A Conceptual Framework for Engineering Chatbots

Pankaj R. Telang
SAS Institute Inc.

Anup K. Kalia
Maja Vukovic
IBM Thomas J. Watson Research Center

Rahul Pandita
Phase Change Software LLC

Munindar P. Singh
North Carolina State University

The increasing popularity of chatbots as virtual assistants has lead to many organizations releasing *If-This-Then-That* frameworks to engineer such chatbots. However, these frameworks often result in inflexible and difficult-to-maintain chatbots. This paper outlines a high-level conceptual framework for realizing flexible chatbots founded upon agent-oriented abstractions: goals, plans, and commitments.

The potential applications and popularity of chatbot technology has resulted in leading technology vendors such as IBM, Facebook, Microsoft, and Google to releasing frameworks to build such chatbots. Most such commercially available frameworks reduce the task of engineering chatbots to a variant of the *If-This-Then-That* (IFTTT) style of programming that was at the core of expert systems proposed in the 1980s.[1]

However, the IFTTT approach is inadequate in terms of the flexibility and maintainability of the chatbots produced. The resulting chatbots all too often are monolithic and they mix rules for managing dialog with rules for executing business logic and generating responses. Additionally, when a chatbot must interact with third-party services to orchestrate a workflow, the resulting orchestration logic is interleaved into the IFTTT rules. Furthermore, IFTTT rules are often order sensitive. As the capability required of a chatbot evolves over time, the complexity of the rules implementing the chatbot increases. Thus, even a simple modification to a chatbot may require substantial effort to 1) determine where in the sequence a new rule should be added; and 2) ensure that the new rule does not cause existing functionality to break due to interactions among the rules.

We posit that leveraging well-studied abstractions of goals and commitments from the area of artificial intelligence (AI) and multiagent systems allows for a flexible realization of chatbots. Goals capture an agent's intentions, and commitments capture meaningful business relationships between agents. We describe how employing goals and commitments enables us to model a chatbot that 1) can be verified at design time or runtime[4]; 2) offers flexible enactments; and 3) provides a basis for judging correctness.

Published by the IEEE Computer Society
1089-7801/18/$33.00 ©2018 IEEE

We next present an enterprise IT management scenario to help illustrate our framework. Our framework, however, can be readily applied to other domains, including consumer chatbots.

CHATBOTS IN CONTEXT OF ENTERPRISE IT SERVICE MANAGEMENT

One important application area for enterprise chatbots is information technology (IT) service management. As IT outsourcing has matured as a business, the service offered has high variability in execution and implementations across clients and hosting environment. This variability demands manual intervention and supervision, which affects the speed and quality of process execution. IT service providers are, therefore, under competitive pressure to continually improve their service quality and reduce operating costs through increased automation.[2] That is, complexity, coupled with a persistent demand to lower service delivery cost, drives enterprises to adopt chatbots in their end-to-end processes.

Scenario: Problem Remediation

Consider an IT service management scenario involving a chatbot. The chatbot interacts with a user to understand the problem: the user's server freezes when booted. It determines a solution: the server needs a firmware update. The chatbot interacts with the user, proposes the solution, explains the reasoning behind it, and requests the user's approval for updating the firmware. If the user agrees, the chatbot requests approvals from other parties whose applications run on the same server. Figure 1 shows a sample of the dialog taking place between the user (in blue) and the chatbot (in gray).

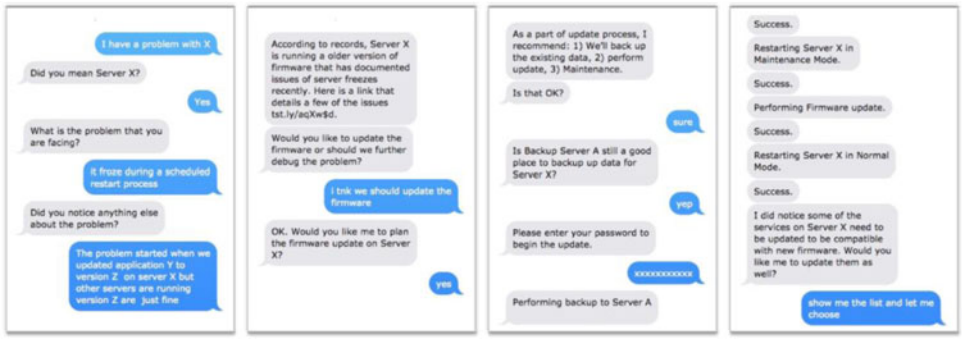

Figure 1. Sample conversation between a chatbot and a user.

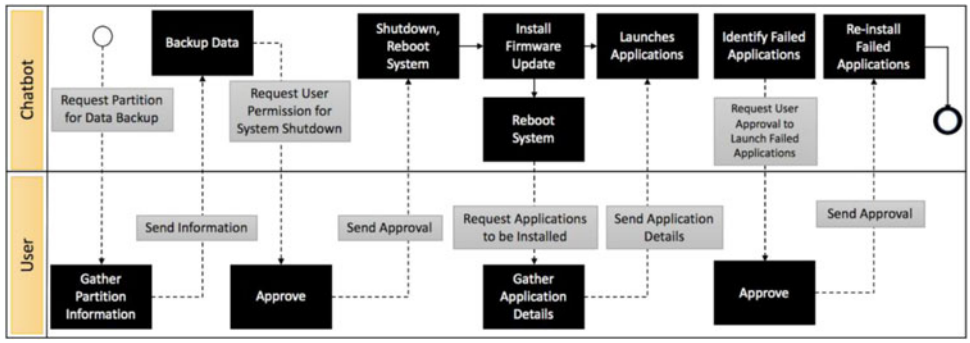

Figure 2. Plan generated by the chatbot to perform a firmware update.

Depending on user approval, the chatbot generates the plan shown in Figure 2 to update the firmware.

Proposed Conceptual Framework

The above-mentioned scenario uncovers enterprise chatbot capabilities that our conceptual framework, as illustrated in Figure 3, captures. Our conceptual framework consists of five components with the associated responsibilities: dialog manager (manages natural language dialog with the user); inference engine (extracts user intents); knowledge base (supports inference and planning); planner (produces execution plans); and external services interface (operationalizes plans).

We next introduce the key abstractions of goals, plans, and commitments from AI and multiagent systems. These abstractions have a theoretical foundation that enables formal design and runtime verification[4] and are accompanied by practical design methodologies.[5] Researchers have successfully employed these abstractions on engineering chatbots. For example, Kalia *et al.*[8] describe a methodology, Quark, to transform a business process into a chatbot service using goals and commitments. Whereas the Quark methodology focuses a on business process, present work outlines a generic conceptual framework to engineer chatbots across domains.

The notion of commitments[3] provides a basis for correct interactions, supporting verification while enabling flexibility in how communications may be constructed and ordered. A commitment C (debtor, creditor, antecedent, and consequent) means that the debtor promises a creditor to bring about a consequent condition if the antecedent condition holds true. For example, in our problem remediation scenario, the chatbot commits to the user that if the user provides symptoms related to her server freezing, the chatbot will identify and remediate the issue. The chatbot may carry out several interactions with the user and backend services to satisfy its commitment.

To capture a chatbot's internal state and abstract away from its detailed plan, we adopt the notion of goals to model what a chatbot seeks to bring about independently of its actions. Goals help relate a chatbot's internal reasoning to its interactions, as characterized by the commitments in which it features as a debtor or creditor. In our scenario, the chatbot has a (top-level) goal of remediating the problem of the system freezing.

Furthermore, the declarative nature of goals and commitments helps to deal with increasing complexity of the dialog models.

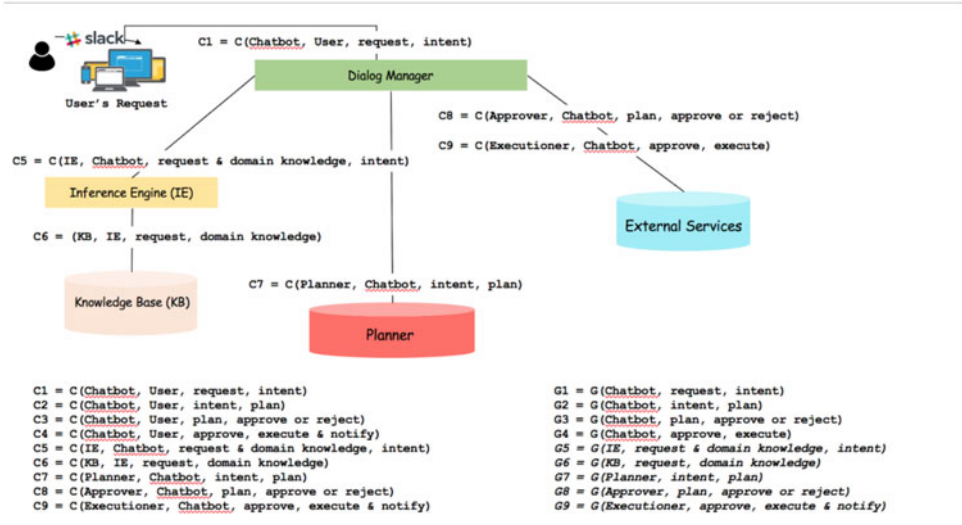

Figure 3. Conceptual framework for an enterprise chatbot.

We next go through each of the identified components and capture the potential benefits of using the aforementioned abstractions.

The *dialogue manager* manages a natural language dialog with a user and interacts with services such as an inference engine, knowledge base, planner, and external business services to carry out the user's request. The supported user interaction modalities may be text and voice.

Dialog management employs natural language processing techniques to extract meaning from a user's messages. For example, it may identify a symptom from a request by extracting named entities and employ the inference engine to capture the relationships between the entities and to invoke a specific rule to infer a symptom.

The current approach (see sidebar) to dialog management relies on intents and actions corresponding to intents. For example, once the chatbot infers the intent of a user's request, i.e., that a server freezes upon reboot, it invokes the firmware update action. The firmware update action internally invokes actions that need multiple interactions with the user and different services. Modeling such interactions naively often lead to inflexible dialogs that cannot readily accommodate changes in client requirements or backend services.

The *inference engine* receives requests from the dialog manager and interacts with the knowledge base to infer an appropriate intent based on a user's request. In addition, it may infer entities from the user's request to map them to appropriate actions.

The purpose of the inference engine is to identify a closest possible intent for a given request. In several cases, the initial corpus required to train a classifier could be small. Thus, the intent inference could have either low recall or low precision.

The *knowledge base* represents the chatbot knowledge, including inputs received from a user and about available services and how they relate to the domain. The knowledge base includes a knowledge graph constituting a set of entities and their relationships, and a set of rules that make inferences based on entities and their relationships. The chatbot designers need to populate the knowledge base with expert knowledge.

In existing frameworks, the knowledge base is often encoded into the same IFTTT rules that govern the chatbot actions. Decoupling knowledge from the chatbot actions promotes flexibility by enabling developing and updating each component independent of the other. In addition, an independent knowledge base enables chatbot "learning" without explicitly writing additional rules in chatbot interaction logic.

The *planner* provides the dialog manager with execution plans for each of its goals. If a plan fails to execute, the dialog manager requests the planner to provide an alternative plan to achieve the goal. In our scenario, to perform the firmware update, the dialog manager requests the planner to create a plan. The resulting plan is shown in Figure 2.

In contrast to existing frameworks, which encode all knowledge in IFTTT rules and their ordering, an explicit planner can create execution plans that are independent of chatbot interaction rules. Furthermore, separating concerns facilitates flexibility to independently update each component with minimal changes to the other.

The dialog manager interacts with *external services* to execute business functions, such as for change request approval, business intelligence service, and remote system management.

In our scenario, a change request approval service is enabled to help obtain approvals from those who may be affected by a firmware update of the user's server. The chatbot may employ a remote system management service to update the firmware. Existing approaches to specify the approval process use business process model and notation.[7] As a result, they cannot accommodate deviations that are not explicitly encoded in the process flow. Thus, a single workflow might fail to handle cases where the requirements keep changing. An example change in requirements is the addition of new approvers. We propose to model the approval process in terms of commitments that involve the dialog manager and approvers. Being declarative representations, commitments

can capture addition of approvals at ease and the verification of commitments can trigger the dialog manager to send a reminder to the approver to respond to the request.

Putting it together. In the problem remediation scenario, the chatbot queries the user for relevant information to build specific context regarding the server freezes. The dialog manager passes these details as input to the inference engine. The engine reasons with these details to determine that a firmware update will solve the problem. The dialog manager generates a plan of action with the help of the planner to interact with external services and execute the firmware update.

OTHER CONSIDERATIONS FOR CHATBOTS

Risk and policy management in enterprise chatbots: Enterprise chatbots may involve making expensive and potentially irreversible actions. For instance, a chatbot may need to consider the risk involved if a firmware update breaks a dependence or incurs another cost. The chatbot must have a predictable way of dealing with such situations.

Modeling dialogs in goal-based chatbots: Most existing implementations of chatbots support one-shot interactions. A chatbot accepts natural language text or utterance and comes up with the best match for the action to be taken. Although simple to engineer and suitable for short conversations in a closed domain, one-shot interactions do not yield natural conversations.

CONCLUSION

We proposed a conceptual framework for enterprise chatbots that employs well-studied abstractions of goals, commitments, and plans. A benefit of using these abstractions is that the complex chatbot can be developed in a flexible manner and as opposed to existing IFTTT frameworks that cause maintenance to be prohibitively resource and cost intensive.

SIDEBAR: FRAMEWORKS TO BUILD CHATBOTS

Vendors such as Google (Dialogflow), IBM (Watson Conversation),[6] Microsoft (LUIS), Facebook (Wit.ai), and Amazon (Lex) provide IFTTT-based methodologies for building chatbots. Typical chatbot implementations realized from these frameworks involve the following steps.

- Extract intents from an utterance. For example “perform the firmware upgrade on the server via bootable ISO image” means hardware:update_firmware.
- Extract entities from the user’s intent. For example, the aforementioned yields firmware and server.
- Extract context from user interactions. For example, extract username from asking the user.
- Apply a dialog model that applies the extracted intents, entities, and context to generate a response.

If the chatbot identifies a conflict in identifying an appropriate response, it can request for additional entities to clearly associate the user’s request to a specific intent.

ACKNOWLEDGMENTS

Thanks to Pradeep Murukannaiah, Ozgur Kafali, and Titus Barik for comments on previous versions of this draft.

REFERENCES

1. P. Jackson, *Introduction to Expert Systems*. Reading, MA, USA: Addison-Wesley, 1998.
2. N. Ayachitula *et al.*, “IT service management automation - A hybrid methodology to integrate and orchestrate collaborative human centric and automation centric workflows,” in *Proc. IEEE Int. Conf. Serv. Comput.*, Salt Lake City, 2007, pp. 574–581.

3. M. P. Singh, "An ontology for commitments in multiagent systems," *Artif. Intell. Law*, vol. 22, no. 60, pp. 97–113, 1999.
4. P. R. Telang and M. P. Singh, "Specifying and verifying cross-organizational business models: An agent-oriented approach," *IEEE Trans. Serv. Comput.*, vol. 5, no. 3, pp. 305–318, Third Quart. 2012.
5. P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos, "Tropos: An agent-oriented software development methodology," *Auton. Agents Multi-Agent Syst.*, vol. 8, no. 3, pp. 203–236, 2004.
6. Watson Assistant, 2018. [Online]. Available: https://www.ibm.com/watson/ai-assistant/
7. OMG, "Business process model and notation (BPMN), version 2.0 beta." [Online]. Available: http://bpmn.org/
8. A. K. Kalia, P. R. Telang, J. Xiao, and M. Vukovic, "Quark: A methodology to transform people-driven processes to chatbot services," in *Proc. Int. Conf. Serv. Comput.*, Malaga, 2017, pp. 545–560.

ABOUT THE AUTHORS

Pankaj R. Telang is a Senior Data Scientist with the Cyber Analytics R&D Department, SAS Institute, Cary, NC, USA. His research interests include service computing, data analytics, and cybersecurity. He received the Ph.D. degree in computer science from North Carolina State University, Raleigh, NC, USA. Contact him at ptelang@gmail.com

Anup K. Kalia is a Research Staff Member with the IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA, in cognitive service management team. His research interests include service computing, multiagent systems, cognitive science, and software engineering. He received the Ph.D. degree in computer science from North Carolina State University, Raleigh, NC, USA. Contact him at Anup.Kalia@ibm.com

Maja Vukovic is a Research Manager and a Research Staff Member with the IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA. Her research interests include services computing, software engineering, IT service management, and AI planning. He is a Senior Member of IEEE, a Member of IBM Academy Technology, and an IBM Master Inventor. Contact him at maja@us.ibm.com

Rahul Pandita is a Senior Research Scientist with Phase Change Software, Golden, CO, USA. His primary research interests include data science and automated software engineering. He specifically works on applying natural language processing techniques on software artifacts to improve developer productivity. Contact him at rpandita@phasechange.ai

Munindar P. Singh is a Professor in Computer Science and the Co-Director of the Science of Security Lablet, North Carolina State University, Raleigh, NC, USA. His research interests include the engineering and governance of sociotechnical systems. He is an IEEE Fellow, an AAAI fellow, a former Editor-in-Chief of the IEEE Internet Computing, and the current Editor-in-Chief of *ACM Transactions on Internet Technology*. Contact him at singh@ncsu.edu