TALKING WITH MACHINES: PROTOTYPING VOICE INTERFACES FOR MEDIA

Henry Cooke, Thomas Howe, Joanne Moore,
Anthony Onumonu and Andrew Wood

BBC R&D, London, United Kingdom

ABSTRACT

Voice User Interfaces (VUI) and consumer products such as Amazon Echo and Google Home are rapidly gaining popularity and are already being used as media devices in the home; by some estimates, 6.5 million voice-first devices shipped in 2016[1] and we've seen significant growth in their use as a player of BBC content.

As broadcasters and creators of media software, we are interested in the potential of these devices for both the delivery of our existing content and the creation of new media experiences native to these devices. However, the user interface shift they represent – from screen to voice - means that many existing processes and patterns for user experience (UX) are no longer relevant or need significant rethinking to be useful in this domain of interface design. The relative novelty of this class of interface means that there is little existing literature describing UX for VUI.

This paper describes a methodology we are developing to address this lack of literature, followed by some design principles we have discovered during our work on prototype VUIs.

INTRODUCTION

In BBC R&D, we have been running a project called “Talking with Machines” which aims to understand how to design and build software and experiences for VUI. The project has two strands. Firstly, a practical strand which builds working software in order to understand the dominant platforms and their ecosystems. Secondly, a design research strand which aims to devise a user experience language, set of design patterns and general approach to creating voice interfaces independent of any particular platform or device.

The **Methodology** section of this paper describes a practical prototyping method we have been developing for VUI, which is currently on its second iteration.

Key insights for VUI design is a broader, general set of recommendations we have

[1] Figures from VoiceLabs’ 2017 Voice Report (5).

gathered while creating these prototypes that are useful to bear in mind while designing voice interfaces.

METHODOLOGY

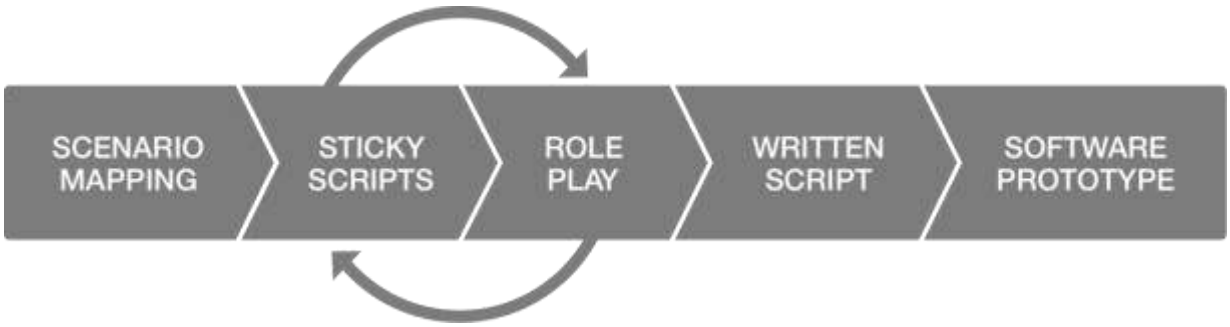

Figure 1: Our VUI prototyping process

At the beginning of our design research, we were faced with a problem: we needed example use cases for VUI in order to work up some possible solutions to those cases. We also needed a way to express and communicate our design ideas for those solutions.

We started by devising a scenario mapping technique to quickly identify possible use cases for VUI. We then incrementally built a design process which allowed us to get from those use cases to fully-featured software prototypes, with intermediate steps that build in complexity and help shape and refine an application idea. We have used a number of common concepts from HCI (Human-Computer Interaction) while building this process. *Fidelity,* as described by Preece et al (4): “[a] low-fidelity prototype does not look very much like the final product and does not provide the same functionality” whereas “[a] high-fidelity prototype looks like the final product and/or provides more functionality than a low-fidelity prototype”. The *role playing* step is influenced by IDEO’s “Experience Prototyping” method(1): “a form of prototyping that enables design team members, users and clients to gain first-hand appreciation of existing or future conditions through active engagement with prototypes”.

The first iteration of this process was developed while initiating our VUI design research. This is the second, refined during a prototyping project with colleagues from BBC Children’s. Figure 1 shows a diagram of the process.

1. Scenario Mapping

Figure 2: Illustration of a scenario mapping board

This technique is a way to think about possible situations in which a VUI could be used. Scenarios come before specific ideas for applications; they're a way to think about the reasons someone might have to use a VUI, and what kind of context they and the device will be in when they're using an application. One scenario may lead to several different ideas for applications.

Scenario mapping is particularly useful in an R&D context where data about existing use cases can be sparse when designing for a new or hypothetical technology. Mapping a number of scenarios can be a effective way to quickly explore a problem space.

Thinking about scenario - the setting of an experience - is especially important for VUI. Voice-driven devices, even more so than mobile, are embedded in a user's existing social situation and surroundings. We cannot assume that we have their undivided attention.

The Scenario Mapping Process

1.1 Categories

We build scenarios from ingredients in a number of **categories**. For our collaboration with Children's, we decided on the following set of categories based on existing research on audience needs and experience from the previous iteration of this methodology.

- **Who?** Someone (or a group of people) using a VUI.
- **Social context.** Are they at home? At school? In a car?
- **Emotional state.** Are they inquisitive? Tired? Task-focused?
- **Brand / character.** Who is the user talking to? What kind of content?
- **Tone of voice.** Peppy? Sensitive? Teacher-like?
- **Device(s).** Amazon Echo? An in-car system? A smart radio?

However, categories are flexible and should be tailored for the needs of a given project.

1.2 Ingredients

Once a set of categories is identified, we populate them with concrete examples: **ingredients**. These are then used to build scenarios. We do this by working in small groups and writing as many examples onto sticky notes as we can bring to mind: for

example, “4 year old boys” for **Who?**, or “excitable” for **Tone of voice**. At this point, we go for quantity rather than quality – the point is to gather plenty of raw material for the following steps.

1.3 Scenario building

Using the categories as column headings, we stick ingredients in rows underneath them that sound like plausible settings for a VUI application: **scenarios.** An illustration of a scenario mapping board showing two built scenarios can be seen in Figure 2.

1.4 Application ideas

After building a few scenarios, we pick a handful that look promising and see if they prompt any ideas for specific applications. Additionally, we often find that ideas emerge early while building the scenarios, which we note down as we go along.

By the end of the scenario building process, we have some ideas for applications which we then work up into prototypes at various levels of fidelity.

2. ‘Sticky note’ scripts

The first step we take when prototyping a voice experience is to map out a rough script with sticky notes. We draw out speech bubbles on a note each for everything said by a person or machine, and colour code the notes for each actor[2]. We also add notes for other kinds of event into the flow - for instance, a machine querying some data, or a person adding ingredients to a recipe. These actions are also colour-coded by actor. It’s also possible to add notes for multimodal events - things like information appearing on the screens of nearby devices, or media being played on a TV. Figure 3 shows a sticky note script from our Children’s VUI prototyping project.

We have found that sketching out an experience like this is a quick, useful way to see how its flow and onboarding will work and which parts seem uneven or need more attention. Using sticky notes means that it is easy to edit and re-order sections as needed.

3. Role playing

[2] We use the term ‘actor’ here to denote a single, discrete voice or conversational entity.

Figure 3: A sticky note script

We have found that the single most useful step in prototyping a VUI is to role-play it in a group setting. Having a 'working' version of an experience immediately gives us a feel for what works and what doesn't – given that the end experience will be voice-driven, it makes sense to test in voice as early as possible. This is roughly equivalent to click-through wireframes for a screen-based experience.

Each person in the team takes a role, read from the sticky note scripts. These can include: system voice, triggered sound effects, props (e.g paper sketches of tablet screens), and other machine tasks (e.g making queries against data sources or running scripts, using categories of sticky note as a guide). The role of user should be played by someone unfamiliar with the idea – they should be 'using' the prototype in as real a way as possible. The whole team then act out the script. This enables us to get early feedback and observations on how the experience flows and any pressure points or parts where the 'user' feels lost or unclear about what's happening.

We use the results of the role-play to drive iterations of our sticky note scripts.

4. Written script

After a few iterations of a sticky script, we write a full script of an experience in a similar format to that of a play or radio drama. This step is more applicable to narrative-led experiences, but can also be useful for data-driven applications if they contain long-running conversations. Every word said by a machine, and approximations of what we think people might say during an experience should be part of this script. Branches can be made from page to page, like an adventure gamebook, for more complex flows.

This written script is a good prototype in its own right; a facilitator can use it in a user-testing session to run an experience prototype with participants taking the role of user.

Running the script through with humans and machines
It's important to keep hearing the script read out loud as it develops - some things which look fine on the page sound strange when spoken, and hearing it out loud helps to get a feel for the cadence of an experience and any parts which seem overly terse or verbose.

We found that using humans to do the reading through every time changes were made to a script was quite a time-consuming process, so we created a tool using Python and macOS native text-to-speech to generate computer-voiced read-throughs of scripts during development. This technique should be considered a poor second to using natural human speech; its purpose is to give a rough idea for how a script sounds before stepping up to 'real' speech. In this way, when we get to a read-through, a lot of the small problems with a script are resolved and we can use people's time to resolve larger or more subtle issues with a script that machine-reading can't catch.

5. Software Prototypes

The highest-fidelity prototype we make is in software – either for a VUI device itself, or a platform which allows us to try things not yet possible on VUI devices. For example, for our recent work with Children's, we built one working prototype on Alexa and one on iOS.

The iOS prototype uses record and playback of the user's voice, a feature not currently present on current VUI devices. However, the platform is irrelevant in this case – we're testing the experience, not the platform – and in user-testing sessions we hide the phone and use a hands-free speaker to focus attention on the sound.

This is the prototyping method which gives a result most illustrative of a final product. Building on a target platform also allows us to test with a larger group of testers in their own homes and, with a disciplined development process, opens the possibility to iterate from a prototype to a releasable product.

KEY INSIGHTS FOR VUI DESIGN

As a result of developing these techniques and working through the Children's prototyping project, we have collected a number of key insights. We present those insights here as seven recommendations for the creators of VUI experiences.

1. Tone of voice

When an interface's only line of communication with its user is voice, we have found that the tone of that voice is overridingly important – as important as the choices made about colour, typeface and layout in a visual application.

This means thinking about the vocabulary and writing style used in an application's voice prompts and responses. An application mostly concerned with responding to direct requests for specific information should be pithy and concise in its responses. An application designed for people in a more relaxed, open frame of mind can be more discursive and chatty.

2. System voice vs talent voice

Our research shows us that the system voices on Alexa or Home are optimised for short, concise answers to requests, and not suitable for reading out long passages of text, especially if that reading requires natural intonation. Using recorded human voice allows for natural-sounding speech at the cost of production overhead. Additionally, once a voice is recorded, it can't be changed. A voice application using recorded talent speech will never be as adaptable as one which generates dynamic speech.

When using a recorded talent voice instead of a synthesised voice, considerations should also include timbre, intonation and delivery style.

3. The expectation of 'smart'

In an earlier project, we prototyped and user-tested some conversational interfaces (CUI), mostly over text messaging channels, but with some VUI. One of the most striking things we found from that testing was the expectation that users had about the intelligence of the entity with which they were conversing. Since people were communicating with something that appeared to be smart enough to respond to natural language and to have a personality, they assumed that it was also smart enough to be able to answer the kinds of question they'd ask another person. This is a effect observed elsewhere in HCI research, and is discussed in e.g. Taylor (2), Vinayagamoorthy et al (3).

This is an important thing to bear in mind when designing conversational systems, because it's very rare that such a system will be able to deal with spontaneous, open language. Most applications will have a limited domain of knowledge: for example, a story about witches or the programme catalogue of a large broadcaster. It's important to communicate to users the limits of a system without driving them away.

4. The importance of suitable data sources

Another finding from our previous CUI project is that while it seems intuitive to be able to ask questions against a large dataset (for example: the news, or a list of programmes), these types of application can only be built if there's an extensive, well-tagged and searchable data source to query. In these cases, the interface itself and parsing the user's intent is a relatively straightforward problem to solve - the hard problems are collating, sifting and re-presenting the data required to answer the user's questions. These kinds of application are a lot more about data processing than they are about natural language.

5. Dealing with a limited vocabulary and letting the user know what they can say

Most VUI systems don't allow completely free, spontaneous speech as input; developers must register upfront, the collection of phrases a user is expected to say in order to interact with an application and keep it updated as unexpected variations arise.

Given that this limitation exists, there is a problem in communicating to people what they can say to navigate an application. Some developers choose to do this upfront, listing possible commands when an application starts for the first time. However, this can sound clunky, provides friction for people wanting to get to the central experience of an application and requires recall at the point where an interaction becomes available.

Another way to do this is to wait until an interaction is about to happen, and then tell a person what they can say: "you can say forward, backward or stop." However, this can seem mechanical, and interrupts the flow of a longer conversation or fictional piece.

Example solutions

- In a fictional piece, a choice could be set up as an argument between two characters
- A set of choices that is naturally limited, e.g numbers from 1-10 or star signs.

6. Turn-taking and letting the user know when they can speak

The 'ding'

This seems like the most straightforward way to let someone know they can speak - "after the ding, say your choice". However, there's subtlety here: does a voice say "ding" or play the sound itself when referring to it? Is this confusing to the user? They have to understand the difference between a referential ding and a real one. If the word "ding" is said, do users understand that this means a "ding" sound when it's played?

Audio mix

A more subtle way of letting the user know that they can speak in fictional, radio-like pieces is by using the audio mix. If music or sound beds are used during the action, these can be dropped out at the time a character or narrator is addressing the user, signifying that focus has moved away from the fiction and that the user is alone with the narrator.

System voice
The system voice itself can be useful as an cue that the user is being asked a direct question: it's the voice that users are accustomed to speaking to on a VUI device. In a piece that includes many voices, the system voice can be used as a 'bridge' or 'mediator' between the user and the fiction world.

7. Modes of address and managing the user, narrator and other characters

When a person interacts with a voice application, they're always interacting with at least one voice - either synthesised or recorded. For simple applications, one voice is often enough - although Google Home's model of handing off to other voices for different functions - "OK Google, talk to Uber about a ride" is interesting, and helps someone understand when they're shifting contexts.

For more complex, narrative-driven applications, it's likely there will be more than one character talking over the course of the experience. In these applications, managing how the characters talk to one another and how the user is addressed becomes a challenge with some subtleties.

In this case, there are a few questions that are useful to consider:

- is the user present in the piece, or an unnoticed observer?
- is the user speaking to characters directly (participating in the action) or using voice to choose branches in a storyline (at a level removed from the action)?
- can all the characters in the piece address the user, or just one?

Using a narrator / mediator to communicate with the user can simplify the model, but it's still important to consider how the user will know when a character is addressing them directly and when characters are talking between themselves (we call this 'turning to the user').

CONCLUSION

In this paper, we have described a prototyping method which we developed to address our own problem: as designers attempting to prototype and describe VUI experiences, there was very little material we could draw on to inform our approach. We intend to continue to develop and refine this methodology by working through live prototyping projects with collaborators inside and outside the BBC.

We have presented some of our key findings as a result of this prototyping work in the second section of this paper. We use these findings as 'rules of thumb' in our own design practise; we hope to gather more of these findings as our work continues and build towards a set of best practises for VUI design.

We present both of these sections in the hope that they will prove useful to other designers encountering the lack of literature that prompted us to initiate this project.

REFERENCES

1. Buchenau, M. and Fulton, J., 2000. Experience Prototyping. Proceedings of the 3rd conference on Designing interactive systems: processes, practices, methods, and techniques. pp 424 to 433.

2. Taylor, A. S, 2009. Machine intelligence. Proceedings of ACM CHI 2009 Conference on Human Factors in Computing Systems. pp 2109 to 2118.

3. Vinayagamoorthy, V. and Gillies, M. and Steed, A. and Tanguy, E. and Pan, X. and Loscos, C. and Slater, M., 2006. Building Expression into Virtual Characters. Proceedings of VRCIA 2006 ACM International Conference on Virtual Reality Continuum and its Applications.

4. Preece, J., Rogers, Y. and Sharp, H., 2015. Interaction Design: Beyond human-computer interaction (4th ed). pp 314 to 320.

5. Marchick, A., 2017. The 2017 Voice Report by VoiceLabs. http://voicelabs.co/2017/01/15/the-2017-voice-report/

ACKNOWLEDGEMENTS

Many thanks to Lisa Vigar, Liz Leakey, Mark O'Hanlon and Suzanne Clarke from BBC Children's who were excellent collaborators on the pilot prototyping project which helped validate and shape our methodology.

We are grateful to Sacha Sedriks, Vinoba Vinayagamoorthy and Tristan Ferne from BBC R&D for their suggestions and support while writing this paper.