Methods and Tools for Prototyping Voice Interfaces

Julia Cambre
Carnegie Mellon University
Pittsburgh, PA, USA
jcambre@cs.cmu.edu

Chinmay Kulkarni
Carnegie Mellon University
Pittsburgh, PA, USA
chinmayk@cs.cmu.edu

CHI'20, April 25–30, 2020, Honolulu, HI, USA
ACM 978-1-4503-6819-3/20/04.
https://doi.org/10.1145/3334480.XXXXXXX

Abstract

Voice technology has experienced tremendous growth in both sophistication and popularity in recent years. This paper considers how researchers and designers might build novel and intuitive voice interfaces. We begin by briefly describing two of the key design techniques for voice design—elicitation methods and Wizard of Oz. We then describe the available tools for prototyping and implementing voice interfaces, and attempt to categorize these according to their function, ease of use, and the fidelity of prototypes they can produce. Finally, we conclude by describing three major aspects of voice communication that are not yet supported by existing prototyping tools, and motivate the need for new design tools in future work.

Introduction

Voice interfaces are notoriously difficult to design well [2]. In addition to the technical challenges of understanding a user's speech and producing natural-sounding synthesized speech in response, the richness and variation in language and considerable social and psychological factors involved make conversational interface design a hard problem [4].

Despite these challenges, voice promises powerful opportunities for interaction: it can enable convenient and more efficient forms of input and output, offers novel forms of entertainment, companionship, and learning opportunities,

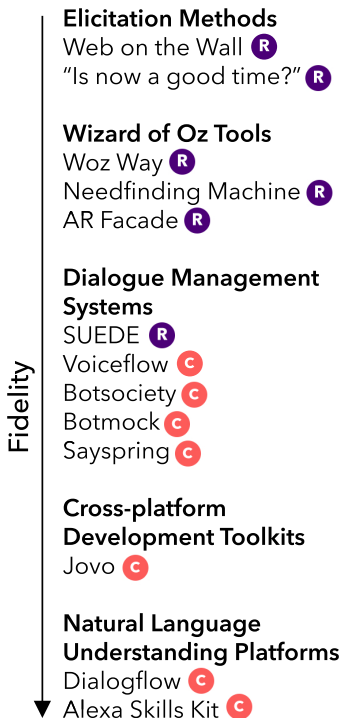

Figure 1: Overview of the voice interface prototyping methods and tools, ordered by increasing levels of fidelity. Beneath each category are examples of relevant research systems (denoted with an R) and commercial systems (denoted with a C). Research studies on Elicitation Methods and Wizard of Oz Tools are not specifically discussed in this paper, but may be of interest to readers.

and provides numerous accessibility benefits. Driven by substantial improvements in speech technology in recent years, there has been a recent resurgence in popularity and interest in voice interfaces, both within the research community, and in commercial applications.

How should we design the next generation of voice interfaces? As Menneckin et al. [2] recently noted, existing tools for voice prototyping "are not fully customizable and will not allow designers to model the depth needed for realistic interactions with a domain-specific assistant. They also do not allow full access to the user utterances and speech data [...], which might be required for the prototyping of the envisioned activities" [2]. At a more fundamental level, voice interfaces still lack the sorts of established design principles that have long guided graphical user interface development, and interaction patterns often do not translate well from graphical to speech interfaces [3]. As we move towards a future in which voice interfaces are ubiquitous, developing the speech-specific design principles—and the methods and tools that allow designers to prototype novel and more intuitive voice interactions—will be critically important.

This paper aims to describe the design space of methods and tools for developing voice user interfaces. Specifically, we consider several "layers" involved in the design of a voice interface, and organize the key prototyping techniques and systems that support voice interface development into five topics, described in Figure 1. In the interest of space, we focus primarily on existing prototyping tools, and close with our main contribution of describing several aspects of voice design that we believe are currently not well-supported by available prototyping tools, and how these opportunities might be addressed in future work.

Low-fidelity prototyping techniques

One crucial prerequisite for designing an effective VUI is establishing a solid understanding of users' needs and expectations around how to interact with the system. Two such methods that help accomplish this are *elicitation methods* and *Wizard of Oz*. At a high level, in an elicitation study, the participant is provided with an outcome of some action and asked to generate whatever command(s) they believe would most naturally yield that outcome state [6]. While the method has been especially successful for designing gestural interfaces, it has also been successful in informing speech-based interaction in capturing the variety—and exposing the similarities—in how users are naturally inclined to interact with an interface.

Perhaps the most common approach to designing voice interfaces is through Wizard of Oz [1, 2]. In a Wizard of Oz (WoZ) study, users interact with a system that appears to be functional, while an experimenter in fact controls some aspect of the user experience behind the scenes. With an interaction modality like voice, WoZ studies are particularly useful in allowing designers to rapidly prototype an interaction without overcoming the considerable technical challenges involved in natural language understanding.

Voice application development tools

Methods such as Wizard of Oz and Elicitations are especially appropriate and useful for open-ended, early-stage exploration. At a certain point, however, once the problem area is well-scoped, a voice interface developer's emphasis may shift from a needfinding approach into a design stage in which the artifacts they build more closely resemble the final voice application they intend to implement. While the boundary between methods and tools for exploratory design and for implementation is difficult to define, we consider the sets of development tools and methods we will

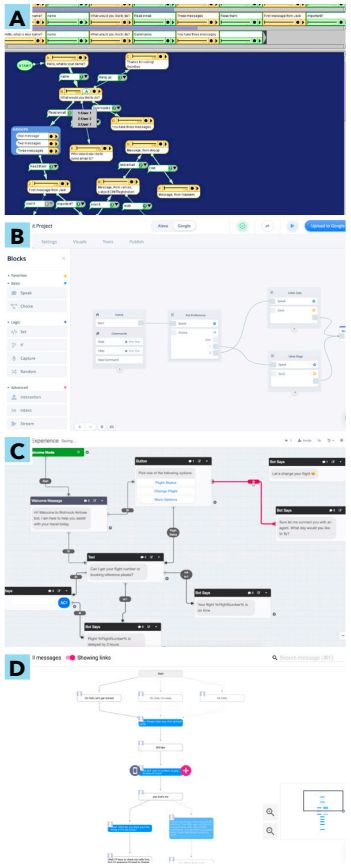

Figure 2: Flowchart-style authoring tools common to Dialogue Management Systems: (A) SUEDE, (B) Voiceflow, (C) Botmock, (D) Botsociety

describe in what follows to operate at a higher level of fidelity.

Dialogue Management Systems

The first category of voice implementation tools is what we will call *Dialogue Management Systems*. At a high level, these systems allow the voice interface designer to fully specify the possible paths in a conversation between a user and a voice agent. Importantly, these systems are intended to have a low barrier to entry by abstracting away lower-level implementation details and allowing for voice interaction design, often without any coding. Figure 2 illustrates the common flowchart-like interfaces common to these dialogue management systems.

Perhaps the earliest example of a dialogue management system is the research tool SUEDE [1]. At the time SUEDE was developed (2000), building speech interfaces required a high degree of technical sophistication, and involved tweaking too many low-level "knobs" and parameters to enable exploration. By contrast, SUEDE was designed to embody design principles analogous to informal tools like sketches and storyboards, and to formalize and augment Wizard of Oz. Based on interviews with six practicing speech UI designers, the creators of SUEDE identified a three-phase process, which they called the "Design / Test / Analysis" method. The SUEDE interface consists of a "script area" for writing high-level dialog examples and a "design graph" that contains the more specific dialog flow, involving branching conversation paths depending on the user's response. Within the dialog flow, the system offered support for conversational support like slot-filling variables based on a user's response earlier in the dialog. To enable easy WoZ testing, SUEDE leveraged user-recorded speech rather than relying upon speech synthesis and recognition technology. In testing sessions, the Wizard operates the interface by listening for the user's response and clicking the appropriate system prompt according to the dialog graph. Each session is then recorded as a full sequence, including the participant's audio response to each prompt. One particularly novel aspect of the SUEDE model of Wizard of Oz design was automatically injecting simulated speech recognition errors by randomly overriding the Wizard's "correct" choice; this technique allows for system designers to capture how users would authentically respond to system failures.

Within today's voice assistant ecosystem, the most popular commercial tools available for prototyping voice interfaces share many of the same core features as SUEDE. Two such systems are Voiceflow and Botsociety.

Much like SUEDE, Voiceflow offers a simple, yet powerful drag-and-drop interface for designing voice interactions for both the Google Assistant and Alexa platforms. Users can drag various action and control blocks onto a canvas and link them together to build complex forking conversations, once again resembling a flowchart (Figure 2B). Relevant pieces of information that a user provides within the context of conversation can be captured as a variable for reference later in the conversation. To design for the multimodal capabilities of devices that have a screen, Voiceflow also allows users to design visual "cards," which display an image and corresponding text to accompany the voice interface's reply. While Voiceflow brands itself as a tool to "visually design, prototype and publish Alexa Skills and Google Actions without writing code," it also offers a high ceiling: alongside basic conversational elements like "speak" and "if/else" blocks, users can also introduce "code" elements into their design. The tool also allows for external integrations, which makes it possible to call a third-party API or database to fetch additional information. Taken together, this functionality offer a

low barrier to entry, but considerable room to build sophisticated interactions for those with an adequate programming background.

Another example of a dialogue management system is Botsociety. Like Voiceflow, Botsociety is intended as a cross-platform conversational authoring tool, but is built to support not just voice interactions, but bots for chat-based platforms as well (e.g. Slack, Facebook Messenger). The authoring tools for designing a voice interaction largely reflects this by representing the conversation between a bot and user through a chat-like interface by default. Botsociety also supports branching conversational paths, which can optionally be visualized as a flowchart (Figure 2D).

While Botsociety offers fewer controls over the details of the voice interface in comparison to Voiceflow, one distinguishing feature is its support of *multi-party conversation*. Prior work has found that voice assistants are often used in group contexts such as with family or friends [5]. Despite these patterns of naturalistic use, most voice prototyping systems have a strict one-to-one conversation model between a single user and a voice assistant. By contrast, Botsociety allows for multiple user *and* bot personas within a design. While *support* for personalization and voice identity recognition in multi-user conversation is not yet available within the underlying technical architecture of skills for the Google Assistant and Alexa, the persona feature within Botsociety offers a useful means for designers to prototype a genre of voice interaction that may be feasible in the near future.

Cross (Voice) Platform Development Toolkits
In other forms of software interface development, toolkits or cross-platform development tools allow a developer to write code once and run it on different environments. With voice interfaces, the dialogue management systems described above (e.g. Voiceflow) largely serve an analogous purpose by allowing users to toggle between different platform development contexts (e.g. for the Alexa or Google Assistant ecosystem) and upload their application to those platforms with minimal additional setup.

However, beyond the cross-platform development capabilities offered by the dialogue managers, Jovo is an "open-source framework that lets you build voice apps for Amazon Alexa and Google Assistant with one codebase." According to its documentation, Jovo abstracts the common functionality from Alexa and Google Assistant applications and allows developers to control both the speech and graphical elements of both voice platforms.

Natural Language Understanding Platforms
To have the finest degree of control over a voice interface, interface designers must work directly with the toolkits that support speech interaction for a given platform. These sets of tools, which comprise commercial options like Dialogflow and the Alexa Skills Kit, operate at a lower level of abstraction from the Dialogue Managers. Whereas Dialogue Managers aid voice interface developers in envisioning the overall flow of a conversation, systems like Dialogflow and the Alexa Skills Kit (ASK) operate at the level of an individual conversational turn, and focus primarily on parsing a user's speech and mapping it to an inferred "intent." According to the platform's own description, "Dialogflow lets you build conversational interfaces on top of your products and services by providing a powerful natural language understanding (NLU) engine to process and understand natural language input." We therefore refer to these systems as *Natural Language Understanding Platforms*.

Both Dialogflow and the ASK consist of similar architectures, and serve in many ways like a "translator" between the user and a voice system. Within these NLU Platforms,

the system designer specifies a series of *intents*, which are the primary unit that attributes meaning to a user's input. According to the Dialogflow guides, intents generally correspond to one "dialog turn within a conversation." These intents are specified by sample utterances that a user might use for that particular turn in the conversation. Within intents, the system designer also specifies "entities" or "slots" which specify a certain category of information or variable definitions that the voice system attempts to extract from user utterances. Slots / entities can optionally be marked as required, such that the system will follow up and ask the user for any missing information before the intent is considered resolved. In these systems, having a well-specified set of sample user utterances and well-scoped slot / entity definitions is crucial in ensuring that the correct intent gets detected at the right time.

NLU platforms make significant trade-offs in sacrificing ease-of-use in favor of extremely granular control over the turn-level mechanics of a conversation (e.g. by tuning the natural language models to "listen" for certain phrases or specialized vocabulary). Managing proper intent matching and maintaining the conversational state can get difficult for voice applications involving more than a handful of conversational turns. Additionally, because NLUs require that the voice designer reads through lengthy documentation and has substantial coding knowledge in order to use them, they face considerable barriers to entry. Because of this, NLU platforms primarily suitable only in the latest stages of the design process when implementing a voice interaction is otherwise impossible through simpler tools like a Dialogue Manager.

Future opportunities for voice prototyping

Already, the tools and methods available have enabled a wide range of applications for voice. In the sections above, we presented the existing design tools roughly in order from those that enable low to high fidelity prototyping. In reality, however, if we consider the Dialogue Management Systems along with the NLU toolkits, it's important to note that the Dialogue Management Systems are very much limited by what the NLU toolkits make available; indeed, systems such as Voiceflow are built to be directly compatible with toolkits like Dialogflow and the ASK. In other words, the designs that Voiceflow, Botsociety and others enable are *constrained* by the affordances of existing voice platforms.

As voice technologies continue to improve, we will soon have new possibilities for interaction that are not currently available in systems like the Google Assistant or Alexa and the prototyping tools that are built upon them [2]. How could voice prototyping tools anticipate and help design for these and other forms of communication? Here, we outline a few important areas for future voice tool development.

Long-term interactions

One area that has immediate practical use but is not currently well-supported by prototyping systems is envisioning how a user will interact with the same voice interface *over time*. Here, we are specifically talking about repeated interactions over several sessions spanning days, months, or even years. At present, a voice designer who wishes to reference data from a prior interaction session with the agent would have to manage a complex series of API calls with a third-party database. Future voice prototyping tools could account for this by allowing users to simulate "time-travel" across multiple sessions, and to offer design blocks that provide easier reference to data from prior conversations.

Turn-taking behavior

Today's voice assistants—and the prototyping tools that support them—only support rigid conversational turn patterns in which the user and voice assistant alternate in neat

Author bios

Julia Cambre is a third year PhD student in the Human-Computer Interaction Institute at Carnegie Mellon University. Her prior and ongoing research has taken both theoretical and applied approaches to understanding and improving voice interaction: in one line of work, she is interested in how the way a smart device sounds shapes users' experiences with the device (CSCW 2019), and in understanding which voices are best for listening to long-form content (CHI 2020). In another line of work, Julia has built voice-based systems, such as a voice assistant specifically designed to support biologists working in a wet lab (DIS 2019). During the summer of 2019, she interned at Mozilla, where she helped to build Firefox Voice, an experimental extension that enables users to control their browser through voice commands. Through an ongoing collaboration with Mozilla, she is currently exploring how to make voice-driven actions easier to prototype and build, particularly for end-users without programming experience. Julia is eager to engage in the workshop as an opportunity to meet the CUI community, contribute relevant experiences working with voice, and find opportunities for collaboration.

Chinmay Kulkarni is an Assistant Professor of Human Computer Interaction at Carnegie Mellon University, where he directs the Expertise@Scale lab. In his research, Chinmay introduces new collaborative computer systems that help people learn and work better.

turns, and cannot speak over one another. Indeed, these turn-taking constraints are baked into the Google Assistant and Alexa speech infrastructure: according to documentation, the systems will consider a conversation "closed" if they do not hear a response from users within 8 or 5 seconds, respectively. However, real human speech is far less structured: conversation partners may interrupt or talk over one another, and can usually intuit whether a given utterance is intended for them without needing to preface each statement with the partner's name (unlike voice assistants, which require wake words like "Hey Siri"). As natural language processing becomes more sophisticated at modeling the intended audience of a user's utterance, we may see a shift towards more continuous, "always-listening" models. Introducing interruptibility and wake-word-free conversation will inevitably change the nature of users' interactions with voice interfaces, and will be important to prototype.

Paralinguistic communication

Speech is an incredibly information-dense mode of communication; in addition to the meaning conveyed by the words themselves, the *paralinguistic* elements of speech (such as pitch, emphasis, or speaking rate) are also highly expressive. The same phrase can take on vastly different meanings as a statement, command, or question based on how it is said. However, most current voice interfaces entirely discard the paralinguistic elements of a user's speech, and instead perform basic speech transcription prior to making inferences about the user's intent. Similarly, the speech *output* from voice interfaces is also largely "flat" or context-agnostic; to better match the intended meaning or inflection of a phrase, voice designers currently need to manually annotate speech content using Speech Synthesis Markup Language (SSML). Moving forward, voice prototyping systems should incorporate richer tools for understanding and acting upon paralinguistic elements in a user's speech, and generating more appropriate-sounding responses in turn. For example, alongside *intent* detection, natural language toolkits could also detect a user's *tone of voice*. Within the dialogue manager, this *tone of voice* feature could then help modify the agent's tone in response (e.g. apologetic if the user sounds angry, sympathetic if the user sounds sad, calm and quiet if the user sounds sleepy, and so on).

REFERENCES

[1] Scott R. Klemmer, Anoop K. Sinha, Jack Chen, James A. Landay, Nadeem Aboobaker, and Annie Wang. Suede: a Wizard of Oz prototyping tool for speech user interfaces. In *Proc. UIST* (2000). 1–10. DOI:http://dx.doi.org/10.1145/354401.354406

[2] Sarah Mennicken, Ruth Brillman, Jennifer Thom, and Henriette Cramer. Challenges and Methods in Design of Domain-specific Voice Assistants. In *2018 AAAI Spring Symposium Series* (2018).

[3] C. Murad, C. Munteanu, B. R. Cowan, and L. Clark. Revolution or Evolution? Speech Interaction and HCI Design Guidelines, Vol. 18. 33–45. DOI: http://dx.doi.org/10.1109/MPRV.2019.2906991

[4] Clifford Nass and Scott Brave. *Wired for speech: How voice activates and advances the human-computer relationship.* MIT press.

[5] Alex Sciuto, Arnita Saini, Jodi Forlizzi, and Jason I Hong. "Hey Alexa, What's Up?": A Mixed-Methods Studies of In-Home Conversational Agent Usage. In *Proc. DIS* (2018). 857–868. DOI: http://dx.doi.org/10.1145/3196709.3196772

[6] Jacob O. Wobbrock, Meredith Ringel Morris, and Andrew D. Wilson. User-defined gestures for surface computing. In *Proc. CHI* (2009). 1083. DOI: http://dx.doi.org/10.1145/1518701.1518866