Don't Believe The Hype! White Lies of Conversational User Interface Creation Tools

Daniel Rough
University College Dublin
Dublin, Ireland
daniel.rough@ucd.ie

Benjamin Cowan
University College Dublin
Dublin, Ireland
benjamin.cowan@ucd.ie

ABSTRACT

Following the initial hype and high expectations of conversational user interfaces (CUIs), a number of creation tools have emerged to simplify development of these complex systems. These have the potential to democratise and expand application development to those without programming skills. However, while such tools allow end-user developers to build language understanding and dialog management capability into a CUI application, actually fulfilling or executing an action still requires programmatic API integration. In this paper, we look at how CUI builders that claim to be "no code required" struggle to yield more than toy examples, with an aim to provoke the community to develop better tools for CUI creation.

CCS CONCEPTS

• **Human-centered computing** → **Natural language interfaces**; **User interface design**.

KEYWORDS

conversational user interfaces, chatbots, end-user programming

ACM Reference Format:
Daniel Rough and Benjamin Cowan. 2020. Don't Believe The Hype! White Lies of Conversational User Interface Creation Tools. In *2nd International Conference on Conversational User Interfaces (CUI 2020), July 23–24, 2020, Bilbao, Spain.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Conversational user interfaces (CUIs), either controlled through voice as Intelligent Personal Assistants (IPAs), or through text as chatbots, are regaining popularity after an initial slump caused by over-estimation of their potential capabilities [10]. Rather than relying on more social forms of conversation, which may not be desirable [5], these assistants have found a niche in both commercial and domestic contexts through simple, task-oriented adjacency-pair dialogues. For domestic use, these types of interactions are employed to check the weather, perform simple search queries, and control smart home devices (if their end-users have the incentive to figure out how to connect them) [1].

Tool name	NLU	DM	NLG	Fulfilment
Dialogflow [7]	intent/entity mapping UI	follow-up/ slot-fill UI	hand crafted	node.js webhook 🟥
Lex [12]	intent/entity mapping UI	slot-fill UI	hand crafted	lambda function 🟥
Watson Assistant[2]	intent/entity mapping UI	follow-up/ slot-fill UI	hand crafted	node.js webhook 🟥
Oracle Assistant[3]	intent/entity mapping UI	XML-like flow script 🟥	hand crafted	node.js webhook 🟥
Rasa [15]	intent/entity scripting 🟥	JSON-like flow script 🟥	hand crafted	python scripts 🟥
FlowXO [9]	keyword matching UI	drag-drop flow editor	hand crafted	built-in integration
Pandorabots [14]	pattern scripting 🟥	XML-like flow script 🟥	hand crafted	no defined process 🟥

Table 1: Popular commercial tools' approaches to each stage of CUI dialog (code-based approaches highlighted in red)

Although they are named as such, many personal assistants are far from personal - they are at best one-size-fits-all Swiss Army knives. Development of their behaviour is driven by organisations or individuals with significant programming skills and resources. To truly democratise agent development by allowing users to personally tailor functionality, we need development platforms that can be accessed by end-users with little technical expertise. In other words, tools for *end-user programming* are needed to allow software end-users - the experts of their own needs and desires - to make their CUIs truly personal [8]. The good news is that a number of these tools exist. The bad news is that fulfilment of services still requires programming experience, yielding nothing more than the ability to develop toy systems for the average end-user. In this paper, we aim to provoke a practical response to our assertion that CUI creation tools hype up their *"no code required"* features, glossing over the need for programming to do anything of use.

2 NO CODE, NO PROBLEM?

Many tools purport to allow complex CUI experiences to be created with no programming whatsoever. For instance, the *Flow XO* pitch insists one can *"Create a chatbot with zero coding skills required"* [9]. Using *Watson Assistant*, IBM claims: *"Don't know how to code? No problem."* [2]. Despite these promises, if a CUI is to perform useful services beyond simple canned responses to anticipated questions, then unfortunately, not knowing how to code is indeed a problem. Figure 1 outlines the traditional CUI pipeline and the general requirements for its implementation in existing tools. Following McTear [13], we illustrate the necessary components of both text and speech-based CUIs, namely **chatbots** and **IPAs**. Further, Table 1 details popular commerial CUI tools, identifying methods used to tailor supported components. Highlighted cells represent where

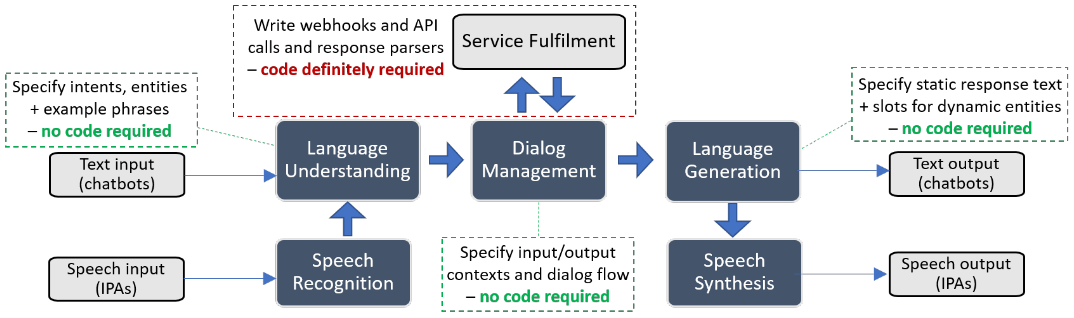
Write webhooks and API calls and response parsers – code definitely required
Service Fulfilment
Specify intents, entities + example phrases – no code required
Text input (chatbots)
Language Understanding
Dialog Management
Language Generation
Specify static response text + slots for dynamic entities – no code required
Text output (chatbots)
Speech input (IPAs)
Speech Recognition
Specify input/output contexts and dialog flow – no code required
Speech Synthesis
Speech output (IPAs)

Figure 1: Traditional pipeline of CUIs and their implementation - adapted from [13]

approaches to a CUI pipeline component require programming experience, or at least an understanding of semi-structured language scripting. Note that speech recognition/synthesis are not annotated in Figure 1, nor described in Table 1 as their use with any of the tools identified is limited to Google's or Amazon's provided services. These components would require significant speech technology expertise to tailor even if they were open-source.

Most commercial tools focus on supporting end-user developers in three key components of the CUI development pipeline; natural language understanding, dialog management, and natural language generation. The current state-of-the-art approach to tailoring language generation in the tools identified is a three-step process. One creates 'intents' - high-level actions a user would wish to perform, specifies example phrases that map to this intent, and annotates these phrases with 'entities' - parameters that customise the intent. Data-driven language models are trained on these phrases, such that no complex grammar creation is required. *Pandorabots* [14] still makes use of a grammar rule-based syntax, but all other commercial tools provide an intent-entity mapping user interface.

Tailoring the dialog management component often involves specifying transitions in the dialog state through a simple graphical user interface [2, 7, 9, 12], or less intuitive semi-structured scripting [3, 14, 15]. Major industry platforms (e.g., Google's *Dialogflow* [7]; Amazon's *Lex* [12]; IBM's *Watson Assistant* [2]) take a straightforward frame-based approach to DM, managing context to ensure that all entities or 'slots' are filled prior to fulfilling an intent. (Harms et al. provide a detailed comparison of DM approaches in both commercial and research tools [11]).

'Natural language generation' is something of a misnomer in current end-user tools in that no programmatic approach is taken to generate natural language replies. In all tools, responses are specified by users as "canned" text with the possibility to include parameterised, entity-specific slots. This alleviates the need to implement any form of semi-structured language grammar.

In short, the tailoring of key CUI components *is* generally accessible to non-programmers in commercial tools. For instance, one can define a pizza ordering intent that recognises entities for size, crust type and toppings, a slot-filling dialog manager that prompts for missing values, and a parameterised confirmation response. Now our dialog is successfully completed, all we need is to actually place the order...and therein lies the trouble.

3 NO CODE? NO! PROBLEM!

Despite bold claims of a no-code-required creation process, involving external services in a developed CUI application in any way - be it dimming a smart bulb or ordering pizza - requires programming.

A CUI's execution of a service is termed "fulfilment" and occurs when an intent has been matched and its requisite entities provided. *DialogFlow* and *Lex* currently offer integration functionality for deploying a CUI as a Google Action or Alexa Skill without the need for coding, but these still require programming experience to connect APIs together. From an end-user programming perspective, previous work has highlighted the difficulty that even enthusiastic end-users have in integrating new devices into their smart home setups [4, 6]. Without support for fulfilment, no-code-required tools can still be used to build FAQ answering CUIs. Yet for applications where some external service needs to be executed based on users' commands, this lack of functionality is a major issue.

4 WHAT NEEDS TO BE DONE?

There are currently a plethora of CUI development tools available, but only really useful to those with the knowledge to implement back-end fulfilment and connect this with user utterances. The average end-user is instead limited to creating a front-end data-gathering dialog. This has the potential to be integrated with external services but, without programming, remains as a non-functional toy. We as a community need to drive the development of CUI creation tools with the following:

- Frank and honest descriptions of capabilities - unrealistic high expectations are a major cause of frustration and abandonment for these tools
- Straightforward integration with existing platforms for service integration such as IFTTT[1]. Such platforms abstract over code-based implementation for a wide (and growing) variety of web services
- No-code integration with speech recognition/synthesis services offered by Google or Amazon, so that developers who desire voice interaction are not locked into using the other components provided by these companies

Addressing these points will be a solid start to meeting the claims of "no code required" tools and consequently democratising CUI development by opening it to those without programming skills.

[1]https://ifttt.com/

REFERENCES

[1] Tawfiq Ammari, Jofish Kaye, Janice Y. Tsai, and Frank Bentley. 2019. Music, Search, and IoT: How people (really) use voice assistants. *ACM Transactions on Computer-Human Interaction* 26, 3 (apr 2019), 1–28. https://doi.org/10.1145/3311956

[2] IBM Watson Assistant. 2020. Watson Assistant | IBM Cloud. https://www.ibm.com/cloud/watson-assistant/. Accessed: 22/2/20.

[3] Oracle Digital Assistant. 2020. Oracle Digital Assistant - Get Started. https://docs.oracle.com/en/cloud/paas/digital-assistant/. Accessed: 22/2/20.

[4] A.J. J.Bernheim Brush, Bongshin Lee, Ratul Mahajan, Sharad Agarwal, Stefan Saroiu, and Colin Dixon. 2011. Home automation in the wild: Challenges and opportunities. In *Conference on Human Factors in Computing Systems - Proceedings*. ACM Press, New York, New York, USA, 2115–2124. https://doi.org/10.1145/1978942.1979249

[5] Leigh Clark, Nadia Pantidi, Orla Cooney, Philip Doyle, Diego Garaialde, Justin Edwards, Brendan Spillane, Emer Gilmartin, Christine Murad, Cosmin Munteanu, Vincent Wade, and Benjamin R. Cowan. 2019. What makes a good conversation? Challenges in designing truly conversational agents. In *Conference on Human Factors in Computing Systems - Proceedings*. ACM Press, New York, New York, USA, 1–12. https://doi.org/10.1145/3290605.3300705

[6] Alexandre Demeure, Sybille Caffiau, Elena Elias, and Camille Roux. 2015. Building and using home automation systems: A field study. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 9083. Springer, Cham, 125–140. https://doi.org/10.1007/978-3-319-18425-8_9

[7] Google DialogFlow. 2020. Dialogflow Documentation | Google Cloud. https://cloud.google.com/dialogflow/docs. Accessed: 22/2/20.

[8] Gerhard Fischer, Daniela Fogli, and Antonio Piccinno. 2017. *New Perspectives in End-User Development*. Springer International Publishing, Cham. 61–97 pages. https://doi.org/10.1007/978-3-319-60291-2

[9] FlowXO. 2020. Flow XO - Easy to use chatbot platform. https://flowxo.com/. Accessed: 22/2/20.

[10] Jonathan Grudin and Richard Jacques. 2019. Chatbots, humbots, and the quest for artificial general intelligence. In *Conference on Human Factors in Computing Systems - Proceedings*. ACM Press, New York, New York, USA, 1–11. https://doi.org/10.1145/3290605.3300439

[11] Jan Gerrit Harms, Pavel Kucherbaev, Alessandro Bozzon, and Geert Jan Houben. 2019. Approaches for dialog management in conversational agents. *IEEE Internet Computing* 23, 2 (mar 2019), 13–22. https://doi.org/10.1109/MIC.2018.2881519

[12] Amazon Lex. 2020. Amazon Lex Documentation. https://docs.aws.amazon.com/lex/. Accessed: 22/2/20.

[13] Michael Mctear. 2018. Conversational Modelling for Chatbots: Current Approaches and Future Directions. In *Conference on Electronic Speech Signal Processing (ESSV 2018)*.

[14] Pandorabots. 2020. Pandorabots: Home. https://home.pandorabots.com/. Accessed: 22/2/20.

[15] Rasa. 2020. Rasa: Open source conversational AI. https://rasa.com/. Accessed: 22/2/20.