Model-Driven Methodology for Developing Chatbots Based on Microservice Architecture

Adel Vahdati[a] and Raman Ramsin[b]
Department of Computer Engineering, Sharif University of Technology, Azadi Avenue, Tehran, Iran

Keywords: Chatbot, Model-Driven Methodology, Natural Language Processing, Microservice Architecture.

Abstract: With recent advancements in natural language processing algorithms and the emergence of natural language understanding services, chatbots have become a popular conversational user interface integrated into social networks and messaging services, providing businesses with new ways to engage with customers. Various tools and frameworks have been developed to create chatbots and integrate them with artificial intelligence services and different communication channels. However, developing chatbots is complex and requires expertise in various fields. Studies have shown that model-driven engineering can help overcome certain challenges of chatbot development. We propose a model-driven methodology that systematically manages the creation of an intelligent conversational agent. The methodology uses metamodels at different abstraction levels that enable the description of the problem domain and solution space. By providing a high-level structure based on microservice architecture, it improves maintainability, flexibility, scalability, and interoperability. A criteria-based analysis method has been used to evaluate the proposed methodology.

1 INTRODUCTION

Software systems are now utilizing a novel type of interface beyond the traditional GUI. Conversational agents, intelligent assistants, and conversational user interfaces (CUI) are gaining in popularity (Planas et al., 2021). Moreover, conversational agents are already supporting software development activities such as automation of deployment tasks, assigning errors and issues to team members, and task scheduling (Perez-Soler et al., 2020). Their integration into social networks as communication channels has facilitated stakeholder participation in task automation and collaborative modeling (Perez-Soler et al., 2019; Perez-Soler et al., 2020).

In conversational agents, user interaction is carried out by sending text, voice messages, or by using interactive images (as in Gesture Bots). In all cases, the agent has a mechanism for dialogue, and the only difference is in the interface/medium through which this dialogue takes place (Planas et al., 2021).

A chatbot simulates human conversation through two-way communication using natural language. A chatbot platform must offer these features to provide a useful conversation (Matic et al., 2021):

- Natural language processing (NLP) and natural language understanding (NLU): understanding user input and extracting relevant information.
- Conversation flow management
- Performing necessary actions: such as searching a database or calling other services.

Various tools and frameworks have been provided by leading companies such as Google (Dialogflow), Microsoft (Microsoft Bot Framework), Amazon (Amazon Lex), and IBM (Watson) to create conversational agents (Perez-Soler et al., 2021). These tools provide a framework, cloud environment, and GUI to define the conversation flow. Existing frameworks use machine learning (ML) algorithms to identify the user's intention based on the message sent by the user; for example, Amazon provides services such as Lex, Comprehend and Polly that can help create intelligent assistants (Mahmood et al., 2020).

In model-driven development (MDD), a system is modeled at different levels of abstraction. Model transformations are used for refining high-level abstract models into lower-level models, or code

[a] https://orcid.org/0009-0006-4571-8130
[b] https://orcid.org/0000-0003-1996-9906

Vahdati, A. and Ramsin, R.
Model-Driven Methodology for Developing Chatbots Based on Microservice Architecture.
DOI: 10.5220/0012433700003645

In *Proceedings of the 12th International Conference on Model-Based Software and Systems Engineering (MODELSWARD 2024)*, pages 247-254
ISBN: 978-989-758-682-8; ISSN: 2184-4348

(Rodrigues da Silva, 2015; Alam et al., 2018). In chatbot development, MDD can help reduce accidental complexity (Alam et al., 2018), leading to higher productivity, performance and reusability (Martínez-Gárate et al., 2023).

We propose a model-driven methodology that guides the process of creating a chatbot. This methodology encompasses four phases: analysis, design, implementation, and test; sets of activities and products have been specified for each phase, and metamodels have been defined at different levels of abstraction to describe the problem/solution domains.

In the analysis phase, we provide a metamodel called CRAC (short for Concept, Responsibilities, Asynchronous Collaboration) for problem domain analysis and requirements elicitation. We also provide a metamodel called Intent to describe the user goals and how they are expressed in natural language.

In the design phase, we propose a microservice architecture that helps improve maintainability, scalability and interoperability. By using various design patterns, we address the issue of dependency on NLU services or communication platforms. For recognizing user intent, we provide a metamodel called Prompt, which helps describe the prompts used for interaction with AI services such as ChatGPT; training phrases are automatically extracted and their key parameters are tagged by calling ChatGPT services based on Prompt models. In addition, we facilitate the description of conversation flow by providing a metamodel called Dialog.

In the implementation phase, models are generated based on platform-specific communication channels and NLU services. Since our proposed methodology is not dependent on any specific platform or service provider, these models and model transformations are introduced in a generic way, and their details are not provided at this stage; in future, we plan to define specific metamodels for common communication platforms and NLU services.

In the test phase, we have defined specific quality attributes along with a set of evaluation criteria and metrics. A criteria-based method is used to evaluate our proposed methodology based on generic software development criteria, as well as specific criteria related to MDD and chatbot development.

This paper is structured as follows: Section 2 provides a review of the research background; the challenges of chatbot development are discussed in Section 3; in Section 4, the proposed methodology and architecture are introduced; the proposed methodology is evaluated in Section 5; and in Section 6, the conclusions and the directions for future work are presented.

2 RELATED WORKS

Various frameworks have been proposed that facilitate the creation and deployment of chatbots. Xatkit (Daniel et al., 2020) is a chatbot development framework that uses MDD and domain-specific languages (DSL) to specify the chatbot's behavior independently of the platform. The designer defines the user's intentions and behavior by binding them to actions and responses, and the runtime engine deploys the chatbot, registers intents, establishes connections, and launches external services.

CONGA (Perez-Soler et al., 2021) is a web-based development environment that uses a DSL to model conversational agents. The specifications are analyzed and compiled into tools like Rasa or Dialogflow, and a recommendation component suggests the best tool for creating a chatbot.

Mahmood et al. (Mahmood et al., 2020) create a dynamic user interface by utilizing the features of microservice architecture and flexibility of natural languages. User intentions are identified based on requirements; utterances are then mapped to these intents. Open API specifications are used to create service models and orchestrate microservices based on their capabilities and availability.

Matic et al. (Matic et al., 2021) propose an architecture that allows the use of different NLU services without dependency on any specific provider. They provide a general metamodel for NLU services and two specific metamodels for Dialogflow and Rasa NLU services, along with mapping rules to automatically create the required objects/information.

Perez-Soler et al. (Perez-Soler et al., 2019) present a solution that automates the creation of a conversational agent that can model by using natural language. Users can express their ideas in an incomplete and inaccurate way, and the framework will store and refine the model.

Ed-douibi et al. (Ed-douibi et al., 2021) propose an approach that uses a chatbot as an interface for querying Open Data resources. Users can ask questions in natural language and the chatbot converts them into API requests. The API model is generated and annotated to provide domain-specific information for configuring the chatbot and querying Web APIs. Xatkit is used for generating the chatbot.

Perez-Soler et al. (Perez-Soler et al., 2020) present the use of a chatbot as an interface for querying domain-specific models using natural language, suitable for non-technical users. The chatbot model is automatically generated based on the domain metamodel, and is implemented using Xatkit.

3 CHALLENGES OF CHATBOT DEVELOPMENT

An important challenge in chatbot development is that some of the vital components and services needed during design/runtime are proprietary, which creates dependency on service providers (Perez-Soler et al., 2021). Ensuring compatibility between platforms and different tool providers is also a critical challenge (Martínez-Gárate et al., 2023).

It seems that MDD can facilitate the process of describing various types of UIs as well as developing a rich UI (Planas et al., 2021). In this approach, different metamodels are defined for new platforms. However, metamodeling and language engineering are complex processes. In addition, we need techniques for analyzing model quality, as well as tools that reflect changes in requirements to the models (Martínez-Gárate et al., 2023).

Current MDD methodologies lack sufficient focus on requirements engineering (Martínez-Gárate et al., 2023). In addition, we need to employ design patterns and quality metrics for the proposed solutions to create chatbots based on the MDD approach (Martínez-Gárate et al., 2023). Chatbots needs to be maintained and synchronized with changing requirements; hence, methods are needed for enhancing maintainability, adaptability, and scalability (Martínez-Gárate et al., 2023). Table 1 shows some of the questions that need to be answered when developing chatbots (Perez-Soler et al., 2021; Matic et al., 2021; Martínez-Gárate et al., 2023).

Table 1: Questions to consider in chatbot development.

#	Question
1	How to find the most suitable tool for creating a chatbot based on its requirements?
2	How to design a chatbot independent of the development tool and platform?
3	How to analyse and evaluate a chatbot before implementation?
4	How to keep up with the rapid growth of the ecosystem and tools for developing chatbots?
5	How to support the migration process of chatbots to a new tool or platform?
6	How to integrate a chatbot with new NLU services provided by different vendors?
7	How to integrate chatbots with new communication channels provided by different vendors?
8	How to solve the coupling between a chatbot and a specific intent recognition service?
9	How to obtain training phrases for ML algorithms to recognize user intents?

4 PROPOSED CHATBOT DEVELOPMENT METHODOLOGY

In this section, we introduce our proposed model-driven methodology for developing chatbots. In this methodology, models are described at different abstraction levels. Model-to-model transformations are used for producing lower-level models from higher-level ones. Model-to-text transformations are ultimately used for generating the solution code based on the desired architecture and design patterns. The methodology consists of four phases: analysis, design, implementation and test. Figure 1 provides an overview of the methodology and the modeling performed at different levels of abstraction.

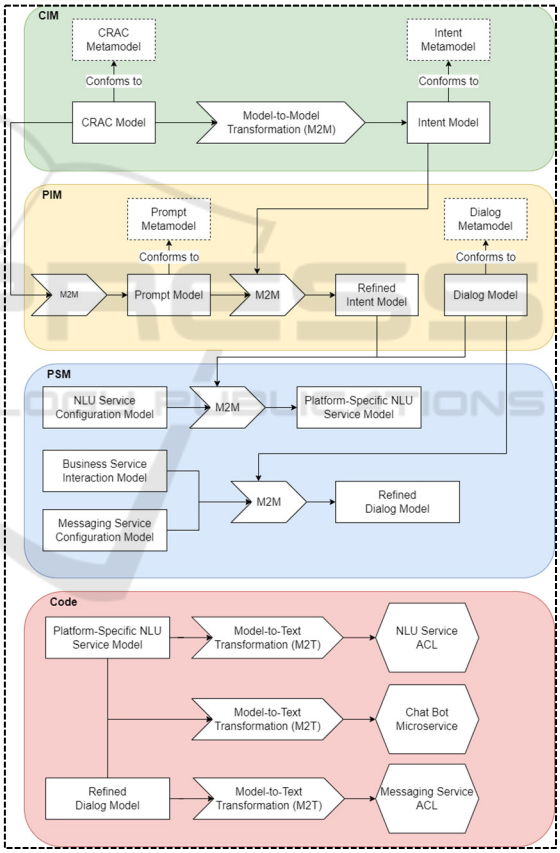

Figure 1: Proposed MDD methodology.

4.1 Analysis Phase

The aim of the analysis phase is to explore the problem domain and understand user goals by natural language conversation. Models are described at the highest level of abstraction, i.e., computation-independent model (CIM). The problem domain and

requirements are first analyzed, and a requirements model is produced. Then, based on this model and by using model transformations, user goals are extracted, and the intent model is generated.

We have proposed the CRAC method (Concept, Responsibilities, Asynchronous Collaboration) for analyzing the problem domain and extracting the requirements. In this method, domain concepts are first modeled as instances of the DomainConcept class. High-level system functions that change the system state from one valid state to another are modeled as Commands. The execution of these commands results in a change to the system state, which is expressed as an Event. Moreover, high-level functions that involve reading and querying data are modeled as Queries. Figure 2 shows the metamodel proposed for describing the problem domain using the CRAC method. We have also presented a metamodel (shown in Figure 3) for describing user intents that are expressed through natural language conversation with a chatbot. In the proposed method, intent models are automatically generated from the requirements model using model-to-model transformations. Table 2 shows how the elements of the CRAC metamodel correspond to the elements of the Intent metamodel.

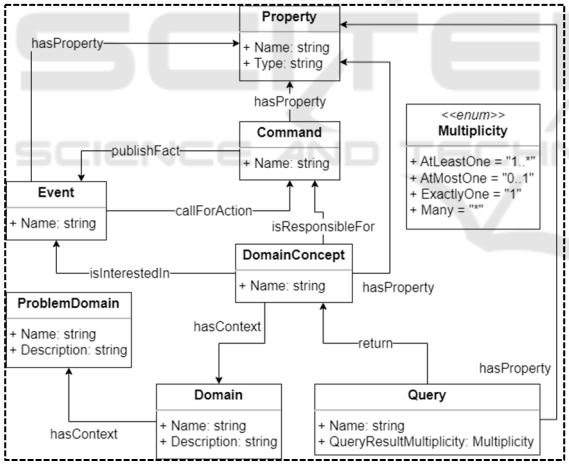

Figure 2: CRAC metamodel.

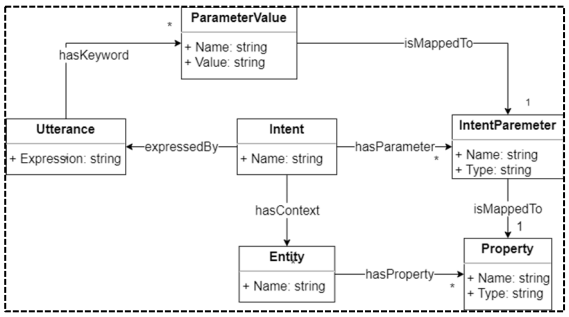

Figure 3: Intent metamodel.

Table 2: Mapping between CRAC and Intent elements.

CRAC metamodel	Intent metamodel
DomainConcept	Entity
Command	Intent
Query	Intent
DomainConcept → Property : hasProperty	Entity → Property : hasProperty
Command → Property : hasProperty	Intent → IntentParameter : hasParameter
Query → Property : hasProperty	Intent → IntentParameter : hasParameter
DomainConcept → Command : isResponsibleFor	Intent → Entity : hasContext
Query → DomainConcept : return	Intent → Entity : hasContext

4.2 Design Phase

The goal of this phase is to design the architecture of the chatbot and its interaction with the services required. The platform-independent models (PIM) produced in this phase are independent of the platform and service providers. The activities of this phase are: preparing an intelligent service for eliciting user intents, designing the conversation flow between the chatbot and the user, and providing an architecture to satisfy functional/non-functional requirements.

The first activity in this phase is detecting the users' intents and extracting the information necessary to fulfil the users' requests. The common approach is to use an AI model and an ML algorithm and train it by defining a set of phrases for each user intent, identifying the key parameters of that phrase, and mapping it to generic or custom entities. In general, the necessary tasks include: 1) finding phrases that are commonly used by users to convey each intent identified in the previous stage, 2) parameterizing these phrases and identifying entities and their properties (parameters) and mapping them to the parameters that are necessary to fulfil the user's request, 3) refining the user's intent model based on information obtained in the previous two steps, and 4) training the AI service to recognize user intent.

The knowledge required to perform tasks 1 and 2 is tacit in nature. Our proposed solution is to first model this tacit knowledge and then convert it to explicit knowledge using ChatGPT. For example, a sample of this prompt template can be raised: "*Give me 5 training phrases about this intent: {intent}*". This template can be refined as follows: "*Give me {number} training phrases about {intent}. I need {intent parameters} to fulfil the request*". ChatGPT can be asked to return its response in a specific format, which facilitates information extraction.

ChatGPT can also be used to specify the parameters desired in training phrases. Table 3 shows a prompt template, instructing the response format of ChatGPT. With the help of MDD techniques, these questions can be described as a prompt model by using a DSL, and then converted automatically to text by using model-to-text transformations. This text can be wrapped as an HTTP request and posted to the relevant API. We have proposed a metamodel to describe these questions, as shown in Figure 4. Figure 5 shows the details of this approach and how to generate questions and interact with ChatGPT.

The second activity in the design phase is identifying the two-way conversation flow between the chatbot and the user. For each intent in the intent model, we need to describe the conversation flow between the chatbot and the user and the necessary actions that should be taken to fulfil the user's request. We have proposed a metamodel for describing the dialogue flow, as shown in Figure 6.

Table 3: An example of tagged training phrases.

Template
Give me {number} training phrases about {intent}. I need {intent parameters} to fulfil the request in my app. {the format of response}
Prompt
Give me 5 training phrases about "BookHotel". I need StartDate, EndDate, HotelId to fulfil request in my app. Tag each training phrases with a parameter I mentioned. Put the tags inside square brackets.
Answer
1) I want to book a hotel from [StartDate] to [EndDate] at [HotelId]. 2) Can you help me reserve a room at [HotelId] from [StartDate] to [EndDate]? 3) I need to book a hotel stay from [StartDate] to [EndDate]. The hotel I want to stay at is [HotelId]. 4) Book me a room at [HotelId] for the dates between [StartDate] and [EndDate]. 5) I would like to reserve a room at [HotelId] from [StartDate] to [EndDate].

The third activity in the design phase is providing an architecture for the chatbot. Our proposed solution for intelligent chatbot conversations is based on the microservice architecture, as shown in Figure 7. Various patterns have been used in this architecture. One of these patterns is the Anti-Corruption Layer (ACL) pattern, which acts as a layer between two subsystems (Richardson, 2018). A chatbot requires AI services to understand natural language and detect user intent. It also needs integration with messaging channels. We have used ACLs as separate technical microservices. For each provider of NLU services or messaging channels, we use a microservice for isolating the external service provider.

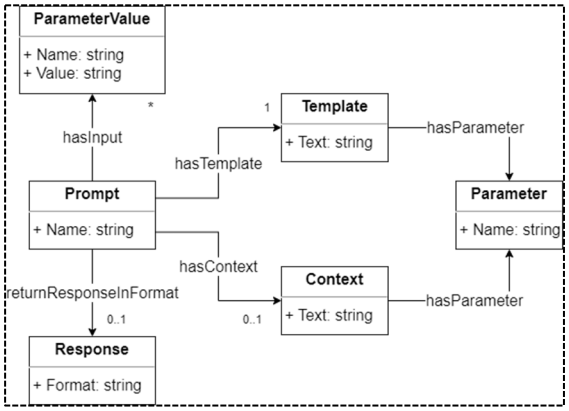

Figure 4: Prompt metamodel.

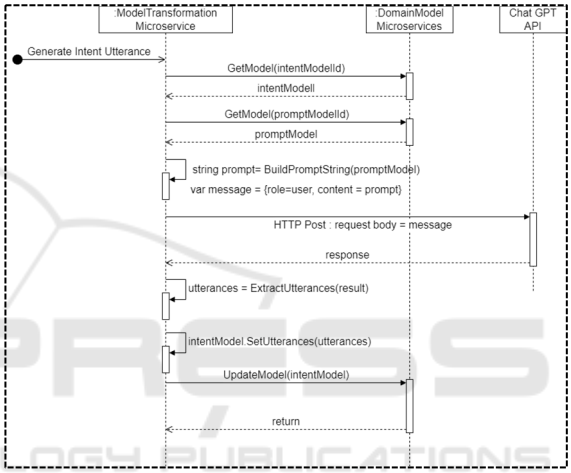

Figure 5: Using ChatGPT to extract utterances.

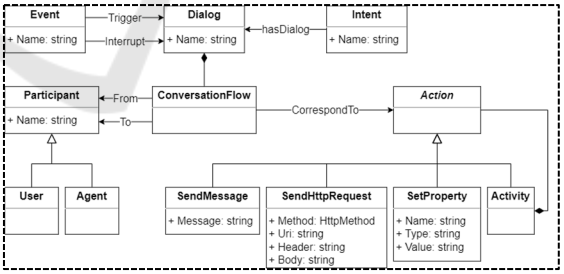

Figure 6: Dialog metamodel.

For each provider of NLU services or messaging channels, the development team can choose the appropriate technology stack and language for implementing the ACL microservices, which can be managed and deployed independently. Changes to the APIs provided by these services will not affect other parts of the system. Thus, migration and updates can be managed without affecting other parts. Since this microservice serves as a layer between the chatbot microservice and the NLU service or messaging

channel, it is possible to switch from one service provider to another without the chatbot microservice being aware of it. The API Gateway pattern, along with NLU and messaging services' ACL microservices, can help us achieve this goal. The API Gateway provides access to internal microservices. One of the main functions of the API Gateway is to direct requests to the relevant microservice, or to compose microservices (Richardson, 2018). Therefore, switching between NLU services and messaging platforms using the functions provided by the API Gateway is possible, and this change will not propagate to the chatbot microservice. Authorization can also be delegated to this layer. Implementing ACL as a microservice allows for horizontal scalability (scaling-out) based on workload, and the API Gateway can act as a Load Balancer.

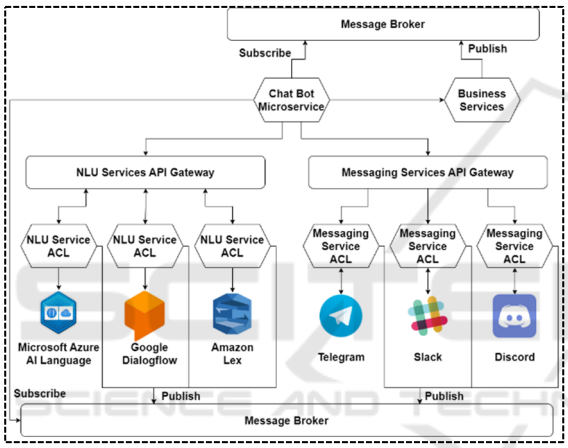

Figure 7: Proposed microservice architecture.

4.3 Implementation Phase

In this phase, platform-specific models (PSM) are created. By combining the information of the NLU Service Configuration Model at the PSM level and the Refined Intent Model and Dialog Model at the PIM level, the Platform-specific NLU Service Model is generated by using model-to-model transformation. Furthermore, the Dialog Model and the model of how the chatbot interacts with business services (Business Service Interaction Model), are combined with the configuration model of the messaging platform services to generate the Refined Dialog Model.

At the code level, model-to-text transformations are used to generate the NLU Service ACL microservice from the Platform-specific NLU Service Model. This microservice enhances the chatbot's ability to understand user intent through interaction with NLU services. Also, the Messaging Service ACL microservice is generated by using information from the Refined Dialog Model. By combining the Platform-specific NLU Service Model and the Refined Dialog Model, the Chatbot microservice is generated.

4.4 Test Phase

The aim of this phase is to perform verification and validation on the chatbot. We have identified the following set of criteria that can serve as a basis for verification and validation: functional effectiveness (interpretation accuracy, text synthesis performance, requested task execution, goal achievement, and linguistic accuracy), efficiency (resource utilization, cost effectiveness, and effective service allocation), robustness (unexpected input handling, graceful degradation, and user error protection), usability (user satisfaction and ease of use), and security (confidentiality, integrity, safety, privacy, authentication, and authorization) (Radziwill & Benton, 2017; Motger et al., 2021). There are various evaluation methods, some automated and some based on human evaluation. Automated methods are faster and less expensive, but they are not as accurate as human judgment (Deriu et al., 2021; Finch & Choi, 2020). Precision, Recall, BLEU, ROUGE, Response Diversity, and Context Coherence are examples of automated evaluation methods (Maroengsit et al., 2019; Finch & Choi, 2020). Human evaluation methods include Lab Experiments, In-field Experiments, Crowdsourcing, Expert Appraisal, In-app Feedback Questions, and Goal/Task Achievement (Maroengsit et al., 2019; Deriu et al., 2021; Motger et al., 2021). We have compiled a checklist, shown in Table 4, for human evaluation of chatbots (Finch & Choi, 2020; Liang & Li, 2021).

Table 4: Checklist for human evaluation.

#	Item
1	Response is understandable
2	Response is fluent
3	Response is natural
4	Response is grammatically correct
5	Response is relevant to the conversation
6	Response is relevant to the input
7	Response is logically appropriate
8	Response is consistent, free of semantic errors
9	Response is coherent with context
10	Response provides guidance on utterances
11	Response is informative
12	Response provides new information
13	Response is context-specific
14	Response is made from diverse words
15	Response is engaging to users
16	Response fulfils objectives of conversation
17	Response indicates understating of user intent

5 EVALUATION

We have used a criteria-based method to evaluate the proposed methodology. In this section, we will introduce the three categories of evaluation criteria; Evaluation results are shown at the end of the section.

Evaluation criteria belong to three categories: related to the generic software development lifecycle (SDLC-related), MDD-related, and chatbot-related. The generic SDLC criteria include: degree of coverage of the generic lifecycle and umbrella activities; degree of support for problem domain analysis, reusability, and adaptability; completeness of methodology definition; and the type of the methodology. MDD-related criteria include: degree of support for modeling at different levels of abstraction (CIM, PIM, PSM, and Code), model-to-model and model-to-text transformations, metadata management, verification and validation, automatic testing, traceability between models, and provision of tools (Asadi & Ramsin, 2008; Ramsin & Paige, 2010). Chatbot-related criteria include: chatbot input/output type (text/speech/voice); domain (open/closed); approach (pattern matching/rule-based/AI-based); knowledge data structures (structured/semi-structured/unstructured); degree of support for domain knowledge modeling, intent modeling, conversation flow modeling, training phrase elicitation, and training phrase annotation; degree of dependency on specific NLU service providers and messaging channels; and degree of support for architectural design, quality attributes, and conversational aspects (Singh & Beniwal, 2022; Motger et al., 2021; Liang & Li, 2021).

The results of evaluation of the proposed methodology based on the three categories of criteria are presented in Tables 5, 6 and 7, respectively.

Table 5: Evaluation based on generic SDLC criteria.

Criteria		Level
Coverage of Generic Lifecycle	Requirements Engineering	●
	Analysis	●
	Design	●
	Implementation	●
	Test	◐
	Deployment	○
	Maintenance	○
Coverage of Umbrella Activities	Project Management	○
	Quality Assurance	◐
	Risk Management	◐
Problem Domain Analysis		●
Reusability		●
Adaptability		○
Legend: Full support ● ; Partial support ◐ ; No Support ○		

Table 6: Evaluation based on MDD-related criteria.

Criteria	Level
CIM Creation	●
PIM Creation	●
PSM Creation	◐
CIM to CIM Model Transformation	●
CIM to PIM Model Transformation	●
PIM to PIM Model Transformation	●
PIM to PSM Model Transformation	◐
PSM to PSM Model Transformation	◐
PSM to Code Model Transformation	◐
Metadata Management	○
Verification & Validation	◐
Automatic Test	○
Traceability between Models	○
Tool Support	○

Table 7: Evaluation based on chatbot-related criteria.

Criteria		Level
Chatbot Input / Output		Text
Domain		Closed-domain
Approaches		AI-based
Knowledge Data Structures		(Semi)Structured
Domain Knowledge Modeling		●
User Intent Modeling		●
Conversation Flow Modeling		●
Training Phrase Elicitation		●
Training Phrase Annotation		●
NLU Service Providers		Vendor-independent
Communication Channels		Vendor-independent
Architectural Design		●
Quality Attributes	Scalability	●
	Flexibility	●
	Maintainability	●
	Interoperability	●
	Modifiability	●
	Usability	◐
	Availability	◐
	Performance	◐
	Security	◐
Conversational Aspects	Understanding	◐
	Answering	◐
	Navigation	◐
	Error handling	◐
	Relevance	◐
	Consistency	◐

6 CONCLUSIONS

The proposed model-driven methodology for chatbot development aims to address existing challenges by improving productivity, reusability, scalability, maintainability, and interoperability. By employing

an MDD approach, code can be automatically generated from models, increasing productivity. The platform-independent definition of chatbot-related artifacts also enhances their reusability. Additionally, the methodology introduces a new approach to obtaining data for training NLU services and utilizes microservice architecture and architectural design patterns to improve scalability, maintainability, and interoperability. We plan to further this research by providing tool support for the methodology and defining metamodels for common communication platforms and NLU services.

REFERENCES

Alam, O., Corley, J., Masson, C., & Syriani, E. (2018). Challenges for reuse in collaborative modeling environments. *MODELS Workshops*, 277–283.

Asadi, M., & Ramsin, R. (2008). MDA-Based Methodologies: An Analytical Survey. In *Model Driven Architecture – Foundations and Applications* (Vol. 5095, pp. 419–431). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-69100-6_30

Daniel, G., Cabot, J., Deruelle, L., & Derras, M. (2020). Xatkit: A Multimodal Low-Code Chatbot Development Framework. *IEEE Access*, *8*, 15332–15346. https://doi.org/10.1109/ACCESS.2020.2966919

Deriu, J., Rodrigo, A., Otegi, A., Echegoyen, G., Rosset, S., Agirre, E., & Cieliebak, M. (2021). Survey on evaluation methods for dialogue systems. *Artificial Intelligence Review*, *54*(1), 755–810. https://doi.org/10.1007/s10462-020-09866-x

Ed-douibi, H., Cánovas Izquierdo, J. L., Daniel, G., & Cabot, J. (2021). A Model-Based Chatbot Generation Approach to Converse with Open Data Sources. In *Web Engineering* (Vol. 12706, pp. 440–455). https://doi.org/10.1007/978-3-030-74296-6_33

Finch, S. E., & Choi, J. D. (2020). Towards Unified Dialogue System Evaluation: A Comprehensive Analysis of Current Evaluation Protocols. *Proceedings of the 21st Annual Meeting of the Special Interest Group on Discourse and Dialogue*, 236–245.

Liang, H., & Li, H. (2021). Towards Standard Criteria for human evaluation of Chatbots: A Survey. *ArXiv Preprint*. http://arxiv.org/abs/2105.11197

Mahmood, R., Joshi, A., Lele, A., & Pennington, J. (2020). Dynamic Natural Language User Interfaces Using Microservices. *HAI-GEN+ User2agent@ IUI*. https://ceur-ws.org/Vol-2848/user2agent-paper-1.pdf

Maroengsit, W., Piyakulpinyo, T., Phonyiam, K., Pongnumkul, S., Chaovalit, P., & Theeramunkong, T. (2019). A Survey on Evaluation Methods for Chatbots. *Proceedings of the 7th International Conference on Information and Education Technology*, 111–119. https://doi.org/10.1145/3323771.3323824

Martínez-Gárate, Á. A., Aguilar-Calderón, J. A., Tripp-Barba, C., & Zaldívar-Colado, A. (2023). Model-Driven Approaches for Conversational Agents Development: A Systematic Mapping Study. *IEEE Access*, *11*, 73088–73103. https://doi.org/10.1109/ACCESS.2023.3293849

Matic, R., Kabiljo, M., Zivkovic, M., & Cabarkapa, M. (2021). Extensible Chatbot Architecture Using Metamodels of Natural Language Understanding. *Electronics*, *10*(18), 2300. https://doi.org/10.3390/electronics10182300

Motger, Q., Franch, X., & Marco, J. (2021). Conversational Agents in Software Engineering: Survey, Taxonomy and Challenges. *ArXiv Preprint*. http://arxiv.org/abs/2106.10901

Perez-Soler, S., Guerra, E., & de Lara, J. (2019). Flexible Modelling using Conversational Agents. *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, 478–482. https://doi.org/10.1109/MODELS-C.2019.00076

Perez-Soler, S., Daniel, G., Cabot, J., Guerra, E., & de Lara, J. (2020). Towards Automating the Synthesis of Chatbots for Conversational Model Query. In *Enterprise, Business-Process and Information Systems Modeling* (pp. 257–265). https://doi.org/10.1007/978-3-030-49418-6_17

Perez-Soler, S., Guerra, E., & de Lara, J. (2021). Creating and Migrating Chatbots with Conga. *2021 IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, 37–40. https://doi.org/10.1109/ICSE-Companion52605.2021.00030

Planas, E., Daniel, G., Brambilla, M., & Cabot, J. (2021). Towards a model-driven approach for multiexperience AI-based user interfaces. *Software and Systems Modeling*, *20*(4), 997–1009. https://doi.org/10.1007/s10270-021-00904-y

Radziwill, N. M., & Benton, M. C. (2017). Evaluating quality of chatbots and intelligent conversational agents. *ArXiv Preprint*. http://arxiv.org/abs/1704.04579

Ramsin, R., & Paige, R. F. (2010). Iterative criteria-based approach to engineering the requirements of software development methodologies. *IET Software*, *4*(2), 91–104. https://doi.org/10.1049/iet-sen.2009.0032

Richardson, C. (2018). *Microservices patterns : with examples in Java*. Simon and Schuster.

Rodrigues da Silva, A. (2015). Model-driven engineering: A survey supported by the unified conceptual model. *Computer Languages, Systems & Structures*, *43*, 139–155. https://doi.org/https://doi.org/10.1016/j.cl.2015.06.001

Singh, S., & Beniwal, H. (2022). A survey on near-human conversational agents. *Journal of King Saud University - Computer and Information Sciences*, *34*(10), 8852–8866. https://doi.org/10.1016/j.jksuci.2021.10.013.