ReleaseScribe: An AI Agent for Automating Compliance Checklists in Product Release Processes

Barun Kumar Saha
Grid Automation R&D
Hitachi Energy
Bangalore 560048, India
barun.kumarsaha@hitachienergy.com

***Abstract*—Organizations typically have multiple processes and internal compliance checks to determine whether a product is ready for market release and that it meets all prerequisites. These assessments are often performed manually by reviewing multiple documents—sometimes hundreds of pages long—and recording the responses in structured checklists, a process that is both time-consuming and error-prone. To address this challenge, we present ReleaseScribe, an Artificial Intelligence (AI) agent powered by a Large Language Model (LLM) that automates the completion of compliance checklists for product release processes. For any given process, ReleaseScribe takes a set of input files and populates a pre-defined checklist with appropriate responses based on the provided data. The agent employs a set of tools to orchestrate response generation by the LLM, enforce a structured output schema, and write data into spreadsheets. To improve reliability, the initial responses undergo an additional LLM-based review step to minimize inconsistencies. Furthermore, for each process, ReleaseScribe leverages customized prompts and metadata to handle interdependent checklist fields accurately. We evaluated ReleaseScribe using real-world data from two distinct processes. The results of the performance evaluation show that ReleaseScribe achieves a response-level accuracy of approximately 95%–98%, demonstrating its effectiveness in reducing manual effort while maintaining high compliance accuracy.**

***Index Terms*—Artificial Intelligence, Large Language Models, Agents, Product Release, Compliance, Checklists, Spreadsheets**

I. Introduction

Recent advances in AI and LLMs are increasingly being applied in industrial contexts [1]. In particular, AI agents, such as Reasoning and Acting (ReAct) [2], go beyond text generation to interact with real-world systems by leveraging tools, enabling new workflows and making existing processes more efficient. As a result, AI agents are finding potential usage across various stages of the product and service lifecycle, including design [3], testing [4], and customer support [5].

Before a product can be released to the market, organizations typically perform multiple internal reviews and quality checks to ensure readiness. Such assessments often include, but are not limited to, quality checks aligned with the software development life cycle, verification and validation activities, as well as the certifications and approvals necessary for product release. Accordingly, there are several processes that typically involve structured checklists with appropriate questions and response options to verify compliance with internal guidelines.

Traditionally, such an assessment is performed manually by going through large volumes of documentation that often has hundreds of pages, gathering evidence for each criterion or question from the target checklist[1], and recording the responses back in the same checklist. This process is repeated for every product release, requiring a significant manual effort to evaluate compliance with internal processes. On the other hand, in the absence of an assessor, delegating such tasks to another person may involve a significant knowledge transfer in a short period. In addition, one needs to avoid potential human errors. Consequently, there is a strong need to automate the product release compliance checklists.

It may be noted that although some LLM- and agent-based approaches exist for working with spreadsheets [6]–[9], they are primarily designed for general-purpose tasks. In contrast, filling out checklists—which are generally formal—for process compliance status can be somewhat different. On the one hand, the task is simpler because, for a given organization, an agent needs to work with a limited set of spreadsheet templates. On the other hand, the task is more difficult because specific rows and columns of these pre-designed checklists need to be filled out accurately based on available data.

To address these challenges, in this work, we investigate the use of AI agents and design ReleaseScribe to assist with product release compliance checks involving multiple processes. We assume that each process has a checklist template where the responses need to be recorded.

ReleaseScribe provides a conversational interface where users can write their queries and upload files, as illustrated in Figure 1. Based on the process specified by the user, it uses the corresponding checklist template and fills it out with appropriate responses gathered from the input data. ReleaseScribe uses a ReAct agent with six tools (Python functions) to orchestrate the response generation and spreadsheet operations. Unlike some prior works, ReleaseScribe introduces a structured output schema-driven orchestration and a reflection mechanism to handle interdependent checklist fields, which are critical for compliance accuracy. Early feedback from users suggests that ReleaseScribe can save up to an hour or a day's effort, depending on the process.

The specific contributions of this work are as follows:

[1]In our context, a "checklist" is a spreadsheet containing one or more sheets, where each sheet contains covering multiple criteria for assessment.

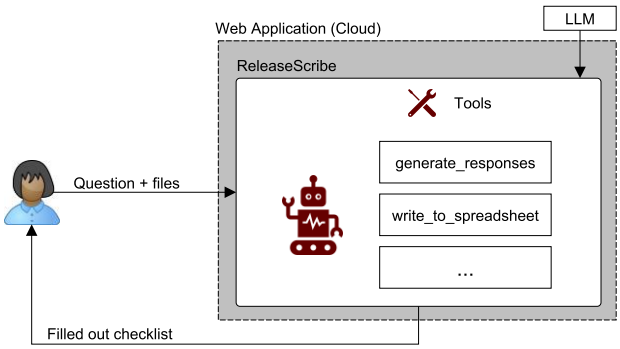

Fig. 1: A high-level illustration of using ReleaseScribe agent to fill out checklists based on input data.

- Designing ReleaseScribe, an AI agent that automates compliance checks for product release processes by leveraging relevant data, reports, and pre-defined checklists.
- Building a checklist helper module that parses input files, invokes an LLM for response generation based on data and instructions, and writes data into spreadsheets while adhering to layout constraints.
- Introducing schema-driven orchestration combined with process-specific metadata and prompts to ensure accurate structured outputs and manage interdependent checklist fields effectively.
- Evaluating the accuracy of ReleaseScribe by considering multiple processes and real-life data pertaining to them.

The remainder of this work is organized as follows: Section II briefly reviews the contemporary use of LLMs and AI agents. Section III details the design of ReleaseScribe, including the definition of "process" and the checklist filling workflow. The experimental setup is discussed in Section IV. The results of the performance evaluation are presented in Section V. Finally, Section VI concludes this work.

II. Background

Koshkin et al. [10] designed a multi-agent system for end-to-end market analysis, report generation, and evaluation by exploiting already available domain knowledge via few-shot prompting. Roy et al. [11] investigated the effectiveness of ReAct agents for root cause analysis in the context of cloud-related incident management. The study revealed that although a typical LLM-based approach, for example, using Chain-of-Thought [12], led to relatively more correct results, the use of an agentic approach significantly lowered hallucinations. In a different context, Saha et al. [13] also observed that an agentic approach, in general, can lead to more accurate power grid fault analysis results as compared to an LLM.

Kulkarni [5] considered the use of agents to address the Standard Operating Procedure (SOP) in the context of resolving customer issues. The author noted that, in the e-commerce industry, it typically involves interacting with users and checking/updating status, which can be automated using agents.

On the other hand, Nandi et al. [14] noted that LLM agents often struggle to adhere to SOPs, especially in domains involving knowledge-based operations. The authors developed SOP-Bench, a synthetic benchmark, and observed that ReAct and other agents fare significantly poorly on SOP-like tasks, although they score higher on other benchmarks, underscoring the complexity of adhering to SOPs.

Given the ubiquity of spreadsheets in business and other domains, several works have investigated the use of LLMs and AI agents to improve spreadsheet operations. Ma et al. [6], for example, designed a benchmark, along with relevant metrics, to enable the evaluation of AI-generated responses.

Dong et al. [7] fine-tuned several LLMs to improve the understanding of spreadsheets for a variety of downstream tasks. In particular, the authors proposed an encoding framework with sheet compression for token efficiency. Liang et al. [8] designed TableTalk, an agent to help users build spreadsheets incrementally and perform various operations, such as table creation and chart generation, based on a structured plan, thereby reducing the cognitive load of users.

Chen et al. [9], on the other hand, designed SheetAgent, an LLM-powered autonomous agent that can help with various tasks related to spreadsheets, such as formatting rows and columns, performing analysis and calculations, and adding charts. SheetAgent consists of the Planner, Informer, and Retriever modules to understand the sheets, use any available examples, and generate Python code to address the query, which results in about 36%–96% successful completions when evaluated against different tasks.

To synthesize, we observe that applications of LLM agents have been investigated pertaining to various stages of the product or service lifecycle. However, the use of agents in the context of assessing product release readiness largely remains unexplored. In particular, the application of agentic workflows for internal process compliance checks—requiring existing, pre-designed checklists to be filled out based on the contents of provided files—largely lacks focus.

Unlike generic spreadsheet tasks, compliance checklists typically require strict adherence to predefined layouts and dependencies across sheets, where even a single incorrect response can potentially invalidate the entire assessment. Contemporary solutions tend to emphasize largely autonomous operations, which is not strictly required for fixed spreadsheet layouts and where such autonomy can potentially affect an agent's understanding. Moreover, the difficulty to adhere to SOPs, in part, is also separately hinted at by Nandi et al. [14]. In order to address these challenges, in this paper, we investigate an AI agent approach to assess product release criteria and fill out necessary checklists.

III. System Design

In this section, we describe the design of ReleaseScribe and how it works for any process requiring a checklist to be filled out. ReleaseScribe is based on the ReAct agent architecture. It consists of a set of tools to help with checklist operations.

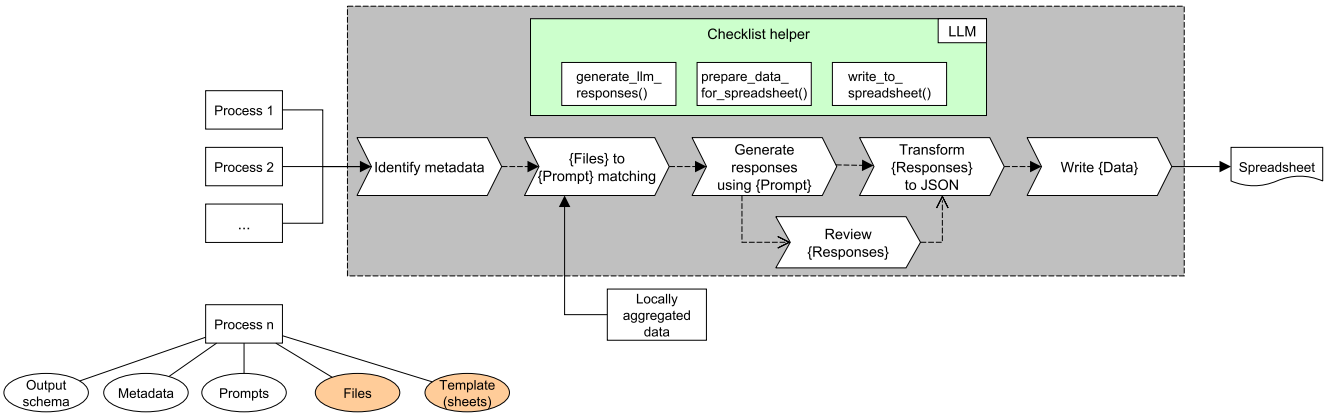

Fig. 2: Illustration of the checklist response generation pipeline. The solid arrows represent deterministic data flows, whereas the dashed arrows indicate flows (tool usage by the agent) enforced by the LLM prompt. The open arrow indicates an optional step in the flow. The checklist helper module implements the functionality used by the tools, exposed to the agent via interfaces.

A. Process Definition

Figure 2 illustrates the different components of ReleaseScribe to help fill out checklists for different processes. Here, a "process" represents any internal (or external) process of an organization for which some form of compliance needs to be assessed. For example, organizations can have internal assessments to identify whether a project is ready to begin or complete. On the other hand, assessing the quality of a product against a set of internal guidelines is perhaps ubiquitous across organizations before any release. Moreover, products may also need to be assessed for certifications and tests. Although these processes may largely vary from one organization to another, a common requirement is to fill out checklists pertaining to the processes based on relevant input data.

Formally, a process $\phi = \langle F, T, S, M, P \rangle$ is represented by a set of input and output entities. Here, F and T, respectively, indicate the set of input files and the checklist template for the concerned process. In other words, responses are meant to be generated for the given spreadsheet template based on the provided input files. In general, each process in an organization is headed by an individual. Accordingly, the process owner identifies and provides F and T in order to integrate said process with ReleaseScribe. It may be noted that F and T are mandatory inputs. Moreover, these two inputs are sufficient to manually evaluate the process compliance.

In order to automate the process, ReleaseScribe requires three additional items. First, an output schema (S) is defined to represent each sheet of the spreadsheet in a structured format. In other words, the LLM is asked to generate responses following the format S.

Next, for each process, a metadata (M) file is manually created as a one-time activity. The metadata contains the name and description of the process as well as the template file. In addition, the file contains other information about each sheet of the spreadsheet. These include, but are not limited to, the name of the sheet, if it is dependent on another sheet, human-readable field names mapped to column or row number (for example, "requirement": "C"), the columns with dropdown lists and associated values, path to the prompt file, the prompt keys, and optional additional notes.

The final component of ϕ is a set of prompts, P, one for each sheet of the concerned spreadsheet. It may be noted that the process checklist templates contain a set of questions that need to be answered or verified, for example, "Is the product version available?" However, we found that often these "standard" instructions can be insufficient for the LLMs. On the other hand, making such instructions more detailed would need process-related approvals as well as make the checklists too verbose and long, for example, by adding a few-shot examples[2]. Consequently, we use a separate prompt for each sheet from the checklist, where we copy the criteria/questionnaire text. In addition, for some questions that seem to be difficult to get correct answers from the LLM, we provided additional instructions, often with examples. To summarize, if the spreadsheet (checklist) for a process contains m different sheets that need to be filled out by ReleaseScribe, then $|S| = |P| = m$.

Unlike F and T, the latter components are not provided by the process owner. Instead, they are designed through consultations with process owners to capture the underlying business logic. Internal processes and checklists in any organization, once defined, typically change infrequently. Therefore, the process quintuple remains stable once all components are defined, while still allowing scope for revision when necessary. We argue that human insight into the design of these components is not only useful but essential when considering process compliance.

[2]For instance, a certain checklist criterion requires comparing bug IDs from multiple tables based on their status. We found that the LLM's response to the original criterion text was often incorrect. Therefore, we provided very detailed instructions for this criterion in the corresponding prompt. As another example, the LLM often struggled to parse information from unstructured document footer text. In this case, we added examples to illustrate the correct parsing technique.

B. Workflow Orchestration

ReleaseScribe contains the checklist helper module, which consists of a set of functions enabling checklist operations. For example, the module can (1) create a copy of a spreadsheet based on the concerned template, (2) generate responses for a given sheet by invoking an LLM, and (3) write given data to a given sheet. The checklist helper module can internally perform several other operations, such as mapping the correct input file names to the LLM prompts and reviewing the responses.

Some of the tools (Python functions) of the ReAct agent invoke certain functions from the checklist helper modules. For example, create_checklist_helper, generate_llm_responses_for_checklist, prepare_data_for_spreadsheet, and write_to_spreadsheet—these tools instantiate a ChecklistHelper class and call the appropriate methods. Another, extract_file_contents, helps with extracting the plain-text contents from any input file. Collectively, the ReAct agent with these five tools fills out the relevant checklists. The final tool, check_document_formatting, helps with checking formatting errors in the input documents.

Figure 2 illustrates a sequence of steps connected by dashed arrows with closed ends, each indicating a different stage of filling out a spreadsheet. For example, when a user asks "Generate the checklist for ⟨Process X⟩," ReleaseScribe begins by identifying the appropriate metadata file. Based on it, the prompt for each sheet of the checklist (spreadsheet) is identified. The prompts may often have keys (placeholders) for file names or their contents. Based on the input files provided by the user along with the query, the files are matched to the prompt keys by invoking the LLM.

In particular, the prompt keys are named semantically, for example, "test_report." To match files with the keys, the file names and a brief snippet from each of them are sent to the LLM. The LLM, based on the relevance and similarity, returns a mapping of keys to files, which is used in the subsequent steps.

It may be noted that some processes may take a variable number of files containing unstructured data as inputs, for example, emails, presentations, web page exports, and so on. Since the number of files in this case is unknown (and large), they cannot be mapped to prompt keys indicating file paths. Rather, we provide the aggregated contents of all the concerned files. To achieve this, the users run a Python script on their systems to generate the aggregated data files, which are then uploaded to ReleaseScribe.

On the other hand, certain files, for example, product manuals, can have up to thousands of pages, thereby occupying a significant portion of an LLM's context window. To mitigate this potential issue, the script also provides an option to specify the maximum text length to consider for each file during aggregation.

With the template and input files available, the agent invokes the LLM, instructing it to respond following the structured schema S. However, these responses (data) are not suitable for a spreadsheet library. Therefore, the next step of ReleaseScribe transforms the LLM responses into an intermediate JSON representation in the form of row/column addresses and values. To do this, the agent leverages layout information of the concerned sheet(s) presented in Markdown format. Finally, the available data are written into the identified cells, and ReleaseScribe responds to the user with the filled-out spreadsheet.

It may be noted that, unlike some other agents, ReAct plans only one step at a time, which can lead to potential uncertainty in the steps executed, as well as the sequence of their execution, leading to a bad user experience. In order to mitigate it, in the system prompt of the ReAct agent, we specify that this particular sequence of steps needs to be executed when any user asks to generate a checklist. Thereby, we achieve the sequential tool calling in a workflow-like manner[3], indicated by dashed arrows in Figure 2.

Finally, it may be noted that the Figure also contains an optional step to review the LLM-generated responses before translating them into JSON format for writing into the spreadsheets. Occasionally, LLMs produce conflicting values, for example, marking a status as compliant while evidence suggests otherwise. To mitigate this challenge, we introduce an additional step, where the LLM is asked to review the responses and fix any inconsistencies. This "reflection" step can potentially help to improve the accuracy, which will be discussed in a later section.

IV. Performance Evaluation

We built ReleaseScribe using Python 3.12 and relevant libraries, such as Chainlit[4], LlamaIndex[5], MarkItDown[6], and OpenPyXL[7]. We used GPT 4.1, with temperature set to 0.

Our prompts instructed the LLM to invoke five tools in sequence when any user asks to generate a checklist. In our trials, we found that, apart from a few instances, these instructions were followed by the LLM, enabling the ReAct agent to exhibit a generally deterministic behavior for this particular use case.

We evaluated the performance of ReleaseScribe using real-life data for product release criteria review (henceforth, process A) and product testing preparations (henceforth, process B).

[3]A pertinent question is why not use a linear, deterministic workflow with a fixed number of steps instead of an agent. While the primary objective of ReleaseScribe is to fill out process checklists with appropriate responses, it also supports answering related queries. For example, users may request a list of anomalies (deviations from guidelines) in a report, or a new user may ask about the criteria for a successful recommendation as defined in the checklists. Furthermore, future requirements may include linking external knowledge bases to handle additional query types. A simple workflow is unsuitable for such advanced use cases. Therefore, we adopt an agent-based approach. Moreover, the agentic design enables robust, context-aware orchestration and error handling across all steps, ensuring that even ambiguous or incomplete inputs are managed without brittle, hard-coded logic. This unified reasoning across tool invocations is critical for maintaining reliability and accuracy in real-world, variable scenarios, where static workflows are prone to silent failure or require constant manual intervention.

[4]https://docs.chainlit.io/get-started/overview

[5]https://github.com/run-llama/llama_index

[6]https://github.com/microsoft/markitdown

[7]https://openpyxl.readthedocs.io/

Our experiments were designed with the aim of measuring, in general, how well pre-defined checklists (spreadsheets) can be filled out based on relevant process data using AI.

The checklist pertaining to process A consists of two sheets, A1 and A2. Sheet A1 primarily verifies the compliance status (along with evidence) for several criteria, for example, the product version number is in the correct format. On the other hand, A2 presents a summary of the checks. Some of the responses in A2 depend on A1, thereby creating a dependency. For example, if there is any non-compliance recorded in A1, the decision in A2 would be negative.

Process A takes two reports as input files, out of which one is optional. Sheet A1 has up to 11 $\times$ 2 (one for status and another for comment) = 22 or 20 $\times$ 2 = 40 cells to be filled with data, based on whether a single report or both reports are provided. Accordingly, A2 has 11 or 15 cells to be filled out, respectively.

For process B, we considered a single sheet, B1, which checked the compliance status of preconditions for product testing. Since there are different kinds of testing, the sheet layout of B1 was relatively more challenging. In particular, B1 consists of five different groups of rows and columns arranged according to the test type, and only one such group of cells must be filled in based on the available data. These groups have 5–7 cells.

Unlike process A, the compliance checks for process B are required to be performed by considering the contents of a large bundle of files. For example, some of the checks include the verification of the presence of required files and the Point of Contact in the concerned data files. The local data aggregation script is executed to create an aggregate data file, which is then uploaded to ReleaseScribe.

In our evaluations, we considered the reports from five different product releases (labeled R1 through R5) for process A. Therefore, there were 5 $\times$ 2 = 10 sheets to evaluate in this case. We also considered four different products planned for testing under process B, so there were 4 $\times$ 1 = 4 sheets with AI-generated responses. We evaluated the accuracy of ReleaseScribe based on these input data.

We considered two accuracy metrics pertaining to different granularity levels. In *response-level accuracy* (alternatively, response accuracy), we considered how accurate the LLM responses are for the checklists, on average. In particular, we considered the aforementioned input files for each process and uploaded them onto ReleaseScribe. A separate conversation session was created for each checklist.

We manually investigated the checklists generated by ReleaseScribe. In particular, we verified whether all the responses are in the correct positions in the checklists and whether all the responses are correct. For each iteration, we counted the number of wrong or misplaced responses and converted them into proportions based on the total number of responses expected. Thereafter, we took the complement to identify the proportion of responses that are not wrong, which is the response accuracy metric. This metric indicates, on average, how capable an LLM is of generating correct responses for checklist criteria.

Separately, we also considered a subset of the aforementioned files to evaluate the *process-level accuracy* (alternatively, process accuracy) metric for process A, sheets A1 and A2. We generated the checklists based on these input files. Subsequently, we examined the responses of a subset of 16 cells (or 21, in case both reports were available) in each sheet. Only when all such cells of both sheets had correct responses was the iteration considered to be successful. Accordingly, we computed the average process accuracy metric by considering the average across all iterations.

The process accuracy metric is stricter than response accuracy. This is generally useful for compliance checks, where even a single wrong response can alter the overall compliance status. In general, both response and process accuracy should be close to 100%. However, when a strict compliance check is not required, a high value of response accuracy can be a suitable indicator to measure the usefulness of ReleaseScribe.

In order to evaluate the impact of our design choices, we conducted additional experiments. In one scenario, we measured the accuracy with and without running the LLM response review step (Figure 2). In another scenario, rather than instructing the LLM to use sheet-specific custom schemas for structured output (S in Section III-A), we asked it to use a generic schema, as shown Listing 1. In this case, we measured the process-level accuracy. It may be noted that said generic response schema was used only in this experiment; all other experiments used the sheet-specific schema.

Listing 1: A generic schema for the spreadsheets

```
class RowResponseItem(BaseModel):
"""
Abstract each row of a spreadsheet with optional ID and
     error message.
"""
      id: Optional[str] = Field(description='ID')
      is_error: bool = Field(
      description='Set to `True` if there is an error in the
           response; default `False`'
      )
      row_response: Optional[str] = Field(
      description='Response for a single row; can contain
          responses for multiple columns'
      )

class SpreadsheetResponseModel(BaseModel):
"""
Abstract a spreadsheet with multiple rows.
"""
      items: list[RowResponseItem] = Field(description='List
           of row response items')
      is_error: Optional[bool] = Field(
      description='Set to `True` if there is an error in the
           response; default `False`'
      )
```

Since the unique input data files were limited, we ran multiple trials of the experiments and generated checklists filled out with data. Based on these results, the average values and 95% confidence intervals were computed.

V. Results

Figure 3 shows the average response-level accuracy for sheets A1 and A2 from process A and sheet B1 from process

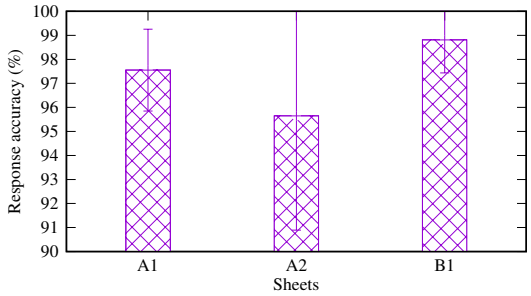

Fig. 3: Response-level accuracy for different sheets.

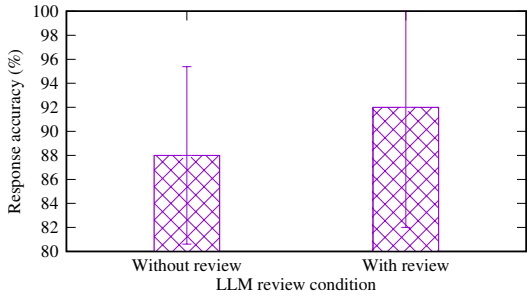

Fig. 4: Process-level accuracy for process A.

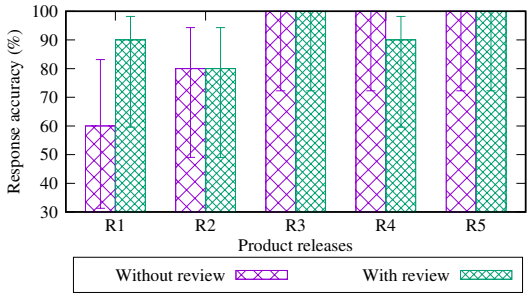

Fig. 5: Process-level accuracy for different product releases from process A.

B. It can be observed that at least 95% accuracy was obtained (in the case of A2), which implies that, on average, at least 95% of the AI-generated checklist responses were correct and placed in the appropriate target cells. On the other hand, in the case of B, the accuracy was around 98%. On investigating the errors, we observed that some of the responses generated for A1 were incorrect because they failed to follow the respective criteria or generated incorrect responses. However, in the case of A2, we found that occasionally the responses are correct but inserted into the wrong rows or columns in the spreadsheet.

Figure 4 illustrates the process-level accuracy for process A. When LLM-generated responses were used without any further review, the average success rate was 88%. However, when responses were further reviewed before being written into the spreadsheets, accuracy improved to 92%. This is largely because the additional review step potentially helped to "sanitize" some of the inconsistent responses, leading to more accurate results.

Figure 5 takes a detailed look at the process-level accuracy by plotting the average values for product release instances R1 through R5 from process A. In the case of R1, a review of the LLM responses before writing to the spreadsheets improved the accuracy from 60% to 90%. On the other hand, in the case of R4, the accuracy reduced from 100% to 90%. In other words, the LLM review step turned some of the correct responses wrong, possibly because of misinterpretation by the LLM. It may be recalled that process-level accuracy measures whether or not all answers are correct, which means even a single mistake can lead to a substantial change in the accuracy.

In the case of R2, R3, and R5, we observed no change when the responses generated by the LLM were reviewed before writing to the spreadsheets. Therefore, considering R1 and R4, a review of the LLM-generated responses generally seemed to have a positive impact. It may be noted that even though Figures 4 and 5 largely indicate marginal improvements, such revision of LLM-generated responses is generally useful and typically practiced.

In addition, we also considered the potential impact of the output schema S (Section III-A). When we set S to a single, generic output schema for all the sheets (Listing 1), we observed that although the LLM generated correct responses like any other case described earlier, the mapping of those responses to the specific rows and columns in the spreadsheets failed. In particular, with a generic schema, the process-level accuracy was 0% for all the cases. This is largely due to the reason that the semantic names of the fields in the schema and the criteria text in the spreadsheets together help toward correctly "aligning" the rows and columns, thereby helping in identifying the correct addresses. However, in the absence of such a sheet-specific schema, and since many criteria may have similar wording, identifying the correct rows and columns becomes difficult. The result indicates that checklist automation would benefit from using a sheet-specific schema to capture the responses from LLMs.

We close this section by discussing a few inconsistencies that we observed in our trials, which also affected the performance evaluation results.

- For some checklist criteria, we noticed that the LLM went beyond the instructions provided to it for the particular check and reasoned incorrectly based on other related text available in the input data files. We have largely addressed the issue by making the prompt instructions for this particular check rather detailed and stricter.
- An interesting result has been observed in case one of the reports that used an outdated template. Although this particular report is part of our performance evaluation, we regard this issue to be of less significance as the document templates have been updated.

VI. Conclusion

LLMs and AI agents are finding their applications in different stages of product and service development. However, the internal compliance checks for the product release process

largely remain manual. In this work, we designed ReleaseScribe to address this challenge. The agent takes a set of input files for any given process and fills out the corresponding checklist (spreadsheet) with appropriate responses. As an intermediate step, the LLM-generated responses are reviewed to minimize potential inconsistent answers before writing to the spreadsheet. Our experimental results indicate that ReleaseScribe generates mostly correct responses on average, saving hours of manual effort.

It may be noted that ReleaseScribe is intended to assist in the decision-making process by collating diverse data points from multiple sources into a single place. However, it is recommended that human supervision continue to be employed to cross-check the compliance status and make the final product release decision. This is particularly important since the approach relies on the correctness of LLM-generated responses, which, despite the reflection step, may occasionally introduce inconsistencies.

In the future, this work can be extended in different ways. The instructions to the LLM can be potentially improved to further reduce the incorrect responses. On the other hand, additional processes (checklists) and larger datasets may be considered to provide a more comprehensive evaluation of performance. Moreover, a final reflection layer can be added to verify the overall consistency and presentation of the checklists.

REFERENCES

[1] X. Tian, J. Xu, S. Wang, and Y. Zhou, "Construction and Application of a Multi-Modal Knowledge Graph Integrated with Large Language Models in the Field of Manufacturing Processes," in *2025 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, 2025, pp. 0288–0292.

[2] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: Synergizing Reasoning and Acting in Language Models," in *International Conference on Learning Representations (ICLR)*, 2023.

[3] A. Allam, Y. Mansour, and M. Shalan, "ASIC-Agent: An Autonomous Multi-Agent System for ASIC Design with Benchmark Evaluation," in *IEEE ICLAD*, 2025, pp. 23–29.

[4] I. Bouzenia and M. Pradel, "You Name It, I Run It: An LLM Agent to Execute Tests of Arbitrary Projects," *Proc. ACM Softw. Eng.*, vol. 2, no. ISSTA, Jun. 2025. [Online]. Available: https://doi.org/10.1145/3728922

[5] M. Kulkarni, "Agent-S: LLM Agentic workflow to automate Standard Operating Procedures," 2025. [Online]. Available: https://arxiv.org/abs/2503.15520

[6] Z. Ma, B. Zhang, J. Zhang, J. Yu, X. Zhang, X. Zhang, S. Luo, X. Wang, and J. Tang, "SpreadsheetBench: Towards Challenging Real World Spreadsheet Manipulation," in *Advances in Neural Information Processing Systems*, vol. 37, 2024, pp. 94 871–94 908.

[7] H. Dong, J. Zhao, Y. Tian, J. Xiong, S. Xia, M. Zhou, Y. Lin, J. Cambronero, Y. He, S. Han, and D. Zhang, "SpreadsheetLLM: Encoding Spreadsheets for Large Language Models," 2025. [Online]. Available: https://arxiv.org/abs/2407.09025

[8] J. T. Liang, A. Kumar, Y. Bajpai, S. Gulwani, V. Le, C. Parnin, A. Radhakrishna, A. Tiwari, E. Murphy-Hill, and G. Soares, "TableTalk: Scaffolding Spreadsheet Development with a Language Agent," *ACM Trans. Comput.-Hum. Interact.*, Sep. 2025.

[9] Y. Chen, Y. Yuan, Z. Zhang, Y. Zheng, J. Liu, F. Ni, J. Hao, H. Mao, and F. Zhang, "SheetAgent: Towards a Generalist Agent for Spreadsheet Reasoning and Manipulation via Large Language Models," in *Proceedings of the ACM on Web Conference 2025*. Association for Computing Machinery, 2025, pp. 158–177.

[10] R. Koshkin, P. Dai, N. Fujikawa, M. Togami, and M. Visentini-Scarzanella, "MaRGen: Multi-Agent LLM Approach for Self-Directed Market Research and Analysis," in *LLM4ECommerce Workshop at KDD '25*. ACM, Aug. 2025.

[11] D. Roy, X. Zhang, R. Bhave, C. Bansal, P. Las-Casas, R. Fonseca, and S. Rajmohan, "Exploring LLM-Based Agents for Root Cause Analysis," in *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*, 2024, pp. 208–219. [Online]. Available: https://doi.org/10.1145/3663529.3663841

[12] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," in *Proceedings of the 36th International Conference on Neural Information Processing Systems*, 2022, pp. 24 824–24 837.

[13] B. K. Saha, A. V, and O. D. Naidu, "DrAgent: An Agentic Approach to Fault Analysis in Power Grids Using Large Language Models," in *2025 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, 2025, pp. 0938–0945.

[14] S. Nandi, A. Datta, N. Vichare, I. Bhattacharya, H. Raja, J. Xu, S. Ray, G. Carenini, A. Srivastava, A. Chan, M. H. Woo, A. Kandola, B. Theresa, and F. Carbone, "SOP-Bench: Complex Industrial SOPs for Evaluating LLM Agents," 2025. [Online]. Available: https://arxiv.org/abs/2506.08119