Quark: A Methodology to Transform People-Driven Processes to Chatbot Services

Anup K. Kalia[1(✉)], Pankaj R. Telang[2], Jin Xiao[1], and Maja Vukovic[1]

[1] IBM T.J. Watson, Yorktown Heights, NY, USA
anup.kalia@ibm.com, {jinoaix,maja}@us.ibm.com
[2] SAS Institute Inc., Cary, NC, USA
ptelang@gmail.com

Abstract. Human is a key cost factor in today's service- and business-oriented processes. To reduce labor, we propose an approach to convert people driven processes to a chatbot service. Current approaches to create a chatbot service are based on formal representations or dialog based methodologies. Formal representations provide techniques for soundness verification and exception handling, however, do not provide a software methodology that capture steps for developers to build a chatbot service. Dialog based methodologies provide different step-wise approaches to create a chatbot service, however, ignore the formal aspects. To bridge the gap, we propose a novel methodology, Quark, that guides developers in producing a model that is complete and sound. Specifically, Quark takes a business process flow as input and produces a Watson Conversation model. Quark employs the notions of goals and commitments which provide a formal means for completeness and soundness. We present Quark using a change management process scenario.

1 Introduction

Traditional business processes involve multiple process steps that humans execute. Such processes suffer from unpredictable delays and errors caused by the humans. The human errors may arise due to inadequate skill level or other cognitive state such as disinterest, distraction, and tiredness. The delays and errors can be reduced by employing automated agents for a subset of business tasks that do not need human oversight. An automated agent that provides an effective natural language interface for a human to interact is a chatbot.

To ensure a desired business outcome, the human-chatbot interactions need to be designed in a principled manner. Researchers have proposed various approaches for designing human-chatbot interactions. These approaches are either too formal for practitioners' effective use, or too informal leading to specifications that cannot be effectively verified. We propose a novel methodology, Quark, to bridge the gap between the formal and informal approaches.

Quark employs well studied abstractions of goals and commitments [11,13] for designing the human-chatbot interactions. A goal models a condition that a human or an (automated) agent desires to bring about. In a commitment,

M. Maximilien et al. (Eds.): ICSOC 2017, LNCS 10601, pp. 53–61, 2017.
https://doi.org/10.1007/978-3-319-69035-3_4

a debtor agent commits to a creditor agent to bring about a consequent condition if an antecedent condition holds.

Quark considers a business process model as its input and produces a dialog model. Specifically, we adopt BPMN [10] for the process model, and Watson Conversation [7] for the dialog model. From the process model, Quark identify roles that can be automated by a chatbot. For each role, Quark identifies goals and commitments. From the goals and commitments, Quark produces a set of interactions that are complete and sound. Finally, it generates a set of intents and a dialog model for the Watson conversation to build a chatbot service.

Contributions. This paper proposes a novel methodology, Quark, for developing a Watson Conversation model starting from a business process model. We demonstrate the methodology on a change management process.

2 Related Work

Researchers have extensively studied the topic of developing conversation models. For example, in services, Ardissono et al. [2] propose a conversation model that enables a conversation flow between the web-service consumers and the web-service providers. Bentahar et al. [3] provide a formal model of conversations based on the concepts of commitments and arguments. Cost et al. [5] propose Colored Petri Nets (CPN) for modeling conversations. Nezhad et al. [9] propose eAssistant that identifies actions in terms of request and promises to auto-triage the user conversations.

In the area of dialog-based approaches Traum [6] provides a methodology to create a computational model for a virtual human meant to operate in a specific domain. In terms of robustness checking, Traum do not clearly emphasize that his model will be robust on all interaction paths. Alès et al. [1] provide a methodology to extract a dialog model from a corpus by extracting several aspects from human dialogs, such as speech acts, social aspects, and gazes.

In multiagent systems, Bresciani et al. [4] propose the Tropos methodology that allows developers to design softwares in terms of goals. The methodology Gaia [14] assists developers to design organizations using responsibilities, permission, activities, and protocols. The methodology Comma [12] helps developers to capture business scenarios using commitments and creates a process that is sound with respect to commitments. The methodology Muon [8] helps developers to capture commitments semantics from interaction scenarios, handle exceptions, and then use the semantics to create sound processes. These methodologies unlike dialog based methodologies are formal, however, do not provide an approach to create a chatbot service.

From the related work, we infer the following requirements for a chatbot service: (1) ascribe meanings to messages or requests, (2) provide a meaningful response at each conversational turn that ensures reliability and flexibility provided by the chatbot, (3) ensure soundness of a conversation. To meet the requirements, we use the notion of goals and commitments to model a chatbot service. Goals capture the agent intentions, and commitments capture the meanings of messages exchanged between the participants in a service. The meanings provide a basis for verifying the soundness of conversations.

3 Quark Methodology

To describe our methodology, we consider the change management business process. In the process, a user interacts with a help desk to provide its request. The help desk validates the request of the user and sends it to a dispatcher. The dispatcher comprehends the intent of the request (e.g., a database change request or a memory change request) and sends it to the appropriate subject matter expert (SME). The SME extracts the parameters in the request. If the SME finds a parameter missing in the request, the SME requests the parameter from the user. Once the user provides the missing parameter, the SME sends the complete parameters to the change owner for a technical risk assessment. Then, the change owner sends them to the account owner for a business risk assessment. The account owner after its assessment, sends the request to the approver. Once the approver approves the request, it sends the request to the change owner. The change owner sends the request to the executor. The executor executes the request and sends the report to the user.

Quark takes a business process model as its input and produces an IBM Watson model of human-chatbot interactions that are necessary to realize the business process. We now describe the steps of the Quark methodology.

3.1 M$_1$: Identify Roles Served by Humans That Can Be Automated

This step identifies roles served by humans that can be automated by a service. The step requires organizational knowledge and domain expertise. A human is necessary for a role in a process if that role's business tasks are not clearly defined, or if the business tasks inherently require human insight. Generally, a role whose business tasks are formally defined is a good candidate for automation. After identifying the roles for automation, this step combines those roles in a single role. For convenience, we call this role BOT. Next, the step reduces the business process by removing the roles identified for automation, and adds the single role BOT. All the tasks under the roles identified for automation are transferred over to the role BOT.

For the change management process, we identify HELP DESK, DISPATCHER, SME, ACCOUNT OWNER, CHANGE OWNER, APPROVER, and EXECUTOR as roles that can be automated.

3.2 M$_2$: Identify Goals of Each Role

This step identifies the goals of each role. For each role, the tasks from the business process map into the (success conditions of) goals of that role. Specifically, for a task a in the process model, we specify a goal $G(x, p, a, f)$ in which x

is the role, p is the preceding task that is a necessary precondition for task a, the success condition is the task a, and the failure condition is f. If the failure condition is not explicitly modeled in the business process, then process domain expertise is necessary to identify it. For example, USER wants to execute its change request. Thus, *execute change* is as a goal of USER. Similarly, given a change request, BOT needs to identify the correct parameters for the change. Thus, *correct params* is a goal of BOT with the precondition that the change request has been provided.

Table 1 shows the goals of the roles: USER and BOT. In the table, t is a timeout that represents the failure condition, and the operator $\wedge$ represents logical conjunction.

Table 1. Roles and their goals for the change management process.

Actors	Goals
USER	$\mathsf{G}_1 = \mathsf{G}(\text{USER}, T, \textit{execute change}, t)$
BOT	$\mathsf{G}_2 = \mathsf{G}(\text{BOT}, \textit{provide request}, \textit{validate request}, t)$
	$\mathsf{G}_3 = \mathsf{G}(\text{BOT}, \textit{validate request}, \textit{identify intent}, t)$
	$\mathsf{G}_4 = \mathsf{G}(\text{BOT}, \textit{identify intent}, \textit{correct params}, t)$
	$\mathsf{G}_5 = \mathsf{G}(\text{BOT}, \textit{correct params}, \textit{perform tech risk assessments}, t)$
	$\mathsf{G}_6 = \mathsf{G}(\text{BOT}, \textit{correct params}, \textit{perform biz risk assessments}, t)$
	$\mathsf{G}_7 = \mathsf{G}(\textit{Bot}, \textit{perform tech risk assessments} \wedge \textit{perform biz risk assessments}, \textit{approve}, t)$
	$\mathsf{G}_8 = \mathsf{G}(\text{BOT}, \textit{approve}, \textit{execute change}, t)$
	$\mathsf{G}_9 = \mathsf{G}(\text{BOT}, \textit{execute change}, \textit{send report to user}, t)$

3.3 M$_3$: Identify Commitments Between Roles

This step identifies the commitments by analyzing goals of each role. For each goal, the step first asks the question: can the role satisfy the goal on its own, that is, can the role bring about the success condition of the goal on its own? If yes, then a commitment is not necessary. If no, then the step adds a commitment. The different elements of the commitment are identified as below.

– Debtor: is the role that can be bring about the goals's success condition.
– Creditor: is the role that has the given goal.
– Antecedent: is a form of a precondition that the creditor brings about in exchange of bringing about the goal's success condition. The precondition might be a form of a payment, or some other action. In some cases, the precondition might be already met. In those cases, it is set to true ($\top$).
– Consequent: is the success condition of the goal.

In the change management scenario from user has the goal of *execute change*. The user cannot satisfy this goal on her own. BOT can bring about *execute change*. So the step identifies a commitment: C(BOT, USER, provide request, send report to user). Observe that provide request is a necessary precondition for BOT to execute the change and to send a report. Similarly, BOT has a goal to identify correct parameters for the change. BOT cannot achieve this goal on her own. So, the step identifies a commitment: C(USER, BOT, modification request, correct params). Here, USER commits to BOT to provide the correct parameters when BOT makes a modification request.

Table 2 shows the commitments from the change management process.

Table 2. Commitments for the change management process.

Roles	Commitments
USER	C_1 = C(USER, BOT, *modification request*, *correct params*)
BOT	C_2 = C(BOT, USER, *provide request*, *send report*)

3.4 M$_4$: Produce a Set of Interactions

The fourth step in our methodology is to identify a set of interactions based on the roles, goals, and commitments identified from Steps M$_1$, M$_2$, and M$_3$. Kalia et al. [8] provides several guidelines for developers to create such interactions. They are as follows.

– *Interactions should represent the core positive outcomes.* For example the scenario where USER provides a request and it's executed by BOT provides desirable enactments. However, a scenario where USER refusing to provide appropriate request when asked for is not useful.
– *Interactions should reflect social or organizational relationships.* For example, the interactions between BOT and USER should lead to creation of commitments.
– *Interactions should ignore irrelevant messages.* For example, greeting message exchanged between USER and BOT can be ignored.
– *Interactions should avoid irrelevant roles and role instances.* For example, we cannot add additional roles that are not part of desirable enactments.

Based on the guidelines, we create a set of interactions that activate and satisfy the goals and commitments for USER and BOT. Table 3 describes a set of interactions that captures the modeled goals and commitments.

Table 3. A set of interactions between USER(S) and BOT(R), act represents active, and sat represents satisfied.

S	R	Message	Goals	Commitments
USER	BOT	*Hi*		
BOT	USER	*provide your request*		act(C_2)
USER	BOT	*add 2GB of memory to my server*	act(G_1) ∧ act(G_2) ∧ act(G_3) ∧ act(G_4)	det(C_2)
BOT	USER	*provide the server details*	sat(G_2, G_3)	act(C_1) ∧ det(C_1)
USER	BOT	*server info is cobalt.ibm.com*	sat(G_4) ∧ act(G_4) ∧ act(G_5) ∧ act(G_6) ∧ act(G_7) ∧ act(G_8) ∧ act(G_9)	sat(C_1)
BOT	USER	*here is the report*	sat(G_4) ∧ sat(G_5) ∧ sat(G_6) ∧ sat(G_7) ∧ sat(G_8) ∧ sat(G_9)	sat(C_2)

3.5 M$_5$: Repeat Steps M$_2$ and M$_3$ to Produce Additional Goals and Commitments

The fifth step of our methodology is to repeat steps M$_2$ and M$_3$ to identify additional goals and commitments required to provide robustness to the chatbot service. In this step, developers can consider scenarios that are not present in the current change management process. For example, developers can think of possible deviations such as *what happens if* BOT *could not identify the intent of* USER*'s request?* or *what happens if* BOT *does not approve* USER*'s request?*. To address such scenarios, we can first create goals and commitments required to address the scenarios and then we can add additional set of interactions to satisfy the goals and commitments.

For addressing *what happens if* BOT *could not identify the intent of* USER*'s request?*, we look for the goals and commitments. Since, the goal of BOT to identify the intent is present in Table 1, we do not add any new goal. To achieve the goal, BOT needs a new request from USER. Thus, it requests a commitment from USER to provide a correct change request. If USER agrees, the new commitment is created. Similarly, for the second scenario, the goal of BOT to approve is present in Table 1. However, if BOT cannot achieve its goal, it can request USER to create a commitment to provide a correct change request. In both the cases, USER has the autonomy to create new commitments or terminate the existing commitments. Based on new commitments identified we generate a new set of interactions that create and satisfy the commitments. We show the interactions in the next step of our methodology (Table 4).

Table 4. Additional commitments for the change management process.

Roles	Commitments
USER	$\mathsf{C}_3 = \mathsf{C}($USER, BOT, *intents not identifiable*, *provide new request*$)$
USER	$\mathsf{C}_4 = \mathsf{C}($USER, BOT, *cannot be approved*, *provide new request*$)$

3.6 M$_6$: Translate the Interactions to IBM Watson Model

In this step, we accomplish two things. First, we add the natural language interface to the current methodology. Then, we build a chatbot service. In IBM Watson's model [7], the first step is create intents and entities.

An intent captures the purpose of USER'S request. BOT, identifies the intent and then make an appropriate response. The guidelines to construct an intent is as follows.

– *Collect as many as* USER*'s request and categorize them.* For example, for the change management process, we can gather USER'S request for different kinds of change requests such as hardware, database, os management, and so on.
– *If intents are too similar cluster them together or else keep them separate.* For example, we can keep hardware related change requests such as cpu and memory specific changes together. Similarly, we can keep database run operations and management together.
– *Keep refining the intents.* With more data, intents can be refined further.

Based on specific entities, BOT chooses specific actions to perform. For example, consider two change requests. One, where USER requests to add memory to its VM and another where USER requests to add cpu to its VM. Both the change request gets identified with #hardware intent. Then, based on the entities such as *cpu* and *memory*, appropriate action is taken by BOT.

A child node is same as a node, however, it matches a user requests with an entity as its condition. Recall, that the output from the previous step M$_4$ was a set of additional commitments. Using the additional commitments, we refine the interactions in Table 5. We considered the interactions that is verified with respect to goals and commitments for creating a dialog model. Based on user's request, we identify intents and entities.

Based on intents and entities identified, we construct a dialog model as shown in Fig. 1. Consider an example, if USER provides a hardware request *add 2cpu to server cobalt.ibm.com*. The dialog model in can identify cpu as one of the entity, however, it cannot identify the server information, if it's not explicitly provided. Thus, goals and commitments to validate cpu request could have been used to refine the dialog model.

Table 5. A set of interactions between USER and BOT that captures the intents I and the entities E.

S	R	Message	Intents	Entities
USER	BOT	*Hi*	I_1 = #conv_start	
BOT	USER	*provide your request*	resp(I_1)	
USER	BOT	*add 2 to my server*	I_2 = #none	
BOT	USER	*provide a valid request*	resp(I_2)	
USER	BOT	*add 2GB memory to my server*	I_3 = #hardware	E_1 = memory
BOT	USER	*provide server details*	resp(I_3)	
USER	BOT	*server info is cobalt.ibm.com*		E_2 = server
BOT	USER	*here is the report*	resp(E_2)	

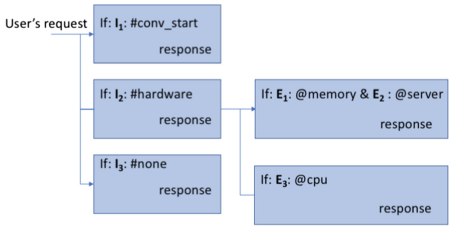

Fig. 1. *Output*: the dialog model created from intents and entities.

4 Conclusion and Future Work

This paper describes a novel methodology for developing human-chatbot interactions. We develop Quark using a typical change management process. We formally define the concepts that Quark employs. We evaluate Quark on a loan processing scenario. Quark produces human-chatbot interactions that are complete and sound.

In the future, we plan to develop tooling for Quark that will guide developers through the methodology steps. We will also conduct developer studies to evaluate Quark's effectiveness.

References

1. Alès, Z., Duplessis, G.D., Şerban, O., Pauchet, A.: A methodology to design human-like embodied conversational agents. In: International Workshop on Human-Agent Interaction Design and Models, Valencia, Spain, pp. 1–16 (2012)
2. Ardissono, L., Cardinio, D., Petrone, G., Segnan, M.: A framework for the server-side management of conversations with web services. In: Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers and Posters, pp. 124–133. ACM, New York (2004)

3. Bentahar, J., Moulin, B., Chaib-draa, B.: Towards a formal framework for conversational agents. In: Proceedings of Agent Communication Languages and Conversation Policies Workshop, Melbourne, Australia, pp. 1–11 (2003)
4. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: an agent-oriented software development methodology. J. Auton. Agents Multi Agent Syst. **8**(3), 203–236 (2004)
5. Cost, R.S., Chen, Y., Finin, T., Labrou, Y., Peng, Y.: Using colored petri nets for conversation modeling. In: Dignum, F., Greaves, M. (eds.) Issues in Agent Communication. LNCS, vol. 1916, pp. 178–192. Springer, Heidelberg (2000). doi:10.1007/10722777_12
6. Traum, D.: Talking to virtual humans: dialogue models and methodologies for embodied conversational agents. In: Wachsmuth, I., Knoblich, G. (eds.) Modeling Communication with Robots and Virtual Humans. LNCS, vol. 4930, pp. 296–309. Springer, Heidelberg (2008). doi:10.1007/978-3-540-79037-2_16
7. IBM. Conversation. IBM (2016). https://www.ibm.com/watson/developercloud/conversation.html
8. Kalia, A.K., Singh, M.P.: Muon: designing multiagent communication protocols from interaction scenarios. JAAMAS **29**(4), 621–657 (2015)
9. Nezhad, H.R.M., Gunaratna, K., Cappi, J.: eAssistant: cognitive assistance for identification and auto-triage of actionable conversations. In: Proceedings of the 26th International Conference on World Wide Web Companion, WWW, Perth, Australia, pp. 89–98 (2017)
10. OMG. Business process model and notation (BPMN), version 2.0 beta. Object Management Group (2010). http://bpmn.org/
11. Singh, M.P.: An ontology for commitments in multiagent systems: toward a unification of normative concepts. Artif. Intell. Law **7**(1), 97–113 (1999)
12. Telang, P.R., Kalia, A.K., Singh, M.P.: Engineering service engagements via commitments. IEEE Internet Comput. **18**(3), 46–54 (2014)
13. Telang, P.R., Meneguzzi, F., Singh, M.P.: Hierarchical planning about goals and commitments. In: Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems, St. Paul, MN, USA, pp. 877–884 (2013). IFAAMS
14. Woolridge, M., Jennings, N., Kinny, D.: The gaia methodology for agent-oriented analysis and design. J. Auton. Agents Multi Agent Syst. **3**(3), 285–312 (2000)