# The Role of Data in Voice Interaction Design

*By Jon Bloom, Louise Tranter, Susan Hura, Peter Krogh, Jenni McKienzie, Mary Constance Parks, Darla Tucker*

## Introduction

Here is a common scene: You are in a meeting with a customer. The customer makes a design recommendation. Perhaps they say, "Near the beginning of the call flow, we would like you to mention that the caller can say 'main menu' at any time." If there is a voice interaction designer in the room – and perhaps it is you – they respond to the customer request by saying, "That is not best practice. Advertising global commands at the top of a call taxes the caller's working memory."

Even if you have never said something like that, we will admit to having said it countless times. Voice interaction designers, the authors included, espouse best practices like they espouse carbon dioxide. The problem is, do we know for sure that they are true? How do we know that? What data do we have to back it up? If we do have data, is it sufficient to back up our claim? And let's say for argument's sake that for any particular one we actually do have data. There are plenty more claims out there that are backed up more by anecdotal evidence and gut instinct.

Voice interaction designers have tried hard to come up with a set of rules for how to write prompts, construct menus, handle retries, direct voice talents, write grammars, and so forth. Most of this effort has come from trial and error with whatever projects each designer has happened to work on. A tiny bit has come from pure research. Individual designers and companies have thus built their catalogs of best practices. They aren't always in agreement with each other. The process of coming up with a universal set of best practices is further complicated by the fact that the field is relatively new and small, and each project has unique qualities that do not necessarily allow for generalization.

As a field, we simply may not ever be able to put forth a list of universal best practices. Given the aforementioned challenges, we need to communicate several things to our customers when discussing best practices. First, when we have data to back it up, state it and share it. Second, if the data is more anecdotal in nature, be honest about that. Third, be clear that best practices are purely a starting place. Fourth, data needs to drive our decisions to use these best practices or branch out and find what's truly best for the project at hand.

To be fair, some research does enter into our field from the academic community, specifically from the fields of memory research and sociolinguistics. But it is hard to map those results onto

the practice of IVR design. Too often, we abuse those results to make our points (George A. Miller's 1956 paper on "The magic number seven, plus or minus two" comes to mind). In addition to academic research, we have taken from Jakob Nielsen and others in the field of graphic design the practice of usability testing our work. However, usability testing comes in handy after a design is implemented, or at least storyboarded, not when we are just starting to put pencil to paper. Also, as mentioned earlier, generalizations are hard to make when looking at qualitative results from one project.

The good news is that the field is beginning to change. Many of us are now looking at multiple sources of qualitative and quantitative data over the lifespan of a project in order to design our applications. Some of these sources lend themselves more to the creation and tweaking of best practices than others, but altogether they help us create more informed designs. *In fact, the entire endeavor of achieving a set of universal best practices is becoming secondary to understanding the current project and letting the data at hand steer your design*. Some day soon, it may even be safe to say to a customer "Gee, Mrs. Customer, I don't know the answer to that. But I do know how to find out!" Seen this way, the voice interaction designer's specialty becomes less about blindly applying general best practices than knowing how to *find out* what is best for the *current* project. Certainly, best practices and data-driven design are not mutually exclusive. However, it is much harder to come up with defensible best practices than it is to come up with data that informs your current design. So the more focus we can put on the latter, the better.

So let's look at an ideal project in roughly chronological order to review the healthy mix of qualitative and quantitative data that can be used to inform a design. If we may tax your working memory for a moment, keep in mind while you read this list that items at the end of the list inform the items at the beginning when the project goes into its next version.

## Ways to Gather Data

### Exploratory User Eesearch - Quantitative

Here we use data from pre-existing solutions to determine if the application we are building is a sensible idea, and if so, what input should it be prepared for? This is usually quantitative data which includes: overall call volumes, breakdown of call reasons, customer satisfaction data, and if you are replacing another app, agent requests and hang-up rates.

### Exploratory User Research - Qualitative

The pre-existing data we mentioned earlier – call volumes, call reason breakdowns, opt-out rates, etc. – give us a barebones, flat description of our callers. It is like a play-by-play description minus the color commentary. To get at the heart of our callers' motivations, we need to dig deeper. We may know, for example, that 40% of callers pick flight status on an airline's IVR, but we could not know from such quantitative data that it is because the company's web designers have done a bad job exposing their flight information tool online. To

obtain the color commentary, several methods are available. Ethnographic studies, focus groups with potential callers, user needs assessments, etc. can all be used. These are usually conducted with few participants, but each participant provides large amounts of qualitative data.

## Call Center Visits

Call center visits preferably include three parts, all qualitative in nature. First, the designer sits with agents while they take calls. Second, the designer interviews the agents about what they heard. It is best if the interviews occur one at a time so that no agent feels intimidated by more charismatic, vocal agents. Finally, the designer interviews all of the agents in a group about any conflicting stories the designer heard in one-on-one interviews.

## Usability Testing

Usability testing can be broken down into exploratory testing before code is complete, and validative testing afterwards. But this breakdown hides the fact that usability testing is ideally on-going. So although we are placing it here in our chronological list, testing design decisions with potential callers should be *iterative*. Early on, it may involve role-playing with participants, then one might move on to Wizard-of-Oz testing, and then finally the designer can test the actual application. Much has already been written about the merits of usability tests, so we will not go into detail here. Suffice it to say that designers should consider them mandatory and that they provide some of the most useful qualitative data imaginable. An unforgettable usability test was one in which we learned before it was too late that “check account balances” was a bad menu option and that “review account balances” was a good one. We leave it to you to figure out why.

## Performance Analysis

Data obtained from post-deployment performance analysis is arguably the most crucial data one can get. It is the pat on the back for a job well done or the kick in the pants for a total flame-out. Most often, it is a little bit of both. But despite its importance, performance analysis data is also the most often overlooked.

Once an application is deployed and taking calls, quantitative analysis starts to take on a bigger role. With access to the proper tools, call logs, and recordings, designers can find successes and uncover areas for improvement in their designs. Designers can subsequently begin ramping up for the next version of the application.

A logical starting point for performance analysis involves looking at high-level data like percentage of callers who complete their call within the IVR, agent requests, hang ups, and call durations. Once we uncover irregularities at that level, then we can start drilling down into specific paths as well as individual dialog modules and grammars in order to uncover causes of those irregularities. The specific data one might review at this micro level includes a dialog module’s retry rate, speaker verification success rate, and speech science data like out-of-grammar rates, false accepts, false rejects, correct accepts, and correct rejects.

In the optimal situation, this analysis is performed with the previous solution still taking calls from the same population as the new solution. With this kind of 50/50 traffic split, gauging performance can be very safe. We know that the calling populations are "the same" so there is little fear in comparing and contrasting. We can safely perform inferential statistics on the data (e.g. chi square, t-test, ANOVA, MANOVA, etc.) in order to determine if there is a significant difference in performance. The alternatives to a traffic split – like comparing the old solution running at Time A to the new solution running at Time B, or comparing the two solutions running at different call centers – are fraught with hazards. One cannot tell if a change in performance is due to the new solution or some other difference in caller populations. Running a traffic split allows us to avoid fixing things that aren't really broken or are broken for reasons external to the thing we just designed.

Of course, if your application is not replacing another application but is instead taking over the work of call center agents, then the creation of more abstruse success metrics is required.

There is another very important opportunity for quantitative data collection during performance analysis: A/B comparisons of more than one design possibility. Earlier on in the design process, you may have come across differing opinions about how a certain part of the call should go. Instead of solving the problem by seeing who can pontificate the loudest, or by comparing alternative designs in a usability test (absolutely the *wrong* use of a usability test), one can build all of the alternatives that have been debated, run a randomization script within the call flow, and send each call to one of the alternatives. Because the assignment of a call to a design alternative is random, the resulting data are clean. As long as you obtain enough data from each condition ("How much is enough" is a good question which we do not have time for in this paper), you are in a good position to confidently choose the best design alternative for your current application. Of course, the name "A/B comparisons" is somewhat weak because the comparison could be between multiple design alternatives – five, ten, fifty, whatever. The only limit on experimental conditions is the amount of data one can get in the time required to make a decision.

In addition to these quantitative methods of post-deployment performance analysis, one must also listen to individual calls. How are callers responding? Where are they getting lost? Where are they hanging up? Information like the kind found in live calls and call recordings often provides the "why" after the quantitative data provides the "what," "where," and "when." You cannot simply say that callers are asking for an agent at Dialog Module X and feel like your job is done. To stop callers from asking for an agent, you need to hear what is happening at Dialog Module X. An example from a recent application: there was one location in the call flow with an unexpected number of hang ups. After listening to calls, it turned out the prompt was making an assumption about the caller's account that was not a safe assumption at all. Countless callers were protesting what we had just said and then demanding an agent or getting automatically transferred because they had reached the maximum number of timeouts permitted by the application. Without call listening, that prompt problem may have persisted.

Pity the poor designers who have to “throw their designs over the wall” and who are denied the opportunity to do performance analysis on their applications. It is like taking an exam and never finding out the grade, or writing a letter to a loved one and never knowing what the emotional response was. The detective work involved in performance analysis is quite gratifying, especially when it results in clear insights that in turn point to obvious fixes.

## Conclusion

We have tried to make two overlapping points in this paper. The first is that we need to obtain as much data as possible when creating and analyzing our designs, both qualitative and quantitative. We briefly reviewed some methods and metrics that are quite useful. The second point is that post-deployment performance analysis has been grossly overlooked by design professionals and needs to take a more prominent place in our line of work. Overlaying both points is that we need to do a better job of being *researchers*, and the best researchers go in with an open mind. We have long been an industry focused on creating a set of universal best practices. Such an endeavor is admirable, but it can stand in the way of an open-minded approach. If we go into a project with the attitude that we are the expert because we know a list of yet-to-be-validated best practices, we are doing a disservice to our customers and to our profession. A good start toward the goal of data-driven design requires more of a focus on post-deployment performance analysis and running A/B comparisons. The faster that we enhance the role of data in design creation and the faster we perfect our research capabilities, the faster our customers and our field will flourish.