SOP-Bench: Complex Industrial SOPs for Evaluating LLM Agents

Subhrangshu Nandi*
Arghya Datta*
subhrn@amazon.com
arghya.dat@gmail.com
Amazon
Seattle, WA, USA

Rohith Nama*
Udita Patel*
Nikhil Vichare*
rohinama@amazon.com
patudita@amazon.com
Amazon
Seattle, WA, USA

Indranil Bhattacharya*
Shivam Asija
Arushi Gupta
Amazon
Seattle, WA, USA

Giuseppe Carenini
University of British Columbia
Vancouver, BC, Canada

Jing Xu
Shayan Ray
Huzefa Raja
Amazon
Seattle, WA, USA

Aaron Chan
Francesco Carbone
Esther Xu Fei
Gaoyuan Du
Amazon
Seattle, WA, USA

Zuhaib Akhtar
Prince Grover
Sreyoshi Bhaduri
Amazon
Seattle, WA, USA

Weian Chen
Wei Zhang
Amazon
San Jose, CA, USA

Ming Xiong
Amazon
Santa Clara, CA, USA

Harshita Asnani
Amazon
Seattle, WA, USA

Jeetu Mirchandani
Amazon
Seattle, WA, USA

Abstract

LLM-based agents struggle to execute complex, multi-step Standard Operating Procedures (SOPs) that are fundamental to industrial automation. Existing benchmarks fail to capture the procedural complexity and tool orchestration demands of real-world workflows. We introduce **SOP-Bench**, a benchmark of 2,000+ tasks from human expert-authored SOPs across 12 business domains (healthcare, logistics, finance, content moderation, etc.). Using a human-AI collaborative framework, experts crafted authentic SOPs while AI generated artifacts (tools, APIs, datasets), all human-validated, yielding realistic tasks with executable interfaces and ground-truth outputs.

SOP-Bench serves as a research enabler for systematically investigating agent architectures, model capabilities, and deployment considerations across diverse procedural tasks. We demonstrate its utility through illustrative experiments with a subset of frontier models across Function-Calling (FC) and ReAct agents, revealing critical insights. For example, (1) newer models do not guarantee better performance - Claude 4 family outperforms Claude 4.5 family on ReAct tasks (Claude 4 Opus: 72.4% vs. Claude 4.5 Sonnet: 63.3% task success rate), demonstrating that production upgrades require validation; (2) no single model-agent combination dominates: best performances range from 57% to 100% depending on domain. These examples illustrate how SOP-Bench enables isolating and studying specific dimensions of agent performance without costly production experiments. Our goal is not to rank model capabilities or build optimal agents, but to provide a rigorous evaluation framework that enables researchers and practitioners to systematically investigate agent design choices, model selection, and deployment strategies. We release the benchmark at https://github.com/amazon-science/sop-bench.

CCS Concepts

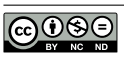
• **Computing methodologies** → **Natural language processing**; **Artificial intelligence**; • **General and reference** → **Evaluation**; **Experimentation**;

Keywords

Agentic AI, Benchmark, LLM Applications, SOP Benchmark, Evaluation, Industry SOPs

ACM Reference Format:
Subhrangshu Nandi, Arghya Datta, Rohith Nama, Udita Patel, Nikhil Vichare, Indranil Bhattacharya, Shivam Asija, Arushi Gupta, Giuseppe Carenini, Jing Xu, Shayan Ray, Huzefa Raja, Aaron Chan, Francesco Carbone, Esther Xu Fei, Gaoyuan Du, Zuhaib Akhtar, Prince Grover, Sreyoshi Bhaduri, Weian Chen, Wei Zhang, Ming Xiong, Harshita Asnani, and Jeetu Mirchandani. 2026. SOP-Bench: Complex Industrial SOPs for Evaluating LLM Agents. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD 2026), August 9–13, 2026, Jeju Island, Republic of Korea.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3770855.3817538

*Primary authors



KDD 2026, Jeju Island, Republic of Korea.

ACM ISBN 979-8-4007-2259-2/2026/08
https://doi.org/10.1145/3770855.3817538

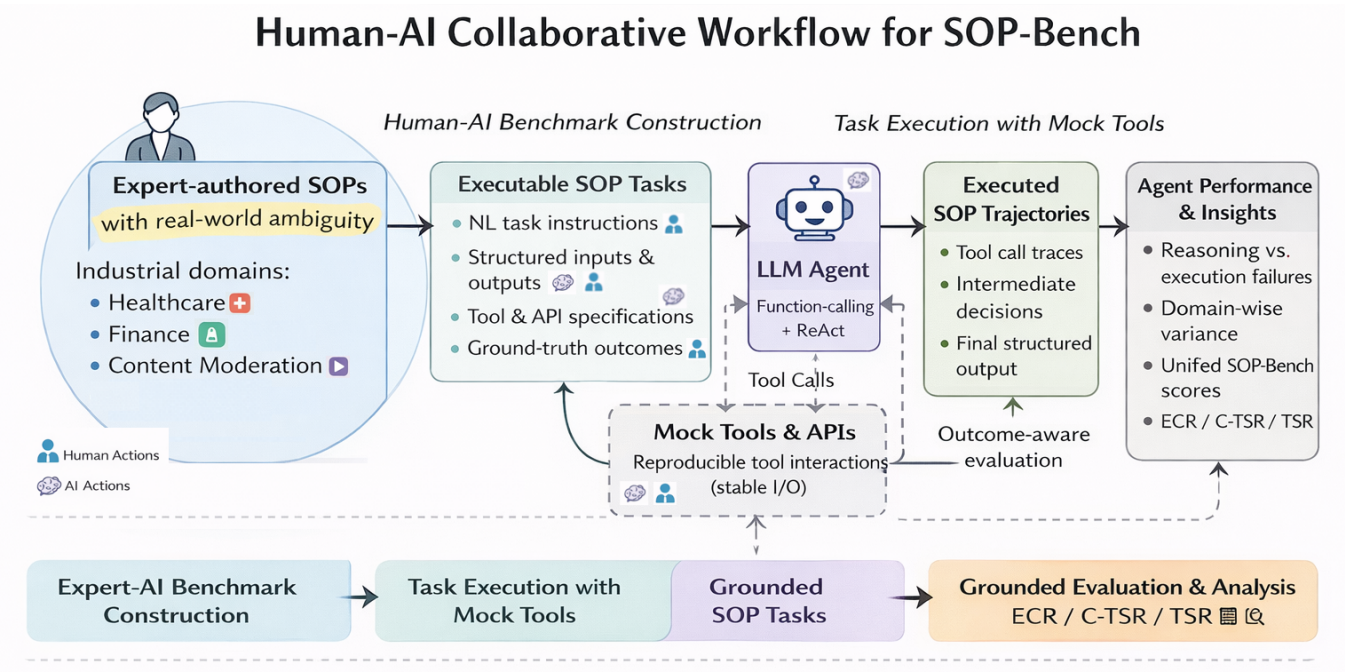


Figure 1: SOP-Bench evaluation overview. Realistic business process SOPs authored by human experts across diverse domains are converted into executable task instances with structured tool/API interfaces and ground-truth outputs. LLM agents (Function-Calling and ReAct) execute tasks via reproducible tool interactions, producing execution trajectories that include tool traces, intermediate decisions, and final structured outputs. SOP-Bench evaluates trajectories using grounded, outcome-aware metrics (ECR, C-TSR, TSR), enabling systematic analysis of agent robustness, domain-specific difficulty, and architecture trade-offs.

1 Introduction

Standard Operating Procedures (SOPs) are central to reliable operations across industries, encoding domain expertise, compliance rules, and decision logic into structured workflows [9, 11]. Recent advances in large language models (LLMs) have sparked growing interest in automating SOP execution using LLM-based agents [7, 13, 31, 36]. However, deploying such agents in production exposes challenges that are not captured by existing benchmarks such as Gorilla [23], API-Bank [16], or ComplexBench [34].

Table 1: Diversity of SOPs in industry, assessed by GPT-4-turbo [21] and validated by human associates with extensive cross-domain SOP review experience.

Criteria	SLUHN	UNTHS
Length	Long (7 pgs)	Short (3 pgs)
Ambiguity	Low	Moderate
Step complexity	High	Low-Mod
Implicit knowledge	High	Low-Mod
Modularity	High	Moderate

These challenges are particularly acute in knowledge operations like verification services, support ticket triaging, and customer support, where SOPs involve ambiguous language, implicit domain knowledge, and complex decision trees [27]. For example, consider the two patient registration SOPs, one from St. Luke's University Health Network (SLUHN)[1], and one from University of North Texas Health Science Center (UNTHS)[2]. Table 1 shows how different two SOPs of the same domain can be.

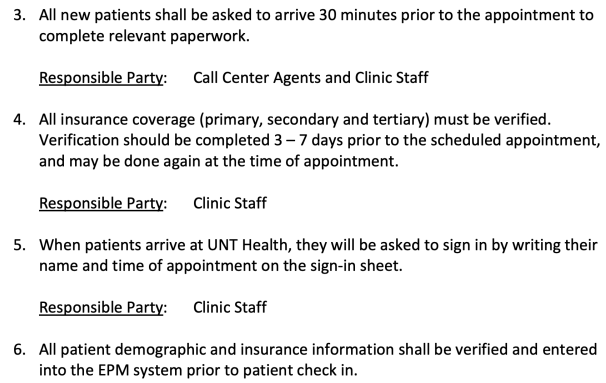
Such business knowledge operations differ fundamentally from traditional process automation tasks [26]. They require contextual interpretation, multi-step information synthesis, and judgment under ambiguity, often with evolving guidelines and subjective classifications [12, 27]. Figure 2 shows a representative example: instructions that are straightforward for human operators with implicit domain knowledge but ambiguous for LLM agents. Existing evaluation frameworks typically isolate individual capabilities—such as tool use [14], planning [36], or instruction following [33]—using clean, synthetic prompts that fail to reflect the messiness of real-world SOPs.

While recent advances in LLMs have enabled sophisticated instruction following and tool use [33, 36], automating SOP execution poses unique challenges that existing benchmarks fail to capture. Current evaluation frameworks primarily focus on isolated capabilities—tool use [14], planning [36], or instruction following [33], using clean, synthetic prompts that sidestep the messiness of real-world procedures.

To address this gap, we introduce **SOP-Bench**, a comprehensive benchmark for evaluating LLM agents on realistic business SOP execution. SOP-Bench is constructed from human expert-authored SOPs spanning twelve diverse domains, including healthcare intake, hazardous goods classification, customer service, content moderation, and financial compliance. Industry experts authored authentic

[1] https://www.slhn.org/-/media/slhn/Research/File/PDF/SOPs-and-Policies/SOP-302Patient-Registration-and-Ongoing-Subject-MgmtV30CLEAN71216.ashx

[2] https://www.unthsc.edu/wp-content/uploads/sites/23/Patient_Registration.pdf


3. All new patients shall be asked to arrive 30 minutes prior to the appointment to complete relevant paperwork.

   Responsible Party: Call Center Agents and Clinic Staff

4. All insurance coverage (primary, secondary and tertiary) must be verified. Verification should be completed 3 – 7 days prior to the scheduled appointment, and may be done again at the time of appointment.

   Responsible Party: Clinic Staff

5. When patients arrive at UNT Health, they will be asked to sign in by writing their name and time of appointment on the sign-in sheet.

   Responsible Party: Clinic Staff

6. All patient demographic and insurance information shall be verified and entered into the EPM system prior to patient check in.

Figure 2: Example of instructions that can be simple for humans but confusing for LLMs. In this excerpt of page 2 of the UNHTS-SOP, we can see that steps 4 and 6 both have instructions on verifying the insurance, but does not have explicit information on how to verify it. Moreover, it is not clear why there are two steps of insurance verifications. For human associates with implicit knowledge of the domains this might not be as ambiguous, while our experiments indicate that it is for LLMs. In our experience, this is a common occurrence in business process SOPs.

procedures reflecting real-world ambiguity, branching logic, and implicit knowledge requirements. To enable reproducible evaluation, we adopt a human–AI collaborative workflow in which AI models generate supporting artifacts (mock tools, APIs, and test datasets) that are subsequently validated by human experts. This process yields over 2,000 executable tasks with structured tool interfaces and grounded outputs that closely mirror production environments.

We make three primary contributions: **(1) Human–AI collaborative benchmark construction framework:** We introduce a scalable methodology where industry experts author authentic SOPs while AI assists in generating executable artifacts, all human-validated for correctness and realism. **(2) A comprehensive benchmark for systematic agent evaluation:** SOP-Bench provides 2,000+ tasks across 12 domains with varying procedural complexity, document length, branching logic, and tool usage, enabling controlled study of agent architectures and foundation models without costly production experimentation. **(3) Empirical insights via illustrative experiments:** Through experiments and ablations on representative frontier models, we demonstrate how SOP-Bench reveals non-obvious trade-offs in architecture choice, reasoning cost, and domain generalization.

Our goal is not merely to rank models, but to establish SOP-Bench as an evaluation substrate for studying agent design, model selection, and deployment strategies under realistic procedural constraints. We release the full benchmark, baseline agents, and evaluation framework at https://github.com/amazon-science/sop-bench, enabling the community to extend SOP-Bench with new domains and contribute back to a shared evaluation ecosystem.

Table 2: SOP-Bench uniquely combines all seven capabilities essential for real-world industrial workflow automation, addressing critical gaps in existing agent benchmarks.

Benchmark	IF	MS	Tool	Amb.	Err.	Dep.	Ind.
AlpacaEval	✓	✗	✗	✗	✗	✗	✗
FollowEval	✓	✗	✗	✗	✗	✗	✗
ComplexBench	✓	✓	✗	✗	✗	✓	✗
InFoBench	✓	✓	✗	✗	✗	✗	✗
Gorilla	✓	✗	✓	✗	✗	✗	✗
API-Bank	✓	✗	✓	✗	✗	✗	✗
AgentBench	✓	✓	✓	✗	✗	✗	✗
PlanBench	✓	✓	✓	✗	✗	✓	✗
ALFWorld	✓	✓	✓	✗	✗	✓	✗
SOP-Maze	✓	✓	✗	✓	✗	✓	✓
SOP-Bench	✓	✓	✓	✓	✓	✓	✓

Legend: IF = Instruction Following; MS = Multi-step Tasks; Tool = Tool/API Integration; Amb. = Real-world Ambiguity; Err. = Error Handling; Dep. = Workflow Dependencies; Ind. = Industrial Context.

2 Related Work

LLM Agents for Planning and Tool Use. LLMs have enabled the emergence of AI agents capable of task planning, reasoning, and tool use. Architectures such as ReAct [36], AutoGPT [35], and LangChain [7] have demonstrated autonomous decision-making via multi-step reasoning and API calls. ToolChain [37] and ToolLLM [24] extend this by integrating dynamic tool execution. However, these systems are mostly evaluated in highly simplistic settings, limiting their applicability to real-world operational workflows.

Instruction Following and Benchmarking. Instruction following ability has been benchmarked through datasets like SUPER-NATURAL-INSTRUCTIONS [33], AlpacaEval [17], and FollowEval [15]. More recent multi-step benchmarks like ComplexBench [34] and InFoBench [25] explore instruction complexity, but typically use machine-formatted inputs that omit the ambiguity and variability of human-authored SOPs.

Tool Use and API-Centric Benchmarks. Benchmarks like Gorilla [23], API-Bank [16] and BENCHAGENTS [6] assess tool invocation and API usage capabilities. However, these focus on isolated tool interactions with minimal procedural context. In contrast, SOP execution involves coordinated tool use across dependent steps, often requiring error handling and state tracking.

Agent Evaluation Frameworks. AgentBench [19] and PlanBench [29] offer general frameworks to evaluate LLM agents on planning and tool use. Yet, their task settings are narrow in scope and lack the procedural structure, conditional logic, and real-world ambiguity typical of industrial SOPs. Similarly, embodied agents are evaluated in ALFWorld [28] and BabyAI [8], but those environments (e.g., a kitchen) differ fundamentally from workflow automation tasks.

SOP Automation and Business Process Modeling. Traditional business process automation relies on rule-based systems and formal process modeling languages [11, 18, 30]. While effective, these require manual formalization of procedures, making them brittle and labor-intensive. Prior work on SOP formalization in organizational contexts [9] has not yet translated into LLM-native solutions. Recently, [13] proposes a novel LLM Classifier-Augmented

Generation (CAG) approach that translates procedures described in natural language into executable workflows. However, their target descriptions are rather short and limited to the domain of data integration. Most importantly, the dataset used to evaluate the approach is not publicly available. Recently, SOP-Maze [32] has published business operations SOPs, but they lack the tools or ground truth data for agent evaluation.

How SOP-Bench Differs SOP-Bench addresses these gaps by offering (1) realistic SOP-style instructions, (2) diverse domain coverage, (3) structured APIs for execution, and (4) evaluation protocols grounded in real-world complexity. Compared to benchmarks like Gorilla that isolate API usage, SOP-Bench evaluates agents on full workflows with inter-dependencies, ambiguity, and error handling requirements. It provides a more comprehensive testbed for assessing agent robustness in enterprise automation settings.

3 Generating Industry-grade SOPs

SOP-Bench is built on a novel human-AI collaborative framework where human domain experts author authentic SOPs while AI models generate supporting artifacts, all subsequently human-validated. This approach ensures procedural authenticity while maintaining reproducibility and scalability. The workflow employs a hierarchical prompting strategy using Anthropic's Claude 3.5 Sonnet v2 to refine human-authored SOPs for consistency and completeness, and to generate associated artifacts (tools, APIs, datasets) that capture real-world nuances: industry jargon, interdependent logic, implicit domain knowledge, ambiguous instructions, and cross-referenced content (Figure 3).

3.1 Human-AI Collaborative Workflow

Human Expert Input: Domain experts provide two critical inputs that define the benchmark scope:
Business Task: The first version of an industrial use-case SOP
Task Context: Procedural steps required to process requests successfully. For patient intake, this includes validating patient information, insurance details, medical history, and risk assessment.

- **Business Task**: `SOP outlining steps for registering new patients`
- **Task Context**: `Validate insurance eligibility, prescription coverage, ...`

3.1.1 Dataset Schema *Generation (LLM + Human).* The first step is to create a structured dataset schema via one-shot prompting. This schema specifies input parameters, decision points, compliance checks, and expected outcomes—acting as a semantic scaffold that minimizes hallucinations and ensures consistency. Human domain experts validate that the schema captures essential task components, including implicit knowledge and domain constraints. It uses the SOP and task context to generate the dataset schema. For example:

- **Schema Entry**: **smoking_status** – `string`, `Choices = Never, Former, Current`
- **Without Schema** ✗: `Calculate smoking risk using preferred methodology`
- **With Schema** ✓: `Calculate smoking risk (Never, Former, Current)`

3.1.2 SOP Refinement *(Human-Authored, AI-Refined).* Our workflow next uses the first version of the SOP, along with *<Business Task, Task Context, Dataset Schema>* to refine the SOP and make sure it is consistent with the data schema. The resulting SOP outlines procedure objectives, prerequisites, input requirements, detailed instructions, and decision logic in natural language. Human oversight validates domain standards, task logic, and corrects inconsistencies, ensuring SOPs are executable and contextually appropriate for industrial use.

3.1.3 Dataset *Generation (AI-Generated, Human-Validated).* Using *<Business Task, Task Context, Dataset Schema, SOP>*, AI generates datasets representing full procedural context: structured inputs/outputs for every tool invocation, task parameters, decision pathways, and final outcomes. Human experts validate that each datapoint's logic is unambiguous, data types are appropriate, and the dataset accurately reflects SOP logic. The dataset includes challenging scenarios: positive/negative decision pathways, edge conditions, and task failure scenarios to test agent reasoning, compliance, error-handling, and tool selection.

3.1.4 API & Tool Specs *Generation (AI-Generated, Human- Validated).* Using *<Task Context, Dataset, SOP>*, our workflow generates APIs specifying required inputs (e.g., `patient_name`) and expected outputs (e.g., `patient_id`), along with corresponding tool specifications. Human experts validate that APIs and ToolSpecs align with the dataset schema and access only relevant columns. Example:

- **API Spec**: **getPatientID** – retrieves patient identifier from name
- **Without API** ✗: `def getPatientID(self): return {}`
- **With API** ✓: `def getPatientID(self, patient_name): return self.patientDB.lookupIDByName(patient_name)`

3.1.5 Tools Code *Generation (AI-Generated, Human-Validated).* Using *<Dataset, APIs>*, our workflow next generates executable tool code reflecting procedural logic. The dataset and API specs provide input-output contracts and data characteristics, ensuring code aligns with operational requirements. Human experts validate code correctness and consistency by executing them.

3.2 Human Authoring and Validation

Domain experts perform five critical validation steps: (1) **SOP Authoring**: experts draft initial SOPs based on real industrial procedures, providing business context and task requirements; (2) **Schema Validation**: experts verify that AI-generated schemas capture essential task components, implicit domain knowledge, and compliance constraints; (3) **SOP Refinement**: experts review AI-refined SOPs for domain accuracy, logical consistency, and alignment with industrial standards; (4) **Dataset Validation**: experts validate **each** generated datapoint for logical correctness, appropriate data types, and coverage of edge cases; (5) **API & Code Validation**: experts verify that generated APIs and executable code align with dataset schemas and operational requirements.

To quantify benchmark difficulty, three domain experts independently rated each SOP across three dimensions: ease of understanding, ambiguity/implicit knowledge requirements, and reasoning complexity (scale 1-10). These ratings, averaged to produce $\mathbb{C}_H$ in Table 3, reflect human-perceived complexity.

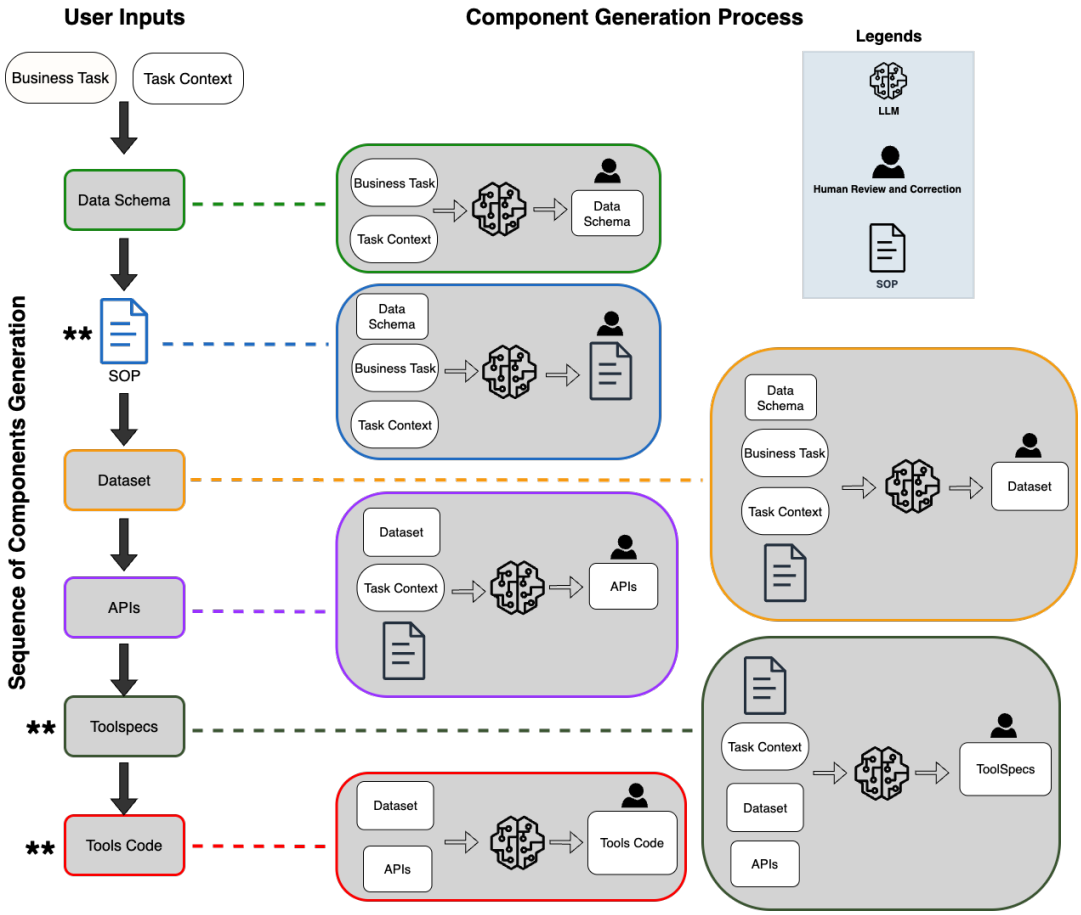


Figure 3: SOP-Bench Generation Workflow: Human experts author SOPs and validate all artifacts; AI generates tools, APIs, and datasets. ** denotes post-generation complexity introduction.

3.3 SOP Descriptions

SOP-Bench consists of 12 diverse SOPs instantiated in 2,411 tasks representing realistic industrial procedures. Each SOP tests different agent capabilities: complex decision-making, tool usage, and structured output generation across business modalities including classification (email intent), structured verification (business verification), industrial processes (aircraft inspection), and generic assignments (annotation, content flagging, customer service). Table 3 shows task count (n), API count (t), SOP token count (τ), and complexity ratings by three human domain experts ($\mathbb{C}_H$) and Claude 3.5 v2 ($\mathbb{C}_{llm}$).

3.3.1 Controlled Complexity Variation. To evaluate how procedural complexity affects agent performance, we include two versions of the referral abuse detection SOP. Version 1 (v1) implements a foundational fraud detection workflow with 3 tools and a simple three-outcome decision model based on current account indicators. Version 2 (v2) introduces significantly greater complexity through temporal pattern analysis, historical violation context, and risk-based enforcement, requiring 6 tools and producing 7 graduated enforcement actions. This controlled complexity variation enables systematic evaluation of how agents handle increasingly sophisticated reasoning requirements.

3.3.2 Real-World Complexity in SOP-Bench. To mirror authentic industrial conditions, human experts and AI collaboratively introduce controlled complexity at multiple levels across different SOPs. At the **SOP-level**, procedures contain branching logic, redundant details, tangential information, and ambiguous instructions that agents must filter—reflecting the reality that real-world SOPs are rarely perfectly structured. At the **tool-level**, we include semantically similar but functionally redundant tools to challenge agents' critical evaluation and tool selection capabilities. At the **data-level**, we introduce distractor variables that appear relevant but are not required for task completion.

Specific examples include: (1) **Video Annotation** and **Video Classification** contain 26 and 25 tools respectively, with multiple distractor tools that perform similar functions (e.g., different object detection methods, overlapping segmentation approaches), requiring agents to identify the correct tool sequence from numerous plausible alternatives; (2) **Patient Intake** includes distractor input variables such as patient demographic details and medical history fields that are provided but not necessary for certain decision

Table 3: Characteristics of Benchmark SOPs in SOP-Bench. n = number of tasks per SOP; t = number of tools available; τ = token count, $\mathbb{C}_{\mathbb{H}}$ = human-perceived complexity, average of three domain experts of three dimensions, ease of understanding, ambiguity or implicit domain knowledge and reasoning complexity, on a scale of 1 - 10; $\mathbb{C}_{llm}$ = complexity estimated by Claude 3.5 (scale 1-10, calibrated against Berkeley function calling tasks as baseline 2)

SOP Topic	Domain	n	t	τ	$\mathbb{C}_H$	$\mathbb{C}_{llm}$
Content Flagging	Content Moderation	226	4	850	7.67	9
Customer Service	Support	208	10	1457	9	8
Dangerous Goods	Supply Chain	327	5	775	5	7
Aircraft Inspection	Transportation	150	7	834	6.67	9
Email Intent	Retail Seller Mgmt	122	4	766	5.33	7
Know Your Business	Finance	122	8	1359	8.33	9
Patient Intake	Healthcare	90	6	753	4.33	7
Video Annotation	Autonomous Driving	168	26	4492	9.67	10
Video Classification	Media	198	25	1249	8.33	9
Warehouse Package Inspection	Logistics	200	12	653	4.67	9
Referral Abuse v1	Trust & Safety	200	3	1502	6.33	7
Referral Abuse v2	Trust & Safety	200	6	2576	8.67	9
Traffic Spoofing	Fraud Detection	200	6	854	6.83	8
Average		185	9	1394	6.96	8.31
Total		2411	122	18120		

pathways, testing whether agents can distinguish essential from supplementary information; (3) **Content Flagging** and **Customer Service** embed ambiguous conditional logic where threshold values and decision criteria require careful interpretation of SOP text rather than explicit enumeration. These interventions, validated through human oversight, rigorously assess tool selection behavior, information filtering, and reasoning under ambiguity.

4 Agents for automating SOPs

To evaluate the effectiveness of LLM-based agents in executing industry-grade SOPs, we implement two complementary agent architectures in SOP-Bench: (a) **Function Calling (FC) Agent** and (b) **ReAct Agent**. These implementations serve as baseline agents to demonstrate SOP-Bench's evaluation capabilities and are not intended to represent optimal architectures. We encourage the research community to develop and publish more sophisticated agent designs using SOP-Bench as the evaluation framework. Both baseline agents utilize LLM's native tool-calling interface to dynamically invoke tools/APIs during task execution.

FC Agent is a prompt-driven, lightweight execution engine leveraging native tool calling capabilities of LLMs. It accepts SOP text and task input, maintains a conversation loop with the LLM, and processes tool calls iteratively (up to 10 iterations). When the model triggers a tool use event, the agent executes the corresponding function via a ToolManager and returns results to the model. Final outputs are enclosed in XML tags for structured parsing.

ReAct Agent is built on a custom ReAct framework implementation [36], designed for SOPs with complex branching logic and multi-step reasoning. It interleaves reasoning traces (Thoughts) with tool calls (Actions), feeding Observations back into the reasoning loop for up to 15 iterations. The agent enforces that at least one tool is executed before conclusions, maintaining comprehensive execution traces including intermediate steps and tool call records.

We evaluate both agents on SOP-Bench using standardized scripts that: (1) load SOP documents and structured task datasets; (2) convert each task into natural language instructions; and (3) execute agents while logging outputs, tool traces, and reasoning steps. To simulate API calls, each dataset includes precomputed inputs and outputs for every tool call, stored as columns. These mocks replace live APIs to enable stable, reproducible evaluation without runtime variability. In deployment, they would be swapped for real APIs.

4.1 Evaluation Methodology

We evaluate FC-Agent and ReAct-Agent on SOP-Bench using three inputs: (1) Task, (2) SOP (instructions), and (3) ToolSpecs. The outputs include (1) *intermediate tool outputs*, (2) *final response*, and (3) *trace of tool use and reasoning*. We define: n (total number of tasks per SOP), n_c (number of tasks per SOP marked complete by the agent), and n_a (number of correctly completed tasks per SOP). Evaluation metrics are: (1) Execution Completion Rate (*ECR*): proportion of tasks the agent marked as complete, $ECR = \frac{n_c}{n}$; (2) Conditional Task Success Rate (*C-TSR*): fraction of completed tasks that match the ground truth, $C\text{-}TSR = \frac{n_a}{n_c}$; and (3) Task Success Rate (*TSR*): overall accuracy, $TSR = \frac{n_a}{n}$. Though this work reports only *ECR* and *TSR*, future research should incorporate tailored metrics to better assess steps planning, tool mapping, tool calling accuracies, to better understand agent performance bottlenecks. We welcome the research community to contribute such agent evaluation metrics to SOP-Bench.

For all evaluations, we have kept the LLM parameters temperature at 0.5 and output token limits at 8000.

5 Results

5.1 Results and Insights

We evaluate the two agents, FC and ReAct, with 11 different LLMs in our experiments: Llama-3.3-70B-Instruct [20], GPT-OSS-120B [22], DeepSeek-R1 [10], Claude 3.5 Sonnet v2 [1], Claude 3.7 Sonnet [2], Claude 4 Sonnet [3], Claude 4 Opus [3], Claude 4.1 Opus [3], Claude Sonnet 4.5 [5], and Claude Opus 4.5 [4]. These models were selected based on availability within our organization and their known strong performance on agentic tasks, explaining the focus on larger, frontier models rather than smaller alternatives.

5.1.1 LLM–Agent Architecture Must Be Task Dependent. Our evaluation shows that no single agent architecture-model combination consistently dominates across SOP-Bench, highlighting the need for task-dependent architecture selection in SOP automation.

Table 4: Claude 4 Opus: Function Calling vs ReAct Agent Performance, ranked in descending order of ReAct TSR.

SOP Topic	Function Calling				ReAct			
	ECR	C-TSR	TSR	Time (s)	ECR	C-TSR	TSR	Time (s)
Patient Intake	1.00	0.92	0.92	70.33	1.00	1.00	1.00	102.28
Email Intent	1.00	0.98	0.98	25.24	0.99	0.99	0.98	31.26
Referral Abuse Detection Easy	1.00	0.95	0.95	53.99	1.00	0.97	0.97	66.45
Aircraft Inspection	1.00	0.45	0.45	83.11	1.00	0.97	0.97	87.46
Referral Abuse Detection Hard	1.00	0.98	0.98	81.79	0.99	0.96	0.96	89.16
Video Classification	1.00	0.79	0.79	78.69	1.00	0.94	0.94	62.89
Dangerous Goods	1.00	0.73	0.73	41.15	1.00	0.83	0.83	28.02
Warehouse Package Inspection	1.00	0.56	0.56	47.50	1.00	0.65	0.65	76.02
Traffic Spoofing Detection	1.00	0.79	0.79	25.23	0.99	0.61	0.60	130.20
Customer Service	1.00	0.56	0.56	72.83	1.00	0.56	0.56	145.43
Video Annotation	1.00	0.37	0.37	71.04	1.00	0.38	0.38	73.97
Know Your Business	0.99	0.44	0.43	159.84	1.00	0.31	0.31	110.50
Content Flagging	1.00	0.36	0.36	49.93	0.99	0.28	0.27	138.08
Average	**1.00**	**0.68**	**0.68**	**66.21**	**1.00**	**0.73**	**0.72**	**87.82**
Standard Error ϵ		**±0.07**	**±0.07**	**±9.57**		**±0.08**	**±0.08**	**±10.30**

Using Claude 4 Opus as a controlled comparison, Table 4 contrasts Function-Calling (FC) and ReAct agents across all 13 SOPs. ReAct achieves a higher average Task Success Rate (TSR) than FC (72% vs. 68%), but at a significant latency cost: 87.82s ± 10.30s per task compared to 66.21s ± 9.57s for FC, a 33% increase. Moreover, ReAct outperforms FC on only 8 of 13 SOPs, indicating that average gains obscure strong task-level variation.

Architecture–Model Co-Design Is Non-Monotonic. Performance does not scale monotonically with stronger models, particularly for reasoning-centric architectures. FC agents show modest, consistent improvements (Claude 3.5 v2: 65.8% → Claude 4.5 Sonnet: 67.5% TSR), while ReAct agents degrade: Claude 3.5 v2 achieves 65.4% TSR, Claude 4 Sonnet improves to 68.7%, but Claude 4.5 Sonnet drops to 63.3% (−2.1 points). These results show that newer foundation models are not automatically better for existing agent architectures, and naive upgrades can silently reduce task success.

Task Characteristics Dominate Aggregate Metrics. Aggregate averages mask clear task-specific reversals. Across SOPs, FC slightly outperforms ReAct on average (65.9% vs. 61.1% TSR), yet individual tasks exhibit distinct architectural preferences. Dangerous Goods classification shows near parity (FC: 69.1%, ReAct: 69.8%), while Customer Service reveals identical failure modes (FC: 47.8%, ReAct: 47.7%). Other SOPs strongly favor one architecture, with ReAct excelling in multi-step inspection workflows and FC performing better in deterministic, tool-driven procedures.

Overall, these findings demonstrate that agent architecture must be selected based on SOP structure, tool interaction patterns, and latency constraints rather than global averages. SOP-Bench enables systematic evaluation of these trade-offs, supporting principled architecture selection for real-world procedural workflows.

5.1.2 Domain Variance Exposes Generalization Limits. Agent performance on SOP-Bench varies substantially across procedural domains, exposing clear limits in cross-domain generalization. Table 5 shows the top two model–agent combinations per SOP and confirms that no single configuration dominates across tasks. Best-case performance ranges from near-perfect accuracy on Patient Intake (100% TSR) to only 57% on Know Your Business, underscoring the necessity of domain-specific evaluation. Aggregated results in Table 6 further illustrate this variability: the FC agent achieves its highest average performance with Claude 4 Opus (68.2% TSR), while ReAct performs best with Claude 4 Opus as well (73.8% TSR). These rankings reflect a fixed agent configuration per model; further agent–model co-optimization could alter absolute performance but does not change the observed domain sensitivity.

Table 5: Top two performing model-agent combinations per SOP, ranked by Task Success Rate (TSR).

SOP Topic	Best			Second Best		
	Model	Agt	TSR	Model	Agt	TSR
Patient Intake	Claude 4.1 Opus	ReAct	1.00	Claude 4.1 Opus	FC	1.00
Aircraft Inspection	Claude 3.7 Sonnet	ReAct	0.99	Claude 4.5 Opus	ReAct	0.97
Email Intent	Claude 4 Sonnet	ReAct	0.99	Claude 4.1 Opus	FC	0.98
Referral Abuse Easy	Claude 3.5 v2 Sonnet	ReAct	0.98	Claude 3.5 v2 Sonnet	FC	0.97
Referral Abuse Hard	Claude 4 Opus	FC	0.98	Claude 4.5 Opus	FC	0.97
Video Classification	Claude 4 Sonnet	FC	0.95	Claude 4 Sonnet	ReAct	0.95
Dangerous Goods	Claude 4 Sonnet	FC	0.87	Claude 4.1 Opus	ReAct	0.83
Traffic Spoofing	Claude 4.5 Sonnet	FC	0.86	Claude 4 Sonnet	FC	0.79
Customer Service	Llama 3.3 70B	ReAct	0.79	Claude 4.5 Opus	FC	0.71
Warehouse Inspection	Claude 4.1 Opus	ReAct	0.69	Claude 4 Opus	ReAct	0.65
Content Flagging	Deepseek R1	ReAct	0.60	Claude 3.5 v2 Sonnet	ReAct	0.60
Video Annotation	GPT OSS 120B	ReAct	0.58	Deepseek R1	ReAct	0.58
Know Your Business	Claude 4.5 Opus	ReAct	0.58	Claude 4.5 Haiku	FC	0.57

Despite strong average performance for select models, domain-level results reveal a pronounced generalization gap. The easiest SOPs—Referral Abuse Detection (Easy) (94.3% average TSR), Email Intent (88.7%), and Patient Intake (88.1%)—are over three times

Table 6: Average Task Success Rate (TSR) by model and agent type, averaged across all SOPs. FC agent was only evaluated with Claude models due to its design for native tool calling. ReAct agent was evaluated across all models to compare reasoning capabilities.

ReAct Agent		FC Agent	
Model (rank)	**TSR**	**Model (rank)**	**TSR**
Claude 4 Opus (1)	0.724	Claude 4 Opus (1)	0.682
Claude 4.1 Opus (2)	0.694	Claude 4 Sonnet (2)	0.678
Claude 4 Sonnet (3)	0.687	Claude 4.5 Sonnet (3)	0.675
Claude 4.5 Opus (4)	0.683	Claude 4.5 Opus (4)	0.667
Claude 3.5 v2 Sonnet (5)	0.654	Claude 4.1 Opus (5)	0.664
Claude 4.5 Sonnet (6)	0.633	Claude 3.5 v2 Sonnet (6)	0.658
Llama 3.3 70B (7)	0.565	Claude 4.5 Haiku (7)	0.624
Deepseek R1 (8)	0.557	Claude 3.7 Sonnet (8)	0.623
Claude 4.5 Haiku (9)	0.522		
Claude 3.7 Sonnet (10)	0.518		
GPT OSS 120B (11)	0.484		

easier than the most challenging tasks, including Video Annotation (27.8%), Aircraft Inspection (33.4%), Content Flagging (38.0%), and Warehouse Inspection (43.5%), yielding a 3.4× performance range across SOPs.

Harder SOPs also exhibit greater performance variance across models. Video Annotation shows the highest standard deviation in TSR (13.8%), indicating brittle and inconsistent agent behavior in long-horizon, tool-intensive workflows. Overall, these results demonstrate that success on one domain does not reliably transfer to others, validating SOP-Bench's design goal of capturing orthogonal dimensions of procedural reasoning, tool use, and constraint adherence.

5.2 Ablation studies

5.2.1 Tool selection under registry bloat. To investigate the impact of tool registry size on agent performance, we conducted a controlled ablation study on the Video Annotation SOP using Claude Sonnet 4.5. We evaluated agent(react) performance under two conditions: (1) Bloated Registry: 26 tools (6 task-relevant + 20 distractor tools), and (2) Minimal Registry: 6 tools (task-relevant only). The minimal registry condition achieved 37% TSR compared to 20.8% TSR in the bloated condition, representing a 1.7x improvement. This suggests tool registry bloat could be a performance bottleneck: introducing 20 distractor tools reduced success rates by 16.2 percentage points despite agents having access to all necessary tools. Task-specific tool pruning may be necessary for practical deployment.

5.2.2 Controlled Complexity Variation in Referral Abuse Detection. We conduct a controlled ablation on the Referral Abuse Detection SOP by comparing its *Easy* and *Hard* variants, which share identical task objectives, inputs, and tool interfaces but differ in logical complexity. As shown in Table 4, Claude 4 Opus maintains similarly high TSR on both variants for both FC and ReAct agents, indicating robustness to increased procedural complexity. However, this accuracy comes at a clear reasoning cost: execution time increases by 51.5% for the FC agent (53.99s → 81.79s) and by 34.2% for the ReAct agent (66.45s → 89.16s) on the *Hard* SOP.

5.2.3 Long-Context Extraction, Not Conditional Reasoning, could limit Agent Performance. We observe variability in how context length and branching complexity affect performance. Long-context SOPs (e.g., Video Annotation: 4,492 tokens, 8 sequential steps, 2 branching decision points) show lower TSR (0.28), while high-branching SOPs (e.g., Referral Abuse Detection Hard: 2576 tokens, 28 sequential steps, 12 decision points) maintain high TSR (0.84). However, these SOPs differ in multiple dimensions (tool count, domain), preventing definitive conclusions. Further investigation with controlled SOP variants is needed to isolate the independent effects of context length versus branching complexity.

6 Conclusions and Next Steps

We present SOP-Bench, a comprehensive benchmark of 2,000+ executable tasks derived from human expert-authored SOPs across 12 industrial domains. SOP-Bench enables systematic evaluation of LLM agents under realistic procedural constraints, supporting principled investigation of agent architectures, model capabilities, and deployment strategies. Our baseline experiments across representative frontier models validate a central finding: **agent performance is highly task- and context-dependent, making rigorous evaluation on realistic SOPs essential prior to production deployment**.

Our results show that newer models do not necessarily yield better outcomes (e.g., Claude 4 family: 73.8% vs. Claude 4.5 family: 63.3% average TSR on ReAct), and that no single model–agent combination dominates across domains (TSR ranging from 57% to 100%). These findings illustrate how SOP-Bench can be used to analyze architecture–model interactions, surface performance bottlenecks, and inform deployment decisions without costly production experimentation. Rather than serving as a leaderboard, SOP-Bench provides an evaluation substrate for targeted scientific inquiry. It enables researchers to study procedural ambiguity, tool selection, and complexity scaling, while allowing practitioners to validate architecture choices against domain-specific requirements before deployment.

Looking ahead, we plan to extend SOP-Bench along several dimensions, while actively inviting community participation. We will introduce controlled variants of the same SOP with increasing complexity, incorporate multimodal instructions such as images and tables, and model hierarchical SOPs with nested structure and context switching. Beyond these extensions, we invite the community to leverage our human–AI collaborative methodology to author new SOPs, generate executable artifacts, and contribute domain-specific tasks back to SOP-Bench. This enables the benchmark to evolve organically, expand to new industries, and serve as a shared evaluation substrate for studying agent behavior.

We release the complete benchmark, baseline agents, and evaluation framework at https://github.com/amazon-science/sop-bench.

References

[1] Anthropic. 2024. Claude 3.5 Sonnet. https://www.anthropic.com/news/claude-3-5-sonnet.
[2] Anthropic. 2025. Claude 3.7 Sonnet and Claude Code. https://www.anthropic.com/news/claude-3-7-sonnet.
[3] Anthropic. 2025. Introducing Claude 4. https://www.anthropic.com/news/claude-4.

[4] Anthropic. 2025. Introducing Claude Opus 4.5. https://www.anthropic.com/news/claude-opus-4-5.

[5] Anthropic. 2025. Introducing Claude Sonnet 4.5. https://www.anthropic.com/news/claude-sonnet-4-5.

[6] Natasha Butt, Varun Chandrasekaran, Neel Joshi, Besmira Nushi, and Vidhisha Balachandran. 2024. Benchagents: Automated benchmark creation with agent interaction.

[7] Harrison Chase. 2022. Langchain. https://github.com/hwchase17/langchain.

[8] Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. 2018. Babyai: First steps towards grounded language learning with a human in the loop.

[9] Suzanne De Treville, John Antonakis, and Norman M. Edelson. 2009. Standard operating procedures: A novel perspective. *Business Horizons* 52, 6 (2009), 563–572.

[10] DeepSeek. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. https://api-docs.deepseek.com/news/news250120.

[11] Marlon Dumas, Marcello La Rosa, Jan Mendling, and Hajo A. Reijers. 2018. *Fundamentals of business process management*. Springer.

[12] Mitchell L. Gordon, Michelle S. Lam, Ji Su Park, Kshitij Patel, Jeff Hancock, Tatsunori Hashimoto, and Michael S. Bernstein. 2022. Jury learning: Integrating dissenting voices into machine learning models. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. 1–19.

[13] Thomas Gschwind, Shramona Chakraborty, Nitin Gupta, and Sameep Mehta. 2025. Classifier-Augmented Generation for Structured Workflow Prediction. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*. 1227–1238.

[14] Yue Huang, Jiawen Shi, Yuan Li, Chenrui Fan, Siyuan Wu, Qihui Zhang, Yixin Liu, Pan Zhou, Yao Wan, Neil Zhenqiang Gong, and Lichao Sun. 2024. Metatool benchmark for large language models: Deciding whether to use tools and which to use. In *The Twelfth International Conference on Learning Representations (ICLR)*. URL: https://openreview.net/forum?id=R0c2qtalgG.

[15] Yimin Jing et al. 2023. Followeval: A multi-dimensional benchmark for assessing the instruction-following capability of large language models. *arXiv preprint arXiv:2311.09829* (2023).

[16] Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023. API-bank: A comprehensive benchmark for tool-augmented LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 3102–3116.

[17] Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. AlpacaEval: An Automatic Evaluator of Instruction-following Models.

[18] Ann Lindsay, Denise Downs, and Ken Lunn. 2003. Business processes—attempts to find a definition. *Information and software technology* 45, 15 (2003), 1015–1019.

[19] Xiao Liu, Hao Chen, Hanchen Chen, et al. 2023. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688* (2023).

[20] Meta. 2024. Meta. Introducing llama 3.1: Our most capable models to date. https://ai.meta.com/blog/meta-llama-3-1/.

[21] OpenAI. 2024. Chatgpt (april 2024 version). https://openai.com/chatgpt.

[22] OpenAI. 2025. Introducing gpt-oss. https://openai.com/index/introducing-gpt-oss/.

[23] Kartikeya Patil, Yuan Tian, Xiangning Lin, Chen Lin, Kevin Lin, Xinying Liu, Eric Xing, Dawei Lu, Nazneen Rajani, Ranjit Krishnan, et al. 2023. Gorilla: Large language model connected with massive apis. In *Advances in Neural Information Processing Systems*.

[24] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789* (2023).

[25] Yiwei Qin, Xiang Ren, et al. 2024. Infobench: Evaluating instruction following ability in large language models. *arXiv preprint arXiv:2401.03601* (2024).

[26] Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 4902–4912.

[27] Sarah T. Roberts. 2019. *Behind the Screen: Content Moderation in the Shadows of Social Media*. Yale University Press.

[28] Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Cote, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2020. Alfworld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*.

[29] Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. 2023. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. *Advances in Neural Information Processing Systems* 36 (2023), 38975–38987.

[30] Wil M. P. Van Der Aalst, Arthur H. M. Ter Hofstede, Bartek Kiepuszewski, and Alistair P. Barros. 2003. Workflow patterns. *Distributed and parallel databases* 14, 1 (2003), 5–51.

[31] Feng Wang et al. 2023. Autogpt: An autonomous gpt-4 based agent system. *arXiv preprint arXiv:2304.03277* (2023).

[32] Jiaming Wang, Zhe Tang, Yilin Jin, Peng Ding, Xiaoyu Li, and Xuezhi Cao. 2025. SOP-Maze: Evaluating Large Language Models on Complicated Business Standard Operating Procedures. *arXiv preprint arXiv:2510.08942* (2025).

[33] Yizhong Wang et al. 2022. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. *arXiv preprint arXiv:2204.07705* (2022).

[34] Bosi Wen, Shaohan Li, Pengcheng He, Weizhu Liu, et al. 2024. Benchmarking complex instruction-following with multiple constraints composition. *arXiv preprint arXiv:2407.03978* (2024).

[35] Hui Yang, Sifu Yue, and Yunzhong He. 2023. Auto-gpt for online decision making: Benchmarks and additional opinions.(2023). *URL https://arxiv. org/abs/2306.02224* (2023).

[36] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629* (2023).

[37] Yuchen Zhuang, Xiang Chen, Tong Yu, Saayan Mitra, Victor Bursztyn, Ryan A. Rossi, Somdeb Sarkhel, and Chao Zhang. 2023. Toolchain*: Efficient action space navigation in large language models with a* search. *arXiv preprint arXiv:2310.13227* (2023).

Appendix

Use of Generative AI

This work employed generative AI models as part of a human-AI collaborative framework for benchmark construction. Specifically, we used Anthropic's Claude 3.5 Sonnet v2 for the following purposes:

Dataset Schema Generation: Claude 3.5 Sonnet v2 generated structured dataset schemas from human-authored SOPs and task contexts using one-shot prompting. These schemas specified input parameters, decision points, and expected outcomes.

SOP Refinement: The model refined human-authored SOPs for consistency and completeness, ensuring alignment with dataset schemas while preserving domain-specific procedural logic.

Artifact Generation: Claude 3.5 Sonnet v2 generated supporting artifacts including: (1) synthetic datasets representing full procedural contexts with diverse decision pathways and edge cases, (2) API specifications defining tool inputs and outputs, (3) tool specifications for agent interaction, and (4) executable tool code implementing procedural logic.

Complexity Assessment: Claude 3.5 v2 provided complexity ratings for each SOP on a 1-10 scale, calibrated against Berkeley function calling tasks as baseline.

Human Validation: All AI-generated content underwent rigorous human validation by domain experts. Human experts: (1) authored all original SOPs based on real-world industrial procedures, (2) validated dataset schemas for completeness and domain accuracy, (3) reviewed and corrected SOP refinements for procedural correctness, (4) validated datasets for logical consistency and appropriate data types, (5) verified API and tool specifications aligned with dataset schemas, (6) executed and validated tool code for correctness, and (7) introduced controlled complexity to mirror real-world conditions. The human-AI collaborative approach ensured procedural authenticity while maintaining reproducibility and scalability.

Evaluation: The benchmark evaluation itself used 11 different LLMs (Claude 3.5 v2 Sonnet, Claude 3.7 Sonnet, Claude 4 Sonnet, Claude 4 Opus, Claude 4.1 Opus, Claude 4.5 Haiku, Claude 4.5 Sonnet, Claude 4.5 Opus, DeepSeek R1, GPT OSS 120B, Llama 3.3 70B) as the agents being evaluated, not as tools for analysis or writing.

Manuscript Writing: Claude 4.5 Sonnet was used to correct grammatical errors in the manuscript. All technical content, experimental design, analysis, and conclusions were authored by the human researchers.

Main Image: The main figure of the paper, which presents an overview of the Human–AI Collaborative workflow for SOP-Bench, was initially generated using ChatGPT, with the paper content provided as grounding material. Since the automatically generated figure did not fully capture the required structural and cosmetic refinements, the authors subsequently refined the figure manually to accurately represent the workflow, clearly delineate the Human and AI steps, and ensure faithful depiction of the overall process.

A Technical Details on Agents

A.1 FC Agent (Function-Calling Agent)

This is a prompt-driven, lightweight execution engine leveraging the native tool calling capability of LLMs (e.g., Claude via AWS Bedrock). It accepts an *SOP text* and a structured *task input*, dynamically generates prompts using a parameterized template engine, and maintains a conversation loop with the LLM throughout execution. During agent initialization, it registers external tools by converting them from OpenAPI-style tool specifications into Bedrock's function-calling format. During execution, when the model triggers a *tool_use* event, the agent extracts the relevant tool name and parameters, executes the corresponding function via a domain-specific *ToolManager*, and delivers the output to the model through a *tool_result* message. The agent implements an iterative conversation pattern with a maximum of 10 iterations, where each iteration processes tool calls sequentially until no further tool uses are detected. Final outputs are enclosed in XML-style tags (e.g., <*final_decision*>) and are parsed into structured formats for downstream evaluation. This architecture emphasizes traceability, minimal supervision, and broad applicability across various SOP domains. Note that this agent is incompatible with Amazon Nova models, which do not support native function calling.

A.2 ReAct Agent

Built on a custom implementation of the ReAct framework [36], this agent is designed to handle SOPs involving complex branching logic, intermediate reasoning, and decision pathways. It ingests the *SOP text*, structured *task input*, and tool specifications, similar to the FC Agent. Tools are dynamically discovered from the ToolManager and their descriptions are formatted with detailed parameter information including types, requirements, and allowed values. The ReAct Agent interleaves reasoning traces (*Thoughts*) with explicit tool calls (*Actions*), which are executed via the ToolManager. The resulting *Observations* are fed back into the reasoning loop, enabling iterative planning and dynamic adaptation to SOP-specific control flows. The agent implements strict enforcement mechanisms to prevent premature final answers without tool usage, ensuring that at least one tool is executed before providing conclusions. Response parsing uses regular expressions to extract Thought, Action, Action Input, and Final Answer components, with robust error handling for malformed JSON inputs. This process continues for up to 15 iterations until a conclusive *Final Answer* is reached, marked using tags such as <*final_decision*>. The agent maintains comprehensive execution traces including intermediate steps, tool call records with success status, and iteration counts. This agent architecture supports sophisticated workflows, enabling adaptive behavior in SOPs requiring fallback logic, verification steps, or multi-phase reasoning.

B Examples of Real SOPs

C SOP-Bench: Patient Intake and Registration

This appendix provides an example of a Standard Operating Procedure (SOP) package for patient registration in a healthcare setting. It includes the main SOP document, JSON tool specifications, and sample patient data, illustrating how operational procedures are formalized and implemented in healthcare information systems.

C.1 SOP: Patient Intake and Registration

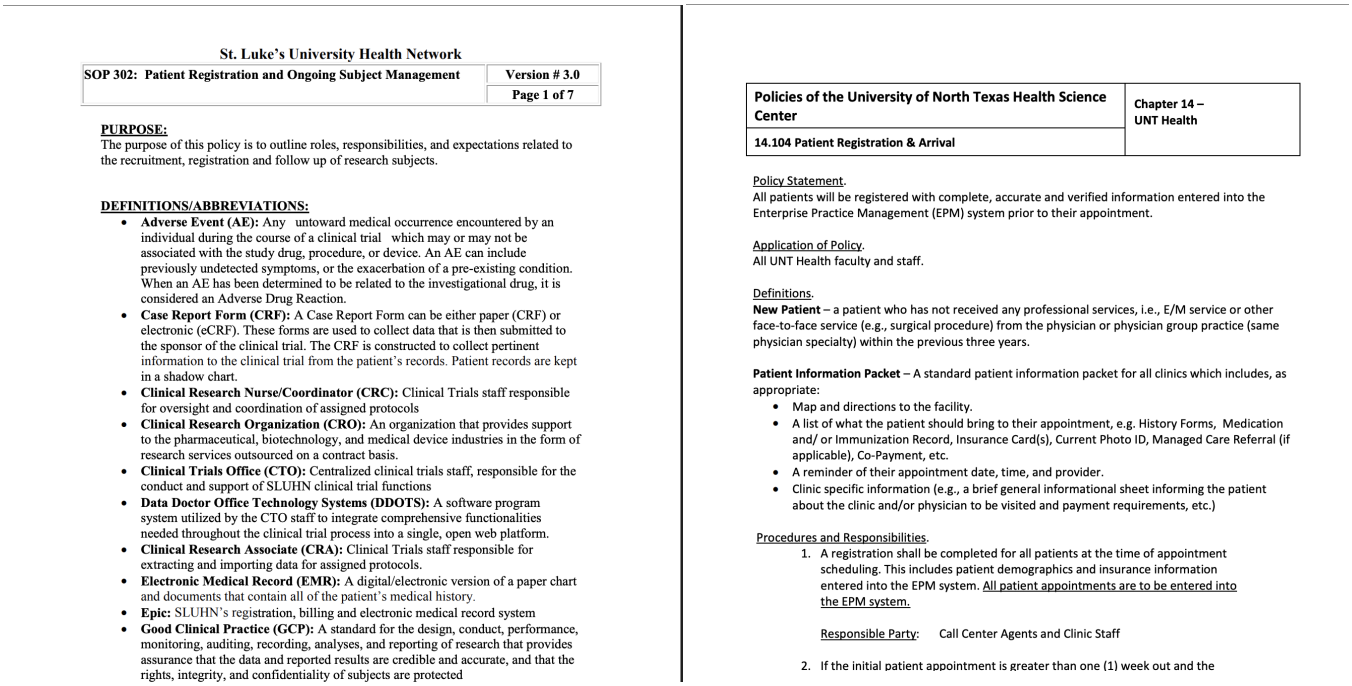
Patient Intake and Registration SOP - Part 1

```
1. Purpose
This Standard Operating Procedure establishes a framework for intake
and registration of new patients, encompassing demographic data
collection, insurance verification, risk stratification, and pharmacy
network validation in compliance with HIPAA regulations.

2. Scope
This procedure applies to all healthcare personnel involved in patient
registration, including Patient Access Representatives, Insurance
Verification Specialists, and Clinical Intake Coordinators.

3. Definitions
3.1 Protected Health Information (PHI): Individually identifiable
    health information as defined by HIPAA
3.2 Benefits Coordination Protocol (BCP): Systematic verification of
    primary and secondary insurance benefits
3.3 Clinical Risk Index (CRI): Composite score derived from medical
    history and lifestyle factors
3.4 Pharmacy Benefit Management (PBM) Validation: Real-time
    verification of prescription coverage
3.5 Network Adequacy Verification (NAV): Assessment of provider
    network status
3.6 Risk Stratification Algorithm (RSA): Mathematical model for
    patient risk assessment

4. Input
4.1 Required Documentation (some are optional)
- Government-issued photo identification (optional)
- Current insurance card(s)
- Complete medical history questionnaire
- Pharmacy benefit card
- Prior authorization documentation (optional)
- Release of medical records forms (optional)

4.2 System Requirements
- Electronic Health Record (EHR) access
- Insurance eligibility verification portal
- PBM interface connectivity
- Risk assessment calculation tool

5. Main Procedure

5.1 Insurance Validation Protocol
5.1.1 Primary Insurance Verification
- Access payer portal using provider credentials
- Verify coverage status and effective dates
- Document benefit levels and copayment requirements
- Confirm network participation status
- Record authorization requirements

5.1.2 Prescription Benefit Validation
- Query PBM database for current coverage
- Verify formulary compliance parameters
- Document prior authorization requirements
- Confirm specialty pharmacy protocols
- Record copayment structure

5.2 Risk Assessment Protocol
5.2.1 Lifestyle Risk Evaluation
```

Patient Intake and Registration SOP - Part 2

```
- Calculate smoking risk index (Never=0, Former=1, Current=2)
- Assess alcohol consumption (None=0, Occasional=1, Moderate=2, Heavy=3)
- Evaluate exercise patterns (5+ times=-1, 3-4 times=0, 1-2 times=1, None=2)
- Compute aggregate lifestyle score

5.2.2 Clinical Risk Assessment
- Review surgical history chronology
- Evaluate chronic condition severity
- Calculate comorbidity interaction score
- Generate weighted risk factor index

5.3 Patient Registration Protocol
5.3.1 Eligibility Confirmation
- Verify insurance_validation status = "valid"
- Confirm prescription_insurance_validation = "valid"
- Validate life_style_risk_level within parameters
- Assess overall_risk_level compatibility

5.3.2 Pharmacy Network Verification
- Query preferred pharmacy database
- Verify pharmacy_check status = "yes"
- Confirm pharmacy network participation
- Document special handling requirements

6. Output
6.1 Required Documentation
The procedure generates a verification report (json) with all results.
These include insurance validation check, prescription validation check,
pharmacy check, life style risk, overall risk, and registration status.
All outputs must be archived in compliance with regulatory requirements.

6.2 System Updates
- Active patient status in EHR
- Documented risk levels
- Verified insurance records
- Confirmed pharmacy selection
- Completed registration status
```

C.2 Sample Patient Data (Synthetically Generated)



St. Luke's University Health Network

SOP 302: Patient Registration and Ongoing Subject Management	Version # 3.0
	Page 1 of 7

PURPOSE:
The purpose of this policy is to outline roles, responsibilities, and expectations related to the recruitment, registration and follow up of research subjects.

DEFINITIONS/ABBREVIATIONS:

- **Adverse Event (AE):** Any untoward medical occurrence encountered by an individual during the course of a clinical trial which may or may not be associated with the study drug, procedure, or device. An AE can include previously undetected symptoms, or the exacerbation of a pre-existing condition. When an AE has been determined to be related to the investigational drug, it is considered an Adverse Drug Reaction.
- **Case Report Form (CRF):** A Case Report Form can be either paper (CRF) or electronic (eCRF). These forms are used to collect data that is then submitted to the sponsor of the clinical trial. The CRF is constructed to collect pertinent information to the clinical trial from the patient's records. Patient records are kept in a shadow chart.
- **Clinical Research Nurse/Coordinator (CRC):** Clinical Trials staff responsible for oversight and coordination of assigned protocols
- **Clinical Research Organization (CRO):** An organization that provides support to the pharmaceutical, biotechnology, and medical device industries in the form of research services outsourced on a contract basis.
- **Clinical Trials Office (CTO):** Centralized clinical trials staff, responsible for the conduct and support of SLUHN clinical trial functions
- **Data Doctor Office Technology Systems (DDOTS):** A software program system utilized by the CTO staff to integrate comprehensive functionalities needed throughout the clinical trial process into a single, open web platform.
- **Clinical Research Associate (CRA):** Clinical Trials staff responsible for extracting and importing data for assigned protocols.
- **Electronic Medical Record (EMR):** A digital/electronic version of a paper chart and documents that contain all of the patient's medical history.
- **Epic:** SLUHN's registration, billing and electronic medical record system
- **Good Clinical Practice (GCP):** A standard for the design, conduct, performance, monitoring, auditing, recording, analyses, and reporting of research that provides assurance that the data and reported results are credible and accurate, and that the rights, integrity, and confidentiality of subjects are protected

Policies of the University of North Texas Health Science Center	Chapter 14 – UNT Health
14.104 Patient Registration & Arrival	

Policy Statement.
All patients will be registered with complete, accurate and verified information entered into the Enterprise Practice Management (EPM) system prior to their appointment.

Application of Policy.
All UNT Health faculty and staff.

Definitions.
New Patient – a patient who has not received any professional services, i.e., E/M service or other face-to-face service (e.g., surgical procedure) from the physician or physician group practice (same physician specialty) within the previous three years.

Patient Information Packet – A standard patient information packet for all clinics which includes, as appropriate:

- Map and directions to the facility.
- A list of what the patient should bring to their appointment, e.g. History Forms, Medication and/ or Immunization Record, Insurance Card(s), Current Photo ID, Managed Care Referral (if applicable), Co-Payment, etc.
- A reminder of their appointment date, time, and provider.
- Clinic specific information (e.g., a brief general informational sheet informing the patient about the clinic and/or physician to be visited and payment requirements, etc.)

Procedures and Responsibilities.

1. A registration shall be completed for all patients at the time of appointment scheduling. This includes patient demographics and insurance information entered into the EPM system. All patient appointments are to be entered into the EPM system.

   Responsible Party: Call Center Agents and Clinic Staff

2. If the initial patient appointment is greater than one (1) week out and the

Figure 4: Examples of patient registration SOPs

Table 7: Sample Patient Registration Data (First Five Records)

Patient ID	First Name	Last Name	DOB	Age	Insurance	Smoking	Alcohol	Exercise	Insurance Valid	Risk Level	Reg
P100001	John1	Dove	1980-05-15	43	Blue Cross	Never	Occasional	3-4 times	valid	low	success
P100002	John2	Dove	1995-08-22	28	Aetna	Current	Moderate	1-2 times	valid	medium	success
P100003	John3	Dove	1972-03-10	51	United	Former	Heavy	None	valid	high	failure
P100004	John4	Dove	1990-11-30	33	Cigna	Never	None	5+ times	valid	low	success
P100005	John5	Dove	1965-07-25	58	Medicare	Former	Occasional	1-2 times	invalid	high	failure