Agentic Generation of Structured Clinical Specifications for Digital Healthcare Services

Bruno Guindani, Matteo Camilli, Livia Lestingi, Marcello Maria Bersani
name.surname@polimi.it
Politecnico di Milano
Milan, Italy

ABSTRACT

Clinical guidelines are essential for standardized and evidence-based patient care, yet their narrative, unstructured form often leads to inconsistencies and interpretation errors. As emphasized by the WHO, there is a need for machine-readable specifications to support clinical decision-making, but these are currently developed by hand, making the process error-prone. We present MARACTUS, an agentic methodology for converting unstructured healthcare documents into structured Machine-readable Action Trees (MATs) that represent clinical procedures. The produced MATs can feed model-driven engineering pipelines, enable formal verification, and integrate into clinical decision support systems, maximizing their impact on safer, more transparent, and continuously updatable healthcare software. MARACTUS combines an input constructor, a Large Language Model (LLM), and a syntax checker within an orchestration framework, ensuring adherence to a fixed grammar while maintaining clinician oversight via a curated knowledge base. We validate MARACTUS using two exemplar clinical cases and two state-of-the-art LLMs, comparing zero-shot and few-shot prompting strategies across subsets of documents. Our evaluation demonstrates a syntactic correctness rate of 97.5% and a clinical success rate of 62.5% relative to human ground truth, while domain experts rated over half of the MATs as clinically valid (with scores ≥ 4/5) across multiple evaluation dimensions.

CCS CONCEPTS

• **Applied computing → Health informatics**.

KEYWORDS

Medical Guidelines, Digital Healthcare, Machine-readable Artifacts, Large Language Models, Agentic AI

ACM Reference Format:
Bruno Guindani, Matteo Camilli, Livia Lestingi, Marcello Maria Bersani. 2026. Agentic Generation of Structured Clinical Specifications for Digital Healthcare Services. In *2026 IEEE/ACM 48th International Conference on Software Engineering (ICSE-SEIS '26), April 12–18, 2026, Rio de Janeiro, Brazil.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3786581.3786943

1 INTRODUCTION

Inefficiencies and inaccuracies in diagnoses, patient treatment, and clinical practice critically hinder the quality of patient care in the public health sector [40]. Paper-based clinical guidelines are a valuable collection of medical evidence and expert knowledge, which are instrumental in training staff effectively and making informed decisions in clinical contexts [36]. However, being written in Natural Language (NL) in a non-standardized way, clinical guidelines are intrinsically subject to human interpretation and potentially ambiguous [19, 31]. As a result, two practitioners may reach different conclusions despite relying on the same document. Moreover, the patient at hand may suffer from a specific combination of medical conditions that the guidelines may not exhaustively cover. Perfect adherence to clinical guidelines is often unfeasible, conflicting with their goal of standardizing medical practice.

The World Health Organization (WHO) has drawn attention to this issue by developing standards-based, machine-readable, adaptive, requirements-based, and testable (SMART) guidelines to operationalize medical practice starting from narrative documentation [37]. SMART guidelines envision 5 digitalization levels: (L1) paper-based documentation, (L2) semi-structured functional requirements elicited from documentation, (L3) machine-readable specifications, (L4) software implementing static algorithms, and (L5) dynamic algorithms optimized with advanced analytics.

Advancements in software engineering, edge, and cloud computing are fueling the digitalization of the public sector. *Medical informatics*, which lies at the intersection of computer science, social science, and healthcare practices [24], is the backbone of recent progress in patient data management, staff training, clinical decision making, and numerous services in the public health area. However, despite the potential beneficial impact on society, the SMART guidelines are hardly incorporated into end-to-end software engineering pipelines for healthcare services.

Several factors hamper this change. Firstly, the production of L2 and L3 artifacts is currently done by hand [10, 52]. The manual creation of clinical decision trees requires significant practitioner effort, is error-prone like all manual processes, and may mirror the author's subjective perspective and experience. Data-driven approaches suffer from limited field-collected data on underrepresented conditions and regions, which may result in L3 artifacts that perpetuate societal biases and disparities [9, 63].

Our work introduces a methodology to engineer machine-readable guidelines into healthcare services. The methodology, called MARACTUS[1], leverages, at its core, Large Language Models (LLMs). LLMs are intrinsically designed to process NL inputs, and their ability to generate structured outputs from unstructured or semi-structured inputs is currently being investigated [29]. In this work, general-purpose LLMs are instructed to process a collection of unstructured healthcare documents (e.g., official guidelines from

[1] MAchine-Readable ACtion Trees from Unstructured Specifications

ICSE-SEIS '26, April 12–18, 2026, Rio de Janeiro, Brazil

ACM ISBN 979-8-4007-2424-4/2026/04
https://doi.org/10.1145/3786581.3786943

healthcare organizations), all focusing on a specific medical condition, and then produce a structured machine-readable specification of the procedure to treat such condition. Specifically, we instruct LLMs to structure the output as a Machine-readable Action Tree (MAT), amenable to model-to-model transformation into several well-known target formalisms (e.g., finite-state automata).

To this end, MARACTUS adopts an *agentic* approach [62] where an *input constructor*, an LLM, and a *syntax checker* collaborate through an *orchestration layer*. The collection of NL documents, together with predefined clinical case-MAT pairs (i.e., the *shots*), constitutes a *knowledge base* maintained by domain experts. By doing so, clinicians remain central by curating the knowledge base, ensuring automation relieves them of manual repetitive tasks and emphasizes their expertise rather than replacing them. The *input constructor* relies on the body of knowledge to craft the prompt according to a selected prompting strategy. Given the lack of assurance on LLMs' ability to comply with a fixed grammar, every LLM-generated artifact is processed by a deterministic *syntax checker* before proceeding through the pipeline.

In software engineering processes, artifacts derived from clinical guidelines can help analysts elicit requirements of (healthcare) software [33], and they serve as a snapshot of the operational context in which the application is expected to be used, highlighting the alignment between technical specifications and real-world clinical scenarios. They facilitate automation of model-driven engineering pipelines [7]: artifacts can, in fact, be transformed into executable models for testing, debugging, or analysis purposes; for instance, they can feed simulation environments or can be integrated into clinical decision-support systems with limited manual intervention. Machine readable artifacts also enable formal verification and validation, enabling the automated analysis of healthcare software or medical devices against key properties such as functional correctness and application responsiveness, or to perform quantitative assessments of clinical scenarios [23, 44, 59]. Finally, structured machine-readable guidelines allow continuous integration and evolution of healthcare systems, easing adaptation to updated medical evidence and reducing maintenance costs.

By providing machine-readable, clinically grounded representations of medical procedures, our methodology has the potential to improve patient safety, enhance the transparency of automated systems, and increase trust in Artificial Intelligence (AI)-assisted care. Beyond direct patient impact, it supports medical education, training, and regulatory oversight, ultimately contributing to more reliable and equitable healthcare delivery.

We validate MARACTUS through two exemplar clinical cases and two state-of-the-art LLMs. The experimental campaign compares two prompting strategies, zero-shot and few-shot learning, each tested with different subsets of NL documents from the knowledge base. The experimental campaign investigates several aspects, including the LLMs' ability to conform to MAT grammar, the clinical validity of LLM-generated procedures, and the similarity of their effects on patients compared to a human-produced ground truth. Our quantitative evaluation yields a syntactic correctness rate of 97.5% and a clinical success rate of 62.5% relative to the human ground truth. Additionally, we conducted a qualitative user study with domain experts in anesthesia and intensive care, who assessed the clinical validity of the generated MATs. The results show that over half of the MATs received a validity score of at least 4 on a 1–5 scale across multiple evaluation dimensions.

The paper is structured as follows: Section 2 outlines preliminary concepts underpinning the work, Section 3 presents MARACTUS in detail, Section 4 reports on the experimental results, Section 5 elaborates on the potential impact of this research on society, Section 6 surveys related work, and Section 7 concludes.

2 PRELIMINARIES

2.1 Large Language Models

LLMs are probabilistic generative models of text based on Deep Neural Networks trained on massive amounts of data. State-of-the-art LLMs are either closed-access (commercial), including models with hundreds or thousands of billions of parameters like GPT [39] and Gemini [4], or open-access (community), including models with few or tens of billions of parameters such as Llama [57].

Fine-tuning is generally used to turn the LLM into an *instruction-following* agent [50] or a *chatbot-assistant* [51]. Chatbot-assistant fine-tuning adapts the model to interact with a user in conversational settings, making it suitable for applications like virtual assistants. During training, the model weights are updated based on the likelihood of generating the target response given an input sequence composed of: (1) the *system message* (initial task instructions in a specific chatbot-assistant use case); and (2) a *user prompt* (user's question or request in a specific chatbot-assistant use case).

2.2 Agentic AI

Agentic AI refers to systems that extend the capabilities of LLMs by endowing them with autonomy, goal-directed behavior, and the ability to act upon the external world through tools and structured reasoning frameworks [62].

The architecture of an agent is typically organized around three foundational components: the *model*, the *tools*, and the *orchestration layer*. The model acts as the central reasoning engine, usually instantiated as one or more LLMs capable of following logic-oriented prompting strategies. Tools bridge the gap between the agent and the external world by enabling interaction with data sources and services, such as APIs or information retrieval systems. Finally, the orchestration layer governs the agent's cognitive architecture by structuring the cyclical process of information intake, reasoning, action selection, and refinement until a goal is achieved. A key distinction between LLMs and agentic systems lies in their operational scope. Whereas an LLM passively generates a single output based on its input context, an agent maintains session history, performs multi-turn reasoning, and dynamically selects tools.

3 METHODOLOGY

The key challenges addressed by MARACTUS are the ambiguity and lack of structure in clinical procedures described in guidelines from health organizations and medical literature. While clinical care is inherently human-centered, often involving nuanced and context-dependent decision-making, software supporting medical staff requires precise instructions that such documents typically do not provide [17, 65]. We address this challenge by presenting a methodology to extract *Machine-readable Action Trees (MATs)* from a body of reliable knowledge. An MAT is a structured representation of a segment of a clinical procedure, encoding actions performed by medical staff under specific conditions derived from patient vital signs. Unlike unstructured guidelines such as NL, MATs capture clinical workflows in an unambiguous machine-readable format. This design makes MATs particularly well-suited for automated processing, including model-to-model transformations that facilitate monitoring and verification. More broadly, the ability to extract clinically accurate MATs opens up a wide spectrum of applications, ranging from straightforward visualization aids to advanced systems that directly support clinical decision-making.

3.1 Procedural Model

MARACTUS is an automated iterative methodology for extracting clinical procedures for specific use cases from a body of knowledge. We provide a rigorous definition of procedure and MAT as follows.

Definition 3.1 (Clinical procedure)**.** Given a time-variant system characterized by a set of controllable parameters ϑ and a set of observed metrics x, both belonging to (potentially infinite) domains, a *clinical procedure* is defined as a tree structure with *actions* in each node and *conditions* on each edge. For each node i, the *action* associated with i is a rule (function) that updates the parameters according to the measured system's state: $\vartheta \leftarrow f_i(x, \vartheta)$. For each edge j, the *condition* associated with j is a predicate that applies to the system's current state: $P_j(x, \vartheta)$. Conditions on edges leaving the same node are mutually exclusive.

Definition 3.2 (Machine-readable Action Tree)**.** A *Machine-readable Action Tree (MAT)* is a formal artifact encoding the tree structure of a procedure, as defined in Definition 3.1, whose syntax follows a specified grammar (see Appendix A).

A (partial) *path* of a procedure is a sequence of $k \geq 0$ nodes $n_0, \ldots, n_{k-1}$ such n_0 is the root node and every pair (n_i, n_{i+1}) is an edge of the procedure. A complete path (henceforth referred to simply as a path) is a partial path such that the k-th node of the sequence is a leaf of the tree. Given an initial system state (x_0, ϑ_0), an *execution* of a procedure is an ordered sequence of actions that generate a sequence of states: that is, given a path of the procedure $n_0, \ldots, n_{k-1}$ an execution is a sequence $(x_0, \vartheta_0), (x_1, \vartheta_1), \ldots, (x_n, \vartheta_n)$ such that for every pair $(x_i, \vartheta_i), (x_{i+1}, \vartheta_{i+1})$:

- $P_i(x_i, \vartheta_i)$ holds true, where P_i is the predicate labeling the edge (n_i, n_{i+1});
- $\vartheta_{i+1} = f_i(x_i, \vartheta_i)$ holds, where f_i is the action associated with node n_i.
- x_{i+1} is updated based on field observations.

The executions of a procedure implicitly account for logical time through the ordering of actions and states. Actual wall-clock timing in clinical procedures is rarely documented in source material and is often loosely defined in clinical practice. Accurately modeling timed events would require either data-driven estimates or a complete physiological model of the patient, both of which fall outside the scope of this work. For these reasons, we consider (clinical) procedures as timeless sequences of actions.

In procedures, controlled parameters ϑ represent variables that can be directly controlled by the system operator, while x represents the set of observable outputs or measurements that respond to the system's state and the applied actions. In the healthcare domain, for example, ϑ corresponds to quantities that clinical operators can directly control, such as the parameters of a mechanical ventilation device, whereas x corresponds to the patient's observed vitals, such as heart rate or respiration rate, which cannot be manipulated directly but are influenced by clinical actions.

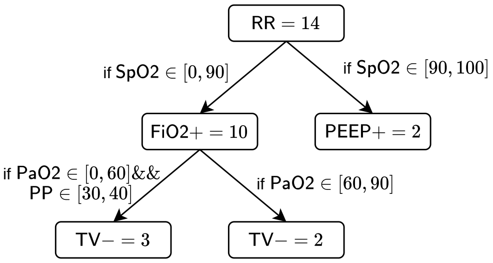


Figure 1: Graphical representation of example procedure. Units of measure are implied for simplicity.

Broadly speaking, a procedure in a clinical setting is a portion of a workflow that aims to keep observable vitals under control, i.e., to safeguard the patient's well-being, by dynamically adapting parameters based on the patient's condition.

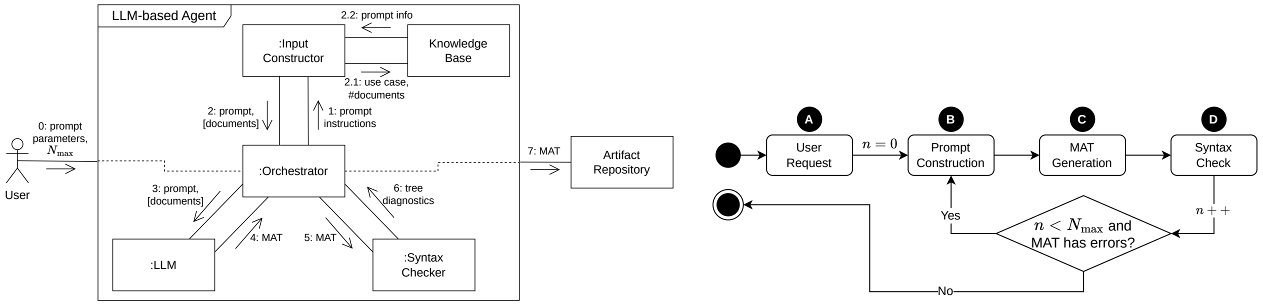
Figure 1 shows a graphical representation of a simple example procedure involving a patient undergoing mechanical ventilation. The controllable parameter set is $\vartheta = \{\mathsf{FiO2}, \mathsf{PEEP}, \mathsf{RR}, \mathsf{TV}\}$, while the observed metric set is $x = \{\mathsf{PaO2}, \mathsf{PP}, \mathsf{SpO2}\}$. The components of ϑ correspond to the knobs of the ventilation device that the healthcare operator can directly manipulate: fraction of inspired oxygen, positive end-expiratory pressure, forced respiration rate, and tidal volume, respectively. Conversely, x represents the subset of patient vitals observed in the procedure: arterial blood oxygen tension, plateau pressure, and peripheral oxygen saturation. Figure 1 illustrates a portion of the clinical workflow in which the healthcare operator adjusts the ventilator's settings ϑ in response to changes in the observed vitals x. Specifically, the action at the root node leaves ϑ unchanged except for RR, which is set to 14 breaths/min. Other actions increase or decrease a single parameter by a fixed amount while leaving the remaining parameters unchanged. Predicates on the tree edges exemplify the most common case, namely, checking whether one or more measured vitals fall within a safe interval. For instance, the leftmost edge leaving the root node verifies whether SpO2 lies in the range 0%–90%.

3.2 Procedural Workflow Extraction

Figure 2a presents a high-level overview of the proposed methodology as a collaboration diagram, illustrating the components and data flow. Figure 2b presents the sequence of steps in the iterative process as an activity diagram. MARACTUS employs an *LLM-based agent* that extracts clinical workflows from a knowledge base and represents them as textual, structured artifacts—the *MATs*[2].

We first describe the *knowledge base*. It is a curated collection, built and maintained by domain experts, of authoritative information sources that serve as the foundation for generating MATs. It

[2] MATs artifacts generated by MARACTUS are in JSON format, following the EBNF grammar specified in Appendix A.



(a) Collaboration diagram. **(b) Diagram of the methodology's workflow.**

Figure 2: Overview of proposed methodology.

aggregates documents from diverse origins, such as official guidelines, scientific literature, and textbooks, and organizes them into a structured repository. Its role is to provide reliable, domain-specific content that the LLM-based agent can reference when constructing machine-readable decision logic. The knowledge base serves as a contextual grounding layer, helping to guide the generation of artifacts in line with established medical standards and practices. In practice, the knowledge base is a heterogeneous collection encompassing documents for multiple clinical use cases.

The *LLM-based agent* is a distributed stack of software components that exposes two external interfaces: one that accepts the specification of the clinical use case, and one that delivers the MAT artifacts. Besides the knowledge base, the stack is made of four main components, as shown in Fig. 2a:

- *orchestrator*: coordinates the interaction among components, manages iterations, and ensures the overall workflow progresses toward valid outputs;
- *input constructor*: prepares the prompting strategy by combining use case information, task instructions, and selected documents from the knowledge base;
- *LLM*: reasoning engine which generates candidate MATs based on the constructed input;
- *syntax checker*: external tool validating the structural correctness of the MATs; if errors are found, returns them to the orchestrator for correction.

The communication and actions of the components of the LLM-based agent follow the sequence shown in Fig. 2b, executed iteratively. The agent generates an initial MAT and refines it in subsequent iterations as follows.

The workflow starts with a *user request* (step Ⓐ in Fig. 2b). Here, the user initiates a request to produce an MAT by specifying the clinical use case, the number of documents to sample from the knowledge base, and optional parameters such as the number of shots and whether to include a formal description of the output grammar (this information corresponds to message 0 in Fig. 2a).

Next, the input constructor is instructed to prepare a prompt (message 1). In the first iteration, it receives the information provided by the user. In subsequent iterations, the orchestrator instead supplies the MAT from the previous run alongside error diagnostics.

After receiving the instructions, the *prompt construction* (step Ⓑ) takes place. The input constructor builds a prompt according to the given instructions. It starts from a predefined template that specifies general information about the task, the grammar in Definition 3.1, and the expected MAT output format. In the first iteration only, two additional steps (corresponding to messages 2.1 and 2.2) are performed: the constructor queries the knowledge base with the use case and the requested number of documents, and retrieves the necessary information (use case description, list of valid actions, requested number of shots, and source documents). The retrieved information is then used to customize the template. Finally, the input constructor returns the assembled prompt to the orchestrator, along with the attached documents, if any (message 2).

Next, the orchestrator forwards the constructed input to the LLM module (message 3). In the *MAT generation* step (step Ⓒ), the LLM produces a candidate MAT based on the provided input and returns it to the orchestrator (message 4). The orchestrator then initiates a *syntax check* (step Ⓓ) by sending the generated MAT to the syntax checker (message 5).

Finally, the syntax checker validates the MAT against a grammar that guarantees the outcome is well-formed according to the specification defined by the input constructor. These rules are independent of the clinical use case and do not assess semantic validity. The syntax checker then returns diagnostics, such as a list of errors, for the MAT under analysis (message 6).

If step Ⓓ confirms that the MAT is free of syntax errors, the loop terminates and the MAT is returned as the final output. Otherwise, the orchestrator restarts the process at step Ⓑ by sending the faulty MAT and the diagnostics report back to the input constructor. The loop continues until a syntactically valid MAT is produced or a user-defined maximum number of iterations $N_{\max}$ is reached, thereby limiting computational and economic costs. The output MAT is then sent (message 7) and stored in an *artifact repository*.

It is important to note that the generated artifacts are not guaranteed to be semantically correct or clinically appropriate. Therefore, a formal clinical validation step should be performed before any MATs are integrated into digital healthcare services, ensuring patient safety and compliance with medical standards.

4 EMPIRICAL EVALUATION

This section reports on the experimental campaign[3] validating the reliability and cost of the presented methodology. We answer the following research questions:

RQ1: What is the syntactical correctness of the MATs produced by MARACTUS?
RQ2: How clinically relevant and accurate are the MATs, according to human evaluators?
RQ3: How effective are MAT executions in stabilizing patient metrics compared to the ground-truth decision-making process of a human physician?
RQ4: What is the cost of generating the MATs?

4.1 Design of the evaluation

4.1.1 Evaluation subjects. The evaluation subjects are two clinically relevant scenarios (chosen due to the availability of human experts for participation in the evaluation), both involving a single patient assisted by healthcare personnel who operate a medical device by adjusting its parameters in response to the patient's condition:

- **Acute Respiratory Distress Syndrome (ARDS)**: A patient is admitted to an Intensive Care Unit (ICU) with ARDS and supported by a mechanical ventilator. An intensivist operates the ventilator adjusting five device parameters that affect the patient's respiratory status. The intensivist determines these actions by monitoring vital signs, such as carbon dioxide concentration and respiratory rate. When a clinically relevant change is detected, they apply a mitigation strategy by manipulating ventilator settings according to standard medical procedures [43].
- **Target-Controlled Infusion (TCI)**: During surgery, a patient is administered intravenous anesthesia via a TCI pump. An anesthesiologist adjusts the concentration of drugs delivered to the patient to control the level of anesthesia and maintain physiological stability. As in the previous case, the expert monitors patient vitals, such as electroencephalogram signals and alertness indices, and modifies drug concentrations accordingly [13].

Both cases assume an average adult patient with no other severe medical conditions. The knowledge base contains 10 textual documents for each case, selected from official health organization guidelines, scientific articles, and textbooks. We list these documents in Table 1 alongside their size expressed in number of tokens.

The data regarding the two subjects (ARDS and TCI) has been collected with the involvement of three physicians with an average of four years of experience in intensive care and anesthesiology. Two of them assessed the quality of MATs produced by our methodology (see RQ2) and the third provided the simulated ground truth to compare with the MAT executions (see RQ3).

4.1.2 Methods under comparison. We test two different state-of-the-art models as the basis for our LLM-based agents: GPT-5 mini[4]and Google Gemini 2.5 Flash[5]. Preliminary experiments executed with smaller (locally deployed) models such as Llama-2-7B and Mistral-7B-Instruct-v0.3 showed that their capacity was insufficient to handle tasks with such large inputs. Moreover, the size of the required context window (see Table 1) ruled out many mid- to high-range open-source LLMs.

Table 1: Sources in the knowledge base and their size.

Use Case	Document	#tokens
ARDS	Grasselli et al. [21]	24886
ARDS	Fan et al. [16]	10966
ARDS	Griffiths et al. [22]	18277
ARDS	WHO (2020) [41]	17059
ARDS	Liaqat et al. [32]	7250
ARDS	Diamond et al. [12]	5714
ARDS	Qadir et al. [46]	11527
ARDS	Majumder and Minko [34]	17910
ARDS	Owens [43]	37817
ARDS	WHO (2022) [42]	61170
TCI	Sukumar et al. [54]	17284
TCI	Nimmo et al. [38]	54760
TCI	Thomson et al. [56]	22781
TCI	Struys et al. [53]	30689
TCI	Absalom et al. [1]	42639
TCI	Eleveld et al. (2018) [13]	50458
TCI	Al-Rifai and Mulvey (2016a) [2]	21548
TCI	Al-Rifai and Mulvey (2016b) [3]	22280
TCI	Lai et al. [30]	37880
TCI	Eleveld et al. (2020) [14]	29922

Table 2: Experimental factors and their tested values.

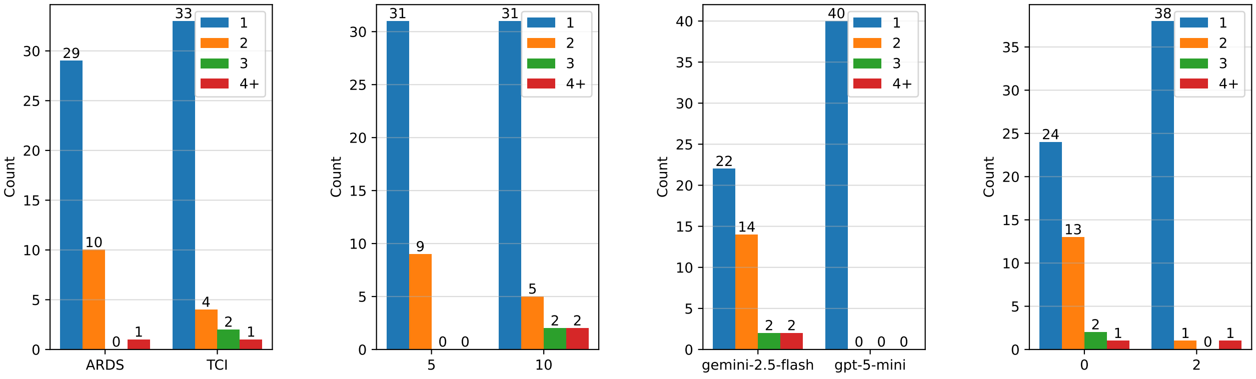
Factor	Values
Use Case	ARDS, TCI
Model	gemini-2.5-flash, gpt-5-mini
#input Docs	5, 10
Shots	0, 2
Repeats	0, 1, 2, 3, 4
Total Combinations	80

We evaluate the impact of few-shot learning by either adding two examples or providing no examples in the prompt. Each experiment setting (defined as the combination of use case, LLM, number of input documents, and number of shots) is repeated 5 times to enhance statistical robustness, with a maximum number of iterations $N_{\max}$ set to 3, a value chosen based on preliminary experiments. The experimental campaign uses a factorial design and explores all combinations of the factors summarized in Table 2, resulting in a total of 80 evaluated MATs.

We do not include an end-to-end evaluation baseline, as no prior system exists that performs the full process of generating, validating, and executing machine-readable action trees in healthcare. Consequently, direct comparisons to a complete alternative methodology are not possible.

4.1.3 Statistical tests. Following the recommendations of Arcuri and Briand [5], we employ the non-parametric Mann–Whitney U

[3] Replication package found at https://doi.org/10.5281/zenodo.17233097.
[4] https://platform.openai.com/docs/models/gpt-5-mini. Last updated in August 2025 at the time of writing.
[5] https://ai.google.dev/gemini-api/docs/models#gemini-2.5-flash. Last updated in June 2025 at the time of writing.



(a) Case **(b) Docs** **(c) Model** **(d) Shots**

Figure 3: Number of RQ1 iterations divided by factor classes.

test [35] to assess whether observed differences in our results are statistically significant by examining the test's p-value. A small p-value (e.g., less than 0.01) for a test comparing two populations indicates rejection of the null hypothesis that the distributions are the same, suggesting a statistically significant difference. To complement this analysis, we report Vargha and Delaney's $\hat{A}_{12}$ measure [58], which captures the effect size by estimating the degree of stochastic dominance between two samples. Effect sizes are interpreted using conventional thresholds: $\hat{A}_{12}$ ($= 1 - \hat{A}_{21}$) values of at least 0.44, 0.56, 0.64, and 0.71 are classified as none (N), small (S), medium (M), and large (L), respectively.

4.1.4 Evaluation testbed. For the experiments, the LLMs were accessed as cloud services, while the other components of the agents were deployed on an Ubuntu 25.04 machine equipped with an 8-core Intel Core i5 processor at 4.2 GHz and 16 GB of RAM.

4.2 Evaluation results

4.2.1 RQ1 (syntactic correctness). For each experimental repeat, we measure the number of iterations required to obtain a syntactically correct MAT, ranging from 1 (if the MAT is correct on the first attempt without any follow-up) to 3 (the maximum number of iterations $N_{\max}$ requested). We indicate as "4+" the cases in which syntax errors persist after $N_{\max}$ tries. Figure 3 reports these results, grouped by experimental factors (see Table 2): use case, number of input documents, LLM, and use of shots.

Overall, the methodology yields syntactically correct MATs in the majority of cases: only 2 out of 80 remained faulty after 3 iterations (red bars). The factors with the strongest influence are the choice of LLM and the absence of shots. In particular, all MATs generated with gpt-5-mini were syntactically correct at the first attempt, whereas gemini-2.5-flash produced 18 faulty trees out of 40 (14 corrected after 2 iterations and 2 corrected after 3 iterations). Similarly, prompts with shots produced only 2 faulty trees out of 40, compared to 16 without few-shot learning.

RQ1 summary. MARACTUS consistently produces syntactically correct MATs, with only 2.5% of the runs failing after $N_{\max} = 3$ iterations. The strongest improvements derive from using gpt-5-mini as generation engine and few-shot learning in the prompting strategy.

4.2.2 RQ2 (clinical validity). To address RQ2, we engaged two human raters, both experienced physicians specializing in anesthesia and intensive care. They were asked to complete a survey evaluating the MATs generated by our methodology, rating their clinical validity on a Likert scale. The raters, both working in a hospital facility in Western Europe, were selected for their domain expertise to strengthen the reliability of the assessment. Each rater received a different version of the survey, so that together they covered all 80 MATs produced in the experimental campaign, with 40 items per survey and three evaluation dimensions per item. Each survey included MATs representing all combinations of experimental factors, presented in a fixed randomized order.

The raters assessed the clinical accuracy of each diagram by quantifying their agreement with the following statements, or evaluation dimensions:

(1) "The physical *quantities* indicated in the actions and conditions are relevant to the clinical practices you know."
(2) "The *actions* (blocks) are physically consistent and in line with the clinical practices you know."
(3) "The *conditions* (arrows) are complete and in line with the clinical practices you know."

Responses were given on a five-point Likert scale: "Strongly disagree" (1), "Slightly disagree" (2), "Neutral" (3), "Slightly agree" (4), and "Strongly agree" (5).

The Likert-scale evaluations of clinical validity are summarized in Table 3, where averages are computed within groups sharing the same factor values. These results show clear differences across factors. The most prominent effect is the use case: procedures generated for ARDS received consistently higher ratings (around 3.6–3.9 across categories) than those for TCI (around 2.4–2.7). Statistical tests in Table 4 confirm this difference, with large or medium effect

Table 3: Average RQ2 rating divided by factor classes.

Factor	Value	quantities	actions	conditions
Case	ARDS	3.90	3.64	3.69
	TCI	2.54	2.74	2.36
Shots	0	3.08	3.08	2.92
	2	3.36	3.31	3.13
Model	gemini-2.5-flash	3.13	3.24	3.00
	gpt-5-mini	3.30	3.15	3.05
Docs	5	3.33	3.10	3.02
	10	3.11	3.29	3.03

Table 4: Results of statistical tests for RQ2 ratings.

		p-val	$\hat{A}_{12}$
quantities	ARDS vs TCI	5.40e-10	L
	0 vs 2	2.17e-01	N
	gemini-2.5-flash vs gpt-5-mini	5.52e-01	N
	5 vs 10	5.18e-01	N
actions	ARDS vs TCI	3.39e-05	M
	0 vs 2	1.96e-01	N
	gemini-2.5-flash vs gpt-5-mini	8.80e-01	N
	5 vs 10	2.94e-01	N
conditions	ARDS vs TCI	7.85e-10	L
	0 vs 2	3.36e-01	N
	gemini-2.5-flash vs gpt-5-mini	8.45e-01	N
	5 vs 10	9.96e-01	N

sizes and highly significant differences across all three evaluation dimensions. By contrast, the other factors (shots, model, and number of documents) show only minor variations in average scores, none of which reach statistical significance.

Figure 4 illustrates the impact of this difference. The figure presents the distributions of validity ratings for dimension 2 (assessing the correctness of MAT *actions*), split by use case (Fig. 4a), number of documents (Fig. 4b), LLM (Fig. 4c), and presence of shots (Fig. 4d). In Fig. 4a, the ARDS distribution clearly shows higher ratings compared to TCI. Across the other subfigures, the median (red) is frequently much higher than the arithmetic mean (blue). This pattern can be attributed to a few low ratings on TCI cases pulling down the average, while the majority of ratings remain concentrated around 4–5. In particular, the proportion of MATs that received a score of at least 4 is 50% for dimension 1 (*quantities*), 60% for dimension 2 (*actions*), and 46% for dimension 3 (*conditions*).

RQ2 summary. The clinical validity evaluation on a five-point Likert scale shows that most MATs receive generally positive ratings across all three evaluated dimensions (validity of physical quantities, physician actions, and action conditions), with ARDS cases averaging 3.6–3.9 and TCI cases averaging 2.4–2.7 across the three evaluation dimensions (a few low ratings on TCI cases lower the overall mean). The obtained median value proves that half of the responses remain concentrated around 4–5. Other experimental factors (shots, model, and number of documents) have only a minor influence on the ratings.

4.2.3 *RQ3 (effectiveness w.r.t. ground truth).*

This RQ compares the patient state resulting from following an MAT procedure with the state resulting from the clinical actions of a human physician. For each MAT, we evaluate how many of the five monitored patient vitals (carbon dioxide, heart rate, oxygen saturation, respiration rate, and tidal volume) fall within safe stabilization ranges, as defined in previous research [23], at the end of a fixed observation period, and we compare this number to the human ground truth. We consider the application of an MAT procedure *successful* if the number of stabilized metrics is equal to or greater than the human baseline.

The ICU clinical setting with a patient affected by ARDS is simulated through BREATHE [11], a high-fidelity simulation platform that supports a wide range of health complications affecting a patient with varying severity[6]. BREATHE also provides real-time monitoring of patient vitals and interactive control of mechanical ventilation devices.

For each MAT, we induce ARDS in the simulated patient and use BREATHE's interface to automatically control the mechanical ventilator according to each MAT. Starting from the root node of the MAT, with an initial parameter set ϑ_0 corresponding to the ventilator's default settings in BREATHE (plus any changes specified in the root node), the simulator tracks the current MAT node. At periodic time intervals, it checks whether the patient's current vitals satisfy the conditions of any outgoing edges; if so, it advances to the corresponding node and applies the associated actions (i.e., changes to ventilator parameters). This process continues until a leaf node is reached, at which point the procedure is completed. In this way, we obtain one execution for each generated MAT.

Ground-truth data were collected with the involvement of a third physician specializing in intensive care, who executed the ARDS scenario using BREATHE. The intensivist monitored real-time patient vitals and adjusted ventilation settings according to their clinical expertise and established medical practices.

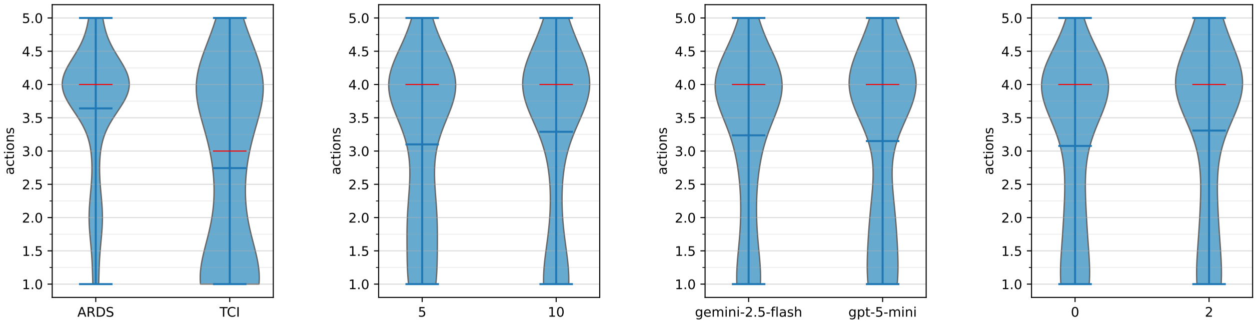
Results indicate that 25 out of the 40 MATs are successful according to the above definition. This figure corresponds to a *success rate* of 25/40 = 62.5%. Figure 5 shows success rates computed similarly, but with MATs grouped according to the three varying factors: number of documents (Fig. 5a), LLM (Fig. 5b), and number of shots (Fig. 5c). The results indicate that each factor has a measurable impact of approximately 10% on the success rate, with higher numbers of input documents, the gpt-5-mini LLM, and the inclusion of shots all contributing positively.

RQ3 summary. Executions of MATs stabilized patient vitals at least as effectively as the human ground truth in 62.5% of the simulated cases. An improvement of 10% can be observed when adopting larger input document sets, gpt-5-mini as a reasoning engine, and few-shot learning in the prompting strategy.

4.2.4 *RQ4 (cost of generation).*

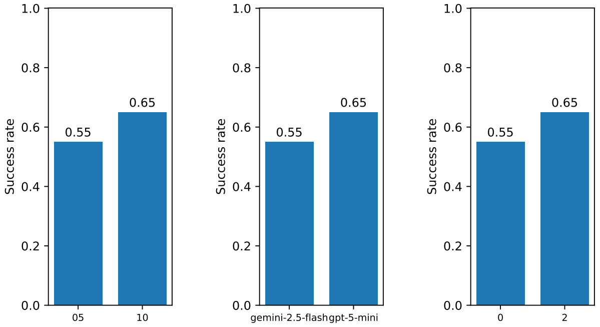
Figure 6 shows boxplots of MAT generation times split by each experimental factor. Median generation times generally range between 80 and 90 seconds, depending on the experimental factor. Overall, nearly all generations take between 1 and 3 minutes. Overall, generation times are fairly consistent across use cases, models, and the number of documents,

[6] In RQ3, we restrict our evaluation to the ARDS subject, since the current version of BREATHE does not provide support for TCI-based simulations.



(a) Case **(b) Docs** **(c) Model** **(d) Shots**

Figure 4: Distributions of RQ2 ratings for evaluation dimension 2 (*actions*) divided by factor classes. Red lines represent medians, while blue lines represent arithmetic means.



(a) Docs **(b) Model** **(c) Shots**

Figure 5: RQ3 success rates divided by factor classes.

Table 5: Results of statistical tests for RQ4 generation times.

	p-val	$\hat{A}_{12}$
ARDS vs TCI	3.10e-01	N
0 vs 2	1.44e-05	L
gemini-2.5-flash vs gpt-5-mini	5.12e-02	S
5 vs 10	7.37e-01	N

with only minor variations in median values. In contrast, including few-shot examples increases the median generation time by roughly 40%, as evident from the boxplots. Mann–Whitney U tests and Vargha–Delaney effect sizes reported in Table 5 confirm that only the presence of shots produces a statistically significant and large effect on generation time, whereas the other factors show no meaningful impact.

RQ4 summary. Generation times remain stable across use cases, models, and document counts, with the only statistically significant increase (about 40%) arising from the inclusion of few-shot examples.

4.3 Threats to validity

We limit external validity threats by considering more than one evaluation subject that are representative instances of clinical cases: ARDS and TCI. We also selected mainstream LLMs having state-of-the-art complexity in terms of size (million parameters). We did not include open-source models, as our approach interacts with models in a black-box manner, i.e., without access to internal details such as architecture, weights, or token probability distributions.

We mitigate the risk of obtaining results by chance by repeating all experiments five times for each experimental setting, including the evaluation subject, LLM, number of input documents, and number of shots, for a total of 80 runs. Where applicable, we assess both the statistical significance (Mann–Whitney U test) and effect size (Vargha-Delaney's) of our results.

We did not fine-tune the parameters of the different versions of MARACTUS. Therefore, we do not exclude the possibility that the effectiveness of some variants could be further enhanced with an optimal configuration.

The whole set of MATs produced in the experimental campaign has been evaluated by two human raters having domain expertise. Both raters are physicians specializing in anesthesia and intensive care with an average of four years of experience. A potential threat to validity is the homogeneity of the human assessors, all of whom are of Western European origin and working in a Western European hospital facility, which may introduce bias. The small number of raters is due to the difficulty of recruiting participants with the required expertise and may also introduce bias, as it does not allow for an inter-rater agreement analysis. Furthermore, our results are constrained by the capabilities of the selected LLMs, which have limited context windows, as well as by the quantity and coverage of clinical documents in the knowledge base.

5 PROSPECTIVE IMPACT

The precise, machine-readable representation of procedural knowledge that MATs provide may enhance several healthcare services. We summarize the envisioned impact in Fig. 7.

From a **scientific and technological perspective**, MATs serve as a structured backbone for multiple innovations. Visualization and explainability tools make MATs interpretable by humans. They

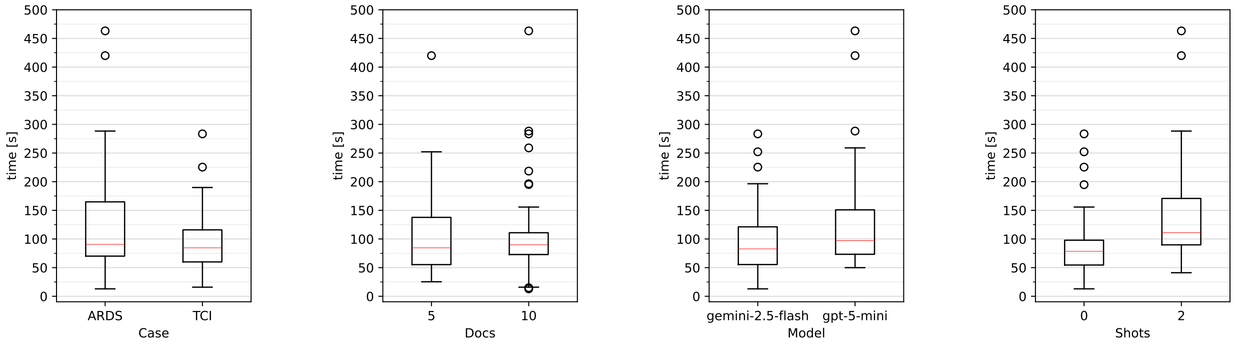


Figure 6: RQ4 generation time divided by factor classes.

render the machine-readable structures into graphical decision trees, interactive flowcharts, or annotated diagrams that highlight conditions and actions. Such tools help clinicians and researchers quickly understand the logic of the generated procedures and compare them against existing guidelines. They also provide an entry point for education and training, enabling healthcare personnel or students to explore clinical protocols in a visual and accessible way.

Verification tools can analyze MATs to ensure that they satisfy properties such as safety, completeness, or consistency, for instance, by checking that no branch of a procedure prescribes unsafe ventilator settings, or that all clinically relevant conditions are covered. This requires combining the procedural knowledge encoded in the MATs with requirement specifications expressed in a rigorous, or even formal way, such as temporal logic formalisms [47]. This type of evaluation provides regulatory bodies and hospital IT departments with a rigorous basis to ensure that automated procedures meet safety requirements and comply with medical standards.

By linking MATs with clinical datasets (e.g., electronic health records or ICU monitoring data), predictive models can be trained to anticipate outcomes of different decision paths. As an example, integrating an MAT for ARDS ventilation with patient data enables the development of models that estimate the impact of PEEP modifications on blood oxygenation. These models can support classification tasks (e.g., identifying patients at high risk of deterioration) or prediction tasks (e.g., estimating likely responses to interventions). In this way, MATs serve as structured scaffolds that align free-form data with procedural knowledge.

Simulation environments can combine MATs with mathematical or computational models of patient physiology to create dynamic, interactive representations of clinical practice. For example, a simulator could apply an MAT describing ventilator management to a virtual lung model that responds realistically to changes in the mechanical ventilation parameters. These environments enable testing of both existing and newly generated procedures under a wide range of simulated patient conditions. Simulators can be used for training medical personnel, evaluating the robustness of protocols before clinical adoption, or conducting in silico research studies where real trials would be infeasible.

At the **organizational and socio-economic level**, the most complex tools directly integrate MATs into real-time clinical workflows. The MAT logic is combined with continuous patient monitoring, verification layers, predictive models, and simulation backends to provide actionable recommendations at the bedside. For instance, a Clinical Decision Support System (CDSS) could monitor a mechanically ventilated patient, automatically evaluate their status against the MAT, and suggest parameter adjustments to the intensivist. To ensure trust and safety, such systems rely on visualization tools for explainability, verification programs for formal assurance, and predictive models to anticipate risks. This category represents the ultimate application of MATs, enabling knowledge-grounded, AI-augmented support for clinical decision-making.

In the **clinical domain**, MATs contribute to the standardization of procedures and improves patient safety by reducing ambiguity and error. Visualization and simulation tools make protocols more comprehensible and testable, while predictive models support personalized decision-making by anticipating patient-specific responses. When embedded into CDSSs, MATs can directly influence bedside practice, integrating monitoring data, verification layers, and simulation backends to provide transparent recommendations.

The different kinds of software enabled by MATs involve distinct categories of human users:

- *Oversight bodies* interact with verification programs and data-driven models to ensure that generated procedures meet safety, compliance, and quality standards before clinical adoption.
- *Educators and trainees* benefit from visualization and explainability tools and from simulation engines, which provide interactive training environments based on realistic physiological responses.
- *Technical developers* use data-driven prediction models as a foundation for building and refining AI-based tools that integrate procedural knowledge with clinical datasets, enabling more advanced decision-support technologies.
- *Clinical staff* rely on visualization and explainability tools to quickly understand and validate recommended procedures, and on CDSSs for real-time assistance during patient care.

6 RELATED WORK

In the software engineering community, there has been significant focus on synthesizing artifacts from NL comments. For instance,

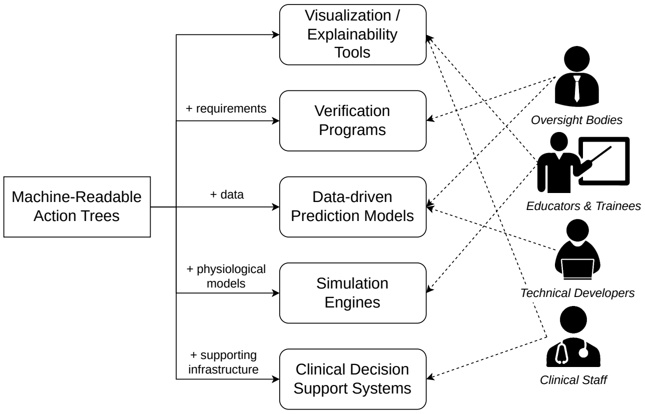

Machine-Readable Action Trees
Visualization / Explainability Tools
+ requirements
Verification Programs
+ data
Data-driven Prediction Models
+ physiological models
Simulation Engines
+ supporting infrastructure
Clinical Decision Support Systems
Oversight Bodies
Educators & Trainees
Technical Developers
Clinical Staff

Figure 7: Impact of the proposed methodology (boxes are software tools linked to the prospective users).

the automatic generation of software specifications, such as preconditions and post-conditions, can support program verification and test case generation. These specifications may take the form of code assertions or formal specification languages. Zhai et al. [66] present C2S, a tool that automates specification synthesis from NL comments using a novel abstract language model. Blasi et al. [6] propose Jdoctor, which employs Natural Language Processing (NLP) techniques to translate Javadoc comments into Java expressions. A similar goal is pursued by Endres et al. [15] and Xie et al. [64], who use LLMs to generate software specifications for Java and Python. Ryu et al. [49] introduce CLOVER, which translates NL sentences into compositional first-order logic formulae through a multi-step reasoning process using an LLM. Similarly, Tagliaferro et al. [55] start from NL informal specifications, but focus on evaluating LLM performance in translating them into UML component diagrams.

Beyond specification synthesis, LLMs have shown strong performance in a broader range of information extraction tasks. Wan et al. [60] propose TnT-LLM, a zero-shot, multi-stage reasoning approach for taxonomy labeling and document classification. A related task is Named Entity Recognition (NER), which identifies key information and assigns it to predefined categories. Wang et al. [61] introduce GPT-NER, which improves LLM labeling accuracy by framing it as a token-decorated text generation task. Perot et al. [45] present LMDX, a method for extracting repeated and hierarchical entities from visually rich, semi-structured documents into structured JSON forms. Knauer et al. [28] investigate how pretrained LLMs can leverage their compressed knowledge to generate interpretable decision trees without external training data.

A closely related line of work involves capturing processes from NL descriptions, a task known as procedural text mining, in addition to extracting static information. This topic has long been a research interest in the information systems community [20]. Here too, LLM-based approaches have increasingly emerged as an alternative to traditional NLP techniques. Rula and D'Souza [48] and Carriero et al. [8] both use LLMs to extract procedure steps and represent them as human-readable procedural knowledge graphs based on specific ontologies. The latter further conduct a user study to assess the quality and usefulness of the generated graphs.

The healthcare domain has recently adopted similar ideas, motivated by the need to capture and operationalize clinical processes from unstructured sources. A notable example is the Text2DT task at the 8th China Conference on Health Information Processing (CHIP 2022) [68], which aimed to extract Medical Decision Trees (MDTs) from manually annotated Chinese medical guidelines and textbooks to improve CDSSs. Some approaches in this track employ LLMs, including those by He et al. [25] and Zhu et al. [67]. While the setting and overall objectives of these works resemble ours, the resulting artifacts differ substantially. Specifically, prior work produces binary decision trees with broad applicability, vague split conditions, and loosely defined actions. Such representations are primarily intended for human interpretability, by physicians (e.g., general practitioners during routine check-ups) or chatbot-based CDSSs. In contrast, MARACTUS adopts a lower-level perspective, targeting specific diseases and action spaces. Our methodology generates Machine-readable Action Trees (MATs) with rigorously defined conditions and actions, making them substantially more useful than the state of the art for automating medical knowledge.

Beyond procedural knowledge, LLMs have also been applied to other information extraction tasks in clinical settings like NER [26], classification [27], and structured information mining [18].

7 CONCLUSION

We introduce MARACTUS, a methodology for the agentic generation of MATs from unstructured clinical guidelines. By leveraging LLMs within an orchestrated agentic workflow, we show how narrative medical knowledge can be transformed into structured, machine-readable specifications suitable for automated processing, verification, and integration into digital healthcare services.

Our empirical evaluation across two clinically relevant subjects showed that the proposed methodology reliably produces syntactically correct MATs (97.5% success rate) and achieves clinically meaningful results, with 62.5% of executions stabilizing patient vitals as effectively as or more effectively than a human-produced ground truth. Expert evaluations further confirmed the clinical validity of the generated artifacts, with over half of them receiving high scores from human raters on multiple dimensions of quality.

Beyond quantitative results, we highlighted how MATs serve as foundational artifacts for a broad ecosystem of healthcare software: from visualization and explainability tools, to verification frameworks, simulation engines, and ultimately clinical decision support systems. These applications collectively contribute to improving patient safety and enhancing trust in AI-assisted care.

While our findings are encouraging, limitations remain. Current results are bounded by the capabilities of the selected LLMs and the coverage of available clinical documents. Moreover, clinical assessment was performed by a limited pool of experts. Addressing these limitations by extending to broader clinical conditions, incorporating diverse medical expertise, and integrating real-world patient data constitutes an important direction for future work.

ACKNOWLEDGMENTS

This work was partially supported by the PRIN project SAFEST (award No: 20224AJBLJ) funded by the Italian Ministry of Education, Universities, and Research (MIUR).

REFERENCES

[1] AR Absalom, V Mani, Tom De Smet, and MMRF Struys. Pharmacokinetic models for propofol—defining and illuminating the devil in the detail. *British journal of anaesthesia*, 103(1):26–37, 2009.

[2] Z Al-Rifai and D Mulvey. Principles of total intravenous anaesthesia: basic pharmacokinetics and model descriptions. *Bja Education*, 16(3):92–97, 2016.

[3] Z Al-Rifai and D Mulvey. Principles of total intravenous anaesthesia: practical aspects of using total intravenous anaesthesia. *Bja Education*, 16(8):276–280, 2016.

[4] Rohan Anil et al. Gemini: A family of highly capable multimodal models. *CoRR*, abs/2312.11805, 2023.

[5] Andrea Arcuri and Lionel Briand. A practical guide for using statistical tests to assess randomized algorithms in software engineering. In *Proceedings of the 33rd international conference on software engineering*, pages 1–10, 2011.

[6] Arianna Blasi, Alberto Goffi, Konstantin Kuznetsov, Alessandra Gorla, Michael D. Ernst, Mauro Pezzè, and Sergio Delgado Castellanos. Translating code comments to procedure specifications. In *Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 242–253. Association for Computing Machinery, 2018.

[7] Lola Burgueño, Davide Di Ruscio, Houari Sahraoui, and Manuel Wimmer. Automation in model-driven engineering: A look back, and ahead. *ACM Trans. Softw. Eng. Methodol.*, 34(5), May 2025.

[8] Valentina Anita Carriero, Antonia Azzini, Ilaria Baroni, Mario Scrocca, and Irene Celino. Human Evaluation of Procedural Knowledge Graph Extraction from Text with Large Language Models. In Mehwish Alam, Marco Rospocher, Marieke van Erp, Laura Hollink, and Genet Asefa Gesese, editors, *Knowledge Engineering and Knowledge Management*, pages 434–452. Springer Nature Switzerland, 2025.

[9] Sribala Vidyadhari Chinta, Zichong Wang, Avash Palikhe, Xingyu Zhang, Ayesha Kashif, Monique Antoinette Smith, Jun Liu, and Wenbin Zhang. Ai-driven healthcare: Fairness in ai healthcare: A survey. *PLOS Digital Health*, 4(5), 2025.

[10] Ludovico Gennaro Cobuccio, Vincent Faivre, Rainer Tan, Alan Vonlanthen, Fenella Beynon, Emmanuel Barchichat, Alain Fresco, Quentin Girard, Sinan Ucak, Sylvain Schaufelberger, et al. medAL-suite: a software solution for creating and deploying complex clinical decision support algorithms. *BMC Medical Informatics and Decision Making*, 25:249, 2025.

[11] Alessandro Colombo, Gionatha Pirola, and Angelo Gargantini. BREATHE - Biomedical Respiratory Engine for Advanced Training and Human Evaluation. https://github.com/GionathaPirola/BREATHE, 2025.

[12] Matthew Diamond, Hector L Peniston, Devang K Sanghavi, and Sidharth Mahapatra. Acute respiratory distress syndrome. In *StatPearls [internet]*. StatPearls Publishing, 2024.

[13] Douglas J Eleveld, Pieter Colin, Anthony R Absalom, and Michael MRF Struys. Pharmacokinetic–pharmacodynamic model for propofol for broad application in anaesthesia and sedation. *British journal of anaesthesia*, 120(5):942–959, 2018.

[14] Douglas J Eleveld, Pieter Colin, Anthony R Absalom, and Michel MRF Struys. Target-controlled-infusion models for remifentanil dosing consistent with approved recommendations. *British Journal of Anaesthesia*, 125(4):483–491, 2020.

[15] Madeline Endres, Sarah Fakhoury, Saikat Chakraborty, and Shuvendu K. Lahiri. Can Large Language Models Transform Natural Language Intent into Formal Method Postconditions? *Proc. ACM Softw. Eng.*, 1(FSE):84:1889–84:1912, 2024.

[16] Eddy Fan, Lorenzo Del Sorbo, Ewan C Goligher, Carol L Hodgson, Laveena Munshi, Allan J Walkey, Neill KJ Adhikari, Marcelo BP Amato, Richard Branson, et al. An official american thoracic society/european society of intensive care medicine/society of critical care medicine clinical practice guideline: mechanical ventilation in adult patients with acute respiratory distress syndrome. *American journal of respiratory and critical care medicine*, 195(9):1253–1263, 2017.

[17] Desiree Fleck, Hossam Gad, Beth Hogan Quigley, Mohamed Antar, Ahmed Sayed Ahmed, Mohamed A Mahmoud, and Krzysztof Laudanski. The effect of clinical ambiguity on the decision-making process among intensive care unit providers in northern america using clinical vignettes in mixed methods study. *Journal of Multidisciplinary Healthcare*, pages 3091–3104, 2025.

[18] Raffaello Fornasiere, Nicolò Brunello, Vincenzo Scotti, and Mark Carman. Medical information extraction with large language models. In *Proceedings of the 7th International Conference on Natural Language and Speech Processing (ICNLSP 2024)*, pages 456–466, 2024.

[19] Juan Víctor Ariel Franco, Marcelo Arancibia, Nicolás Meza, Eva Madrid, and Karin Kopitowski. Clinical practice guidelines: Concepts, limitations and challenges. *Medwave*, 20(3), 2020.

[20] Fabian Friedrich, Jan Mendling, and Frank Puhlmann. Process Model Generation from Natural Language Text. In *Advanced Information Systems Engineering*, pages 482–496. Springer, 2011.

[21] Giacomo Grasselli, Carolyn S Calfee, Luigi Camporota, Daniele Poole, Marcelo BP Amato, Massimo Antonelli, Yaseen M Arabi, Francesca Baroncelli, Jeremy R Beitler, Giacomo Bellani, et al. Esicm guidelines on acute respiratory distress syndrome: definition, phenotyping and respiratory support strategies. *Intensive care medicine*, 49(7):727–759, 2023.

[22] Mark JD Griffiths, Danny Francis McAuley, Gavin D Perkins, Nicholas Barrett, Bronagh Blackwood, Andrew Boyle, Nigel Chee, Bronwen Connolly, Paul Dark, Simon Finney, et al. Guidelines on the management of acute respiratory distress syndrome. *BMJ open respiratory research*, 6(1), 2019.

[23] Bruno Guindani, Matteo Camilli, Livia Lestingi, and Marcello Maria Bersani. Detecting Dependability Failures in Healthcare Scenarios via Digital Shadows. In *IEEE 36th International Symposium on Software Reliability Engineering (ISSRE)*, pages 502–513. IEEE, 2025.

[24] Reinhold Haux. Medical informatics: past, present, future. *International journal of medical informatics*, 79(9):599–610, 2010.

[25] Yuxin He, Buzhou Tang, and Xiaoling Wang. Generative Models for Automatic Medical Decision Rule Extraction from Text. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7034–7048. Association for Computational Linguistics, 2024.

[26] Yan Hu, Qingyu Chen, Jingcheng Du, Xueqing Peng, Vipina Kuttichi Keloth, Xu Zuo, Yujia Zhou, Zehan Li, Xiaoqian Jiang, Zhiyong Lu, Kirk Roberts, and Hua Xu. Improving large language models for clinical named entity recognition via prompt engineering. *Journal of the American Medical Informatics Association*, 31(9):1812–1820, 2024.

[27] Jingwei Huang, Donghan M. Yang, Ruichen Rong, Kuroush Nezafati, Colin Treager, Zhikai Chi, Shidan Wang, Xian Cheng, Yujia Guo, Laura J. Klesse, Guanghua Xiao, Eric D. Peterson, Xiaowei Zhan, and Yang Xie. A critical assessment of using ChatGPT for extracting structured data from clinical notes. *npj Digital Medicine*, 7(1):106, 2024.

[28] Ricardo Knauer, Mario Koddenbrock, Raphael Wallsberger, Nicholas M. Brisson, Georg N. Duda, Deborah Falla, David W. Evans, and Erik Rodner. "Oh LLM, I'm Asking Thee, Please Give Me a Decision Tree": Zero-Shot Decision Tree Induction and Embedding with Large Language Models. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, pages 1196–1206, 2025.

[29] Hanbum Ko, Hongjun Yang, Sehui Han, Sungwoong Kim, Sungbin Lim, and Rodrigo Hormazabal. Filling in the gaps: LLM-based structured data generation from semi-structured scientific data. In *ICML 2024 AI for Science Workshop*, 2024.

[30] Hou-Chuan Lai, Yi-Hsuan Huang, Jen-Yin Chen, Chih-Shung Wong, Kuang-I Cheng, Ching-Hui Shen, and Zhi-Fu Wu. Safe practice of total intravenous anesthesia with target-controlled infusion in taiwan: A recommendation. *Asian Journal of Anesthesiology*, 2021.

[31] Siri Lange, Aziza Mwisongo, and Ottar Mæstad. Why don't clinicians adhere more consistently to guidelines for the integrated management of childhood illness (IMCI)? *Social science & medicine*, 104:56–63, 2014.

[32] Adnan Liaqat, Matthew Mason, Brian J Foster, Sagar Kulkarni, Aisha Barlas, Awais M Farooq, Pooja Patak, Hamza Liaqat, Rafaela G Basso, Mohammed S Zaman, et al. Evidence-based mechanical ventilatory strategies in ards. *Journal of clinical medicine*, 11(2):319, 2022.

[33] Sachiko Lim, Aron Henriksson, and Jelena Zdravković. Data-driven requirements elicitation: A systematic literature review. *SN Computer Science*, 2(1):16, 2021.

[34] Joydeb Majumder and Tamara Minko. Recent developments on therapeutic and diagnostic approaches for covid-19. *The AAPS journal*, 23(1):14, 2021.

[35] Henry B Mann and Donald R Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, pages 50–60, 1947.

[36] Josephine Mayer, Christopher Kipps, and Hannah R Cock. Implementing clinical guidelines. *Practical Neurology*, 19(6):529–535, 2019.

[37] Garrett Mehl, Özge Tunçalp, Natschja Ratanaprayul, Tigest Tamrat, María Barreix, David Lowrance, Kidist Bartolomeos, Lale Say, Nenad Kostanjsek, et al. WHO SMART guidelines: optimising country-level use of guideline recommendations in the digital age. *The Lancet Digital Health*, 3(4):e213–e216, 2021.

[38] Alastair F Nimmo, Anthony R Absalom, O Bagshaw, A Biswas, TM Cook, A Costello, S Grimes, D Mulvey, S Shinde, T Whitehouse, et al. Guidelines for the safe practice of total intravenous anaesthesia (tiva) joint guidelines from the association of anaesthetists and the society for intravenous anaesthesia. *Anaesthesia*, 74(2):211–224, 2019.

[39] OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023.

[40] World Health Organization. WHO guideline: recommendations on digital interventions for health system strengthening, 2019.

[41] World Health Organization. Clinical management of severe acute respiratory infection (sari) when covid-19 disease is suspected: interim guidance, 13 march 2020. *World Health Organization*, 13(03), 2020.

[42] World Health Organization. Clinical care of severe acute respiratory infections – tool kit: Covid-19 adaptation, update 2022, 2022.

[43] William Owens. *The ventilator book*. First Draught Press, 2018.

[44] B. Pérez and I. Porres. Authoring and verification of clinical guidelines: a model driven approach. *Journal of Biomedical Informatics*, 43(4):520–536, August 2010. Epub 2010 Mar 4.

[45] Vincent Perot, Kai Kang, Florian Luisier, Guolong Su, Xiaoyu Sun, Ramya Sree Boppana, Zilong Wang, Zifeng Wang, Jiaqi Mu, Hao Zhang, Chen-Yu Lee, and Nan Hua. LMDX: Language Model-based Document Information Extraction and

Localization. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 15140–15168. Association for Computational Linguistics, 2024.

[46] Nida Qadir, Sarina Sahetya, Laveena Munshi, Charlotte Summers, Darryl Abrams, Jeremy Beitler, Giacomo Bellani, Roy G Brower, Lisa Burry, Jen-Ting Chen, et al. An update on management of adult patients with acute respiratory distress syndrome: an official american thoracic society clinical practice guideline. *American journal of respiratory and critical care medicine*, 209(1):24–36, 2024.

[47] Nicholas Rescher and Alasdair Urquhart. *Temporal logic*, volume 3. Springer Science & Business Media, 2012.

[48] Anisa Rula and Jennifer D'Souza. Procedural Text Mining with Large Language Models. In *Proceedings of the 12th Knowledge Capture Conference 2023*, pages 9–16. Association for Computing Machinery, 2023.

[49] Hyun Ryu, Gyeongman Kim, Hyemin S. Lee, and Eunho Yang. Divide and Translate: Compositional First-Order Logic Translation and Verification for Complex Logical Reasoning. *International Conference on Representation Learning*, 2025:24935–24964, 2025.

[50] Victor Sanh et al. Multitask prompted training enables zero-shot task generalization. In *Intl. Conf. on Learning Representations*. OpenReview.net, 2022.

[51] Vincenzo Scotti, Licia Sbattella, and Roberto Tedesco. A primer on seq2seq models for generative chatbots. *ACM Comput. Surv.*, 56(3):75:1–75:58, 2024.

[52] Jennifer Shivers, Joseph Amlung, Natschja Ratanaprayul, Bryn Rhodes, and Paul Biondich. Enhancing narrative clinical guidance with computer-readable artifacts: authoring FHIR implementation guides based on WHO recommendations. *Journal of Biomedical Informatics*, 122:103891, 2021.

[53] M Struys, M Sahinovic, BJ Lichtenbelt, H Vereecke, and A Absalom. Optimizing intravenous drug administration by applying pharmacokinetic/pharmacodynamic concepts. *British journal of anaesthesia*, 107(1):38–47, 2011.

[54] Vasanth Sukumar, Arathi Radhakrishnan, and Venkatesh H Keshavan. Effect site concentration of propofol at induction and recovery of anaesthesia-a correlative dose-response study. *Indian Journal of Anaesthesia*, 62(4):263–268, 2018.

[55] Alberto Tagliaferro, Simone Corboe, and Bruno Guindani. Leveraging llms to automate software architecture design from informal specifications. In *2025 IEEE 22nd International Conference on Software Architecture Companion (ICSA-C)*, pages 291–299. IEEE, 2025.

[56] AJ Thomson, G Morrison, E Thomson, C Beattie, AF Nimmo, and JB Glen. Induction of general anaesthesia by effect-site target-controlled infusion of propofol: influence of pharmacokinetic model and ke0 value. *Anaesthesia*, 69(5):429–435, 2014.

[57] Hugo Touvron et al. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971, 2023.

[58] András Vargha and Harold D Delaney. A critique and improvement of the cl common language effect size statistics of mcgraw and wong. *Journal of Educational and Behavioral Statistics*, 25(2):101–132, 2000.

[59] Marlon Vieira, Xiping Song, Gilberto Matos, Stephan Storck, Rajanikanth Tanikella, and William M. Hasling. Applying model-based testing to healthcare products: preliminary experiences. In Wilhelm Schäfer, Matthew B. Dwyer, and Volker Gruhn, editors, *30th International Conference on Software Engineering (ICSE 2008), Leipzig, Germany, May 10-18, 2008*, pages 669–672. ACM, 2008.

[60] Mengting Wan, Tara Safavi, Sujay Kumar Jauhar, Yujin Kim, Scott Counts, Jennifer Neville, Siddharth Suri, Chirag Shah, Ryen W. White, Longqi Yang, Reid Andersen, Georg Buscher, Dhruv Joshi, and Nagu Rangan. TnT-LLM: Text Mining at Scale with Large Language Models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 5836–5847. Association for Computing Machinery, 2024.

[61] Shuhe Wang, Xiaofei Sun, Xiaoya Li, Rongbin Ouyang, Fei Wu, Tianwei Zhang, Jiwei Li, Guoyin Wang, and Chen Guo. GPT-NER: Named Entity Recognition via Large Language Models. In Luis Chiruzzo, Alan Ritter, and Lu Wang, editors, *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 4257–4275. Association for Computational Linguistics, 2025.

[62] Julia Wiesinger, Patrick Marlow, and Vladimir Vuskovic. Agents. Technical report, Google DeepMind and Google Research, February 2025. Whitepaper.

[63] Hang Wu, Yuanda Zhu, Wenqi Shi, Li Tong, and May D Wang. Fairness artificial intelligence in clinical decision support: Mitigating effect of health disparity. *IEEE Journal of Biomedical and Health Informatics*, 2024.

[64] Danning Xie, Byoungwoo Yoo, Nan Jiang, Mijung Kim, Lin Tan, Xiangyu Zhang, and Judy S. Lee. How Effective are Large Language Models in Generating Software Specifications? In *2025 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 1–12, 2025.

[65] Jingzhi Yu, Jennifer A Pacheco, Anika S Ghosh, Yuan Luo, Chunhua Weng, Ning Shang, Barbara Benoit, David S Carrell, et al. Under-specification as the source of ambiguity and vagueness in narrative phenotype algorithm definitions. *BMC medical informatics and decision making*, 22(1):23, 2022.

[66] Juan Zhai, Yu Shi, Minxue Pan, Guian Zhou, Yongxiang Liu, Chunrong Fang, Shiqing Ma, Lin Tan, and Xiangyu Zhang. C2S: translating natural language comments to formal program specifications. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 25–37. Association for Computing Machinery, 2020.

[67] Wei Zhu, Wenfeng Li, Xing Tian, Pengfei Wang, Xiaoling Wang, Jin Chen, Yuanbin Wu, Yuan Ni, and Guotong Xie. Text2MDT: Extracting Medical Decision Trees from Medical Texts, 2024.

[68] Wei Zhu, Wenfeng Li, Xiaoling Wang, Wendi Ji, Yuanbin Wu, Jin Chen, Liang Chen, and Buzhou Tang. Extracting Decision Trees from Medical Texts: An Overview of the Text2DT Track in CHIP2022. In Buzhou Tang, Qingcai Chen, Hongfei Lin, Fei Wu, Lei Liu, Tianyong Hao, Yanshan Wang, Haitian Wang, Jianbo Lei, Zuofeng Li, and Hui Zong, editors, *Health Information Processing. Evaluation Track Papers*, pages 89–102. Springer Nature, 2023.

A APPENDIX

The EBNF grammar used for output artifacts in our experiments is:

```
Tree        = Node ;
Node        = "{"
                '"actions"' ":" ActionArray
                [ "," '"branches"' ":" BranchArray ]
                [ "," '"else"' ":" Node ]
              "}" ;

ActionArray = "[" [ Action { "," Action } ] "]" ;
Action      = "{"
                '"param"' ":" Param ","
                '"op"' ":" Op ","
                '"value"' ":" Value
              "}" ;
Param       = String | "null" ;
Op          = "inc" | "dec" | "set" | "null" ;
Value       = Number | "null" ;

BranchArray = "[" Branch { "," Branch } "]" ;
Branch      = "{"
                '"guard"' ":" GuardArray ","
                '"then"' ":" Node
              "}" ;

GuardArray  = "[" Guard { "," Guard } "]" ;
Guard       = "{"
                '"metric"' ":" String ","
                '"interval"' ":" Interval
              "}" ;
Interval    = "{"
                '"low"' ":" Number ","
                '"high"' ":" Number
              "}" ;
```