Automated Care Pathway Modeling using Agentic and Knowledge-Aware LLMs

by

Alireza Houshidari

Thesis submitted to the
Faculty of Engineering
In partial fulfillment of the requirements
for the M.Sc. degree in
Digital Transformation and Innovation

School of Engineering Design and Teaching Innovation
Faculty of Engineering
University of Ottawa

Abstract

Clinical pathways (CPWs) translate evidence-based guidance into stepwise care but are often disseminated as free text, obscuring control-flow semantics needed for clarity and computability. Formalizing CPWs as process models—e.g., in the Business Process Model and Notation (BPMN)—improves comprehensibility and enables downstream automation.

This thesis designs, implements, and evaluates **LLM4CPW**, a pipeline for automatic guideline-to-BPMN modeling using large language models (LLMs). We compare two contemporary frameworks—*MAO* (agentic, multi-role orchestration) and *ProMoAI* (single-agent with self-refinement loop)—under controlled execution with standardized evaluation. Automated metrics combine node-level and structural similarity (after Dijkman et al.) with graph-edit distance; a clinician provides fidelity ratings with qualitative annotations. We then investigate *knowledge-aware modeling* via a categorization of recurrent errors and curated clinical statements, testing prompt-only injections versus a dedicated *Knowledge Advisor* agent placed at different stages of the workflow.

Across four Ontario stroke Quality-Based Procedures (QBPs; $n = 15$ runs per framework), MAO attains higher node similarity (≈ 0.782 vs. ≈ 0.696) and structural similarity (≈ 0.630 vs. ≈ 0.585) than ProMoAI with large effects and $p < 0.001$, and exhibits markedly lower run-to-run variability; expert ratings also favor MAO. Knowledge-aware variants of MAO yield measurable gains: introducing a Knowledge Advisor after semantic review phase improves node similarity by > 4 points and reduces Graph Edit Distance by ~ 13 (to ~ 97), with statistically comparable outcomes when placed before review; expert deltas likewise favor the advisor-based designs. Refining statement wording improves medians and interpretability without shifting means.

Contributions. This thesis contributes (i) an auditable LLM4CPW pipeline and evaluation protocol; (ii) empirical evidence that agentic orchestration improves fidelity and stability; and (iii) a principled, deployable strategy for knowledge-enhanced modeling via a specialized advisory phase. Collectively, the findings demonstrate the feasibility of reliable, automatically extracted BPMN models from concise clinical guidelines and chart a path toward broader, clinically grounded automation.

Acknowledgments

I would like to express my deepest gratitude to my supervisors, **Prof. William Van Woensel** and **Prof. Daniel Amyot**, for their invaluable guidance, encouragement, and continuous support throughout the course of this thesis. Their expertise, insightful feedback, and availability as mentors were instrumental in shaping both the direction and the quality of this work.

I would also like to extend my sincere thanks to my committee members, **Prof. Gregory Richards**, Vice-Dean of Graduate Professional Programs, Telfer School of Management, University of Ottawa, and **Prof. Mana Azarm**, Assistant Professor of Analytics and Information Systems, University of San Francisco, for their time, thoughtful feedback, and constructive suggestions that helped strengthen this thesis.

I gratefully acknowledge the financial support of the Discovery Grant program of the *Natural Sciences and Engineering Research Council (NSERC)* of Canada, which made this research possible.

My deepest thanks go to my partner, parents and sister, whose unwavering love, patience, and belief in me have sustained and inspired me throughout this journey.

Finally, I would like to extend my heartfelt thanks to my extended family, friends, and all those who supported me along the way.

Table of Contents

List of Tables

List of Figures

List of Listings

List of Acronyms

ABC	Airway, Breathing, and Circulation
AAI	Agentic AI
BPMN	Business Process Management and Notation
CPG	Clinical Practice Guideline
CPW	Clinical Pathway
CRF	Conditional Random Field
CV	Coefficient of Variation
DMN	Decision Model and Notation
FHIR	Fast Healthcare Interoperability Resources
GED	Graph Edit Distance
HL7	Health Level 7
IG	Input Guideline
KG	Knowledge Graph
LLM	Large Language Model
LLM4CPW	LLM for Clinical Pathways
MAO	Multi Agent Orchestration

ML	Machine Learning
NLP	Natural Language Processing
PM	Process Mining
POWL	Partially Ordered Workflow Language
ProMoAI	Process Modeling with Generative AI
QBP	Quality-Based Procedure
UI	User Interface

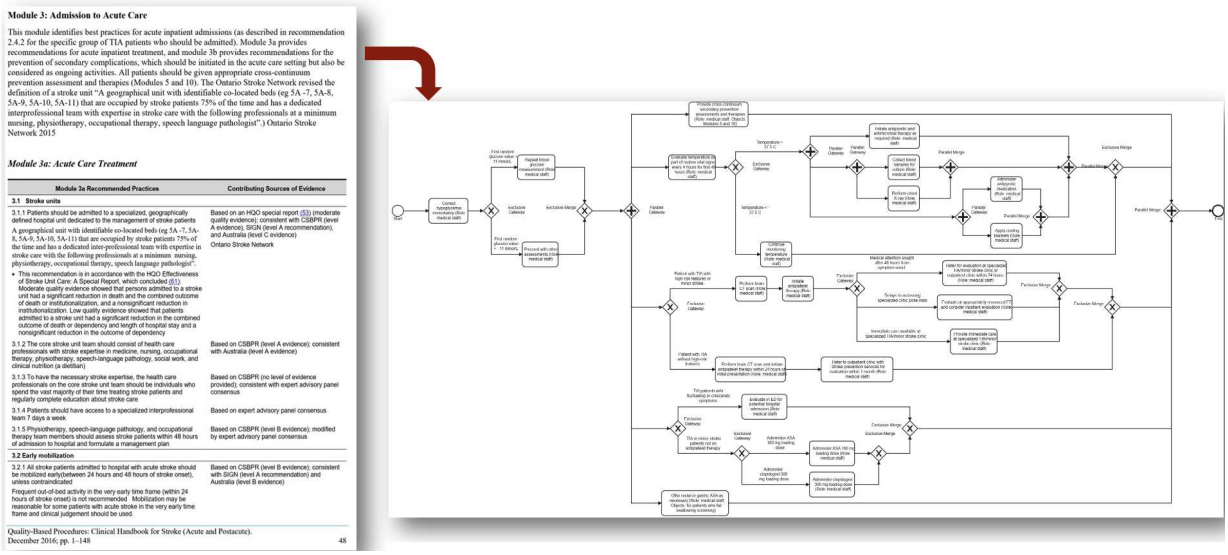
Chapter 1

Introduction

This chapter sets the stage by outlining the problem context and motivation, followed by a clear statement of the research problem, the methodology employed, and the contributions made by this work.

1.1 Motivation and Problem Context

Clinical Pathways (CPWs) operationalize evidence-based guidelines, such as Clinical Practice Guidelines (CPGs), into step-by-step protocols that standardize care and reduce unwarranted practice variation. They are known to cut in-hospital complications, shorten length of stay, lower costs, and improve documentation [Rotter et al., 2010, Grathwol et al., 2025]. However, free-text guidelines, even in the form of CPWs, typically lack clear control-flow semantics, complicating their interpretation, and reducing clarity, consistency, and machine readability [Grathwol et al., 2025]. Translating CPWs into formal, graphical process models such as the BPMN [OMG, 2014] explicates their control-flow semantics, which (a) combined with their visual nature, makes the clinical pathway less ambiguous and easier to understand; and (b) makes them fully machine-readable, enabling their integration into Clinical Decision Support Systems, as well as automated compliance checking with real-world care trajectories [Rodríguez-Loya et al., 2014, Kassim et al., 2022].



Module 3: Admission to Acute Care

This module identifies best practices for acute inpatient admissions (as described in recommendation 2.4.2 for the specific group of TIA patients who should be admitted). Module 3a provides recommendations for acute inpatient treatment, and module 3b provides recommendations for the prevention of secondary complications, which should be initiated in the acute care setting but also be considered as ongoing activities. All patients should be given appropriate cross-continuum prevention assessment and therapies (Modules 5 and 10). The Ontario Stroke Network revised the definition of a stroke unit "A geographical unit with identifiable co-located beds (eg 5A -7, 5A-8, 5A-9, 5A-10, 5A-11) that are occupied by stroke patients 75% of the time and has a dedicated interprofessional team with expertise in stroke care with the following professionals at a minimum nursing, physiotherapy, occupational therapy, speech language pathologist".) Ontario Stroke Network 2015

Module 3a: Acute Care Treatment

Module 3a Recommended Practices	Contributing Sources of Evidence
3.1 Stroke units	
3.1.1 Patients should be admitted to a specialized, geographically defined hospital unit dedicated to the management of stroke patients A geographical unit with identifiable co-located beds (eg 5A -7, 5A-8, 5A-9, 5A-10, 5A-11) that are occupied by stroke patients 75% of the time and has a dedicated inter-professional team with expertise in stroke care with the following professionals at a minimum nursing, physiotherapy, occupational therapy, speech language pathologist". • This recommendation is in accordance with the HQO Effectiveness of Stroke Unit Care: A Special Report, which concluded (53): Moderate quality evidence showed that persons admitted to a stroke unit had a significant reduction in death and the combined outcome of death or institutionalization, and a nonsignificant reduction in institutionalization. Low quality evidence showed that patients admitted to a stroke unit had a significant reduction in the combined outcome of death or dependency and length of hospital stay and a nonsignificant reduction in the outcome of dependency	Based on an HQO special report (53) (moderate quality evidence); consistent with CSBPR (level A evidence), SIGN (level A recommendation), and Australia (level C evidence) Ontario Stroke Network
3.1.2 The core stroke unit team should consist of health care professionals with stroke expertise in medicine, nursing, occupational therapy, physiotherapy, speech-language pathology, social work, and clinical nutrition (a dietitian)	Based on CSBPR (level A evidence); consistent with Australia (level A evidence)
3.1.3 To have the necessary stroke expertise, the health care professionals on the core stroke unit team should be individuals who spend the vast majority of their time treating stroke patients and regularly complete education about stroke care	Based on CSBPR (no level of evidence provided); consistent with expert advisory panel consensus
3.1.4 Patients should have access to a specialized interprofessional team 7 days a week	Based on expert advisory panel consensus
3.1.5 Physiotherapy, speech-language pathology, and occupational therapy team members should assess stroke patients within 48 hours of admission to hospital and formulate a management plan	Based on CSBPR (level B evidence); modified by expert advisory panel consensus
3.2 Early mobilization	
3.2.1 All stroke patients admitted to hospital with acute stroke should be mobilized early/between 24 hours and 48 hours of stroke onset), unless contraindicated Frequent out-of-bed activity in the very early time frame (within 24 hours of stroke onset) is not recommended Mobilization may be reasonable for some patients with acute stroke in the very early time frame and clinical judgement should be used.	Based on CSBPR (level B evidence); consistent with SIGN (level A recommendation) and Australia (level B evidence)

Quality-Based Procedures: Clinical Handbook for Stroke (Acute and Postacute). December 2016; pp. 1–148 48

Figure 1.1: Example of converting a stroke QBP excerpt into a BPMN process model. The figure underscores how narrative recommendations are rendered with explicit control-flow semantics (gateways, sequence flows, and parallelism). (The textual details are not meant to be legible.)

To illustrate such a translation, Figure 1.1 shows an excerpt from an Ontario stroke Quality-Based Procedure (QBP) (a type of CPW), and the corresponding BPMN model. The example highlights how implicit ordering, branching, and concurrency in the prose become explicit gateways and sequence flows.

However, there are some pain points in manually creating and maintaining CPW process models. Just in 2023, over 26,000 cancer-related papers were published on PubMed. As a result, CPGs are continually updated to incorporate the latest evidence. In turn, CPW processes should be continuously revised to align with current CPGs. However, the manual extraction of CPWs in form of formal graphical process models (BPMN) from free-text narrative guidelines is complex and susceptible to interpretation bias; these difficulties are compounded by the need to keep pace with the volume of new research [Grathwol et al., 2025].

Large Language Models (LLMs), pre-trained on extensive text corpora, have demonstrated remarkable potential across various Natural Language Processing (NLP) tasks [Yang et al., 2024a]; they are highly effective at parsing natural language, extracting structured information from text, and including general knowledge when answering prompts. Early research suggests that LLMs perform "comparably to or better than existing solu-

tions" such as other, more "traditional" NLP methods [Grohs et al., 2023]. Hence, they have the potential to automate the process of CPW extraction form CPGs. Earlier research highlighted that LLMs struggled with process extraction from clinical guidelines, due to the ambiguity and domain-specific nature of guidelines, and can thus miss the nuanced information embedded therein [Grathwol et al., 2025]. Nevertheless, recent architectural advances, including multi-agent frameworks (MAO [Lin et al., 2025]) and improved error handling (ProMoAI [Kourani et al., 2024]), can be used to address these issues. Moreover, previous studies have highlighted that enhancing LLMs with structured knowledge, such as Knowledge Graphs (KGs), significantly boosts their factual accuracy and reasoning capabilities [Sequeda et al., 2024, Su et al., 2025].

Need for systematic evaluation. The rapid evolution of agentic LLM frameworks and augmentation techniques (e.g., knowledge integration) makes accuracy claims difficult to compare across studies. A task-specific, standardized evaluation regime—grounded in shared datasets, consistent protocols, and transparent metrics—is therefore essential to ensure commensurable and reproducible results.

1.2 Problem Statement

This thesis identifies important problems in automatically converting textual CPGs to CPW process models in BPMN:

- Manual methods are slow and existing automatic approaches that use "traditional" NLP methods often fail to capture domain-specific nuances and maintain factual accuracy due to complex clinical terminologies and relationships in CPGs [Grathwol et al., 2025].
- Fully automated extraction of CPWs from CPGs has been deemed infeasible just recently [Grathwol et al., 2025]
- To the best of our knowledge current approaches do not integrate domain-specific knowledge to guide LLMs.

- To the best of our knowledge there is no existing LLM based-framework or pipeline for CPW extraction *and* evaluation from CPGs

1.3 Research Questions

- **RQ1:** To what extent can current LLM frameworks be used to automatically extract CPW models from natural text guidelines (CPGs)?
- **RQ2:** To what extent can injecting structured clinical knowledge into an LLM-based pipeline improve the accuracy and consistency of extracted CPW models?

1.4 Methodology

This thesis adopts an *artifact-centric, stage-gated* methodology whose primary objective is to design, implement, and empirically evaluate **LLM4CPW**, a modular pipeline for extracting BPMN models from textual clinical guidelines. The method couples comparisons of state-of-the-art frameworks with dual evaluation (automated metrics and expert review), followed by targeted knowledge-augmentation experiments. Decision gates make the inquiry auditable and reproducible and map directly to the implementation and results reported in Chapters 3–6. Figure 1.2 provides a high-level overview of the pipeline.

Logic of inquiry. The inquiry proceeds in two movements: a controlled baseline comparison addressing RQ1, and a series of knowledge-enhanced versions that treat "domain context" as the experimental factor addressing RQ2.

Artifact under study: **LLM4CPW** integrates (i) LLM framework execution (§3.2), (ii) a canonical standardization module to BPMN 2.0 with auto-layout (§3.3), and (iii) a standardized evaluation stack (§3.4, §3.5). Evaluating an integrated pipeline (rather than isolated prompts or scripts) lets us hold constant the surrounding infrastructure (conversion, layout, logging), compare frameworks under uniform conditions (Chapter 5), and

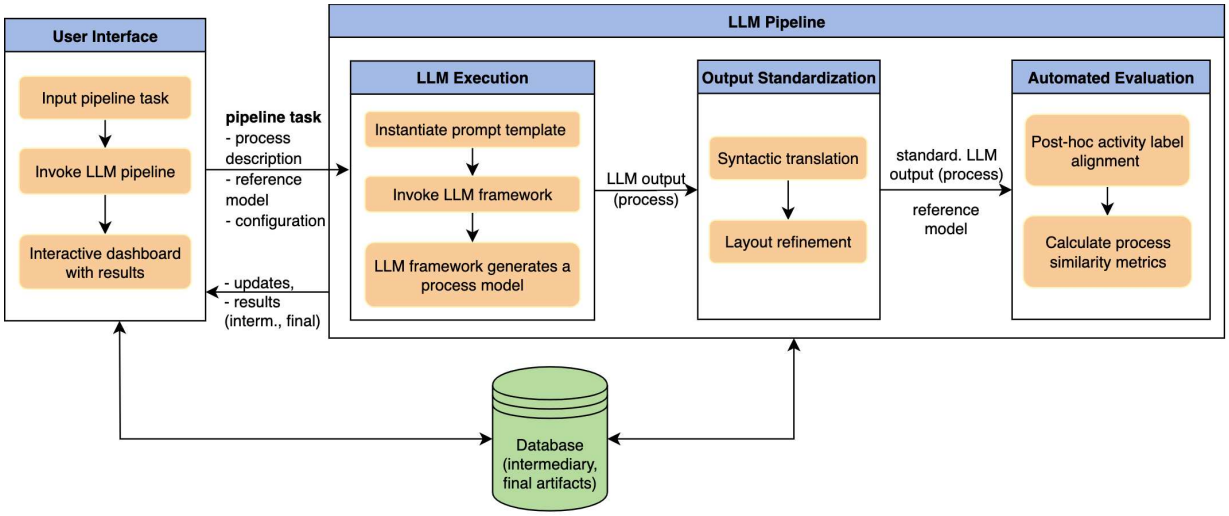


Figure 1.2: High-level overview of the thesis' LLM4CPW pipeline.

iterate cleanly on knowledge-integration strategies (Chapter 4). More information on LLM4CPW is available in Chapter 3.

Comparative assessment and justification. MAO and ProMoAI are selected as baselines because they target text-to-BPMN extraction and compile via explicit intermediary languages (BPMN-Text/POWL), yielding outputs that are commensurable after standardization (§2.1.4, §3.3). We quantify structural fidelity using node/structure similarity and Graph Edit Distance (GED) [Dijkman et al., 2011], metrics widely used in BPMN process comparison and available in robust open implementations that capture gateway types and label matches (§3.4). Different automated evaluation metrics are assessed in Section 3.4. Because structural metrics can miss clinically salient semantics (e.g., conditional logic, eligibility) and their focus on structural isomorphism also penalizes logically equivalent refactorings, we complement them with human expert review (Likert fidelity ratings with qualitative annotations) to cross-validate findings (Chapter 6). For inferential comparisons, we report Welch's t (robust to unequal variances[1]) and Cohen's d (effect size[2]), alongside run-to-run variability via the Coefficient of Variation (CV) to separate systematic gains from sampling noise.

[1] https://en.wikipedia.org/wiki/Welch%27s_t-test
[2] https://en.wikipedia.org/wiki/Effect_size#Cohen's_d

Stage-gated procedure (inputs, activities, outputs, and gate decisions).

P0) Scope and protocol registration (setup). We fix the research questions, target domain (Ontario stroke QBPs [Health Quality Ontario & Ministry of Health and Long-Term Care, 2023]), comparison pairs (MAO vs. ProMoAI; knowledge-enhanced vs. baseline), and the evaluation plan (metrics, expert rubric, statistical tests). The output is a concise protocol and reusable run templates. This front-loads choices to reduce researcher degrees of freedom and supports reproducibility.

P1) Data ingestion and reference model preparation. We curate four randomly selected QBPs from the Stroke QBPs of Ontario and confirm expert-validated BPMN references for the corresponding care pathways. After a final sanity check, we produce task files, a dataset manifest, and reference BPMN models. *Gate G1:* if the references are incomplete or out of scope, we revise before continuing. This anchors all subsequent comparisons to consistent targets.

P2) Baseline pipeline execution (MAO vs. ProMoAI). Using GPT-4.1 under similar hyperparameters, detailed in Chapter 5, each framework is run on each QBP for n trials (default n=15). Outputs are standardized to canonical BPMN 2.0 (with auto-layout) and, for automated evaluation metrics only, labels are aligned post-hoc to enable fair string-sensitive comparisons (§3.4 elaborates on this concepts, their necessity, and the procedures by which they were applied); original labels are preserved for expert review. We retain all BPMN, execution logs (including agent conversations), and timings. *Gate G2:* models must parse and satisfy minimal completeness; frameworks apply bounded retries and the pipeline automatically excludes failed models. This isolates framework effects while normalizing downstream evaluation.

P3) Dual evaluation of baselines. LLM-generated BPMN outputs are compared to reference models using node/structure similarity and GED; we summarize central tendency and run-to-run variability and perform Welch's t with Cohen's d. Independently, a clinician reviewer rates overall fidelity on a Likert scale and annotates sequencing, conditional, and completeness issues. *Gate G3:* if errors are systematic

across runs/QBPs, we proceed to knowledge integration; otherwise we document and conclude. Combining automated metrics with clinician review mitigates single-method bias and exposes specific, remediable failure patterns.

P4) Issue categorization and context construction. We code expert comments and model differences compared to the reference model into a common issue categorization (temporal relations, task granularity, task options). From this, we construct statement libraries at three abstraction levels (Concise, Expanded, Explicit) and design integration strategies (prompt-only vs. dedicated agent; integration timing). The output is a set of MAO versions with their prompts and configurations. This converts qualitative diagnosis into testable interventions. More details and justifications are available in Chapter 4.

P5) Knowledge-enhanced execution (MAO versions). Starting from the baseline MAO (the original version), we re-run the pipeline for each version across all QBPs (by default with n=15; ablation studies with n=5), producing standardized BPMN and logs as before. Holding the infrastructure constant isolates the marginal contribution of domain knowledge.

P6) Dual evaluation of knowledge-enhanced versions. We compare each version to baseline MAO on node/structure similarity and GED, reporting Welch's t and Cohen's d and summarizing dispersion. Expert review is expressed as a per-QBP net Δ (positives minus negatives, counting "slight" as ± 0.5) relative to the baseline. *Gate G4:* we identify a preferred integration strategy if improvements are consistent and free of systematic regressions; otherwise we refine statements or placement and iterate (P4–P6).

P7) Synthesis, validity checks, and artefact release. We complement the systematic evaluation with expert evidence, discuss threats to validity (construct, internal, external) and their mitigations, and package all artefacts (prompts, configs, BPMN, SQLite/CSV outputs, dashboards) to facilitate independent replication. The results and limitations are consolidated in Chapters 6 and 7.

Analysis plan (and why these statistics). For each comparison we report mean/-median node and structure similarity and GED, run-to-run stability via the coefficient of variation, Welch's t for significance under unequal variances, Cohen's d for effect magnitude, and expert outcomes (Likert fidelity and net Δ). Post-hoc label alignment is restricted to automated metrics to avoid conflating semantic judgments during expert review. This mix provides both *statistical* and *practical* significance and aligns our assessment with related work while making effect sizes comparable across settings.

Auditability. All runs execute through the same User Interface (UI) (§3.5) and versioned pipeline; failures after bounded retries are logged and excluded by rule; and every artefact is persisted and exportable. This ensures that conclusions reflect the behavior of *frameworks and interventions*, not incidental implementation variance.

Positioning with respect to related work. Methodologically, this thesis extends extraction of formal process models from free-text along four axes. First, it conducts a comparison of MAO and ProMoAI within a single standardized artifact—**LLM4CPW**—so differences reflect frameworks rather than tooling (Chapters 3–6). Second, it presents a controlled examination of *knowledge-enhanced* modeling through a library of domain knowledge statements derived from issue categorization and explicit integration strategies (Chapter 4). Third, it adopts a *dual evaluation* regime that complements structural metrics (node/structure similarity and GED) with clinician judgment and quantifies run-to-run variability, addressing gaps in purely structural or single-shot evaluations. Fourth, it emphasizes *reproducibility*: standardized BPMN 2.0 conversion, pre-specified protocols and hyperparameters, expert-validated references in a real clinical domain (Ontario stroke QBPs), and the release of prompts/configurations and BPMN artefacts. Together, these choices permit controlled tests of prior claims on structural fidelity, robustness, and the contribution of domain context, and they structure the comparative discussion in Chapters 2 and 7.

1.5 Thesis Contributions

This thesis contributes an automated pipeline, called LLM4CPW, which integrates:

1. Existing LLM frameworks tailored to extract process models from text. Currently, these include MAO and ProMoAI, running over GPT-4.1 [OpenAI, 2025]. Other LLM frameworks and foundational models can be plugged into our pipeline as they become available.

2. A new unified conversion module that translates heterogeneous LLM outputs (processes) into canonical BPMN models with auto-layout.

3. A new automated evaluation of generated process models against reference process models using standard node- and structural-similarity metrics [Dijkman et al., 2011], for repeatable and scalable performance measurement.

4. A new GPT-based label-alignment component that harmonizes LLM-generated activity labels with the reference terminology. This ensures that semantically correct but differently phrased labels can still be matched.

5. A new web-based tool for launching experiments and inspecting results.

Additional contributions include:

1. A principled, iterative design and empirical evaluation of structured clinical-knowledge integrations—via curated statement libraries, targeted prompt integration, and a dedicated Knowledge Advisor—showing that knowledge-enhanced prompting improves the accuracy and consistency of extracted CPW models (RQ2).

2. A controlled benchmark of state-of-the-art LLM frameworks for automatic CPW extraction from CPGs, demonstrating the feasibility of end-to-end automated process extraction with current technology and identifying the agentic MAO framework as the best-performing approach (RQ1).

3. The public release of the pipeline, prompts, KGs, datasets, and dashboards to ensure reproducibility and community contribution (RQ1/RQ2). This is in line with the spirit of open science.

1.6 Publications

This thesis has led to two publications so far:

1. **Houshidari, A.**, Van Woensel, W., Amyot, D., Bouattane, E.M. (2025) An LLM Pipeline for Automatic Extraction and Evaluation of Care Pathways from Clinical Guidelines. *Workshop on Natural Language Processing for Business Process Management (NLP4BPM)*, BPM'25, Sevilla, Spain, September 2025. LNBIP, Springer.

    - The content of this paper is covered across: Chapter 2 (problem framing and related work), Chapter 3 (pipeline architecture and conversion layer), Chapter 5 (datasets and protocol), Chapter 6 (quantitative and qualitative evaluation), and Chapter 7 (implications and threats to validity).

2. Van Woensel, W., **Houshidari, A.**, Amyot, D. (2025) Towards Automated Compliance Checking for Care Trajectories: Process Extraction Using Large Language Models. *2nd International Workshop on Process Mining Applications for Healthcare (PM4H25)*, AIME'25, Pavia, Italy, June 2025. LNCS, Springer.

    - The paper's material is integrated across: (i) Chapter 2 (compliance-checking landscape: model- vs. log-based comparison); (ii) Chapters 3–5 (LLM execution protocol: systems, inputs, conversion, alignment, and assessments); (iii) Chapter 6 (head-to-head MAO vs. ProMoAI findings, variability, correlations, and effect sizes); and (iv) Chapter 7 (limits from guideline incompleteness and the case for knowledge injection).

1.7 Thesis Outline

The remainder of this thesis is organized as follows.

- Chapter 2 surveys related work on clinical pathway modeling, BPMN extraction, and knowledge-enhanced LLMs.
- Chapter 3 details the system architecture, LLM frameworks, and conversion module design.
- Chapter 4 describes the identification and integration of structured clinical knowledge.
- Chapter 5 presents the experimental setup, data sources (four Ontario stroke QBPs [Health Quality Ontario & Ministry of Health and Long-Term Care, 2023]), and reference BPMN creation.
- Chapter 6 reports quantitative and qualitative evaluation results.
- Chapter 7 synthesizes results and discusses related work, threats to validity, and practical implications.
- Finally, Chapter 8 concludes and outlines future research directions.

Chapter 2

Background and Related Work

This chapter positions the developed approach in the state of the art.

Section 2.1 describes the healthcare context, automatic process extraction, and reviews techniques spanning rules and classical machine learning to contemporary LLM-based frameworks with orchestration (e.g., error-handling loops and multi-agent designs). Section 2.2 then discusses some basic concepts, differentiations and challenges related to checking processes and their models for compliance and validation. Section 2.3 finally examines mechanisms for enriching LLMs with external knowledge—with an emphasis on knowledge graph integration—and analyzes their effects on factuality, robustness, and hallucination mitigation.

2.1 Automatic Process Extraction

2.1.1 Healthcare Context

Clinical Practice Guidelines (CPGs) articulate evidence-based recommendations that establish standards of care [Institute of Medicine (US) Committee on Standards for Developing Trustworthy Clinical Practice Guidelines, 2011], while Clinical Pathways (CPWs) operationalize those recommendations as coordinated, multidisciplinary care plans [Rotter

et al., 2019, European Pathway Association, 2025]. Care trajectories, in turn, are the realized patient journeys observed in practice [Vanasse et al., 2021]. In Ontario (Canada), Quality-Based Procedures (QBPs) are CPW standardized by the provincial government.

In this thesis, we focus on enabling *compliance checking*, i.e., comparing CPWs to trajectories to identify deviations and opportunities for refinement. This is distinct from performance checking or comparative process mining across sites. Compliance checking has been shown to surface both deviations from evidence-based practice and opportunities to improve guidelines [Munoz-Gama et al., 2022, Chiudinelli et al., 2020, Gatta et al., 2019, Savino et al., 2024].

Formalizing the free-text CPGs and CPWs into an executable process representation (e.g., a BPMN model) makes ordering, branching, and concurrency explicit, and improves clarity and machine-readability [Kassim et al., 2022]. Moreover, formalizing text as BPMN provides a basis for downstream analyses and integration with clinical decision-support systems [Rodríguez-Loya et al., 2014].

From a healthcare interoperability perspective, pathway process models benefit from explicit mappings between guideline specifications and BPMN. A bidirectional transformation between Health Level 7 (HL7) Fast Healthcare Interoperability Resources (FHIR) PlanDefinition and BPMN—via an intermediate graph with rule-based mappings—enables visualization and potential execution of guideline logic in BPM tooling, while exposing gaps (e.g., decision logic via the Decision Model and Notation (DMN) standard, roles, and loops) that inform standards integration in clinical settings [Helm et al., 2022]. Therefore, automated BPMN extraction from CPGs or CPWs not only enables guideline logic to be visualized, but also supports seamless bidirectional transformation with FHIR PlanDefinition—allowing BPMN models to be converted back into computable FHIR artifacts.

Beyond single pathways, compliance checking must scale: institutions manage many conditions and regional variants. Recent estimates identified on the order of *tens of thousands* of guidelines within a single country (e.g., ∼90,000 in Denmark), underscoring the need for automated extraction and comparison [Grathwol et al., 2025].

Within an end-to-end compliance-checking pipeline, automatic process extraction of

CPWs (specified as BPMN models) from CPGs/QBPs is the second stage:

a) Mining care trajectories from healthcare system data via Process Mining (PM): PM has been successfully applied to derive trajectory models in healthcare [Mannhardt and Blinde, 2017, Chiudinelli et al., 2020, Dagliati et al., 2017, Partington et al., 2015, Suriadi et al., 2014], though challenges persist (e.g., high variability and infrequent yet purposeful behaviors) [Munoz-Gama et al., 2022].

b) Extracting pathway models from clinical guidelines: Methodologically, process extraction approaches have evolved from rules and conventional Machine Learning (ML) toward LLM-driven frameworks [Van Woensel and Motie, 2024]. While early LLM work on clinical guideline extraction judged full automation unrealistic [Grathwol et al., 2025], recent architectures introduce error-handling loops [Kourani et al., 2024] and multi-agent orchestration [Lin et al., 2025], making automation increasingly plausible and motivating the deeper treatment of these methods in the sections that follow.

c) Comparing trajectories and pathways using expert-facing visualizations [van der Aalst et al., 2013, Pini et al., 2015, Partington et al., 2015, Suriadi et al., 2014] and metrics such as holistic Graph Edit Distance [Dijkman et al., 2011], similar to how we compared the process models in this research in Section 3.4.

Running example. To illustrate the style and granularity of the textual inputs we target, the following QBP example reproduces a verbatim excerpt from a QBP document [Health Quality Ontario & Ministry of Health and Long-Term Care, 2023, chap. "Recommended Practices for TIA or Minor (Nondisabling) Stroke", Module 3] such as the one on the left of Figure 1.1; we will refer back to this excerpt when discussing extraction and evaluation.

QBP Example (verbatim excerpt of LLM pipeline input)

Module 3: Admission to Acute Care

This module identifies best practices for acute inpatient admissions (as described in recommendation 2.4.2 for the specific group of TIA patients who should be admitted). Module 3a provides recommendations for acute inpatient treatment, and module 3b provides recommendations for the prevention of secondary complications, which should be initiated in the acute care setting but also be considered as ongoing activities. All patients should be given appropriate cross-continuum prevention assessment and therapies (Modules 5 and 10). The Ontario Stroke Network revised the definition of a stroke unit "A geographical unit with identifiable co-located beds (eg 5A -7, 5A-8, 5A-9, 5A-10, 5A-11) that are occupied by stroke patients 75% of the time and has a dedicated interprofessional team with expertise in stroke care with the following professionals at a minimum nursing, physiotherapy, occupational therapy, speech language pathologist".)
Ontario Stroke Network 2015

Module 3a: Acute Care Treatment
Module 3a Recommended Practices

3.1 Stroke units
3.1.1 Patients should be admitted to a specialized, geographically defined hospital unit dedicated to the management of stroke patients A geographical unit with identifiable co-located beds (eg 5A -7, 5A- 8, 5A-9, 5A-10, 5A-11) that are occupied by stroke patients 75% of the time and has a dedicated inter-professional team with expertise in stroke care with the following professionals at a minimum nursing, physiotherapy, occupational therapy, speech language pathologist".
• This recommendation is in accordance with the HQO Effectiveness of Stroke Unit Care: A Special Report, which concluded (61): Moderate quality evidence showed that persons admitted to a stroke unit had a significant reduction in death and the combined outcome of death or institutionalization, and a nonsignificant reduction in institutionalization. Low quality evidence showed that patients admitted to a stroke unit had a significant reduction in the combined outcome of death or dependency and length of hospital stay and a nonsignificant reduction in the outcome of dependency
3.1.2 The core stroke unit team should consist of health care professionals with stroke expertise in medicine, nursing, occupational therapy, physiotherapy, speech-language pathology, social work, and clinical nutrition (a dietitian)
3.1.3 To have the necessary stroke expertise, the health care professionals on the core stroke unit team should be individuals who spend the vast majority of their time treating stroke patients and regularly complete education about stroke care
3.1.4 Patients should have access to a specialized interprofessional team 7 days a week

3.1.5 Physiotherapy, speech-language pathology, and occupational therapy team members should assess stroke patients within 48 hours of admission to hospital and formulate a management plan

3.2 Early mobilization

3.2.1 All stroke patients admitted to hospital with acute stroke should be mobilized early (between 24 hours and 48 hours of stroke onset), unless contraindicated

Frequent out-of-bed activity in the very early time frame (within 24 hours of stroke onset) is not recommended

Mobilization may be reasonable for some patients with acute stroke in the very early time frame and clinical judgement should be used.

3.3 Assessment

3.3.1 Clinicians should use standardized, valid assessment tools to evaluate patients' stroke-related impairments and functional status

3.3.2 AlphaFIM should be completed on or by day 3 after admission(target day 3, admission day is day 1)

3.3.3 All stroke patients should be assessed for risk of developing venous thromboembolism

3.3.4 Early mobilization and adequate hydration should be encouraged for all acute stroke patients to help prevent venous thromboembolism

3.3.5 The nutrition and hydration status of stroke patients should be assessed within the first 48 hours of admission using a valid screening tool. Stroke patients with nutritional concerns, hydration deficits, dysphagia, or other comorbidities should be referred to a dietitian. Referral to a dietitian should be made within 7 days of admission for recommendations and for consideration of enteral nutrition support for patients who are unable to meet nutritional and fluid requirements

3.4 Recommendations are not applicable to TIA or minor (nondisabling) stroke

3.5 Cross-continuum prevention assessment and therapies

3.5.1 All patients, whether admitted to in-hospital rehabilitation or discharged, should be given appropriate cross-continuum secondary prevention assessments and therapies (Modules 5 and 10)

Abbreviations: AHA/ASA, American Heart Association/American Stroke Association; Australia, Australian Clinical Guidelines for Stroke Management; CSBPR, Canadian Best Practices Recommendations; FIM, Functional Independence Measure; HQO, Health Quality Ontario; OSN, Ontario Stroke Network; SIGN, Scottish Intercollegiate Guidelines Network; TIA, transient ischemic attack.

The following implementation considerations were noted by members of the expert advisory panel.

Module 3a Implementation Considerations
General considerations
• All hospitals providing stroke/TIA care should ensure that the interprofessional stroke team uses standardized, validated assessment tools; where possible, these tools should be in electronic format linked to the EHR
• Patients should be screened/assessed for diabetes, dental issues, cognitive impairment, and depression upon presentation to the ED, and the results should be forwarded to the TIA clinic for follow-up
• Hospitals should ensure adequate staffing 7 days per week

Abbreviations: ED, emergency department; EHR, electronic health record; TIA, transient ischemic attack.

2.1.2 Traditional NLP Methods

Pre-LLM process extraction progressed substantially between 2011 and 2023. Early systems relied on rules and feature engineering [Halioui et al., 2018, Nasiri et al., 2023]: pattern matching and syntactic rules identified activity mentions and control-flow relations but suffered from limited coverage and weak domain transfer [Etikala et al., 2020, Quishpi et al., 2021]. Statistical sequence labeling with Conditional Random Field (CRF) and Bi-LSTM improved flexibility, yet still depended on handcrafted features and large labeled datasets, constraining scalability and generalization [Van Woensel and Motie, 2024]. Transformer architectures (e.g., BERT) further boosted semantic understanding and relation extraction, but typically required extensive domain-specific fine-tuning that is difficult in low-resource settings [Van Woensel and Motie, 2024].

End-to-end text-to-BPMN efforts showed both feasibility and brittleness. An *early, influential* pipeline automatically mapped free text to BPMN via syntactic dependencies, semantic role cues, and bespoke anaphora handling, reconstructing much of the target model yet degrading under ambiguous phrasing, abstraction shifts, and parser errors [Friedrich et al., 2011]. Complementary empirical studies of real process descriptions found recurrent sentence templates *and* intrinsically ambiguous phrasings, which motivates two mitigations we adopt downstream: controlled phrasing where feasible and validation/repair passes before and during extraction [Silva et al., 2018].

Despite these advances, pre-LLM pipelines remained fallible on naturalistic guideline prose and sensitive to domain shift, which motivates the LLM-based methods discussed next [Van Woensel and Motie, 2024].

2.1.3 LLM-based Methods

Large Language Models (LLMs) such as GPT-4 [OpenAI, 2025] have transformed process extraction by leveraging massive unsupervised pre-training and in-context learning, reducing reliance on annotated datasets. Grohs et al. [2023] showed that GPT-4 can generate structured BPMN diagrams directly from textual descriptions, matching or surpassing traditional NLP pipelines in precision and recall. Similarly, Lin et al. [2025] demonstrated that LLM produce high-quality process models with minimal human intervention, yielding substantial reductions in annotation effort and error rates. These results suggest that LLM offer an effective, scalable solution for process extraction across diverse domains.

Grathwol et al. [2025] conducted, to the best of our knowledge, the only prior study using LLM to extract CPWs from CPGs in particular. In their work, they introduced a dedicated dataset and a conceptual modeling framework, and carried out preliminary experiments with both pre-trained and few-shot LLM, specifically BioLink-BERT and OpenAI's GPT-4, to perform Named Entity Recognition (NER) and Relation Extraction (RE). The results highlighted significant challenges arising from semantic ambiguities within clinical guidelines. Their results also suggest that fully automated CPW extraction was not feasible at that time and emphasizing the need for further advancements to improve au-

tomation [Grathwol et al., 2025]. Since then, however, novel frameworks have aimed to improved operationalized LLM-driven extraction via custom output languages, error correction and orchestration; see ProMoAI/POWL (Figures 2.1–2.2) and MAO/BPMNText (Figures 2.3–2.4).

Beyond one-shot generation, recent work frames modeling as an LLM-mediated dialogue: a domain expert iteratively co-edits the model with a chatbot, applying changes via prompts. Studies report promising syntactic/semantic correctness and user satisfaction, reducing expert/modeler workload and aligning with the iterative orchestration loops considered next [Klievtsova et al., 2025a]. In this setup, the human effectively acts as an additional agent, engaging in a back-and-forth with the LLM by providing iterative feedback.

2.1.4 Recent Architectures: ProMoAI and MAO

Building on LLM capabilities, recent frameworks incorporate structured orchestration and error-handling mechanisms to further improve robustness and output quality.

ProMoAI. Kourani et al. [2024] introduced their Process Modeling with Generative AI (ProMoAI) framework, which uses POWL as its intermediary language (Figure 2.2), which supports start/end activities and two control-flow constructs—exclusive (XOR) and parallel (AND) gateways. ProMoAI divides the extraction process into two distinct stages: (1) it leverages prompt engineering to generate Python code that constructs a POWL process model, and (2) it executes this code within a Python environment to instantiate the POWL model. An iterative refinement loop feeds execution errors back to the LLM—up to five attempts—to correct syntax, achieving high success rates in generating syntactically correct process models. Figure 2.1 summarizes the ProMoAI pipeline.

In a subsequent study, Kourani et al. [2025] systematically evaluate LLMs for process modeling. They assemble a benchmark of 20 ground-truth processes, simulate event logs, and ask the LLMs to generate POWL models under a closed-label setup. This means that the prompt includes the shuffled ground-truth activity labels to standardize naming and

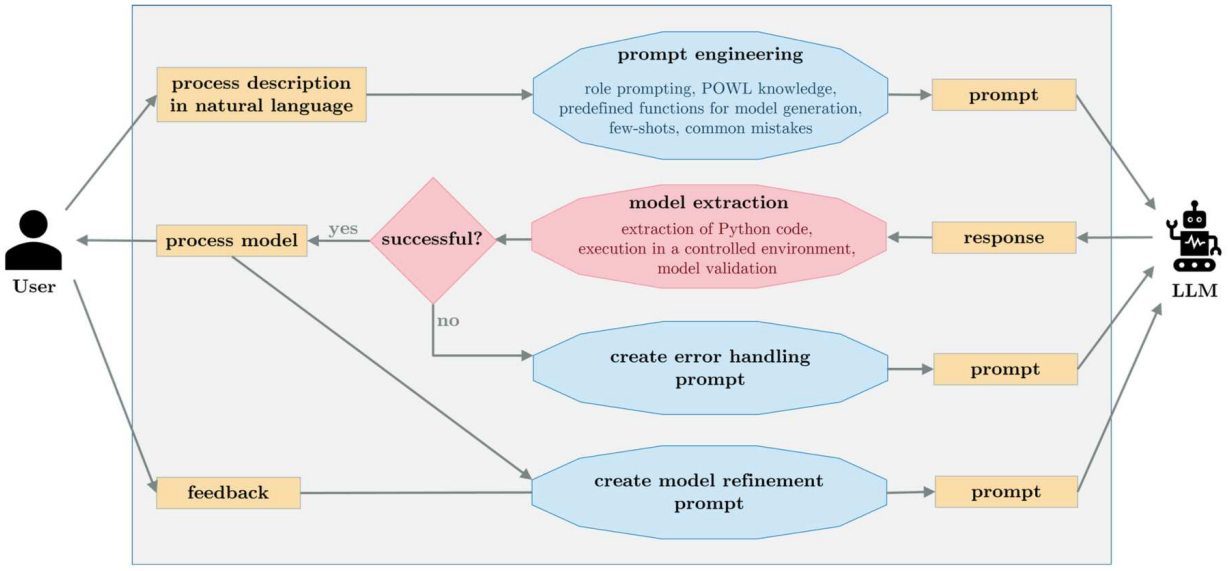


Figure 2.1: Illustration of ProMoAI's architecture from Kourani et al. [2024] (with permission).

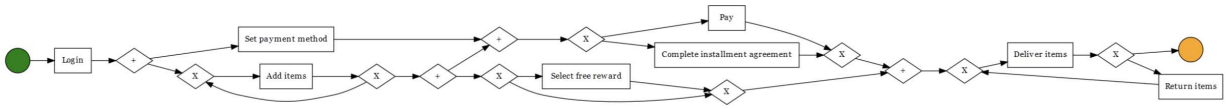


Figure 2.2: Illustration of a POWL example form Kourani et al. [2024]. It is ProMoAI's intermediary/abstract language. It encodes start/end activities and control-flow via exclusive (XOR) and parallel (AND) gateways; the POWL program is executed and exported to BPMN 2.0 in our pipeline (see §3.3).

isolate control-flow learning. An automated error-handling loop diagnoses and feeds back detected issues related to the execution of the python code until a valid model (syntactically correct) is produced, after which quality is scored by conformance checking against the simulated logs using token-based fitness and ET-Conformance precision metric [Muñoz-Gama and Carmona, 2010], aggregated via their harmonic mean. They observe substantial variance between 16 contemporary LLM, but strong intra-family consistency (i.e., consistency among models from the same provider family, such as OpenAI's GPT-4o and GPT-o1), and a positive association between fewer corrective iterations and higher conformance. They also analyze three self-improvement tactics—self-evaluation (the model generates multiple candidates and chooses via its own critique), input optimization (rewriting/clarifying the prompt or process description to improve guidance), and output optimization (the model iteratively revises its own generated POWL using self-critique and error feedback)—finding that iterative output revision yields the most reliable gains, especially for weaker models.

MAO. Lin et al. [2025] proposed their Multi Agent Orchestration (MAO) framework, comprised of four phases: generation, refinement, reviewing, and testing. Each phase assigns specialized LLM *agents* to subtasks (e.g. draft skeleton, enforce control-flow consistency, validate domain rules). In comparative experiments, MAO outperformed ProMoAI and manual modeling, delivering models with higher structural fidelity and reducing end-to-end modeling time by approximately 40% [Lin et al., 2025]. Figure 2.3 depicts the multi-agent orchestration in MAO. MAO emits *BPMN Text* (Figure 2.4)—a concise abstract language that supports start/end activities; exclusive (XOR), inclusive (OR), and parallel (AND) gateways; condition annotations; and role/object tags.

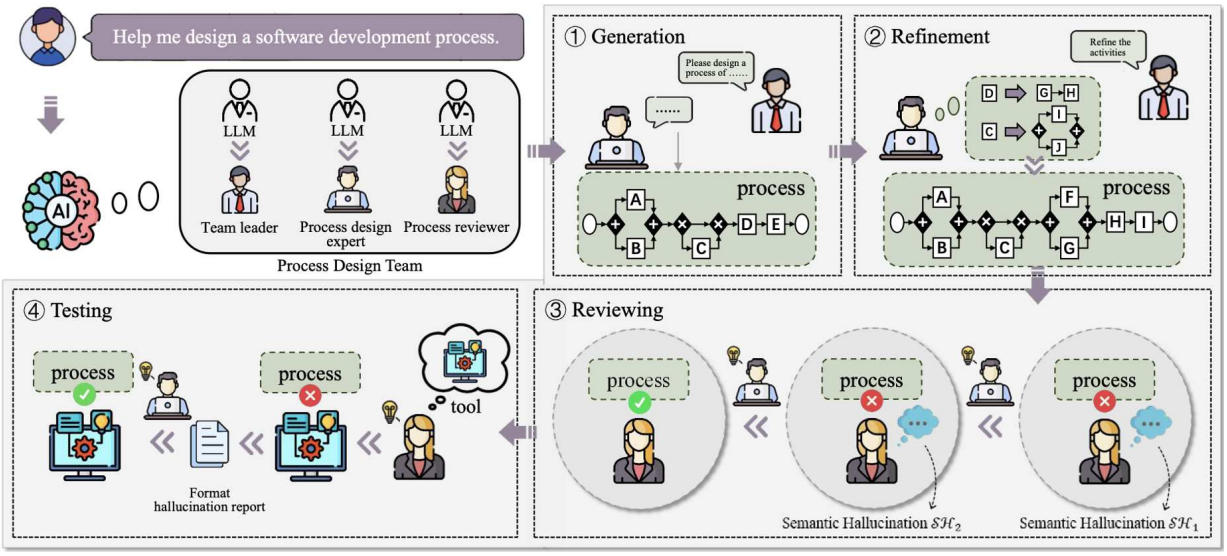


Figure 2.3: Illustration of MAO's architecture from Lin et al. [2025] (with permission): generation, refinement, reviewing, and testing phases coordinated by specialized LLM agents.

Agentic AI (operational view used here) We use "agentic" in a narrow, operational sense: an LLM-driven pipeline that (i) decomposes the task into substeps, (ii) invokes tools or structured validators, and (iii) adapts subsequent steps based on feedback from those tools or intermediate checks [Huang, 2025, Krishnan, 2025, Sapkota et al., 2025, Silfverskiöld, 2025]. Under this definition, **MAO is agentic**: its specialized LLM roles perform *generation*, *refinement*, *reviewing*, and *testing*, with conditional branching and revisions when tests or critiques fail, including tool invocation for validation [Lin et al., 2025]. MAO is also more specialized to business process models than generic Agentic

BPMN text

```
<process>
  <activity role='sender' action='prepare to send a package' objects='package' id='a1'/>
  <exclusiveGateway id='exclusivegateway1'>
    <branch condition='select pick-up' id='branch1'>
      <parallelGateway id='parallelgateway1'>
        <branch id='branch2'>
          <activity role='sender' action='confirm the pickup location' objects='' id='a2'/>
        </branch>
        <branch id='branch3'>
          <activity role='express company' action='assign a courier for pickup' objects='' id='a3'/>
        </branch>
      </parallelGateway>
    </branch>
    <branch condition='send by yourself' id='branch4'>
      <activity role='sender' action='go to the mailing point to send' objects='package, address' id='a4'/>
    </branch>
  </exclusiveGateway>
  <activity role='courier' action='start the delivery' objects='package' id='a5'/>
</process>
```

Figure 2.4: Illustration of BPMN Text from Lin et al. [2025]. This is MAO's intermediary/abstract language. It encodes start/end activities and gateway types (XOR/OR/AND), with explicit condition annotations plus role and object metadata; our pipeline translates this representation to BPMN 2.0.

AI (AAI) frameworks such as Microsoft AutoGen[1] or LangChain[2]. By contrast, we treat ProMoAI as a *tool-driven execute–repair loop* over POWL code execution rather than an agentic system in our sense [Kourani et al., 2024].

Rationale for framework selection. We focus on *ProMoAI* and *MAO* because both are suitable candidates for accurate text-to-process extraction with BPMN as the target notation. First, they pair the LLM with a constrained, intermediary language—POWL for ProMoAI and BPMN Text for MAO—that can be deterministically compiled to BPMN 2.0, enabling uniform downstream standardization and evaluation in our pipeline (see Section 3.3) [Kourani et al., 2024, Lin et al., 2025]. Moreover, they explicitly support core BPMN control-flow constructs required by clinical pathways (XOR/AND in ProMoAI; XOR/OR/AND with condition labels and role/object tags in MAO), aligning with BPMN-centric tools that are widely available and healthcare interoperability work that maps FHIR to process models [Helm et al., 2022]. Second, both frameworks incorporate validation and error-handling loops—ProMoAI's execute-repair cycle and MAO's generate-review-test—that enhance the likelihood of producing well-formed, semantically coherent models

[1] urlhttps://microsoft.github.io/autogen/stable//index.html
[2] urlhttps://www.langchain.com/

[Kourani et al., 2024, Lin et al., 2025]. Indeed, recent benchmarking work has identified such loops, also called output optimization, as the most effective self-improvement strategy, as they were shown to deliver the strongest and most consistent gains across models [Kourani et al., 2025].

2.2 Model–Model vs. Log–Model Validation

Terminology. We distinguish *validation* from *compliance checking*. **Validation** assesses the quality of a produced model against a reference (e.g., is an LLM-extracted CPW faithful to a reference CPW derived from the same text?). **Compliance checking** assesses whether observed executions (event logs or patient trajectories) conform to a normative CPW model.

Two orthogonal axes. Both validation and compliance can be conducted in either a *model–model* or *log–model* fashion. In this thesis we focus on validation:

- **Model–model validation (this thesis).** We compare an *LLM-extracted* CPW to a *gold/reference* CPW derived from the same guideline text. A principled comparison requires (i) aligning vocabularies so both models refer to the same domain concepts and (ii) computing similarity with established process-model distance measures [Dijkman et al., 2011]. Accordingly, we perform post-hoc label alignment for (i), then compute node- and structural-similarity following Dijkman et al. (with structural similarity grounded in graph-edit distance) [Dijkman et al., 2011]. Operational details are in §3.4; results appear in Chapter 6.
- **Log–model validation (related work).** Modeling quality can also be validated via conformance of (real or simulated) *logs* to a *reference* model (e.g., fitness/precision aggregates), as in recent LLM-for-process benchmarks [Kourani et al., 2025] where they used token-based fitness (how well the model reproduces the observed behavior) and ET-Conformance precision (how well it avoids introducing behavior not present in the log).

Note. The same alignment and similarity machinery can be repurposed for *compliance checking* (model–model or log–model), but we do not elaborate on compliance here.

Scope of this thesis. This thesis focuses exclusively on *model–model validation* to quantify extraction fidelity. Compliance analyses are outside our scope, but the resulting machine-readable CPWs form the prerequisite artifact for future compliance checking when combined with patient trajectories and conformance techniques, as outlined in our PM4H work [Van Woensel et al., 2025].

Relation to prior work. Recent work validates LLM-generated process models in a *log–model* manner (e.g., scoring via log conformance to a ground-truth model) [Kourani et al., 2025]. We complement that line by targeting *model–model* similarity to a reference CPW using Dijkman-style metrics, directly addressing the extraction-fidelity question central to this thesis [Dijkman et al., 2011].

2.3 Providing LLM with Contextual Knowledge

2.3.1 Knowledge Graph Integration Studies

To address issues of factual accuracy and hallucination, several works integrate KGs with LLMs. Sequeda et al. [2024] evaluated enterprise question answering over SQL databases and found that KG augmentation improved answer precision by up to 20%. In the medical domain, Su et al. [2025] introduce KGAREVION, a knowledge-graph–based agent that has an LLM generate candidate biomedical triplets from the question, reviews them by verifying against a grounded KG with a KG-completion–fine-tuned reviewer, revises any errors, and then answers—yielding notable accuracy gains on standard medical QA benchmarks. Pan et al. [2024] surveyed LLM-KG synergy, proposing three paradigms—KG-enhanced LLMs, LLM-augmented KGs , and synergized LLM+KG systems—that demonstrate reduced hallucinations and improved reasoning.

2.3.2 Approaches for Integration

Yang et al. [2024b] categorize LLM–KG integration into three primary strategies:

- **Pre-training enhancement**: enriching the pre-training corpus with KG-derived structure and signals to ground LLM representations in factual entities and relations. Representative patterns include (i) expanding input structures with KG context, (ii) enriching corpora with entity links and relation annotations, (iii) generating synthetic KG-grounded text, and (iv) applying entity/relational masking that biases representations toward factual units.

- **During-training integration**: incorporating graph-structural information into the model via joint optimization objectives that align textual and KG embeddings. Common realizations include coupling a knowledge encoder with the LLM, inserting knowledge-aware layers or adapters, modifying pretraining objectives to incorporate KG constraints, and multi-task alignment of text–graph representations.

- **Post-training fine-tuning**: adapting a pre-trained LLM on knowledge-augmented datasets or KG-grounded tasks to refine domain-specific reasoning. This stage also includes lightweight, inference-time *prompt-based knowledge injection*, such as:

  - **Embedding verbalized KG snippets (free-form fact injection).** *What it injects:* short natural-language sentences that directly verbalize KG triples/-subgraphs relevant to the query. *How it is formed:* retrieve a small subgraph around the query entities and convert each (head, relation, tail) into declarative text; concatenate before/after the user query without a fixed schema. *Training needed:* none (pure prompting). *When to use:* to ground a single question with explicit facts when simple factual context suffices [Yang et al., 2024b].

  - **Knowledge-aware prompt templates (schema-guided prompts).** *What it injects:* the *same facts* as above, but slotted into a *structured template* (discrete or continuous) that dictates where entities/relations appear (e.g., "Given {entity}, facts: $\{(r_1, t_1), \ldots\} \rightarrow$ answer ..."). *How it is formed:* automatically build a knowledge subgraph for the current context, transform it into

natural language via learned rules, and *optimize the template itself* (e.g., with prompt-relevance inspection and masked-prompt modeling) so the LLM pays attention to the right facts. *Training needed:* optional lightweight self-supervision to learn/score templates, not to change the LLM's core weights. *When to use:* when consistent formatting and better controllability over what the model reads is desired [Yang et al., 2024b].

– **Dynamic few-shot prompting with KG retrieval (in-context exemplars).** *What it injects: examples* (e.g., [question, KG-derived facts] → answer) selected/constructed at inference time from a KG (or a dynamically built KG) and verbalized into natural language, so the model learns by analogy within the prompt. *How it is formed:* retrieve tuples/examples most similar to the current query, verbalize them, and place them as few-shot demonstrations before the query. *Training needed:* none—this is in-context learning. *When to use:* when patterns from nearest factual cases help the model generalize or when up-to-date KG facts must be injected per-query without parameter updates [Yang et al., 2024b]. Our approach is similar to this, but instead of examples, we add knowledge statements.

These prompt injection methods require no additional training or fine-tuning, offering a highly flexible mechanism to keep LLM outputs grounded in authoritative knowledge bases at runtime [Yang et al., 2024b].

2.4 Chapter Summary

This chapter established the conceptual and technical groundwork for the thesis. We discussed the need for automatic extraction of clinical pathways from free-text guidelines within scalable process-mining and clinical decision-support pipelines. We traced the methodological arc from rule-based and classical machine-learning approaches to LLM-centric pipelines with orchestration, highlighting ProMoAI (execute–repair loops) and MAO (multi-agent generation–review–test) and situating them within the broader agentic-

AI paradigm. We clarified compliance checking by distinguishing model–model comparison (evaluated here to assess extraction accuracy) from log–model conformance (the downstream use-case that extracted CPWs enable). Finally, we surveyed strategies for enriching LLMs with external knowledge—especially knowledge-graph integration across pre-/during-/post-training and prompt-time injection.

Chapter 3

LLM4CPW Pipeline

In this chapter, we introduce a comprehensive, modular pipeline named LLM for Clinical Pathways (LLM4CPW) that leverages LLMs to automate the extraction of process models from free text, together with their standardization, evaluation, and visual comparison of evaluation metrics. Correspondingly, our pipeline comprises four core components—LLM Execution, Output Standardization, Automated Evaluation, and User Interface—each designed with clear inputs, processing steps, and outputs which are preserved to ensure reproducibility and extensibility. As per the name of the pipeline, we focus primarily on CPWs, and developed specialized prompt templates for them as part of the LLM Execution component. However, the architecture can accommodate any textual process description given suitable prompt templates. Figure 3.1 illustrates the end-to-end pipeline, from textual input through model generation and standardization to quantitative assessment and user interaction. The implementation is publicly available online, complete with deployment instructions and example configurations [Houshidari, 2025].

3.1 Pipeline Overview

Our pipeline includes four modular components—(1) LLM Execution, (2) Output Standardization, (3) Automated Evaluation, and (4) User Interface.

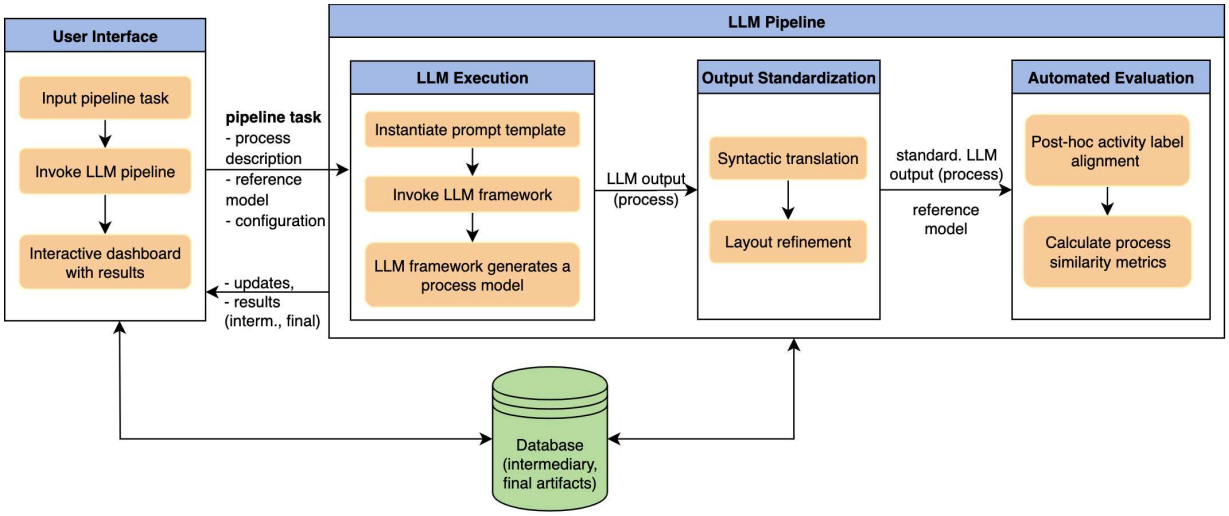


Figure 3.1: High-level overview of the thesis' LLM4CPW pipeline (replicated from Fig.1.2 for convenience).

Given a CPW text and a configuration, the *LLM Execution module* invokes the selected LLM framework, which will produce a process model in a particular format (e.g., Partially Ordered Workflow Language (POWL) script or BPMN-Text). These intermediate outputs are passed to the *Output Standardization* module, which transforms them into standard BPMN diagrams and applies auto-layout for readability. The *Automated Evaluation* module accepts the resulting BPMN models, together with a gold-standard reference BPMN, and outputs a comparison of the two in terms of similarity metrics. The pipeline is accessible via a web-based User Interface (UI) for experiment configuration, running, and visual comparison of evaluation / similarity metrics. All artifacts (logs, model files, evaluation scores) are persisted to a relational database. The following sections discuss the main modules of LLM4CPW; Figures 3.2–3.10 depict their inputs, processing steps, and outputs.

3.2 LLM Execution (Module I)

A pipeline *task* (see Fig. 3.1) bundles two elements: process description (e.g., input guideline) and a configuration that includes the underlying LLM framework (currently, ProMoAI or MAO), the foundation LLM model to use (e.g., `gpt-4.1` [OpenAI, 2025]), prompt-

template settings (i.e., predefined setup to use), the number of runs, and a project/job identifier. Figure 3.2 illustrates Module I, from narrative inputs and configuration to the raw process output returned by the selected framework, with failure logging. To make this concrete, Appendix A presents a verbatim excerpt of MAO's raw BPMN-Text (textual output since BPMN-text does not provide integrated visualization tooling) for the same QBP instance used in previous chapter (see Listing 2.1.1). Meanwhile, Fig. 3.3 shows the corresponding ProMoAI raw visualized output in POWL (output is also text but POWL has visualization support).

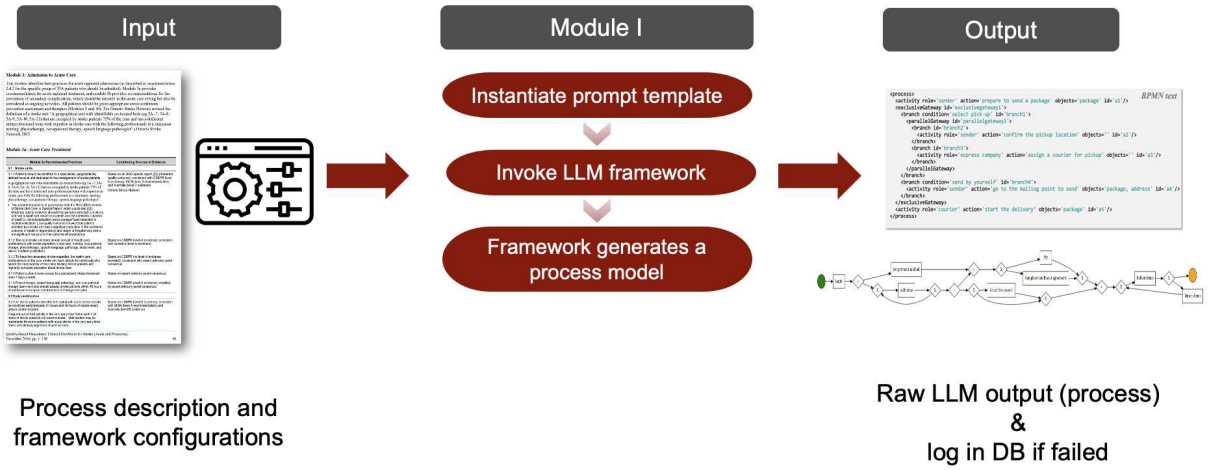


Figure 3.2: Module I — LLM Execution. Inputs (process description and configuration) are injected into a framework-specific prompt, the framework is invoked, and a raw process model is returned; failures are logged for reproducibility.

When a *task* is submitted, this component injects the process description into the framework-specific prompt template, and then dispatches the filled prompt to the chosen LLM framework via its pipeline adapter. During execution, each framework emits structured events—such as "request sent", "partial response", "retry attempt", and "final response"—to a (framework-agnostic) pipeline listener. An LLM framework will have internal retry logic to perform error recovery (e.g., ProMoAI's self-refinement loop, MAO's multi-agent review). The listener captures and persists these events, together with metadata (timestamps, input/output paths, retry counts). Note that the "final response" event encapsulates the generated process model as this component's output.

Our modular design means new LLM frameworks can be easily plugged in, by adding

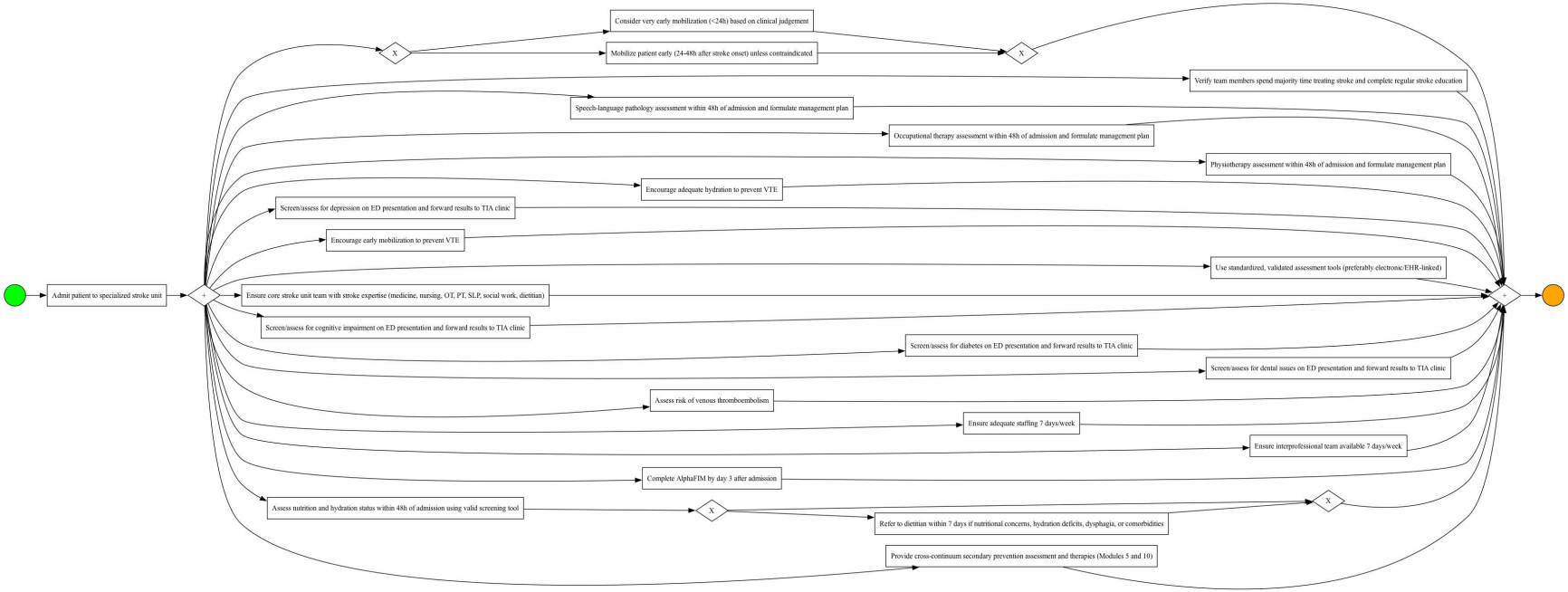


Figure 3.3: ProMoAI raw output for the QBP example (POWL). See Listing 2.1.1 for the corresponding input.

the provider's Software Development Kit (SDK) and implementing an adapter and listener mapping.

3.3 Output Standardization (Module II)

Next, this component accepts the generated process models and converts them into standardized BPMN 2.0. This allows for their automated evaluation using the same component and similarity metrics (Section 3.4), and their visual interpretation by clinicians and process analysts with knowledge of BPMN. We provide a canonicalization module *only* for the two upstream frameworks evaluated in this work—MAO and ProMoAI—translating their intermediary formats (BPMN-Text and POWL, respectively) into BPMN 2.0. Figure 3.4 therefore illustrates the two conversion approaches and the inputs/outputs of this module. Applying this step yields the canonical BPMN models shown in Figure 3.5 (MAO) and Figure 3.6 (ProMoAI). The component is modular; additional frameworks can be supported by implementing a small adapter that maps their intermediate representation to BPMN 2.0.

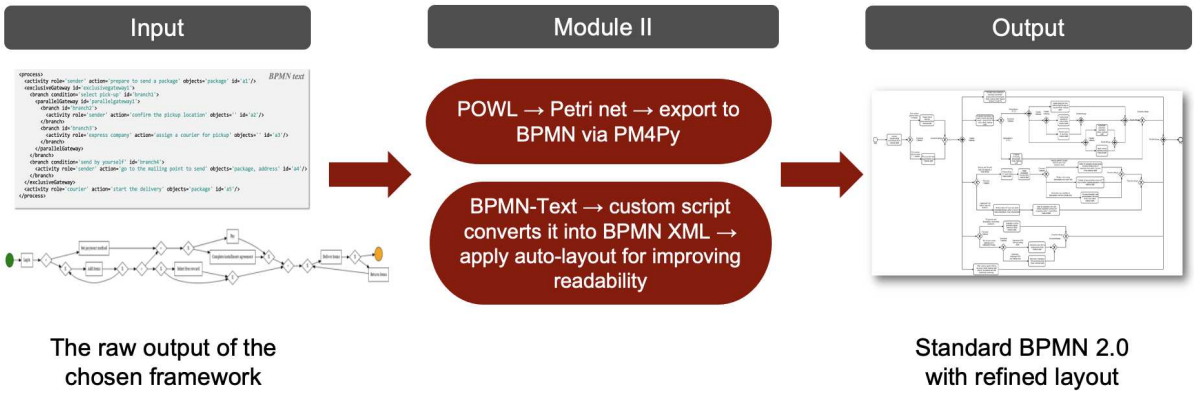


Figure 3.4: Module II — Output Standardization. The result is canonical BPMN 2.0.

Depending on the LLM framework output, steps may include *syntactic translation* as well as *layout refinement*:

- ProMoAI. For *syntactic translation*, we translate the generated POWL script into a Petri net and then into BPMN using the PM4Py library [Process Intelligence

Solutions, 2025]. *Layout refinement* was already performed by PM4Py in the former step.

- MAO. For *syntactic translation*, a custom script parses its generated BPMN-Text and emits standard BPMN XML. For *layout refinement*, we call an external auto-layout library [Barelkowski and Tulaev, 2025] for node coordinates and improving readability.

MAO conversion overview. As ProMoAI already provides a conversion algorithm implemented in the PM4Py library—where the transformation to BPMN is handled internally—we focus our description here on MAO's conversion process. Our custom converter compiles MAO's concise BPMN-Text into standards-compliant BPMN 2.0 XML. At a high level, it reads the process in order, turning activities into BPMN tasks and mapping control-flow elements (parallel, exclusive, and inclusive choices) to the corresponding BPMN gateways. Branch conditions are preserved as labels so the intent of each path remains explicit. The converter also enforces a consistent start–end skeleton (with proper start/end points) and unique identifiers, yielding a canonical representation that can be evaluated with the same metrics across models (Section 3.4) and inspected in any BPMN tool. The full conversion script (`BPMN Converter`) is provided in Appendix B.

MAO layout strategy. Since MAO does not provide diagram layout (or any tool that does it for BPMN-Text), the generated diagram is not human-readable. To make diagrams readable, we apply an automatic layouter [Barelkowski and Tulaev, 2025]. Because the generic layouter had difficulty determining where a branch should begin when it contained multiple steps, the converter temporarily adds small anchor nodes at the start of such branches to help the layouter correctly position the connecting edges. A lightweight post-processing pass then removes these anchors and reconnects the lines, preserving the original control flow while still producing an improved layout. The full script for the `Remove Extra Activities`) is provided in Appendix B. The visual impact of auto-layout is illustrated by the full diagram before auto-layout in Figure 3.7, and a zoomed-in detail of the auto-

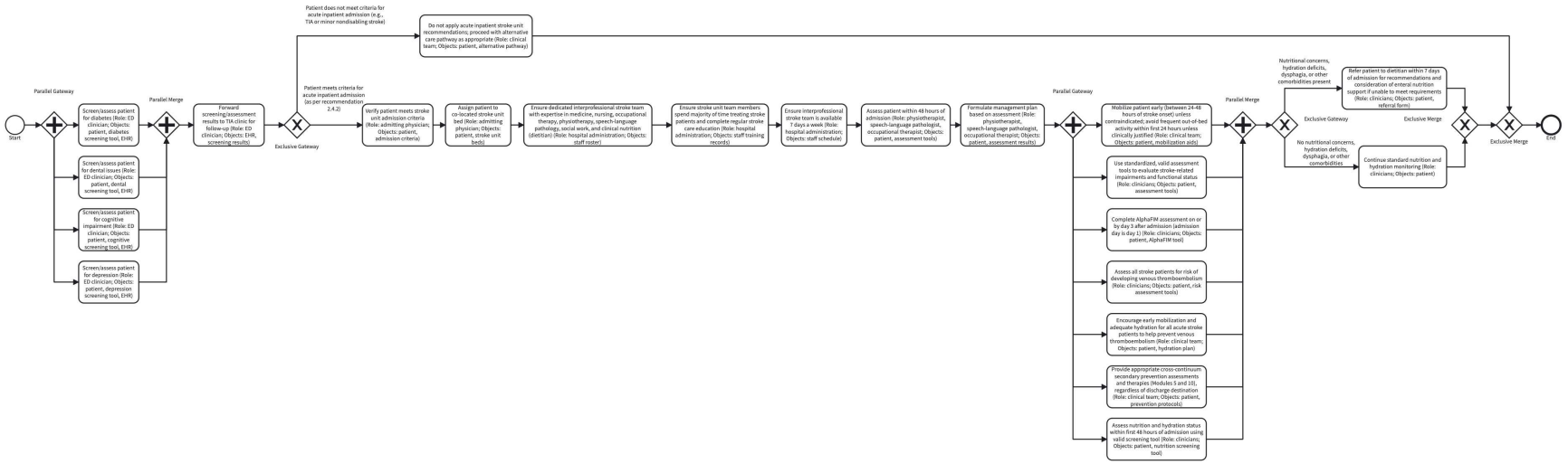


Figure 3.5: MAO standardized output for the QBP example (BPMN). See Listing 2.1.1 for the corresponding input. Minor changes made manually to fit in this area.

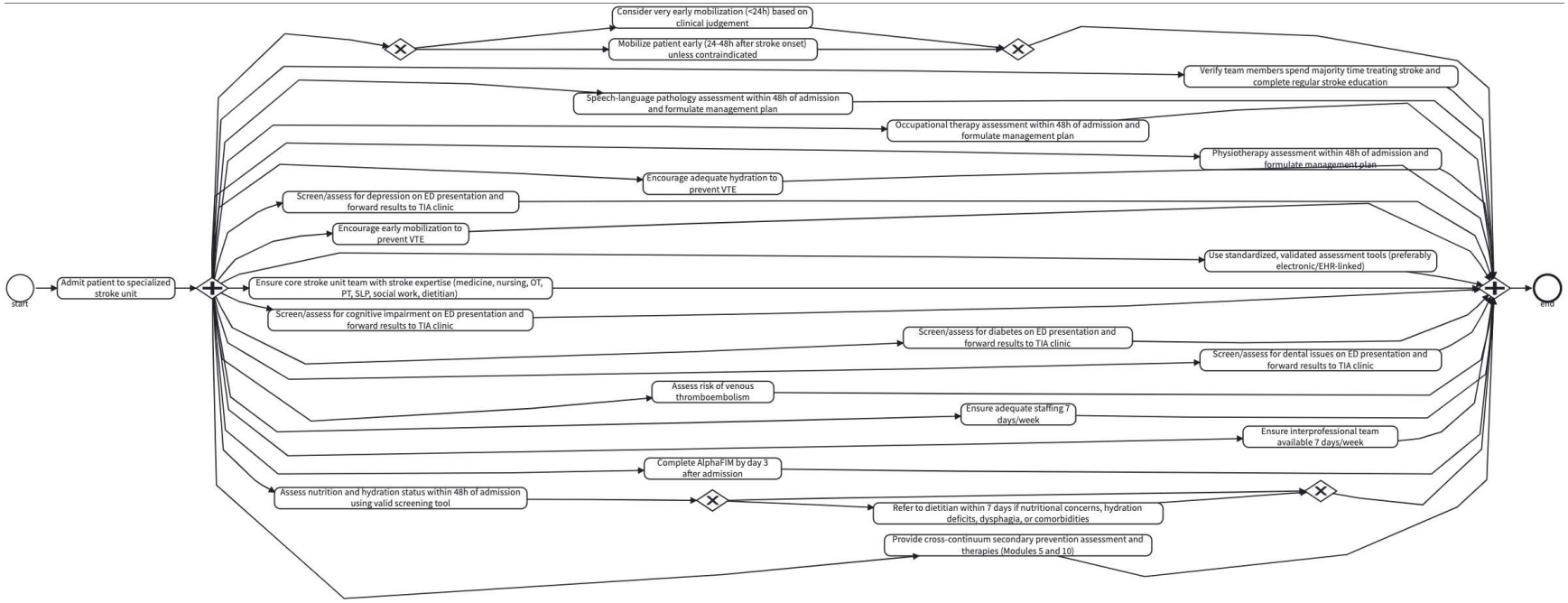


Figure 3.6: ProMoAI standardized output for the QBP example (BPMN). See Listing 2.1.1 for the corresponding input.

laid-out diagram highlighting the temporary branch-entry anchor is shown in Figure 3.8; the complete, cleaned diagram after removing the anchor already appeared in Figure 3.5. Since BPMN-Text lacks a layout, BPMN visualizers automatically assign layout attributes after conversion to BPMN 2.0, resulting in the pre-auto-layout version shown in Figure 3.7.

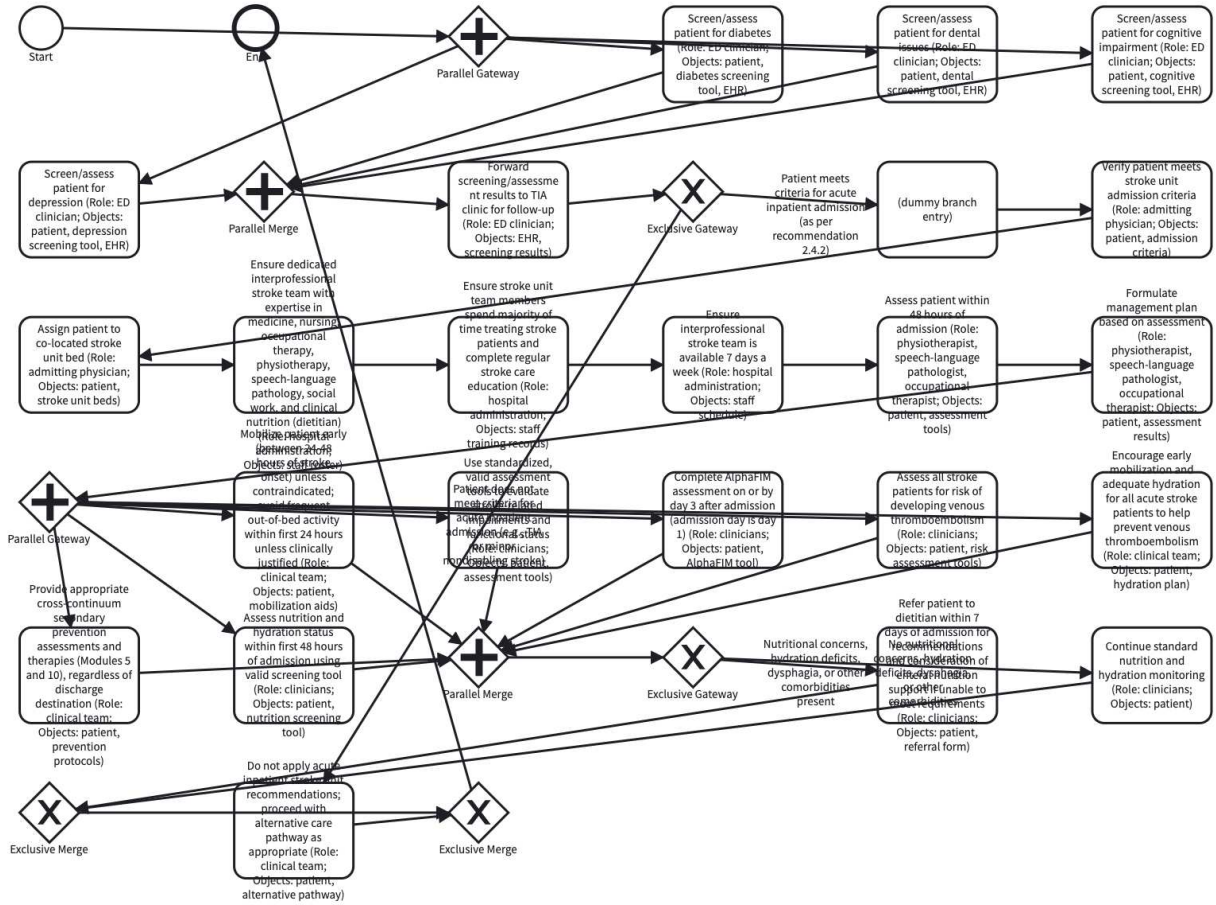


Figure 3.7: MAO's standardized output for the QBP example (BPMN) before auto-layout. See Listing 2.1.1 for the corresponding input.

3.4 Automated Evaluation (Module III)

The *Automated Evaluation* component takes as inputs (**a**) the generated BPMN model, as the standardized output of an LLM framework (e.g., POWL or BPMN Text), and (**b**) an expert-validated reference model, and then calculates similarity scores between (a) and (b). This component involves the following steps:

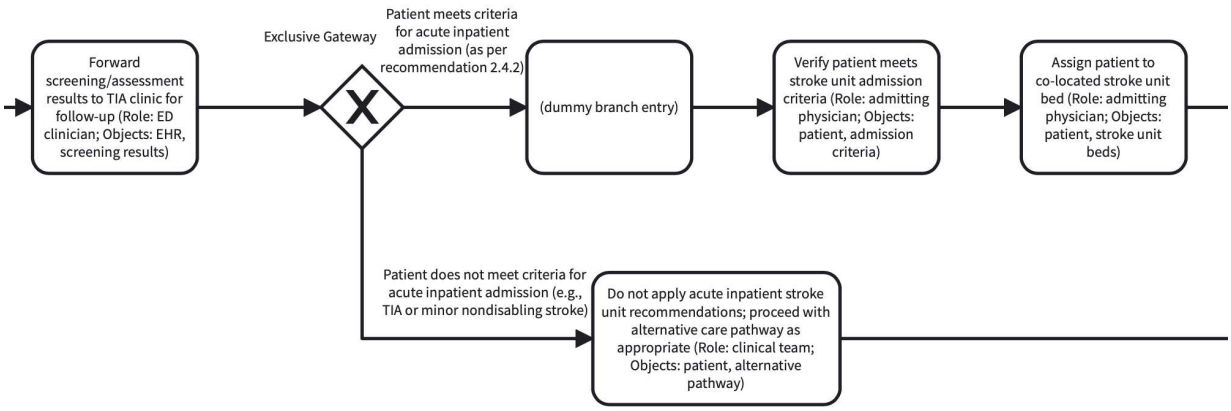


Figure 3.8: Excerpt of MAO's standardized output for the QBP example (BPMN) after auto-layout and before removing the extra activity for a branch that has several activities. See Listing 2.1.1 for the corresponding input.

Post-hoc activity label alignment.

In our experience, the LLM-generated activity labels are often semantically similar to the reference labels; however, they are too far apart syntactically for existing similarity metrics (e.g., using string edit distance). We thus first attempt to align the labels from models (a) and (b). Specifically, we extract the list of activity names from both models and invoke an LLM model (currently, GPT-4.1) with a structured prompt that specifies the alignment task and desired output format. The first LLM API call asks for a mapping from generated labels to reference labels; the second call asks to verify and refine each match. We then apply this mapping to update the activity labels in the generated model to match the reference terminology. Our initial experiments demonstrated that a single LLM invocation for activity alignment often produces imperfect label mappings; issuing a second, refinement-focused call more reliably yielded the correct alignments. As illustrated in Figure 3.9, this step aligns each activity label in the generated model with its corresponding label in the reference model when available, while retaining the original generated label in cases where no match is found. The script for the post-hoc activity label alignment is provided in Appendix B.3.

Unlike a prior study [Kourani et al., 2025]—that append the ground-truth activity-label list (in shuffled order) to the prompt and require the LLM to reuse those labels so their log-based conformance scores remain comparable—we deliberately *withhold* labels and perform

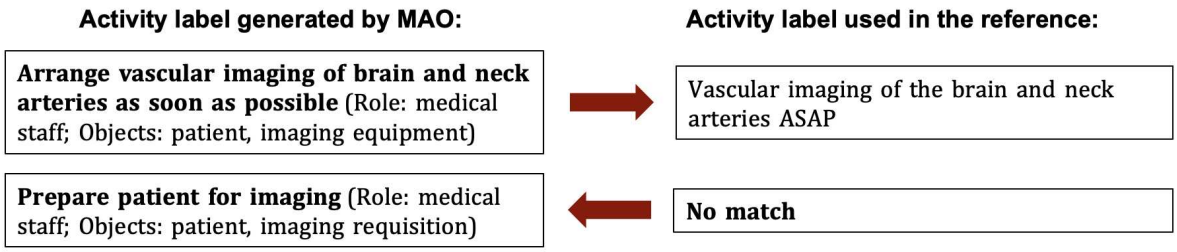


Figure 3.9: Post-hoc activity label alignment. Example showing how generated labels (left) are mapped to the reference terminology (right) prior to similarity computation.

post-hoc label alignment. Post-hoc alignment maps the model's discovered activity names to the reference vocabulary only for evaluation, removing superficial wording differences while *preserving* the model's choices about which activities to include, merge, or split. This allows us to also assess activity discovery and granularity. Our setup is feasible because we evaluate model-to-model via structural similarity after alignment, which does not require identical activity counts or a one-to-one label correspondence.

Calculate process similarity metrics.

We compare the aligned process models by calculating node-similarity, structural-similarity, and graph edit distance (GED) using Dijkman's 2011 method [Dijkman et al., 2011]. We selected Dijkman's method over alternatives (EGED [Waspada and Sarno, 2020], BPMN-Sim [Garcia et al., 2023]) because it natively incorporates label matching, captures gateway types in its soft-cost GED, and has an open-source implementation [Wamej et al., 2025]. We implemented and tested all three methods, and found Dijkman's approach to be more accurate and better at discriminating high vs low quality models for our use case. Moreover, we also considered a broad set of published tools and algorithms, including BPMNDiffViz [Ivanov et al., 2015], the NetworkX GED functions [NetworkX Developers, 2025], GMatch4py [Fize, 2019], BP-Diff [Armas-Cervantes et al., 2014], the Behavioral Similarity metric [Kunze et al., 2011], Apromore's BPMN comparison tool[Apromore Team, 2025], and bpmn-js-differ [bpmn-io team, 2024]. These were excluded from further consideration as they either addressed only a restricted set of structural or behavioral differences, or lacked an accessible, maintained implementation that could be adopted in our pipeline. Figure 3.9 gives a concrete example of post-hoc label alignment; Figure 3.10 summarizes

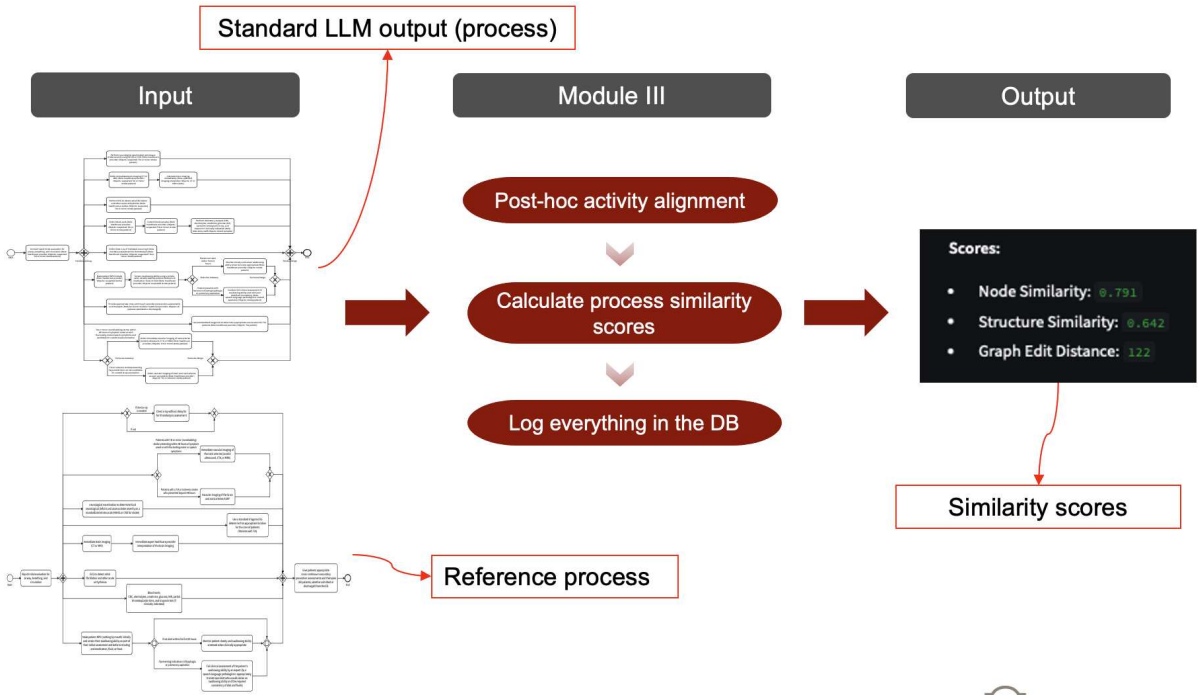


Figure 3.10: Automated evaluation workflow. The aligned generated model and the reference model are compared using Dijkman's node/structure similarity and graph edit distance; all artifacts and scores are persisted.

the evaluation workflow and outputs. The pipeline's modular design also allows plugging in other evaluation metrics.

3.5 User Interface (Module IV)

Users interact with LLM4CPW through a web interface that provides two entry points: a *Pipeline Runner* for configuring and launching experiments, and an *Experiment Results Dashboard* for analysis and export of outcomes (generated process models, similarity metrics, and dashboard visualizations). The UI provide access to intermediate and final artifacts, and support rapid comparison across frameworks and guideline instances.

Notation. Figures 3.11–3.14 include numbered call-outs **[1–n]** overlaid on the screenshots; beneath each figure, a short *Legend* explains the role of every field and control.

Legend for Figure 3.11.

① OpenAI API key — secure input for authenticating OpenAI LLM calls (visibility toggle

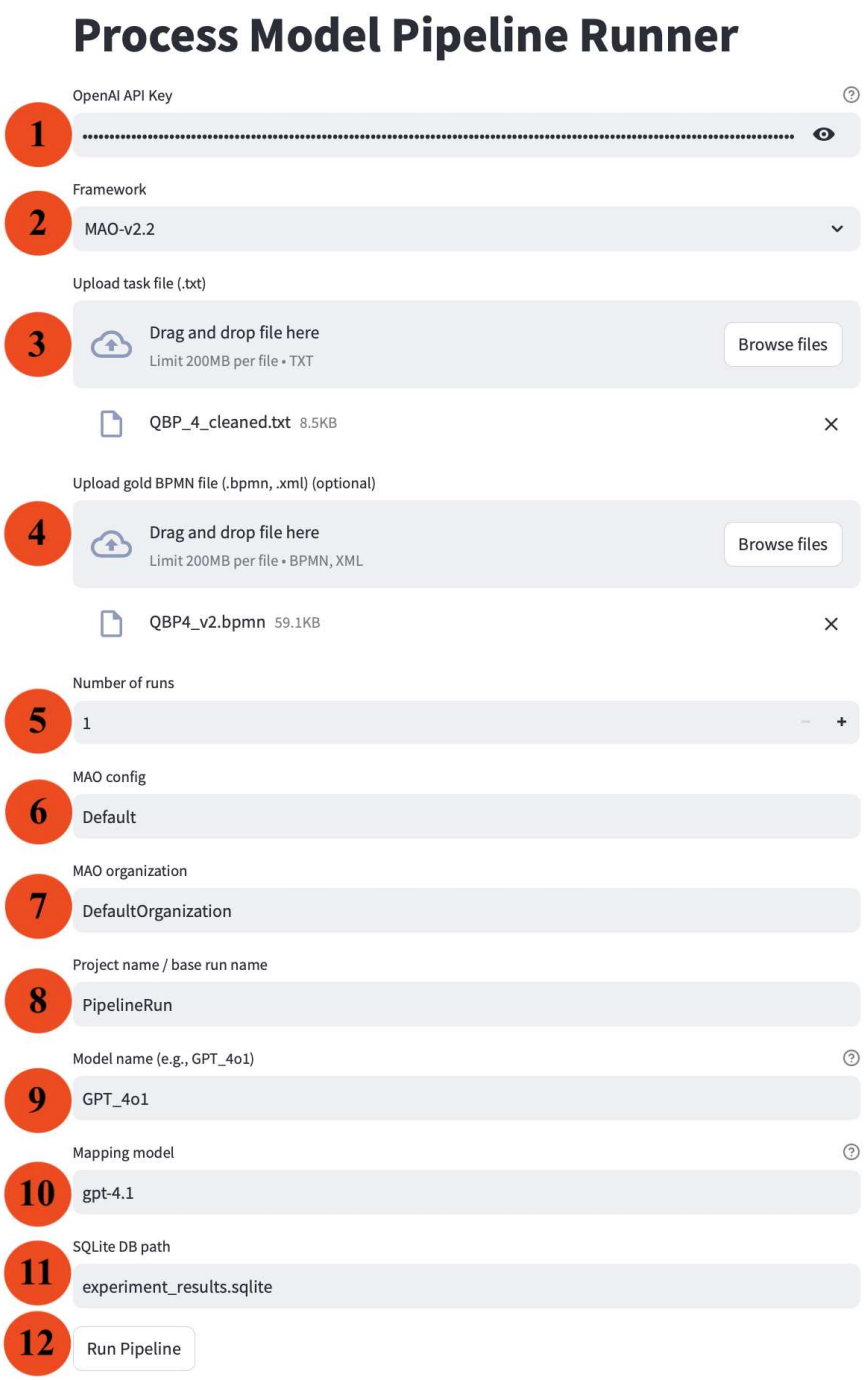


Figure 3.11: Pipeline Runner — configuration form. Users select the framework, upload the task text and optional gold BPMN, set run and model parameters, and specify the results database.

via eye").

② Framework — selected extraction framework (MAO or ProMoAI).

③ Task file (.txt) — Input guideline (or other natural language process description) (re-

quired).

④ Gold BPMN (optional) — expert reference used for evaluation and label alignment.

⑤ Number of runs — batch size for identical executions (useful for generating descriptive statistics of multiple executions).

⑥ MAO config — preset defining agent budgets and phase counts (active for MAO).

⑦ MAO organization — namespace used in output paths and artifact grouping.

⑧ Project/base run name — prefix for artifacts and database entries.

⑨ Model name — identifier of the LLM used (stored with results).

⑩ Mapping model — LLM used for post-hoc label alignment.

⑪ SQLite DB path — path to the results database consumed by the dashboard.

⑫ Run Pipeline — control to execute the pipeline with the current settings.

Pipeline Runner (configure and launch). Figure 3.11 shows the configuration form. Users provide their OpenAI API key, select the extraction framework (e.g., MAO or ProMoAI), upload the narrative process description and an optional gold BPMN reference file, and specify run parameters (number of runs, framework configuration, organization/project identifiers, model names, and the path to the results SQLite database). After execution, the form presents structured logs, similarity scores, elapsed time, and file paths (Figure 3.12); it also exposes one-click downloads for both the *generated* BPMN and the *mapped* BPMN produced after label alignment.

Legend for Figure 3.12.

① Execution log — shows structured, real-time events and logs.

② Command — shows the command used, for validating and debugging

③ Status banner — shows success or descriptive error.

④ Scores — shows node similarity, structure similarity (higher is better), GED (lower is better).

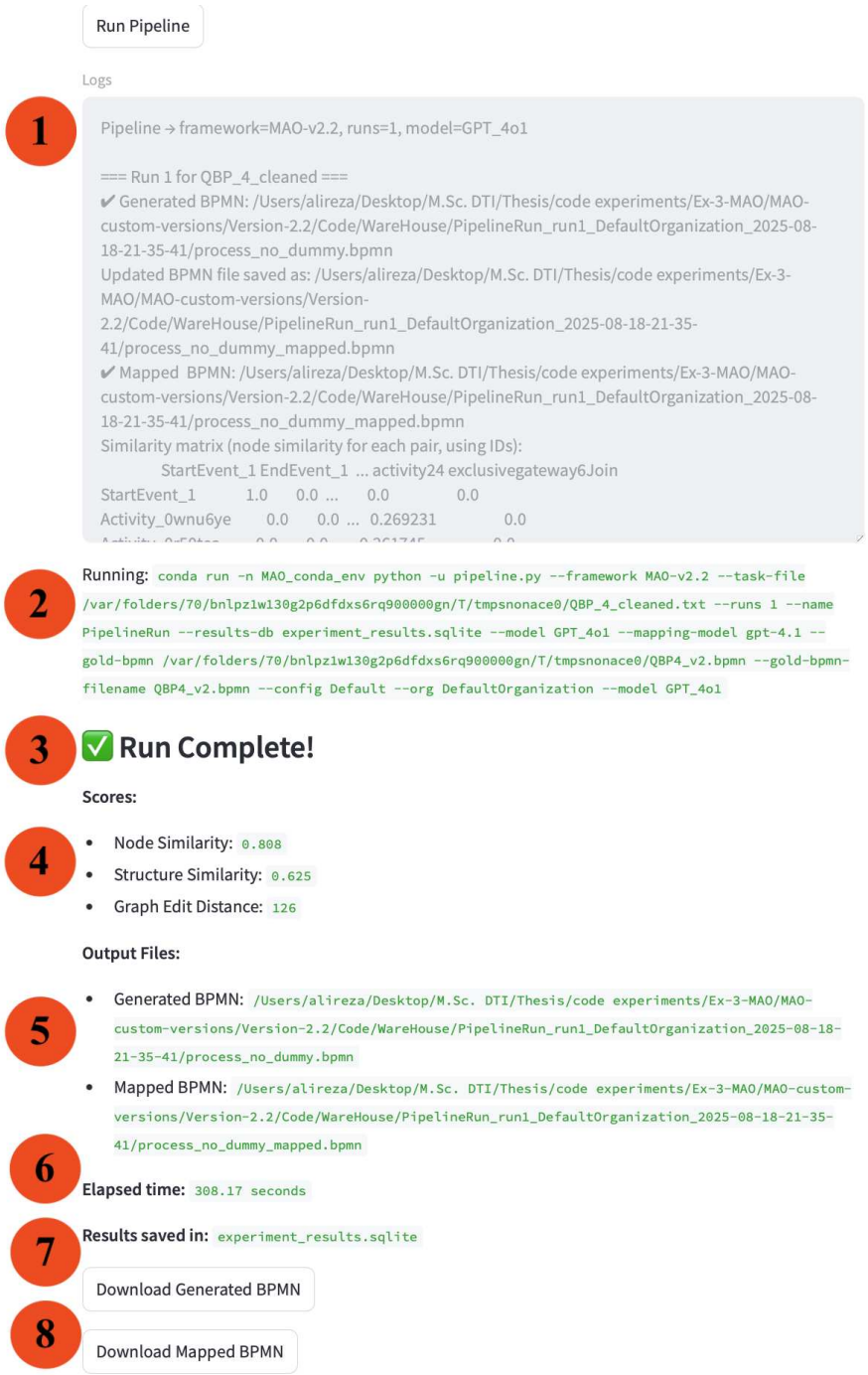


Figure 3.12: Pipeline Runner — run summary. The interface displays structured logs, similarity scores, elapsed time, and file paths, and provides direct downloads for the generated and mapped BPMN models.

⑤ Output files — shows absolute paths for generated and mapped BPMN models.

⑥ Elapsed time — shows duration of the run.

⑦ Results database — confirms the SQLite file updated.

⑧ One-click exports — downloads generated/mapped BPMN for inspection.

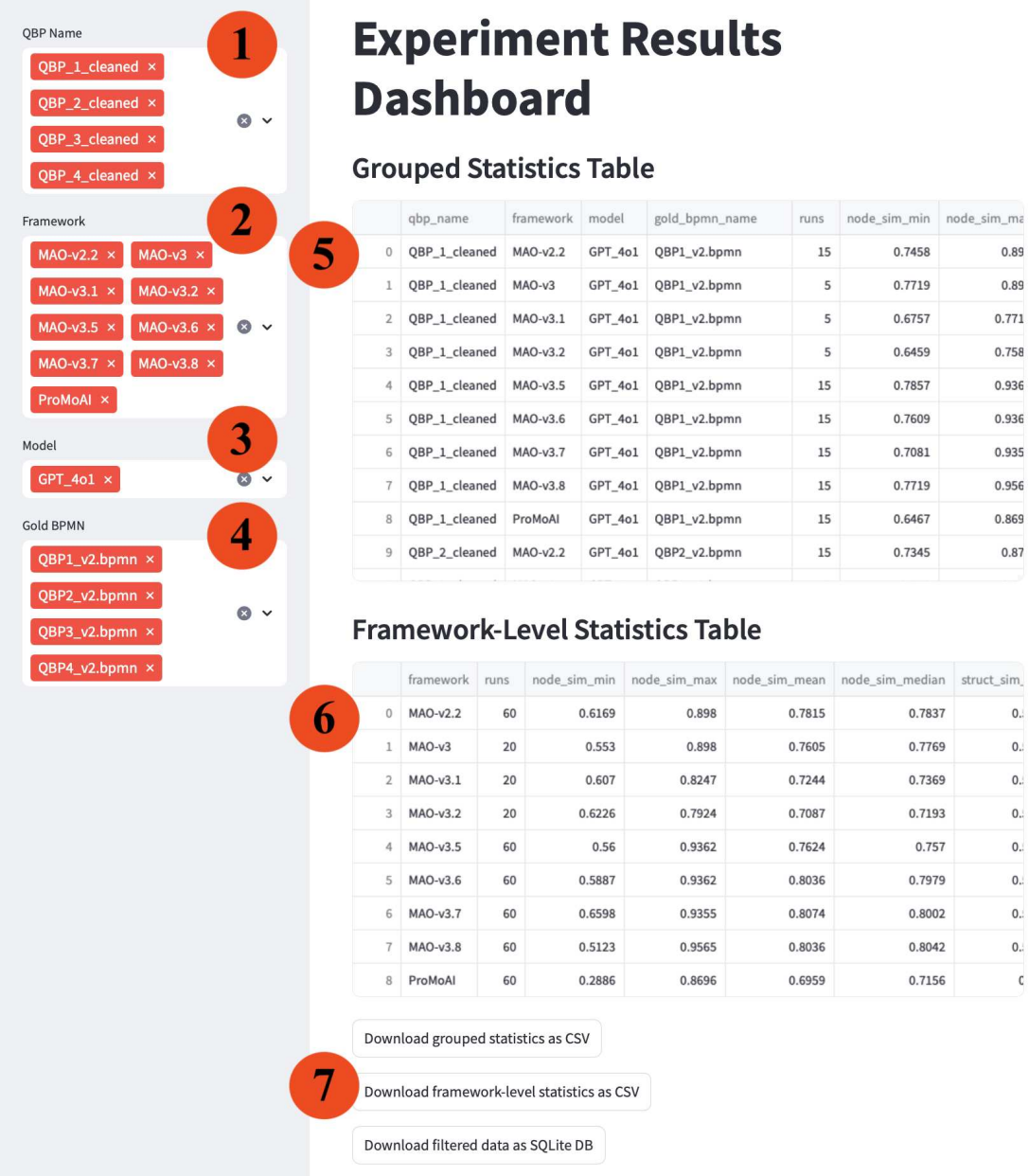


Figure 3.13: Experiment Results Dashboard — overview. Interactive filters (left) and grouped/framework-level statistics tables (right), with export of the *currently filtered* results to CSV or SQLite.

Legend for Figure 3.13.

① Input process filter — multi-select guideline instances; chips removable with "×".

② Framework filter — MAO/ProMoAI selection.

③ Model filter — LLM tag saved at run time (e.g., `GPT_4o1`).

④ Gold BPMN filter — choose reference model(s) for evaluation.

⑤ Grouped statistics — per (input process, framework, model, gold) with runs and min/-max/mean/median for all metrics.

⑥ Framework-level statistics — aggregated by framework across current filters.

⑦ Export current view — CSVs for the two tables; SQLite for the filtered underlying rows.

Experiment Results Dashboard (analyze and export). The dashboard reads the results database and provides interactive filtering by input process name, framework, model, and gold reference (Figure 3.13). It renders grouped and framework-level statistics tables and, for convenience and reproducibility, allows users to download the *currently filtered* data either as CSV files or as a compact SQLite database. In addition, users can export the *filtered tables and charts* for reporting.

Figure 3.14 illustrates the visualization controls (e.g., bar, dot, difference-from-min) and example plots for node similarity, structure similarity, and graph edit distance. Figure 3.15 shows the chart types available in the dashboard (Bar, Per-Process grouped, Stacked bar, Dot, and Difference-from-min).

The full application is containerized and can be deployed from a public image with minimal configuration [Houshidari, 2025].

Legend for Figure 3.14

① Chart-type selector — Allows selecting the type of chart (Bar, Dot, or Difference-from-Min) per panel.

② GED Mean — allows viewing the GED means. Lower is better; "Difference-from-Min" shows distance above best GED.

③ Node Similarity Mean — allows viewing the mean by framework chart; legend toggles frameworks.

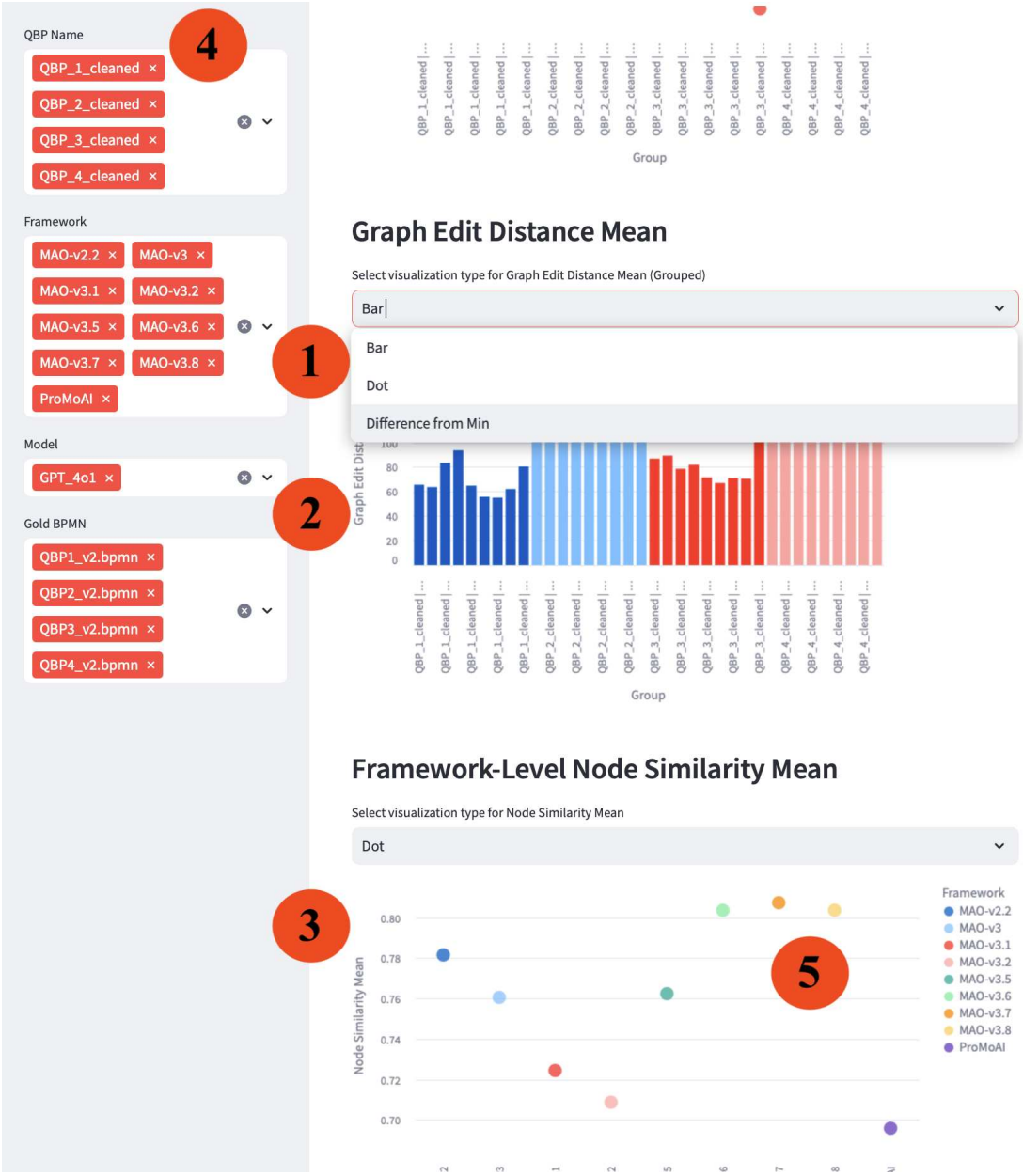


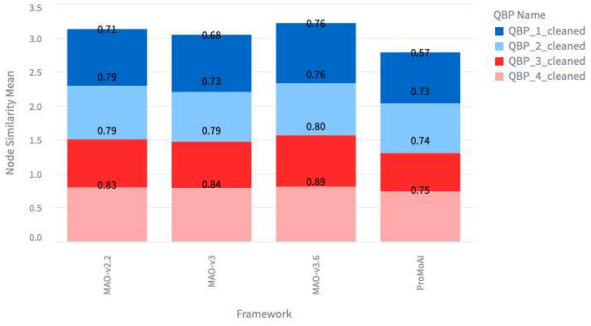
Figure 3.14: Experiment Results Dashboard — interactive visualizations. Users can switch chart types and download the *filtered* tables and charts for reporting.

④ Linked filtering — allows filtering the results. All charts/tables recompute with the left-sidebar filters.

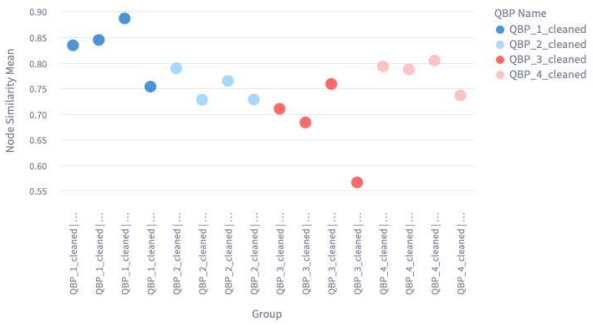
⑤ Download — allows downloading the plots.

(a) Bar.

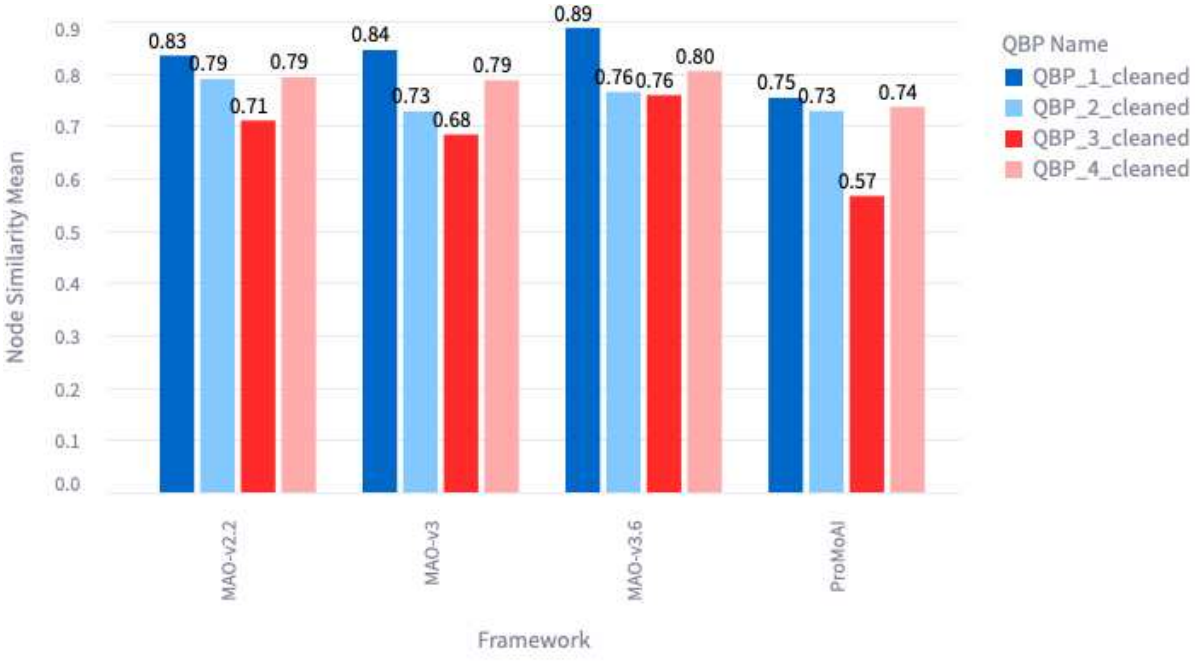
(b) Difference-from-min.



(c) Stacked bar.



(d) Dot.



(e) Per-Process grouped (bar).

Figure 3.15: Dashboard chart types supported by LLM4CPW. Each panel shows a different visualization option available in the Experiment Results Dashboard.

3.6 Chapter Summary

This chapter has detailed the design and implementation of our four-stage LLM pipeline (LLM4CPW). Section 3.1 provided an architectural overview, situating each module within the broader experimental workflow. Section 3.2 described the LLM Execution module, which injects narrative descriptions into framework-specific prompt templates, orchestrates calls to ProMoAI or MAO, and logs all intermediate events. Section 3.3 covered Output Standardization, where heterogeneous model outputs (POWL, BPMN-Text) are converted into BPMN 2.0 XML with auto-layout for downstream evaluation and human inspection. Section 3.4 explained the Automated Evaluation component, including post-hoc label alignment via GPT-4.1 and the computation of node-, structural-, and GED metrics based on Dijkman's [Dijkman et al., 2011] method. Finally, Section 3.5 introduced the web-based User Interface, which supports experiment configuration, task execution, visual comparison of evaluation / similarity metrics, and artifact download. Together, these components constitute a modular framework for studying and comparing LLM-driven process extraction approaches.

Chapter 4

Incorporating Contextual Knowledge

This chapter presents the motivation and methodology for incorporating contextual knowledge into the proposed process-extraction frameworks. We begin by discussing the underlying motivations driving this integration. We then describe the systematic approach employed to identify recurring issues and derive knowledge hypotheses and develop the supplementary information required to address them. Finally, we detail the method used to embed this curated contextual knowledge within the frameworks.

4.1 Motivation

Recent advances in Large Language Models (LLMs) have demonstrated their potential to generate plausible clinical process models directly from Clinical Practice Guideline (CPG) texts [Lin et al., 2025, Kourani et al., 2024]. Moreover, as highlighted in the literature (see Section 2.3.1), integrating external sources of knowledge, such as Knowledge Graphs and additional contextual information, can significantly enhance the accuracy and reliability of LLM-generated outputs. This is particularly important in the clinical domain, where subtle workflow conventions and domain-specific practices are often not explicitly stated in the guidelines themselves.

For example, Grathwol et al. [2025] investigated the extraction of Clinical Pathways (CPWs) from CPGs using LLMs and found that ambiguities and inconsistencies within

the guidelines present substantial challenges for automated extraction. Their findings underscore the need for further research into methods that can mitigate these issues and improve the automation of clinical process extraction.

Our own experiments with the LLM-based pipeline, as detailed in Section 6.1.1, further reinforce this need. During expert evaluation, the clinical expert repeatedly emphasized the importance of providing the LLM with additional contextual information. This is because CPGs are typically written for expert audiences and often omit "common-knowledge" details that are nevertheless essential for accurate process modeling. For instance, without explicit instruction, the LLM frequently misplaced "cross-continuum secondary prevention assessments and therapies", positioning them in the middle of the pathway rather than as the final wrap-up activity. Similarly, composite Airway, Breathing, and Circulation (ABC) testing was often split into three separate steps, contrary to standard clinical practice.

These observations motivate the work presented in this chapter. Rather than presupposing "knowledge gaps", we treat the observed problems as *recurring issues* in LLM-generated clinical process models and formulate the following hypothesis: a substantial portion of these issues arise because clinical practice guidelines omit implicit contextual knowledge that LLMs do not reliably infer (while acknowledging that some errors may reflect foundational model limitations). To make this testable, we:

1. Introduce a categorization of recurring issues;
2. Describe a structured expert-analysis protocol that surfaces them and derives candidate contextual statements; and
3. Present integration strategies that inject this context at different points in the modeling pipeline.

This design operationalizes the hypothesis—if explicit context is the missing factor, issue rates should decline; if not, residual errors indicate model limitations. Chapter 6 evaluates this hypothesis and quantifies the impact on the automation and reliability of clinical process extraction from textual guidelines.

4.2 Identifying Recurring Issues and Deriving Knowledge Hypotheses

4.2.1 Expert Analysis Protocol

The evaluation of LLM-generated process models was conducted through a structured expert-analysis protocol designed to systematically identify and categorize recurring issues. The process mining expert (the thesis author) began by selecting a representative sample of process models generated by the LLM pipeline for each QBP variant. The sample was created from the process models scoring the highest node and structural similarity on the systematic evaluation tool. For each sample, the expert performed a detailed comparison against reference models that had been previously constructed and validated in collaboration with a clinical domain expert. These reference models incorporated both established clinical guidelines and explicit contextual knowledge, ensuring they reflected best practices and real-world workflow requirements.

During the comparison, the process mining expert examined each generated model to identify any discrepancies, omissions, or errors. This assessment was informed not only by the reference models but also by direct feedback from the clinical expert, who provided insights into domain-specific conventions, contextual dependencies, and the appropriate sequencing and grouping of clinical activities.

All identified issues were documented, including a description of the problem, its location within the process model, and any relevant contextual factors. To ensure a robust analysis, the expert tracked the frequency of each issue across multiple samples, allowing for the identification of patterns and the recognition of common, recurring problems. Issues that appeared repeatedly were flagged as systematic gaps rather than isolated errors.

Once the review was complete, the expert grouped similar recurrent issues together and developed a categorization according to their nature.

Rationale for prompt-based knowledge integration. As outlined in Section 2.3.2, LLM–knowledge integration can be done at pre-training, during training, or via post-

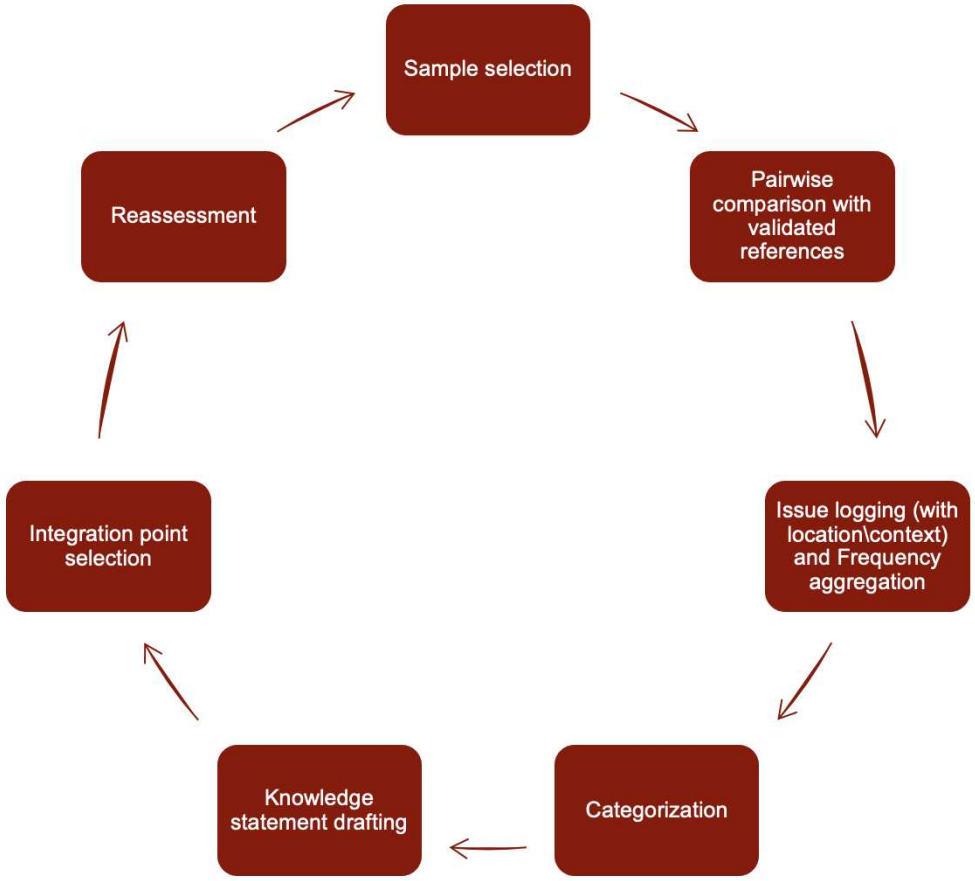


Figure 4.1: Structured protocol used to surface systematic gaps and derive contextual statements.

training fine-tuning [Yang et al., 2024b]. In this work we deliberately adopt *prompt-time* (inference-time) injection for three main reasons. First, it is *practical and model-agnostic*: it requires no parameter updates, scales across foundation models, and incurs no training cost, while prior work shows that KG-augmented prompting can reduce hallucinations and improve factual accuracy without retraining [Sequeda et al., 2024, Su et al., 2025, Pan et al., 2024, Yang et al., 2024b]. Second, it affords *experimental control*: by keeping model weights fixed, we can ablate *what* context is injected, *where* it is injected (generation/review/Knowledge Advisor), and *how* it is phrased, isolating the effect of context on errors. Third, it improves *interpretability and auditability*: explicit statements and the Knowledge Advisor's one-change-per-turn contract yield traceable edits suitable for clinical review.

This scope choice does not preclude other integration modes; rather, it provides a controlled setting to test our hypothesis that explicit context reduces recurring issues (Chap-

ter 6).

Crafting Contextual Knowledge for Integration: For each category of issues determined to stem from a lack of contextual knowledge, tailored knowledge statements were formulated to address these gaps. Initially, four issues were identified, which, through iterative revisions and reassessment, expanded to a total of sixteen distinct common issues, each paired with a corresponding contextual knowledge statement for integration into the framework.

All integration points and strategies were identified, including the steps in the MAO framework where contextual knowledge can be inserted. In addition, a new agent—the Knowledge Advisor—and an accompanying phase dedicated to knowledge integration were developed. This enables us to assess all prompt-template integration strategies possible within the MAO framework. After each iteration, we reassessed the outputs via manual evaluation to select appropriate strategies for the next cycle.

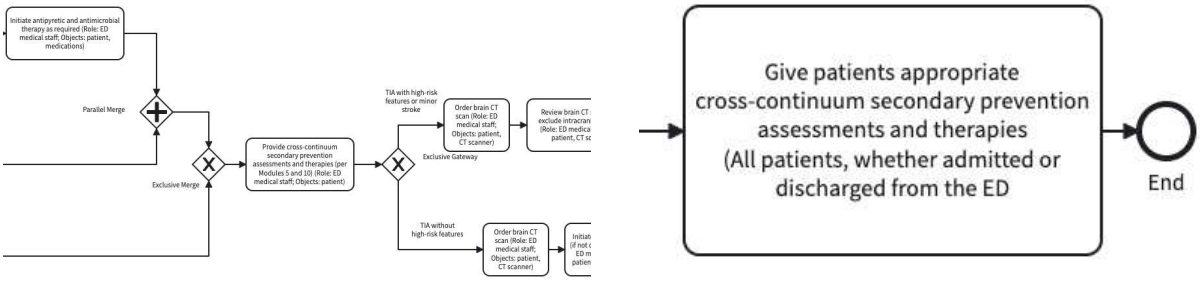
4.2.2 Categorization of Recurring Issues

In our review of LLM-generated process models, several recurring types of errors were identified. These issues were categorized into three main types: temporal relations, task granularity, and task options. Below is a detailed description of each category, along with examples from the stroke clinical pathway (QBP):

1. **Temporal Relations**

    - Temporal relation errors occur when activities are placed in an incorrect sequence.
    - *Example 1:* In the stroke care pathway, the activity "Provide cross-continuum secondary prevention assessments and therapies" should always be the final task in the process. However, LLM-generated models often placed this activity mid-process or at an incorrect position within the sequence (see Figure 4.2).

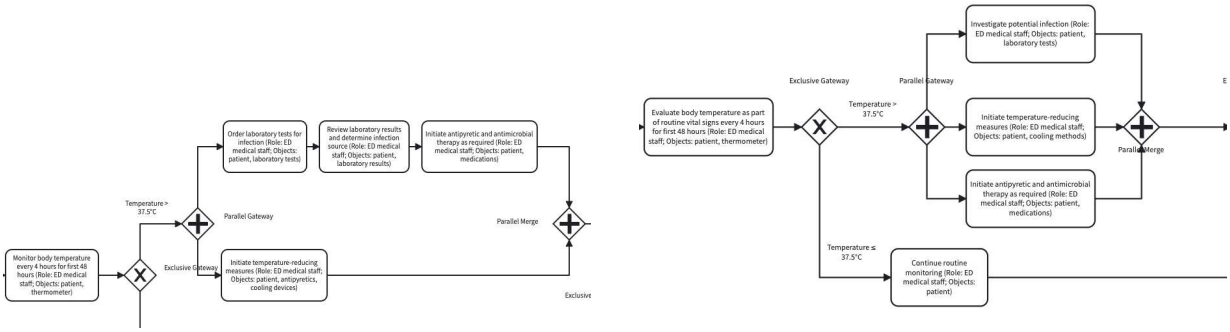
- *Example 2:* In QBP 2, temperature reduction measures should be executed in parallel to reduce time and optimize care. However, LLM-generated models often place these three activities sequentially, resulting in unnecessary delays (see Figure 4.3). Similarly, in QBP 1, the "neurological examination" activity should be performed in parallel with other assessments, but this requirement is often ignored by the LLM.



(a) Issue: misplaced terminal activity in mid-process.

(b) Corrected: terminal activity properly placed at the end.

Figure 4.2: Issue (left) versus corrected version (right) for the stroke pathway segment.



(a) Issue: Temperature reducing activities in sequence.

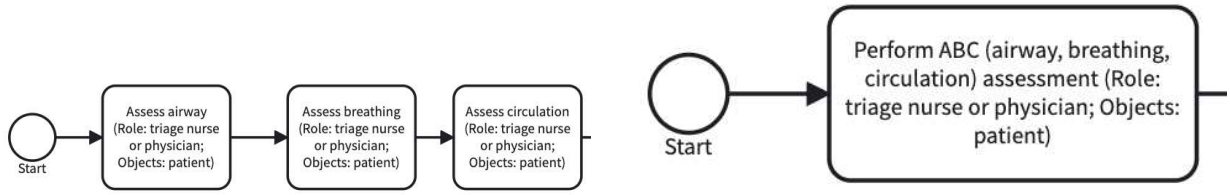
(b) Corrected: Temperature activities in parallel.

Figure 4.3: Issue (left) versus correct version (right) for the stroke pathway segment.

2. **Task Granularity**

- Overly granular tasks can complicate the process model by breaking down activities that should logically be treated as a single step.
- *Example:* The combined activity of Airway, Breathing, and Circulation (ABC) testing should be a single task in the clinical context. LLMs often split these

into three separate steps, leading to unnecessary complexity in the process model (see Figure 4.4).



(a) Issue: ABC testing activity divided to three activities.

(b) Corrected: ABC testing activity properly placed in one activity.

Figure 4.4: Issue (left) versus correct version (right) for the stroke pathway segment.

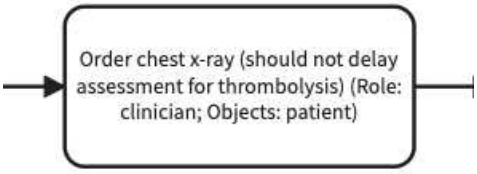
3. **Task Options**: Missing or misrepresented *choice* structures.

   (a) *Skip logic (optional tasks):* Skip logic refers to conditional tasks that are dependent on prior actions or clinical judgment. LLMs sometimes fail to incorporate such decision-making flows, leading to mandatory steps where optionality is needed.

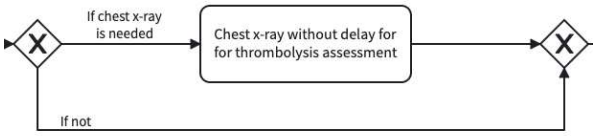
   Example: In QBP 1, the chest X-ray is an optional task that should be skipped unless explicitly indicated by prior clinical observations. However, the LLM often treats this as a mandatory step, which contradicts the clinical guidelines that suggest skipping it if not necessary (see Figure 4.5).

   (b) *Missing Options:* In some cases the LLM does not cover all the conditions and options required to perform an activity.

   Example: In QBP 3, the activity of early mobilization is conditional on not being contraindicated. Also, it is not recommended within the first 24 hours after stroke onset. However, the LLM often misses some of the cases that can happen and conditions (see Figure 4.6).

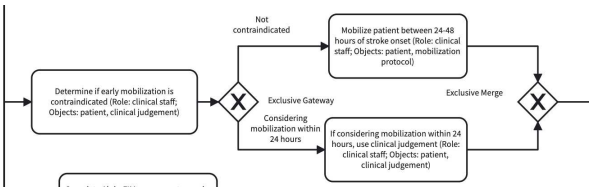


(a) Issue: Chest X-ray is represented as mandatory although it is not.

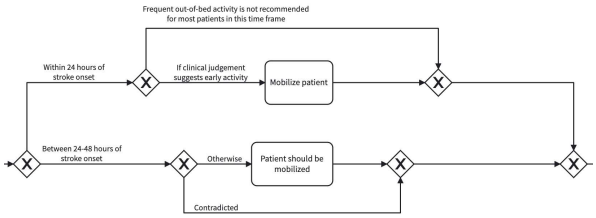


(b) Corrected: Skip flow for chest X-ray.

Figure 4.5: Issue (left) versus correct version (right) for the stroke pathway segment.



(a) Issue: Not covering all possible options and conditions for the activity.



(b) Corrected: Covering all options and conditions for the activity.

Figure 4.6: Issue (left) versus correct version (right) for the stroke pathway segment.

4.3 Knowledge Integration Strategies

In response to the identified knowledge gaps, we developed a series of knowledge integration strategies. These strategies involve injecting contextual knowledge into the LLM's process generation pipeline at various stages. By doing so, we aim to correct the errors observed in the LLM-generated models compared to the reference models.

We tested eight different versions of the MAO agent pipeline, each implementing a distinct knowledge integration strategy. The versions vary in terms of:

1. The timing of knowledge injection;
    - The knowledge statements can be injected at different stages and phases of the process.
2. The number of refinement cycles;

- The pipeline may perform k iterative revise-and-validate passes, where each pass assesses the current output against the knowledge and edits it to resolve detected gaps.

3. The verbosity of the knowledge statements.

   - We used 3 different knowledge templates: *Concise* set with 4 statements, *Expanded* with 16 statements, and *Explicit* with the same coverage as Expanded but with more explicit phrasing (see Table 4.2).

Table 4.1 provides an overview of each version considered in our experiments.

Version	**Injection timing**			**Cycles**	**Template**	**Order of phases**
	Gen	Review	KG - Advisor	(Rev/KA)	scope	
v3.0	•	✓	•	5/–	Concise	Gen → Review
v3.1	✓	•	•	5/–	Concise	Gen → Review
v3.2†	✓	•	•	5/–	Concise	Gen (role-config) → Review
v3.3	•	✓	•	10/–	Concise	Gen → Review
v3.4	•	•	✓	5/5	Concise	Gen → Review → KG-Adv
v3.5	•	✓	•	10/–	Expanded	Gen → Review
v3.6	•	•	✓	5/5	Expanded	Gen → Review → KG-Adv (post-review)
v3.7	•	•	✓	5/5	Expanded	Gen → KG-Adv (pre-review) → Review
v3.8	•	•	✓	5/5	Explicit	Gen → Review → KG-Adv (post-review)

Table 4.1: Design space explored across versions: timing, refinement budget, and statement design.

Legend: Gen and Review refer to the "Generation" and "Review" phases in the MAO framework; ✓ = knowledge injected at that stage; • = not injected. Cycles are reviewer/KG-Advisor suggestion loops; baseline reviewer cycles are 5 unless noted. †Injection at the Process Design Expert *role-configuration* (pre-generation) prompt.

The selection and progression of these integration strategies followed an iterative, ablation-driven methodology to efficiently explore the design space while isolating the effects of key variables such as integration timing, refinement cycles, and statement verbosity. Starting with the Concise template in versions v3.0–v3.4 allowed for baseline testing of core mechanics, revealing that configurations with extended template (e.g., v3.3) or a dedicated Knowledge Advisor phase (e.g., v3.4) showed promising reductions in recurring issues like temporal relation errors. Underperforming versions, such as those relying solely on early Generation-phase injection (v3.1–v3.2), were excluded from further escalation to avoid redundant experimentation, as preliminary expert evaluations indicated that additional

knowledge scope would not address their fundamental limitations. Advancing to the Expanded template in v3.5–v3.7 targeted broader domain coverage for residual gaps, with phase ordering ablated to optimize workflow (e.g., post-Review vs. pre-Review Advisor). Finally, the Explicit template was applied exclusively to the top performer (v3.6 as v3.8) to assess phrasing impacts without confounding factors. Knowledge integration was constrained to the Generation, Review, and Knowledge Advisor phases, as the Refinement and Testing phases serve distinct roles focused on decomposition and syntactic validation, respectively, rendering contextual integrations misaligned with their objectives.

4.3.1 Crafting Contextual Knowledge for Integration

For each category of issues, tailored knowledge statements were formulated to address them. Each issue that was determined by the expert that is stemmed from a lack of contextual knowledge was paired with a corresponding contextual knowledge statement, which was then integrated into the framework at the different injection points.

After some initial experiments with different types of knowledge statements and their verbosity, we curated *three templated sets* of contextual statements that the pipeline can inject at different stages. These templates differ along two axes: (i) *scope*—the number and breadth of domain rules included; and (ii) *verbosity*—the explicitness with which prerequisites, concurrency, and skip logic are articulated.

As an illustration, one representative statement that recurs across templates is:

> *All tests, including any type of imaging and blood tests, should be immediately interpreted by a healthcare expert.*

For completeness, the full, verbatim text of each template is provided in Appendix C (Concise: 4 items; Expanded: 16 items; Explicit: 16 items). Table 4.2 summarizes the different templates.

Table 4.2: Summary of contextual statement templates used in MAO variants.

Template	Scope	Coverage (illustrative)	Used in
Concise	4 items	Correct placement of secondary prevention assessment activities; prioritize institutional setup steps; immediate interpretation of tests; concurrency for slow/long tests.	v3.0–v3.4
Expanded	16 items	Adds domain-specific rules: ABC as one activity; neuro exam/imaging parallelism for EVT evaluation; merge lab panels; enumerate antiplatelet options; parallel temperature reduction; chest X-ray skip flow; ED evaluation in parallel; merge screening tasks; plan formulation precedence; mobilization timing.	v3.5–v3.7
Explicit	16 items	Same coverage as *Expanded* but with more explicit prerequisite and concurrency phrasing (e.g., marking dependencies, clarifying skip conditions).	v3.8

4.3.2 Knowledge Advisor Prompt Structure

The Knowledge Advisor phase prompt adopts the same slot-based structure used by the other MAO agents (e.g., *Process Design Expert*, *Process Design Reviewer*) and was derived from those agents' prompts to ensure interface consistency and ease of ablation across versions. This structure is constructed from different blocks (e.g., knowledge injection of format description and process constraints, few shot learning examples, etc.) and is dynamically filled during inference. Note: For versions without the dedicated Knowledge Advisor agent (v3.0–v3.3, v3.5), contextual knowledge is integrated directly into existing MAO phase prompts following its default prompt structure.

Constituent blocks.

Roles

Declares assistant/user roles (e.g., *Knowledge Graph Advisor* / *Team Leader*) to bind responsibilities.

Task context

Natural-language summary of the clinical module or guideline segment under review (prob-

lem framing).

Reviewed process

The current process model text to be critiqued, passed verbatim as `{reviewed_process_text}`.

Extra knowledge

A selected template of contextual statements (Concise/Expanded/Explicit) that encode domain rules the suggestion must respect.

Format schema

A compact specification of allowable tags/structure, provided via `{format}`, to keep suggestions structurally valid.

Output contract

A fixed two-field response—`Suggested modification: ...` and `Reason: ...`—with a strict *one-suggestion-per-turn* constraint and a canonical stop phrase when no further issues remain.

Prompt skeleton (schematic). The assembly order and response shape mirror the other MAO agents, and is visualized in Figure 4.7. Note that placeholders are shown to indicate variables, not literals.

For transparency and reproducibility, the complete, *verbatim* prompt of our example—including the clinical task text, the reviewed process example, the selected contextual statements, the formatting schema, and the output contract—is reproduced in Appendix C.2.

4.4 Chapter Summary

This chapter established the rationale and methods for incorporating contextual knowledge into an LLM-based pipeline for guideline-to-process modeling. The main points are:

- **Motivation and Problem Framing.** We argued that Clinical Practice Guidelines (CPGs) often omit implicit conventions that generic foundational LLMs do not

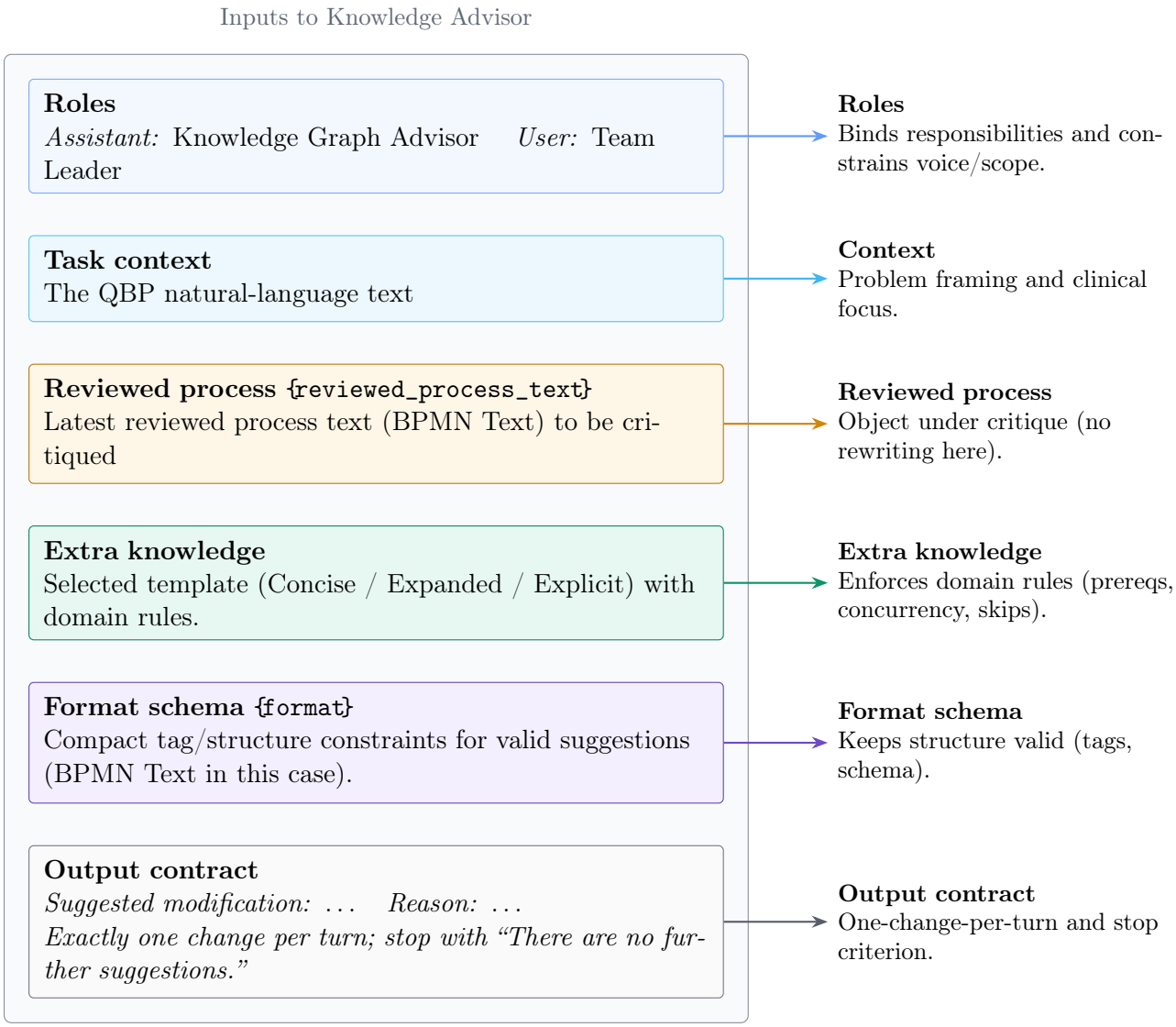


Figure 4.7: Prompt structure used by the Knowledge Advisor for consistent, one-change-per-turn suggestions.

reliably infer. Typical failure modes include: misplacing terminal activities (e.g., "cross-continuum secondary prevention"), over-splitting composite assessments (e.g., ABC testing), missing choices (e.g., options or skip logic), missing required follow-ups (e.g., interpretation after imaging), and failing to model concurrency for time-sensitive options (e.g., temperature reduction). These issues motivate explicit provision of contextual knowledge to the pipeline.

- **Expert Analysis Protocol.** A process-mining expert (the thesis author), informed by a clinical specialist, reviewed generated models against validated reference models,

documented discrepancies, tracked recurrence of issues, and mapped them to the categorization. This procedure yielded concrete targets for knowledge injection while maintaining clinical relevance.

- **Recurring issues and hypothesis.** We group issues into temporal relations, task granularity, and task options. We hypothesize that injecting explicit contextual knowledge reduces these issues. Chapter 6 tests that hypothesis.

- **Integration Strategies (MAO v3.0–v3.8).** We designed eight variants (Table 4.1) that vary along:

  - *Injection timing*
  - *Number of refinement cycles*
  - *Knowledge statement verbosity*

- **Knowledge Statement Templates.** To address the knowledge gaps, we curated three template sets (Table 4.2) that differ in both *scope* and *verbosity* namely "Concise", "Expanded", and "Explicit".

- **Methodological Contributions.** This chapter delivers:

  - A principled rationale for making implicit clinical conventions explicit during model generation;
  - A rigorous, expert-driven procedure for detecting and categorizing LLM modeling issues;
  - A structured suite of strategies for knowledge injection;
  - Clear documentation of the statement templates and the Knowledge Advisor prompt structure to support reproducibility.

- **Scope and Forward Path.** This chapter does *not* report experimental outcomes. Quantitative and qualitative evaluations of the integration variants, using expert Δ-scores, node- and structure-similarity metrics, and statistical analyses, are presented in Chapter 6. But first, the experimental setup is introduced in the Chapter 5.

Chapter 5

Experimental Setup

This chapter specifies our experimental setup across two distinct experiments. **Experiment 1 (Baseline)** is a head-to-head comparison of *published* ProMoAI and MAO frameworks—run under a common configuration and execution protocol—to attribute differences to framework design rather than tuning (Sect. 5.2). **Experiment 2 (Knowledge-enhanced)** builds on the stronger baseline framework by integrating domain knowledge into MAO via several variants, and compares these variants against the baseline MAO version. We begin by describing the dataset—four stroke-related QBPs from Health Quality Ontario—and input preprocessing (Sect. 5.1); we then detail Experiment 1 (configuration, protocol, and dual evaluation methods), followed by the definition and evaluation of the knowledge-enhanced MAO variants (Experiment 2). Throughout, we emphasize reproducibility so that the results reported later can be audited and independently replicated.

5.1 Evaluation Dataset

Our evaluation relies on stroke-related Quality-Based Procedures (QBPs) from Health Quality Ontario's public repository [Health Quality Ontario and Ministry of Health and Long-Term Care, 2016]. QBPs are part of Ontario's Quality-Based Funding initiative under the Excellent Care for All Act, which mandates evidence-based recommendations to standardize care and align funding with quality and efficiency [Health Quality Ontario

and Ministry of Health and Long-Term Care, 2016].

We selected the latest version of the Clinical Handbook for Stroke (Acute and Postacute), from December 2016. We use four of its modules as evaluation artifacts, referred to as "Input Guideline (IG)" from now on, each spanning approximately 2-3 pages. For each input guideline, we removed text related to acknowledgements and references, which we consider noise for our purposes.

These QBPs in this clinical handbook are particularly relevant to regional projects on compliance checking in Ontario, such as those at Montfort Hospital, which focus on aligning care pathways (CPWs) with care trajectories. Also, we found the QBPs to be clear, well defined, and aligned with the goals of standardizing stroke care across regions, making them a suitable basis for this assessment.

5.2 Baseline Evaluation

5.2.1 Framework Configuration and Execution Protocol

Goal and scope. We compare two state-of-the-art extraction frameworks (ProMoAI and MAO [Lin et al., 2025, Kourani et al., 2024]) using a common configuration and execution protocol, described in this section, to ensure that differences in outcomes reflect *framework design* rather than tooling or tuning.

Framework execution. We use our **LLM4CPW** environment to run both frameworks, using the same underlying foundational model (GPT-4.1), invoked through Module I (LLM Execution), and funneled through identical downstream standardization and evaluation modules (see Chapter 3 and Section 1.4).

Framework prompts. ProMoAI and MAO follow the same overall prompt structure; each consumes a framework-specific, template-based prompt with few-shot exemplars. We adapted these templates to signal that the task is framed as *Clinical Pathways (CPWs)*,

not generic business processes, while preserving each framework's expected output format (e.g., BPMN-Text or POWL script). This keeps the interaction faithful to each framework's design while aligning the task domain with our dataset (input guidelines).

Decoding settings. Following common practice for controlled LLM studies in process–model extraction [Kourani et al., 2024, Lin et al., 2025], we fix decoding hyperparameters across all runs to prioritize fidelity and stability: `temperature`=0.2, `top_p`=1.0, `max_tokens`=2000–2048, `frequency_penalty`=0, `presence_penalty`=0. A low `temperature` paired with `top_p`=1.0 (i.e., no nucleus filtering) reduces sampling variability and makes temperature the sole stochastic control; `frequency`/`presence` penalties are set to zero to avoid unnecessary rewording of activity labels that would confound string-sensitive evaluation. The `max_tokens` budget prevents truncation of control-flow fragments while respecting model limits. Holding this suite of settings fixed equalizes decoding conditions across frameworks, so observed differences are attributable to framework logic (e.g., MAO's multi-agent review vs. ProMoAI's self-refinement) rather than sampling artefacts.

Parameter	Value	Rationale	Expected effect
`temperature`	0.2	Minimize sampling randomness so observed differences reflect framework logic rather than noise.	Low run-to-run variance; stable structure and labels.
`top_p`	1.0	Disable nucleus filtering so temperature solely governs stochasticity.	Avoids interaction effects between sampling controls.
`max_tokens`	2000–2048	Prevent truncation of control-flow fragments while respecting model limits.	Completeness of outputs; fewer incomplete models.
`frequency_penalty`	0.0	Do not penalize necessary lexical reuse (clinical terms, activity names).	Preserves canonical labels for string-sensitive metrics.
`presence_penalty`	0.0	Avoid encouraging novelty that could drift from the prompt/few-shots.	Tighter adherence to domain terminology and intent.

Table 5.1: Fixed decoding hyperparameters held constant across frameworks (ProMoAI and MAO) unless otherwise noted, to isolate framework effects.

Robustness and failure handling. Errors in the output format trigger automatic exponential-backoff retries (up to three attempts). After exhausting retries, the run is marked as failed; its metadata and logs are persisted for auditability, and the instance is excluded from quantitative analysis.

Extraction runs and variability. Each extraction is an independent run. In our experiment, for variance estimation and statistical testing, we execute $n > 1$ identical runs (same configuration, independent calls). Note that the web UI exposes a "number of runs" control (item ⑤ in Figure 3.11) and records the results in the results database to enable batch runs.

Outputs and audibility. For every run, the pipeline persists: (i) the filled prompt, (ii) the raw framework output, (iii) standardized BPMN 2.0 (auto-layout), (iv) label-aligned BPMN used for automated metrics, (v) structured logs and retry events, and (vi) a database row with identifiers (framework, model, run name) and metric scores. Versioned YAML configuration files and prompt templates are checked into the repository alongside the code [Houshidari, 2025].

Computing environment. The stack is containerized (Docker[1]) for consistency across hosts. Experiments were executed on a MacBook Pro (2020, Intel i5, 8 GB RAM) without GPU acceleration. Key versions: Python 3.12/3.9, PM4Py 2.7.12.4, OpenAI SDK 1.79.0/-1.63.0. Container digests, image tags, and package locks are available through the shared repository [Houshidari, 2025].

Protocol (step-by-step).

P1. Select configuration. Choose framework (MAO or ProMoAI), base model (GPT-4.1), prompt template preset, and number of runs (n). Provide the CPW text (input guideline excerpt) and optional reference BPMN (Section 3.5).

[1] https://www.docker.com/

P2. Prompt construction. Inject the CPW text into the framework's prompt skeleton (few-shot included), with clinical-pathway task framing (Section 3.2).

P3. Dispatch and monitoring. Send the request via the framework adapter. A framework-agnostic listener records structured events (request sent, partial response, retry attempt, final response) with timestamps and paths.

P4. Retry on failure. On errors, perform exponential-backoff retries (max 3). After final failure, persist logs and mark the instance as failed (excluded from metrics) (Section 3.2).

P5. Output standardization. Convert the framework's intermediate representation (e.g., BPMN-Text or POWL) to standard BPMN 2.0 with auto-layout (Section 3.3).

P6. (Metrics-only) label alignment. Harmonize generated activity labels to the reference terminology to enable fair, string-sensitive comparisons; retain the unaligned BPMN as the model actually produced (Section 3.4).

P7. Evaluation and export. Compute node/structure similarity and GED against the reference BPMN; persist scores and artifacts; surface results in the UI for inspection and download (Section 3.4).

P8. Repeat for n runs with identical settings to compute mean performance and run-to-run variability.

5.2.2 Evaluation Methods

We follow two major types of evaluations from the process extraction literature [Van Woensel and Motie, 2024]:

- **Expert Evaluation:** A physician and healthcare executive from Montfort Hospital (and publication co-author), with expertise in clinical process management and prior experience with BPMN, conducted a comprehensive review of a sample of the standardized (BPMN) LLM output (from P5 in the execution protocol)[2]. For

[2]We omitted the activity alignment step so as to review the original LLM outputs.

each process, the expert: (a) scored the process's fidelity to the corresponding input guideline on a 5-point Likert scale, and (b) provided free-text comments on missing activities, incorrect branch labels, sequencing errors, and unclear layouts. The expert-evaluation handout (instructions and scoring forms) used in the baseline experiment is provided in Appendix D. We report the mean of the overall ratings and a qualitative summary of the expert's observations.

- **Systematic (Graph-Based) Evaluation:** Using our pipeline's *Automated Evaluation* component (Section 3.4), we quantitatively compared each standardized LLM output (from P6 in the execution protocol) against an expert-validated reference process. Each guideline underwent 15 pipeline runs, and we report the mean results in this thesis. To evaluate performance differences between MAO and ProMoAI, we applied Welch's t-test[3] (as it is robust to unequal variances) and calculated Cohen's d[4] to quantify effect sizes.

5.3 Knowledge-enhanced Evaluation

5.3.1 Framework Configuration and Execution Protocol

Based on our baseline evaluation, MAO offered a superior performance over ProMoAI (Section 6.1). Hence, we chose MAO for the knowledge-enhanced experiment. We applied different knowledge-enhancement approaches (Section 4.3), which led to different "knowledge-enhanced" MAO framework versions. Then, we compared the best-performing knowledge-enhanced framework version with the baseline version (as described in Sect. 5.2). All other configurations and hyperparameters remained the same as for the baseline evaluation (Sect. 5.2.1). An analogous execution protocol is applied as well.

Here is how the ten versions used in the experiment differ from one another (Table 4.1 provides a high-level overview of each version):

[3] https://en.wikipedia.org/wiki/Welch%27s_t-test
[4] https://en.wikipedia.org/wiki/Effect_size#Cohen's_d

- V2.2: The MAO version used for the baseline evaluation (Section 5.2).
- V3.0: Knowledge statements added to the Process Design Reviewer agent's prompt in the Give Suggestion phase without increasing the maximum cycle number.
- V3.1: Knowledge statements added to the Process Design Expert agent's prompt in the Process Generation phase.
- V3.2: Knowledge statements added to the Process Design Expert agent's Role Config Prompt.
- V3.3: Same as V3.0, but the Process Design Reviewer's Give Suggestion cycle budget is doubled (10 cycles).
- V3.4: Dedicated Knowledge Advisor agent and knowledge integration phase (5 cycles) placed after the review phase.
- V3.5: Knowledge statements added to the Process Design Reviewer agent's prompt in the Give Suggestion phase with doubling the maximum cycle number.
- V3.6: Dedicated Knowledge Advisor agent and knowledge integration phase placed after the review phase. The difference with V3.4 is the knowledge template used (concise vs expanded).
- V3.7: Dedicated Knowledge Advisor agent and knowledge injection phase placed before the review phase.
- V3.8: Same as 3.6 but with more explicit knowledge statements.

5.3.2 Evaluation Methods

Similar to the evaluation conducted in the Baseline Experiment, we follow two primary types of evaluation:

- **Expert Evaluation:** A process model specialist, who is also the author of this thesis, conducted a thorough review of the standardized (BPMN) LLM output (from P5

in the execution protocol). To identify the most effective integration strategy within the study's time constraints, the expert provided free-text feedback on the overall performance of representative samples from versions v3.0–v3.7, rather than producing a fully documented evaluation for each run. In the subsequent step, the expert conducted a structured comparison of the best-performing knowledge-enhanced versions against the baseline's issue list, recording only (a) baseline issues that were resolved and (b) new issues introduced by the KG-enhanced model. These observations were then coded and aggregated into a net change score, Δ (details in the following paragraph). Finally, to assess the impact of wording and structural refinements in the knowledge statements, the same issue-centred evaluation and coding procedure was applied to compare the best knowledge-enhanced versions against MAO v3.8, which incorporates the explicit knowledge statement set.

Issue-centred expert evaluation and coding For each Input Guideline (IG), the expert first compared the *baseline* MAO model to the reference process and compiled an *issue list* (e.g., missing activities, misordering, incorrect/ambiguous branch labels, unnecessary duplication, layout obscuring logic). The models generated from the knowledge-enhanced framework was then evaluated *relative to that baseline issue list*:

- **Positive (+1):** A baseline issue is resolved in the KG model.
- **Negative (–1):** A *new* issue appears in the KG model that was *not* present in baseline.
- **Neutral (0):** No change with respect to the baseline issue (still present / not improved), and no new issue introduced.

To reflect minor/partial effects used in our summaries, the expert also used half-steps: *Slight Positive* = +0.5 for partial or low-impact fixes; *Slight Negative* = –0.5 for minor regressions (e.g., wording degradation without logical effect).

Per IG and version, we report the *net change* as per Equation 5.1:

$$\Delta = \sum \text{codes} = (\#\text{resolved} + 0.5\,\#\text{partially resolved}) - (\#\text{new} + 0.5\,\#\text{minor new}) \quad (5.1)$$

This Δ metric captures the balance of resolved baseline issues versus newly introduced ones. This issue-centred evaluation complements the quantitative similarity metrics reported in Chapter 6.

- **Systematic (Graph-Based) Evaluation:** Using our *Automated Evaluation* module, we quantitatively compared each standardized LLM output process (from P6 in the execution protocol), generated by a knowledge-enhanced framework version, against an expert-validated reference process. For the later versions (V3.5–V3.8), each guideline underwent 15 pipeline runs (for higher precision in these high-performing versions), while for the earlier versions (V3.0–V3.4), we conducted only 5 runs per guideline. The mean results are reported. To assess the performance differences between the knowledge-enhanced and baseline versions, we applied Welch's t-test (which is robust to unequal variances) and calculated Cohen's d to quantify effect sizes.

5.4 Chapter Summary

This chapter established the experimental foundation for the thesis in two stages. **Experiment 1 (Baseline)** performs a controlled comparison of ProMoAI and MAO under a common configuration to attribute performance differences to *framework design* rather than tuning. **Experiment 2 (Knowledge-enhanced)** extends the stronger baseline (MAO) with domain knowledge through multiple variants and compares these against the unmodified MAO baseline.

We grounded all evaluations in four stroke-related quality-based procedures (input guidelines) from Health Quality Ontario's clinical handbook (Acute and Post-acute, 2016), after removing non-instructional content. Both frameworks are executed within the same

LLM4CPW pipeline using GPT-4.1, framework-specific prompts adapted to clinical pathways, and identical downstream standardization and evaluation modules. Decoding hyperparameters are fixed (`temperature`=0.2, `top_p`=1.0, `max_tokens`=2000–2048, no frequency/presence penalties) to minimize stochasticity and equalize conditions across frameworks. The stack was containerized, and all artifacts are persisted with versioned configuration to ensure audibility.

Two complementary evaluation methods are applied. First, an expert qualitatively assesses representative BPMN outputs for guideline fidelity (5-point Likert) and provides diagnostic comments. Second, a systematic graph-based evaluation compares standardized models against expert-validated references using node/structure similarity and GED, with n independent runs per guideline to summarize mean performance and variability (15 runs). Group differences are tested with Welch's t-test and effect sizes summarized via Cohen's d.

For the knowledge-enhanced study, MAO is augmented via a family of variants that injects curated knowledge at different agents/phases (e.g., reviewer prompts, expert role configuration, or via a dedicated Knowledge Advisor), sometimes with extended review/integration cycles. Early versions are run five times per guideline; later versions are run fifteen times. All other settings and the execution protocol mirror the baseline to permit fair comparison.

Together, these design choices provide a controlled, transparent, and fully auditable setup for the results reported in the next chapter, enabling independent replication and targeted analysis of where (and why) knowledge integration improves clinical pathway extraction.

Chapter 6

Results

In this chapter, we show the results of our empirical evaluation of MAO and ProMoAI, and the knowledge-enhanced versions of MAO, for automated process extraction from clinical guidelines. We use our previously introduced LLM4CPW pipeline (Chapter 3) to evaluate these 3 systems. We first benchmark MAO against the ProMoAI in the Baseline Evaluation Experiment (Section 6.1) using expert ratings and systematic evaluation using quantitative similarity metrics. Next, we evaluate the impact of knowledge-enhancement strategies—from prompt modifications in already existing phases in MAO to a dedicated *Knowledge Advisor* agent—on outcomes, similarly using expert ratings and systematic evaluation (Section 6.2). We compare Knowledge-enhanced pipeline against the Baseline, and further assess how precise wording refinements affect outcomes. Finally, Section 6.3 will summarize the findings. An at-a-glance summary of the systematic evaluation results appears in Figures 6.9, 6.10, and 6.11. Figure 6.4 provides a high-level overview of each Knowledge-enhanced MAO version.

6.1 Baseline Evaluation Results

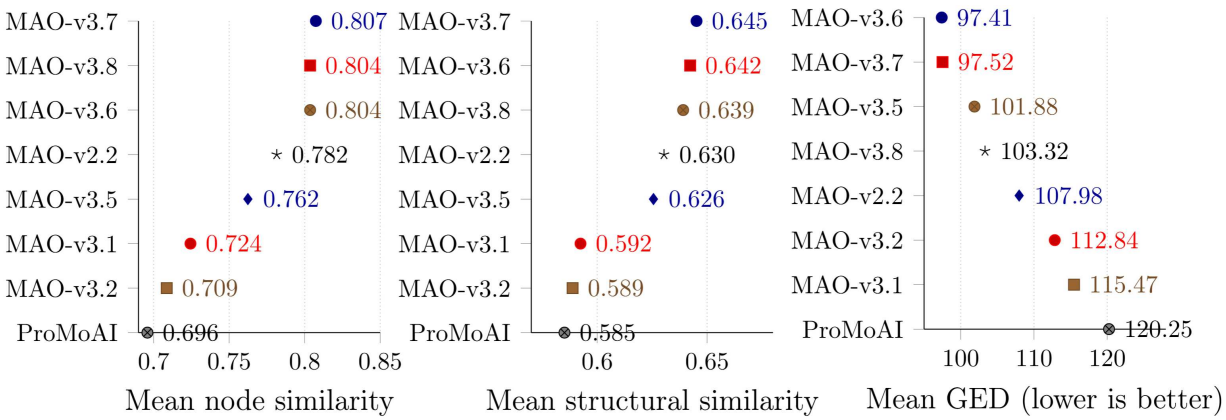
The following tables and figure, which involve our four Input Guidelines (IGs), i.e., the selected stroke-related QBPs, will be referenced and explained in the next subsections.

Table 6.1: Expert Ratings of LLM-Generated Process Models in the Baseline Experiment

Model	Mean Score	Highest IG Rating	Lowest IG Rating
MAO	**4.25**	5.0 (IG 2, IG 4)	3.0 (IG 3)
ProMoAI	4.00	5.0 (IG 2)	3.0 (IG 4)

Table 6.2: Baseline Experiment's Systematic Evaluation Statistics.

Statistic	Node Similarity	Structure Similarity
Mean (MAO)	**0.782**	**0.630**
Mean (ProMoAI)	0.696	0.585
CV (MAO)	**0.058**	**0.046**
CV (ProMoAI)	0.158	0.090
t (df)	5.29 (91)	5.16 (113)
p	<0.001	<0.001
Cohen's d	0.97	0.94



(a) Node similarity **(b)** Structural similarity **(c)** Graph-edit distance

Figure 6.1: Overall performance across frameworks based on the systematic evaluation results (means over all IGs).

6.1.1 Expert Evaluation

As shown in Table 6.1, the healthcare expert rated the MAO-generated models slightly higher. Among the evaluated IGs, IG2 achieved the highest rating at 5/5, while IG3 received the lowest at 3.5/5. In their feedback, the expert noted that, although the LLM-generated models generally captured the essential clinical tasks, their accuracy was limited

by the lack of contextual and background information within the input guidelines. Clinical guidelines are typically intended for expert audiences and often omit details considered common knowledge among practitioners.

To increase the performance of the pipeline, the expert highlighted the importance of providing the LLM with additional contextual information, such as institutional constraints and standard emergency department workflows.

Observation 6.1: Expert ratings favor MAO but point out missing context

MAO received a slightly higher mean expert score than ProMoAI (4.25 vs. 4.00; Table 6.1). The reviewer attributed most residual errors to tacit clinical context omitted from guidelines (e.g., institutional constraints, ED intake routines). This indicates that performance could be improved by supplying background knowledge alongside the source text.

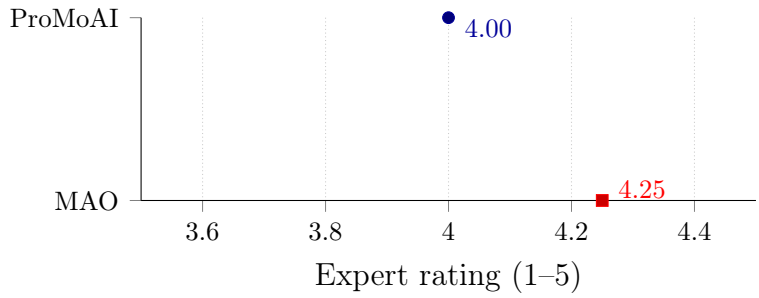


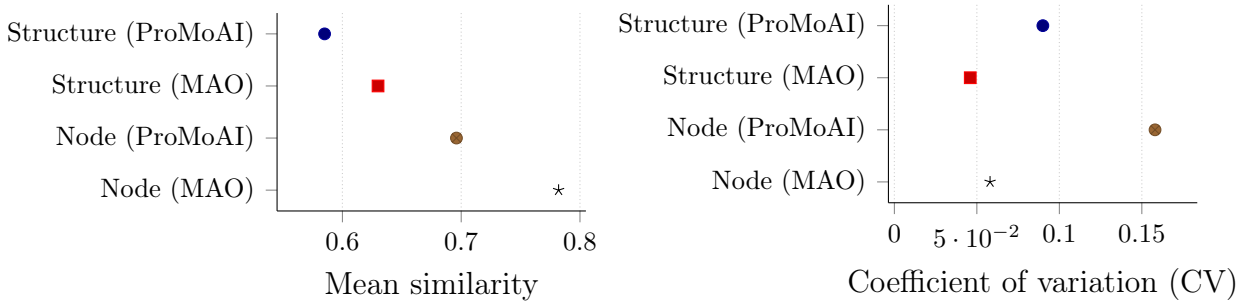
Figure 6.2: Expert mean ratings by model (from Table 6.1).

6.1.2 Systematic Similarity Analysis

As shown in Table 6.2, MAO demonstrated better performance, achieving the highest average node-level similarity (0.78) and structural similarity (0.63). ProMoAI exhibited greater variability in performance, with the highest run-to-run variability in node similarity (Coefficient of Variation (CV) = 0.158). We note that the two similarity measures also showed very strong correlation (r = 0.89; not in the table), indicating that LLM frameworks performing well on node-level matching also tend to excel at structural alignment.

Statistical analysis confirms that MAO's superior performance is statistically significant. Welch's t-tests revealed significant differences for both metrics: node similarity

(MAO: 0.781 vs. ProMoAI: 0.696; $t(\approx 91) = 5.294$, $p < 0.001$, Cohen's $d = 0.967$) and structural similarity (MAO: 0.630 vs. ProMoAI: 0.585; $t(\approx 113) = 5.159$, $p < 0.001$, Cohen's $d = 0.942$). Both comparisons yielded large effect sizes and p-values well below the 0.05 significance threshold. Hence, we observe that MAO has significantly superior performance in automated process extraction from clinical guidelines. Figure 6.3 provides an overall view of the mean similarity and coefficient of variation results. Also, per-IG results can be seen in Figure 6.9 and 6.10.



(a) Means (Table 6.2).

(b) Run-to-run variability (CV).

Figure 6.3: MAO vs ProMoAI: means and variability.

Observation 6.2: MAO is both accurate and more stable than ProMoAI

MAO significantly outperformed ProMoAI on node similarity (0.782 vs. 0.696) and structural similarity (0.630 vs. 0.585) with large effects ($d \approx 0.95$) and $p < 0.001$ for both (Table 6.2). Run-to-run variability was markedly lower for MAO ($\mathrm{CV}_{\mathrm{node}} = 0.058$ vs. 0.158; $\mathrm{CV}_{\mathrm{structure}} = 0.046$ vs. 0.090). This indicates that an agentic workflow, as employed by MAO, could be more suitable for automated process extraction.

6.1.3 Discussion

Dominant error source: lack of contextual knowledge. Differences between the generated and the reference process models frequently arose where the guideline text presupposed clinical routines (e.g., ED intake pathways) or institutional constraints. In several cases, the "missing" information was not present in the IG text upon re-check, prompting plausible but incorrect LLM inferences. This pattern supports the design choice of integrating contextual knowledge (Chapter 4).

As stated in Chapter 4, in this research, "Knowledge Enhancement" means adding explicit, curated clinical context to the LLM at inference time so it can model guidelines more faithfully. The workflow is: (1) use expert review to spot recurring errors in LLM-generated process models; (2) turn those into concise domain rules ("contextual statements"); and (3) inject them into prompts at different stages while varying cycles and statement verbosity. The hypothesis here is: *if implicit clinical conventions are made explicit, these recurring errors will drop; if not, remaining mistakes reflect model limits.*

We observe that MAO's agent-based workflow—where dedicated agents take turns drafting, critiquing, and polishing segments of the process—leads to higher similarity to the reference IGs and models, according to both the healthcare expert and systematic evaluation. By delegating responsibilities such as generation, refinement, error checking, and structure validation to distinct LLM agents, we found that MAO reduces errors, and yields process models that experts find more accurate.

6.2 Knowledge-enhanced Pipeline

Section roadmap. This section has two complementary subsections, each with three subparts. Additionally, a discussion subsection synthesizes the findings from both evaluation methods at the end.

(I) Expert Evaluation (manual, qualitative). (§6.2.1)

a) **Integration Strategies:** design choices and advisor placement for v3.0–v3.7 (Table 6.3 and Figure 6.4).

b) **Knowledge-enhanced vs Baseline:** per-IG net expert Δ comparing KG variants to MAO v2.2 (Table 6.4, Fig. 6.5).

c) **Wording and Structure of Knowledge Statements:** expert critique of statement precision and remaining ambiguities (e.g., v3.6–v3.8) (Table 6.5 and Figure 6.6).

(II) Systematic Similarity Analysis (quantitative, metric-based). (§6.2.2)

a) **Integration Strategies:** node/structural similarity and GED for v3.0–v3.7 (Table 6.6, Fig. 6.7).

b) **Knowledge-enhanced vs Baseline:** head-to-head comparisons of v2.2, v3.6, and v3.7 (Table 6.7, Fig. 6.8).

c) **Wording and Structure of Knowledge Statements:** metric effects of wording refinements (v3.6 vs. v3.8; Table 6.8).

(III) Discussion.(§6.2.3)

a) **Synthesizing insights across the two evaluation methods.**

6.2.1 Expert Evaluation

Integration Strategies

This subsection reports *manual expert* results by version; design/placement is shown in Fig. 6.4, and quantitative metrics appear in Section 6.2.2.

Results at a glance.

- **v3.0–v3.2 (prompt-only):** The model *used* the injected knowledge but *regressed* on previously strong behaviors; net effect was mixed. This points to the need for a *dedicated knowledge role*.
- **v3.5 (more reviewer cycles):** Manual assessment showed *higher accuracy and compliance* with the added knowledge, but *run-to-run stability* remained an issue.
- **v3.6 (specialized advisor post-review):** *Largest and most stable* gains observed; minimal regressions. (v3.7, advisor pre-review, also improved over baseline but was slightly less consistent.)

Version-wise outcome summary (manual expert).

Table 6.3: Version-wise outcome summary (manual expert).

Version	Injection point	Cycles (Rev/KA)	Outcome (manual expert)
v3.0	None in Gen/Ref/Rev	5/–	Mixed: knowledge picked up in places, but fidelity dropped elsewhere.
v3.1	Gen (pre)	5/–	Similar mixed pattern; no net improvement.
♦ v3.2†	Role-config (pre-generation)	5/–	Responds to new knowledge, but prior strengths regress; suggests need for a dedicated role.
v3.3	Rev (post)	10/–	Inconsistent (concise knowledge set); not included in subsequent analysis.
v3.4	KG-Adv (post)	5/5	Inconsistent (concise knowledge set); not included in subsequent analysis.
v3.5	Same as v3.0; doubled reviewer cycles	10/–	Better accuracy/compliance with added knowledge; *not stable*.
v3.6	KG-Adv *after* review	5/5	**Much better & more stable** improvements; minimal regressions.
v3.7	KG-Adv *before* review	5/5	Clear gains vs. baseline; slightly less consistent than v3.6.

Note on omitted versions (v3.3 and v3.4): Versions 3.3 and 3.4 are not included in the subsequent analysis, as their results were found to be inconsistent and difficult to interpret. Manual expert assessment suggested that the primary cause was the limited number of knowledge statements available in these versions. When only one or two issues relevant to the lack of contextual knowledge were present, the remaining review cycles were effectively repurposed for general model improvements rather than targeted knowledge integration. To address this limitation, we re-ran these integration strategies in versions 3.5 and 3.6 using the expanded set of knowledge statements, which yielded more stable and interpretable results. Versions 3.0–3.2 also employed the same concise set of knowledge statements. However, given the specific integration points used in these versions, there is no evidence that the limited number of statements influenced their outcomes. Therefore, they were not re-run with the expanded set of knowledge statements.

Summary and Discussion: Across integration strategies, *manual expert* assessment shows that prompt-only variants (v3.0–v3.2) caused the model to pick up the injected knowledge yet regress on previously strong behaviors—consistent with increased prompt complexity against a fixed cycle budget that forced trade-offs. Increasing reviewer capacity (v3.5) improved accuracy and compliance with the added knowledge, but gains were not reliably stable across runs. The largest and most stable improvements arose when introducing a *dedicated Knowledge Advisor* agent; placement mattered: inserting the advisor

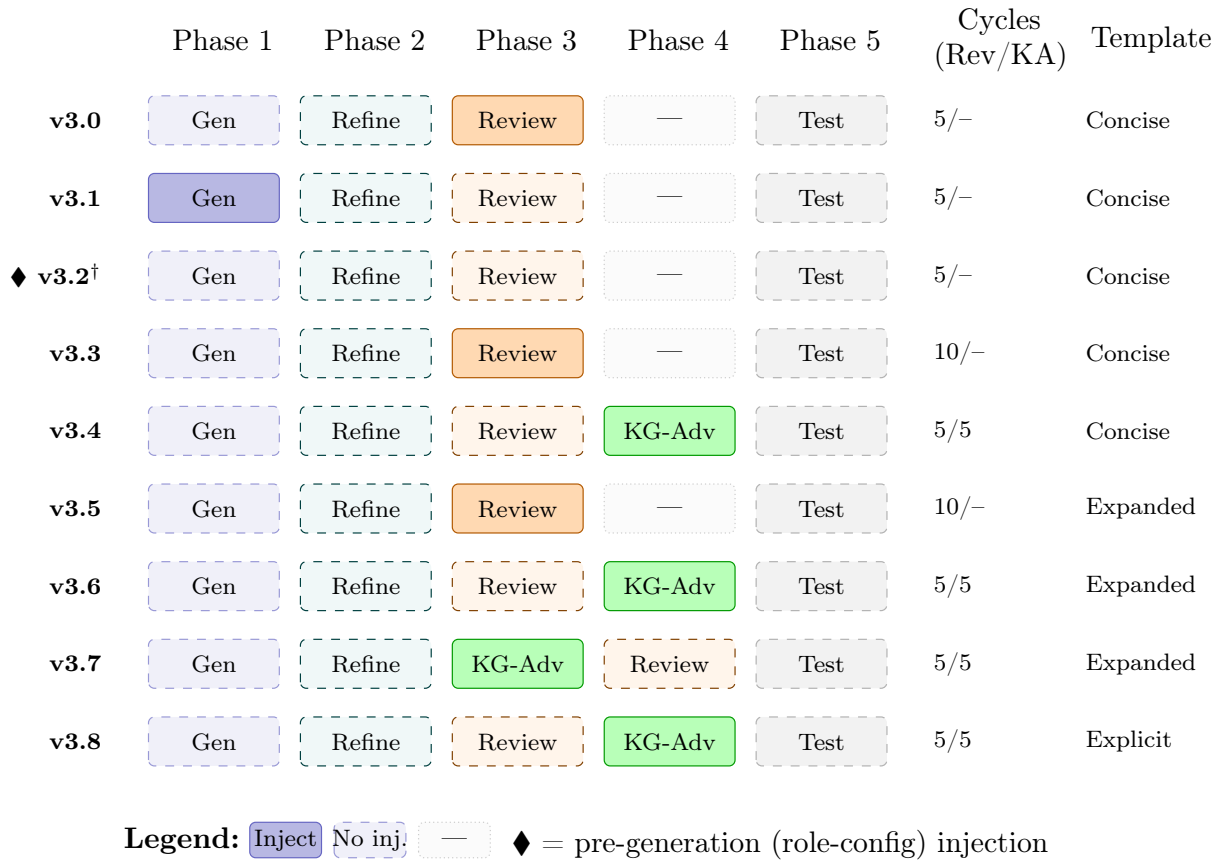


Figure 6.4: Phase-grid for MAO v3.0–v3.8 with **Phase 1–5** columns. Mapping: Phase 1=Generation, Phase 2=Refinement, Phase 5=Testing. Phase 3/4 switch between **KG-Adv** and **Review** depending on whether knowledge is injected *pre* or *post* Review. Solid fill = knowledge injected at that phase; dashed/light = phase present with no injection; — = phase not present; † ♦ denotes pre-generation (role-configuration) injection.

after the review phase (v3.6) yielded the highest net improvements with minimal regressions, whereas placing it *before* review (v3.7) also improved over baseline but was slightly less consistent. Overall, these results favor a *specialized-agent* approach over prompt-only modifications and indicate that the optimal workflow inserts the Knowledge Advisor immediately *after* review and before formatting-consistency checks, highlighting the importance of both capacity allocation and phase ordering.

Observation 6.3: Dedicated Knowledge Advisor yields the largest expert-judged gains

Prompt-only integration (v3.0–v3.2) produced mixed results, while doubling reviewer cycles (v3.5) improved consistency without surpassing baseline. Introducing a specialized *Knowledge Advisor* immediately *after* semantic review (v3.6) delivered the most reliable improvements with minimal regressions.

Knowledge-enhanced Pipeline vs Baseline

The expert conducted a manual assessment of the KG-enhanced variants (MAO v3.6 and v3.7) against the non-enhanced baseline (MAO v2.2). For each IG, a *net change* Δ was computed as Equation 5.1. All values are measured relative to MAO v2.2. Detailed evaluation results are available in Appendix E.1.

Table 6.4 reports these Δ scores for each IG and variant.

Input Guideline	Δ (MAO v3.6 vs v2.2)	Δ (MAO v3.7 vs v2.2)
1	+2.0	+1.0
2	+5.0	+1.5
3	+8.0	+6.0
4	+4.5	+1.0

Table 6.4: Net expert-comment changes relative to MAO v2.2 (Positive = +1, Neutral = 0, Negative = −1, Slight ±0.5).

Overall, MAO v3.6 yields the most substantial improvements across all IGs. MAO v3.7 also outperforms the baseline but with smaller gains and some regressions (IG 1 and IG 4).

Summary and Discussion: Comparing the best versions of the Knowledge-enhanced MAO versions (V3.6 and V3.7) with the baseline MAO demonstrated improved results for both Knowledge-enhanced versions. Among the V3.6 and V3.7, the expert evaluation indicated better gains on v3.6.

The expert's primary concern with the previous versions was that several knowledge statements were formulated too broadly, leaving room for multiple interpretations or lack-

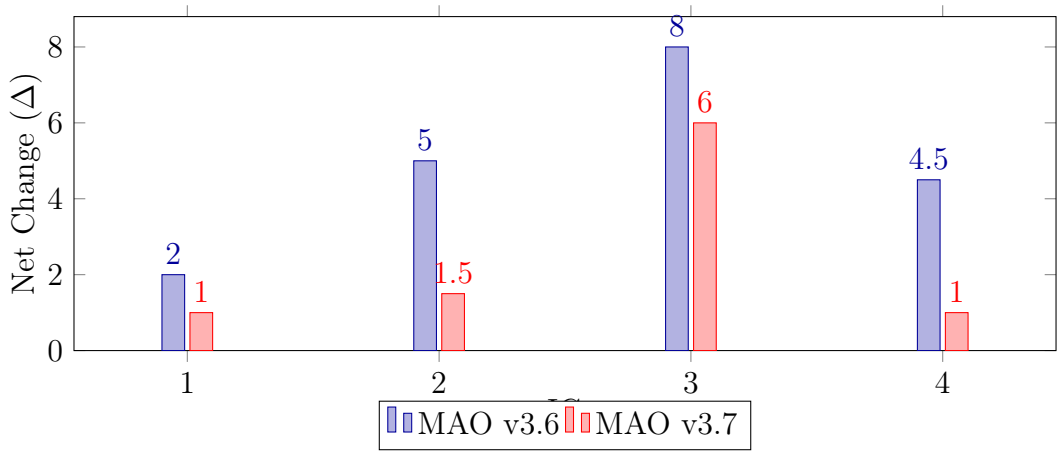


Figure 6.5: Visualization of net expert-comment improvements for MAO v3.6 and v3.7 relative to v2.2.

ing sufficient operational detail. This imprecision often resulted in modeling inconsistencies, particularly around activity sequencing, concurrency, and prerequisite handling.

Future work should refine the Knowledge Advisor prompts to include more domain reasoning and explicit action plans, which may further improve activity placement and sequencing.

Observation 6.4: Expert deltas: v3.6 dominates across all Input Guidelines

Relative to v2.2, v3.6 achieved the strongest net positive changes on every IG, peaking at $\Delta = +8$ for IG 3, whereas v3.7 improved over baseline but introduced occasional regressions (Table 6.4, Fig. 6.5). This supports the post-review placement of the advisor. Expert suggests making the knowledge statements more explicit and operational (clear sequencing, concurrency rules, and prerequisites) should reduce residual ambiguities and further improve outcomes.

The Structure of Knowledge Statements

The expert evaluation aimed to assess whether the refinements introduced in version 3.8 of the knowledge contexts effectively resolved issues identified in earlier iterations (versions 3.6 and 3.7).

Manual assessment of v3.8 vs. v3.6 and v3.7 Using the same net-change scoring as in Equation 5.1, Table 6.5 reports Δ per IG when comparing MAO v3.8 to v3.6 and v3.7. Figure 6.6 visualizes these deltas. Detailed evaluation results are available in Appendix E.2.

Input Guideline	Δ (v3.8 vs v3.6)	Δ (v3.8 vs v3.7)
1	0.0	+1.0
2	-1.0	+2.0
3	+1.5	+1.5
4	+0.5	+1.5

Table 6.5: Net expert-comment changes for MAO v3.8 relative to v3.6 and v3.7 (Positive = +1, Neutral = 0, Negative = −1, Slight ±0.5).

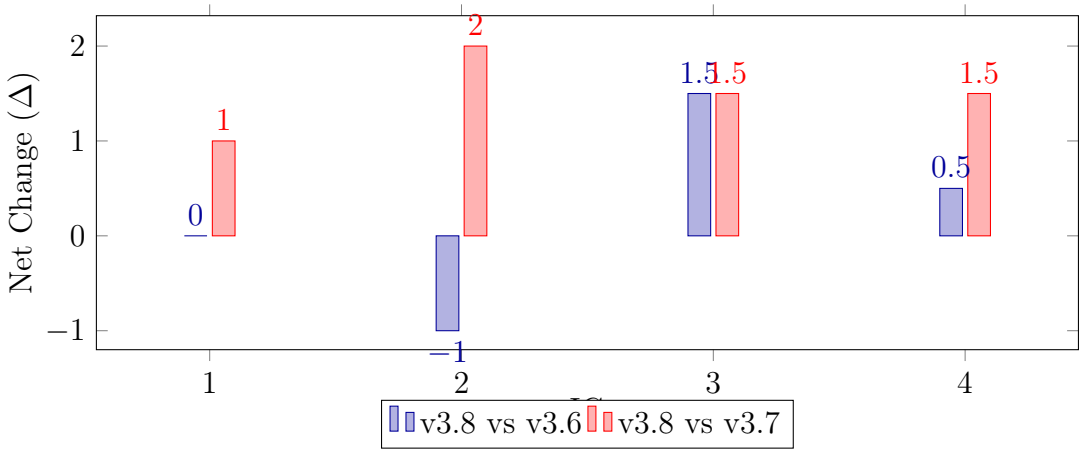


Figure 6.6: Net expert-comment changes for MAO v3.8 relative to v3.6 and v3.7.

Summary and Discussion. Aggregated across IGs, v3.8 scores a modest net +1.0 over v3.6 and a stronger net +6.0 over v3.7, with the only regression vs. v3.6 appearing on IG 2 which is the most complex input guideline.

Despite these improvements, the expert identified residual ambiguities in several areas:

- **Interpretation timing:** In complex branches involving multiple concurrent tests, the required immediacy of expert interpretation was still not fully explicit.
- **Conditional pathways:** The skip logic for optional diagnostics (e.g., chest X-ray) could be reinforced with clearer conditional phrasing and examples.

The expert concluded that version 3.8 represents a net positive refinement over earlier iterations. Most of the issues previously observed were corrected, and model behavior now aligns more closely with the intended clinical pathway. However, the expert also noted that further iterations should aim to make the knowledge contexts even more explicit and example-driven, ensuring that the remaining ambiguities are fully resolved.

Observation 6.5: Expert deltas for v3.8: clear gains over v3.7; modest edge over v3.6

Relative to v3.7, v3.8 shows consistent improvements across all IGs (total $\Delta = +6.0$), and relative to v3.6 it delivers a small net gain (total $\Delta = +1.0$) with a single shortfall on IG 2. These trends align with the qualitative finding that tightening statement specificity reduced sequencing and concurrency ambiguity.

6.2.2 Systematic Similarity Analysis

Integration Strategies

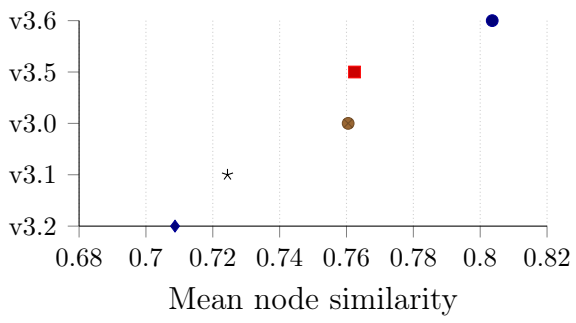
MAO versions 3.0–3.2 were run five times per guideline (20 runs total), while the higher versions 3.5, 3.6, and 3.7 were run fifteen times per guideline (60 runs total). Table 6.6 and Figure 6.7 provide the results for v3.0-3.6.

Table 6.6: Performance of integration strategies (MAO-v3.0–v3.6) on systematic evaluation metrics.

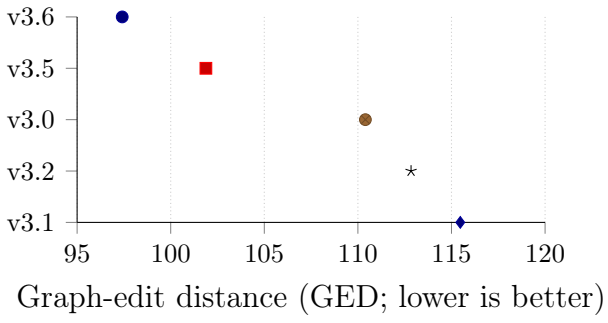
Version	Mean Node Similarity	Mean Structural Similarity	Mean GED
MAO-v3.0	0.7605	0.6277	110.40
MAO-v3.1	0.7244	0.5924	115.47
MAO-v3.2	0.7087	0.5888	112.84
MAO-v3.5	0.7624	0.6256	101.88
MAO-v3.6	**0.8036**	**0.6423**	**97.41**

To assess the impact of the advisor's placement, we directly compared versions 3.6 (advisor after review) and 3.7 (advisor before review) in Table 6.7:

Summary and Discussion. As shown in Table 6.6, early-phase integration (v3.0) yields a modest gain over baseline in both node and structural similarity, indicating that incor-



(a) Node similarity.



(b) GED.

Figure 6.7: Performance across knowledge-integration strategies (Table 6.6).

Table 6.7: Comparison of MAO-v3.6 and MAO-v3.7 integration strategies with MAO-v2.2 baseline.

Version	Mean Node Similarity	Mean Structural Similarity	Mean GED
MAO-v2.2	0.7815	0.6303	107.98
MAO-v3.6	0.8036	0.6423	**97.41**
MAO-v3.7	**0.8074**	**0.6452**	97.52

porating knowledge into the Process Design Reviewer has a positive effect. However, embedding statements solely into the Process Design Expert prompts (v3.1 and v3.2) consistently degrades performance, suggesting that later-phase knowledge insertion fails to resolve early semantic ambiguities. Doubling the review cycles while injecting knowledge (v3.5) recovers baseline fidelity and reduces graph-edit distance, implying that extended deliberation alone is insufficient without a dedicated knowledge component. Notably, the introduction of a specialized Knowledge Advisor agent after the standard review phase (v3.6) achieves the largest improvements across all metrics—node similarity increases by over 4 points and GED decreases by nearly 13—demonstrating the effectiveness of isolating contextual reasoning into its own workflow phase.

Both placements of the Knowledge Advisor yield near-identical performance, with version 3.7 showing only marginal increases in similarity and a negligible change in GED. These differences lie well within the observed run-to-run variability, indicating that once a dedicated advisor is employed, its exact positioning relative to the review cycles has minimal effect.

Overall, the results demonstrate that the most impactful strategy is to introduce a standalone Knowledge Advisor agent (v3.6/v3.7), which delivers statistically reliable gains in both node-level accuracy and structural alignment compared to earlier integration approaches.

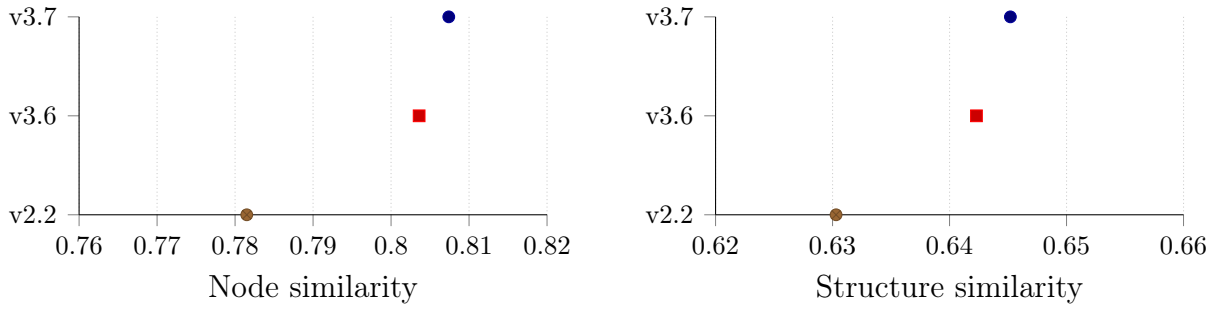
Observation 6.6: Isolating contextual reasoning improves all graph metrics

Among v3.0–v3.7, v3.6 and v3.7 achieved the best combination of higher node/structural similarity and lower GED (e.g., GED ↓ from 110–115 to 97; Table 6.6) with near-identical performance. This suggests that isolating knowledge reasoning into its own phase is more effective than embedding statements in generation/review prompts.

Knowledge-enhanced vs Baseline

As shown in Table 6.7, MAO-v3.7 demonstrates improved performance over MAO-v2.2, yielding higher average node similarity (0.807 vs. 0.782) and structural similarity (0.645 vs. 0.630). Variability between runs remained similar across versions, and the two similarity measures were strongly correlated (r = 0.88; not shown), suggesting that improvements in node-level matching generally coincide with better structural alignment.

Welch's t-tests indicate that the increase in node similarity is statistically significant ($t(\approx 118) = 2.38$, $p = 0.019$, Cohen's $d = 0.43$), reflecting a moderate effect size. The improvement in structural similarity, while positive, did not reach significance ($t(\approx 115) = 1.75$, $p = 0.083$, Cohen's $d = 0.32$). MAO-v3.7 provides a statistically reliable gain in node-level accuracy, with only a non-significant trend toward improved structural similarity.



(a) Node similarity. **(b)** Structural similarity.

Figure 6.8: Baseline vs knowledge-enhanced variants (Table 6.7).

Observation 6.7: Knowledge-enhanced MAO outperforms baseline on expert and graph metrics

Relative to MAO v2.2, both v3.6 and v3.7 increase node similarity (0.804/0.807 vs. 0.782) and structural similarity (0.642/0.645 vs. 0.630) while reducing GED by ≈ 10 to ~ 97 (Table 6.7). The v3.7 node-similarity gain is statistically significant ($t \approx 2.38$, $p = 0.019$, $d = 0.43$), with a positive but non-significant structural trend. Expert deltas also favor the KG variants—v3.6 shows the largest net improvements across all Input Guidelines (Table 6.4). Overall, isolating domain knowledge produces measurable accuracy gains without added variability.

Structure of Knowledge Statements

Table 6.8: Comparison of MAO-v3.6 and MAO-v3.8 performance metrics (mean and median).

Version	Node Similarity		Structural Similarity		GED	
	Mean	Median	Mean	Median	Mean	Median
MAO-v3.6	0.8036	0.7979	**0.6423**	0.6266	**97.41**	89.19
MAO-v3.8	0.8036	**0.8042**	0.6391	**0.6320**	103.32	**88.60**

MAO-v3.8 vs. MAO-v3.6: As shown in Table 6.8, MAO-v3.8 and MAO-v3.6 exhibit virtually indistinguishable mean performance—average node similarity (0.804 vs. 0.804), structural similarity (0.639 vs. 0.642), and graph-edit distance (103.3 vs. 97.4). However, median scores under v3.8 are consistently slightly better, with node similarity rising from 0.7979 to 0.8042, structural similarity from 0.6266 to 0.6320, and median graph-edit distance decreasing from 89.19 to 88.60. Variability across runs remains comparable. Welch's t-tests confirm no significant differences (node similarity: $t(\approx 117) = 0.005$, $p = 0.996$, Cohen's $d = 0.001$; structural similarity: $t(\approx 114) = -0.37$, $p = 0.711$, $d = -0.07$; graph-edit distance: $t(\approx 115) = 0.74$, $p = 0.454$, $d = 0.14$), indicating negligible effect sizes and statistical equivalence between MAO-v3.6 and MAO-v3.8, albeit with modest median gains favoring v3.8.

Observation 6.8: Wording refinements aid interpretability more than metrics

v3.8 and v3.6 have statistically equivalent means on all metrics, but v3.8 shows slightly better medians. This implies clearer wording improves expert interpretability and local consistency without materially shifting global similarity measures (Table 6.8).

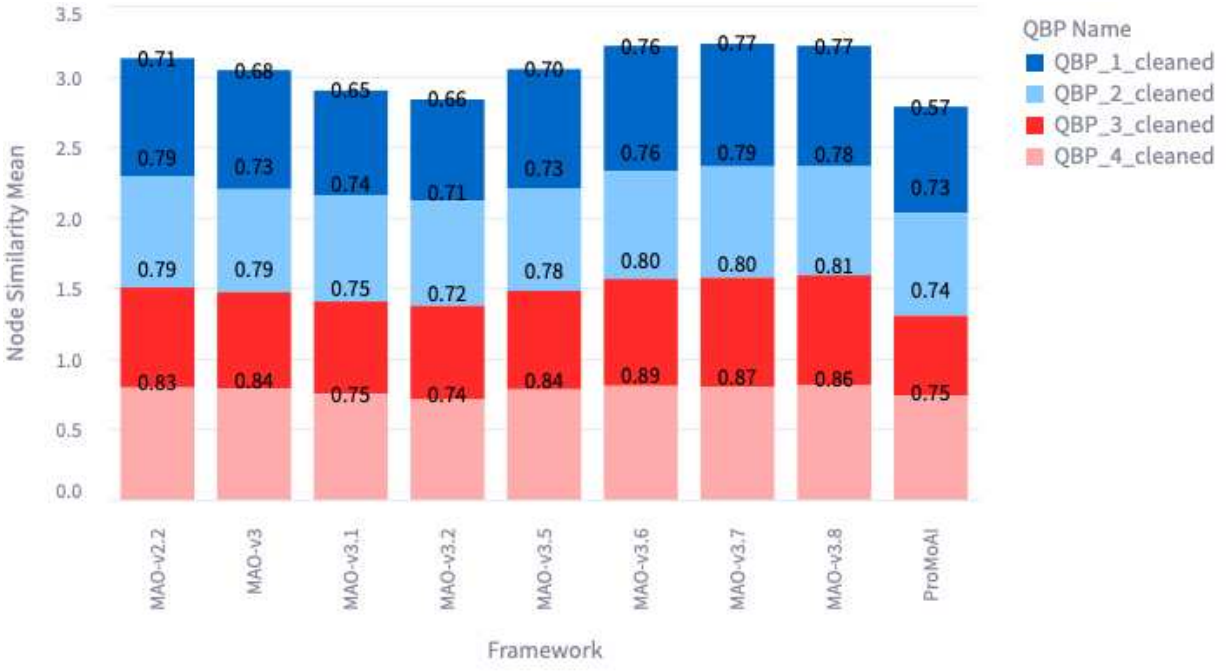


Figure 6.9: Per-IG node similarity across frameworks.

6.2.3 Discussion

Converging findings. Across both evaluation methods, the most consistent gain comes from introducing a dedicated *Knowledge Advisor* phase. Expert deltas from the *process modeling expert* favor the knowledge–enhanced variants over the baseline (MAO v2.2), with the largest expert-judged improvements under MAO v3.6 (post-review placement). The systematic similarity analysis aligns: both v3.6 and v3.7 improve node similarity and reduce GED relative to the baseline (Tables 6.6 and 6.7; Figs. 6.7 and 6.8). Together, these results indicate that isolating contextual reasoning in a standalone phase outperforms prompt-only integration (v3.0–v3.2) and cycle-count increases without specialization (v3.5).

Relative weighting of evidence. Because the systematic evaluation considers *only*

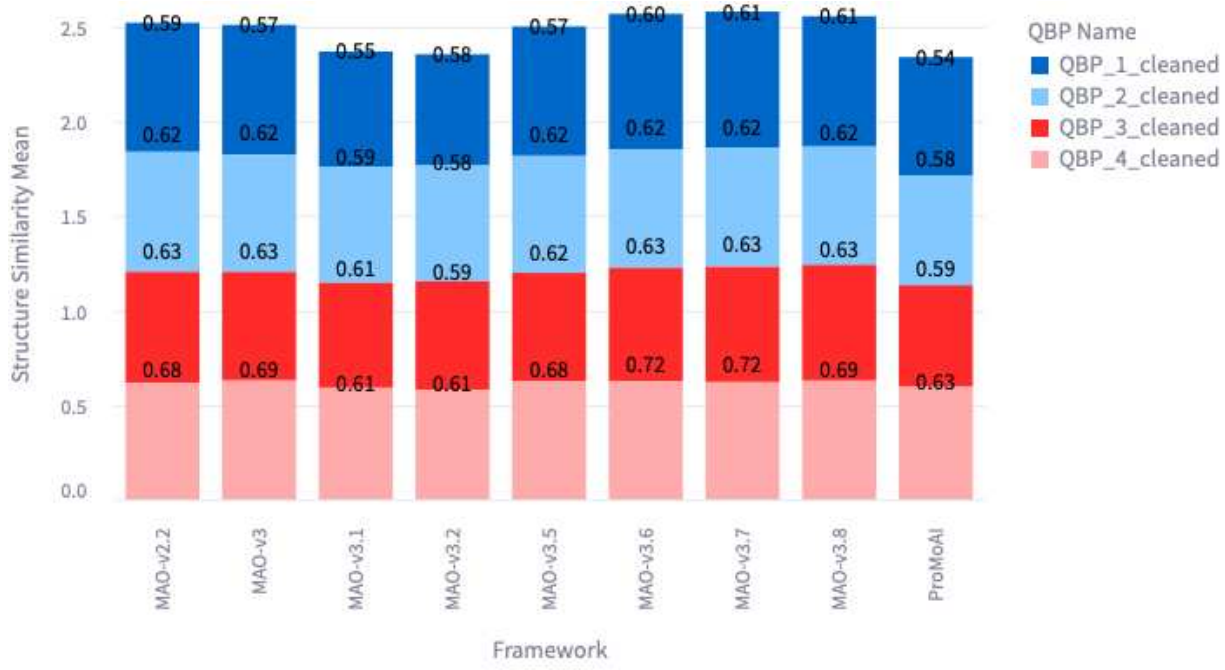


Figure 6.10: Per-IG structural similarity across frameworks.

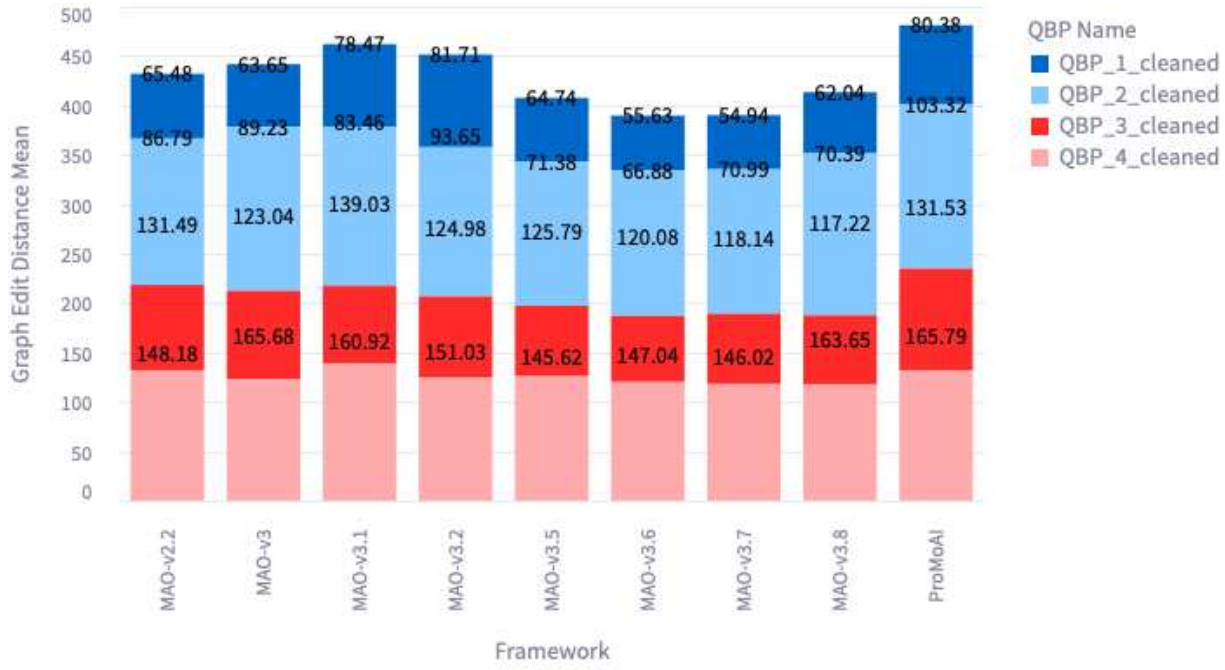


Figure 6.11: Per-IG GED means across frameworks.

the graph structure, it offers a topology-level view of fidelity. In contrast, the process-modeling expert explicitly judges local clinical-logic behaviors (e.g., sequencing, concur-

rency, skip conditions) that small topological changes may not fully capture. Accordingly, we treat the expert evaluation as the *primary* signal and use graph metrics as *corroborative* evidence.

Placement sensitivity. The expert prefers the *post-review* placement (v3.6) to the *pre-review* placement (v3.7), citing fewer local regressions, whereas the graph metrics show near parity between the two. This divergence is expected given the different sensitivities of the methods: global similarity is similar for both, but post-review better satisfies expert criteria focused on local process semantics.

Statement wording vs. metrics. Tightening the *wording and structure of knowledge statements* (v3.8) yields expert-perceived improvements and slightly better medians, yet mean node/structural similarity and GED are statistically indistinguishable from v3.6 (Table 6.8; Fig. 6.6). Clearer statements therefore primarily enhance interpretability and local consistency without materially shifting global similarity scores.

Implications for the pipeline. (i) A dedicated Knowledge Advisor phase is preferred over prompt-only injection; (ii) *Post-review* placement is recommended; (iii) Knowledge statements with explicit sequencing, concurrency, and skip logic should be authored and fed to reduce residual ambiguities noted by the expert.

6.3 Chapter Summary

In this chapter, we empirically evaluated MAO and ProMoAI, plus knowledge-enhanced variants of MAO, using the LLM4CPW pipeline. In the baseline experiment, MAO outperformed ProMoAI on expert ratings and on systematic metrics (higher node/structural similarity with large, significant effects) while exhibiting lower run-to-run variability. The main error source is tacit clinical context missing from guidelines, motivating knowledge enhancement. We tested integration strategies from prompt-only tweaks to a dedicated Knowledge Advisor. The Advisor yields the largest gains: versions v3.6/v3.7 improved similarity scores and reduced GED relative to baseline; expert deltas were strongest for v3.6. Refining the wording and structure of knowledge statements (v3.8) improved median

behavior and expert interpretability with minimal change to mean metrics. Overall, an agentic workflow that isolates contextual reasoning—via a specialized advisor—consistently improved automated process extraction from clinical guidelines. We conclude by noting limits (residual ambiguities around concurrency, skip logic, and prerequisite handling) and point to future work on more explicit, example-driven knowledge statements and richer domain context.

Chapter 7

Discussion

This chapter interprets the empirical findings and situates them within the broader literature on automatic process extraction from clinical guidelines. We discuss what the results imply for agentic LLM frameworks, knowledge-enhanced process extraction, and evaluation practice, and we delineate remaining limitations.

7.1 Synthesis of Findings

Agentic orchestration improves accuracy and stability. Under the same conditions, the multi-agent framework outperformed the monolithic framework on both expert judgment and graph-based metrics. Delegating generation, refinement, and reviewing to specialized agents reduced common BPMN failure modes (e.g., dangling branches, missing joins, inconsistent gateway semantics) and produced models judged more coherent by an expert reviewer. These observations support the premise that *role specialization* and *phase gating* help LLMs maintain control-flow consistency and have improved performance.

Knowledge-enhanced framework yields targeted and reliable gains. Introducing a dedicated *Knowledge Advisor* agent—configured via a categorization of common clinical/process issues and corresponding prompt templates—improved node-level similarity and reduced pathological outliers, with a modest uplift in structural similarity. Placing

the advisor before or after the review stage had negligible effect, suggesting that isolating contextual reasoning is more consequential than its exact position within the loop. Making the injected statements more explicit further tightened robustness, indicating improved robustness without necessarily changing averages.

Pipeline controls enable objective comparisons. A stage-gated pipeline—spanning LLM execution, output standardization to BPMN, automated similarity metrics, and a results dashboard-enabled like-for-like comparisons between frameworks while preserving detailed logs for auditability. Holding conversion, layout, and evaluation code constant ensured that observed differences are attributable to framework design and knowledge injection rather than to incidental preprocessing or visualization choices.

Having multiple evaluation strategies is necessary. Purely structural metrics (node / structural similarity and graph edit distance) are informative and great for automatically assessing the structural and node similarity of process models. However, they can penalize semantically equivalent refactorings. Augmenting these metrics with expert judgment surfaced clinically salient errors (omitted preconditions, ambiguous XOR/AND usage) that would otherwise be underdetected. Moreover, expert evaluation uncovered the underlying causes of such discrepancies—insights that are difficult to obtain through quantitative analysis alone. Combining quantitative and qualitative evidence provides a more faithful picture of model quality.

7.2 Comparison with Related Work

Prior work on text-to-process modeling has demonstrated that LLMs can draft process models from prose, but persistent issues include control-flow inconsistencies, low accuracy, and limited use of clinical knowledge beyond local context [Grathwol et al., 2025, Grohs et al., 2023, Lin et al., 2025]. Agentic approaches address some of these gaps by assigning dedicated roles (generation, refinement, review, and testing) and iterating until checks pass or reach the maximum number of iterations [Lin et al., 2025]. The present study adds

empirical support for this paradigm in the *clinical pathway* setting: agentic orchestration yielded higher structural fidelity and robustness than a monolithic framework under the same prompts, data, and evaluation code.

A second axis in the literature concerns *knowledge integration*. Many systems rely on in-context exemplars or broad system prompts; fewer operationalize a principled, reusable set of knowledge interventions [Su et al., 2025, Yang et al., 2024b]. Our knowledge integration, driven by issue categorization, makes two key contributions: (i) it specifies *what* to inject—recurring content and control-flow issues observed in clinical pathways—and (ii) it provides templated forms (concise, expanded, explicit) that can be positioned within the agentic workflow. The observed gains, while moderate, are consistent and practically useful, especially for stabilizing outputs across repeated runs.

Methodologically, the study advances evaluation practice by (a) standardizing heterogeneous LLM outputs to BPMN 2.0, (b) quantifying similarity with established measures, and (c) complementing it with expert ratings and feedback. The modular architecture also accommodates additional frameworks and metrics, facilitating cumulative comparison.

7.3 Threats to Validity

7.3.1 Construct Validity

Our automated evaluation emphasizes structural similarity and graph edit distance. While appropriate for BPMN comparisons, these metrics prioritize control-flow isomorphism and label matches; they may undervalue semantically equivalent refactorings. This risk was mitigated by expert review with qualitative annotations, but a more complete assessment would add automated behavioral conformance tests [Dijkman et al., 2011]. Prompt sensitivity is another concern: outputs vary with phrasing and decoding. We fixed templates and parameters across conditions and executed multiple runs per guideline to estimate central tendency and dispersion; nonetheless, alternative prompt designs or larger samples could move the distributions.

7.3.2 Internal Validity

Expert evaluation relied on a single domain expert who is also a co-author, introducing potential rater bias and precluding inter-rater reliability estimates. Mitigations included predefined criteria, partial blinding to specific knowledge variants where feasible, and full artifact retention (prompts, drafts, logs) to enable replication and external audit. Future work should incorporate multiple independent raters and report agreement statistics to strengthen causal inferences about framework effects.

7.3.3 External Validity

The dataset comprises four Input Guidelines (IGs) (2–3 pages each) from the stroke domain. Generalizability to longer guidelines, other specialties (e.g., oncology, pediatrics), or multi-document care bundles remains to be established. Results also reflect the capabilities of the foundation LLM and tested frameworks; different or future models may exhibit different error profiles. Finally, while structural fidelity and expert-rated coherence improved, end-to-end clinical correctness and outcomes impact were not assessed.

7.4 Implications for LLM Frameworks

Two recommendations follow from the evidence. First, adopt *agentic decomposition* with explicit acceptance criteria across drafting, review, and testing to reduce control-flow defects. Second, incorporate a *modular knowledge-integration* phase grounded in a reusable issue-categorization, with explicit, well-scoped statements rather than broad formulations, as these improve robustness across repeated runs. Pair structural metrics with expert review, and preserve complete logs and standardized BPMN for reproducible, auditable evaluations.

7.5 Practical Implications

The end-to-end pipeline from guideline text to BPMN unlocks several practical capabilities for pathway design, deployment, and oversight:

- **Visual clarity and consistent interpretation.** Converting free-text recommendations into explicit control-flow elements (tasks, gateways, loops) reduces interpretive variability across teams and makes edge cases less likely to be missed. While raw BPMN diagrams can be too detailed or complex for routine clinical use, simplified process visualizations—for instance, comparable to directly-follows graphs—can be shown. An extreme example here is a streamlined interface that zooms in on only the possible next steps and associated conditions at the point of care. By separating the computable model from the surface UI, this approach preserves clarity and consistency for clinicians without requiring them to engage directly with full BPMN notation.

- **BPMN as machine-interpretable structure: integration, guidance, and auditability.** Once represented as BPMN, pathways become machine-readable data models rather than static PDFs. Such models enable search, structural validation, simulation, and integration without manual re-interpretation. Moreover:

  - Structured tasks and entry criteria can be consumed directly by **clinical decision support systems (CDSS)**, enabling context-aware "next best action" suggestions conditioned on patient state and pathway logic. Because these steps are derived automatically from guideline text using an LLM framework, the recommendations reflect the latest computable evidence.

  - **Operational and retrospective compliance support.** The same machine-interpretable BPMN model enables real-time compliance checking—alerting clinicians when actions are due, omitted, or executed out of sequence. This introduces a meaningful potential delta for patient care by ensuring that clinicians receive timely, evidence-aligned recommendations during actual care delivery. In

parallel, versioned pathway models combined with event logs also support retrospective comparison of real patient trajectories to intended pathways, enabling safety reviews, quality audits, and iterative improvements for both practice and guidelines.

- **Reduced modelling effort and faster maintenance (LLMs reducing overhead).** Auto-generated first drafts significantly reduce the manual burden of extracting process logic from prose, allowing domain experts to focus on targeted corrections rather than creating diagrams from scratch. When guidelines update, scripted reruns and localized edits refresh only the affected fragments, supporting near-real-time updates and helping keep operational pathway logic current at low cost.

Note. Chapter 8 presents formal answers to the research questions and directions for future work; here we focus on interpretation, positioning, and limitations, per the thesis outline.

Chapter 8

Conclusion and Future Work

This chapter synthesizes the thesis contributions, answers the research questions, and delineates promising next steps.

8.1 Contributions

The main contributions of this thesis are:

- **LLM4CPW, a reproducible, end-to-end pipeline for text-to-BPMN.** The system integrates two state-of-the-art LLM frameworks (ProMoAI from Kourani et al. [2024] and MAO from Lin et al. [2025]), a standardization module that converts heterogeneous outputs to canonical BPMN with auto-layout, an automated evaluation suite (node/structure similarity and GED; drawing on similarity metrics from Dijkman et al. [2011]), a GPT-based post-hoc label-alignment component, and a web UI for configuration and analysis, enabling like-for-like comparisons and auditability.
- **Controlled head-to-head evaluation of agentic vs. monolithic generation.** Under matched conditions (four Ontario stroke QBPs; $n = 15$ trials per framework), MAO consistently outperforms ProMoAI on node and structural similarity, receives higher expert ratings, and exhibits lower run-to-run variability, supporting the value of agentic decomposition for process-model generation.

- **Demonstration of feasibility for automated process model extraction from clinical guidelines.** The experiments establish that guideline-to-BPMN extraction via tailored LLM frameworks is achievable within a reproducible, stage-gated pipeline, providing empirical evidence of feasibility where prior studies had reported otherwise [Grathwol et al., 2025].

- **A principled approach to knowledge-aware modeling driven by issue categorization.** The study codes expert comments and common issues observed in the outputs into a reusable categorization and constructs knowledge statement libraries at three abstraction levels (Concise, Expanded, Explicit). Multiple integration strategies are evaluated (prompt-only vs. dedicated agent; before/after review), showing the potential of a *Knowledge Advisor* agent.

- **Open-source pipeline and comprehensive reproducibility package.** In the spirit of open science, the complete LLM4CPW framework, including prompts, knowledge graphs, evaluation datasets, expert annotations, and analysis dashboards, has been publicly released [Houshidari, 2025] to enable transparent replication, community extension, and systematic benchmarking of future text-to-BPMN approaches.

8.2 Answers to the Research Questions

RQ1. *To what extent can tailored LLM frameworks be used to automatically extract CPW models from natural-text guidelines?*

Answer. Tailored LLM frameworks can reliably produce BPMN models aligned with expert-validated references for the IG used in the evaluation. In matched experiments over four QBPs with $n = 15$ runs per framework, MAO achieved higher node and structural similarity than ProMoAI and received higher expert fidelity ratings, with substantially lower variability across runs. This supports the feasibility of automated extraction and addresses concerns raised by Grathwol et al. [2025].

RQ2. *To what extent can injecting structured clinical knowledge into an LLM-based pipeline improve the accuracy, consistency, and clinical soundness of extracted CPW models?*

Answer. Knowledge-aware prompting via a dedicated *Knowledge Advisor* yields measurable, statistically reliable improvements, especially at the node (label/step) level. Across Knowledge-enhanced versions, isolating contextual reasoning into its own phase improves node similarity and reduces GED relative to the baseline; placement before vs. after review is largely immaterial once the advisor is introduced. More explicit statements tighten dispersion (robustness) without materially shifting means.

8.3 Future Work

Building on these findings, we outline directions with immediate payoff for clinical and methodological utility:

Conversational process modeling. Embed interactive modeling dialogs that let experts confirm intent, supply missing preconditions, and trigger local refactorings (e.g., reorganizing branches, adding skip flows) directly within the UI, informed by recent conversational modeling frameworks [Klievtsova et al., 2025b].

Behavioral compliance metrics. Complement structural similarity with behavioral conformance to avoid penalizing logically equivalent refactorings—addressing a documented limitation of purely structural metrics [Dijkman et al., 2011].

Visual BPMN diff and expert validation. Integrate diff/merge tooling that highlights insertions, deletions, gateway rewires, and role/object changes between two process models; tight UI integration should shorten expert review time and yield machine-actionable feedback for iterative repair which can be combined with conversational process modeling mentioned earlier, drawing on established approaches [La Rosa et al., 2013, Ivanov et al., 2015].

Rater calibration and inter-rater reliability. To mitigate the internal-validity lim-

itation caused by relying on a single domain expert, future studies will involve multiple independent raters under a publicly documented, pre-registered protocol stored in the LLM4CPW repository, with raters blinded—where possible—to which LLM framework or knowledge-enhancement variant produced each model. The current 5-point expert rating will be expanded into a structured, multi-criteria rubric derived from the issue taxonomy (e.g., temporal relations, task granularity, task options). Raters will undergo a brief calibration round before evaluation to ensure all raters interpret and apply the rubric consistently, and agreement will be quantified using weighted κ (Cohen's or Fleiss') and intraclass correlation coefficients (ICC) with 95% confidence intervals. Disagreements below a reliability threshold (e.g., $\kappa < 0.60$) will be resolved by consensus. All rater scores and agreement metrics will be stored in the evaluation database and displayed in the dashboard alongside node/structure similarity and GED for transparency and auditability.

Compliance checking with real patient trajectories. Deploy the extracted CPW models for their intended downstream use: comparing guideline-prescribed pathways against observed care trajectories mined from hospital information systems. This would involve (i) extracting patient trajectories from electronic health records using process mining techniques, (ii) performing conformance checking to identify deviations between the normative CPW models and trajectory instances, and (iii) quantifying compliance via fitness and precision metrics. Such analyses would surface both unwarranted variations from evidence-based practice and opportunities to refine guidelines based on purposeful clinical adaptations. Integration with LLM4CPW would enable scaling across multiple conditions and regional variants, ultimately supporting quality improvement initiatives and clinical decision support systems as outlined in our preliminary work [Van Woensel et al., 2025].

Richer knowledge grounding and terminology alignment. Operationalize links to clinical terminologies (e.g., SNOMED CT) and computable artifacts (e.g., HL7 FHIR `PlanDefinition`) to normalize task/decision labels, explicitly represent required data elements and preconditions, and support execution/simulation.

Generalization and ablations. To probe the generality of our feasibility claim, future work should extend beyond stroke to a heterogeneous set of IGs varying in length/complex-

ity, specialty (e.g., neurology, cardiology, oncology), and document structure (narrative vs. algorithmic). Sampling across these strata (≈12–20 IGs with repeated runs) would allow subgroup analyses and test whether our pipeline holds under more demanding conditions, thereby confirming—or refuting—our conclusions on feasibility. Ablation studies (agent roles, advisor placement, statement verbosity, label alignment) and controlled comparisons of foundation models/decoding settings will further isolate marginal effects while keeping pipeline parameters fixed.

Future prompt design. While the current prompt (C.2) relies on declarative instructions, future iterations can also embed *few-shot* examples (e.g., brief *knowledge statement* → *BPMN correction* pairs) within the Knowledge Advisor or reviewer prompts to teach desired refactorings by analogy. Such exemplars would provide the model with concrete demonstrations of desired reasoning patterns, complementing the declarative guidance and potentially enhancing consistency across runs.

Concluding remark

In conclusion, LLM-based agentic orchestration plus principled, knowledge integration driven by a categorization of recurrent issues substantially improves automated guideline-to-BPMN extraction. The pipeline and findings establish an objective baseline for future clinical pathway maintenance, compliance, and evidence-to-practice translation.

References

Apromore Team. Apromore platform. https://documentation.apromore.org/releasenotes/releasenotes.html, March 2025. Release notes; accessed 2025-08-16.

Abel Armas-Cervantes, Paolo Baldan, Marlon Dumas, and Luciano García-Bañuelos. Bp-diff: A tool for behavioral comparison of business process models. In *Proceedings of the BPM Demo Sessions 2014, co-located with the 12th International Conference on Business Process Management (BPM 2014), Eindhoven, The Netherlands, September 10, 2014*, volume 1295 of *CEUR Workshop Proceedings*, pages 1–5. CEUR-WS.org, 2014. URL http://ceur-ws.org/Vol-1295/paper20.pdf.

Maciej Barelkowski and Ivan Tulaev. bpmn-auto-layout, April 2025. URL https://github.com/bpmn-io/bpmn-auto-layout.

bpmn-io team. bpmn-js-differ: A diffing utility for BPMN 2.0 documents. https://github.com/bpmn-io/bpmn-js-differ, 2024. MIT License; accessed 2025-08-16.

Lorenzo Chiudinelli, Arianna Dagliati, Valentina Tibollo, Sara Albasini, Nophar Geifman, Niels Peek, John H. Holmes, Fabio Corsi, Riccardo Bellazzi, and Lucia Sacchi. Mining post-surgical care processes in breast cancer patients. *Artificial Intelligence in Medicine*, 105:101855, 2020. doi:10.1016/j.artmed.2020.101855.

A. Dagliati, L. Sacchi, A. Zambelli, V. Tibollo, L. Pavesi, J.H. Holmes, and R. Bellazzi. Temporal electronic phenotyping by mining careflows of breast cancer patients. *Journal of Biomedical Informatics*, 66:136–147, February 2017. doi:10.1016/j.jbi.2016.12.012.

Remco Dijkman, Marlon Dumas, Boudewijn van Dongen, Reina Käärik, and Jan Mendling.

Similarity of business process models: Metrics and evaluation. *Information Systems*, 36 (2):498–516, 2011. doi:10.1016/j.is.2010.09.006.

Vedavyas Etikala, Ziboud Van Veldhoven, and Jan Vanthienen. Text2Dec: extracting decision dependencies from natural language text for automated DMN decision modelling. In *Business Process Management Workshops (BPM 2020)*, pages 367–379. Springer, 2020. doi:10.1007/978-3-030-66498-5_27.

European Pathway Association. EPA Care Pathways, 2025. URL http://e-p-a.org/care-pathways/.

Jacques Fize. GMatch4py: A graph matching library for Python. https://github.com/jacquesfize/GMatch4py, 2019. MIT License; accessed 2025-08-16.

Fabian Friedrich, Jan Mendling, and Frank Puhlmann. Process model generation from natural language text. In *Advanced Information Systems Engineering*, pages 482–496. Springer, 2011. doi:10.1007/978-3-642-21640-4_36.

Marcia Tavares Garcia, Marina Macedo Nunes, Marcelo Fantinato, Sarajane Marques Peres, and Lucinéia Heloisa Thom. BPMN-Sim: A multilevel structural similarity technique for BPMN process models. *Information Systems*, 116:102211, 2023. doi:10.1016/j.is.2023.102211.

Roberto Gatta, Mauro Vallati, Carlos Fernandez-Llatas, et al. Clinical guidelines: A crossroad of many research areas. Challenges and opportunities in process mining for healthcare. In *Business Process Management Workshops (BPM 2019)*, pages 545–556. Springer, 2019. doi:10.1007/978-3-030-37453-2_44.

Daniel Grathwol, Han van der Aa, and Hugo A. López. Automating pathway extraction from clinical guidelines: A conceptual model, datasets and initial experiments. In *Cooperative Information Systems*, pages 296–312. Springer, 2025. doi:10.1007/978-3-031-81375-7_17.

Michael Grohs, Luka Abb, Nourhan Elsayed, and Jana-Rebecca Rehse. Large language models can accomplish business process management tasks. In *Business Process Man-*

agement Workshops (BPM 2023), pages 453–465. Springer, 2023. doi:10.1007/978-3-031-50974-2_34.

Ahmed Halioui, Petko Valtchev, and Abdoulaye Baniré Diallo. Bioinformatic workflow extraction from scientific texts based on word sense disambiguation. *IEEE/ACM transactions on computational biology and bioinformatics*, 15(6):1979–1990, 2018. doi:10.1109/TCBB.2018.2847336.

Health Quality Ontario & Ministry of Health and Long-Term Care. Quality-Based Procedures: Clinical Handbook for Heart Failure (Acute and Postacute). Technical report, Health Quality Ontario, 2023. URL https://www.ontario.ca/files/2023-12/moh-qbp-clinical-handbook-heart-failure-en-2023-12-11.pdf.

Health Quality Ontario and Ministry of Health and Long-Term Care. Quality-Based Procedures: Clinical Handbook for Stroke (Acute and Postacute). Technical report, Health Quality Ontario, December 2016. URL https://www.ontario.ca/page/quality-based-procedures.

Emmanuel Helm, Andreas Pointner, Oliver Krauss, Andreas Schuler, Barbara Traxler, Klaus Arthofer, and Gerhard Halmerbauer. FHIR2BPMN: delivering actionable knowledge by transforming between clinical pathways and executable models. In *Healthcare of the Future 2022*, volume 292, pages 9–14. IOS Press, 2022. doi:10.3233/SHTI220311.

Alireza Houshidari. LLM4QBP, 2025. URL https://github.com/alireza-hsi/LLM4QBP.

Ken Huang, editor. *Agentic AI: Theories and Practices*. Progress in IS. Springer, 2025. doi:10.1007/978-3-031-90026-6.

Institute of Medicine (US) Committee on Standards for Developing Trustworthy Clinical Practice Guidelines. *Clinical Practice Guidelines We Can Trust*. Washington (DC): National Academies Press (US), 2011. URL https://www.ncbi.nlm.nih.gov/books/NBK209539/.

Sergey Ivanov, Anna A. Kalenkova, and Wil M. P. van der Aalst. BPMNDiffViz: A tool for BPMN models comparison. In *BPM Demo Session 2015*, volume 1418 of *CEUR*

Workshop Proceedings, pages 35–39. CEUR-WS.org, 2015. URL https://ceur-ws.org/Vol-1418/paper8.pdf.

Said Abasse Kassim, Jean-Baptiste Gartner, Laurence Labbé, Paolo Landa, Catherine Paquet, Frédéric Bergeron, Célia Lemaire, and André Côté. Benefits and limitations of business process model notation in modelling patient healthcare trajectory: a scoping review protocol. *BMJ Open*, 12(6):e060357, 2022. doi:10.1136/bmjopen-2021-060357.

Nataliia Klievtsova, Timotheus Kampik, Juergen Mangler, and Stefanie Rinderle-Ma. Conversationally actionable process model creation. In *Cooperative Information Systems*, pages 39–55. Springer, 2025a. doi:10.1007/978-3-031-81375-7_3.

Nataliia Klievtsova, Timotheus Kampik, Juergen Mangler, and Stefanie Rinderle-Ma. Conversational process model redesign, 2025b. URL https://arxiv.org/abs/2505.05453.

Humam Kourani, Alessandro Berti, Daniel Schuster, and Wil M.P. van der Aalst. ProMoAI: Process modeling with generative AI. In *Thirty-Third International Joint Conference on Artificial Intelligence*, IJCAI '24, August 2024. doi:10.24963/ijcai.2024/1014.

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. doi:10.1007/s10270-025-01318-w.

Naveen Krishnan. AI Agents: Evolution, architecture, and real-world applications. *arXiv preprint arXiv:2503.12687*, March 2025. doi:10.48550/arXiv.2503.12687. URL https://arxiv.org/abs/2503.12687.

Matthias Kunze, Matthias Weidlich, and Mathias Weske. Behavioral similarity – a proper metric. In Stefanie Rinderle-Ma, Farouk Toumani, and Karsten Wolf, editors, *Business Process Management (BPM 2011)*, volume 6896 of *Lecture Notes in Computer Science*, pages 166–181, Berlin, Heidelberg, 2011. Springer. doi:10.1007/978-3-642-23059-2_15.

Marcello La Rosa, Marlon Dumas, Reina Uba, and Remco Dijkman. Business process

model merging: An approach to business process consolidation. *ACM Trans. Softw. Eng. Methodol.*, 22(2), March 2013. doi:10.1145/2430545.2430547.

Leilei Lin, Yumeng Jin, Yingming Zhou, Wenlong Chen, and Chen Qian. MAO: A framework for process model generation with multi-agent orchestration. *IEEE Transactions on Services Computing*, pages 1–14, 2025. doi:10.1109/TSC.2025.3611816.

Felix Mannhardt and Daan Blinde. Analyzing the Trajectories of Patients with Sepsis using Process Mining. In *RADAR+EMISA 2017*, volume 1859 of *CEUR Workshop Proceedings*, pages 72–80. CEUR-WS.org, 2017. URL https://ceur-ws.org/Vol-1859/bpmds-08-paper.pdf.

Jorge Muñoz-Gama and Josep Carmona. A fresh look at precision in process conformance. In Richard Hull, Jan Mendling, and Stefan Tai, editors, *Business Process Management*, pages 211–226, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-15618-2.

Jorge Munoz-Gama, Niels Martin, et al. Process mining for healthcare: Characteristics and challenges. *Journal of Biomedical Informatics*, 127:103994, 2022. doi:10.1016/j.jbi.2022.103994.

Samia Nasiri, Amina Adadi, and Mohammed Lahmer. Automatic generation of business process models from user stories. *International Journal of Electrical and Computer Engineering*, 13(1):809, 2023. doi:10.11591/ijece.v13i1.pp809-822.

NetworkX Developers. NetworkX: Graph edit distance (similarity) functions. https://networkx.org/documentation/stable/reference/algorithms/similarity.html, 2025. Version 3.5 documentation; accessed 2025-08-16.

OMG. Business Process Model and Notation (BPMN), version 2.0.2, 2014. URL https://www.omg.org/spec/BPMN/2.0.2.

OpenAI. GPT 4.1, 2025. URL https://platform.openai.com/docs/models/gpt-4.1.

S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, and X. Wu. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*, 36(7):3580–3599, 2024. doi:10.1109/TKDE.2024.3352100.

Andrew Partington, Moe Wynn, Suriadi Suriadi, Chun Ouyang, and Jonathan Karnon. Process mining for clinical processes: A comparative analysis of four Australian hospitals. *ACM Trans. Manage. Inf. Syst.*, 5(4), January 2015. doi:doi/10.1145/2629446.

A. Pini, R. Brown, and M. T. Wynn. Process visualization techniques for multi-perspective process comparisons. In *Asia Pacific Business Process Management*, pages 183–197. Springer, 2015. doi:10.1007/978-3-319-19509-4_14.

Process Intelligence Solutions. PM4Py - Process Mining for Python, 2025. URL https://processintelligence.solutions/pm4py.

Luis Quishpi, Josep Carmona, and Lluís Padró. Extracting decision models from textual descriptions of processes. In *Business Process Management (BPM 2021)*, pages 85–102. Springer, 2021. doi:10.1007/978-3-030-85469-0_8.

Samuel Rodríguez-Loya, Ali Aziz, and Craig Chatwin. A service oriented approach for guidelines-based clinical decision support using BPMN. In *Studies in Health Technology and Informatics*, volume 205, pages 107–113. IOS Press, 2014. doi:10.3233/978-1-61499-432-9-43.

Thomas Rotter, Leigh Kinsman, Erica James, Andreas Machotta, Holger Gothe, Jon Willis, Pamela Snow, and Joachim Kugler. Clinical pathways: effects on professional practice, patient outcomes, length of stay and hospital costs. *Cochrane Database of Systematic Reviews*, 17(3):CD006632, March 2010. doi:10.1002/14651858.CD006632.pub2.

Thomas Rotter, Robert Baatenburg de Jong, Sara Evans Lacko, Ulrich Ronellenfitsch, and Leigh Kinsman. Clinical pathways as a quality strategy. In *Improving healthcare quality in Europe: Characteristics, effectiveness and implementation of different strategies*, Health Policy Series, No. 53. European Observatory on Health Systems and Policies, 2019. URL https://www.ncbi.nlm.nih.gov/books/NBK549262/.

Ranjan Sapkota, Konstantinos I. Roumeliotis, and Manoj Karkee. AI agents vs. agentic AI: A conceptual taxonomy, applications and challenges. *arXiv preprint arXiv:2505.10468*, May 2025. doi:10.48550/arXiv.2505.10468.

Mariachiara Savino, Carlos Fernandez-Llatas, et al. Error-correcting methodology for evaluating compliance to clinical guidelines: A case study on rectal cancer. In *Process Mining Workshops (ICPM 2023)*, pages 309–320. Springer, 2024. doi:10.1007/978-3-031-56107-8_24.

Juan Sequeda, Dean Allemang, and Bryon Jacob. A benchmark to understand the role of knowledge graphs on large language model's accuracy for question answering on enterprise sql databases. In *Proceedings of the 7th Joint Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA)*, GRADES-NDA '24. ACM, 2024. doi:10.1145/3661304.3661901.

Ida Silfverskiöld. Agentic AI: Comparing new open-source frameworks. https://medium.com/data-science-collective/agentic-ai-comparing-new-open-source-frameworks-21ec676732df, April 2025.

Thanner Soares Silva, Lucinéia Heloisa Thom, Aline Weber, José Palazzo Moreira de Oliveira, and Marcelo Fantinato. Empirical analysis of sentence templates and ambiguity issues for business process descriptions. In *On the Move to Meaningful Internet Systems. OTM 2018 Conferences*, pages 279–297. Springer, 2018. doi:10.1007/978-3-030-02610-3_16.

Xiaorui Su, Yibo Wang, Shanghua Gao, Xiaolong Liu, Valentina Giunchiglia, Djork-Arné Clevert, and Marinka Zitnik. KGARevion: An AI agent for knowledge-intensive biomedical QA. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=tnB94WQGrn.

Suriadi Suriadi, Ronny S. Mans, Moe T. Wynn, Andrew Partington, and Jonathan Karnon. Measuring Patient Flow Variations: A Cross-Organisational Process Mining Approach. In *Asia Pacific Business Process Management*, pages 43–58. Springer, 2014. doi:10.1007/978-3-319-08222-6_4.

Wil M. P. van der Aalst, Shengnan Guo, and Pierre Gorissen. Comparative process mining in education: An approach based on process cubes. In *Third IFIP WG 2.6, 2.12 International Symposium, SIMPDA*, volume 203, pages 110–134, 2013. doi:10.1007/978-3-662-46436-6_6.

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. doi:10.1080/17517575.2024.2417404.

William Van Woensel, Alireza Houshidari, and Daniel Amyot. Towards automated compliance checking for care trajectories: Process extraction using large language models. In *2nd International Workshop on Process Mining Applications for Healthcare (PM4H25)*, LNAI. Springer, 2025.

Alain Vanasse, Yohann M Chiu, Josiane Courteau, Marc Dorais, Gillian Bartlett, Kristina Zawaly, and Mike Benigeri. Cohort profile: The care trajectories—enriched data (TorSaDE) cohort. *International Journal of Epidemiology*, 50(4):1066–1066h, 11 2021. doi:10.1093/ije/dyaa167.

Paweł Wamej et al. bpmn-python: Comparing the similarity of two BPMN graphs, April 2025. URL `https://github.com/KrzyHonk/bpmn-python/pull/38`.

Indra Waspada and Riyanarto Sarno. An improved method of graph edit distance for business process model similarity measurement. In *4th International Conference on Informatics and Computational Sciences (ICICoS)*, pages 1–6. IEEE, November 2020. doi:10.1109/ICICoS51170.2020.9299037.

Jingfeng Yang, Hongye Jin, Ruixiang Tang, Xiaotian Han, Qizhang Feng, Haoming Jiang, Shaochen Zhong, Bing Yin, and Xia Hu. Harnessing the power of LLMs in practice: A survey on ChatGPT and beyond. *ACM Trans. Knowl. Discov. Data*, 18(6), April 2024a. doi:10.1145/3649506.

Linyao Yang, Hongyang Chen, Zhao Li, Xiao Ding, and Xindong Wu. Give us the facts: Enhancing large language models with knowledge graphs for fact-aware language modeling. *IEEE Transactions on Knowledge and Data Engineering*, 36(7):3091–3110, 2024b. doi:10.1109/TKDE.2024.3360454.

Appendix A

MAO Raw Execution Output

This appendix provides the detailed MAO output discussed in Section 3.2.

MAO Execution Output (verbatim excerpt of MAO raw execution output)

```
<process>
 <parallelGateway id='parallelgateway0'>
  <branch id='branch0-1'>
   <activity role='ED clinician' action='Screen/assess patient for diabetes' objects='patient,
   diabetes screening tool, EHR' id='activity1-1'/>
  </branch>
  <branch id='branch0-2'>
   <activity role='ED clinician' action='Screen/assess patient for dental issues' objects='patient,
   dental screening tool, EHR' id='activity1-2'/>
  </branch>
  <branch id='branch0-3'>
   <activity role='ED clinician' action='Screen/assess patient for cognitive impairment'
   objects='patient, cognitive screening tool, EHR' id='activity1-3'/>
  </branch>
  <branch id='branch0-4'>
   <activity role='ED clinician' action='Screen/assess patient for depression' objects='patient,
   depression screening tool, EHR' id='activity1-4'/>
  </branch>
 </parallelGateway>
```

```
<activity role='ED clinician' action='Forward screening/assessment results to TIA clinic for follow-up' objects='EHR, screening results' id='activity1-5'/>
<exclusiveGateway id='exclusivegateway1'>
  <branch condition='Patient meets criteria for acute inpatient admission (as per recommendation 2.4.2)' id='branch1'>
    <activity role='admitting physician' action='Verify patient meets stroke unit admission criteria' objects='patient, admission criteria' id='activity2-1'/>
    <activity role='admitting physician' action='Assign patient to co-located stroke unit bed' objects='patient, stroke unit beds' id='activity2-2'/>
    <activity role='hospital administration' action='Ensure dedicated interprofessional stroke team with expertise in medicine, nursing, occupational therapy, physiotherapy, speech-language pathology, social work, and clinical nutrition (dietitian)' objects='staff roster' id='activity3'/>
    <activity role='hospital administration' action='Ensure stroke unit team members spend majority of time treating stroke patients and complete regular stroke care education' objects='staff training records' id='activity4'/>
    <activity role='hospital administration' action='Ensure interprofessional stroke team is available 7 days a week' objects='staff schedule' id='activity5'/>
    <activity role='physiotherapist, speech-language pathologist, occupational therapist' action='Assess patient within 48 hours of admission' objects='patient, assessment tools' id='activity6-1'/>
    <activity role='physiotherapist, speech-language pathologist, occupational therapist' action='Formulate management plan based on assessment' objects='patient, assessment results' id='activity6-2'/>
    <parallelGateway id='parallelgateway1'>
      <branch id='branch3'>
        <activity role='clinical team' action='Mobilize patient early (between 24-48 hours of stroke onset) unless contraindicated; avoid frequent out-of-bed activity within first 24 hours unless clinically justified' objects='patient, mobilization aids' id='activity7'/>
      </branch>
      <branch id='branch4'>
        <activity role='clinicians' action='Use standardized, valid assessment tools to evaluate stroke-related impairments and functional status' objects='patient, assessment tools' id='activity8'/>
      </branch>
      <branch id='branch5'>
```

```
    <activity role='clinicians' action='Complete AlphaFIM assessment on or by day 3 after
    admission (admission day is day 1)' objects='patient, AlphaFIM tool' id='activity9'/>
  </branch>
  <branch id='branch6'>
    <activity role='clinicians' action='Assess all stroke patients for risk of developing venous
    thromboembolism' objects='patient, risk assessment tools' id='activity10'/>
  </branch>
  <branch id='branch7'>
    <activity role='clinical team' action='Encourage early mobilization and adequate hydration
    for all acute stroke patients to help prevent venous thromboembolism' objects='patient,
    hydration plan' id='activity11'/>
  </branch>
  <branch id='branch8'>
    <activity role='clinical team' action='Provide appropriate cross-continuum secondary
    prevention assessments and therapies (Modules 5 and 10), regardless of discharge
    destination' objects='patient, prevention protocols' id='activity15'/>
  </branch>
  <branch id='branch13'>
    <activity role='clinicians' action='Assess nutrition and hydration status within first 48
    hours of admission using valid screening tool' objects='patient, nutrition screening tool'
    id='activity12'/>
  </branch>
</parallelGateway>
<exclusiveGateway id='exclusivegateway2'>
  <branch condition='Nutritional concerns, hydration deficits, dysphagia, or other
  comorbidities present' id='branch9'>
    <activity role='clinicians' action='Refer patient to dietitian within 7 days of admission for
    recommendations and consideration of enteral nutrition support if unable to meet
    requirements' objects='patient, referral form' id='activity13'/>
  </branch>
  <branch condition='No nutritional concerns, hydration deficits, dysphagia, or other
  comorbidities' id='branch10'>
    <activity role='clinicians' action='Continue standard nutrition and hydration monitoring'
    objects='patient' id='activity14'/>
  </branch>
```

```
    </exclusiveGateway>
  </branch>
  <branch condition='Patient does not meet criteria for acute inpatient admission (e.g., TIA or
  minor nondisabling stroke)' id='branch12'>
    <activity role='clinical team' action='Do not apply acute inpatient stroke unit
    recommendations; proceed with alternative care pathway as appropriate' objects='patient,
    alternative pathway' id='activity16'/>
  </branch>
 </exclusiveGateway>
</process>
```

Appendix B

Conversion and Post-processing Scripts

This appendix includes the Python implementations of our i) BPMN converter for MAO's output, ii) script to remove extraneous activities, and iii) post-hoc activity label alignment script. Parts i) and ii) are discussed in Section 3.3, whereas part iii) is discussed in Section 3.4.

B.1 BPMN Converter

The script below compiles MAO's BPMN-Text into BPMN 2.0 XML with explicit gateways, conditions, and well-formed sequence flow references.

Listing B.1: BPMN Converter (bpmn-converter.py)

```
import xml.etree.ElementTree as ET
import argparse
import re

# ----------------------------
# Namespaces
# ----------------------------
BPMN = "http://www.omg.org/spec/BPMN/20100524/MODEL"
BPMNDI = "http://www.omg.org/spec/BPMN/20100524/DI"
OMGDC = "http://www.omg.org/spec/DD/20100524/DC"
OMGDI = "http://www.omg.org/spec/DD/20100524/DI"
XSI = "http://www.w3.org/2001/XMLSchema-instance"

```

```
ET.register_namespace("bpmn", BPMN)
ET.register_namespace("bpmndi", BPMNDI)
ET.register_namespace("omgdc", OMGDC)
ET.register_namespace("omgdi", OMGDI)
ET.register_namespace("xsi", XSI)

# ----------------------------
# Global counter for sequence flow IDs
# ----------------------------
flow_counter = 1
def new_flow_id():
    global flow_counter
    fid = f"Flow_{flow_counter}"
    flow_counter += 1
    return fid

# ----------------------------
# Converters
# ----------------------------
def process_activity(elem):
    """
    Convert an <activity> element from the input BPMN text to a BPMN task.
    Returns (entry_id, exit_id, list_of_task_elements, list_of_sequenceFlows).
    """
    act_id = elem.attrib.get('id')
    role = elem.attrib.get('role')
    action = elem.attrib.get('action')
    objects = elem.attrib.get('objects', '')
    if objects.strip():
        name = f"{action} (Role: {role}; Objects: {objects})"
    else:
        name = f"{action} (Role: {role})"
    task = ET.Element(f"{{{BPMN}}}task", id=act_id, name=name)
    return (act_id, act_id, [task], [])

def process_branch(elem):
    """
    Process a <branch> element if the branch contains multiple activities,
    insert a dummy connector at the start. Returns (branch_entry, branch_exit, nodes
        , flows)
    """
    nodes = []
    flows = []
    children = list(elem)

    if len(children) > 1:
        dummy_id = "dummy_" + new_flow_id()
        dummy = ET.Element(f"{{{BPMN}}}task", id=dummy_id, name="(dummy branch entry)
            ")
        nodes.append(dummy)
        branch_entry = dummy_id
        prev_exit = dummy_id
    else:
        branch_entry = None
        prev_exit = None
```

```
67
68      for child in children:
69          s, e, n, f = process_element(child)
70          nodes.extend(n)
71          flows.extend(f)
72          if s is None or e is None:
73              continue
74          if branch_entry is None:
75              branch_entry = s
76          if prev_exit is not None:
77              sf = ET.Element(f"{{{BPMN}}}sequenceFlow",
78                              id=new_flow_id(), sourceRef=prev_exit, targetRef=s)
79              flows.append(sf)
80          prev_exit = e
81      return (branch_entry, prev_exit, nodes, flows)
82
83  def process_parallel_gateway(elem):
84      """
85      <parallelGateway> with one or more <branch> children.
86      Returns (diverge_id, converge_id, nodes, flows)
87      """
88      pg_id = elem.attrib.get('id')
89      diverge = ET.Element(f"{{{BPMN}}}parallelGateway", id=pg_id, name="Parallel
            Gateway")
90      nodes = [diverge]
91      flows = []
92      branch_entries = []
93      branch_exits = []
94      for branch in elem.findall('branch'):
95          s, e, n, f = process_branch(branch)
96          nodes.extend(n)
97          flows.extend(f)
98          branch_entries.append(s)
99          branch_exits.append(e)
100     converge_id = pg_id + "Join"
101     converge = ET.Element(f"{{{BPMN}}}parallelGateway", id=converge_id, name="
            Parallel Merge")
102     nodes.append(converge)
103     for s in branch_entries:
104         if s is None: # defensive
105             continue
106         sf = ET.Element(f"{{{BPMN}}}sequenceFlow",
107                         id=new_flow_id(), sourceRef=pg_id, targetRef=s)
108         flows.append(sf)
109     for e in branch_exits:
110         if e is None:
111             continue
112         sf = ET.Element(f"{{{BPMN}}}sequenceFlow",
113                         id=new_flow_id(), sourceRef=e, targetRef=converge_id)
114         flows.append(sf)
115     return (pg_id, converge_id, nodes, flows)
116
117 def _add_condition(flow_elem, text):
118     """Attach a proper <bpmn:conditionExpression xsi:type="bpmn:tFormalExpression">
            """
```

```
    flow_elem.attrib["name"] = text
    ce = ET.Element(f"{{{BPMN}}}conditionExpression", attrib={f"{{{XSI}}}type": "
        bpmn:tFormalExpression"})
    ce.text = text
    flow_elem.append(ce)

def process_inclusive_gateway(elem):
    """
    <inclusiveGateway> with <branch condition='...'> children.
    """
    ig_id = elem.attrib.get('id')
    diverge = ET.Element(f"{{{BPMN}}}inclusiveGateway", id=ig_id, name="Inclusive
        Gateway")
    nodes = [diverge]
    flows = []
    branch_info = []
    for branch in elem.findall('branch'):
        s, e, n, f = process_branch(branch)
        nodes.extend(n)
        flows.extend(f)
        condition = branch.attrib.get('condition', '').strip()
        branch_info.append((s, e, condition))
    converge_id = ig_id + "Join"
    converge = ET.Element(f"{{{BPMN}}}inclusiveGateway", id=converge_id, name="
        Inclusive Merge")
    nodes.append(converge)
    for (s, e, cond) in branch_info:
        if s is None or e is None:
            continue
        sf = ET.Element(f"{{{BPMN}}}sequenceFlow", id=new_flow_id(), sourceRef=ig_id,
            targetRef=s)
        if cond:
            _add_condition(sf, cond)
        flows.append(sf)
        sf2 = ET.Element(f"{{{BPMN}}}sequenceFlow", id=new_flow_id(), sourceRef=e,
            targetRef=converge_id)
        flows.append(sf2)
    return (ig_id, converge_id, nodes, flows)

def process_exclusive_gateway(elem):
    """
    <exclusiveGateway> with <branch condition='...'> children.
    """
    eg_id = elem.attrib.get('id')
    diverge = ET.Element(f"{{{BPMN}}}exclusiveGateway", id=eg_id, name="Exclusive
        Gateway")
    nodes = [diverge]
    flows = []
    branch_info = []
    for branch in elem.findall('branch'):
        s, e, n, f = process_branch(branch)
        nodes.extend(n)
        flows.extend(f)
        condition = branch.attrib.get('condition', '').strip()
        branch_info.append((s, e, condition))
```

```
    converge_id = eg_id + "Join"
    converge = ET.Element(f"{{{BPMN}}}exclusiveGateway", id=converge_id, name="
        Exclusive Merge")
    nodes.append(converge)
    for (s, e, cond) in branch_info:
        if s is None or e is None:
            continue
        sf = ET.Element(f"{{{BPMN}}}sequenceFlow", id=new_flow_id(), sourceRef=eg_id,
             targetRef=s)
        if cond:
            _add_condition(sf, cond)
        flows.append(sf)
        sf2 = ET.Element(f"{{{BPMN}}}sequenceFlow", id=new_flow_id(), sourceRef=e,
            targetRef=converge_id)
        flows.append(sf2)
    return (eg_id, converge_id, nodes, flows)

def process_element(elem):
    """
    Determine the element type and process accordingly.
    Returns (entry_id, exit_id, list_of_BPMN_nodes, list_of_sequenceFlows)
    """
    tag = elem.tag
    if tag == "activity":
        return process_activity(elem)
    elif tag == "parallelGateway":
        return process_parallel_gateway(elem)
    elif tag == "inclusiveGateway":
        return process_inclusive_gateway(elem)
    elif tag == "exclusiveGateway":
        return process_exclusive_gateway(elem)
    elif tag == "branch":
        return process_branch(elem)
    else:
        return (None, None, [], [])

# ----------------------------
# Wiring helpers
# ----------------------------
def add_flow_references(process_elem, seq_flow):
    """
    For each sequenceFlow, add <incoming> to target and <outgoing> to source.
    """
    ns = f"{{{BPMN}}}"
    source_id = seq_flow.attrib["sourceRef"]
    target_id = seq_flow.attrib["targetRef"]
    source_elem = process_elem.find(f".//*[@id='{source_id}']")
    if source_elem is not None:
        out_elem = ET.Element(ns + "outgoing")
        out_elem.text = seq_flow.attrib["id"]
        source_elem.append(out_elem)
    target_elem = process_elem.find(f".//*[@id='{target_id}']")
    if target_elem is not None:
        in_elem = ET.Element(ns + "incoming")
        in_elem.text = seq_flow.attrib["id"]
```

```
220         target_elem.append(in_elem)
221
222 def build_minimal_di(definitions, process_elem):
223     """
224     Create a minimal BPMN-DI section so viewers can render ALL elements.
225     One BPMNShape per node (tasks/events/gateways) and one BPMNEdge per sequenceFlow
            .
226     Simple grid layout, left-to-right in document order.
227     """
228     bpmndi = f"{{{BPMNDI}}}"
229     omgdc = f"{{{OMGDC}}}"
230     omgdi = f"{{{OMGDI}}}"
231
232     diagram = ET.SubElement(definitions, f"{bpmndi}BPMNDiagram", id="BPMNDiagram_1")
233     plane = ET.SubElement(diagram, f"{bpmndi}BPMNPlane",
234                           id="BPMNPlane_1", bpmnElement=process_elem.attrib["id"])
235
236     # Layout params
237     start_x, start_y = 100, 100
238     dx, dy = 180, 130
239     row_width = 6 # wrap after N nodes
240
241     centers = {} # id -> (cx, cy)
242     idx = 0
243
244     # Shapes for flow nodes
245     for node in list(process_elem):
246         local = node.tag.split('}')[-1]
247         if local not in ("task", "startEvent", "endEvent", "exclusiveGateway", "
                parallelGateway", "inclusiveGateway"):
248             continue
249
250         # sizes
251         if local == "task":
252             w, h = 100, 80
253         elif local in ("exclusiveGateway", "parallelGateway", "inclusiveGateway"):
254             w, h = 50, 50
255         else: # start/end
256             w, h = 36, 36
257
258         col = idx % row_width
259         row = idx // row_width
260         bx = start_x + col * dx
261         by = start_y + row * dy
262
263         shape = ET.SubElement(plane, f"{bpmndi}BPMNShape",
264                               id=f"{node.attrib['id']}_di", bpmnElement=node.attrib["id
                                    "])
265         ET.SubElement(shape, f"{omgdc}Bounds",
266                       x=str(bx), y=str(by), width=str(w), height=str(h))
267
268         centers[node.attrib["id"]] = (bx + w/2.0, by + h/2.0)
269         idx += 1
270
271     # Edges for sequence flows
```

```
    for flow in [n for n in list(process_elem) if n.tag.split('}')[-1] == "
        sequenceFlow"]:
        sid = flow.attrib["sourceRef"]
        tid = flow.attrib["targetRef"]
        (sx, sy) = centers.get(sid, (start_x, start_y))
        (tx, ty) = centers.get(tid, (start_x + dx, start_y))

        edge = ET.SubElement(plane, f"{bpmndi}BPMNEdge",
                             id=f"{flow.attrib['id']}_di", bpmnElement=flow.attrib["id"
                                 ])
        ET.SubElement(edge, f"{omgdi}waypoint", x=str(int(sx)), y=str(int(sy)))
        ET.SubElement(edge, f"{omgdi}waypoint", x=str(int(tx)), y=str(int(ty)))

# ----------------------------
# Main
# ----------------------------
def main():
    parser = argparse.ArgumentParser(description="Convert custom BPMN-like text to
        BPMN XML with flows, conditions, and DI")
    parser.add_argument("input", help="Input BPMN text file (custom XML-like format,
         e.g. input.txt)")
    parser.add_argument("output", help="Output BPMN XML file")
    args = parser.parse_args()

    # Read and lightly escape attribute values that might break XML
    with open(args.input, "r", encoding="utf-8") as f:
        text = f.read()

    def escape_xml_attr_values(text):
        # Replace <= first to avoid double-escaping
        text = text.replace("<=", "&lt;=")
        # Replace & (but not already-escaped ones)
        text = re.sub(r'&(?!lt;|gt;|amp;|apos;|quot;)', '&amp;', text)
        # Replace < and > in attribute values (but not tags)
        def repl_attr(m):
            val = m.group(2)
            val = (val.replace('<', '&lt;')
                      .replace('>', '&gt;')
                      .replace('"', '&quot;')
                      .replace("'", '&apos;'))
            return f"{m.group(1)}='{val}'"
        text = re.sub(r"(\w+)\s*=\s*'([^']*)'", repl_attr, text)
        return text

    fixed_text = escape_xml_attr_values(text)

    # Parse the custom input (root expected to be <process> with children)
    root = ET.fromstring(fixed_text)

    # Build BPMN document
    definitions = ET.Element(f"{{{BPMN}}}definitions",
                             id="Definitions_1",
                             targetNamespace="http://bpmn.io/schema/bpmn")
    process_elem = ET.SubElement(definitions, f"{{{BPMN}}}process",
                                 id="Process_1", isExecutable="true", name="Stroke
```

```
                                    Process")

    # Start event
    start_event = ET.SubElement(process_elem, f"{{{BPMN}}}startEvent",
                                id="StartEvent_1", name="Start")
    last_exit = "StartEvent_1"

    all_nodes = []
    all_flows = []

    # Process children sequentially, chaining them from start to end
    for child in root:
        s, e, nodes, flows = process_element(child)
        if s is None or e is None:
            continue
        all_nodes.extend(nodes)
        all_flows.extend(flows)
        sf = ET.Element(f"{{{BPMN}}}sequenceFlow",
                        id=new_flow_id(), sourceRef=last_exit, targetRef=s)
        all_flows.append(sf)
        last_exit = e

    # End event and final flow
    end_event = ET.SubElement(process_elem, f"{{{BPMN}}}endEvent",
                              id="EndEvent_1", name="End")
    sf_end = ET.Element(f"{{{BPMN}}}sequenceFlow",
                        id=new_flow_id(), sourceRef=last_exit, targetRef="EndEvent_1")
    all_flows.append(sf_end)

    # Append nodes and flows to the process
    for node in all_nodes:
        process_elem.append(node)
    for flow in all_flows:
        process_elem.append(flow)

    # Add incoming/outgoing refs for each sequenceFlow
    for flow in all_flows:
        add_flow_references(process_elem, flow)

    # Generate minimal DI so viewers can render everything
    build_minimal_di(definitions, process_elem)

    # Write out
    tree_out = ET.ElementTree(definitions)
    tree_out.write(args.output, encoding="UTF-8", xml_declaration=True)

if __name__ == "__main__":
    main()
```

B.2 Remove Extra Activities

The post-processor below removes temporary "dummy" anchors introduced to stabilize layout, merging their incident flows and updating BPMN DI edges accordingly.

Listing B.2: Remove Extra Activities (remove-dummy.py)

```
import xml.etree.ElementTree as ET
import sys

NS = {
    'bpmn': 'http://www.omg.org/spec/BPMN/20100524/MODEL',
    'bpmndi': 'http://www.omg.org/spec/BPMN/20100524/DI',
    'dc': 'http://www.omg.org/spec/DD/20100524/DC',
    'di': 'http://www.omg.org/spec/DD/20100524/DI'
}

def remove_element_from_parent(parent, child):
    """Remove a child element from its parent."""
    for elem in list(parent):
        if elem is child:
            parent.remove(elem)
            return

def find_bpmn_edge(root, flow_id):
    """
    Given a sequence flow id, search for the BPMNEdge element in the BPMNPlane
    that has bpmnElement==flow_id.
    """
    for plane in root.findall('.//bpmndi:BPMNPlane', NS):
        for edge in plane.findall('bpmndi:BPMNEdge', NS):
            if edge.get('bpmnElement') == flow_id:
                return plane, edge
    return None, None

def create_merged_edge(new_flow_id, source_waypoint, target_waypoint):
    """
    Create a new BPMNEdge element for the merged flow.
    Uses the given source and target waypoint coordinates.
    """
    new_edge = ET.Element("{%s}BPMNEdge" % NS['bpmndi'], id="edge_" + new_flow_id,
        bpmnElement=new_flow_id)
    wp1 = ET.Element("{%s}waypoint" % NS['di'])
    wp1.set("x", source_waypoint.get("x"))
    wp1.set("y", source_waypoint.get("y"))
    wp2 = ET.Element("{%s}waypoint" % NS['di'])
    wp2.set("x", target_waypoint.get("x"))
    wp2.set("y", target_waypoint.get("y"))
    new_edge.extend([wp1, wp2])
    return new_edge

def delete_dummy_tasks(input_file, output_file):
```

```
    tree = ET.parse(input_file)
    root = tree.getroot()

    process_elem = root.find('bpmn:process', NS)
    if process_elem is None:
        print("No <bpmn:process> found.")
        return

    new_flows = []

    # Find dummy tasks (by id starting with "dummy_")
    dummy_tasks = [t for t in process_elem.findall('bpmn:task', NS) if t.get("id", "").startswith("dummy_")]

    for dummy in dummy_tasks:
        dummy_id = dummy.get("id")
        print(f"Processing dummy task: {dummy_id}")

        incoming = [f for f in process_elem.findall('bpmn:sequenceFlow', NS) if f.get("targetRef") == dummy_id]
        outgoing = [f for f in process_elem.findall('bpmn:sequenceFlow', NS) if f.get("sourceRef") == dummy_id]

        if len(incoming) != 1 or len(outgoing) != 1:
            print(f" Skipping dummy {dummy_id}: expected 1 incoming and 1 outgoing, got {len(incoming)} and {len(outgoing)}.")
            continue

        inc_flow = incoming[0]
        out_flow = outgoing[0]

        source_ref = inc_flow.get("sourceRef")
        target_ref = out_flow.get("targetRef")
        new_flow_id = f"merged_{inc_flow.get('id')}_{out_flow.get('id')}"

        print(f" Merging flow from '{source_ref}' to '{target_ref}' (new flow id: {new_flow_id})")

        merged_flow = ET.Element("{%s}sequenceFlow" % NS['bpmn'],
                                 id=new_flow_id,
                                 sourceRef=source_ref,
                                 targetRef=target_ref)
        cond = inc_flow.find("bpmn:conditionExpression", NS)
        if cond is not None:
            new_cond = ET.Element("{%s}conditionExpression" % NS['bpmn'], attrib=cond.attrib)
            new_cond.text = cond.text
            merged_flow.append(new_cond)
            if "name" in inc_flow.attrib:
                merged_flow.set("name", inc_flow.get("name"))

        new_flows.append(merged_flow)

        # Remove the old flows and dummy task from the process.
        remove_element_from_parent(process_elem, inc_flow)
```

```
        remove_element_from_parent(process_elem, out_flow)
        remove_element_from_parent(process_elem, dummy)

        # Remove the BPMNShape for the dummy task.
        for plane in root.findall('.//bpmndi:BPMNPlane', NS):
            for shape in plane.findall('bpmndi:BPMNShape', NS):
                if shape.get("bpmnElement") == dummy_id:
                    remove_element_from_parent(plane, shape)
                    print(f" Removed BPMNShape for dummy '{dummy_id}'.")

        # Update BPMNDiagram edges: replace the two edges with one merged edge.
        plane_inc, edge_inc = find_bpmn_edge(root, inc_flow.get("id"))
        plane_out, edge_out = find_bpmn_edge(root, out_flow.get("id"))
        if edge_inc is not None and edge_out is not None and plane_inc is not None
            and plane_out is not None:
            remove_element_from_parent(plane_inc, edge_inc)
            remove_element_from_parent(plane_out, edge_out)
            waypoints_inc = edge_inc.findall("{%s}waypoint" % NS['di'])
            waypoints_out = edge_out.findall("{%s}waypoint" % NS['di'])
            if waypoints_inc and waypoints_out:
                source_wp = waypoints_inc[0]
                target_wp = waypoints_out[-1]
            else:
                source_wp = ET.Element("{%s}waypoint" % NS['di'], x="0", y="0")
                target_wp = ET.Element("{%s}waypoint" % NS['di'], x="0", y="0")
            new_edge = create_merged_edge(new_flow_id, source_wp, target_wp)
            plane_inc.append(new_edge)
            print(f" Created BPMNEdge for merged flow '{new_flow_id}'.")
        else:
            print(f" Warning: Could not locate BPMNEdges for flows of dummy '{
                dummy_id}'.")

    for nf in new_flows:
        process_elem.append(nf)

    tree.write(output_file, encoding="UTF-8", xml_declaration=True)
    print(f"Done. Output written to '{output_file}'.")

if __name__ == "__main__":
    if len(sys.argv) != 3:
        print("Usage: python delete_dummy.py input.bpmn output.bpmn")
        sys.exit(1)
    delete_dummy_tasks(sys.argv[1], sys.argv[2])
```

B.3 Activity Label Alignment

The Python code in Listing B.3 automatically aligns activity labels between two BPMN process models by leveraging an LLM (namely GPT). It extracts activity names from both models, uses GPT to semantically match each activity from the source to the best

candidate in the target (or mark it as *No match*), and then updates the source BPMN file with the harmonized labels, facilitating model comparison and evaluation.

Listing B.3: Activity label alignment script

```
#!/usr/bin/env python3
import os
import xml.etree.ElementTree as ET
from openai import OpenAI
import ast
import re

def extract_python_lists(text):
    """
    Extracts Python list assignments for Set_A and Set_C from the LLM response.
    Returns a string with only the two assignments.
    """
    # Try to extract code from a markdown-style code block
    match = re.search(r"```python(.*?)```", text, re.DOTALL)
    if match:
        return match.group(1).strip()
    # Fallback: extract lines that look like Python list assignments
    lines = text.splitlines()
    code_lines = [line for line in lines if line.strip().startswith("Set_A") or line
        .strip().startswith("Set_C")]
    return "\n".join(code_lines).strip()

# ---------------------------------------------------------------------
# Initialize OpenAI client (requires OPENAI_API_KEY in your environment)
# ---------------------------------------------------------------------
client = OpenAI(api_key=os.getenv("OPENAI_API_KEY"))

# ---------------------------------------------------------------------
# 1) Activity-name extraction
# ---------------------------------------------------------------------
def get_namespace(tag):
    if tag.startswith('{'):
        return tag[1:].split('}', 1)[0]
    return ''

def extract_activity_names(file_path):
    try:
        tree = ET.parse(file_path)
    except Exception as e:
        raise RuntimeError(f"Error parsing {file_path}: {e}")
    root = tree.getroot()
    ns = get_namespace(root.tag)
    activity_tags = [
        'task','userTask','scriptTask','serviceTask','businessRuleTask',
        'manualTask','sendTask','receiveTask','subProcess','callActivity'
    ]
    names = []
    for tag in activity_tags:
```

```
        full_tag = f"{{{ns}}}{tag}"
        for elt in root.findall(f'.//{full_tag}'):
            name = elt.get('name')
            if name:
                names.append(name)
    return names

# ----------------------------------------------------------------
# 2) Call GPT to align Set A --> Set B, keeping original for "No match."
# ----------------------------------------------------------------
def get_alignment(set_a, set_b, model="gpt-4"):
    prompt = f"""
I have two sets of activity labels from two BPMN process models. I want to align each label from Set A to at most one label in Set B if they mean the same thing. If none of the labels in Set B means the same thing, say 'No match.' Return your answer as a table of three columns: (Label from A), (Best Match in B), and (Explanation). Always use full names. Here are the labels:
Set A: {set_a}
Set B: {set_b}

After finding the pairings give two python lists. Set A which is just set A. And Set C which is the pair from Set B or "No match".
"""
    response = client.chat.completions.create(
        model=model,
        messages=[
            {"role": "system", "content": "You are a helpful assistant that aligns labels."},
            {"role": "user", "content": prompt}
        ],
        temperature=0
    )
    text = response.choices[0].message.content

    # Parse the markdown table into mapping and Set C
    lines = text.strip().splitlines()
    start_idx = next((i for i, line in enumerate(lines)
                      if line.strip().startswith("|") and "Label from A" in line), None)
    table_rows = lines[start_idx+2:] if start_idx is not None else []

    mapping = {}
    set_c = []
    for row in table_rows:
        if not row.strip().startswith("|"):
            continue
        cols = [col.strip() for col in row.strip().split("|")[1:-1]]
        if len(cols) < 3:
            continue
        label_a, label_b, _ = cols
        mapped_name = label_b if label_b != "No match." else label_a
        mapping[label_a] = mapped_name
        set_c.append(mapped_name)

    return mapping, set_c, text
```

```
96
97  # ---------------------------------------------------------------------
98  # 3) Call GPT again for revision, but revisions will be post-processed
99  # ---------------------------------------------------------------------
100 def get_revision(set_a, set_b, initial_response, model="gpt-4"):
101     prompt = f"""
102 Below is the original alignment response you gave and the original sets.
103
104 {initial_response}
105
106 Set A: {set_a}
107 Set B: {set_b}
108
109 Please revise your alignment, ensuring you use the full activity names, and return
        only two Python lists:
110 - Set_A: the original Set A list.
111 - Set_C: the corresponding list of best matches (or "No match.").
112 Return them as valid Python list definitions and nothing else.
113 """
114     response = client.chat.completions.create(
115         model=model,
116         messages=[
117             {"role": "system", "content": "You are a helpful assistant that aligns
                labels."},
118             {"role": "user", "content": prompt}
119         ],
120         temperature=0
121     )
122     return response.choices[0].message.content
123
124 # ---------------------------------------------------------------------
125 # 4) Apply mapping to rename activities
126 # ---------------------------------------------------------------------
127 def update_activity_names(input_file, output_file, mapping):
128     tree = ET.parse(input_file)
129     root = tree.getroot()
130     for elt in root.iter():
131         name = elt.get('name')
132         if name in mapping:
133             elt.set('name', mapping[name])
134     tree.write(output_file, encoding="UTF-8", xml_declaration=True)
135     print(f"Updated BPMN file saved as: {output_file}")
136
137 # ---------------------------------------------------------------------
138 # 5) Main flow
139 # ---------------------------------------------------------------------
140 if __name__ == "__main__":
141     file_ai = "/Users/alireza/Desktop/DTI/Thesis/code experiments/Similarity_BPMN/
        BPMNs for test/QBP1_v2.1_Take4_gpt4.1.bpmn"
142     file_gold = "/Users/alireza/Desktop/DTI/Thesis/code experiments/Similarity_BPMN/
        BPMNs for test/QBP_1_.bpmn"
143     out_file = "/Users/alireza/Desktop/DTI/Thesis/code experiments/Similarity_BPMN/
        BPMNs for test/QBP1_v2.1_Take4_changeName_4.1_AAM_7.bpmn"
144
145     set_a = extract_activity_names(file_ai)
```

```
    set_b = extract_activity_names(file_gold)
    print(f"Set A has {len(set_a)} activities.")
    print(f"Set B has {len(set_b)} activities.")

    # First GPT alignment
    mapping, set_c, full_table = get_alignment(set_a, set_b)
    print("\nInitial GPT alignment table:\n", full_table)

    # Revision call
    revised_text = get_revision(set_a, set_b, full_table)
    print("\nRevised Python lists (raw):\n", revised_text)

    # Parse revised lists and enforce original-name fallback
    namespace = {}
    cleaned_code = extract_python_lists(revised_text)
    exec(cleaned_code, {}, namespace)
    rev_set_a = namespace.get('Set_A', set_a)
    rev_set_c = namespace.get('Set_C', set_c)
    # Build final mapping: fallback to original when still 'No match.' or 'No match'
    final_mapping = {}
    for a, c in zip(rev_set_a, rev_set_c):
        if c.strip() in ("No match.", "No match"):
            final_mapping[a] = a
        else:
            final_mapping[a] = c

    print("\nFinal mapping applied:")
    for a, c in final_mapping.items():
        print(f"- {a} -> {c}")

    # Apply to BPMN
    update_activity_names(file_ai, out_file, final_mapping)
```

Appendix C

Contextual Knowledge Statements

This appendix provides the full descriptions of the three contextual knowledge statements defined in Chapter 4 and used in the experiments of Chapter 5. It also provides a verbatim description of a full LLM prompt that uses such knowledge.

C.1 Knowledge Contexts

Concise Knowledge Context (v3.0–v3.4)

Below are additional knowledge contexts to guide the process design:

1) The activity 'Provide cross-continuum secondary prevention assessments and therapies...' should be the last activity after everything else is done and before the end activity.

2) Activities related to stroke units, staffing, or hospitals in the QBP should be prioritized as the initial steps in their respective flow or branch.

3) All tests, including any imaging and blood tests, should be immediately interpreted by a healthcare expert.

4) Blood tests or any tests that take a long time to get results should occur concurrently with other activities unless required for a specific task, in which case they will be merged into the flow or branch of that activity.

Expanded Knowledge Context (v3.5–v3.7)

Below are additional knowledge contexts to guide the process design:

1) The activity 'Provide cross-continuum secondary prevention assessments and therapies...' should be the last activity after everything else is done and before the end activity.

2) Activities related to stroke units, staffing, or hospitals in the QBP should be prioritized as the initial steps in their respective flow or branch.

3) All tests, including any type of imaging and blood tests, should be immediately interpreted by a healthcare expert.

4) Blood tests or any tests that take a long time to get results should occur concurrently with other activities unless required for a specific task, in which case they will be merged into the flow or branch of that activity.

5) Activities related to stroke units, staffing, or hospitals in the QBP should be put in a single activity.

6) Assessing airway, breathing, and circulation is called ABC testing and is ONE activity.

7) Neurological examination and imaging results are needed to Evaluate patients for endovascular therapy. However, they can be in parallel to other activities that don't need these results.

8) CBC, electrolytes, creatinine, glucose, INR, partial thromboplastin time, and troponin test are all types of blood tests, and they can all be in ONE blood works activity if they are at the same place of a process.

9) Anti platelet therapy has 4 general options (ASA, Clopidogrel, Extended-release Dipyridamole + Aspirin (Aggrenox), and rectal or gastric ASA if can't swallow) and they should be indicated right after each initiation of antiplatelet therapy activity in the process.

10) Temperature reducing activities should happen in parallel.

11) The activity "chest x-ray should not delay..." should have a skip flow as it should be skipped if not needed.

12) If activity "should be evaluated in the ED to be considered for admission to hospital" is needed, it should happen in parallel and not in sequence.

13) When patients should be screened/assessed for diabetes, dental issues, cognitive impairment, and depression in the same part of the process, they should be merged into one activity along with forwarding the results.

14) Assessments related and needed to formulating management plan should be merged into one activity along with formulating the management plan.

15) Formulating the management plan should happen before the activities and assessments after it.

16) Between 24-48 hour of stroke onset IF no contradiction, mobilization is encouraged. However, in the first 24 hour it is only encouraged IF clinical judgement suggests otherwise, it is not.

Explicit Knowledge Context (v3.8)

Below are additional knowledge contexts to guide the process design:

1) The activity 'Provide cross-continuum secondary prevention assessments and therapies...' should be the last activity after everything else is done and before the end activity.

2) Activities related to stroke units, staffing, or hospitals in the QBP should be prioritized as the initial steps in their respective flow or branch as they are prerequisites for other activities.

3) All tests, including any type of imaging and blood tests, should be immediately interpreted (followed) by a healthcare expert.

4) Blood tests or any tests that take some time to get results should occur concurrently with other activities to allow faster treatment unless the test results are required for a specific task, in which case they will be merged into the flow or branch of that activity as they are now prerequisites.

5) Activities related to stroke units, staffing, or hospitals in the QBP should be put in a single activity.

6) Assessing airway, breathing, and circulation is called ABC testing and is ONE activity not multiple activities.

7) Neurological examination and imaging results are both needed (prerequisites) to Evaluate patients for endovascular therapy and should happen before this. However, they can be in parallel to other activities of the pathway that don't require these results to allow efficient time management.

8) CBC, electrolytes, creatinine, glucose, INR, partial thromboplastin time, and troponin test are all types of blood tests, and they all should be merged in ONE blood works activity if they are at happening in the same place of the process.

9) Anti platelet therapy has 4 general options (ASA, Clopidogrel, Extended-release Dipyridamole + Aspirin (Aggrenox), and rectal or gastric ASA if can't swallow) and they always should be indicated right after each initiation of antiplatelet therapy activity in the process.

10) Different Temperature reducing activities (options) should happen concurrently (in regard to each other) to reduce the the temperature as fast as possible and continue care.

11) The activity "chest x-ray should not delay..." means IF chest x-ray is required it should not delay assessment and IF not required it should be skipped and continue.

12) If the evaluation in the ED to be considered for admission to hospital is needed, it should happen in parallel (concurrent) to everything else and not in sequence.

13) When patients should be screened/assessed for diabetes, dental issues, cognitive impairment, and depression in the same part of the process, they should be merged into ONE activity along with forwarding the results.

14) Assessments related and needed in order to formulate a management plan should be merged into one activity along with formulating the management plan.

15) Formulating the management plan is a prerequisite for all subsequent activities and assessments.

16) Mobilization should be considered only if clinical judgment supports it within the first 24 hours of stroke onset; between 24–48 hours, proceed with mobilization if no contraindications are identified, otherwise skip and continue with the care pathway.

C.2 Knowledge Advisor Agent Prompt (verbatim)

You're a Knowledge Graph Advisor. Now, we share a common interest in collaborating to successfully complete a task assigned by a new customer.
You are familiar with exclusive gateway, parallel gateway and inclusive gateway in BPMN. Your job is to review process models and provide actionable suggestions for improvement, one at a time. You use extra knowledge provided to you to inform your suggestions. Your main responsibilities include giving suggestions and your reasoning to the process design expert.
Here is a new customer's task: Module 2: Early Treatment
This module identifies best practices for the ED treatment of TIA or minor (nondisabling) stroke. The
majority of TIA patients and some patients with minor stroke do not require admission to hospital and should be referred to an urgent TIA/minor stroke unit/TIA clinic/stroke-prevention clinic or

comparable ambulatory care setting for rapid diagnostic and medical evaluation. All patients should
be given appropriate cross-continuum prevention assessment and therapies (Modules 5 and 10).
Module 2 Recommended Practices
2.1 Glucose management
2.1.1 Hypoglycemia should be corrected immediately
2.1.2 Blood glucose measurement should be repeated if the first
random glucose value is $>$ 11 mmol/L
2.2 Body temperature
2.2.1 Temperature should be evaluated as part of routine vital signs
every 4 hours for first 48 hours
For temperature $>$ 37.5 C:
• initiate temperature-reducing measures
• investigate potential infection
• initiate antipyretic and antimicrobial therapy as required
2.3 Cross-continuum prevention assessment and therapies
2.3.1 All patients, whether admitted to hospital or discharged, should
be given appropriate cross-continuum secondary prevention
assessments and therapies (Modules 5 and 10)
2.4 Management of patients with TIA or minor (nondisabling) stroke
2.4.1 The following OHTAC recommendations should be followed:
• OHTAC recommends that patients presenting with a TIA with high-
risk featuresa or a minor strokeb undergo a brain CT scan and
initiation of antiplatelet therapy (provided this is not contraindicated)
as soon as possible and no later than 24 hours after symptom onset,
followed by other stroke-prevention treatments tailored to each
patient. With respect to the location of care, OHTAC recommends
that:
o such immediate care be provided at a specialized TIA/minor
stroke clinicc
o where delays to accessing a specialized TIA/minor stroke clinic
pose risks to patient health, evaluation (as outlined above) occur
at an appropriately resourced ED, and further consideration be
given to inpatient evaluation and management for stroke
prevention. OHTAC further recommends the establishment of

accreditation standards for TIA/minor stroke care to ensure equitable access to appropriate, high-quality care, irrespective of the location of initial presentation

o where medical attention has been sought after 48 hours from symptom onset, patients be referred for evaluation at a specialized TIA/minor stroke clinic or alternatively an outpatient clinic with stroke-prevention servicesd within 24 hours of initial presentation

• OHTAC recommends that patients presenting with a TIA without high-risk featuresa undergo a brain CT scan and initiation of antiplatelet therapy (provided this is not contraindicated) as soon as possible and no later than 24 hours after initial presentation, followed by referral to an outpatient clinic with stroke-prevention servicesd for comprehensive evaluation and management within 1 month of symptom onset

2.4.2 Some TIA patients, such as those who present with fluctuating or crescendo hemibodye motor weakness, sensory loss, or speech difficulty, should be evaluated in the ED to be considered for admission to hospital

2.4.3 Patients with TIA or minor (nondisabling) stroke who are not on an antiplatelet agent at the time of presentation should be started on antiplatelet therapy immediately with 1 of the following after brain imaging has excluded intracranial hemorrhage:

• ASA 160 mg loading dose, followed by enteric coated ASA (81–325 mg) daily. Most patients should be on a maintenance dose of 81 mg/day

• clopidogrel 300 mg loading dose, followed by 75 mg/day

• extended-release dipyridamole 200 mg/ASA 25 mg bid (load with ASA 160 mg first)

2.4.4 Rectal or gastric ASA should be offered as necessary, such as for patients who fail swallowing screening

2.5–2.8 Recommendations are not applicable to TIA or minor (nondisabling) stroke

Abbreviations: AHA/ASA, American Heart Association/American Stroke Association; ASA, acetylsalicylic acid; Australia, Australian Clinical analysis; ED, emergency department; HQO, Health Quality Ontario; OHTAC, Ontario Health Technology Advisory Committee; NHS/NICE,

National Collaborating Centre for Chronic Conditions; SIGN, Scottish Intercollegiate Guidelines Network; TIA, transient ischemic attack.
aSymptoms consistent with a hemispheric event, including sudden hemiparesis, speech difficulties, or monocular vision loss, and/or known high
risk conditions associated with stroke, including atrial fibrillation (especially if inadequately anticoagulated) or known carotid artery
atherosclerosis with > 50% stenosis (narrowing) on the side consistent with the hemispheric event.
bPatients with very mild persistent symptoms or no residual symptoms but with a small asymptomatic infarct (stroke) on imaging
cA clinic with stroke expertise and the resources to conduct all necessary investigations in 1 place to initiate rapid treatment, including brain and
vascular imaging, heart monitoring, and laboratory tests.
dFor example, a provincial stroke-prevention clinic or community neurology/internal medicine clinic with a stroke-prevention focus.
eMotor weakness may be in 1 body segment (face, arm, or leg), but sensory loss must involve at least 2 contiguous segments (face/arm or
arm/leg) on 1 side of the body to be considered highest risk.

The following implementation considerations were noted by members of the expert advisory panel.
Module 2 Implementation Considerations
General considerations
• Standardized priority protocols should be established for all suspected stroke patients to receive treatment as soon as
possible and be admitted to a stroke unit within the first few hours after presenting to an ED
• Hospitals should have ready access to rectal ASA
Abbreviations: ASA, acetylsalicylic acid; ED, emergency department..]

Current process text:
<process>
<activity role='ED medical staff' action='Obtain patient history' objects='patient'
id='activity1-1'/>
<activity role='ED medical staff' action='Perform physical examination' objects='patient'
id='activity1-2'/>
<activity role='ED medical staff' action='Review presenting symptoms' objects='patient'
id='activity1-3'/>

```
<activity role='ED medical staff' action='Measure blood glucose' objects='patient'
id='activity2'/>
<exclusiveGateway id='exclusivegateway1'>
  <branch condition='If hypoglycemia detected' id='branch2'>
    <activity role='ED medical staff' action='Correct hypoglycemia immediately'
    objects='patient' id='activity3'/>
  </branch>
  <branch condition='If blood glucose > 11 mmol/L' id='branch3'>
    <activity role='ED medical staff' action='Repeat blood glucose measurement'
    objects='patient' id='activity4'/>
  </branch>
  <branch condition='If blood glucose normal (not hypoglycemic and <= 11 mmol/L)'
  id='branch4'>
    <activity role='ED medical staff' action='Continue standard care' objects='patient'
    id='activity5'/>
  </branch>
</exclusiveGateway>
<activity role='ED medical staff' action='Evaluate body temperature as part of routine vital
signs every 4 hours for first 48 hours' objects='patient' id='activity6'/>
<exclusiveGateway id='exclusivegateway2'>
  <branch condition='If temperature > 37.5°C' id='branch6'>
    <parallelGateway id='parallelgateway2'>
      <branch id='branch7'>
        <activity role='ED medical staff' action='Initiate temperature-reducing measures'
        objects='patient' id='activity7'/>
      </branch>
      <branch id='branch8'>
        <parallelGateway id='parallelgateway3'>
          <branch id='branch8-1'>
            <activity role='ED medical staff' action='Order infection-related laboratory tests'
            objects='patient' id='activity8-1'/>
          </branch>
          <branch id='branch8-2'>
            <activity role='ED medical staff' action='Assess for clinical signs of infection'
            objects='patient' id='activity8-2'/>
```

```
        </branch>
      </parallelGateway>
    </branch>
    <branch id='branch9'>
      <activity role='ED medical staff' action='Initiate antipyretic and antimicrobial therapy
      as required' objects='patient' id='activity9'/>
    </branch>
  </parallelGateway>
 </branch>
 <branch condition='If temperature <= 37.5°C' id='branch10'>
  <activity role='ED medical staff' action='Continue routine monitoring' objects='patient'
  id='activity10'/>
 </branch>
</exclusiveGateway>
<activity role='ED medical staff' action='Assess for high-risk features or minor stroke per
OHTAC criteria' objects='patient' id='activity12'/>
<parallelGateway id='parallelgateway4'>
 <branch id='branch-parallel1'>
  <exclusiveGateway id='exclusivegateway3'>
   <branch condition='TIA with high-risk features or minor stroke' id='branch13'>
    <activity role='ED medical staff' action='Order brain CT scan' objects='patient'
    id='activity13-1'/>
    <activity role='ED medical staff' action='Review CT scan results' objects='patient'
    id='activity13-2'/>
    <activity role='ED medical staff' action='Initiate antiplatelet therapy if no
    contraindication' objects='patient' id='activity13-3'/>
    <exclusiveGateway id='exclusivegateway4'>
     <branch condition='Immediate care available at specialized TIA/minor stroke clinic'
     id='branch14'>
      <activity role='ED medical staff' action='Provide immediate care at specialized
      TIA/minor stroke clinic' objects='patient' id='activity14'/>
     </branch>
     <branch condition='Delay to specialized clinic poses risk' id='branch15'>
```

```
      <activity role='ED medical staff' action='Perform evaluation at appropriately
      resourced ED and consider inpatient evaluation for stroke prevention'
      objects='patient' id='activity15'/>
    </branch>
    <branch condition='Medical attention sought after 48 hours from symptom onset'
    id='branch16'>
      <activity role='ED medical staff' action='Refer for evaluation at specialized
      TIA/minor stroke clinic or outpatient clinic with stroke-prevention services within 24
      hours of initial presentation' objects='patient' id='activity16'/>
    </branch>
  </exclusiveGateway>
  <inclusiveGateway id='inclusivegateway1a'>
    <branch condition='Patient presents with fluctuating or crescendo hemibody motor
    weakness, sensory loss, or speech difficulty' id='branch18a'>
      <activity role='ED medical staff' action='Evaluate in ED for consideration of hospital
      admission' objects='patient' id='activity19a'/>
    </branch>
    <branch condition='Patient not on antiplatelet agent at presentation and brain imaging
    excludes intracranial hemorrhage' id='branch19a'>
      <exclusiveGateway id='exclusivegateway5a'>
        <branch condition='Patient can swallow' id='branch20a'>
          <activity role='ED medical staff' action='Start antiplatelet therapy immediately
          (choose one: ASA 160 mg loading dose then 81–325 mg daily, clopidogrel 300 mg
          loading then 75 mg daily, or extended-release dipyridamole 200 mg/ASA 25 mg bid
          after ASA 160 mg loading)' objects='patient' id='activity20a'/>
        </branch>
        <branch condition='Patient cannot swallow' id='branch21a'>
          <activity role='ED medical staff' action='Offer rectal or gastric ASA as necessary'
          objects='patient' id='activity21a'/>
        </branch>
      </exclusiveGateway>
    </branch>
  </inclusiveGateway>
</branch>
<branch condition='TIA without high-risk features' id='branch17'>
```

<activity role='ED medical staff' action='Order brain CT scan' objects='patient' id='activity17-1'/>
<activity role='ED medical staff' action='Review CT scan results' objects='patient' id='activity17-2'/>
<activity role='ED medical staff' action='Initiate antiplatelet therapy if no contraindication' objects='patient' id='activity17-3'/>
<activity role='ED medical staff' action='Refer to outpatient clinic with stroke-prevention services for evaluation and management within 1 month of symptom onset' objects='patient' id='activity18'/>
<inclusiveGateway id='inclusivegateway1b'>
<branch condition='Patient presents with fluctuating or crescendo hemibody motor weakness, sensory loss, or speech difficulty' id='branch18b'>
<activity role='ED medical staff' action='Evaluate in ED for consideration of hospital admission' objects='patient' id='activity19b'/>
</branch>
<branch condition='Patient not on antiplatelet agent at presentation and brain imaging excludes intracranial hemorrhage' id='branch19b'>
<exclusiveGateway id='exclusivegateway5b'>
<branch condition='Patient can swallow' id='branch20b'>
<activity role='ED medical staff' action='Start antiplatelet therapy immediately (choose one: ASA 160 mg loading dose then 81–325 mg daily, clopidogrel 300 mg loading then 75 mg daily, or extended-release dipyridamole 200 mg/ASA 25 mg bid after ASA 160 mg loading)' objects='patient' id='activity20b'/>
</branch>
<branch condition='Patient cannot swallow' id='branch21b'>
<activity role='ED medical staff' action='Offer rectal or gastric ASA as necessary' objects='patient' id='activity21b'/>
</branch>
</exclusiveGateway}
</branch}
</inclusiveGateway}
</branch}
</exclusiveGateway}
</branch}
<branch id='branch-parallel2'>

<activity role='ED medical staff' action='Conduct secondary prevention assessment' objects='patient' id='activity11-1'/>
<activity role='ED medical staff' action='Initiate secondary prevention therapy' objects='patient' id='activity11-2'/>
</branch}
</parallelGateway}
</process}

Extra knowledge:

1) The activity 'Provide cross-continuum secondary prevention assessments and therapies...' should be the last activity after everything else is done and before the end activity.

2) Activities related to stroke units, staffing, or hospitals in the QBP should be prioritized as the initial steps in their respective flow or branch as they are prerequisites for other activities.

3) All tests, including any type of imaging and blood tests, should be immediately interpreted (followed) by a healthcare expert.

4) Blood tests or any tests that take some time to get results should occur concurrently with other activities to allow faster treatment unless the test results are required for a specific task, in which case they will be merged into the flow or branch of that activity as they are now prerequisites.

5) Activities related to stroke units, staffing, or hospitals in the QBP should be put in a single activity.

6) Assessing airway, breathing, and circulation is called ABC testing and is ONE activity not multiple activities.

7) Neurological examination and imaging results are both needed (prerequisites) to Evaluate patients for endovascular therapy and should happen before this. However, they can be in parallel to other activities of the pathway that don't require these results to allow efficient time management.

8) CBC, electrolytes, creatinine, glucose, INR, partial thromboplastin time, and troponin test are all types of blood tests, and they all should be merged in ONE blood works activity if they are at happening in the same place of the process.

9) Anti platelet therapy has 4 general options (ASA, Clopidogrel, Extended-release Dipyridamole + Aspirin (Aggrenox), and rectal or gastric ASA if can't swallow) and they always should be indicated right after each initiation of antiplatelet therapy activity in the process.

10) Different Temperature reducing activities (options) should happen concurrently (in regard to each other) to reduce the the temperature as fast as possible and continue care.

11) The activity "chest x-ray should not delay..." means IF chest x-ray is required it should not delay assessment and IF not required it should be skipped and continue.

12) If the evaluation in the ED to be considered for admission to hospital is needed, it should happen in parallel (concurrent) to everything else and not in sequence.
13) When patients should be screened/assessed for diabetes, dental issues, cognitive impairment, and depression in the same part of the process, they should be merged into ONE activity along with forwarding the results.
14) Assessments related and needed in order to formulate a management plan should be merged into one activity along with formulating the management plan.
15) Formulating the management plan is a prerequisite for all subsequent activities and assessments.
16) Mobilization should be considered only if clinical judgment supports it within the first 24 hours of stroke onset; between 24–48 hours, proceed with mobilization if no contraindications are identified, otherwise skip and continue with the care pathway.

<end of extra knowledge>

When making recommendations, don't forget the format of the process text:

When the process starts, please output '<process>'
An activity is represented as follows:
<activity role='' action='' objects='' id=''/>
The exclusive gateway is represented as follows:
<exclusiveGateway id=''>
  <branch condition='' id=''>
    <activity role='' action='' objects='' id=''/>
  </branch>
  <branch condition='' id=''>
    <activity role='' action='' objects='' id=''/>
  </branch>
</exclusiveGateway>
The parallel gateway is represented as follows:
<parallelGateway id=''>
  <branch id=''>
    <activity role='' action='' objects='' id=''/>
  </branch>
  <branch id=''>

<activity role='' action='' objects='' id=''/>
</branch>
</parallelGateway}
The inclusive gateway is represented as follows:
<inclusiveGateway id=''>
<branch condition='' id=''>
<activity role='' action='' objects='' id=''/>
</branch>
<branch condition='' id=''>
<activity role='' action='' objects='' id=''/>
</branch>
</inclusiveGateway}
At the end of the process, please output '</process}'

There are a few requirements that should be noted when generating process text:

1) role, action, objects and id in an activity represent the subject of the activity, the activity behavior, the items needed to perform in the activity and the activity id, respectively.

2) There are at least two branches within the exclusive gateway and the parallel gateway.

3) Activity, gateway, and branch id attributes must be unique.

4) When the branch in the exclusive gateway has a condition, fill the condition into the condition attribute. When the branch unconditional within the exclusive gateway, the condition attribute of the branch =''. However, all branches within a exclusive gateway can only be conditional, or both unconditional.

5) The branch in the parallel gateway has no condition attribute.

6) Each branch in the inclusive gateway must have condition.

7) There must be at least one activity inside the branch tag.

8) The activity of a sequential relation need not be represented by a gateway, that is, there is no 'sequenceGateway' label.

Format your suggestion in the following format:
Suggested modification: <your suggestion>
Reason: <your reasoning>

Based on the above extra knowledge, provide ONE actionable suggestion along with your reasoning to improve the process. If there are no further suggestions, just reply with: 'There are no further suggestions.

Appendix D

Baseline Experiment: Expert Evaluation Template

This appendix provides the expert evaluation template used in the evaluation method of Section 5.2.2.

Evaluation of LLM Output — Instructions

For each QBP and LLM framework, please answer the following:

1. **Is the knowledge represented by the extracted process model equivalent to the textual description?**
2. **Score:** (5 = Fully Equivalent — 1 = Fully Non-Equivalent)
3. **Please explain** which elements of the process model are not equivalent to the textual description, and why.
4. **Any other comments** (optional).

See the following pages for the templates to be filled out.

QBP 1: Recommended Practices for TIA or Minor (Nondisabling) Stroke: Early Assessment (pp. 44–45)

MAO output 1

1) Score (circle/mark one): 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

(5 = Fully Equivalent — 1 = Fully Non-Equivalent)

2) Please explain which elements of the process model are not equivalent to the textual description, and why.

3) Any other comments (optional).

ProMoAI output 1

1) Score (circle/mark one): 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

(5 = Fully Equivalent — 1 = Fully Non-Equivalent)

2) Please explain which elements of the process model are not equivalent to the textual description, and why.

3) Any other comments (optional).

Appendix E

Manual Evaluation Results

This appendix provides the detailed expert evaluation output discussed in Section 6.2.1.

E.1 Knowledge-enhanced vs Baseline

These are the expert evaluation results for the knowledge-enhanced vs Baseline experiment discussed in Section 6.2.1.

IG 1 ($\Delta_{\mathbf{v3.6}} = +2.0$, $\Delta_{\mathbf{v3.7}} = +1.0$):

- **MAO v3.6:**
  - Chest X-ray skip-flow absent (0)
  - "Secondary prevention" correctly placed (+1)
  - Blood-work activities merged into one (+1)
- **MAO v3.7:**
  - Swallowing assessments delayed (–1)
  - Chest X-ray skip-flow absent (0)
  - "Secondary prevention" correctly placed (+1)
  - Blood-work activities merged (+1)

IG 2 ($\Delta_{\mathbf{v3.6}} = +5.0$, $\Delta_{\mathbf{v3.7}} = +1.5$):

- **MAO v3.6:**
  - "Secondary prevention" correctly ordered (+1)
  - Temperature-reduction steps in parallel (+1)
  - Antiplatelet options present on every branch (+1)
  - Staffing mis-sequenced (0)
  - Conditioning for antiplatelet initiation improved (+1)
  - "Should be evaluated in ED" correctly placed (+1)
- **MAO v3.7:**
  - "Secondary prevention" misplaced (0)
  - Temperature steps sequential (0)
  - Missing antiplatelet options on "no initiation" branch (–0.5)
  - Staffing sequence unchanged (0)
  - Antiplatelet conditioning improved (+1)
  - ED-evaluation placement correct (+1)

IG 3 ($\Delta_{\mathbf{v3.6}} = +8.0$, $\Delta_{\mathbf{v3.7}} = +6.0$):

- **MAO v3.6:** Retains all seven baseline positives, plus correct placement of merged diabetes-screening activity (+1).
- **MAO v3.7:** Retains six positives but misplaces the diabetes-screening activity (–1).

IG 4 ($\Delta_{\mathbf{v3.6}} = +4.5$, $\Delta_{\mathbf{v3.7}} = +1.0$):

- **MAO v3.6:**
  - "Consider for endovascular" omitted (–1)

- Chest X-ray, blood-work, and ECG in parallel (+3)
- ABC testing and blood-work merged (+2)
- Chest X-ray conditionality preserved (+0.5)
- Anticoagulant-history condition present (+1)
- Vascular imaging remains sequential (–1)

- **MAO v3.7:**
  - Endovascular condition only in label (–0.5)
  - Brain-imaging precedes endovascular decision (+1)
  - Missing skip-flow after NPO (–1)
  - Diabetes-assessment condition overly broad (–1)
  - ABC testing and blood-work merged (+2)
  - Chest X-ray conditionality preserved (+0.5)
  - Anticoagulant-history condition present (+1)
  - Vascular imaging remains sequential (–1)

E.2 Structure of Knowledge Statements

These are the expert evaluation results for the Structure of Knowledge Statements experiment discussed in Section 6.2.1.

IG 1 ($\Delta_{\text{v3.8 vs 3.6}} = 0.0$, $\Delta_{\text{v3.8 vs 3.7}} = +1.0$):

- **vs v3.6:**
  - Chest X-ray does not have skip-flow (0).
- **vs v3.7:**
  - Chest X-ray does not have skip-flow (0).
  - Monitoring activities in correct order (+1).

IG 2 ($\Delta_{\text{v3.8 vs 3.6}} = -1.0$, $\Delta_{\text{v3.8 vs 3.7}} = +2.0$):

- **vs v3.6:**
  - Sequential temperature-reduction measures (Slight Negative, -0.5).
  - Slightly worse antiplatelet therapy options (Slight Negative, -0.5).
- **vs v3.7:**
  - Sequential temperature-reduction measures (0).
  - Antiplatelet therapy options (0).
  - "Formulating the management plan" precedes other activities ($+1$).
  - "Secondary prevention" correctly placed ($+1$).

IG 3 ($\Delta_{\text{v3.8 vs 3.6}} = +1.5$, $\Delta_{\text{v3.8 vs 3.7}} = +1.5$):

- **vs v3.6:**
  - Patient-mobilization conditions improved but not 100% correct (Slight Positive, $+0.5$).
  - "Formulating the management plan" happens before other activities ($+1$).
- **vs v3.7:**
  - Patient-mobilization conditions improved but not 100% correct (Slight Positive, $+0.5$).
  - "Secondary prevention" activity correctly placed ($+1$).

IG 4 ($\Delta_{\text{v3.8 vs 3.6}} = +0.5$, $\Delta_{\text{v3.8 vs 3.7}} = +1.5$):

- **vs v3.6:**
  - Endovascular-therapy conditions improved but not 100% correct (Slight Positive, $+0.5$).

  - Chest X-ray does not have skip-flow (0).

- **vs v3.7:**
  - Endovascular-therapy conditions improved but not 100% correct (Slight Positive, +0.5).
  - Chest X-ray does not have skip-flow (0).
  - Vascular-therapy condition correct (+1).