A Conversational Digital Assistant for Intelligent Process Automation

Yara Rizk, Vatche Isahagian(✉), Scott Boag, Yasaman Khazaeni, Merve Unuvar, Vinod Muthusamy, and Rania Khalaf

IBM Research AI, Cambridge, MA, USA
vatchei@ibm.com

Abstract. Robotic process automation (RPA) has emerged as the leading approach to automate tasks in business processes. Moving away from back-end automation, RPA automated the mouse-click on user interfaces; this outside-in approach reduced the overhead of updating legacy software. However, its many shortcomings, namely its lack of accessibility to business users, have prevented its widespread adoption in highly regulated industries. In this work, we explore interactive automation in the form of a conversational digital assistant. It allows business users to interact with and customize their automation solutions through natural language. The framework, which creates such assistants, relies on a multi-agent orchestration model and conversational wrappers for autonomous agents including RPAs. We demonstrate the effectiveness of our proposed approach on a loan approval business process and a travel preapproval business process.

Keywords: Business process automation · Interactive automation · Robotic process automation · Conversational assistant · Orchestration

1 Introduction

Business processes are the backbone of business enterprises and organizations [31]. A business process is a collection of tasks or activities that must be executed in a certain sequence to achieve a goal. In the era of digital transformation, robotic process automation (RPA) presents a low cost approach to inject automation in business processes. An RPA is developed for tasks that are frequent, repetitive, and error-prone. RPAs learn to execute such tasks in the user interface from humans through demonstration, behavior logs or business process descriptions. Unlike back-end automation approaches, this approach reduces the overhead of adopting automation by operating on top of legacy software.

However, highly-regulated industries still require human-in-the-loop automation due to compliance regulations, increased risk, and liability. Unfortunately, the end users are not tech-savvy, making it difficult for them to interact with RPAs and other automation solutions in their current format [12]. This lack of accessibility hinders a business user's ability to monitor and customize these

A. Asatiani et al. (Eds.): BPM Blockchain and RPA Forum 2020, LNBIP 393, pp. 85–100, 2020.
https://doi.org/10.1007/978-3-030-58779-6_6

solutions [8]. A human-consumable interactive automation, through natural language, would reduce the barrier of entry to business process automation.

We propose a framework to build conversational digital assistants, merging the two multi-billion dollar markets of RPA [3] and enterprise chatbots[1]. An assistant consists of multiple conversational software bots that automate specific tasks within a business process. It can be used by business users and domain experts who do not possess any programming or software development skills to customize and interact with their business process automation solutions through natural language. The assistant can be adopted in different enterprises depending on the domain of the agents, from banking and finance to retail, customer care and others. This interaction would foster trust because it makes the system more transparent to the users, allowing them to gain valuable insights into the system's operations. This trust increases the probability of the business users adopting more automation solutions [21]. With the increased interest in trusted business processes in this digital transformation era [25], the conversational assistant would serve as a crucial enabler to this paradigm.

Beyond trusted processes, researchers have investigated business process individualization [33]. Pursued by many businesses to differentiate themselves from a large pool of competitors, process individualization has historically been a challenge because it reduces efficiency and increases costs. The digital revolution overcame these challenges by enabling automated modification of individualized processes. Our framework can contribute to this space by further reducing the overhead of creating custom processes and monitoring them through natural language interactions.

The main research question we address in this work is: *what are the necessary characteristics of a software framework that enables interactive automation in business processes through natural language?* To that end, we present a unified conversational multi-agent orchestration assistant, which consists of multiple building blocks. Skills, including RPAs, automate tasks within a business process They can be composed into more powerful automation bots, and wrapped as conversational agents to interact with business users. As the user converses with the assistant, the orchestrator determines which agents should respond to the user. Therefore, the assistant provides business users access to their RPAs through natural language. We present a taxonomy of skills and agents, define an agent contract to enable such an orchestration and present an orchestration workflow that integrates diverse conversational agents.

Next, we briefly discuss related work in two main relevant fields: business process automation and conversational agents. Then, we present our proposed framework, before discussing our qualitative analysis on two use cases: travel preapproval, and loan application processing.

[1] https://www.businessinsider.com/chatbot-market-stats-trends.

2 Related Work

2.1 Business Process Automation

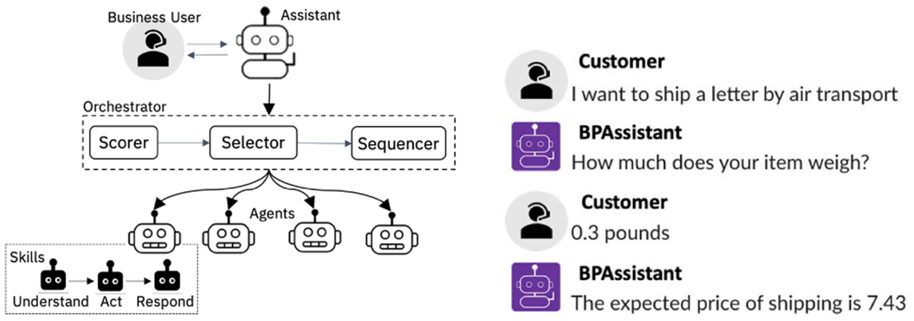
RPAs have recently been the main driver of the digital transformation with their light-weight approach to automating repetitive tasks [1]. They have enabled automation in multiple enterprises including accounting [19], auditing [6], human resources [22], banking [28], public administration [10] and energy sectors [15]. Multiple RPA vendors have offered state-of-the-art solutions to clients in various sectors. A survey of these products can be found in [2].

These RPAs have leveraged diverse technological advancements in the fields of artificial intelligence and software development. Gao et al. adopted deep learning optical character recognition (OCR) and classification models in document flow automation within a debt collector business process [8]. RPAs identified relationships between tasks from user behavior in [32] and used first-order logic to deduce automation rules. Crucial to RPA's success is automatically identifying RPA-eligible tasks; [16] relied on supervised machine learning and natural language processing of business process descriptions to identify these tasks.

Business process automation takes a step beyond RPAs to automate decision making in business processes. Marella et al. identified the field of automated planning as an enabler to more sophisticated business process automation [17]. Machine learning is another enabler; deep learning models, long short-term memory recurrent neural networks specifically, have also been trained to model business processes, a crucial component for more advanced automation [4]. Machine learning algorithms like support vector machines, shallow neural networks and random forests have been adopt in process mining applications [29]. An interactive process mining recommender system for business process discovery based on machine learning has also been investigated in the literature [26]. However, end-to-end process automation has not been widely adopted in enterprises due to business users' lack of trust in such technology [13] and limited accessibility and customization capabilities. Since business users lack the technical skills to monitor and customize automation solutions, a natural language interface may be key to the success of the digital transformation.

2.2 Conversational Agents

Enterprises have been interested in natural language processing advancements, given their heavy reliance on this communication modality. Considering the breadth of this field, we will only focus on conversational agents that are most relevant to the scope of our framework. Enterprise chatbots, a multi-billion dollar market dominated by giants like IBM, Google and Amazon, have been researched for decades and experienced significant improvements recently [7]. They have evolved from simple question answering customer support bots to fully autonomous assistants capable of performing tasks on behalf of humans [7]. One shopping bot is capable of custom-pricing products to increase sales [9]. Food delivery services adopted a delivery bot to reduce the effort customers need to order pizza [9]. Another bot crowdsourced answers generated by multiple chatbots, gradually reducing the reliance on the crowd through learned selection models [11]. The model estimated the likelihood of an agent returning the correct answer based on the crowd's votes and previous answer selections. Instead, we adopt a different orchestration model that doesn't solely rely on user feedback but has a more sophisticated scoring and selection model to pick one or more agents.



Customer
I want to ship a letter by air transport

BPAssistant
How much does your item weigh?

Customer
0.3 pounds

BPAssistant
The expected price of shipping is 7.43

Fig. 1. A conversational digital assistant framework

Researchers have also combined RPAs and chatbots to increase automation. A chatbot for agile software development teams was developed to provide insights into a team's performance by analyzing commits in version control software [18]. From a business process workflow, Kalia et al. derived a dialog tree-based chatbot to converse about the process [14]. Despite the systematic approach to chatbot design, it required significant effort and domain expertise.

3 Proposed Framework

The conversational digital assistant provides business users with two core functionalities beyond RPAs: conversational interaction and collaborative automation. Our proposed framework, illustrated in Fig. 1, achieves these functionalities by relying on an orchestrator capable of coordinating the execution of agents within the system, and agents that can converse with users in natural language while executing tasks in a business process [23]. Agents are composed of skills which can perform natural language understanding and generation in addition to task automation (e.g. RPAs). The orchestrator expects agents to adhere to a specific contract to determine which agents respond to a user's utterance. This modular approach simplifies the task of converting RPA to conversational agents, the system's maintenance and life cycle. It also allows us to add or remove certain domains and functionality from the scope of the assistant with minimal effort.

3.1 Skills

Skills are the assistant's building blocks and consist of atomic functions to *understand* a user's request, *act* to satisfy the user's request, and *respond* to the user.

Understand skills are generally natural language understanding functions that determine the user's intent and identify any entities in the user's utterance that would be needed by the *act* skill to properly execute. Such skills can be created using existing tools that create dialog bots such as IBM's Watson Assistant or Google's Dialogflow which can involve defining entities and intents and providing examples of them.

Act skills are generally RPAs that execute the user's intent to produce an outcome. They can be of two types. World-changing skills can change the state of the world, i.e. have side-effects on their environment; examples include skills that send emails or check credit scores. Non-world changing skills do not have side effects on their environment; examples include skills that read emails or check a bank account status. *Act* skills automate business process tasks and can be placed at decision points within a process to move it forward.

Respond skills produce a human-consumable response from the *act* skill's output. This can be a natural language utterance, a visual representation, or another modality. This skill can be as simple as adopting a template response or as sophisticated as a deep learning text generation model.

3.2 Agents

In our framework, an agent is simply a conversational RPA: the RPA is preceded by an *understand* skill and succeeded by a *respond* skill. Thus, we obtain an interactive RPA that communicates with business users in natural language. This modular approach enables the integration of RPAs that were not inherently interactive, while reducing the overhead of improving agents throughout their lifetime. However, not all RPAs are suitable to be conversational RPAs.

Composing an agent using the *understand-act-respond* pipeline achieves the interactive RPA aspect of the assistant and creates a standardized agent creation method. Enforcing a contract enables their integration within the same assistant. Thus, agents could cooperate on tasks and achieve more powerful functionality. Agents receive the input utterance and current context or state; they return a response, an updated context, a confidence in their response (a numerical value between 0 and 1 that quantifies their comprehension of the input and relevance of their response) and possibly other flags related to the conversation.

Within the conversational digital assistant paradigm, we identified five main types of agents. While many more can be created, we consider the ones that would be most relevant and commonly used in process automation: dialog, information retrieval, task execution, data analytics, and alert agents.

Dialog agents provide more human-like interactions. They are composed of an *understand* skill and a *respond* skill to answer user queries. They do not change the world and they can be implemented using any conversational agent technology; examples include chit-chat, FAQ or "help", etc.

Information retrieval agents query information sources to achieve their goal. They perform advanced reasoning to respond to user queries or information retrieval tasks in a process. They are composed of an *understand* skill, an information retrieval RPA and a *respond* skill. In general, most agents within this category do not contain world-changing skills (exception: credit score checks).

Task execution agents perform tasks within a business process that change the state of the world by moving the business process forward. Examples include submitting applications, filling in information in forms, and making decisions at decision points. Such agents may more intuitively fit within the RPA definition.

Data analytics agents cover a wide scope of functionality from data transformation and modeling to predictions and recommendations. Such agents go beyond information retrieval which provide factual and statistical data but do not go as far as task execution agents that act on the information; they still rely on humans to drive the business process. Examples include visualization and data export agents that manipulate and transform the data to more complex business process forecasting and performance prediction models.

Alerting agents allow users to conversationally customize alerts and notifications triggered by the occurrence of specific events within or related to the process, in essence enabling asynchronous monitoring of the process [24].

3.3 Orchestration

The orchestrator is the core component that allows interactive and cooperating agents in our framework. Its main functionalities include selecting a subset of agents that must respond to a user's request, managing the context and passing it among agents, and acting as the central dialog manager (controls a multi-turn conversation's state and flow).

Selecting the agent(s) to respond to a user's request can be achieved by different orchestrators. Stateless orchestrators, unlike their stateful counterparts, do not possess a central state tracker, i.e. context is maintained in every turn by passing context variables between the orchestrator and the agents. Maintaining context about the dialog also influences agent selection: if the user is in the middle a conversation with an agent, the orchesetrator must ensure a smooth interaction. If users digress from the conversation (i.e. move away before completing the conversation) or provide ambiguous statements, the dialog manager should properly handle such situations. Apriori orchestrators select agents based on the user input and some knowledge about agent capabilities. On the other hand, posterior orchestrators request a preview response from agents that factors into the orchestrator's selection. Such orchestrators could also request a confidence score to better assess an agent's response. Confidence is a quantification of an agent's understanding of the input and relevance of its response to it. Posterior orchestrators require agents to have a preview mode in order not to change the world if they are not selected for execution. They are agnostic to what agents exist in the assistant as long as they abide by the contract; they simply need to know of the agents' existence and how to reach them (API endpoint).

Each orchestration model has specific computational costs, architectural requirements, and contract constraints. These factors influence the assistant's orchestration model. In this work, we adopt a stateless, posterior orchestrator, called 3 S orchestrator, that consists of three main steps: scoring, selecting, and sequencing [30, 34]. Outlined in Algorithm 1, it assumes an agent (a_i) contract that returns a preview response (r_i), a confidence in the response (c_i), and a stickiness value (κ_i) that indicates whether the agent has been interacting with the user in previous turns (i.e. is in the middle of a conversation). A preview mode ensures that world-changing agents do not cause irreversible changes before the orchestrator makes its selection. Furthermore, preview and execute modes can be adopted to optimize the computation/latency of non-world changing agents. The nomenclature that is adopted throughout this work is defined in Table 1.

Table 1. Nomenclature

Symbol	Definition	Symbol	Definition
A	Set of agents	$n = \|A\|$	Number of agents
$a_i \in A$	Agent i	r_i	Response of agent a_i
$c_i \in [0, 1]$	Agent a_i's confidence	$\kappa_i \in \{0, 1\}$	Agent a_i's stickiness
u	User's utterance	$f_i(.)$	Function per agent a_i
s_i	Agent a_i's score	$g(.)$	Scorer function, computes a score per agent a_i
A_s	Set of selected agents	$h(.)$	Selector function
R	Final response	$k = \|A_s\|$	Number of selected agents

Algorithm 1. 3S Orchestration

```
1: procedure BROADCASTER($u$, $A$)
2:     for $a_i \in A = \{a_1, ..., a_n\}$ do
3:         $(c_i, \kappa_i, r_i) = f_i(u)$
4: procedure ORCHESTRATOR(Responses, Confidences)
5:     Scorer: $s_i = g(c_i, \kappa_i), \forall a_i \in A$
6:     Selector: $A_s = h(\{s_i \forall a_i \in A\})$
7:     Sequencer: $R = order(\{r_i, \forall a_i \in A_s\})$
8:     return $R$
```

Scorer. When the orchestrator receives a natural language utterance from the user, as shown in Fig. 2, it forwards the input to all agents. Once it receives the agents' preview responses, the scorer processes the agent confidences (and possibly other variables) to obtain normalized values. This is necessary in multi-agent environments where agents are developed by independent software developers or diverse frameworks that compute confidences differently. A scorer can be as simple as an identity function that does not modify the confidences (if inherently

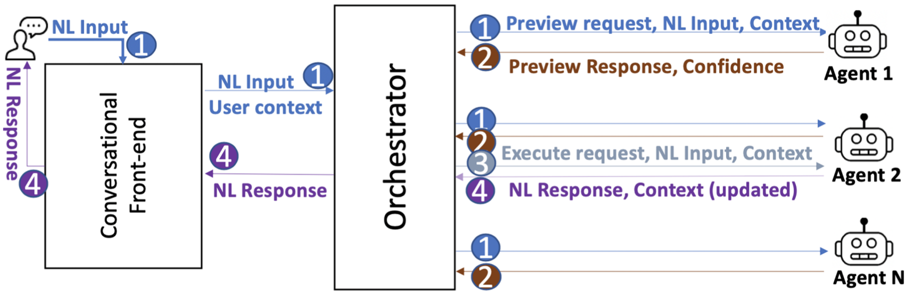

NL Input
1
NL Response
4
Conversational Front-end
NL Input
User context
1
4
NL Response
Orchestrator
1 Preview request, NL Input, Context
2 Preview Response, Confidence
Agent 1
1
2
3 Execute request, NL Input, Context
4 NL Response, Context (updated)
Agent 2
1
2
Agent N

Fig. 2. Graphical representation of a posterior orchestration pipeline

normalized) or as complex as a Bayesian approach that incorporates statistical and probabilistic models to scale the confidences. For example, per our agents' contract, a confidence between 0 and 1 is returned in addition to a flag, called stickiness equal to 0 or 1, that states whether an agent had been selected in the previous turn and is expecting an answer from the user. One scorer we adopted simply takes the maximum of both values, $s_i = max(c_i, \kappa_i)$.

Selector. Next, the selector processes the scores and determines which agent(s) must execute to respond to the user. The selector model can be a simple *Top 1 (or Top K)* selector that picks the agent (or K agents) with the highest score (above a minimum threshold, T, to identify cases outside the scope of the assistant), e.g. $A_s = max\{s_i : a_i \in A \quad \text{s.t.} \quad s_i > T\}$ (adopted in our experiments). It could also be a machine learning algorithm that utilizes other features such as previous conversation turns to select the next best agent(s). In a supervised learning domain, labels can represent agents that are the output of classifiers and the input to the classifier would be a feature vector consisting of the agents' scores and/or user utterances. In a reinforcement learning domain, actions can represent agents and the environment produces a reward when the correct agent is selected. With enough data, deep learning models can be trained.

Sequencer. If multiple agents were selected for execution, a sequencer determines the order to execute these agents and show their responses' to the user. This is crucial to the proper execution of collaborating agents since one agent's output is another's input; agents' execution is not independent. Various approaches ranging from relatively simple heuristic rules to more complex planning-based algorithms can be adopted.

4 Empirical Results

4.1 Implementation and Use Cases

The proposed framework is implemented in Python where most skills are accessible through API endpoints. A *top K* orchestrator is exposed as a Rest API, and called by a Slack bot[2]. Conversational skills were implemented in Watson Assistant[3]. *Act* skills were either external microservices or internal Python functions. The assistant, referred to as *BPAssistant*, consists of multiple agents (see Fig. 3) to handle two simplified business processes: loan application and travel preapproval [23]. Some agents are domain agnostic and can operate on multiple business processes without any overhead. Others are domain specific, developed for a very specific task in one of the business process. Finally, a set of agents were created from domain agnostic agents but must be instantiated for a specific domain. Hence, they require some configuration when deployed in a process.

The travel use case considers two persons that can interact with the *BPAssistant*: employees and managers. The travel preapproval process, shown in Fig. 4, was implemented in a business process management software. Employees submit a travel preapproval request to attend a conference (or event) by filling a form. Once submitted, the employee's manager processes the application. If approved by the manager, the application is forwarded to the director who makes the final decision on the travel request.

The simplified loan application process consists of a bank customer submitting a loan application and a bank officer processing this request to determine whether to approve or reject it. The assistant can help a loan officer automate certain parts of the process, in addition to query and analyze the process data.

These use cases are interesting because of their relevance to many enterprises, especially those that are about to start or are in the middle of their digital transformation journey. These processes (or parts of them) may need to suddenly become agile overnight in response to a pandemic (e.g. loan officer remotely approving a loan), or readjusted processes to address a sudden change in company policies (no flights to certain countries). Creating interactive agents from RPAs that automate tasks would enable employers' adoption of such systems. Natural language interactions reduce the necessary learning curve, leading to quick deployment.

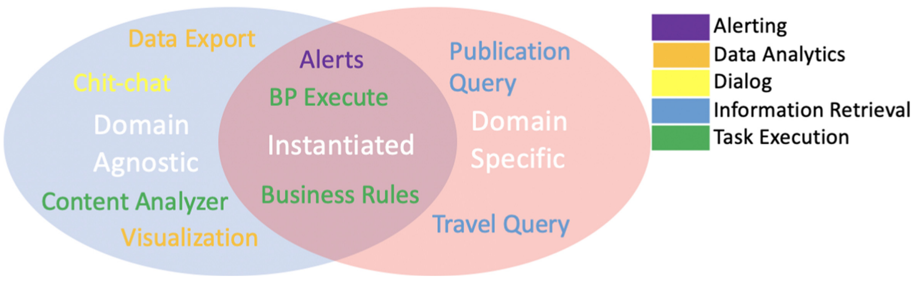


Fig. 3. Agents in *BPAssistant*

[2] https://api.slack.com/bot-users#bots-overview.
[3] https://www.ibm.com/cloud/watson-assistant.

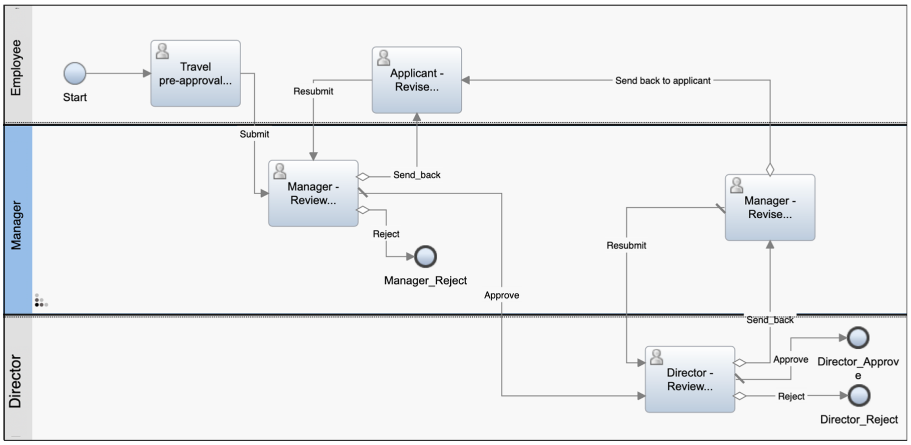
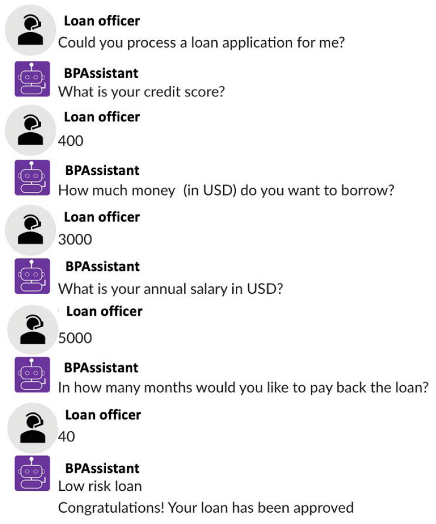


Fig. 4. Travel preapproval process



Loan officer
Could you process a loan application for me?

BPAssistant
What is your credit score?

Loan officer
400

BPAssistant
How much money (in USD) do you want to borrow?

Loan officer
3000

BPAssistant
What is your annual salary in USD?

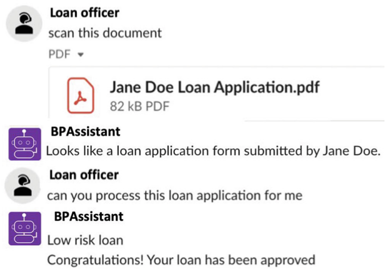
Loan officer
5000

BPAssistant
In how many months would you like to pay back the loan?

Loan officer
40

BPAssistant
Low risk loan
Congratulations! Your loan has been approved

Conversation driven.



Loan officer
scan this document
PDF ▾
Jane Doe Loan Application.pdf
82 kB PDF

BPAssistant
Looks like a loan application form submitted by Jane Doe.

Loan officer
can you process this loan application for me

BPAssistant
Low risk loan
Congratulations! Your loan has been approved

RPA cooperation.

Fig. 5. Conversation with *BPAssistant*: business rules agent.

4.2 Conversations with *BPAssistant*

BPAssistant allows users to accomplish different goals by relaying their intentions in natural language. First, consider a loan officer who wants to process a customer's loan application while consulting the bank's rule engine. The "Business Rules" agent would allow the loan officer to make decisions about tasks in a business process such as approving or rejecting the loan application. Figure 5a shows a sample multi-turn conversation with this agent through the assistant.

The agent was instantiated for the banking domain and implemented in IBM Decision Service[4], where users input the available information and the agent recommends whether to approve or reject the loan based on the business rules.

However, the interaction can be improved by automating information gathering using an RPA (configured as an agent) to extract this information from a document, for example. Figure 5b shows the same example but with cooperation among agents. The "content analyzer" agent, capable of extracting information from PDF documents, retrieves key-value pairs from the loan application file and saves them in the context of the orchestrator. Then, the "business rules" agent obtains this information through the orchestrator to determine whether the loan should be approved or rejected. Outside of the assistant framework, enabling such an RPA cooperation would require the creation of another RPA that transfers the data from the first RPA (content analyzer) to the second RPA (business engine). The orchestrator presents a more generalizable paradigm to achieve this functionality.

The loan officer can also use *BPAssistant* to analyze recent loan applications by querying the data associated with the loan process, as shown in Table 2. The natural language utterances in the table invoke the "Business Process Query" agent capable of converting natural language sentences to formal queries that are executed on a database to retrieve the data [27]. Furthermore, the loan officer can manipulate the data by plotting it and exporting it to a file, using the Visualization and Data Export agents, respectively. Both agents use the data from information retrieval agents, stored in the context of the orchestrator, as input to fulfill the user's request. Figure 6 displays these multi-modal responses, namely the visualization and CSV file generated based on the queried data.

Now, let's say the loan officer needs to attend training or a seminar on new loan processing procedures at the bank's headquarters. The travel costs can be expensed to the bank since it is a work trip[5]. The "Business Process (BP) Execute" agent can move the business process forward by asking the assistant to execute a step in the process on their behalf, such as submitting a travel request by saying "submit a travel request to the headquarters". *BPAssistant* would then submit a request on the employee's behalf and automatically populate the fields of the form by using appropriate *act* skills that can retrieve information.

Furthermore, a bank manager can use the same *BPAssistant* to set up alerts about employees' submitted travel requests using the "Alerts" agent. This agent enables users to create and customize alerts related to a process [24]. Once an notification is received, the manager can review the employee's travel preapproval request and determine whether to approve it or not, as shown in Table 2.

[4] https://www.ibm.com/products/operational-decision-manager.

[5] Travel preapproval processes are common in many enterprises including IBM Research. Employees submit preapprovals to attend academic conferences, client meetings, training events, etc.

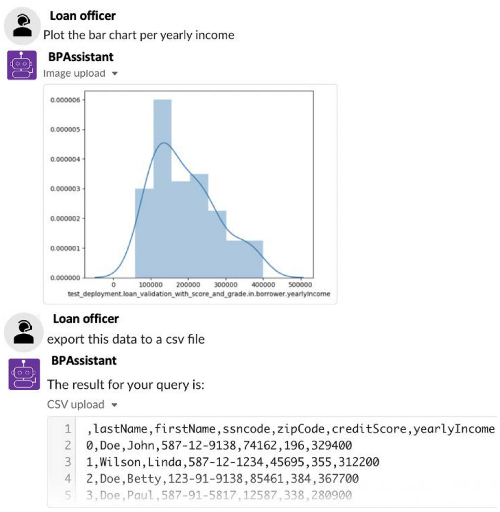
Table 2. Sample conversation with *BPAssistant*

Conversation	Responding agent
Loan Officer: Who are the top 3 borrowers with average amount more than 10000 *BPAssistant*: These are the average value: 1). 584,917$ for J. Smith, 2). 575,692$ for V. Doe, 3). 557,615$ for Y. Doe	Data query
Loan Officer: List all borrowers with yearly income more than 50000 but credit score less than 150 *BPAssistant*: Total records found are 82. Here is the link: <url>	Data query
Loan Officer: Plot the bar chart per yearly income *BPAssistant*: <image> (Fig. 6)	Visualization
Loan Officer: Export this data to a CSV file *BPAssistant*: The result for your query is: <csv> file (Fig. 6)	Data export
Manager: Hello *BPAssistant*: Hi there	Chit-chat
Manager: How many travel requests does John Smith have? *BPAssistant*: John Smith has 1 application	Travel query
Manager: Approve John Smith's request *BPAssistant*: John Smith's application has been approved	Business process Task execution

4.3 Discussion

Based on our prototypical implementation above, we observed multiple advantages and some limitations of the proposed framework. First, the assistant successfully handled the users' diverse requests by orchestrating agents from multiple domains. Business users interacted with RPAs such as the "alerts" agent or the "business rules" agent through natural language without switching between multiple interfaces. Querying their data did not require business users to possess the knowledge of formulating formal queries executed on their databases; they simply formulated natural language statements using business domain terminology that they frequently use. Performing tasks within their business processes no longer required them to juggle multiple user interfaces either.

The framework eliminated the need to create custom RPAs by the business user; instead, developers created general RPAs that could be customized through natural language by the business user in the assistant. However, this shift came at the cost of creating conversational wrappers for the RPAs. Future iterations



Loan officer
Plot the bar chart per yearly income

BPAssistant
Image upload

Loan officer
export this data to a csv file

BPAssistant
The result for your query is:
CSV upload

```
,lastName,firstName,ssncode,zipCode,creditScore,yearlyIncome
0,Doe,John,587-12-9138,74162,196,329400
1,Wilson,Linda,587-12-1234,45695,355,312200
2,Doe,Betty,123-91-9138,85461,384,367700
3,Doe,Paul,587-91-5817,12587,338,280900
```

Fig. 6. Multi-modal responses from the *BPAssistant*.

should consider approaches to automate this task to further reduce the coding overhead. Additionally, automated composition techniques, such as the ones explored in [5,20], can further reduce the overhead of authoring these agents by automatically composing sophisticated agents out of more atomic skills.

As more heterogeneous agents are added to the assistant, the orchestration problem becomes more complex, especially when considering posterior orchestration approaches. One method to reduce the cost of previewing agent responses is to implement a hybrid orchestration method that adopts an apriori algorithm to select a subset of agents to ask for a preview. This reduces the computational cost as the system is scaled, while maintaining high selection accuracy.

The stateless orchestrator offered a lightweight solution to enable cooperation among agents, the second contribution of this work. However, as the size of data exchanged between agents increases, a stateful implementation may be more suitable. Shareable data can be stored in memory while the orchestrator keeps track of other state-related information to handle more complex dialog constructs and cooperation opportunities. A stateful orchestrator that combines apriori and posterior models could also alleviate the problem of legacy programs not having a preview mode to invoke in posterior orchestrators. The statefulness of the orchestrator would compensate for the missing preview responses by using other features of the system to make the best selection possible.

Furthermore, the orchestrator's modular principles simplify the problem of system lifecycle. As business processes become obsolete, the agents linked to them can simply be deactivated without incurring costs to update the framework or other agents. The framework also supports agent versions: as updates to agents are rolled out, the affected agents can be independently updated.

We believe this framework generalizes well to many types of RPAs because the goal of RPAs is to reduce the amount of work done by humans. Hence, they require minimal human intervention beyond launching the RPAs which can be implemented as a natural language instruction. Functions that still require a large amount of input from users may not be suitable for conversational interaction and should remain in their original user interface. However, as the number of RPAs increases, users will have too many RPAs to keep track of individually; we believe a unified, conversational interface would simplify the life of users.

In summary, our framework addressed some of the key weaknesses of RPAs, namely providing an interactive, human-in-the-loop automation assistant and cooperative RPAs. Even though the framework can undergo further improvements, it is a solid step forward towards interactive business process automation that can gain business users' trust in process automation solutions and advance service lines like customer care and process automation.

5 Conclusion

Conversational automation solutions will be critical to the digital transformation. To that end, we presented a framework that combines RPAs with conversational agents (or chatbots), both popular paradigms in business enterprises, to create an interactive business process automation solution. The framework relies on multi-agent orchestration where conversational agents are composed from RPA skills. The resulting assistant allows business users to monitor and customize their business process automation solutions through natural language. Future work will incorporate more sophisticated orchestration models and autonomous agents to address existing challenges in the current framework.

Acknowledgement. The authors would like to acknowledge Tathagata Chakraborti, Pierre Feillet, and Stephane Mery for the valuable conversations that contributed to expanding the vision of this work.

References

1. van der Aalst, W.M.P., Bichler, M., Heinzl, A.: Robotic process automation. Bus. Inf. Syst. Eng. **60**(4), 269–272 (2018). https://doi.org/10.1007/s12599-018-0542-4
2. Agostinelli, S., Marrella, A., Mecella, M.: Towards intelligent robotic process automation for BPmers. arXiv preprint arXiv:2001.00804 (2020)
3. Biscotti, F., Mehta, V., Villa, A., Bhullar, B., Tornbohm, C.: Market share analysis: robotic process automation, worldwide, 2019. Technical report (2020)
4. Camargo, M., Dumas, M., González-Rojas, O.: Learning accurate LSTM models of business processes. In: Hildebrandt, T., van Dongen, B.F., Röglinger, M., Mendling, J. (eds.) BPM 2019. LNCS, vol. 11675, pp. 286–302. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26619-6_19
5. Chakraborti, T., Khazaeni, Y.: D3BA: a tool for optimizing business processes using non-deterministic planning. In: AAAI IPA (2020)

6. Fernandez, D., Aman, A.: Impacts of robotic process automation on global accounting services. Asian J. Acc. Gov. **9**, 123–132 (2018)
7. Galitsky, B.: A content management system for chatbots. Developing Enterprise Chatbots, pp. 253–326. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-04299-8_9
8. Gao, J., van Zelst, S.J., Lu, X., van der Aalst, W.M.P.: Automated robotic process automation: a self-learning approach. In: Panetto, H., Debruyne, C., Hepp, M., Lewis, D., Ardagna, C.A., Meersman, R. (eds.) OTM 2019. LNCS, vol. 11877, pp. 95–112. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-33246-4_6
9. Heo, M., et al.: Chatbot as a new business communication tool: the case of naver talktalk. Bus. Commun. Res. Pract. **1**(1), 41–45 (2018)
10. Houy, C., Hamberg, M., Fettke, P.: Robotic process automation in public administrations. In: Digitalisierung von Staat und Verwaltung (2019)
11. Huang, T.H., Chang, J.C., Bigham, J.P.: Evorus: a crowd-powered conversational assistant built to automate itself over time. In: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, pp. 1–13 (2018)
12. Jakob, M., Krcmar, H.: Which barriers hinder a successful digital transformation in small and medium-sized municipalities in a federal system? In: Central and Eastern European eDem and eGov Days, pp. 141–150 (2018)
13. Jan, S.T., Ishakian, V., Muthusamy, V.: AI trust in business processes: the need for process-aware explanations. In: IAAI 2020 (2020)
14. Kalia, A.K., Telang, P.R., Xiao, J., Vukovic, M.: Quark: a methodology to transform people-driven processes to chatbot services. In: Maximilien, M., Vallecillo, A., Wang, J., Oriol, M. (eds.) ICSOC 2017. LNCS, vol. 10601, pp. 53–61. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69035-3_4
15. Lacity, M., Willcocks, L.P., Craig, A.: Robotic process automation: mature capabilities in the energy sector. Technical report (2015)
16. Leopold, H., van der Aa, H., Reijers, H.A.: Identifying candidate tasks for robotic process automation in textual process descriptions. In: Gulden, J., Reinhartz-Berger, I., Schmidt, R., Guerreiro, S., Guédria, W., Bera, P. (eds.) BPMDS/EMMSAD -2018. LNBIP, vol. 318, pp. 67–81. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-91704-7_5
17. Marrella, A.: What automated planning can do for business process management. In: Teniente, E., Weidlich, M. (eds.) BPM 2017. LNBIP, vol. 308, pp. 7–19. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-74030-0_1
18. Matthies, C., Dobrigkeit, F., Hesse, G.: An additional set of (automated) eyes: chatbots for agile retrospectives. In: Proceedings of the 1st International Workshop on Bots in Software Engineering, pp. 34–37. IEEE Press (2019)
19. Moffitt, K.C., Rozario, A.M., Vasarhelyi, M.A.: Robotic process automation for auditing. J. Emerg. Technol. Account. **15**(1), 1–10 (2018)
20. Muise, C., et al.: Planning for goal-oriented dialogue systems. arXiv preprint arXiv:1910.08137 (2019)
21. Muthusamy, V., Slominski, A., Ishakian, V.: Towards enterprise-ready AI deployments minimizing the risk of consuming AI models in business applications. In: 2018 First International Conference on Artificial Intelligence for Industries (AI4I), pp. 108–109. IEEE (2018)
22. Papageorgiou, D.: Transforming the HR function through robotic process automation. Benefits Q. **34**(2), 27–30 (2018)
23. Rizk, Y., et al.: A unified conversational assistant framework for business process automation. In: AAAI IPA (2020)

24. Rizk, Y., Isahagian, V., Unuvar, M., Khazaeni, Y.: A snooze-less user-aware notification system for proactive conversational agents. In: Intelligent User Interfaces Workshop on User-Aware Conversational Agents (2020)
25. Rosemann, M.: Trust-aware process design. In: Hildebrandt, T., van Dongen, B.F., Röglinger, M., Mendling, J. (eds.) BPM 2019. LNCS, vol. 11675, pp. 305–321. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26619-6_20
26. Seeliger, A., Sánchez Guinea, A., Nolle, T., Mühlhäuser, M.: ProcessExplorer: intelligent process mining guidance. In: Hildebrandt, T., van Dongen, B.F., Röglinger, M., Mendling, J. (eds.) BPM 2019. LNCS, vol. 11675, pp. 216–231. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26619-6_15
27. Sen, J., et al.: Natural language querying of complex business intelligence queries. In: Proceedings of the 2019 International Conference on Management of Data, pp. 1997–2000. ACM (2019)
28. Stople, A., Steinsund, H., Iden, J., Bygstad, B.: Lightweight it and the it function: experiences from robotic process automation in a Norwegian bank. Bibsys Open J. Syst. **25**(1) (2017)
29. Tello, G., Gianini, G., Mizouni, R., Damiani, E.: Machine learning-based framework for log-lifting in business process mining applications. In: Hildebrandt, T., van Dongen, B.F., Röglinger, M., Mendling, J. (eds.) BPM 2019. LNCS, vol. 11675, pp. 232–249. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26619-6_16
30. Upadhyay, S., Agarwal, M., Bounneffouf, D., Khazaeni, Y.: A bandit approach to posterior dialog orchestration under a budget. In: NeurIPS Conversational AI Workshop (2019)
31. Weske, M.: Business Process Management (2012)
32. Wróblewska, A., Stanisławek, T., Prus-Zajączkowski, B., Garncarek, Ł.: Robotic process automation of unstructured data with machine learning. Ann. Comput. Sci. Inf. Syst. **16**, 9–16 (2018)
33. Wurm, B., Goel, K., Bandara, W., Rosemann, M.: Design patterns for business process individualization. In: Hildebrandt, T., van Dongen, B.F., Röglinger, M., Mendling, J. (eds.) BPM 2019. LNCS, vol. 11675, pp. 370–385. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26619-6_24
34. Yurochkin, M., Upadhyay, S., Bouneffouf, D., Agarwal, M., Khazaeni, Y.: Online semi-supervised learning with bandit feedback. In: LLD Workshop at ICLR (2019)