$ ; recur-
       sively handle nested lanes
10:  if  $\tau \in \{\text{task}, \text{event}, \text{gateway} \dots\}$  then
11:    append  $\{id, \text{normType}(\tau), \text{name}(e), in, out\}$  to  $J.\text{allElements}$  with proper-
       ties if applicable
12:  if  $\tau = \text{sequenceFlow}$  then
13:    append  $\{id, \text{sourceRef}(e), \text{targetRef}(e), \text{condition}(e)\}$  to  $J.\text{sequenceFlows}$ 
       and  $J.\text{flows}$ 
14:  if  $\tau = \text{messageFlow}$  then
15:    append  $\{id, \text{name}(e), \text{sourceRef}(e), \text{targetRef}(e), \text{messageRef}(e)\}$  to
        $J.\text{messageFlows}$ 
16:  if  $\tau = \text{dataObject}$  then
17:    append  $\{id, \text{name}(e), \text{references}(e)\}$  to  $J.\text{dataObjects}$ 
18:  if  $\tau = \text{dataObjectReference}$  then
19:    append  $\{id, \text{name}(e), \text{dataObjectRef}(e)\}$  to  $J.\text{dataObjectRefs}$ 
20:  if  $\tau \in \{\text{dataInputAssociation}, \text{dataOutputAssociation}\}$  then
21:    append  $\{id, \tau, \text{sourceRef}(e), \text{targetRef}(e)\}$  to  $J.\text{dataAssociations}$ 
22:  if  $\tau = \text{message}$  then
23:    append  $\{id, \text{name}(e)\}$  to  $J.\text{messages}$ 
24: Process data associations: resolve property references and update connections for
    data objects/references
25: Initialize all connection arrays ( $\text{incomingConnections}$ ,  $\text{outgoingConnections}$ ) to
    empty arrays
26: for all process  $P$  do
27:   let  $S_P = \{s \in P : \text{type}(s) = \text{startEvent}\}$ ;
28:   for all  $s \in S_P$  do
29:     if  $\text{hasConvergingGateway}(s)$ :
30:       recursively build structure with branch detection and join gateway resolution
31:     else:
32:       perform recursive traversal with stop conditions
33:       append the resulting hierarchical structure to  $J.\text{processes}$ 
34: return  $J$ 

```

Following extraction, the converter conducts a depth-first traversal from each **startEvent** to reconstruct partial-order segments of the control flow, with special handling for converging gateways to detect join points and branch structures. Parallelism is preserved by representing gateways as explicit fan-out and fan-in relations, and branch conditions (**conditionExpression**) are attached to outgoing edges. The procedure’s overall complexity is linear in the number of elements, $\mathcal{O}(|V| + |E|)$, where V and E represent the sets of nodes and flows, respectively. The resulting JSON representation is deterministic and namespace-aware, ensuring that all identifiers, relationships, and data dependencies are retained.

Table 4.1 presents an example of the XML-to-JSON mapping for a *Send Itinerary Details SecBPMN model’s* task. The XML fragment is transformed into a normalized JSON object with explicit identifiers and directional connections, preserving the semantics of the original model.

BPMN XML

```
<bpmn:sendTask id="Activity_04hluen"
  name="Send Itinerary details">
  <bpmn:incoming>Flow_10pi8xx</bpmn:incoming>
  <bpmn:outgoing>Flow_1o8s0a5</bpmn:outgoing>
</bpmn:sendTask>
```

Generated JSON

```
{
  "type": "sendtask",
  "id": "Activity_04hluen",
  "label": "Send Itinerary
  details",
  "incomingConnections":
    ["Flow_10pi8xx"],
  "outgoingConnections":
    ["Flow_1o8s0a5"]
}
```

Table 4.1: Excerpt of BPMN XML to JSON transformation for the *Send Itinerary Details* task in the iternery details transmission process.

The intermediate JSON representation plays a central role in subsequent stages of the pipeline: (i) it simplifies BPMN input data for LLM-based analysis with all extracted BPMN elements, (ii) it provides unambiguous anchors, such as tasks, message flows, and data objects, for potential enrichment in later stages, and (iii) it ensures that annotated models can be faithfully re-exported into schema-valid BPMN XML with complete structural and semantic equivalence.

4.2.2 Step 2: LLM-based Cybersecurity Label Extraction

The second module of the pipeline is responsible for extracting cybersecurity labels from unstructured natural-language process descriptions, such as requirement specifications or textual business-process narratives. Its purpose is to bridge the semantic

gap between informal textual input and the formalized security annotations required by the SecBPMN representation. By translating descriptive sentences into structured constraints, this component enables the subsequent automation of secure business-process model generation.

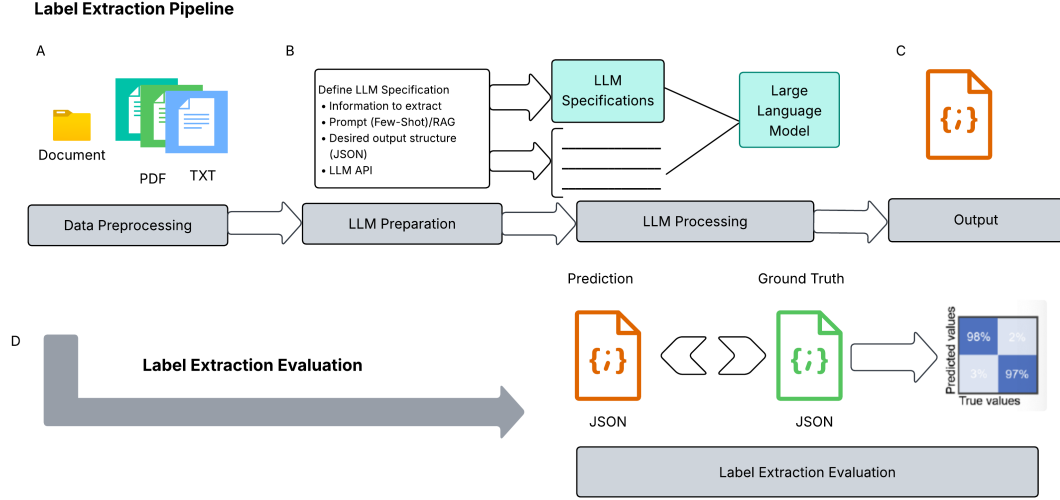


Figure 4.2: Pipeline of the LLM-based cybersecurity label extractor.

As illustrated in Figure 4.2, the pipeline begins with data preprocessing, which ensures the input is normalized, tokenized, and segmented into coherent linguistic units. The specific preprocessing steps are described in Section 5.2.1; here, the focus is on the extraction stage. At the core of this module lies a schema-guided large language model (LLM) that interprets the processed text and produces structured outputs compliant with the SecBPMN schema. The prompts supplied to the model explicitly define the expected JSON structure, ensuring consistency and syntactic validity of the generated annotations.

To enhance factual grounding and mitigate the intrinsic hallucination tendency of LLMs, the extraction process employs a retrieval-augmented generation (RAG) mechanism [Lewis et al. \(2020\)](#). Prior to model inference, the system retrieves semantically and lexically relevant excerpts from a multi-source SecBPMN knowledge base comprising SecBPMN2 specifications (70% of retrieved chunks), SecBPMN2BC documentation (20%), and STS-ML references (10%). Retrieval employs a hybrid approach combining BM25 lexical search with FAISS semantic similarity using OpenAI's `text-embedding-3-small` embeddings. These contextual passages are incorporated into the LLM prompt together with the process description, thereby providing authoritative definitions and examples of relevant security constructs. Conditioning the

model on this retrieved evidence strengthens its ability to identify and classify constraint types (e.g., confidentiality, integrity, availability) with higher semantic fidelity. This RAG-enhanced prompting strategy guides the model to generate labels that are both syntactically valid and semantically consistent with the SecBPMN formalism. By grounding the extraction in domain knowledge, the module produces outputs that exhibit reduced ambiguity and improved conceptual alignment with established security ontologies. The resulting annotations are returned as structured JSON objects normalized under the `securityConstraints` key as illustrated in the listing 4.1. Prior to model inference, the system retrieves semantically and lexically relevant excerpts from a domain-specific SecBPMN knowledge base that consolidates normative specifications and curated examples of security-aware process models. These contextual passages are incorporated into the LLM prompt together with the process description, thereby providing authoritative definitions and examples of relevant security constructs. Conditioning the model on this retrieved evidence strengthens its ability to identify and classify constraint types (e.g., confidentiality, integrity, availability) with higher semantic fidelity. This RAG-enhanced prompting strategy guides the model to generate labels that are both syntactically valid and semantically consistent with the SecBPMN formalism. By grounding the extraction in domain knowledge, the module produces outputs that exhibit reduced ambiguity and improved conceptual alignment with established security ontologies. The resulting annotations are returned as structured JSON objects normalized under the `securityConstraints` key as illustrated in the listing 4.1.

Listing 4.1: Representative JSON output of the LLM-based cybersecurity label extractor for the *Itinerary Details Transmission* scenario.

```
{
  "securityConstraints": [
    {
      "id": "integrityMF_1234",
      "type": "integrityMF",
      "secType": "integrity",
      "targetType": "messageFlow",
      "element_label": "Itinerary Details Transmission",
      "source_participant_label": "Travel Agency System",
      "target_participant_label": "Amadeus Service",
      "parameters": {
        "enfBy": []
      },
      "trace": {
        "evidence": [
          {
            "chunk_id": "secbpmn2#000099",
            "doc_type": "secbpmn2",
            "section_path": ["INTEGRITY"]
          }
        ],
        "rationale": "The process description states 'content is guarded so it arrives exactly as
```

```

    composedno additions, omissions, or edits along the way', which aligns with the integrity of
    transmission as described in SecBPMN2."
  }
},
{
  "id": "integrityD0_5678",
  "type": "integrityD0",
  "secType": "integrity",
  "targetType": "dataObjectReference",
  "element_label": "Stored Itinerary",
  "participant_label": "Amadeus Service",
  "parameters": {
    "enfBy": []
  },
  "trace": {
    "evidence": [
      {
        "chunk_id": "secbpmn2#000099",
        "doc_type": "secbpmn2",
        "section_path": ["INTEGRITY"]
      },
      {
        "chunk_id": "secbpmn2#000057",
        "doc_type": "secbpmn2",
        "section_path": ["DATA STORE"]
      }
    ],
    "rationale": "The requirement that 'The saved record is kept as an exact copy of what was
    received and is protected from after-the-fact changes' directly implies an integrity constraint
    on the stored Itinerary data object, as per SecBPMN2's definition of integrity and data stores."
  }
},
{
  "id": "availabilityMF_9012",
  "type": "availabilityMF",
  "secType": "availability",
  "targetType": "messageFlow",
  "element_label": "Itinerary Details Transmission",
  "source_participant_label": "Travel Agency System",
  "target_participant_label": "Amadeus Service",
  "parameters": {
    "enfBy": []
  },
  "trace": {
    "evidence": [
      {
        "chunk_id": "secbpmn2#000093",
        "doc_type": "secbpmn2",
        "section_path": ["AVAILABILITY"]
      }
    ],
    "rationale": "The phrase 'the delivery path is kept reachable' indicates an availability
    requirement for the message flow, ensuring it remains operational for transmission, as defined in
    SecBPMN2."
  }
}
]
}

```

Each constraint entry specifies the task to which it applies, the security mechanism involved, and its enforcement properties. The structured outputs generated by this step serve as direct inputs to the security-aware label mapping where they are attached to the corresponding BPMN model elements to construct complete SecBPMN JSON.

4.2.3 Step 3: Security Label Mapping to BPMN Elements

The third module of the pipeline connects the extracted security constraints with the corresponding elements of the business process model. Its purpose is to produce a unified **SecBPMN** JSON file that extends the original BPMN structure with cybersecurity information while preserving all process semantics. As shown in Figure 4.3, the module receives two structured inputs: (i) the BPMN model in JSON format from 4.2.1, and (ii) the extracted security labels in JSON format from 4.2.2. These inputs are combined into a structured prompt and processed by a large language model (LLM), which performs schema-aware reasoning to assign each security label to the correct BPMN element.

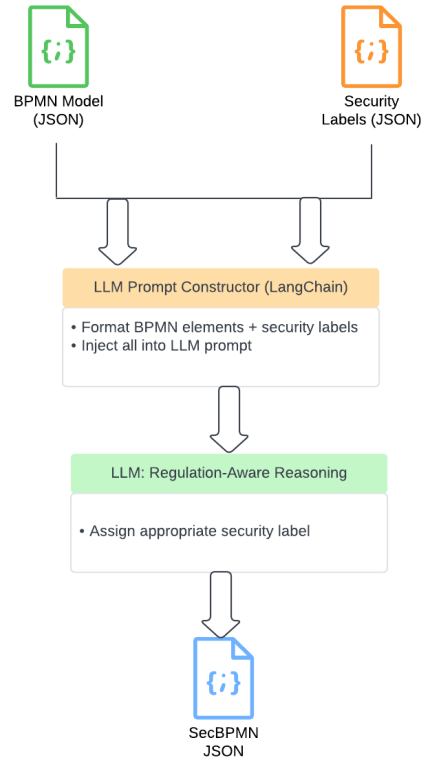


Figure 4.3: LLM-based security labels mapper with BPMN elements

To ensure systematic and reproducible reasoning, the LLM is guided by a schema-constrained prompt that defines explicit reasoning steps and a strict output format. The model is instructed to act as a “BPMN–Security Mapping Specialist” and performs type-aware element matching: first filtering candidate elements by constraint target type, then matching labels within the filtered set, validating type compatibility between constraint suffixes and element types, and finally generating associations according to cardinality rules. An excerpt version of the prompt is presented in Listing 4.2.

Listing 4.2: Schema-guided mapping prompt used for security label integration into BPMN models. The prompt uses type-aware filtering followed by label matching and validation.

```
You are a BPMN–Security mapper. Map input security labels to BPMN elements
and produce complete SecBPMN JSON.

INPUT STRUCTURE:
BPMN BASE (unchanged): {{ bpmn_base }}
COMPACT INDEX (for matching): POOLS, ELEMENTS, DATA_OBJECT_REFS,
    MESSAGE_FLOWS
SECURITY LABELS: {{ security_labels }}

MATCHING STRATEGY (type-aware):
1) Filter by constraint.targetType FIRST, then match by label:
    - targetType="task"    search ONLY in: task, sendTask, receiveTask, ...
    - targetType="dataObjectReference" search ONLY in DATA_OBJECT_REFS
    - targetType="messageFlow" search ONLY in MESSAGE_FLOWS
    - targetType="gateway" search ONLY in exclusiveGateway, parallelGateway,
    ...
    - targetType="participant" search ONLY in POOLS

2) Within filtered candidates, match by element_label:
    - Prefer exact label match (case-insensitive)
    - Else: semantic similarity with normalization

3) Type compatibility validation (MANDATORY):
    - Constraint suffix Required element type:
      * "Act" task, sendTask, receiveTask, userTask, serviceTask, ...
      * "DO" dataObject, dataObjectReference
      * "MF" messageFlow
      * "GW" exclusiveGateway, parallelGateway, inclusiveGateway, ...
      * "ORG" participant, pool (requires 2 associations)
```

```

4) Association cardinality:
  - Act/DO/GW/MF: exactly 1 association
  - ORG: exactly 2 associations (fromPool, toPool)

5) Deduplication: At most one constraint per (secType, type, targetRef)

OUTPUT STRUCTURE:
{
  ...bpmn_base... (unchanged),
  "securityConstraints": [...], // Flattened from SECURITY LABELS
  "securityAssociations": [...], // Link constraints to BPMN elements
  "constraintsByTarget": {...}, // Reverse index: elementconstraints
  "associationsByElement": {...} // Reverse index: elementassociations
}

```

During execution, the LLM receives the intermediate BPMN model and the extracted security labels as structured JSON inputs. It analyzes both and produces a complete SecBPMN JSON that preserves the original process model while adding two new sections: `securityConstraints` and `securityAssociations`. Each security constraint represents a specific protection requirement, such as confidentiality or integrity, while each association links that constraint to the BPMN element to which it applies.

To improve matching accuracy, the prompt includes a COMPACT INDEX that distills the BPMN model into pools, elements, data objects, and message flows while filtering out structural complexity. This focused representation helps the LLM identify matching elements without parsing the entire process structure, improving both accuracy and efficiency.

The mapping process uses a type-aware matching approach: (1) filter candidates by constraint type, (2) match labels within the filtered set via exact name or semantic similarity, (3) validate type compatibility between the constraint suffix and element type, and (4) set association cardinality (1 for Act/DO/GW/MF; 2 for ORG). A fallback mechanism activates on LLM failure, removing trace metadata and attempting rule-based associations using exact or semantic name matching.

Semantic similarity between extracted constraint labels and BPMN element names is determined through the LLM’s own reasoning rather than an external embedding or distance metric. The model is instructed, within the prompt, to assess conceptual equivalence and contextual relevance (e.g., synonyms or paraphrases such as “customer record” vs. “client data”) using its internal language understanding capabilities. This design choice prioritizes interpretability and leverages the model’s contextual comprehension to resolve ambiguous mappings without relying on predefined embedding thresholds.

Type compatibility is enforced by suffix matching: constraint names ending in **Act** map to activities (task, sendTask, receiveTask, userTask, serviceTask, scriptTask, businessRuleTask, manualTask, callActivity); **DO** to data objects; **MF** to message flows; **GW** to gateways (exclusive, parallel, inclusive, complex, event-based); and **ORG** to participants (requiring two associations: fromPool and toPool).

The mapper also generates two reverse lookup indexes to enable efficient navigation: **constraintsByTarget** maps each BPMN element ID to its associated constraint IDs, while **associationsByElement** maps both constraint and target element IDs to their association IDs. These indexes support downstream processing where security policies must be queried by element or constraint. The mapper flattens nested constraint parameters into the top-level constraint object, preserving data types (booleans as booleans, numbers as numbers, arrays as arrays) to ensure compatibility with downstream SecBPMN XML generation.

The final output is a single, valid SecBPMN JSON document that combines both functional and security perspectives of the business process. This enriched representation enables downstream reasoning, visualization, and secure process execution while ensuring full structural and semantic consistency with the original BPMN model.

4.2.4 Step 4: SecBPMN JSON to XML Reconstruction

The final stage of the pipeline converts the enriched SecBPMN JSON representation back into canonical BPMN 2.0 XML. This step is deterministic and rule-based, maintaining strict schema compliance to ensure the generated XML can be visualized, validated, and executed in standard BPMN tools.

Algorithm 2 describes the conversion procedure. The transformation initializes the root XML element with required BPMN and SecBPMN namespaces. It then constructs the collaboration block containing pools (**participants**), message flows, and all security extensions (**securityConstraints** and **securityAssociations**). Messages are added as root-level elements. Next, the process section is generated with all flow elements: tasks, events, gateways, sequence flows, data objects, and data associations. Each JSON entry is mapped to its corresponding BPMN or SecBPMN XML tag through predefined type mappings. Security constraints are enriched with their type-specific properties according to the property schema, ensuring all attribute values are properly formatted (booleans, strings, numbers) and multi-valued arrays are correctly represented.

Algorithm 2 Rule-based SecBPMN JSON \rightarrow XML Conversion

```

1: Input: JSON  $J = (P, L, \Pi, F, M, D, S)$  where  $D = (D_O, D_R, D_A)$ ,  $S = (S_c, S_a)$ 
2: Output: XML document  $X$ 
3: Define tag maps  $\varphi_{\text{bpmn}}$ ,  $\varphi_{\text{sec}}$ ; property schema  $\Phi_{\text{sec}}$ 
4: Initialize root  $X \leftarrow \langle \text{bpmn:definitions} \rangle$  with namespaces
5: Add collaboration  $C \leftarrow \langle \text{bpmn:collaboration} \rangle \subset X$ 
6: for all  $p \in P$  do
7:   insert  $\langle \text{bpmn:participant}(id, name, processRef) \rangle$  into  $C$ 
8: for all  $m \in M$  do
9:   insert  $\langle \text{bpmn:messageFlow}(id, name, sourceRef, targetRef, messageRef) \rangle$ 
   into  $C$ 
10: Add messages to  $X$ :
11: for all  $msg$  do
12:   insert  $\langle \text{bpmn:message}(id, name) \rangle$  into  $X$ 
13: for all  $s \in S_c$  do
14:    $secElem \leftarrow \langle \text{sec}:\varphi_{\text{sec}}(\text{type}(s))(id, secType) \rangle$ 
15:   apply  $\Phi_{\text{sec}}(\text{type}(s))$ : format and add properties as children into  $secElem$ 
16:   insert  $secElem$  into  $C$ 
17: for all  $a \in S_a$  do
18:   insert  $\langle \text{sec:securityAssociation}(id, sourceRef, targetRef) \rangle$  into  $C$ 
19: Create process  $Q \leftarrow \langle \text{bpmn:process}(id, isExecutable=false) \rangle \subset X$ 
20: if  $L \neq \emptyset$  then
21:   add  $\langle \text{laneSet} \rangle$  with lanes into  $Q$ 
22: for all  $\pi \in \Pi$  do
23:   for all  $e \in \pi$  do
24:     insert  $\langle \text{bpmn}:\varphi_{\text{bpmn}}(\text{type}(e))(id, name) \rangle$  into  $Q$ 
25:     if  $e$  has branches then
26:       recursively materialize path( $e$ )
27: for all  $f \in F$  do
28:   insert  $\langle \text{bpmn:sequenceFlow}(id, sourceRef, targetRef) \rangle$  into  $Q$ 
29:   if condition( $f$ )  $\neq \emptyset$  then
30:     add  $\langle \text{conditionExpression} \rangle$  child
31: for all  $o \in D_O$  do
32:   insert  $\langle \text{bpmn:dataObject}(id, name) \rangle$  into  $Q$ 
33: for all  $r \in D_R$  do
34:   insert  $\langle \text{bpmn:dataObjectReference}(id, name, dataObjectRef) \rangle$  into  $Q$ 
35: for all  $a \in D_A$  do
36:   insert  $\langle \text{bpmn:type}(a)(id) \rangle$  into  $Q$  with  $\langle \text{sourceRef}, \text{targetRef} \rangle$  children
37: Format  $X$  with indentation and return  $X$ 

```

An excerpt of the generated XML for the itinerary transmission scenario is shown in Listing 4.3. The snippet demonstrates how confidentiality and integrity constraints are encoded together with their corresponding security associations that link them to BPMN message flows.

Listing 4.3: Excerpt of generated SecBPMN XML for the itinerary transmission model

```
<sec:confidentialityMF id="confidentialityMF_4821" secType="confidentiality">
  <sec:enfBy>TLS</sec:enfBy>
  <sec:readers>Amadeus Service</sec:readers>
  <sec:writers>Travel Agency System</sec:writers>
</sec:confidentialityMF>
<sec:securityAssociation id="SecurityAssociation_1a2b3c4d"
  sourceRef="confidentialityMF_4821" targetRef="Flow_invnd36"/>
```

The mapping preserves all element identifiers and relationships from the JSON structure, ensuring that the final XML output remains complete, structurally sound, and compatible with BPMN schema validators. Each BPMN and SecBPMN element type is mapped deterministically to its corresponding XML tag through predefined mappings. The implementation uses deterministic tag mappings to convert JSON structures to valid BPMN 2.0 XML elements, ensuring schema completeness through structural fidelity.

The generated XML strictly conforms to BPMN 2.0 and SecBPMN specifications, has been validated using standard XML parsers, and can be imported into BPMN modeling tools such as Camunda Modeler and bpmn.io. This confirms that the reverse-engineering procedure preserves both process semantics and security annotations with full schema fidelity.

In summary, this chapter presented the complete methodological framework for automating security-aware business process modeling through a hybrid rule-based and LLM-assisted pipeline. The approach integrates deterministic transformations to ensure schema validity with semantic reasoning modules that enable accurate extraction and mapping of security constraints from natural-language process descriptions. Each component—from XML-to-JSON conversion to SecBPMN reconstruction—was designed to guarantee reproducibility, structural fidelity, and contextual adaptability. The following chapter applies the proposed methodology to a curated dataset of SecBPMN–text pairs. It details the experimental setup, evaluation metrics, and performance analyses across multiple large language models and prompting strategies, providing an empirical validation of the pipeline’s accuracy, structural robustness, and efficiency. The complete pipeline is operationalized through an interactive prototype, which serves as an experimental validation tool to assess usability and end-to-end performance of the proposed framework.

Chapter 5

Implementation and Evaluation

This chapter presents the empirical evaluation of the proposed SecBPMN automation pipeline. Its primary objective is to systematically assess how effectively Large Language Models (LLMs) can translate natural-language security requirements into formally annotated Security Business Process Model and Notation (SecBPMN) models. The evaluation examines the pipeline’s capacity to preserve both structural validity and semantic fidelity, measuring not only whether generated annotations are syntactically correct, but also whether they accurately capture the intended security semantics of the underlying processes. Furthermore, it analyzes how performance varies across model families, prompting strategies, and generation paradigms, providing insights into the generalization behavior of LLMs in structured security modeling.

To ensure transparency and reproducibility, this chapter details the entire evaluation protocol, from dataset construction to experimental design and performance measurement. It introduces the curated benchmark of SecBPMN–text pairs and describes the experimental setup used to test different LLM configurations and prompting methods, including the retrieval-augmented generation (RAG) variant. Quantitative analyses evaluate correctness, completeness, and structural robustness using objective metrics such as precision, recall, F1-score, Jaccard similarity, and schema validity. Complementary qualitative evaluations employ an LLM-as-a-Judge framework to assess the interpretive quality of generated annotations through rubric-based scoring. In addition, efficiency benchmarks measured in token cost and latency, highlight the computational trade-offs between accuracy, consistency, and resource consumption.

This chapter is organized as follows. Section 5.1 describes the prototype implementation that operationalizes the SecBPMN generation pipeline and validates its usability through an interactive chatbot interface. Section 5.2.1 presents the construction of the benchmark dataset used for evaluation. Section 5.2 defines the experimental setup, including generation paradigms, and LLM model families. Section ?? details the eval-

uation methods and metrics used for quantitative, qualitative, and efficiency analyses. Finally, Section 5.3 reports the results and discusses the key findings derived from these experiments.

5.1 Prototype Implementation

The developed chatbot prototype serves as the user-facing interface that integrates all components of the SecBPMN generation pipeline. Its primary purpose is to offer an interactive environment where users can describe business processes in natural language or upload existing BPMN 2.0 models and receive, in real time, a security-annotated version compliant with the SecBPMN standard. This implementation operationalizes the proposed rule-based and LLM-based modules into a unified assistant for security-aware process modeling and demonstrates the system’s practical usability.

Figure 5.1 presents the overall architecture of the chatbot. The system follows a three-layer design composed of a React-based frontend interface, a Flask-based backend orchestrator, and external LLM services optionally connected to a knowledge base for retrieval-augmented reasoning. The frontend provides a conversational interface combined with an embedded BPMN renderer built on `bpmn-js`. Users can submit textual process descriptions and upload BPMN XML files and select both the reasoning model (e.g., GPT-4.1, Mistral Small) and the generation method (prompt-based or retrieval-augmented). All interactions are transmitted to the backend through lightweight REST endpoints (`/bot/intent`, `/bot/general`, `/annotate`).

The backend acts as the coordination layer that connects the user interface with the SecBPMN generation modules. Upon receiving a user query, it first classifies the intent, distinguishing between general conversation, explanation, or annotation—and triggers the corresponding workflow. In annotation mode, the backend executes the full four-stage transformation pipeline: (i) conversion of BPMN XML into a normalized JSON structure, (ii) extraction of security constraints through prompt-based or retrieval-augmented generation, (iii) mapping of extracted constraints to BPMN elements, and (iv) reconstruction of the enriched SecBPMN XML. To ensure accurate visualization, the backend also compiles an *apply SecBPMN renderer* that specifies how each security annotation should be displayed in the frontend modeler.

The reasoning layer enables both generative and retrieval-augmented inference. When retrieval-augmented generation (RAG) is activated, the chatbot retrieves relevant fragments, definitions, and rules from a curated SecBPMN knowledge base. These retrieved segments are combined with the user input to form a structured prompt that guides the model in producing consistent, schema-conformant security annotations.

The end-to-end workflow begins when a user provides a textual description of a process

or security requirements and uploads a BPMN XML model. The chatbot performs intent classification, executes the extraction and mapping modules, and returns the resulting SecBPMN JSON and XML outputs. The frontend imports the generated XML into the `bpmn-js` rendering engine, reconstructs the process diagram, and overlays security annotations on the corresponding tasks, message flows, or data objects. Each generated model is automatically validated against BPMN 2.0 and SecBPMN schemas to ensure both structural and semantic correctness.

Figure 5.1 illustrates the architecture of the prototype, showing the data flow between the user interface, backend modules, and external LLM services. Dashed arrows denote HTTP communication, while solid arrows represent internal data transformations.

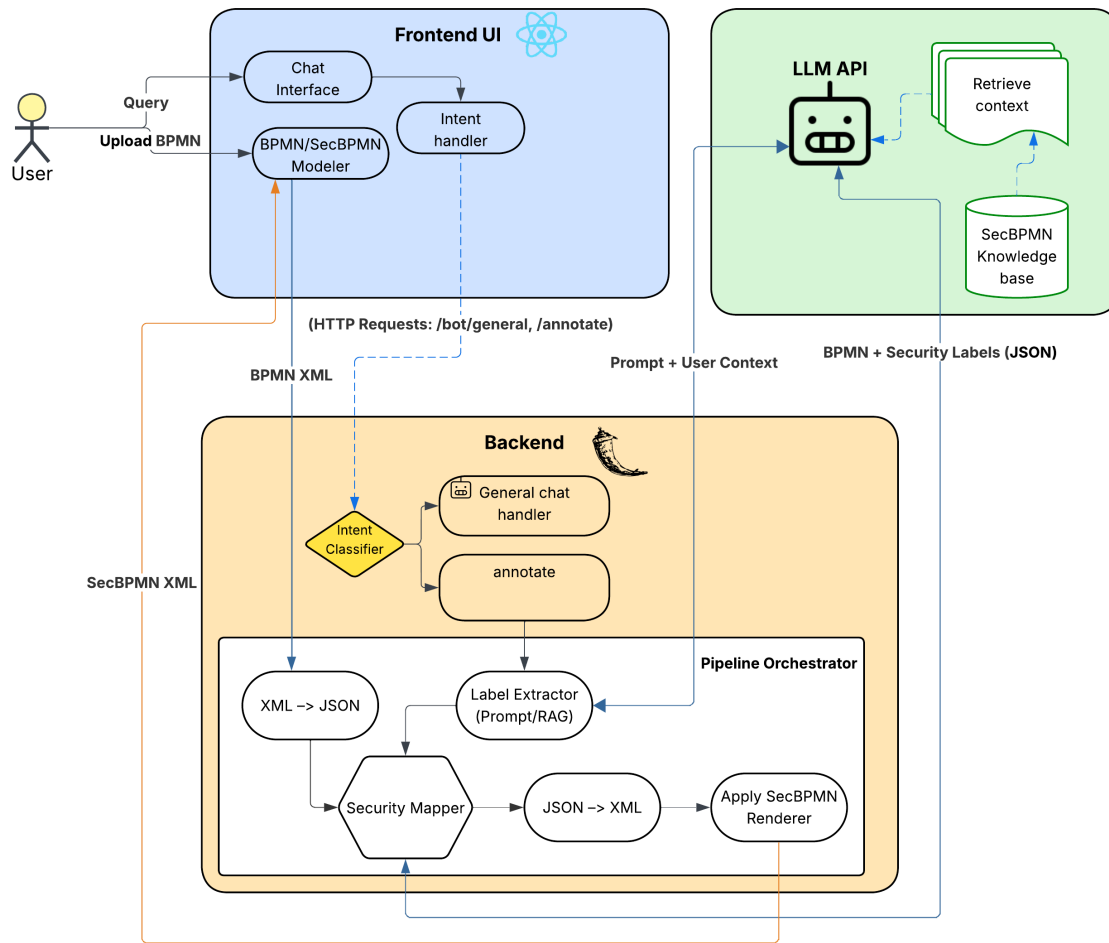


Figure 5.1: SecBPMN-GPT prototype architecture

An example interaction illustrates how the chatbot interprets a security requirement and enriches the corresponding process model:

User: *During the transmission of itinerary information, the communication channel must remain continuously accessible to prevent any interruption. The transmitted data should arrive exactly as composed, without any alteration or loss. Upon receipt, the system must retain a verifiable copy of the itinerary, ensuring that it remains resistant to post-delivery changes.*

SecBPMN-GPT: *The message flow has been secured with availability constraints, integrity during transfer, and integrity of the stored record.*

The chatbot interface then renders the generated SecBPMN XML using the BPMN modeler, visually enriching the process with explicit security annotations. Each annotation is displayed as an overlay linked to its associated task, data object, or message flow, with tooltips summarizing its security properties and enforcement context. Figure 5.2 presents an example for the *Itinerary Details Transmission* process, where the message flow between participants is protected to ensure reliable delivery and data integrity, while the stored record remains preserved as a trustworthy, immutable copy for subsequent operations. This integration allows users to inspect, validate, and refine security requirements directly within the modeling interface.

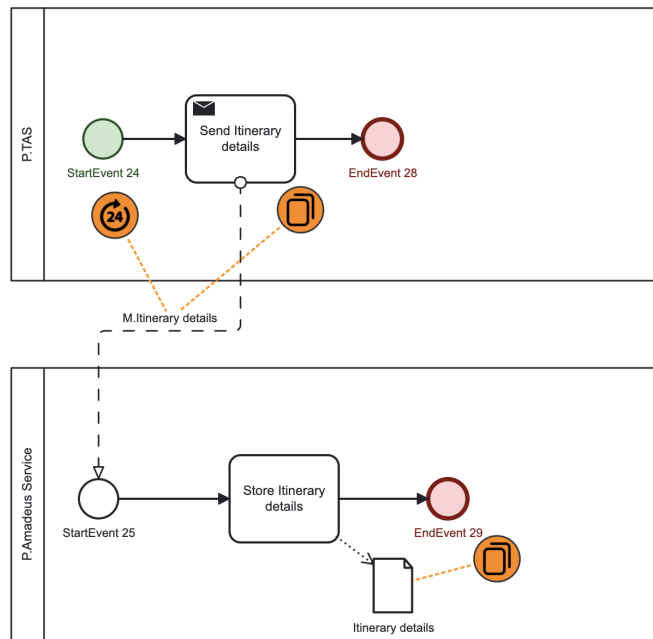


Figure 5.2: Itinerary Details Transmission process model enriched with automatically generated cybersecurity annotations by SecBPMN-GPT.

5.2 Experimental Setup

The experiments were designed to systematically evaluate the proposed pipeline across different generation paradigms, prompting strategies, and model families, using the curated dataset of SecBPMN–text pairs. The setup establishes controlled conditions under which system outputs can be compared in a reproducible and interpretable manner.

5.2.1 Dataset

Dataset Construction

A dedicated dataset is necessary to implement the proposed pipeline and to enable systematic evaluation. As discussed in the previous chapter, no standardized or publicly available corpus currently pairs Security-annotated Business Process Model and Notation (SecBPMN) models with natural-language descriptions. Existing BPMN repositories rarely include security annotations, and available SecBPMN examples are scattered across research papers, tutorials, and project repositories often in non-machine-readable or inconsistent formats. To address these gaps, a curated benchmark of 27 SecBPMN–text pairs was constructed, specifically designed for evaluation rather than large-scale model training. The dataset emphasizes canonical representation, semantic alignment, and reproducibility over raw quantity. It serves as a focused benchmark for testing and validating automated annotation, reasoning, and explanation methods in security-augmented process modeling.

The dataset was assembled from four principal sources. First, ten SecBPMN models were reconstructed from the *Travel Agency Service (TAS)* tutorial of the Security, Trust, and Service (STS) project [STS-Tool team \(2016\)](#), covering booking, payment, itinerary transmission, and delegation workflows. Second, seven models were contributed by domain experts in healthcare and telemedicine, capturing realistic tele-visit, remote consultation, hospital admission, and diagnostic-reporting processes. Third, several case studies were integrated from academic and industrial research, including the *Route-Based Trajectory (RBT)* negotiation process [Salnitri et al. \(2015b\)](#), the *road mis-construction claim process* [Köpke et al. \(2023\)](#), and aviation orchestration and flight-planning examples from the RSSVJ18 repository [Ramadan et al. \(2018\)](#). Additionally, four blockchain-oriented process models were adopted from [Meroni et al.](#), illustrating SecBPMN2BC case studies for token minting, transfer, and interoperability scenarios.

The domain coverage of the curated dataset, shown in Figure 5.3, demonstrates its thematic breadth across travel, healthcare, blockchain, aviation, financial, and administrative domains.

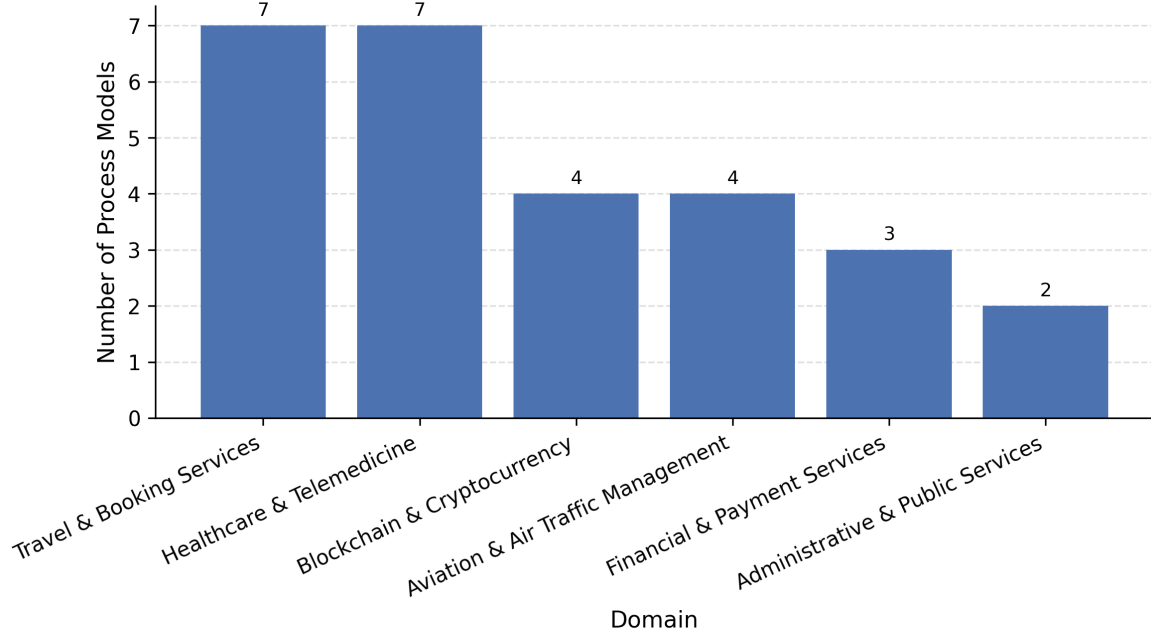


Figure 5.3: Domain coverage of the curated dataset.

To systematically evaluate performance variation across heterogeneous process complexities, the curated dataset was categorized into three distinct complexity tiers through a multi-factor assessment framework. This classification scheme employs empirically-derived thresholds based on structural characteristics (process size, branching complexity, collaboration depth) and semantic density (security annotation frequency) to distinguish between lightweight, moderate, and highly complex workflows. A process is classified as Simple if it satisfies all of the following constraints: ≤ 5 activities, ≤ 2 gateways, ≤ 2 message flows, ≤ 6 security annotations, ≤ 2 pools, and ≤ 3 data objects. These processes typically represent linear, single-participant workflows with minimal branching logic and straightforward security requirements, such as simple payment verification or basic token transfer operations. The low threshold across all dimensions ensures that Simple processes require neither sophisticated coordination nor dense security semantics.

Conversely, a process is classified as Complex if it satisfies any of the following conditions:

- ≥ 15 activities, indicating substantial operational scope
- ≥ 12 security annotations, reflecting high security-sensitive components
- The combination of ≥ 15 activities and ≥ 13 security annotations

- The combination of ≥ 5 gateways and ≥ 7 message flows, suggesting intricate branching and multi-participant interactions
- The combination of ≥ 15 activities and ≥ 14 message flows
- ≥ 23 activities, representing extensive process scale
- ≥ 30 security annotations, indicating high security density
- The combination of ≥ 3 pools and ≥ 15 activities, denoting complex multi-organizational orchestration

These criteria capture processes characterized by multi-participant orchestrations, extensive branching logic, or dense security semantics. Processes not satisfying either Simple or Complex criteria are classified as Medium, representing moderate-scale workflows with balanced structural and security requirements.

Under this classification scheme, the dataset distribution comprises 11 Simple processes (40.7%), 7 Medium processes (25.9%), and 9 Complex processes (33.3%). This balanced representation across complexity tiers enables robust evaluation of model performance under varying structural and semantic demands, while preventing results from being dominated by any single complexity category.

Table 5.1 provides a detailed structural overview of all Security Business Process Model and Notation (SecBPMN) instances included in the benchmark. For each model, the table enumerates the number of organizational and behavioral components (*pools*, *lanes*, *activities*, *events*, and *gateways*), as well as collaboration artifacts (*message flows*, *data objects*), and the total number of security annotations. The final columns further break down the distribution of security targets across activities (ACT), data objects (DO), message flows (MF), gateways (GW), and organizational constraints (ORG), reflecting the extended SecBPMN schema that supports both process-level (behavioral) and collaboration-level (structural) security constraints.

SecBPMN Model Name	Pools	Lanes	Activities	Events	Gateways	Message Flows	Data Objects	Security Annotations	Security Targets				
									ACT	DO	MF	GW	ORG
Dymaxion Mint Tokens	2	0	6	12	2	5	5	34	18	14	0	1	1
Dymaxion Transfer Tokens	3	0	15	18	2	9	9	64	36	25	0	1	2
ICRC Receive Tokens from QR User v2	3	0	10	13	0	7	10	25	12	13	0	0	0
ICRC Send Tokens to QR User v2	3	0	6	9	0	4	7	19	8	10	0	0	1
Amadeus service produces flight tickets	1	0	1	2	0	0	2	3	3	0	0	0	0
Appropriate diagnosis obtained	4	0	17	6	1	8	3	7	5	2	0	0	0
External services1	4	0	18	11	8	5	8	9	3	0	6	0	0
External services2	5	0	39	21	15	17	19	20	4	0	16	0	0
Patient Hospitalised	4	0	6	3	2	2	0	2	2	0	0	0	0
Alert agency process	2	0	2	4	0	1	0	3	3	0	0	0	0
Credit card verified process	1	0	3	3	1	0	0	6	6	0	0	0	0
Example2	2	0	5	5	2	1	2	2	1	0	1	0	0
Flight ticket booked delegation	2	0	2	4	0	1	0	1	1	0	0	0	0
Flight ticket booked process	1	0	4	3	1	0	0	3	3	0	0	0	0
Flight plane	3	0	23	16	17	14	14	22	5	0	17	0	0
Give advise and final report	3	0	3	2	0	1	1	4	2	2	0	0	0
Healthcare business process	3	2	10	13	4	2	4	8	2	3	3	0	0
Hotel booked delegation	2	0	4	6	1	2	0	4	4	0	0	0	0
Hotel booked delegation2	2	0	2	4	0	1	0	2	2	0	0	0	0
Itinerary details transmission	2	0	2	4	0	1	1	3	0	1	2	0	0
ObtainTelevisit	3	0	5	6	3	2	0	1	0	0	1	0	0
Perform tele visit	3	0	7	4	1	3	2	7	2	4	1	0	0
Prepayment made process	1	0	3	2	2	0	0	3	3	0	0	0	0
RBT negotiation process model	2	2	14	2	5	2	3	13	9	3	1	0	0
Road misconstruction claim process	3	4	6	10	5	3	3	2	1	1	0	0	0
Scenario3 moving to EU country	4	0	14	4	0	7	1	4	1	3	0	0	0
Teleconsultation setup process	1	0	1	2	0	0	0	4	4	0	0	0	0

Table 5.1: Comprehensive structural and security overview of all curated SecBPMN process models. **Simple** models are shown in green, **Medium** in orange, and **Complex** in red. The final five columns detail the distribution of security targets across activities (ACT), data objects (DO), message flows (MF), gateways (GW), and organizational constraints (ORG).

The variation in size, collaboration depth, and annotation density across models illustrates the heterogeneity of real-world business processes, ranging from lightweight transactional workflows to multi-participant collaborations with rich security semantics. Each diagram was redrawn or imported into a customized SecBPMN editor that normalized namespaces, validated schema conformance, and exported canonical XML representations for downstream task. Every model was paired with a natural-language description summarizing functional behavior (actors, message handoffs, and object interactions) and the associated security constraints. Descriptions were refined using large language models (GPT-5 and Claude 4 Sonnet) and then manually verified to

ensure correctness and consistency with the annotated models. Provenance metadata, including source, description method, and validation status, were recorded for reproducibility.

To concretize how these normalized diagrams are stored and exchanged, we provide an example of the underlying XML representation format. To illustrate the canonical structure used throughout the benchmark, a compact SecBPMN XML excerpt is presented in Listing 5.1. This example corresponds to the *Itinerary details transmission* scenario and demonstrates how security constructs are embedded within the BPMN schema. Each security requirement is modeled as a dedicated extension element and linked to its corresponding BPMN element through `sec:securityAssociation` references. This representation ensures both semantic clarity and machine readability while preserving full compatibility with standard BPMN rendering tools.

Listing 5.1: Compact SecBPMN XML excerpt for the *Itinerary Details Transmission*.

```
<?xml version="1.0" encoding="UTF-8"?>
<bpmn:definitions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:bpmn="http://www.omg.org/spec/BPMN/20100524/MODEL"
  xmlns:bpmndi="http://www.omg.org/spec/BPMN/20100524/DI"
  xmlns:dc="http://www.omg.org/spec/DD/20100524/DC"
  xmlns:bioc="http://bpmn.io/schema/bpmn/biocolor/1.0"
  xmlns:color="http://www.omg.org/spec/BPMN/non-normative/color/1.0"
  xmlns:di="http://www.omg.org/spec/DD/20100524/DI"
  xmlns:sec="http://example.com/secbpmn" id="Definitions_1kj200r"
  targetNamespace="http://bpmn.io/schema/bpmn" exporter="bpmn-js
  (https://demo.bpmn.io)" exporterVersion="18.6.1">
  <bpmn:collaboration id="Collaboration_0xkc2b8">
    <bpmn:participant id="Participant_0qt8989" name="P.TAS"
      processRef="Process_05yamrl" />
    <bpmn:participant id="Participant_1xzbx4" name="P.Amadeus Service"
      processRef="Process_0jaus41" />
    <bpmn:messageFlow id="Flow_1nvnd36" name="M.Itinerary details"
      sourceRef="Activity_04hluen" targetRef="Event_0ldt097" />
    <sec:securityAssociation id="SecurityAssociation_Ouiannnd"
      sourceRef="availabilityMF_467" targetRef="Flow_1nvnd36" />
    <sec:securityAssociation id="SecurityAssociation_0p2miby"
      sourceRef="integrityMF_2088" targetRef="Flow_1nvnd36" />
    <sec:securityAssociation id="SecurityAssociation_1hqzb41"
      sourceRef="integrityDO_2076" targetRef="DataObjectReference_Of6utak" />
  </bpmn:collaboration>
  <bpmn:process id="Process_05yamrl" isExecutable="false">
    <bpmn:startEvent id="StartEvent_0c9qnlk" name="StartEvent 24">
      <bpmn:outgoing>Flow_10pi8xx</bpmn:outgoing>
    </bpmn:startEvent>
    <bpmn:sendTask id="Activity_04hluen" name="Send Itinerary details">
```

```

    <bpmn:incoming>Flow_10pi8xx</bpmn:incoming>
    <bpmn:outgoing>Flow_1o8s0a5</bpmn:outgoing>
  </bpmn:sendTask>
  <bpmn:endEvent id="Event_1y5imsf" name="EndEvent 28">
    <bpmn:incoming>Flow_1o8s0a5</bpmn:incoming>
  </bpmn:endEvent>
  <bpmn:sequenceFlow id="Flow_10pi8xx" sourceRef="StartEvent_0c9qnlk"
targetRef="Activity_04hluen" />
  <bpmn:sequenceFlow id="Flow_1o8s0a5" sourceRef="Activity_04hluen"
targetRef="Event_1y5imsf" />
  <sec:availabilityMF id="availabilityMF_467" secType="availability">
    <sec:enfBy></sec:enfBy>
    <sec:enfBy></sec:enfBy>
    <sec:level>0</sec:level>
  </sec:availabilityMF>
  <sec:integrityMF id="integrityMF_2088" secType="integrity">
    <sec:enfBy></sec:enfBy>
    <sec:enfBy></sec:enfBy>
  </sec:integrityMF>
</bpmn:process>
<bpmn:process id="Process_0jaus41">
  <bpmn:startEvent id="Event_0ldt097" name="StartEvent 25">
    <bpmn:outgoing>Flow_02i7fbr</bpmn:outgoing>
  </bpmn:startEvent>
  <bpmn:endEvent id="Event_12jk32m" name="EndEvent 29">
    <bpmn:incoming>Flow_1eazv3v</bpmn:incoming>
  </bpmn:endEvent>
  <bpmn:task id="Activity_1mjywp" name="Store Itinerary details">
    <bpmn:incoming>Flow_02i7fbr</bpmn:incoming>
    <bpmn:outgoing>Flow_1eazv3v</bpmn:outgoing>
    <bpmn:dataOutputAssociation id="DataOutputAssociation_0jl1qi9">
      <bpmn:targetRef>DataObjectReference_0f6utak</bpmn:targetRef>
    </bpmn:dataOutputAssociation>
  </bpmn:task>
  <bpmn:dataObjectReference id="DataObjectReference_0f6utak" name="Itinerary
details" dataObjectRef="DataObject_0fgl1fk" />
  <bpmn:dataObject id="DataObject_0fgl1fk" />
  <bpmn:sequenceFlow id="Flow_02i7fbr" sourceRef="Event_0ldt097"
targetRef="Activity_1mjywp" />
  <bpmn:sequenceFlow id="Flow_1eazv3v" sourceRef="Activity_1mjywp"
targetRef="Event_12jk32m" />
  <sec:integrityDO id="integrityDO_2076" secType="integrity">
    <sec:enfBy></sec:enfBy>
    <sec:enfBy></sec:enfBy>
  </sec:integrityDO>
</bpmn:process>

```

Security requirements are represented as explicit extension nodes (e.g., `integrityMF`, `availabilityMF`, `integrityDO`) and are bound to BPMN elements by `sec:securityAssociation` entries. This approach keeps security semantics explicit and structured, allowing parameterization, reuse, and multi-target associations and supports automated policy checks and downstream enforcement mapping.

The dataset includes canonical Security Business Process Model and Notation (SecBPMN) XML representations with consistent schema enforcement, paired natural-language descriptions aligned with annotated diagrams, broad security annotation coverage across multiple SecBPMN constructs, and explicit provenance metadata for all entries. These characteristics support both rigorous evaluation of the proposed pipeline and use as a reusable benchmark for the research community. Although the dataset is limited by its relatively small size and uneven domain distribution, these constraints are mitigated by the diversity of scenarios and reproducibility guarantees, which facilitate future extension.

5.2.2 Human Annotation Baseline

To establish a ground truth reference and contextualize automated generation performance, a controlled human annotation baseline was conducted. This baseline provides both the reference dataset for automated systems and a benchmark for comparative analysis. Three domain experts with documented experience in SecBPMN modeling (designated as Tester1, Tester2, and Tester3) were recruited for the annotation task. Each annotator received a curated set of process descriptions from the dataset introduced in Section 5.2.1, along with the corresponding BPMN 2.0 structural models (participants, activities, events, gateways, flows). Annotators were instructed to extract security constraints following the SecBPMN2 specification, identifying all applicable security annotations based solely on the textual descriptions and process structure.

A key methodological innovation in this annotation study was the use of two distinct presentation formats to investigate how information structure affects annotation quality and efficiency:

- **Format A (Plain paragraph):** Process descriptions presented as continuous natural-language paragraphs, mirroring typical requirement specifications. This format requires annotators to identify security-relevant information through implicit reasoning.
- **Format B (Structured views):** Process descriptions organized into three explicit views following the Socio-Technical Security Modeling Language (STS-ML) [Paja et al. \(2015\)](#) methodology: *social view* (stakeholders, goals, delegations, organizational constraints), *information view* (information entities, ownership,

composition hierarchies), and *authorization & security needs view* (permissions, scope, security requirements). This format provides explicit decomposition of socio-technical security concerns.

Each process description was assigned to annotators under one of the two formats according to a balanced design protocol. Annotators were provided with the SecBPMN2 Reference Guide [STS-Tool Development Team \(2015a\)](#) and a brief tutorial on constraint types and parameter requirements before beginning the task. The annotation protocol specified two explicit tasks for each process description: (i) identify security requirements from the text, and (ii) map each requirement to the appropriate BPMN element type (Activity, Data Object, Message Flow, Gateway, or Participant).

Annotation Effort and Timing

Annotation timing data was systematically collected to quantify human effort requirements and establish efficiency baselines. The data, summarized in Table 5.2, reveals substantial variation in annotation duration across process complexity and presentation format.

Annotator	Models Annotated	Avg. Time (min)
Tester1	7	7.5
Tester2	10	7.5
Tester3	10	11.5
Overall	26	9.0

Table 5.2: Annotation effort distribution across domain experts.

The recorded timing data shows annotation durations ranging from 2 minutes for simple delegation workflows to 21 minutes for complex multi-participant scenarios. Analysis across the two presentation formats indicates comparable efficiency: Format A required an average of 8.2 minutes per process (based on 13 annotations), while Format B required 9.8 minutes per process (based on 13 annotations). This modest difference suggests that while structured presentation may improve annotation consistency, it introduces a relatively small overhead in annotation time. The average annotation duration across all recorded annotations was 9.0 minutes per process description, demonstrating the significant time investment required for expert-level security annotation.

To contextualize this baseline efficiency, one must consider that each annotator processed, on average, 27 process descriptions, requiring approximately 3.9 hours of focused

annotation work per expert. This substantial human investment emphasizes the practical value of automating the security annotation process while maintaining annotation quality.

Inter-Annotator Agreement

To establish quality benchmarks and assess annotation consistency, a subset of eight process models were annotated by multiple domain experts, creating overlap annotations for inter-annotator reliability analysis. These overlapping models enabled quantitative assessment of annotation variability through standard agreement metrics.

Inter-annotator agreement was quantified using two complementary metrics. The Jaccard Index measures the proportion of shared security constraints relative to the total unique constraints identified by either annotator:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{\text{intersection}(A, B)}{\text{union}(A, B)} \quad (5.1)$$

where A and B represent the sets of security constraints identified by annotators A and B , respectively.

Cohen’s Kappa evaluates agreement beyond chance, accounting for random agreement probability:

$$\kappa = \frac{P_o - P_e}{1 - P_e} \quad (5.2)$$

where P_o is the observed proportion of agreement (both assigning or both not assigning a constraint) and P_e is the expected proportion of agreement by chance, calculated as the marginal probability product across both annotators’ assignment distributions.

Process Model	Annotators	Jaccard	Cohen's κ
Credit card verified process	T1 vs T2	0.500	0.333
	T1 vs T3	0.333	-0.333
	T2 vs T3	0.600	0.333
RBT negotiation process model	T1 vs T2	0.083	-0.179
Healthcare business process	T1 vs T2	0.455	-0.179
Flight ticket booked delegation	T1 vs T2	0.714	0.000
Amadeus service produces flight tickets	T1 vs T2	0.429	0.000
Prepayment made process	T2 vs T3	0.600	-0.250
Itinerary details transmission	T2 vs T3	0.250	-0.174
Hotel booked delegation	T2 vs T3	0.167	-0.667
<i>Summary Statistics</i>			
Mean	–	0.413	-0.111
Standard Deviation	–	± 0.193	± 0.285
Range	–	[0.083, 0.714]	[-0.667, 0.333]
Total comparisons	10	–	–

Table 5.3: Inter-annotator agreement metrics across overlapping SecBPMN process models. T1, T2, T3 denote Tester1, Tester2, and Tester3 respectively.

Table 5.3 presents the pairwise agreement metrics across eight overlapping process models, yielding 10 annotator pairs. The mean Jaccard Index of 0.413 ± 0.193 indicates moderate overlap in constraint identification, with substantial variability across model complexity (range: 0.083–0.714). The best pairwise agreement (Jaccard = 0.714) was achieved for the *flight ticket booked delegation* model, while the lowest agreement (Jaccard = 0.083) occurred for the more complex *RBT negotiation process model*, reflecting the greater interpretational diversity in security requirement specification for intricate multi-participant processes.

Cohen's Kappa, which accounts for chance agreement, yielded a mean of -0.111 ± 0.285 (range: -0.667 to 0.333). Negative Kappa values indicate that observed agreement fell below chance expectation for several model pairs, suggesting divergent annotation strategies among annotators. This result highlights the inherent subjectivity in security requirement interpretation and the absence of deterministic annotation guidelines, despite shared domain expertise.

The observed inter-annotator variance establishes an empirical upper bound on achievable annotation consistency for this task. Automated approaches that achieve agree-

ment levels approaching the mean Jaccard Index (0.413) can be considered competitive with expert manual annotation, while methods significantly exceeding this threshold suggest potential over-fitting to specific annotator biases. For the remaining process models in the dataset, each description was assigned to a single annotator and subsequently validated by a SecBPMN expert to ensure schema compliance and parameter completeness in the ground-truth reference set.

5.2.3 Security Label Extraction Paradigms and Prompting Strategies

To assess the capacity of large language models (LLMs) to infer, structure, and formalize security annotations from natural-language process descriptions, three complementary paradigms were experimentally evaluated: (i) *direct LLM-based generation* with multiple prompting strategies, (ii) *retrieval-augmented generation (RAG)*, and (iii) *Socio-Technical Security Modeling Language (STS-ML) guided extraction*. All three were implemented within a unified security annotations extraction pipeline sharing identical pre-processing, decoding, and post-processing validation mechanisms. This comparative design isolates the impact of internal reasoning, external knowledge grounding, and systematic modeling methodology on the quality and validity of generated annotations.

General LLM-Based Generation

The generation paradigm measures an LLM’s ability to infer and structure security constraints without external grounding. Each prompt defines the SecBPMN extraction task, outlining mapping rules between textual elements and BPMN targets (Activity, Data Object, or Message Flow) and specifying the allowed security types (e.g., confidentiality, integrity, availability, accountability, authenticity, and non-repudiation). The model is instructed to output a structured JSON object under a single top-level key, `securityConstraints`, conforming to a predefined schema that enumerates the expected attributes and value types. An example of this schema and its typical structure is illustrated in listing 4.1, which demonstrates how the extractor encodes multiple security constraints, each linked to its corresponding BPMN element and annotated with the relevant `secType` and enforcement fields. A schema validator enforces strict structural compliance, while an automatic repair mechanism ensures valid JSON serialization if formatting deviations occur.

All prompts were constructed with an explicit *persona instruction* that frames the model as a "*professional Security Requirements Engineer and SecBPMN expert*" responsible for identifying and extracting relevant security constraints. This persona

guidance consistently steers the model’s generation behavior toward analytical, schema-compliant responses while mitigating stylistic drift and verbosity.

Five LLM-based prompting strategies were evaluated, each designed to elicit different reasoning behaviors and contextual dependencies:

- **Zero-shot:** A concise, instruction-only template defining the SecBPMN extraction objective, target mapping rules (Activity/Data Object/Message Flow), permissible types, required fields, and a strict directive to "return only valid JSON." No examples are provided; the model relies solely on schema and rule-based guidance.
- **One-shot:** Extends the zero-shot structure with one worked example that demonstrates a complete security labels extraction from a textual process description to the correct JSON schema. This configuration provides minimal guidance while preserving generality.
- **Few-shot:** Incorporates multiple curated examples covering a diverse range of constructs (activities, data objects, and message flows) and categories such as confidentiality, integrity, and availability. The examples serve as static in-context demonstrations, improving both recall and consistency by illustrating what to extract and how to format it.
- **Chain-of-thought (CoT):** Adds an explicit six-step reasoning framework that decomposes the extraction task into structured analytical stages: (i) *Context Initialization*: identifying domain and process boundary, (ii) *Social View reasoning*: enumerating actors, roles, and interactions, (iii) *Information View reasoning*: tracking data flows and document exchanges, (iv) *Authorization View reasoning*: determining access constraints, (v) *Security Derivation*: mapping security needs to constraint types, and (vi) *Cross-View Consistency Validation*: ensuring actor, information, and authorization consistency across all derived constraints. The model performs this reasoning before emitting the final JSON output, enhancing interpretability and logical coherence.
- **Hybrid:** Combines the multiple exemplars of the few-shot approach with the systematic six-step reasoning framework of CoT. The model is guided through structured analytical reasoning while being provided with multiple in-context examples, creating a balanced configuration that benefits from both example-driven learning and structured analytical thinking.

All LLM-based experiments were executed with deterministic decoding (temperature = 0.1, maximum output length = 25000 tokens). Model responses were requested

in JSON mode, and any structural inconsistencies triggered a repair cycle that reformatted the response while preserving the original semantics. This combination of controlled temperature, persona consistency, and robust validation yielded reproducible, schema-conformant outputs across all strategies.

Retrieval-Augmented Generation (RAG)

The retrieval-augmented generation paradigm enriches the process description with contextual knowledge retrieved from a curated SecBPMN knowledge base. The knowledge base integrates three authoritative sources: the *SecBPMN2 Reference Guide* [STS-Tool Development Team \(2015a\)](#), the *Socio-Technical Security Modeling Language (STS-ML) Manual* [STS-Tool Development Team \(2015b\)](#), and the *SecBPMN2BC paper* [Köpke et al. \(2023\)](#). These documents undergo hierarchical sectioning to preserve structural relationships, followed by sentence-aware chunking at 600 tokens per chunk with 300-token overlap to maintain semantic continuity. Each chunk is enriched with metadata including section hierarchy (`section_path`), security constraint classifications (`label_types`), permitted attachment targets (`allowed_targets`), and required parameters (`required_props`).

Retrieval operates through a hybrid search mechanism combining BM25 lexical matching and FAISS semantic search. BM25 retrieves $k \times 5 = 80$ candidate chunks per source using NLTK word tokenization, while FAISS retrieves an equivalent candidate pool using OpenAI’s `text-embedding-3-small` embeddings (1536 dimensions) and cosine similarity. Scores from both methods are independently normalized via min-max scaling, then fused with $\alpha = 0.35$ to favor semantic over lexical matching:

$$\text{score}_{\text{final}} = 0.35 \times \text{BM25}_{\text{norm}} + 0.65 \times \text{FAISS}_{\text{norm}}.$$

Retrieval operates on a fixed `top_k = 16` chunks, distributed across documents according to a 70%–20%–10% weighting for SecBPMN2, SecBPMN2BC, and STS-ML respectively, ensuring primary reliance on specification material while incorporating complementary conceptual and business-component context. Retrieved passages are deduplicated by `chunk_id`, trimmed to a maximum of 1500 characters per chunk to fit within token budgets, and formatted as a structured contextual block in the LLM prompt. The prompt template explicitly instructs the model to generate constraints only when supported by retrieved evidence, mandating each constraint include a `trace` field containing `evidence` (with `chunk_id`, `doc_type`, and `section_path`) and a brief `rationale` linking the constraint to the cited source. This traceability mechanism prevents hallucination and ensures domain grounding.

The RAG prompt template mandates evidence traceability: each generated constraint must include a `trace` object containing at least one `evidence` citation with `chunk_id`, `doc_type`, and `section_path`, plus a concise `rationale` (≤ 2 sentences) explicitly link-

ing the constraint to the cited source. This requirement enforces domain grounding and eliminates unsupported inferences. Generation uses identical decoding parameters to direct LLM methods (temperature = 0.1, `max_tokens` = 25,000) with JSON mode enabled. A repair cycle handles schema violations by re-invoking the model with relaxed JSON constraints when initial parsing fails. The experimental evaluation measures both the accuracy of generated constraints and the correctness of their evidence traces against manual ground-truth annotations.

STS-ML Guided Extraction

The STS-ML guided extraction implements a structured multi-stage methodology applying Socio-Technical Security Modeling Language principles to derive security requirements systematically [Paja et al. \(2015\)](#). Unlike direct LLM generation, this approach constructs three interconnected views before deriving requirements.

The pipeline executes five sequential stages, each producing structured intermediate outputs:

- **View Generation** constructs the social view (stakeholders, goals, delegations, document transmissions, organizational constraints), the information view (information entities, ownership, composition hierarchies), and the authorization view (permissions, scope, transferability) from the natural-language description. Temperature is set to 0.3 and `max_tokens` = 10,000 to encourage diverse perspective in generation.
- **Security Analysis** validates well-formedness (structural consistency, absence of cycles, valid ownership), detects security violations (no-delegation conflicts, authorization inconsistencies, organizational constraint violations), and performs risk propagation (threat propagation through delegation chains and information dependencies). This stage uses temperature 0.1 and `max_tokens` = 8,000 for deterministic evaluation.
- **View Refinement** applies fixes when analysis reveals violations or inconsistencies, adjusting the three views to resolve identified issues while preserving business intent. Refinement executes only when the analysis status is not PASS (`max_tokens` = 10,000, temperature 0.1).
- **Security Requirements Generation** transforms the refined views into formal security requirements using STS-ML rules (e.g., the opposite-direction rule for security needs over delegations), producing structured requirement tuples (responsible party, requirement, requester) with explicit descriptions (`max_tokens` = 10,000, temperature 0.1).

- **Label Extraction** maps STS-ML requirements to SecBPMN constraints (e.g., no-delegation \rightarrow `NonDelegationAct`, integrity-of-transmission \rightarrow `IntegrityMF`, separation-of-duties \rightarrow `SeparationOfDutiesORG`) using schema-aware translation (`max_tokens` = 30,000, temperature 0.1).

Each stage operates on JSON inputs/outputs with retry logic (max 3 attempts), JSON repair mechanisms for parsing failures, and fallback behaviors when LLM calls fail. The pipeline accumulates metrics across all five stages, tracking input/output tokens, latency, and costs per stage. The experimental evaluation compares STS-ML derived constraints against manual ground-truth annotations to assess the methodology’s accuracy, completeness, and structural consistency.

Comparing these three paradigms enables a controlled analysis across reasoning-driven, knowledge-driven, and methodology-driven generation strategies. Direct LLM prompting evaluates whether pretrained models encode sufficient conceptual priors for SecBPMN annotation, RAG quantifies the benefits of contextual grounding on formal specifications and curated examples, while STS-ML guided extraction assesses whether structured view-based analysis provides superior systematic reasoning and domain coverage. All approaches were assessed across multiple dimensions: syntactic validity (schema compliance, well-formedness), semantic accuracy (constraint correctness and parameter completeness), and constraint completeness (recall of security annotations relative to ground truth), as well as computational efficiency metrics including latency, token consumption, and operational costs. Detailed experimental results comparing all three paradigms across these metrics are presented in Section 5.3.

5.2.4 Model Families

Two representative large language models were selected to capture diversity in provider ecosystems, architectural design, and operational characteristics:

- **GPT-4.1-mini** (OpenAI), a compact model from the GPT-4 family designed for efficient structured output generation while maintaining strong reasoning capabilities.
- **Mistral-small-latest** (Mistral AI), a lightweight, open-weight model optimized for deployment efficiency, providing insights into cost-effective extraction scenarios.

These model selections span three major provider ecosystems (OpenAI, and Mistral AI), offering diverse architectural approaches including Transformer-based variants and

cutting-edge multimodal capabilities. For experimental comparability, all models were evaluated under identical prompt formulations, decoding parameters, and output constraints, enabling direct performance comparison across provider boundaries. This diversity ensures that findings generalize beyond single-vendor solutions and reflect real-world deployment scenarios where model choices may be constrained by cost, latency, or policy requirements.

5.2.5 Evaluation Methods

Evaluating the proposed pipeline requires more than verifying that it produces syntactically valid SecBPMN constraints; it must also be assessed on how faithfully and comprehensively it captures the intended security requirements expressed in natural language. Building on the dataset and pipeline architecture described in the previous sections, this part of the methodology details the protocol through which system outputs are tested and compared against reference annotations.

The evaluation framework pursues three complementary objectives. First, it measures the accuracy, coverage, and structural validity of automatically extracted annotations and annotations after mapping phase relative to ground truth. Second, it incorporates semantics-aware matching to move beyond surface-form comparisons, thereby ensuring robustness against lexical variation and paraphrasing in task names. Third, it benchmarks performance consistently across multiple generation paradigms, model families, and prompting strategies, providing a controlled basis for comparison. By combining objective metrics with semantics-aware alignment and rubric-driven qualitative assessment, the evaluation protocol offers a comprehensive, reproducible, and interpretable account of system performance. This dual perspective ensures that both measurable extraction quality and nuanced semantic fidelity are captured, enabling a fair comparison across experimental conditions.

5.2.6 Experimental Design

The experimental design establishes the conditions, reference data, and comparison procedure used to evaluate the pipeline. It consists of three main components: generation paradigms and model families, the construction of ground truth, and the semantics-aware matching procedure that enables objective comparisons.

Three generation paradigms are considered. The first is a *general LLM method*, where security constraints are generated from textual descriptions using different prompting strategies. The second is a *retrieval-augmented generation (RAG) method*, where process descriptions are enriched with contextual knowledge retrieved from a curated SecBPMN knowledge base before generation. And the third is the *STS-ML guided*

extraction, which systematically constructs security requirements through multi-view modeling and security analysis phases. This design allows us to contrast unconstrained generation with knowledge-grounded and methodology-driven approaches.

For the general LLM method, we investigate a variety of prompting strategies: zero-shot, one-shot, few-shot, chain-of-thought reasoning, and hybrid approaches combining in-context examples with structured reasoning steps. These strategies reflect common approaches to steering large language models in structured prediction tasks.

We test two representative model families from current LLM ecosystems: **GPT-4.1-mini** (OpenAI), and **Mistral-small-latest** (Mistral AI). Together they provide coverage of different architectural choices and provider ecosystems, enabling a fair assessment of portability and robustness across model families.

Ground truth annotations are derived from the curated dataset of SecBPMN models introduced in subsection 5.2.1. Each process description is paired with a canonical SecBPMN JSON representation reconstructed from SecBPMN diagrams. To ensure comparability, we normalize all entries by canonicalizing `secType` values (e.g., `integrity`, `confidentiality`, `availability`) and element kinds (Activity, Data Object, Message Flow). This canonicalization guarantees that variations in naming conventions do not confound evaluation while still preserving the intended security semantics.

Objective comparison between generated and reference annotations is enabled by a semantics-aware matching procedure using a sentence-transformer model. Three rules govern this process:

1. **Type-level canonicalization:** Security categories (`secType`) and element types (Activity, DO, MF) must match exactly between generated and reference annotations. Suffix normalization (e.g., `Act`, `MF`, `GW`, and `ORG`) ensures type-level consistency.
2. **Task-name semantic similarity:** To account for lexical variation and paraphrasing, task names are normalized by removing punctuation and stop words, then embedded using a lightweight sentence-transformer model (*all-MiniLM-L6-v2*) for 384-dimensional semantic embeddings. Similarity is computed via cosine similarity between embeddings.
3. **Two-stage matching:** A constraint pair is considered a *strong match* if their task names exceed a semantic similarity threshold of 0.5. For weaker semantic associations, a secondary *weak threshold* of 0.3 is applied, augmented by token-level Jaccard similarity as a fallback criterion when semantic similarity alone is insufficient.

This matching strategy robustly handles paraphrasing, synonymy, and minor lexical variations while maintaining the requirement that matched constraints must agree on both security type and element kind.

5.2.7 Evaluation Metrics

Having defined the experimental conditions, we now turn to the metrics used to evaluate the quality of automatically generated SecBPMN annotations. The evaluation is designed to capture not only correctness in terms of matches with the reference dataset, but also structural validity, semantic adequacy, and robustness against over- or under-generation. To this end, we adopt a dual approach: (i) *objective metrics*, computed from semantics-aware alignments between generated and ground-truth constraints, and (ii) *qualitative metrics*, provided by an LLM-as-a-judge framework that scores outputs according to a rubric. This subsection introduces the first group of measures.

5.2.7.1 Objective Metrics

Objective metrics are computed after applying the semantics-aware matching procedure described earlier. They quantify performance in terms of accuracy, coverage, overlap, structural validity, and error tendencies. Let R denote the set of ground-truth constraints, G the set of generated constraints, and M the set of matched pairs between them. We denote the number of elements in a set X by $|X|$.

- **Precision** measures the proportion of generated constraints that are correct:

$$\text{Precision} = \frac{|M|}{|G|}. \quad (5.3)$$

- **Recall** measures the proportion of ground-truth constraints that were recovered:

$$\text{Recall} = \frac{|M|}{|R|}. \quad (5.4)$$

- **F1-score** is the harmonic mean of precision and recall:

$$\text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2 \cdot |M|}{|R| + |G|}. \quad (5.5)$$

- **Jaccard Index** measures the set overlap between generated and reference constraints:

$$\text{Jaccard} = \frac{|M|}{|R \cup G|} = \frac{|M|}{|R| + |G| - |M|}. \quad (5.6)$$

- **Completeness** expresses the fraction of ground-truth constraints that were covered:

$$\text{Completeness} = \frac{|M|}{|R|}. \quad (5.7)$$

- **Schema Validity** measures the proportion of generated constraints conforming to valid (`secType`, `type`) pairs as defined by the SecBPMN schema:

$$\text{Schema Validity} = \frac{|\{g \in G \mid g \text{ is valid}\}|}{|G|}. \quad (5.8)$$

- **Spurious Rate** quantifies over-generation by counting unmatched generated constraints:

$$\text{Spurious Rate} = \frac{|G| - |M|}{|G|}. \quad (5.9)$$

- **Redundancy Rate** measures the proportion of duplicate constraints in the generated set:

$$\text{Redundancy Rate} = \frac{|\{g \in G \mid g \text{ is duplicate}\}|}{|G|}. \quad (5.10)$$

- **Average Task Similarity** evaluates semantic similarity between task names in matched pairs using BERT embeddings and cosine similarity:

$$\text{Avg. Similarity} = \frac{1}{|M|} \sum_{(r,g) \in M} \cos(e(r), e(g)), \quad (5.11)$$

where $e(\cdot)$ denotes the embedding of a task name.

In addition to these aggregate measures, recall is also reported separately for each of the nine SecBPMN constraints we introduced in the table 2.1. Finally, macro-averaged recall is computed as the unweighted mean across categories to ensure that performance is not dominated by the most frequent constraint types.

5.2.7.2 Qualitative Metrics (LLM-as-a-Judge)

While objective metrics provide reproducible measures of extraction quality, they are limited by the rigidity of exact or semantics-aware matching. To complement them, we employ a rubric-driven evaluation where a large language model (LLM) acts as a judge. The judge receives both the reference and generated security constraints in their original and normalized forms, and returns three scores in the range $[0, 1]$, together with a concise textual explanation. For all experiments, we use **GPT-4.1-mini** as the evaluation model, configured with deterministic decoding ($T = 0.0$) to eliminate sampling variance and ensure consistent judgments.

- **Accuracy** ($\in [0, 1]$) reflects how well the generated constraints avoid spurious or irrelevant annotations. It is conceptually analogous to precision: higher scores indicate fewer false positives.
- **Completeness** ($\in [0, 1]$) measures whether all security requirements present in the reference appear in the generated output. It is analogous to recall: higher scores indicate better coverage of the ground truth.
- **Correctness** ($\in [0, 1]$) evaluates the fidelity of matched constraints with respect to element kinds (Activity, Data Object, Message Flow), naming conventions, and key properties such as enforcement mechanisms, authorized users, or execution attributes. It isolates errors that may not be captured by accuracy or completeness alone.
- **Reasoning** (textual output) provides a concise justification of the scores, highlighting specific strengths and weaknesses in the generated annotations. This qualitative feedback aids interpretability and helps identify systematic error patterns.

Together, these metrics disentangle three complementary dimensions of quality: accuracy penalizes over-generation, completeness penalizes under-generation, and correctness captures fine-grained fidelity. The additional reasoning output provides transparency that purely numeric scores cannot convey.

To assess the reliability of LLM as a judge, we compute correlations between the rubric-based scores (accuracy, completeness, correctness) and the objective metrics described previously (precision, recall, F1-score). The choice of GPT-4.1-mini is motivated by its state-of-the-art reasoning capabilities, and robustness under deterministic decoding. These properties make it a suitable candidate for rubric-driven evaluation in a domain where subtle mismatches (e.g., between activity types, data objects, or enforcement properties) can significantly affect correctness.

Correlations are computed using both the *Pearson coefficient*, which measures linear association, and the *Spearman rank correlation*, which captures monotonic relationships independent of scale. Formally, for two metrics X and Y measured across n evaluation instances, the **Pearson correlation coefficient** is defined as:

$$\rho_{\text{Pearson}}(X, Y) = \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^n (X_i - \bar{X})^2} \sqrt{\sum_{i=1}^n (Y_i - \bar{Y})^2}}, \quad (5.12)$$

where \bar{X} and \bar{Y} denote the sample means of X and Y , respectively.

The **Spearman rank correlation** is obtained by applying the Pearson correlation to the rank-transformed variables of X and Y :

$$\rho_{\text{Spearman}}(X, Y) = \rho_{\text{Pearson}}(\text{rank}(X), \text{rank}(Y)). \quad (5.13)$$

By merging judge scores with objective metrics on shared identifiers, we evaluate correlations both globally and per-model. Empirically, GPT-5’s *accuracy* score aligns most strongly with F_1 and precision, confirming its ability to detect spurious constraints. *Completeness* correlates with recall, reflecting coverage of ground-truth constraints, while *correctness* provides complementary insights by penalizing schema or type mismatches. These results justify GPT-5 as a reliable qualitative evaluator whose rubric-based scores meaningfully complement traditional metrics.

5.2.8 Efficiency Benchmark

In addition to evaluating the quality of generated security constraints, we assess the efficiency of different model configurations. Since commercial LLMs are billed in tokens and exhibit heterogeneous latency profiles, efficiency provides a complementary dimension for comparing models beyond accuracy. To ensure fairness and reproducibility, we focus on provider-independent measures: token counts and wall-clock latency, while explicitly omitting monetary costs to avoid price-model confounds.

Efficiency is measured using a lightweight benchmarking harness. For each model call, we record:

- **Input tokens** (T_{in}): the number of tokens consumed by the input prompt, estimated using provider-specific tokenizers where available (e.g., `tiktoken` for GPT models, or heuristic approximations otherwise).

- **Output tokens** (T_{out}): the number of tokens produced in the generated response.
- **Total tokens** ($T_{\text{out}} + T_{\text{in}}$): total number of tokens produced in the generated response.
- **Latency** (L): the wall-clock time, in seconds, between sending the request and receiving the complete response.

Formally, each evaluation instance i yields an efficiency tuple:

$$E_i = (T_{\text{in}}^{(i)}, T_{\text{out}}^{(i)}, L^{(i)}).$$

Aggregate statistics (mean, variance, and distribution plots) are then computed across tasks, enabling cross-model comparison of efficiency under matched experimental conditions. For example, models with similar accuracy but substantially lower $T_{\text{in}} + T_{\text{out}}$ or latency are preferable in practical deployment scenarios where throughput or inference cost is constrained.

This benchmarking protocol ensures that efficiency results are directly comparable across LLM families (GPT, Mistral), regardless of provider-specific pricing or service-level optimizations. It complements the quality-oriented metrics by providing a holistic view of trade-offs between performance and resource usage.

5.3 Results

5.3.1 Classification Performance

This section analyzes the classification performance of SecBPMN-GPT using standard metrics such as F1-score, Precision, and Recall—across process complexities (Simple, Medium, Complex). Results are presented for both pre- and post-mapping stages to assess the effect of schema validation, comparing GPT-4.1-mini and Mistral-small under three annotation strategies: Hybrid Prompt, RAG, and the STS-ML guided extraction.

Pre-Mapping Performance

Table 5.4 illustrates baseline scores prior to schema mapping. GPT-4.1-mini outperformed Mistral-small across methods and complexity levels. RAG yielded the highest F1 in most configurations (Simple: 0.606; Medium: 0.315; Complex: 0.341). Hybrid Prompting exhibited higher recall (Simple: 0.720; Medium: 0.337) with comparatively lower precision, whereas RAG showed a more balanced precision–recall profile.

BPMN Complexity	Method	GPT-4.1-mini			Mistral-small		
Simple	Hybrid Prompt	0.518	0.433	0.720	0.375	0.306	0.523
	RAG	0.606	0.620	0.614	0.498	0.554	0.508
	STS-ML	0.054	0.031	0.227	0.182	0.111	0.568
Medium	Hybrid Prompt	0.303	0.314	0.337	0.347	0.386	0.383
	RAG	0.315	0.392	0.306	0.256	0.318	0.236
	STS-ML	0.095	0.070	0.207	0.120	0.101	0.268
Complex	Hybrid Prompt	0.295	0.382	0.253	0.250	0.271	0.333
	RAG	0.341	0.431	0.306	0.221	0.249	0.203
	STS-ML	0.188	0.117	0.490	0.267	0.201	0.436

Table 5.4: Classification performance (F1, Precision, Recall) by method, model, and process complexity (Pre-Mapping).

The STS-ML guided extraction produced low F1-scores (≤ 0.18 in most settings; max 0.188) with sporadically higher recall (e.g., 0.490 for GPT on Complex) and low precision, indicating over-generation. Performance decreased with increasing complexity across methods. Pre-mapping results demonstrate that LLM-based approaches substantially outperform deterministic baselines, with RAG–GPT showing the best overall balance between recall and precision. However, notable variation across complexity levels highlights the need for structural validation to mitigate noise and incomplete annotations prior to final schema generation.

Post-Mapping Performance

Following post-mapping schema enforcement, scores increased for most configurations (Table 5.5). For GPT-4.1-mini, RAG achieved the highest overall F1 on Simple (0.715), while Hybrid Prompting achieved the highest recall (0.788). Mistral-small showed moderate gains on Simple and Medium; precision remained below GPT’s.

BPMN Complexity	Method	GPT-4.1-mini			Mistral-small		
		F1	Precision	Recall	F1	Precision	Recall
Simple	Hybrid Prompt	0.618	0.610	0.788	0.397	0.333	0.548
	RAG	0.715	0.755	0.712	0.571	0.684	0.561
	STS-ML	0.129	0.089	0.296	0.251	0.168	0.553
Medium	Hybrid Prompt	0.370	0.391	0.418	0.371	0.390	0.455
	RAG	0.346	0.445	0.323	0.320	0.386	0.325
	STS-ML	0.157	0.129	0.233	0.182	0.165	0.307
Complex	Hybrid Prompt	0.295	0.407	0.243	0.258	0.296	0.316
	RAG	0.366	0.451	0.330	0.179	0.209	0.159
	STS-ML	0.086	0.055	0.208	0.167	0.126	0.259

Table 5.5: Classification performance (F1, Precision, Recall) by method, model, and process complexity (Post-Mapping).

In contrast, RAG underperformed on complex processes for Mistral (F1 = 0.179), indicating sensitivity to context window limitations during schema alignment. The STS-ML guided extraction continued to show the weakest performance across all settings, reaffirming the importance of LLM-based contextual understanding for accurate constraint identification. The post-mapping phase demonstrated measurable improvements in consistency, precision, and structural validity. The results confirm that coupling LLM reasoning with rule-based schema enforcement not only refines output quality but also ensures that automatically generated SecBPMN annotations remain syntactically valid and semantically coherent across process complexities.

Comparison between Pre-Mapping and Post-Mapping

Table 5.6 summarizes F1 deltas. The largest gain was for GPT-4.1-mini with RAG on Simple (+0.109 to 0.715); Hybrid on Simple rose by +0.100 (to 0.618). Medium showed moderate increases (+0.03 to +0.07), while Complex gains were smaller; for Mistral-small with RAG on Complex, F1 decreased (−0.042). Across both models, post-mapping was associated with improved balance between precision and recall on Simple and Medium.

BPMN Complexity	Method	GPT-4.1-mini			Mistral-small		
		<i>Pre</i>	<i>Post</i>	Δ	<i>Pre</i>	<i>Post</i>	Δ
Simple	Hybrid Prompt	0.518	0.618	+0.100 \uparrow	0.375	0.397	+0.022 \uparrow
	RAG	0.606	0.715	+0.109 \uparrow	0.498	0.571	+0.073 \uparrow
	STS-ML	0.054	0.129	+0.075 \uparrow	0.182	0.251	+0.069 \uparrow
Medium	Hybrid Prompt	0.303	0.370	+0.067 \uparrow	0.347	0.371	+0.024 \uparrow
	RAG	0.315	0.346	+0.031 \uparrow	0.256	0.320	+0.064 \uparrow
	STS-ML	0.095	0.157	+0.062 \uparrow	0.120	0.182	+0.062 \uparrow
Complex	Hybrid Prompt	0.295	0.295	+0.000 \rightarrow	0.250	0.258	+0.008 \uparrow
	RAG	0.341	0.366	+0.025 \uparrow	0.221	0.179	-0.042 \downarrow
	STS-ML	0.188	0.086	-0.101 \downarrow	0.267	0.167	-0.100 \downarrow

Table 5.6: F1-score before vs. after post-mapping by method, model, and process complexity (absolute Δ). Upward arrows (\uparrow) denote improvement, downward arrows (\downarrow) indicate decline, and horizontal arrows (\rightarrow) represent no change.

Across both models, rule-based schema validation improved precision and recall balance, particularly for simple and medium domains where structural misalignments were most frequent prior to mapping.

The supplementary metrics in For GPT-4.1-mini (Table 5.7), the Jaccard Index increased (e.g., RAG-Simple +0.107 to 0.594), Completeness rose (+0.098 for RAG-Simple), and Spurious Rate decreased (Hybrid-Simple -0.177; RAG-Simple -0.134).

BPMN Complexity	Method	Jaccard Index			Completeness			Spurious Rate		
		<i>Pre</i>	<i>Post</i>	Δ	<i>Pre</i>	<i>Post</i>	Δ	<i>Pre</i>	<i>Post</i>	Δ
Simple	Hybrid Prompt	0.388	0.464	+0.076 \uparrow	0.720	0.788	+0.068 \uparrow	0.567	0.390	-0.177 \downarrow
	RAG	0.487	0.594	+0.107 \uparrow	0.614	0.712	+0.098 \uparrow	0.380	0.246	-0.134 \downarrow
	STS-ML	0.028	0.072	+0.044 \uparrow	0.227	0.296	+0.069 \uparrow	0.969	0.617	-0.352 \downarrow
Medium	Hybrid Prompt	0.202	0.255	+0.053 \uparrow	0.337	0.418	+0.081 \uparrow	0.686	0.589	-0.097 \downarrow
	RAG	0.217	0.246	+0.029 \uparrow	0.306	0.323	+0.017 \uparrow	0.608	0.548	-0.060 \downarrow
	STS-ML	0.052	0.091	+0.039 \uparrow	0.207	0.233	+0.026 \uparrow	0.930	0.618	-0.312 \downarrow
Complex	Hybrid Prompt	0.181	0.180	-0.001 \downarrow	0.253	0.243	-0.010 \downarrow	0.618	0.579	-0.039 \downarrow
	RAG	0.215	0.244	+0.029 \uparrow	0.306	0.330	+0.024 \uparrow	0.569	0.549	-0.020 \downarrow
	STS-ML	0.107	0.048	-0.059 \downarrow	0.490	0.208	-0.282 \downarrow	0.884	0.640	-0.244 \downarrow

Table 5.7: Structural quality metrics before vs. after post-mapping (Jaccard, Completeness, Spurious Rate) for GPT-4.1-mini.

Mistral-small (Table 5.8) showed similar but smaller changes (Simple/Medium Jaccard +0.033–0.056; Completeness +0.025–0.089); in Complex RAG, Jaccard decreased (–0.027) and Spurious increased (+0.040).

BPMN Complexity	Method	Jaccard Index			Completeness			Spurious Rate		
		<i>Pre</i>	<i>Post</i>	Δ	<i>Pre</i>	<i>Post</i>	Δ	<i>Pre</i>	<i>Post</i>	Δ
Simple	Hybrid Prompt	0.257	0.268	+0.011 \uparrow	0.523	0.548	+0.025 \uparrow	0.694	0.632	-0.062 \downarrow
	RAG	0.383	0.439	+0.056 \uparrow	0.508	0.561	+0.053 \uparrow	0.446	0.307	-0.139 \downarrow
	STS-ML	0.105	0.150	+0.045 \uparrow	0.568	0.553	-0.015 \downarrow	0.889	0.655	-0.234 \downarrow
Medium	Hybrid Prompt	0.248	0.281	+0.033 \uparrow	0.383	0.455	+0.072 \uparrow	0.614	0.595	-0.019 \downarrow
	RAG	0.175	0.224	+0.049 \uparrow	0.236	0.325	+0.089 \uparrow	0.682	0.592	-0.090 \downarrow
	STS-ML	0.067	0.108	+0.041 \uparrow	0.268	0.307	+0.039 \uparrow	0.899	0.588	-0.311 \downarrow
Complex	Hybrid Prompt	0.163	0.194	+0.031 \uparrow	0.333	0.316	-0.017 \downarrow	0.729	0.442	-0.287 \downarrow
	RAG	0.148	0.121	-0.027 \downarrow	0.203	0.159	-0.044 \downarrow	0.751	0.791	+0.040 \uparrow
	STS-ML	0.158	0.094	-0.064 \downarrow	0.436	0.259	-0.177 \downarrow	0.799	0.690	-0.109 \downarrow

Table 5.8: Structural quality metrics before vs. after post-mapping (Jaccard, Completeness, Spurious Rate) for Mistral-small.

Averaged across methods and complexities, F1 increased by $\approx +0.06$ to $+0.11$ for GPT-4.1-mini and $\approx +0.02$ to $+0.07$ for Mistral-small. Gains were largest on Simple and Medium, while Complex models showed smaller changes and occasional regressions.

Error Distribution and Confusion Analysis

To complement the quantitative classification metrics, this subsection analyzes confusion matrices to understand how errors such as *True Positives* (TP), *False Positives*

(FP), and *False Negatives* (FN) are distributed across methods and complexity levels before and after schema mapping. Figures 5.4–5.7 illustrate these distributions for both GPT-4.1-mini and Mistral-small models.

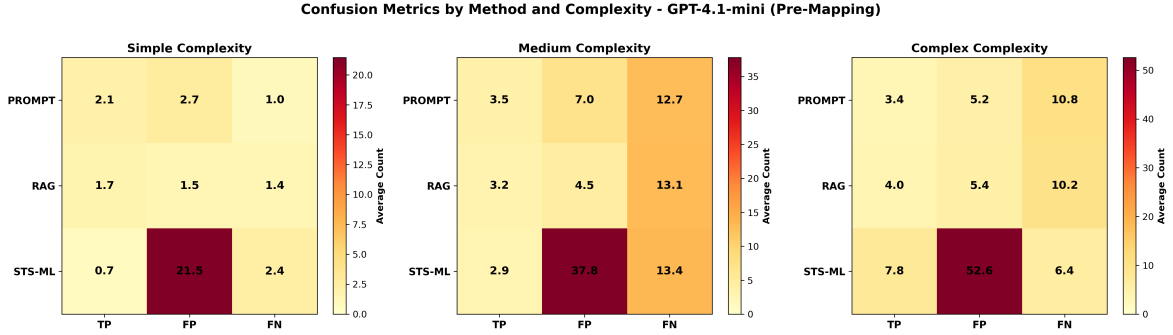


Figure 5.4: Pre-mapping confusion matrix for GPT-4.1-mini across methods and process complexities. Each cell represents the average counts of True Positives (TP), False Positives (FP), and False Negatives (FN).

Before schema validation, GPT-4.1-mini exhibited considerable variability in FP counts across methods and complexity levels (Figure 5.4). The STS-ML guided extraction produced the most significant over-prediction, with FP values reaching 21.5 in simple, 37.8 in medium, and up to 52.6 in complex scenarios. This reflects a strong tendency to over-generate security labels. In contrast, RAG and Hybrid Prompt (PROMPT) maintained lower FP counts (≈ 1.5 – 7.0) with moderate TP levels (≈ 2 – 4), indicating a more balanced trade-off between recall and precision. However, FN values (≈ 10 – 13 for medium and complex cases) suggest that even GPT struggles with recall in multi-pool or gateway-intensive processes where constraint propagation is difficult.

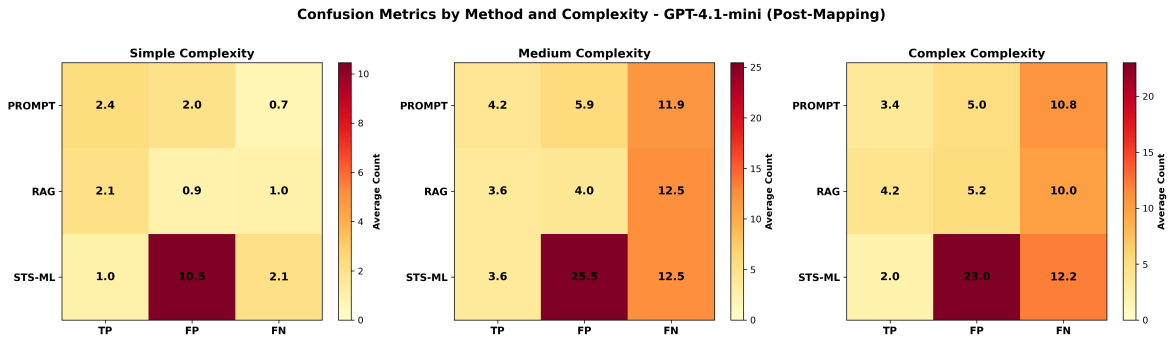


Figure 5.5: Post-mapping confusion matrix for GPT-4.1-mini across methods and process complexities.

After the schema mapping process, the error distribution changes markedly (Figure 5.5). The introduction of LLM-based mapping with structural validation reduces FP values by approximately **50–60%** across all complexity levels. For example, in complex tasks, STS-ML’s FP count drops from 52.6 to 23.0, and in simple workflows, from 21.5 to 10.5. True positives remain stable or slightly higher (e.g., RAG increases from 1.7 to 2.1 in simple cases), while FN values remain consistent, suggesting that the mapping phase effectively removes spurious predictions without reducing valid detections. RAG emerges as the most balanced strategy post-mapping, achieving the lowest FP levels (≈ 0.9 – 5.2) and stable TP–FN ratios, while Hybrid Prompt continues to prioritize recall (FN ≈ 11 – 12).

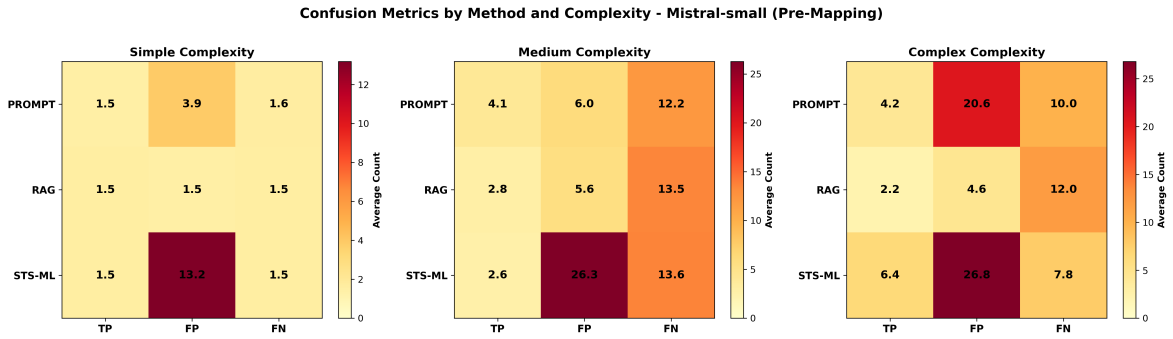


Figure 5.6: Pre-mapping confusion matrix for Mistral-small across methods and process complexities.

Mistral-small demonstrates more conservative annotation behavior than GPT-4.1-mini but still suffers from FP inflation before schema correction (Figure 5.6). The STS-ML baseline shows substantial FP values (≈ 13 – 27 across medium and complex processes), confirming its overgeneralization issue, while RAG and Hybrid Prompt (PROMPT) maintain moderate FP (≈ 3 – 6) with comparable TP levels (≈ 3 – 4). However, FN counts remain high (≈ 12 – 14), particularly in medium and complex scenarios, indicating that Mistral omits valid annotations when facing semantically intricate relationships or conditional security goals. Compared to GPT, Mistral’s pre-mapping performance is characterized by lower FP magnitude but higher FN, reflecting a more cautious generation style with reduced over-prediction but weaker coverage.

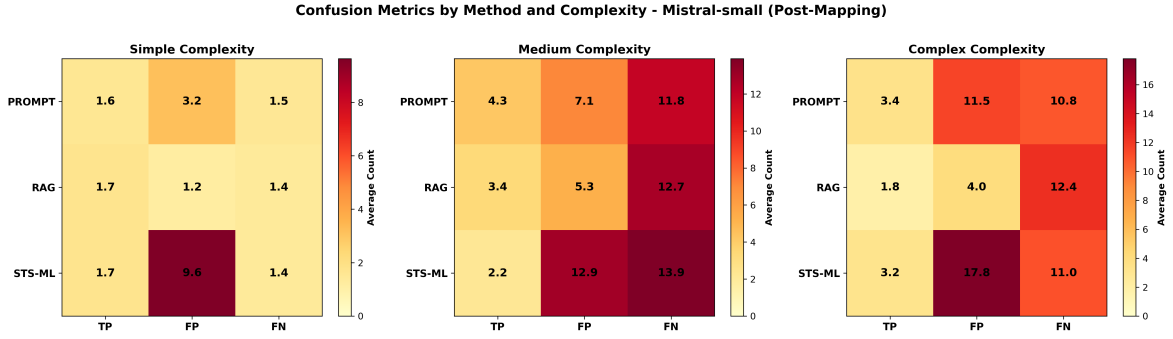


Figure 5.7: Post-mapping confusion matrix for Mistral-small across methods and process complexities.

After applying schema mapping, Mistral’s FP counts show a substantial decline, particularly in simple and medium processes (e.g., STS-ML FP drops from 13.2 to 9.6 in simple cases and from 26.3 to 12.9 in medium). RAG achieves the most balanced improvement, with FP reduced to nearly one per sample in simple scenarios while preserving recall levels. Nevertheless, FN rates remain relatively high (≈ 10 –13 in complex cases), indicating that while mapping enhances precision by filtering redundant outputs, it does not fully address omissions caused by incomplete semantic propagation. Overall, Mistral benefits from schema alignment through improved structural consistency and fewer redundant predictions, though its inherently cautious generation style limits recall gains.

Across both models, post-mapping validation demonstrates a clear corrective effect—reducing false positives by roughly **48% on average** while maintaining or slightly improving true positive rates. GPT-4.1-mini exhibits greater responsiveness to mapping, with more substantial FP suppression and stable recall, whereas Mistral-small achieves precision stability but retains higher FN rates. RAG consistently provides the best balance between precision and recall, Hybrid Prompt favors recall but risks minor FP persistence, and STS-ML remains the most error-prone method. These findings confirm that schema-based mapping not only enhances overall accuracy but also restructures the underlying error distribution, yielding more interpretable and trustworthy annotations in security-annotated BPMN models.

5.3.2 LLM-as-Judge Evaluation

This section evaluates the reliability of the LLM-as-Judge mechanism used to assess the quality of automatically generated SecBPMN annotations. Unlike objective metrics such as F1-score or Completeness which rely on ground-truth comparison, the LLM-as-Judge paradigm measures how accurately another language model can evaluate outputs

based on semantic alignment, structural soundness, and constraint correctness. The evaluation thus explores whether the LLM-judge’s assessments (Accuracy, Completeness, and Correctness) correlate consistently with objective performance metrics.

LLM-as-Judge Metrics

Table 5.9 reports the LLM-as-Judge metrics for both GPT-4.1-mini and Mistral-small across all complexity levels and annotation methods. For GPT-4.1-mini, RAG achieved the highest overall accuracy (0.686–0.609 across Simple to Medium) and correctness (up to 0.957), reflecting strong semantic agreement with human-like judgment. Hybrid Prompting followed closely, particularly in completeness (up to 0.824), while STS-ML guided extraction lagged substantially across all metrics. Mistral-small demonstrated similar relative trends but with consistently lower absolute values, indicating reduced evaluative consistency compared with GPT-based judgments.

BPMN Complexity	Method	GPT-4.1-mini			Mistral-small		
		<i>Acc.</i>	<i>Comp.</i>	<i>Correct.</i>	<i>Acc.</i>	<i>Comp.</i>	<i>Correct.</i>
Simple	Hybrid Prompt	0.467	0.713	0.884	0.318	0.451	0.844
	RAG	0.686	0.674	0.950	0.728	0.553	0.916
	STS-ML	0.207	0.560	0.646	0.157	0.712	0.816
Medium	Hybrid Prompt	0.517	0.824	0.917	0.505	0.681	0.902
	RAG	0.609	0.687	0.957	0.367	0.571	0.833
	STS-ML	0.146	0.186	0.486	0.129	0.281	0.643
Complex	Hybrid Prompt	0.409	0.349	0.762	0.439	0.459	0.847
	RAG	0.533	0.431	0.863	0.412	0.249	0.597
	STS-ML	0.196	0.194	0.519	0.317	0.363	0.900

Table 5.9: LLM-as-Judge evaluation of SecBPMN generation quality using three assessment dimensions: Accuracy, Completeness, and Correctness by method, model, and process complexity.

Correlation with Objective Metrics

To examine whether the LLM-judge’s evaluations align with objective ground-truth metrics, Pearson and Spearman correlations were computed between the LLM-derived scores (Accuracy, Completeness, Correctness) and standard objective measures (F1-score, Precision, Recall). The resulting heatmaps in Figures 5.8 and 5.9 visualize

the strength and direction of these correlations for GPT-4.1-mini and Mistral-small respectively.

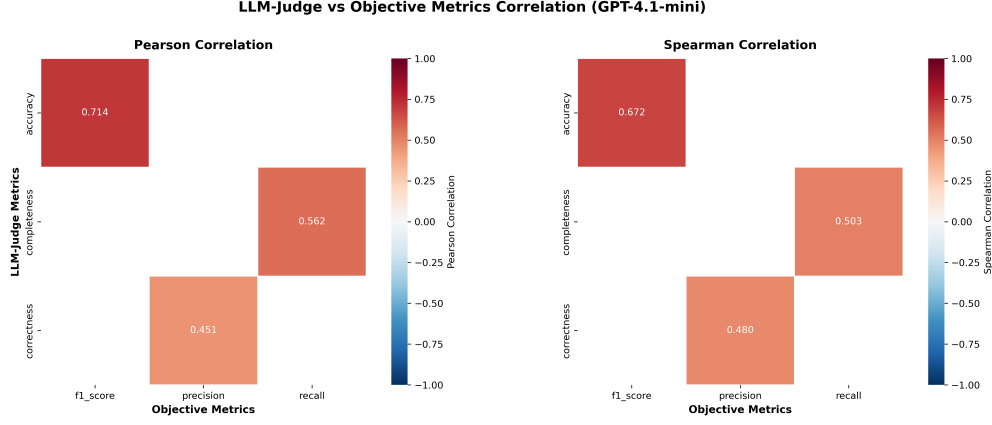


Figure 5.8: Correlation between LLM-as-Judge metrics and objective metrics for GPT-4.1-mini. Left: Pearson correlation coefficients; Right: Spearman rank correlations.

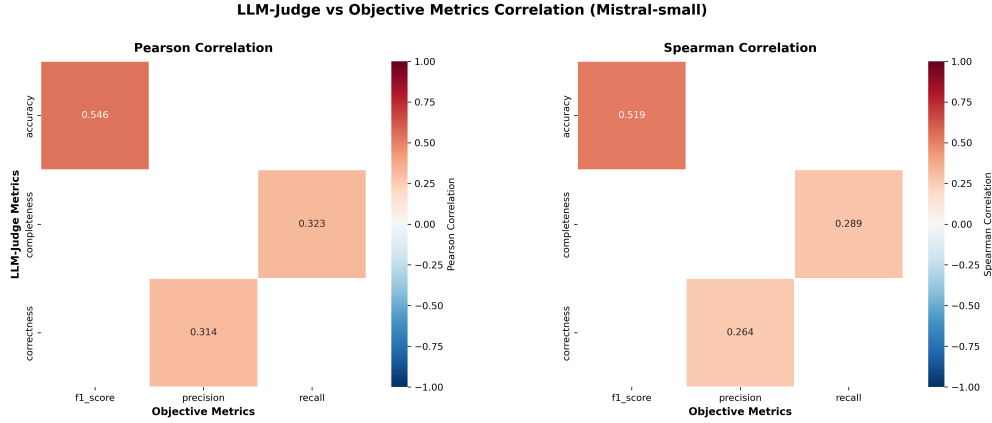


Figure 5.9: Correlation between LLM-as-Judge metrics and objective evaluation metrics for Mistral-small. Left: Pearson correlation coefficients; Right: Spearman rank correlations.

For GPT-4.1-mini (Figure 5.8), Accuracy exhibited the strongest correlation with F1-score (Pearson = 0.714; Spearman = 0.672), followed by moderate associations between Completeness and Recall (Pearson = 0.562; Spearman = 0.503). Correctness correlated most with Precision (Pearson = 0.451; Spearman = 0.480), suggesting that the LLM-judge’s perception of correctness tracks precision-oriented behavior in annotation quality. For Mistral-small (Figure 5.9), overall correlations were weaker but followed

similar patterns, with the highest Pearson correlation (0.546) again between Accuracy and F1-score, reflecting consistent though less robust alignment.

These findings indicate that LLM-as-Judge evaluations, particularly when leveraging GPT-based models, capture trends broadly consistent with objective evaluation measures. While correlations are moderate rather than perfect, they validate the LLM-judge’s utility as a scalable proxy evaluator for security annotation quality, especially when human expert review is infeasible.

5.3.3 Efficiency Benchmark: Token Cost and Latency Analysis

This section evaluates the computational efficiency of the proposed pipeline, focusing on two key operational metrics: total token consumption and processing latency. Token usage reflects the computational cost associated with model generation, while latency quantifies the end-to-end inference time, encompassing retrieval, reasoning, and response construction. These metrics are crucial for assessing scalability and real-world deployment feasibility, especially when comparing lightweight and large language models.

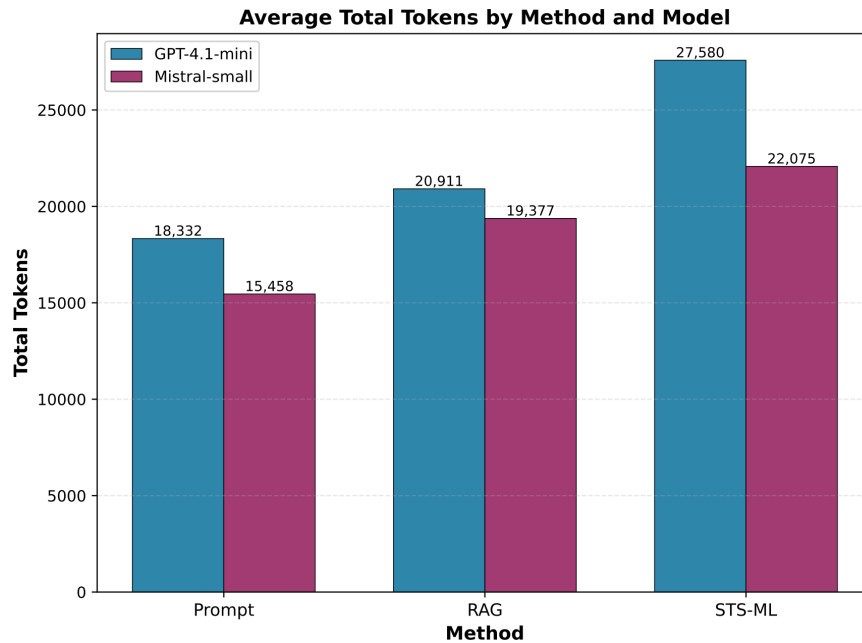


Figure 5.10: Average total token consumption by method and model.

Figure 5.10 compares token utilization across annotation strategies. GPT-4.1-mini consumed more tokens on average (18k–28k) than Mistral-small (15k–22k), consistent with its larger parameterization and higher contextual window. RAG-based inference

increased token usage compared with the Prompt baseline, reflecting retrieval overhead and multi-turn reasoning expansion. STS-ML guided extractor exhibited the highest consumption overall, driven by verbose schema translation sequences. These results suggest that more sophisticated prompting paradigms enhance annotation quality at the cost of higher computational demand.

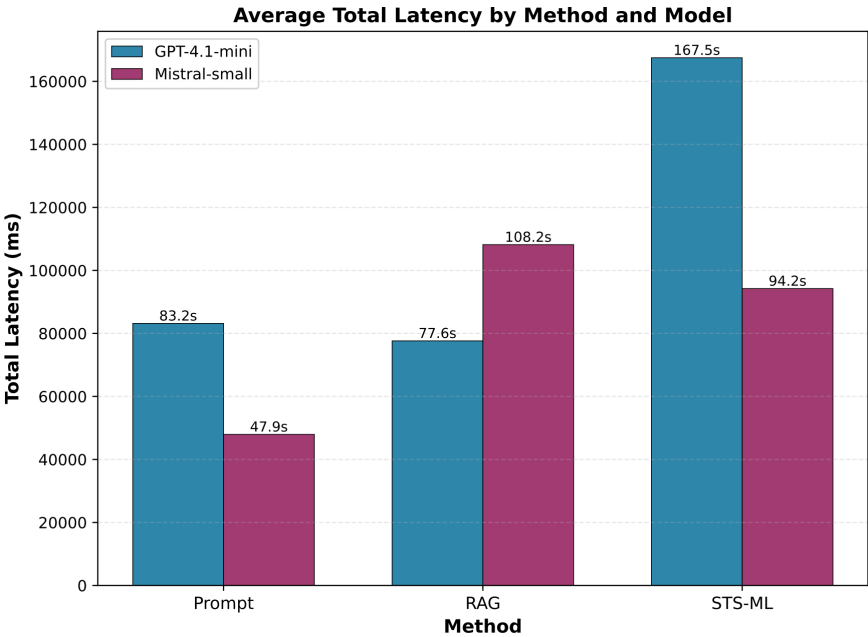


Figure 5.11: Average total latency by method and model. Latency is measured as total end-to-end response time (milliseconds) per evaluation run.

Figure 5.11 illustrates average total latency by model and method. GPT-4.1-mini incurred the longest delays, reaching up to 167.5 s under STS-ML and 83.2 s for Prompt runs. Mistral-small achieved faster average response times (47.9–108.2 s), benefiting from reduced model complexity. Among methods, RAG and STS-ML produced longer latencies due to retrieval lookups and extended decoding, respectively. Despite these differences, both models exhibited stable runtime scaling across process complexities, indicating predictable cost–performance trade-offs. Overall, the efficiency benchmark demonstrates a clear trade-off between accuracy and computational cost. GPT-4.1-mini achieves higher annotation fidelity and stronger correlation with objective metrics but at increased token and time expenditure. Conversely, Mistral-small provides faster and more lightweight inference suitable for iterative or large-scale deployments, albeit with moderate accuracy reduction. These observations highlight the need to balance reasoning depth and efficiency when operationalizing LLM-based security annotation systems in practice.

5.3.4 Expert vs. SecBPMN-GPT Comparison

To contextualize the model’s performance against human judgment, this experiment compared the annotation quality and efficiency of three domain experts with the automated SecBPMN-GPT framework. Each annotator independently labeled a randomly selected subset of business process models using the textual descriptions either in Format A or Format B as discussed in subsection 5.2.2. The comparison was based on F1-score, Precision, Recall, and average annotation time. Results are summarized in Table 5.10.

Annotator	N	F1 Score	Precision	Recall	Time (min)
Expert 1	7	0.079	0.073	0.170	47.0
Expert 2	10	0.439	0.412	0.633	71.0
Expert 3	9	0.417	0.327	0.619	131.0
<i>Human Avg</i>	<i>26</i>	<i>0.334</i>	<i>0.292</i>	<i>0.503</i>	<i>249.0</i>
SecBPMN-GPT	8	0.516	0.583	0.517	10.8

Table 5.10: Comparison between human expert annotators and the automated SecBPMN-GPT chatbot. N denotes the number of annotated models.

As shown in Table 5.10, SecBPMN-GPT outperformed the average human annotator in all objective metrics, achieving an F1-score of 0.516 compared with a human average of 0.334. The system also demonstrated higher precision (0.583 vs. 0.292), indicating greater specificity in constraint identification, and comparable recall (0.517 vs. 0.503), suggesting similar coverage of relevant annotations. In terms of efficiency, the automated framework required approximately 10.8 minutes per model, nearly **23X faster** than the combined average human annotation time (249 minutes).

The results highlight SecBPMN-GPT’s ability to achieve human-comparable accuracy while substantially reducing manual effort. Although domain experts provided more consistent recall across tasks, their performance showed wider inter-annotator variability. In contrast, the LLM-driven approach produced reproducible annotations with balanced precision–recall characteristics, reinforcing its potential as a scalable tool for preliminary or large-scale security modeling. Future work could integrate hybrid human–AI annotation pipelines to combine the efficiency of automated reasoning with expert domain validation for critical tasks.

Chapter 6

Conclusion and Future Work

This thesis presented a hybrid framework that combines rule-based reasoning with large language models (LLMs) to automate the generation of security-annotated business process models following the Security-annotated Business Process Model and Notation (SecBPMN) standard. The study addressed the central challenge of translating unstructured natural-language process descriptions into formal, machine-readable representations that embed security semantics. By coupling deterministic schema constraints with the adaptive reasoning capabilities of LLMs, the proposed system demonstrates that generative models can move beyond textual understanding toward structured, semantically valid process formalization. The end-to-end pipeline integrates process normalization, LLM-driven annotation extraction, schema-based mapping, and reconstruction into valid SecBPMN outputs, forming a coherent workflow that balances interpretability with automation. The implementation of this pipeline through an interactive prototype enabled systematic evaluation across 27 curated benchmarked SecBPMN–text pairs covering simple, medium, and complex scenarios.

The experimental results confirmed the efficacy of this hybrid approach. Retrieval-augmented generation and structured prompting consistently improved classification performance, with GPT-4.1-mini achieving the highest accuracy across most settings. Post-mapping validation further enhanced performance by increasing the average F1-score by 8–12% and improving structural quality indicators such as completeness, schema validity, and reduction of spurious annotations. These improvements illustrate the corrective function of schema-based mapping, which filters redundant or inconsistent outputs without compromising recall. The confusion matrix analysis provided a deeper view of model behavior, revealing that post-mapping substantially reduced false positives by nearly 50% for GPT—while maintaining or slightly improving true positive counts. Mistral-small exhibited a more conservative pattern, producing fewer false positives but higher false negatives, confirming that schema constraints shift the

model toward precision-oriented reasoning. Together, these findings demonstrate that structural enforcement acts as an effective regularization mechanism, stabilizing model outputs and improving semantic coherence.

The evaluation of the LLM-as-Judge approach further expanded the contribution of this work by showing that language models can serve as approximate evaluators of annotation quality. The observed correlations between LLM-based assessment metrics and objective performance measures suggest that models can internalize quality criteria and provide scalable evaluation mechanisms for process annotations. Efficiency benchmarks revealed the trade-off between accuracy and computational cost: GPT-4.1-mini achieved superior performance at the expense of higher token usage and latency, while Mistral-small provided a faster, resource-efficient alternative with moderate accuracy. The human comparison study highlighted that SecBPMN-GPT outperformed with overall higher accuracy ($F1 = 0.516$) and significantly reduced annotation time compared with domain experts, although human annotators demonstrated superior adaptability to complex contextual dependencies. These results collectively confirm the feasibility of using hybrid LLM–rule-based systems to accelerate and standardize the annotation of security requirements while preserving interpretive oversight where needed.

The findings of this thesis contribute to the growing body of research on intelligent, secure-by-design process modeling. By introducing a reproducible hybrid architecture that unites formal schema mapping with generative reasoning, this work establishes a methodological foundation for trustworthy process annotation. The developed benchmark and evaluation framework offer a replicable basis for future studies on LLM performance, schema alignment, and human–AI collaboration in business process engineering. At the same time, several limitations remain. The dataset used in this study, while carefully curated, covers a limited range of process domains, constraining generalization. Moreover, the current prototype does not yet incorporate automated post-generation validation frameworks such as SecBPMN-Q, which would further enhance semantic consistency and compliance verification.

Future work will extend the framework toward end-to-end text-to-SecBPMN generation, allowing both process structure and security annotations to be generated directly from textual descriptions. Integrating automated reasoning modules that combine LLM inference with rule-based validation will further strengthen reliability and transparency. Incorporating compliance-aware reasoning aligned with data protection standards such as the GDPR and HIPAA will improve regulatory relevance and enable context-sensitive model generation. Another promising direction lies in exploring multi-agent LLM architectures in which one model generates annotations and another validates them, fostering self-correcting, interpretable generation pipelines.

In conclusion, this research demonstrates that combining structured rule-based control with the reasoning capabilities of LLMs provides a viable pathway toward secure, interpretable, and efficient automation of business process modeling. The results show that such hybrid architectures can achieve measurable gains in accuracy, structure, and scalability while maintaining human-like judgment and interpretability. Beyond technical performance, this study contributes a conceptual framework for understanding how language models can participate in the formalization of security requirements, thereby advancing the vision of trustworthy, regulation-aware, and secure-by-design process automation.

Bibliography

- Gustav Aagesen and John Krogstie. Bpmn 2.0 for modeling business processes. In Handbook on business process management 1: introduction, methods, and information systems, pages 219–250. Springer, 2014. (Cited on pages 9 and 30.)
- Ahmed Awad. Bpmn-q: A language to query business processes. In Enterprise modelling and information systems architectures—concepts and applications, pages 115–128. Gesellschaft für Informatik e. V., 2007. (Cited on page 30.)
- Ahmed Awad and Sherif Sakr. On efficient processing of bpmn-q queries. Computers in Industry, 63(9):867–881, 2012. (Cited on page 30.)
- Patrice Béchard and Orlando Marquez Ayala. Reducing hallucination in structured outputs via retrieval-augmented generation. arXiv preprint arXiv:2404.08189, 2024. (Cited on page 27.)
- Hugo Boulanger, Nicolas Hiebel, Olivier Ferret, Karën Fort, and Aurélie Névéol. Using structured health information for controlled generation of clinical cases in french. In Proceedings of the 6th Clinical Natural Language Processing Workshop, pages 172–184, 2024. (Cited on page 34.)
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. Advances in neural information processing systems, 33:1877–1901, 2020. (Cited on pages 24 and 25.)
- Achim D Brucker, Isabelle Hang, Gero Lückemeyer, and Raj Ruparel. Securebpmn: Modeling and enforcing access control requirements in business processes. In Proceedings of the 17th ACM symposium on Access Control Models and Technologies, pages 123–126, 2012. (Cited on pages 1, 2 and 30.)
- Yulia Cherdantseva and Jeremy Hilton. A reference model of information assurance & security. In 2013 international conference on availability, reliability and security, pages 546–555. IEEE, 2013. (Cited on pages 13 and 30.)

- Michele Chinosi and Alberto Trombetta. Bpmn: An introduction to the standard. Computer Standards & Interfaces, 34(1):124–134, 2012. (Cited on pages 1, 8, 9 and 29.)
- William De Michele, Abel Armas Cervantes, and Lea Frermann. Automated business process analysis: An llm-based approach to value assessment. In International Conference on Advanced Information Systems Engineering, pages 62–69. Springer, 2025. (Cited on page 33.)
- Fabian Friedrich, Jan Mendling, and Frank Puhlmann. Process model generation from natural language text. In International conference on advanced information systems engineering, pages 482–496. Springer, 2011. (Cited on page 32.)
- Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, et al. A survey on llm-as-a-judge. arXiv preprint arXiv:2411.15594, 2024. (Cited on page 35.)
- Simon Hacks, Robert Lagerström, and Daniel Ritter. Towards automated attack simulations of bpmn-based processes. In 2021 IEEE 25th International Enterprise Distributed Object Computing Conference (EDOC), pages 182–191. IEEE, 2021. (Cited on page 31.)
- Luca Franziska Hörner. Introducing bpmngen: An llm-based conversational framework for bpmn 2.0 process model generation. In EMISA 2025, pages 17–31. Gesellschaft für Informatik eV, 2025. (Cited on page 33.)
- Ana Ivanchikj, Souhaila Serbout, and Cesare Pautasso. From text to visual bpmn process models: design and evaluation. In Proceedings of the 23rd ACM/IEEE international conference on model driven engineering languages and systems, pages 229–239, 2020. (Cited on page 32.)
- Aobo Kong, Shiwan Zhao, Hao Chen, Qicheng Li, Yong Qin, Ruiqi Sun, Xin Zhou, Enzhi Wang, and Xiaohang Dong. Better zero-shot reasoning with role-play prompting. arXiv preprint arXiv:2308.07702, 2023. (Cited on page 24.)
- Julius Köpke and Aya Safan. Efficient llm-based conversational process modeling. In International Conference on Business Process Management, pages 259–270. Springer, 2024. (Cited on pages 3, 33 and 35.)
- Julius Köpke, Giovanni Meroni, and Mattia Salnitri. Designing secure business processes for blockchains with secbpmn2bc. Future Generation Computer Systems, 141: 382–398, 2023. (Cited on pages 38, 56 and 68.)

- Humam Kourani, Alessandro Berti, Daniel Schuster, and Wil MP van der Aalst. Process modeling with large language models. In International Conference on Business Process Modeling, Development and Support, pages 229–244. Springer, 2024a. (Cited on pages 32 and 35.)
- Humam Kourani, Alessandro Berti, Daniel Schuster, and Wil MP Van der Aalst. Pro-moi: process modeling with generative ai. arXiv preprint arXiv:2403.04327, 2024b. (Cited on page 3.)
- Humam Kourani, Alessandro Berti, Daniel Schuster, and Wil MP van der Aalst. Evaluating large language models on business process modeling: Framework, benchmark, and self-improvement analysis. Software and Systems Modeling, pages 1–36, 2025. (Cited on page 33.)
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. Advances in neural information processing systems, 33:9459–9474, 2020. (Cited on pages 26 and 43.)
- Josip Tomo Licardo, Nikola Tanković, and Darko Etinger. A method for extracting bpmn models from textual descriptions using natural language processing. Procedia computer science, 239:483–490, 2024. (Cited on page 32.)
- Michael Xieyang Liu, Frederick Liu, Alexander J Fiannaca, Terry Koo, Lucas Dixon, Michael Terry, and Carrie J Cai. "we need structured output": Towards user-centered constraints on large language model output. In Extended Abstracts of the CHI Conference on Human Factors in Computing Systems, pages 1–9, 2024a. (Cited on page 24.)
- Pai Liu, Wenyang Gao, Wenjie Dong, Lin Ai, Ziwei Gong, Songfang Huang, Zongsheng Li, Ehsan Hoque, Julia Hirschberg, and Yue Zhang. A survey on open information extraction from rule-based model to large language model. arXiv preprint arXiv:2208.08690, 2022. (Cited on page 23.)
- Yu Liu, Duantengchuan Li, Kaili Wang, Zhuoran Xiong, Fobo Shi, Jian Wang, Bing Li, and Bo Hang. Are llms good at structured outputs? a benchmark for evaluating structured output capabilities in llms. Information Processing & Management, 61(5):103809, 2024b. (Cited on pages 24 and 35.)
- Xilai Ma, Jing Li, and Min Zhang. Chain of thought with explicit evidence reasoning for few-shot relation extraction. arXiv preprint arXiv:2311.05922, 2023. (Cited on page 26.)

- Curtis L Maines, David Llewellyn-Jones, Stephen Tang, and Bo Zhou. A cyber security ontology for bpmn-security extensions. In 2015 IEEE international conference on computer and information technology; ubiquitous computing and communications; dependable, autonomic and secure computing; pervasive intelligence and computing, pages 1756–1763. IEEE, 2015. (Cited on page 30.)
- G Meroni, A Dalskov, and A Norta. Improving business processes with blockchain-secbpmn2bc case studies (2024). (Cited on page 56.)
- Giovanni Meroni, Anders Dalskov, and Alex Norta. Secbpmn2bc online editor: A web-based tool for designing secure business processes on blockchains. In International Conference on Advanced Information Systems Engineering, pages 197–204. Springer, 2025. (Cited on page 31.)
- Leonard Nake. Integrating it security aspects into business process models: A taxonomy of bpmn extensions. In CIISR@ Wirtschaftsinformatik, pages 38–48, 2023. (Cited on page 30.)
- Ali Nour Eldin, Nour Assy, Olan Anesini, Benjamin Dalmas, and Walid Gaaloul. Nala2bpmn: Automating bpmn model generation with large language models. In International Conference on Cooperative Information Systems, pages 398–404. Springer, 2024. (Cited on pages 3, 33 and 35.)
- Nikolaos Nousias, George Tsakalidis, and Kostas Vergidis. Beyond gateways: Evaluating llm capabilities to decouple decision logic from business process flows. 05 2025. (Cited on page 33.)
- Elda Paja, Fabiano Dalpiaz, and Paolo Giorgini. Modelling and reasoning about security requirements in socio-technical systems. Data & Knowledge Engineering, 98: 123–143, 2015. (Cited on pages 62 and 69.)
- Pille Pullonen, Jake Tom, Raimundas Matulevičius, and Aivo Toots. Privacy-enhanced bpmn: enabling data privacy analysis in business processes models. Software and Systems Modeling, 18(6):3235–3264, 2019. (Cited on pages 1, 2 and 30.)
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. OpenAI blog, 1(8):9, 2019. (Cited on page 24.)
- Qusai Ramadan, Daniel Strüder, Mattia Salnitri, Volker Riediger, and Jan Jürjens. Detecting conflicts between data-minimization and security requirements in business process models. In European Conference on Modelling Foundations and Applications, pages 179–198. Springer, 2018. (Cited on pages 31 and 56.)

- Qusai Ramadan, Daniel Strüder, Mattia Salnitri, Jan Jürjens, Volker Riediger, and Steffen Staab. A semi-automated bpmn-based framework for detecting conflicts between security, data-minimization, and fairness requirements. Software and Systems Modeling, 19(5):1191–1227, 2020. (Cited on pages 2 and 31.)
- Alfonso Rodríguez, Eduardo Fernández-Medina, and Mario Piattini. A bpmn extension for the modeling of security requirements in business processes. IEICE transactions on information and systems, 90(4):745–752, 2007. (Cited on page 1.)
- Mattia Salnitri and Paolo Giorgini. Modeling and verification of atm security policies with secbpmn. In 2014 International Conference on High Performance Computing & Simulation (HPCS), pages 588–591. IEEE, 2014. (Cited on pages 1, 2 and 30.)
- Mattia Salnitri, Paolo Giorgini, et al. Transforming socio-technical security requirements in secbpmn security policies. In CEUR workshop proceedings, volume 1157. CEUR, 2014a. (Cited on pages 13 and 30.)
- Mattia Salnitri, Elda Paja, and Paolo Giorgini. Preserving compliance with security requirements in socio-technical systems. In Cyber Security and Privacy Forum, pages 49–61. Springer, 2014b. (Cited on page 31.)
- Mattia Salnitri, Achim D Brucker, and Paolo Giorgini. From secure business process models to secure artifact-centric specifications. In International Workshop on Business Process Modeling, Development and Support, pages 246–262. Springer, 2015a. (Cited on page 2.)
- Mattia Salnitri, Fabiano Dalpiaz, and Paolo Giorgini. Designing secure business processes with secbpmn. Software and Systems Modeling, 14(2):633–659, 2015b. doi: 10.1007/s10270-013-0349-2. URL <https://webpace.science.uu.nl/~dalpi001/papers/saln-dalp-gior-15-sosym.pdf>. (Cited on pages 1 and 56.)
- Mattia Salnitri, Fabiano Dalpiaz, and Paolo Giorgini. Designing secure business processes with secbpmn. Software & Systems Modeling, 16(3):737–757, 2017. (Cited on pages v, 2, 12, 13, 19, 20, 30 and 35.)
- Miltiadis Siavvas, Georgia Xanthopoulou, Ilias Kalouptsoglou, Dionysios Kehagias, and Dimitrios Tzovaras. Digital transformation of security standards: Requirements extraction using large language models. In 2024 11th International Conference on Dependable Systems and Their Applications (DSA), pages 430–431. IEEE, 2024. (Cited on page 34.)
- Sonit Singh. Natural language processing for information extraction. arXiv preprint arXiv:1807.02383, 2018. (Cited on page 23.)

- Fabian Stiehle, Hans Weytjens, and Ingo Weber. On llm-assisted generation of smart contracts from business processes. arXiv preprint arXiv:2507.23087, 2025. (Cited on page 33.)
- STS-Tool Development Team. Security Requirements Modeling Tool: SecBPMN2 Elements Reference Guide (rev. 1.0) for STS-Tool Version 2.1, 2015a. URL https://www.sts-tool.eu/assets/documents/manuals/SecBPMN2_Reference_rev_1.0.pdf. Accessed: 2025-09-23. (Cited on pages 38, 63 and 68.)
- STS-Tool Development Team. Security Requirements Modeling Tool: Socio-Technical Security Modeling Language (STS-ml) Manual, Version 2.1.0, 2015b. URL https://www.sts-tool.eu/assets/documents/manuals/Manual_ModelingLanguage_v.2.1.0.pdf. Accessed: 2025-09-23. (Cited on pages 38 and 68.)
- STS-Tool team. Security requirements document: Travel agency service. Technical report, STS-Tool Project, February 2016. URL http://www.sts-tool.eu/assets/documents/tutorials/tas_secbpmn/Scenario_TAS_Report-TravelAgencyService.pdf. Generated by STS-Tool. (Cited on page 56.)
- Carlos Toxtli and Wangfan Li. Automating automation: Using llms to generate bpmn workflows for robotic process automation. In World Congress in Computer Science, Computer Engineering & Applied Computing, pages 221–229. Springer, 2024. (Cited on page 33.)
- William Van Woensel and Soroor Motie. Nlp4pbm: a systematic review on process extraction using natural language processing with rule-based, machine and deep learning methods. Enterprise Information Systems, 18(11):2417404, 2024. (Cited on page 34.)
- Maxim Vidgof, Stefan Bachhofner, and Jan Mendling. Large language models for business process management: Opportunities and challenges. In International conference on business process management, pages 107–123. Springer, 2023. (Cited on page 32.)
- Marvin Voelter, Raheleh Hadian, Timotheus Kampik, Marius Breitmayer, and Manfred Reichert. Leveraging generative ai for extracting process models from multimodal documents. arXiv preprint arXiv:2406.04959, 2024. (Cited on pages 33 and 35.)
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. Advances in neural information processing systems, 35:24824–24837, 2022. (Cited on page 25.)

- Seline Wenger, Maja Spahic-Bogdanovic, and Andreas Martin. Large language models for democratizing business process modeling: Bpmn model generation and style guide adherence. In Southern African Conference for Artificial Intelligence Research, pages 372–389. Springer, 2024. (Cited on page 33.)
- Stephen A White. Introduction to bpmn. Ibm Cooperation, 2(0):0, 2004. (Cited on pages v, 1, 9, 11 and 29.)
- Isabella Catharina Wiest, Fabian Wolf, Marie-Elisabeth Leßmann, Marko van Treeck, Dyke Ferber, Jiefu Zhu, Heiko Boehme, Keno K Bressemer, Hannes Ulrich, Matthias P Ebert, et al. Llm-aix: An open source pipeline for information extraction from unstructured medical text based on privacy preserving large language models. medRxiv, 2024. (Cited on page 34.)
- Derong Xu, Wei Chen, Wenjun Peng, Chao Zhang, Tong Xu, Xiangyu Zhao, Xian Wu, Yefeng Zheng, Yang Wang, and Enhong Chen. Large language models for generative information extraction: A survey. Frontiers of Computer Science, 18(6):186357, 2024. (Cited on page 23.)
- Jialin Yang, Dongfu Jiang, Lipeng He, Sherman Siu, Yuxuan Zhang, Disen Liao, Zhuofeng Li, Huaye Zeng, Yiming Jia, Haozhe Wang, et al. Structeval: Benchmarking llms’ capabilities to generate structural outputs. arXiv preprint arXiv:2505.20139, 2025. (Cited on page 24.)
- Hanwen Zheng, Sijia Wang, and Lifu Huang. A survey of document-level information extraction. arXiv preprint arXiv:2309.13249, 2023. (Cited on page 23.)