A Framework for Understanding Chatbots and their Future*

Elahe Paikari
University of California, Irvine
Irvine, CA 92705 U.S.A.
epaikari@uci.edu

André van der Hoek
University of California, Irvine
Irvine, CA 92705 U.S.A.
andre@ics.uci.edu

ABSTRACT

Chatbots have rapidly become a mainstay in software development. A range of chatbots contribute regularly to the creation of actual production software. It is somewhat difficult, however, to precisely delineate hype from reality. Questions arise as to what distinguishes a chatbot from an ordinary software tool, what might be desirable properties of chatbots, and where their future may lie. This position paper introduces a starting framework through which we examine the current state of chatbots and identify directions for future work.

CCS CONCEPTS

• **Software and its engineering** → software notations and tools
• **Human Computer Interaction** → natural language interfaces

KEYWORDS

Software tools, chatbots, comparison framework

ACM Reference format:

Elahe Paikari and André van der Hoek. 2018. A Framework for Understanding Chatbots and their Future. In *Proceedings of 11th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE'18)*. Gothenburg, Sweden, 4 pages.
https://doi.org/10.1145/3195836.3195859

1 INTRODUCTION

Consider the following hypothetical conversation taking place between a software developer (D) and an electronic whiteboard (B) at which they are working to design some software:

D: Board! I need to design something.
B: Hey D, good to see you back. What are you up to this time?
D: I need to re-design the internals of our client-server system.
B: That's a serious one. Well, let's get going on this. What would you like to look at first? Shall I bring up the architecture?
D: Nah, I can dream that one. I would like to start with a blank slate.
B: Alright. Here's a blank page for you. But let me remind you of some of the past objectives for the architecture. [pops up a list] Are all of those still applicable?
D: Most of them, and especially #3 and #4. But add the following two: scalability and replicability.
B: Done. Here, let me toss you a few possible architectures. Any of which you like?
D: Let's start with the third one, but can you add a couple of components that handle logging like in architecture #1?
B: Sure enough. I highlighted in red, btw, one interface that does not connect as a result. We can address it later.
D: Thank you. So, let me think. Can we zoom in on the server and design what's inside?
B: ...

It has long been a dream of the computing field to develop human-like capabilities of speech and interaction such as illustrated in this scenario [14]. Today, we are inching closer with the emergence of chatbots as a novel form of engagement of software tools in the development process [6]. Riding the wave of speech recognition and artificial intelligence, chatbots of various capabilities are emerging and being relied upon [1].

Chatbots have been identified as having significant promise [9], but at the same time most chatbots perform mundane tasks in rather predictable fashion [4]. While this necessarily represents a starting point, it does raise questions as to what distinguishes a chatbot from a regular software tool, what might be desirable properties of chatbots, and where their future may lie. To help shed light on these and other kinds of questions, this position paper contributes a starting framework to characterize the current state of chatbots and identify directions for future work.

In the remainder of this paper, we first briefly review the nature of chatbots as compared to bots and software tools more generally in Section 2. We introduce our framework in Section 3 and examine a representative set of software chatbots in Section 4.

2 CHATBOTS

Software tools have been created for nearly as long as developers have been writing code. Today, myriad tools exist, ranging from large, complex and powerful to small, simple yet still useful. While the first tools were usually standalone, many current tools appear as plug-ins into software development environments or are entirely web-based.

CHASE'18, May 27, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5725-8/18/05…$15.00
https://doi.org/10.1145/3195836.3195859

Bots can be considered a particular type of software tool, distinguishing themselves as tools that automate repetitive tasks. Googlebot[1] and autolt[2] are just two examples, in their cases performing the task of web crawling and scripting the simulated execution of Windows GUIs, respectively. Bots, because of their convenience, are becoming a major part of development practice [7].

Chatbots are a particular type of bot that engage with their users by textual or voice dialogue. For instance, some chatbots monitor what users type on IM channels and trigger their actions when certain commands or patterns are present (e.g., [2]). Other bots might make contributions to an IM channel based on some external events of which they apprise the users (e.g., AlertBot[7]). More complex bots are envisioned to engage in meaningful conversation with their users, relying on natural language processing and Artificial Intelligence (AI) to become more human-like and intelligent in their engagement [5].

An important aspect of chatbots is that the ultimate vision is for them to mimic written or spoken human language for the purposes of simulating a conversation or interaction with a real person [12]. A variety of chatbot architectures and technologies have arisen recently that begin to support the development of such chatbots (e.g., J.A.R.V.I.S.[3], Hubot[4], Pandorabots[5], Wit.ai[6]) and aim to provide a marketplace for them [8].

3 CHATBOT COMPARISON FRAMEWORK

We developed our framework by studying the broader portfolio of current chatbots such as the ones discussed in Section 2, examining the still somewhat limited literature on chatbots [4], and identifying key dimensions along which they differed. This process resulted in six complementary dimensions (type, direction, guidance, predictability, interaction style, and communication channel) that we introduce one-by-one.

3.1 Type

Chatbots can serve drastically different purposes. For instance, the AlertBot[7] chatbot is relatively straightforward in offering automated notifications of certain relevant events, while J.A.R.V.I.S.[3] is a framework that lets developers create AI powered chatbots that learn fast and can be trained to perform different engineering tasks without the need to explicitly program it to do so. Inspired by [7], who characterize chatbots serving a broad range of purposes compared to our focus on software development chatbots only, we identify three types of chatbots:

- *Information.* Information chatbots assist developers by finding information that may be relevant to the task at hand. For example, they may find and communicate to the developer the set of bugs that are relevant to the code and currently being worked on by accessing Jira or Slack. Information chatbots may leverage any resource, whether a global search engine, a local repository, or another source, to obtain the information needed.
- *Collaboration.* Collaboration chatbots aim to enrich the interactions among developers so that they can work together more effectively. For example, a collaboration chatbot may notify a developer of a conversation that some other developers started of which the outcome is pertinent to the module that the developer being notified is working on. A more advanced collaboration chatbot could engage in conversation on behalf of a developer, essentially representing them and their viewpoints.
- *Automation.* Automation chatbots work with developers to accomplish tasks which affect some type of change in one or more artifacts that they are working on. For example, the chatbot may assist the developer in producing documentation or performing a certain type of refactoring by 'talking them through it'.

3.2 Direction

With different chatbots having different purposes, the kinds of conversations they have differ. Many chatbots, instead of engaging in a full conversation, operate uni-directionally.

- *Input.* Input chatbots typically monitor a communication channel for specific words or phrases that a developer may enter as the trigger, and invoke different actions depending on the specific words or phrases. These chatbots do their work silently.
- *Output.* Output chatbots operate by inserting content to a communication without being needing input that tells it what to do; they are often externally triggered and report on important events elsewhere.
- *Bi-directional*. Bi-directional chatbots both take input and produce output on the communication channel. This may be simple trigger-response interactions, but also approach actual conversations.

3.3 Guidance

Especially when chatbots become more complex, whether they can operate independently is important to their utility and usability.

- *Human-mediated.* Human-mediate chatbots cannot accomplish their tasks alone, typically requiring the developer to provide it with pertinent information or to direct what next steps it should be taking. These kinds of chatbots can still be minimally invasive, if they require little guidance or direction. If the developer is constantly interrupted by its behavior, however, they can become too intrusive and may be deactivated as a result.
- *Autonomous.* Autonomous chatbots operate on their own to accomplish their tasks and either already have all the information they need or collect that information on their own. These kinds of chatbots have the advantage of being able to make valuable contributions independently, but run the risk of 'not doing the right thing', which may actually be worse than interrupting the developer too often.

3.4 Predictability

All chatbots are ultimately programmed by some set of developers, who give each its own unique functionality. The resulting range of functionalities, however, can be vast. A useful distinction among

[1] https://support.google.com/webmasters/answer/182072?hl=en
[2] https://www.autoitscript.com
[3] http://ironman.wikia.com/wiki/J.A.R.V.I.S.
[4] https://hubot.github.com
[5] https://www.pandorabots.com
[6] https://wit.ai
[7] https://www.alertbot.com

all of them can be made when focusing on the nature of what they accomplish: is their functionality deterministic or learned?

- *Deterministic.* Deterministic chatbots are created to exhibit exactly the same behavior each time they encounter the same situation. These chatbots often include a decision tree to decide which action to take given an input or other stimulus. The vast majority of chatbots are of this ilk today, operating in a directed manner that, while perhaps boring and repetitive over time, gets the task done predictably.
- *Evolving.* Evolving chatbots include a learning component that allows them to take past interactions and outcomes and adjust their behavior accordingly, increasing their capabilities to perform tasks in ways that the original programmer may not have planned. However, this is not without its drawbacks: if care is not taken, a chatbot may learn counterproductive functionality. For instance, Facebook designed a chatbot able to engage in multi-issue bargaining in natural language. However, they decided to abandon the project, because the language the chatbots used evolved into being not understandable by humans [11].

3.5 Interaction style

Although we are still far from having human-like chatbots, they are increasingly expected to interact in a human-like manner, partly to integrate seamlessly into the development effort and partly to offer personalized support. As much as user interaction is important for any software application, the 'feel' that a chatbot has can play a major role in its eventual effectiveness. Chatbots that are more advanced in vocabulary and interaction, and perhaps exhibit personality to a degree, then, may actually help developers to have an engaging conversation, which affects their experience and can influence perceptions of the chatbot's roles and capabilities [13].

Complementing whether their task accomplishment is predictable, then, is the question of the interaction style through which chatbots interact with developers.

- *Dull.* Dull chatbots interact with developers using simple words and often simply repeat the same phrases. Every time, they welcome developers with the same greeting or onboarding instructions, or interject with the same phrases.
- *Alternate vocabulary.* Alternate vocabulary chatbots improve over dull chatbots by including bags of alternative phrases that they draw on to vary responses that mean the same thing, but appear to at least change in form to the developer.
- *Relationship builder*. Relationship builder chatbots understand that it is important to build rapport with the developers. Their vocabulary varies from being casual to technical; they may plan conversational flows; and may at times spontaneously reach out with some information or a joke to offer a more accessible and fun experience.
- *Human-like*. Human-like chatbots learn to interact from the history of conversations in which they engage, and thus can adopt subtle patterns that offer more timely, connected, and meaningful interactions.

Note that care must be taken so the chatbot exhibits appropriate behavior (avoiding, for instance, the recent experience of Microsoft AI chatbot Tay, which became offensive and discriminatory [10]).

Table 1: Comparison framework applied to a selection of chatbots related to software development.

		Jarvis-bot-memory[8]	Hubot-ghe[9]	Codebots-plugins[10]	Obie[11]	Sourcing bot[12]	Statsbot[13]	Jira Cloud[14]	Zapier[15]	UptimeRobot[16]	Bitbucket Slack app[17]
Type	Information		•		•	•	•	•	•		•
	Collaboration										
	Automation	•		•						•	
Direction	Input	•	•	•		•					
	Output										•
	Both				•		•	•	•	•	
Guidance	Human-mediated		•	•		•					
	Autonomous	•			•		•	•	•	•	•
Predictability	Deterministic	•	•	•	•	•	•	•	•	•	•
	Evolving										
Interaction style	Dull	•	•	•	•	•	•	•	•	•	•
	Alternate vocabulary										
	Relationship builder										
	Human-like										
Communication channel	Text	•	•	•	•	•	•	•	•	•	•
	Voice										
	Both										

[8] https://www.npmjs.com/package/jarvis-bot

[9] https://github.com/hubot-scripts/hubot-ghe

[10] https://codebots.com/

[11] http://obie.ai

[12] https://botlist.co/bots/sourcing-bot

[13] https://statsbot.co/

[14] https://slack.atlassian.com/

[15] https://zapier.com/apps/slack/integrations

[16] https://uptimerobot.com/

[17] https://marketplace.atlassian.com/plugins/com.slack.bitbucket

3.6 Communication channel

While nearly all chatbots today communicate using text messages, typically integrating themselves into a messaging platform (e.g., Slack, Hipchat, Microsoft Teams), the emergence of voice communication platforms (e.g., Alexa, Siri) offers opportunities.

- *Text*. Text chatbots have the advantage that their textual input and output can easily be interpreted, preserved, and recast – all at a time convenient to the developer. At the same time, requiring textual input can be demanding on the developer.
- *Voice*. Voice chatbots have a more intimate feel, taking speech input and providing spoken output. These chatbots can support more natural and personal interactions, but also lack some of the conveniences that text chatbots offer.
- *Both*. It is possible for chatbots to support both kinds of communication channels, responding to textual and voice input and perhaps being configurable in how it provides its responses.

4 Discussion

Table 1 presents the results of assessing a number of software chatbots according to our framework, with a dot in a cell marking that a given chatbot exhibits that characteristic. We only assessed a relatively small sample of chatbots thus far, with the chatbots selected opportunistically – we included several chatbots from the software engineering literature as well as sampled chatbots that we found for the different chatbot frameworks (e.g., Hubot, Pandorabots, Wit.ai, J.A.R.V.I.S.). By no means is our selection complete at this time; a more comprehensive assessment is planned for future work. Even with the small sample thus far, however, some interesting patterns start to emerge.

First, most chatbots still are simple in their conversations. Many are 'dull' and exhibit a simple request-response pattern that is predictable and does a small job for the developer. A number of them are more advanced in not needing an explicit request, instead monitoring for cues in conversations among developers and then inserting relevant information as needed. Truly conversational chatbots (i.e., ones that build relationships or ones that are more human-like in their conversations), however, still are far off. We did not find any that use alternative vocabulary, which would have been a small step into this direction. Neither did we find any that engage in random chatter to build relationships. While this may be caused by our small sample, we explicitly did search for chatbots like this.

Second, while some general purpose chatbots evolve in their behavior by learning from existing interactions (e.g., Replika[18]), none of the chatbots designed specifically for supporting software development does to date. We expect this kind of functionality to eventually transfer. Note that the learning, that such chatbots need to engage, should be both about the functionality they perform on behalf of the developer and the conversations they have. Both aspects are of course important, and we especially hope to see more of the latter in future software-oriented chatbots.

Third, we were especially surprised that even the collaboration chatbots did not exhibit much of an engaged interaction style or an approach to evolving what they do. It is here that, we had imagined, creative approaches would emerge, but the truth is that to date they operate much like informational and automation chatbots – in a predictable and dull, repetitive manner.

Finally, with the emergence of voice recognition platforms, we had anticipated some chatbots to incorporate voice input and spoken output, but we did not find any thus far. However, we anticipate that this will certainly become more prevalent in future.

Overall, much work remains to be done to realize a future of chatbots that operate more human-like and that might begin to approach the hypothetical conversation we started the paper with. As aptly observed in a recent Google I/O talk:

> "*If you don't spend the time crafting that character and motivation carefully, you run the risk of people projecting motivations, personality traits, and other qualities onto your App and brand that you may not want associated with them." [3]*

This is where we believe the future of software chatbots lies. While the functionality that a chatbot provides is undeniably important, how a developer is able to interact with it, and how natural that interaction feels, will be a critical aspect of designing chatbots for the future. We believe our framework is a small step forward in drawing attention to these kinds of considerations, although we still consider it a starting point to be refined in future iterations.

[18] https://replika.ai/

REFERENCES

[1] Sameera A. Abdul-Kader and John Woods. 2015. Survey on chatbot design techniques in speech conversation systems. *Int. J. Adv. Comput. Sci. Appl.* 6, 7 (2015), 72–80.

[2] Stefan Bieliauskas and Andreas Schreiber. 2017. A Conversational User Interface for Software Visualization. In *Software Visualization (VISSOFT), 2017 IEEE Working Conference on*, 139–143.

[3] Google Developers. *PullString: Storytelling in the Age of Conversational Interfaces (Google I/O '17)*. Retrieved February 4, 2018 from https://www.youtube.com/watch?v=pz8EQHMRr6Y&index=10&list=PL9g5RfWLvLrZd2BhhTj0J4s_iC9j8kgwd

[4] Jennifer Hill, W. Randolph Ford, and Ingrid G. Farreras. 2015. Real conversations with artificial intelligence: A comparison between human–human online conversations and human–chatbot conversations. *Comput. Hum. Behav.* 49, (2015), 245–250.

[5] H. N. Io and C. B. Lee. 2017. Chatbots and conversational agents: A bibliometric analysis. In *Industrial Engineering and Engineering Management (IEEM), 2017 IEEE International Conference on*, 215–219.

[6] Carlene Lebeuf, Margaret-Anne Storey, and Alexey Zagalsky. 2017. How Software Developers Mitigate Collaboration Friction with Chatbots. *ArXiv Prepr. ArXiv170207011* (2017).

[7] Carlene Lebeuf, Margaret-Anne Storey, and Alexey Zagalsky. 2018. Software Bots. *IEEE Softw.* 35, 1 (2018), 18–23.

[8] Bin Lin, Alexey Zagalsky, Margaret-Anne Storey, and Alexander Serebrenik. 2016. Why developers are slacking off: Understanding how software teams use slack. In *Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work and Social Computing Companion*, 333–336.

[9] Rahul Pandita, Steven Bucuvalas, Hugolin Bergier, Aleksandar Chakarov, and Elizabeth Richards. 2018. Towards JARVIS for Software Engineering: Lessons Learned in Implementing a Natural Language Chat Interface. *Workshop NLP Softw. Eng. AAAI Conf. Artif. Intell.* (2018).

[10] Rob Price. 2016. Microsoft is deleting its AI chatbot's incredibly racist tweets. *Bus. Insid.* (2016).

[11] Douglas Robertson. 2017. Facebook shut robots down after they developed their own language. It's more common than you think. *The Independent*. Retrieved February 8, 2018 from http://www.independent.co.uk/voices/facebook-shuts-down-robots-ai-artificial-intelligence-develop-own-language-common-a7871341.html

[12] Bayan Abu Shawar and Eric Steven Atwell. 2005. Using corpora in machine-learning chatbot systems. *Int. J. Corpus Linguist.* 10, 4 (2005), 489–516.

[13] Margaret-Anne Storey and Alexey Zagalsky. 2016. Disrupting developer productivity one bot at a time. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 928–931.

[14] Joseph Weizenbaum. 1966. ELIZA—a Computer Program for the Study of Natural Language Communication Between Man and Machine. *Commun ACM* 9, 1 (January 1966), 36–45. DOI:https://doi.org/10.1145/365153.365168