Asymob: a platform for measuring and clustering chatbots

Jose María López-Morales
Universidad Autónoma de Madrid
Madrid, Spain
JoseMaria.LopezM@uam.es

Pablo C. Cañizares
Universidad Autónoma de Madrid
Madrid, Spain
Pablo.Cerro@uam.es

Sara Pérez-Soler
Universidad Autónoma de Madrid
Madrid, Spain
Sara.PerezS@uam.es

Esther Guerra
Universidad Autónoma de Madrid
Madrid, Spain
Esther.Guerra@uam.es

Juan de Lara
Universidad Autónoma de Madrid
Madrid, Spain
Juan.deLara@uam.es

ABSTRACT

Chatbots have become a popular way to access all sorts of services via natural language. Many platforms and tools have been proposed for their construction, like Google's Dialogflow, Amazon's Lex or Rasa. However, most of them still miss integrated quality assurance methods like metrics. Moreover, there is currently a lack of mechanisms to compare and classify chatbots possibly developed with heterogeneous technologies.

To tackle these issues, we present Asymob, a web platform that enables the measurement of chatbots using a suite of 20 metrics. The tool features a repository supporting chatbots built with different technologies, like Dialogflow and Rasa. Asymob's metrics help in detecting quality issues and serve to compare chatbots across and within technologies. The tool also helps in classifying chatbots along conversation topics or design features by means of two clustering methods: based on the chatbot metrics or on the phrases expected and produced by the chatbot. A video showcasing the tool is available at https://www.youtube.com/watch?v=8lpETkILpv8.

CCS CONCEPTS

• **Human-centered computing** → **Natural language interfaces**; • **General and reference** → **Metrics**; • **Social and professional topics** → **Quality assurance**;

KEYWORDS

Chatbot design, metrics, quality assurance

ACM Reference Format:
Jose María López-Morales, Pablo C. Cañizares, Sara Pérez-Soler, Esther Guerra, and Juan de Lara. 2022. Asymob: a platform for measuring and clustering chatbots. In *44th International Conference on Software Engineering Companion (ICSE '22 Companion), May 21–29, 2022, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3510454.3516843

ICSE '22 Companion, May 21–29, 2022, Pittsburgh, PA, USA

ACM ISBN 978-1-4503-9223-5/22/05…$15.00
https://doi.org/10.1145/3510454.3516843

1 INTRODUCTION

Chatbots are increasingly used to access all sorts of services, including leisure (e.g., shopping, booking flights or hotels), customer services, professional support (e.g., banking) and information services (e.g., weather) [19]. Their success is due to their natural language conversational interface, which can be used through many channels such as social networks, web apps, or intelligent speakers.

The popularity of chatbots has triggered the emergence of a plethora of platforms, libraries and tools for their construction [14]. Some prominent examples are Google's Dialogflow[1], Amazon's Lex[2], IBM's Watson[3], Rasa[4] or Pandorabots[5], to name a few.

The quality of chatbots is critical for their success. In this respect, some researchers have proposed techniques for testing chatbots [3, 5] and guidelines for their design [11]. However, most tools lack static quality assurance mechanisms that can be used at design time to assess desired chatbot properties. Likewise, there is a lack of tools to compare, cluster and classify chatbots along design features or conversation topics. Such tools would enable a better understanding of the current chatbot landscape, the comparison of chatbots across implementation technologies (e.g., Dialogflow, Rasa) and provenance (e.g., open source repositories, proprietary platforms), and the extraction of valuable data for chatbot analysis.

In order to address these challenges, we present the web platform Asymob for chatbot measurement and clustering. The tool features a repository where chatbots developed using different technologies (currently Dialogflow and Rasa) can be uploaded. It offers a suite of 20 metrics that measure aspects of design size, complexity, and user experience. It also enables the clustering and comparison of chatbots based on these metrics; as well as on conversation topics extracted from the bot expected and issued phrases. The envisioned users of our tool are chatbot designers and developers.

In the rest of the paper, Section 2 introduces the basic notions of chatbots, Section 3 presents the Asymob platform and reports on a preliminary evaluation, Section 4 compares with related work, and Section 5 concludes with a summary and open research lines.

2 AN OVERVIEW OF CHATBOTS

Chatbots offer a conversational interface via natural language to software services. They are typically powered by natural language

[1] https://dialogflow.com/
[2] https://aws.amazon.com/en/lex/
[3] https://www.ibm.com/cloud/watson-assistant/
[4] https://rasa.com/
[5] https://home.pandorabots.com/

processing (NLP) technologies that provide good understanding capabilities on sets of predefined topics, called *intents*. Intents are expected conversation topics, which reflect the functionality of the chatbot. Frequently, intents are defined via training phrases that illustrate the different ways a user may approach the chatbot. For example, a chatbot for a pizzeria may have two main intents, one for ordering (expecting phrases like *"A small margherita, please"*) and another for obtaining information about the available pizza types (expecting utterances like *"What pizzas are available?"*).

Intents may define *parameters*, whose value is extracted from the user utterances. For example, when ordering a pizza, the user should specify the type of pizza (e.g., hawaiian) and the size (e.g., medium), via phrases like *"I'd like a medium hawaiian pizza"*. Parameters may be tagged as mandatory, in which case, the chatbot will request their value if absent from the user phrase. Parameters are typed by *entities*, which can be either user-defined (e.g., for pizza types) or pre-defined (e.g., for numbers or dates).

Conversations are defined by means of *flows* of expected intents and resulting bot *actions*. The latter normally involve an output phrase (which may also include parameters), but may also include other elements like images or widgets specific to the deployment channel (e.g., buttons in the Telegram social network). In addition, the chatbot may need to access an external service to manage the intent. For example, in the pizzeria, the chatbot needs to access an information system to store the order.

3 THE ASYMOB PLATFORM

Asymob is a web platform providing static chatbot quality assurance. Next, Section 3.1 describes its architecture, Sections 3.2–3.4 detail its functionality, and Section 3.5 reports on a preliminary evaluation.

3.1 Overview and architecture

Asymob[6] permits *uploading* chatbots of heterogeneous technologies, which then are *measured* using a suite of 20 metrics. Asymob provides *statistics* of the metrics across all chatbots in the repository. In addition, users can *query* the repository to search for chatbots within certain metric bounds and *compare* them against each other according to their metric values. The platform also allows *clustering* chatbots by metric values, or by the conversation topics as given by the words used in training phrases, bot responses and entities.

Fig. 1 shows the architecture of Asymob. Its functionality is offered via a web interface, which interacts with a service layer via a REST API. The *presentation layer* is implemented in HTML and JavaScript, and supports the interactive presentation of metrics and clusters using the libraries Plotly[7] and Cytoscape[8].

The *service layer* (the Asymob core) implements the functionality related to measuring and clustering chatbots. This core has an extensible design, which makes it easy to add new types of metrics, clustering criteria and chatbot technologies. To support the uniform handling of chatbots from heterogeneous technologies, the core relies on a neutral chatbot design notation called Conga [12]. This way, our platform enables the contribution of importers from specific chatbot implementation platforms into

[6]http://miso.ii.uam.es/asymobService
[7]https://plotly.com/
[8]https://cytoscape.org/

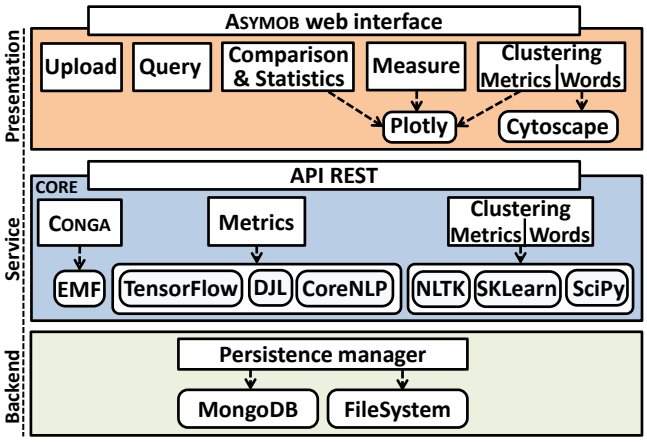

Figure 1: Architecture of Asymob.

Conga, and the measurement and clustering are applied to Conga models. Section 3.2 will provide more details about Conga and the available importers. Then, Section 3.3 will present the current list of supported metrics, built upon established technologies such as TensorFlow[9], CoreNLP[10] and the Deep Java Library (DJL)[11]. Next, Section 3.4 will focus on the chatbot clustering functionality, developed using Python libraries like NLTK[12], SKLearn[13] and SciPy[14].

An additional *backend layer* provides persistence. This stores the uploaded chatbots in the filesystem of the machine where the Asymob core is deployed, and uses mongoDB[15] for storing the data produced in the service layer (i.e., the metric values and the information required for conducting clustering).

3.2 Handling heterogeneous chatbots

Asymob provides static mechanisms to assess chatbot quality. Since there are many chatbot development tools, Asymob implements those mechanisms over a neutral, technology-agnostic chatbot design notation called Conga, and provides importers from different technologies into Conga. This permits reusing the functionality of Asymob with chatbots of heterogeneous technologies.

Conga [12] is designed based on an analysis of 15 popular chatbot tools, so its primitives can be mapped from/to all of them. Conga supports the concepts explained in Section 2 (intents, parameters, entities, conversation flows, bot actions). Intents can be defined via training phrases. User-defined entities can be described as a list of words with synonyms, via a regular expression, or providing a set of strings and other entities. Possible chatbot actions include sending text, images, HTTP requests to external services, or presenting widgets like buttons.

Currently, there are importers from Rasa and Dialogflow chatbots into Conga. Rasa is a framework to develop chatbots using

[9]https://www.tensorflow.org/
[10]https://stanfordnlp.github.io/CoreNLP/
[11]https://djl.ai/
[12]https://www.nltk.org/
[13]https://scikit-learn.org/
[14]https://scipy.org/
[15]https://www.mongodb.com/

Table 1: Metrics for chatbot designs.

Metric	Description	Type
Global metrics		
INT	# intents	design size
ENT	# user-defined entities	vocabulary size
FLOW	# conversation entry points	conversation diversity
PATH	# different conversation flow paths	conversation complexity
CNF	# confusing phrases	bot understanding
SNT	# positive, neutral, negative output phrases	user experience
Intent metrics		
TPI	# training phrases per intent	topic complexity
WPTP	# words per training phrase	topic complexity
VPTP	# verbs per training phrase	topic complexity
PPTP	# parameters per training phrase	topic complexity
WPOP	# words per output phrase	readability
VPOP	# verbs per output phrase	readability
CPOP	# characters per output phrase	readability
READ	reading time of the output phrases	readability
Entity metrics		
LPE	# literals per entity	vocabulary complexity
SPL	# synonyms per literal	vocabulary complexity
WL	word length	readability
Flow metrics		
FACT	# actions per flow	bot response complexity
FPATH	# conversation flow paths	conversation complexity
CL	conversation length	conversation complexity

Python, markdown and YAML. Dialogflow is a lowcode development platform to create chatbots using a graphical web interface, and the chatbots can be exported as JSON files.

Overall, users of Asymob can register on the platform or use a generic user. When uploading a chatbot to the platform, the user must specify the chatbot implementation technology (Dialogflow, Rasa or Conga), its visibility (private, so that only the owner can see the chatbot, or public, to allow other users see it), and its version (to enable version control for the chatbot). Then, the proper importer is automatically applied to the chatbot, and both the original chatbot and the resulting Conga model are stored.

3.3 Measuring chatbots

Asymob includes a metrics engine to analyse static characteristics related to the correct design of chatbots. This provides a suite of 20 metrics, which we proposed in [6], covering both *global* design aspects and specific features concerning the design of *intents*, *entities* and conversation *flows*. The metrics are summarized in Table 1 and we explain them next.

Global metrics capture global properties of the chatbot. Specifically, Asymob measures the number of intents (INT), user-defined entities (ENT), conversation entry points (FLOW), conversation flow paths (PATH), confusing phrases (CNF), and output phrases with positive, neutral or negative sentiment (SNT). Confusing phrases refer to similar training phrases (i.e., with small semantic distance) defined by different intents. They are problematic, since a chatbot may confuse them and end up identifying a wrong intent. Additionally, a chatbot that mostly outputs phrases with negative sentiment may impact negatively the user experience. Overall, global metrics are useful to assess the chatbot design size (INT), the chatbot vocabulary size (ENT), the conversation diversity and complexity (FLOW and PATH), and to report potential problems in bot understanding (CNF) and user experience (SNT).

Intent metrics measure quality and complexity aspects of intents, namely, the number of training phrases per intent (TPI), words/verbs/parameters per training phrase (WPTP/VPTP/PPTP), words/verbs/characters per output phrase (WPOP/VPOP/CPOP) and average reading time of the output phrases (READ). Overall, these metrics quantify the complexity and readability of phrases. Large phrases are difficult to understand, are problematic in social networks with constrained message length (like Twitter), and may require scrolling in mobile devices with small screens. For example, high CPOP and READ values entail long reading times, which may make users not to read fully the bot answers. This is even more problematic for voice-based chatbots, since speaking takes longer than reading [11].

Entity metrics analyse the user-defined entities, which represent domain concepts. They are useful to obtain a measurement of their complexity and readability. Entity metrics include the number of literals per entity (LPE), the synonyms per literal (SPL) and the length of words (WL). These are indicators of the complexity of the concepts and the width of the vocabulary of the chatbot.

Flow metrics are concerned with the complexity of the conversation flows and the sophistication of the bot responses. They comprise the number of actions per flow (FACT), the number of conversation paths (FPATH) and the conversation length (CL).

When a chatbot is uploaded, Asymob computes its metrics and displays their value in a table and also in interactive graphs that compare these values with statistics of the chatbots in the repository. Fig. 2 shows the graph for metric INT. The left bar displays statistics of the chatbot repository, and the bar to the right displays the metric value for the uploaded chatbot. We observe that the new chatbot can be considered large, since it has 25 intents, while the average number of intents of the chatbots is around 10 (with a median of 6). The computed metrics are persisted to speed up the generation of statistics when new chatbots are uploaded, and to facilitate the functionalities we explain next.

Figure 2: Displaying the value of metric INT.

First, Asymob offers statistics of the metrics of all chatbots in the repository (average, minimum, maximum, median and 1st and 3rd

quartiles). They are displayed as a table, as a graph, and side-by-side with the metric values of a specific chatbot, as Fig. 2 shows.

Additionally, Asymob permits comparing a collection of chatbots based on a set of metrics selected by the user. Fig. 3 illustrates this functionality. The x-axis displays the selected metrics (ENT, INT and FLOW in the figure), and the y-axis shows their value for the selected chatbots from the repository (4 bots in this case). We can see that mysteryAnimal stands out in the three metrics, meaning that it has more vocabulary (entities), conversation alternatives (intents) and conversation flows. This comparison can also be performed for several versions of the same chatbot (if different versions were uploaded into the repository) to reason about the evolution between chatbot versions in terms of metrics.

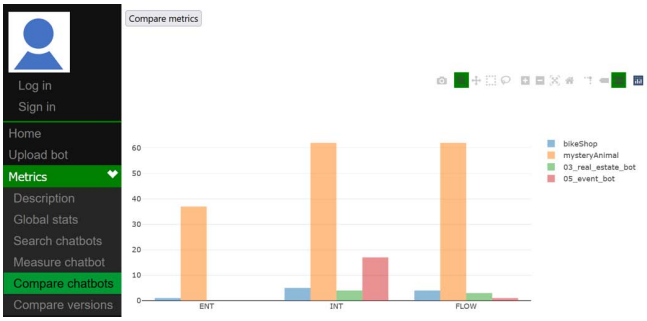

Figure 3: Metric-based comparison of chatbots.

The platform also includes a metric-based chatbot search facility. This permits users to specify the lower and upper limits for the value of some metrics of interest, and Asymob displays the chatbots in the repository with metric values within these boundaries. This is useful to obtain sets of chatbots with certain characteristics. For example, we might be interested in simple chatbots with few intents and no defined entities, or complex chatbots with many intents and complex conversation flows.

3.4 Clustering chatbots

Asymob supports the automated classification of chatbots based on two disjoint criteria: metric values, or the chatbot vocabulary.

Metrics-based clustering is useful to identify groups of chatbots with (dis)similar design features. For this purpose, the user can select one or more metrics, and the chatbots become classified based on the metric(s) values. For example, clustering by metric INT (i.e., number of intents) would create groups of chatbots with similar size complexity, whereas if the user performs the clustering using metrics FLOW and PATH, then the chatbots would be grouped according to the complexity of their conversations.

Technically, the platform implements the K-means algorithm for clustering the chatbots based on the value of the selected metrics. The user can also select the number of clusters to create (i.e., the k-value), or otherwise, it is automatically computed using the silhouette coefficient [16], as supported by SKLearn.

Asymob visualizes the resulting clusters in a table and graphically, as Fig. 4 shows. The graph can display two or three dimensions, so if the user selects more than three metrics, then the platform reduces the number of dimensions using the principal component analysis (PCA). The graphic represents each chatbot as a dot, and uses a different colour for each cluster of chatbots. In Fig. 4, there are two clusters of 3 and 26 chatbots.

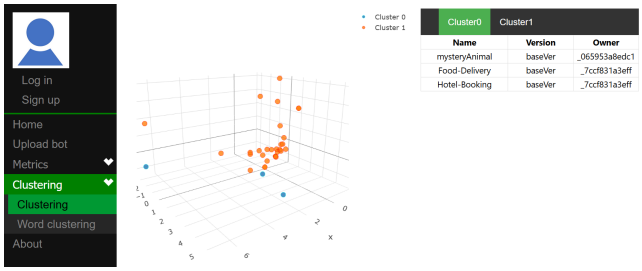

Figure 4: Metrics-based clustering.

Vocabulary-based clustering classifies chatbots by their vocabulary, which is useful to identify chatbots targeting analogous topics. For example, chatbots for booking flights are likely to be in the same cluster, since their vocabulary tends to be similar. We foresee this clustering to be useful as a way to search for chatbots by similarity to a given one, or by existing topics (represented by clusters). We also envision using this clustering method as a way to present and organize a large set of chatbots within a repository.

For this kind of clustering, Asymob stores all the relevant words that appear in the training phrases, chatbot responses and user-defined entities of each chatbot, along with their frequency of occurrence. Stop words such as prepositions, articles and conjunctions are discarded. Then, the similarity of two bots is given by the cosine-similarity of their *bag-of-words* vectors [10]. Note that each chatbot has to be compared with all the other ones, which becomes time-expensive as the repository grows. To reduce this time, Asymob calculates this similarity as a backend process when a chatbot is uploaded, and caches the result in a database.

In the front-end, users can select a set of chatbots and a similarity threshold for the agglomerative clustering algorithm. The results are shown in a table and an interactive hierarchical graph. The first graph layer has a node per cluster, and clicking on a node shows the chatbots it contains. Fig. 5 shows the chatbots within a cluster. The width of the edges conveys the similarity of two chatbots. Clicking on a chatbot displays its metrics on the right.

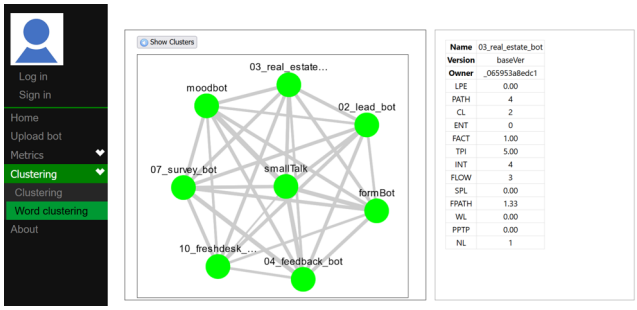

Figure 5: Vocabulary-based clustering.

3.5 Preliminary evaluation

We have evaluated the cost of uploading a chatbot, which implies its measurement and extracting its bag-of-words for clustering. The latter requires updating a global vocabulary index when the chatbot introduces new words. We found this to require constant time, in the order of 100ms. At this point, a backend process compares and caches the cosine similarity between the bag-of-words vector of the uploaded chatbot and all the rest, which we found to grow linear with the number of chatbots (around 99ms per bot). Being a backend process, it does not affect responsiveness, but we are currently working on its parallelization. In the future, we plan to perform more detailed scalability experiments to detect possible bottlenecks in our architecture and optimize where needed.

4 RELATED WORK

Most approaches to assess chatbot quality rely on testing. Some development platforms (e.g., Dialogflow, Lex, Watson) integrate a web chat console to test chatbots manually. There are also dedicated testing tools like Botium [3] and OggyBug [7] which automate the testing of chatbots built with different technologies. Still, developers need to define concrete test conversation cases. To alleviate this burden, Bottester [20] simulates the user interactions, and other works generate challenging test user utterances automatically [4, 5, 17]. Compared to these works, Asymob provides complementary assessment mechanisms to testing in the form of metrics that are collected statically (i.e., without deploying the chatbot), with reduced effort compared to testing, and which can reveal defects on several quality aspects of the chatbot design.

Additionally, some development platforms (e.g., Dialogflow, Watson, Bot Framework) provide chatbot analytics. This information is collected dynamically when the chatbot is in production, while Asymob targets the design time.

Another popular way to evaluate chatbots is by means of user studies [9, 18]. These typically evaluate user satisfaction and chatbot performance, and require the recruitment and participation of users [15]. Asymob complements these studies with chatbot information that can be gathered automatically and inexpensively.

The support of static means for quality assessment – like those in Asymob – is less frequent. Next, we discuss some exceptions. Dialogflow performs some chatbot validations (e.g., detecting intents with similar training phrases) categorized by severity. Almansor and Hussain [1] use fuzzy logic to detect inappropriate responses based on the sentiment and length of utterances, and Gao et al. [8] use machine learning to predict the popularity of chatbots based on static metrics (e.g., number of intents, conversation flow length). These two works use metrics supported by Asymob, showing that our tool could enable the use of artificial intelligence for prediction.

Finally, our tool takes inspiration from services available in repositories of other artefacts, like meta-models (e.g., MDEForge [2]). However, to the best of our knowledge, Asymob is the first proposal of a chatbot repository featuring metrics and clustering.

5 CONCLUSIONS AND FUTURE WORK

This paper has presented Asymob, the first platform enabling measuring and clustering chatbots. The tool fills a gap on current practice, which is providing automatic means for assessing the quality of chatbots prior to their deployment and dynamic testing. The tool comprises a repository of chatbots, static metrics that can be homogeneously evaluated on heterogeneous technologies, and chatbot clustering facilities based on chatbot metrics and vocabulary.

We are currently building converters from other technologies (e.g., Pandorabots, Lex) into Conga. We are also improving the tool with visualization mechanisms able to capture a large amount of informations, e.g., heatmaps and dendograms for clusters. In the future, we plan to use Asymob to evaluate open source chatbots to get a panorama of their features and derive metric thresholds. We also plan to exploit our clustering techniques to provide search facilities over chatbot repositories. Finally, we would also like to integrate Asymob's services within chatbot development tools like the Conga web IDE [13].

ACKNOWLEDGMENT

This work has been funded by the Spanish Ministry of Science (RTI2018-095255-B-I00, project "MASSIVE") and the R&D programme of Madrid (P2018/TCS-4314, project "FORTE").

REFERENCES

[1] Ebtesam Hussain Almansor and Farookh Khadeer Hussain. 2021. Fuzzy prediction model to measure chatbot quality of service. In *FUZZ-IEEE*. IEEE, 1–4.
[2] Francesco Basciani, Juri Di Rocco, Davide Di Ruscio, Ludovico Iovino, and Alfonso Pierantonio. 2016. Automated clustering of metamodel repositories. In *CAiSE (LNCS)*, Vol. 9694. 342–358.
[3] Botium. [n. d.]. `https://www.botium.ai/`. last access in 2021.
[4] Josip Bozic and Franz Wotawa. 2019. Testing chatbots using metamorphic relations. In *ICTSS (LNCS)*, Vol. 11812. Springer, 41–55.
[5] Sergio Bravo-Santos, Esther Guerra, and Juan de Lara. 2020. Testing chatbots with Charm. In *QUATIC (CCIS)*, Vol. 1266. Springer, 426–438.
[6] Pablo C. Cañizares, Sara Pérez-Soler, Esther Guerra, and Juan de Lara. 2022. Automating the measurement of heterogeneous chatbot designs. In *SAC*. ACM, 1–8.
[7] Márcio Braga dos Santos, Ana Paula Carvalho Cavalcanti Furtado, Sidney C. Nogueira, and Diogo Dantas Moreira. 2020. OggyBug: A test automation tool in chatbots. In *SAST*. ACM, 79–87.
[8] Mingkun Gao, Xiaotong Liu, Anbang Xu, and Rama Akkiraju. 2021. Chatbot or Chat-Blocker: Predicting chatbot popularity before deployment. In *DIS*. ACM, 1458–1469.
[9] Jiepu Jiang and Naman Ahuja. 2020. Response quality in human-chatbot collaborative systems. In *SIGIR*. ACM, 1545–1548.
[10] Michael McTear, Zoraida Callejas, and David Griol. 2016. *The Conversational Interface. Talking to Smart Devices.* Springer.
[11] Robert J. Moore and Raphael Arar. 2019. *Conversational UX Design: A Practitioner's Guide to the Natural Conversation Framework.* ACM, New York, NY, USA.
[12] Sara Pérez-Soler, Esther Guerra, and Juan de Lara. 2020. Model-driven chatbot development. In *ER (LNCS)*, Vol. 12400. Springer, 207–222.
[13] Sara Pérez-Soler, Esther Guerra, and Juan de Lara. 2021. Creating and migrating chatbots with Conga. In *ICSE Companion*. IEEE, 37–40.
[14] Sara Pérez-Soler, Sandra Juarez-Puerta, Esther Guerra, and Juan de Lara. 2021. Choosing a chatbot development tool. *IEEE Softw.* 38, 4 (2021), 94–103.
[15] Ranci Ren, John W. Castro, Silvia Teresita Acuña, and Juan de Lara. 2019. Evaluation techniques for chatbot usability: A systematic mapping study. *Int. J. Softw. Eng. Knowl. Eng.* 29, 11&12 (2019), 1673–1702.
[16] Peter J. Rousseeuw. 1987. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.* 20 (1987), 53–65.
[17] Elayne Ruane, Théo Faure, Ross Smith, Dan Bean, Julie Carson-Berndsen, and Anthony Ventresque. 2018. BoTest: A framework to test the quality of conversational agents using divergent input examples. In *IUI Companion*. ACM, 64:1–64:2.
[18] Bayan Abu Shawar and Eric Atwell. 2007. Different measurements metrics to evaluate a chatbot system. In *NAACL-HLT-Dialog*. ACM, 89–96.
[19] Amir Shevat. 2017. *Designing bots: Creating conversational experiences.* O'Reilly.
[20] Marisa Vasconcelos, Heloisa Candello, Claudio S. Pinhanez, and Thiago dos Santos. 2017. Bottester: Testing conversational systems with simulated users. In *IHC*. ACM, 73:1–73:4.