A case study on the planned and actual use of auditory feedback and audio cues in the realization of a Personal Virtual Assistant

Demo paper for the ICAD05 workshop "Combining Speech and Sound in the User Interface"

Bob Cooper

Avaya Corporation
Enterprise Communications Applications Division
Columbia, South Carolina
USA
bob@avaya.com

ABSTRACT

In the following paper, the concept of a Personal Virtual Assistant (PVA) is developed and a collection of user interface problems are enumerated and explained. For each of these areas, this paper discusses how audio feedback was successfully (and sometimes unsuccessfully) used, often with supporting examples. The PVA product discussed is a third generation product and is being sold and adopted in the marketplace Without the benefit of audio feedback and audio cues it is believed that its success would not have been possible. The purpose of this paper is to give the background and context of a workshop demonstration to be held at the conference.

1. BACKGROUND

Over the past 10 years great strides have been made in providing technology aimed at helping the deskbound worker become more efficient and productive. Consider just a few of these:

- Personal computer UI – a rich windowed metaphor which allows easy multitasking and heavily leverages sound and transparency
- Communication oriented applications – email with rich text, attachments & sound; soft phones, instant messaging clients with presence indicators, etc.
- Internet/Intranet access – easy access to all sorts of data; Gigabit LAN is now even included on the motherboard of standard desktop PCs
- Multifunction speakerphones – phones with inbound call announcement, multiple line appearances, message waiting indicators and the ability to transfer, hold, and conference calls
- Etc.

While the deskbound worker has become even more empowered, the typical mobile worker has a cell phone and a laptop. Yes these are vital tools, but easily getting access to corporate data and managing communication with others, especially while in transit, can be extremely challenging.

Years ago, seeing this need and the advance of some enabling technology (namely speech recognition, text-to-speech and the Internet), a small group of engineers and marketing folks decided the time was ripe for a solution to this problem. The solution was realized in the form of a proxy, or Personal Virtual Assistant, which was always available to the user and could help manage and coordinate communications and data access on behalf of the mobile worker. For purposes of this paper, we'll refer to this solution (a software entity running somewhere at the customer's premise) as the PVA.

For PVA to be successful, it had to be powerful (a rich feature-set), intuitive/easy-to-use and most of all…efficient. The theme driving implementation centered on our own truism **"it's all about convenience"** ← essentially meaning that if the PVA was not much more convenient than its alternative (calling someone back at the office and asking them to 'get John Smith on the line' or 'what did John say in his last email' or 'put an appointment on my calendar') then it would be a nice experiment but a market failure.

2. THE CONCEPT

To a large degree, the first version of PVA allowed a worker – using only their voice and a cell phone – to have the capabilities of Microsoft Outlook and a multi-function phone (plus some) as they would have had had they been sitting at their desk. This paper will look at the following subset of the features provided by the PVA:

Outlook Features
- Read/annotate/forward/… email messages – including attachments and finding messages based on sender, status (read, unread), type (email, voice mail, meeting request), priority, …

- Listen to/forward/return voice mail messages
- Accept meeting requests, list meetings, book meetings
- Manage tasks
- Call personal and corporate contacts

Phone Features
- Place outbound calls – initiate calls in response to a voicemail (e.g. 'return this call'), using a personal or corporate contact or just by saying and keying in a number.
- Receive calls with call screening – if you're in a 'PVA Session'[1] then you'd be alerted of the inbound call and be allowed to talk to that caller (separately or as part of a conference call) or send them to voice mail.
- Initiate conference calls – and still be able to control each leg of the conference.

One other requirement was imposed on the PVA…it all must work 'hand's free'.

The closer we considered the feature-set and limited communication medium we had at our disposal the more difficult the problem became. However, the end user could care less about our difficulties…to them "it's all about convenience".

3. THE PROBLEMS

With this vision in mind, the constraints of the problem became all too clear. The most prominent of these were:
- **The need to provide a very rich feature-set using technology which is far from perfect**, specifically automatic speech recognition (ASR) -- especially over a cellular connection. Not only is ASR imperfect, it is blissfully ignorant of its mistakes. For example if you say "call John Smith" and it thinks you said "call Jane Smith", it is unaware that it's made a mistake. ASR vendors do return a confidence score with recognition results but these are far from perfect and are best used as general indicators.
- **The need to accommodate both new and extremely experienced users**. Mobile workers – especially those with type A personalities (a characteristic of many successful sales folks) – will not go to training classes or read a manual; yet once they adopt a technology that works, they can be extremely loyal and will demand that it 'do more'.

---

[1] A PVA Session is essentially the time when the user is connected to the PVA – either listening to messages, talking to a remote party which the PVA called, etc. These sessions could last hours and the PVA is silently monitoring the session should you need it to do something. Only the PVA subscriber can control the PVA once in a session.

As such there is a natural conflict between the new user – one who wants to take things slowly and be led through an operation – and an experienced user. Watch an experienced user drive a voicemail touch tone interface…he/she does not listen to the prompts anymore, they are just placeholders. This latter point was an important consideration in our overall solution.

- **The need to let users work in an operational mode of thought**. Less feature rich ASR based 'email reader' systems have failed in the past. These systems structured commands into a typical IVR style 'menu structure'. In such a solution users would have to 'open their inbox' in order to read an email message. Then utter some search command to find the right email. If they then wanted to call that person, they'd need to leave the 'email mode' and go open their contacts. Unfortunately people, when they are engaged in an activity, don't operate that way. It strongly violated the 'it's all about convenience' goal. So we wanted to provide a very flat grammar, allowing such interaction as:

  User> read my messages from Tom Militello…
  PVA> talk…talk…talk
  User> call Tom Militello on his mobile phone.

- **The irritancy and time wasting nature of repetitive or useless information**. This once again ties into the need to accommodate both new and experienced users. New users don't mind hearing a chatty system that gives him/her more 'information' than needed. For example, imagine the PVA is reading the user an email message. The dialog could sound something like this:

  PVA> What can I do for you?
  User> Read my first message.
  PVA> Your first message, is an email message
  PVA> This message is marked urgent.
  PVA> It was received on April, 28th 2005 at 1:47pm.
  PVA> The message was sent from Jason Hucks
  PVA> The subject of the message reads
  PVA> "Help"
  PVA> The body of the message reads
  PVA> "Bob, I need your help when you get a moment"
  PVA> Message complete
  PVA> What can I do for you?

  I admit I'm using some pretty bad UI principles here but it's just to make a point. The first time you use the system (or in demos) you might think this was great… "the ability to get my messages while I'm on road". Now, imagine you're a highly energized mobile worker, who upon calling into the PVA is told "Hello, Bob you have 35 new messages, two of which are

urgent" and you had to wade through the above diatribe every time. You'd never put up with it. It would be easier to think "if it's important, they'll call me" or "I'll hook up the laptop when I get to the hotel" ← neither of which often happens. Bottom line, people typically want the most relevant information first and in a form as brief as they can take it. Non-verbal audio plays a huge role in solving this problem. One more point to consider, some users will be using their PVA for hours/day.

- **The ease of a user losing state**. When sound is your only communication channel and a machine is on the other end of the line it is very easy for a user and the PVA to get out of sync – i.e. whose turn is it to speak. We call this 'losing state'. This was very common during the early days when a user was soft spoken. For example, if a user said something and the PVA didn't hear then they both just sat there not knowing whose turn it was to speak. To make matters worse the PVA is listening on a grammar which is action related – 'call <contact>', 'read <message>' … where as the user (just before a timeout occurs) tentatively says something like 'uhmmm…hello, anyone there'. Audio feedback helps here as well.
- **The lack of an hourglass**. Also, to help prevent state errors, there's a need to let the user know that the system may take a while to work on something. Without this feedback the user may think they the system has malfunctioned or they weren't heard. If the user barges in during this working time, the system will assume that it is being redirected – i.e. cancel what you're doing and go do something new.
- **The need to provide for asynchronicity**. Imagine you're listening to a voicemail or you're on a phone call and someone calls you. Your PVA needs to tactfully interrupt you. Same holds true for when you're managing legs of a conference call ← audio feedback plays a big role here as well.

Naturally, there were many other big challenges (e.g., keeping the command set small and keeping a consistent mental model so that users could quickly understand and remember how to use the product…); however, the problems listed above were the ones where audio feedback and prosody played critical roles. In fact, it was concluded that without audio feedback the product would never have been successfully adopted.

4. THE APPLICABILITY OF AUDIO FEEDBACK

As mentioned above, audio feedback and audio cues played a huge role in solving the above mentioned challenges. These problems are inter-related – for example, the inaccuracies of ASR leads to 'loss of state', the need to accommodate new and experienced users makes the issue of extraneous/repetitive prompts such an issue, etc. – therefore the three main issues on which this paper will focus are

- Losing state
- Extraneous information and the need for speed
- Asynchronicity

Given this, let's look at some specific cases where we successfully used audio feedback to overcome these issues.

4.1.1. Losing State

When people communicate face-to-face there are many subtle cues which are used to help keep the parties in sync. For example, if someone says something that you don't quite understand or hear, you may raise your eyebrows, tilt you head or even give a reluctant though questioning 'yea…?'. Of course there are other more direct methods too…"what in the world are you talking about".

Likewise, consider some of the affordances one has on a PC to address this problem. Take a web browser for example. Just the process of typing in a URL involves the following feedback:

- Audible click from the mouse and a blinking cursor to let you know your selection (address field) was noticed
- Audio and visual feedback of your typing – i.e. you spelled everything right. Oftentimes this is reformatted visually for you ← www.ebay.com is changed to http://www.ebay.com letting you know that the URL was resolved correctly
- Some type of progress indicator – a spinning icon in the upper right corner, a status indicator (implying % complete) in the lower right, etc.
- Finally some content in the main body of the page

As a result, on the PC you get rich audio, tactile and visual feedback to help you prevent or resolve state issues.

When your only medium is audio, your device a cell phone and your remote party a pretty dumb (relatively speaking) server running ASR algorithms which have no clue about human rules of engagement, state errors show up in all sorts of areas. Let's look at just a few examples we had to address.

'**Understood Tone'** – Someone says something to the system, or thinks that they did, how do they know if the system heard them? After all it could just be processing their request… some requests can take a while after all.

To solve this problem an 'understood tone' is played to the user as soon as the system determines a user has spoken a recognizable utterance – one of sufficient length, doesn't appear to be background noise (wind…) and matches something defined in the loaded grammar. This recognize step typically takes < 500ms. Given that this tone is played after *every* valid recognition, it was chosen to be muted and softened such that a user would eventually hear it subconsciously. In my personal experience, I don't even notice it anymore. What I do notice is its absence. So now when I use the system, if I don't hear this tone I quickly repeat my command and move on.

'**Working Sound'** – Given that some tasks take longer than others there is a need for a working sound. This serves the purpose of the hourglass (or spinning globe…) on the desktop and lets the user know that the PVA is working on their behalf. After the PVA has recognized an utterance from a user, if some result cannot be played back to the user with a timeout period the system will echo a working sound to the user. On our system this is a short repetitive set of tones. When the result is ready, the working sound stops and the PVA replies with the requested result.

'**Logon Sound'** – We also implemented a logon sound which is played whenever a user calls his/her PVA and is authenticated. Various authentication models are supported. The most transparent is 'auto-login' – where a user is automatically logged on based on their calling number. In this mode users typically set auto logon to occur when they call from their personal cell phones. Another mode will request the user to enter their account number and pin. Regardless of approach, once the user has been authenticated a bell-like 'logon sound' is played. The logon sound really serves the purpose of reinforcement and assurance. Users find the inclusion of this tone very welcoming and it reminds them that authentication has been performed ← very important since the auto-logon process is so transparent. The sound chosen was a longer, rising warm tone.

'**Record Start/Stop**' – Using the PVA, people are constantly replying to email and voice mail *via* voice. That is, when a user replies to an email message the system essentially turns on a voice recorder, records their response and attaches this to the replying email. Given that the PVA is a hands free product, it was important to let the user know exactly when the recorder started and stopped. The stop point is determined by silence detection. To accommodate this we implemented a record start sound as a two step sequence which rises (low sound, high sound) and the record stop as the inverse (high sound, low sound). These are quick and used consistently throughout the product. Potential state errors can occur if a user's voice softens substantially and the system believes that the user has stopped talking when in fact they have not. In such a case (pretty rare) the user

would hear a distinct 'record stop' sound while they were talking. Without this, they would be talking at the same time as the PVA and most likely not hear/understand what was said.

'**Tonal Separators**' – Imagine you need to use text-to-speech (TTS) to read the content of a typical email message. You need to read the header, the subject and the body. Also you need to do this as efficiently as possible. Rather than doing what was shown above

> *PVA> The subject of the message reads*
> *PVA> "Help"*
> *PVA> The body of the message reads*
> *PVA> "Bob, I need your help when you get a*
> *moment"*

It makes much more sense to do this

> *PVA> with the subject*
> *PVA> "Help"*
> *PVA> <tonal separator>*
> *PVA> "Bob, I need your help when you get a*
> *moment"*

Without some type of separator, the subject and the body would run together. Even if a delay was used, this wouldn't be sufficient since TTS prosody does not do an adequate job of making the two areas distinct. This is just one example but tonal separators are used throughout the entire product. For example, between lists of items – when listing emails, appointments for a day, etc. – a typical list operation might look like this

> *PVA> your first message is from <user> …*
> *<tonal separator>*
> *PVA>next is from <user> … <tonal separator>*
> *PVA>next is a voice message from <user> …*
> *<tonal separator>*

'**Audio anchors**' – Though not necessarily tonal, we also felt that frequent and consistent usage of audio anchors would help eliminate state errors. An audio anchor is nothing more than a spoken phrase or marker that let's the user know where they are in an application. For example whenever the PVA is ready to start a new operation it says "What can I do for you". This phrase does not vary. The same is true when at the beginning and end of an 'assertion window' (see below). In applications which we have built which were more tree structured, audio anchors were placed at each main branch of any tree.

'**Call handles**' – When you use a multi-feature desk phone, a user has the luxury of leveraging buttons and lights to control the phone. No such luxury exists when you are trying to accomplish the same type of operations in a hands free / eyes free environment. To help remedy

this, whenever the user is given a call leg (line appearance), we give the user a handle …in the form of a number...to the call. For example, if a PVA is hosting a three way call for the user and someone calls the user, the PVA announces the caller's name and tells the subscriber (the user controlling the PVA) the call handle (line 4) of that call. From there the subscriber has a mental model of the call state

> *Line 1: Katie Howe   : in the conference room*
> *Line 2: Rick Ulmer : in the conference room*
> *Line 3: Derek Sanders: in the conference room*
> *Line 4: Vladimir Tokarev   : on hold waiting for*
> *   me to talk to him*
>
> *Line 0 is reserved for the subscriber.*

Now the user can

> *User> put line 4 in the conference room*
> *User> drop line 3*
> *User> drop all lines*
> *User> connect all lines ← shortcut to put*
> *   everyone in the conference room*

As will be discussed later, it's important to be consistent in prompting – e.g. if the PVA is expecting the user to refer to items as 'lines' then all prompts should refer to them the same…this helps build a consistent mental model.

'A Consistent Mental Model' – One item that was of paramount importance was the need to develop and use a consistent mental model. By this I mean that it was important to make the operation of the PVA logical and consistent. For example, the process of navigating messages should be equivalent for other items – tasks, appointments, contacts … The use of tonal separators should also be used for all listed items and between all subparts (subject/body, time/location, …). A consistent model not only helps obviate the need to memorize a bunch of unrelated commands but very much helps eliminate state errors.

4.1.2.   Extraneous information and the need for speed

It can be a tough challenge to balance the needs of an experienced/frequent/expert user with that of a novice/infrequent user. This challenge is amplified when your only medium is audio. Given this, the challenge really becomes one of

Providing just the right amount of information for the user/task at hand
Consider the case of a user going through their email messages. What are the most important items a user needs to know:

- The type of message (email or voicemail) – people's sense of urgency is very different for these two types of messages.
- When the message arrived – a relative measure here is more important than an absolute measure, in general.
- Who is it from.
- Is the message urgent.
- What the message is about.

Less important but very necessary in some cases
- Who else received the message.
- Was the message sent directly to me or was I "cc'd" or "bcc'd".
- When exactly was this message sent/received.
- Does the message contain attachments.

This prioritization is even more important when you're listing items.

Let's go back and look at our poorly formed initial example

> *PVA> What can I do for you?*
> *User> Read my first message.*
> *PVA> Your first message, is an email message*
> *PVA> This message is marked urgent.*
> *PVA> It was received on April, 28th 2005 at*
> *   1:47pm.*
> *PVA> The message was sent from Jason Hucks*
> *PVA> The subject of the message reads*
> *PVA> "Help"*
> *PVA> The body of the message reads*
> *PVA> "Bob, I need your help when you get a*
> *   moment"*
> *PVA> Message complete*
> *PVA> What can I do for you?*

Could better be conveyed as

> *PVA> What can I do for you?*
> *User> Read my first message.*
> *PVA> <understood tone> Your first message if*
> *   from Jason Hucks*
> *PVA> Delivered just minutes ago*
> *PVA> With the subject "Help" <audio delimiter>*
> *PVA> "Bob, I need your help when you get a*
> *   moment"*
> *PVA> What can I do for you?*
> *Continuing*
> *User> Next*
> *PVA> <understood tone> Your next message if*
> *   from Russ Innes*
> *PVA> Delivered yesterday evening*
> *PVA> With the subject "Vacation" <audio*
> *   delimiter>*
> *PVA> "Bob, Just wanted to let you know I'm*
> *   going to be on vacation tomorrow."*

PVA> What can I do for you?

Message, by default, means <u>email</u> message. Other message types are voice mail message, task request, meeting request, etc. Whenever possible we chose to replace words – "and the body contains" – with sounds <audio delimiter>. Shaving off a small amount of time and repetition can have tremendous user adoption benefits.

We also replaced items such as URL's with audio placeholders. You may think that users want to hear this, but our users have much preferred to 'jump over this but let us know it's there'. The first time you get an email with 5 URLs as the first thing in the body I think you'll agree.

One more point that you may be questioning. How wise is it to "skip" information or "summarize" information? For example in the snippet above we substitute relative date information for that which is absolute. Also we don't tell you the other email recipients. Sometimes you need to know this – especially before you 'reply' or 'reply all'. This same issue comes up in all sorts of cases – meeting requests, task requests, etc. The way we solved this was by providing a single consistent mechanism, "more detail". More Detail is context sensitive and when the PVA hears you say that, it tells you almost anything you'd want to know – hopefully in 'most relevant information first' order.

4.1.3.   The Case for Assertion

Keeping the user and the PVA in sync is good. Shortening the length of dialogs – replacing words with sounds – is good. But how can we reduce the number of steps needed to accomplish a task?

For example, when you reply to a message, ~80% of the time you want the PVA to turn on the recorder, record what is spoken, turn off the recorder and send the message. However sometimes you need to do other things such as
- Re-record your message – e.g. your first take was bad.
- Add other recipients to your reply.
- Mark your reply urgent.
- Continue recording where you left off.
- Etc.

So how do you balance the need for these features without sacrificing the 'default' behavior? Alas the case for assertion!

Assertion essentially says "I'm going to do the default behavior unless someone tells me otherwise". The goal is to make the typical case as efficient as possible. If the user wants to do the non-default behavior, then we'll provide for that, it just may take a little longer. If this behavior doesn't suit the user, 'assertion' is something a user can disable via some user configuration settings and in our implementation, it was disabled by default.

Sending a message with assertion enabled looks something like this

> *PVA> What can I do for you?*
> *User> Send a message to Gurmeet Lamba*
> *PVA> Record after the tone*
> *PVA> <record start tone>*
> *User> talk…talk…talk…<silence>*
> *PVA> <record stop tone>*
> *PVA> <assertion window started> "Sending message"*
> *PVA> <3 second pause>*
> *PVA> <assertion window ended> "Done"*
> *PVA> What can I do for you?*

The word "done" always indicates the closing of the assertion window – a tone could just as easily serve this purpose.

How then does one get access to these other features? Just interrupt during the assertion window. Once the PVA notices an interruption during the assertion window it will stop the default behavior and list all the options, so this could have looked like this

> *PVA> What can I do for you?*
> *User> Send a message to Gurmeet Lamba*
> *PVA> Record after the tone*
> *PVA> <record start tone>*
> *User> talk…talk…talk…<silence>*
> *PVA> <record stop tone>*
> *PVA> <assertion window started> "Sending message"*
> *User> stop*
> *PVA> OK, please say one of the following*
> *PVA> "send the message"*
> *PVA> "review the message"*
> *PVA> "continue recording"*
> *PVA> "add a recipient"*
> *…*

4.1.4.   The Need to Support Asynchronicity

How different would your deskbound experience be if asynchronous events from your phone, calendar, IM ... were not supported? It's kind of hard to imagine. It would be kind of like you not hearing the phone ring until after you finished reading a piece of email. In unsuccessful phone-based product attempts in the past, this is exactly what happened – if you were listening to an email message, inbound calls would go straight to voice mail. However the PVA is actually a multi-tasking

entity. Thus, while a user may be having his/her email read to them the PVA is monitoring your public phone number for inbound calls and monitoring your calendar and task lists so that it can alert you when you need a reminder.

When it came time to implement this functionality, mixing audio tones – a different tone per event type – was the only workable solution. The supported event types include

- An "inbound call tone" to signal that a new call had arrived.
- A "calendar reminder tone" to signal that a calendar reminder has fired ← i.e. at the same time that Outlook pushes a meeting reminder to your desktop, the PVA mixes a *calendar reminder tone* with the audio currently being streamed to your ear – whether it's a TTS rendering of email, audio from a voice mail message or audio from a live call.
- A "task reminder tone" to sequence when a task reminder has fired.
- An "administrative tone" + warning when a system administrator needs to do something to the system which would affect your session, e.g. take the system down for maintenance.

This functionality is actually easier to demonstrate than describe on paper, nevertheless I'll try.

Here's a case where a user is listening to a message and a call comes in

> *PVA> What can I do for you?*
> *User> Read my urgent messages from Jeff McElroy*
> *PVA> <Jeff McElroy> Your first urgent message…*
> *PVA> talk…talk…talk…*
> *PVA> talk …<inbound call tone played to user> …talk…*
> *User> stop*
> *PVA> you have a call from <Al Schumann> would you like to take this call?*
> *User> yes*
> *PVA> connecting to line 1 <connection tone>*
> *User+Line1 > <connected>*

Notice that only the tone is played to the user while they are reading this message. From this tone the user knows the type of event but not, in this case, the caller's identity. This was deliberate in that if you got a lot of inbound calls or had a lot of reminders firing it could get distracting. It's up to the user to decide whether he/she wants to stop what they are doing to get more information. In the above example if the user didn't respond to the PVA event within a timeout period, the PVA would tell the caller that the subscriber is

unavailable and offer to take a message. A better implementation would be to allow the subscriber to tune the amount of detail conveyed with a specific event type.

There are other uses of asynchronous behavior, for example in the management of conference calls; however the case for supporting asynchronicity *via* audio and mixing has been made.

5. A FAILED INITIAL THEORY

No story would be complete without discussing at least one of the major failures. Actually I'm not sure this was a total failure, the results led us to a strong belief in the need for a user model based approach. Nonetheless when we built our second PVA[1] we believed that frequent users would naturally want to push the system to be as efficient as possible. A catalyst to this way of thinking was watching one of our attorneys one day 'drive' his voice mail system. Without blinking an eye or listening to a single prompt for > 1 second, he went through 5 levels of menus and performed four operations…all the while carrying on a conversation with three of us. It was clear that barge in was essential and the user only needs to hear enough information so that state was not lost.

A theory was born.

There were certain interactions which could be made extremely efficient as long as the user was 'properly accustomed' to recognize them. Consider the case where the PVA did not recognize what the user said. One approach is to say something like

> *User > talk…talk…talk*
> *PVA> I'm sorry, could you say that again*
> *User > talk…talk…talk*
> *PVA> I'm sorry, I didn't understand you… ← insulting a user*

And on and on and on. The PVA could try to help the user along…"try to talk louder", "eliminate any background noise", etc. even backoff and try to suggest commands. That's all well and good for a new user however once you are experienced with the system these prompts provide *zero* value to the user…all the user really needs to know is that the PVA didn't understand them. Given this, it seems like a perfect opportunity to replace these useless words with a substitute sound…call it <unrec>.

Now we knew we needed to select <unrec> with great care as (1) it would be played more often than any other

---

[1] The one discussed above is the third revision…don't ask me about rev 1.

tone and (2) it would be the bearer of bad news. Consequently, we chose a soft short distinct tone for this.

There were other areas too in which repetitive prompts could be replaced with tones. For example
- The "what can I do for you" anchor
- Any "yes/no" question
- Any time we needed to prompt the user to say a contact's name
- At the beginning and end of the assertion window
- Etc.

In a purely theoretical sense, anything other than dynamic content could be replaced by a corresponding tone. No we never went there but we thought it was possible one day for a small handful of users.

What we did do though was implement a few of these – and combined this idea with moving toward shorter and shorter prompts. The way it worked was we defined 4 levels of verbosity.
- Level 1 = longer prompts and little to no audio feedback
- Level 2 = slightly shorter prompts and begin introducing audio feedback
- Level 3 = shorten prompts more and begin substituting the audio feedback for words.
- Level 4 = shortest prompts and aggressive substitution

It was anticipated that there would be a direct correlation between these levels and a user's level of experience. And depending on your experience, we would gradually phase out wordy prompt and begin substituting tones as replacements.

The developers were actually quite happy with this level of flexibility but when we went to market it led to confusion. Even our own marketing team that was using the product had problem[1]. Why? Well we designed the system such that new users would begin at Level 1 and the slowly migrate to Level 4. Accordingly once you learned the flow of the system on Level 1 it would be easy to adapt to some shorter prompts. Also at Level 2 the shorter prompts are accompanied with some of the audio tones. Moving to Level 3, you would start dropping some prompts altogether and once you went to Level 4 you would use even more aggressive prompt/tone substitution.

What went wrong? Users would try the system out on level 1, get over confident and then immediately move to Level 4. Then they would complain that they didn't know what was going on … massive state error. Well even though the 4 Levels of prompt/tone substitution never survived the market, it did influence concepts that did make it – verbosity and user modeling.

6. VERBOSITY AND USER MODELING

What got salvaged from the above experiment were a couple concepts – verbosity and user modeling.

The feature of verbosity adjustment is one which the user controls via a personal configuration panel, and though it does not do any tone/prompt replacement, it does allow the user to modify the length of prompts. Consider the two dialogs

Verbosity = High

> *PVA> What can I do for you*
> *User> Send a message*
> *PVA> To whom would you like to send a message*
> *User> Jim Brigman*
> *PVA> <Jim Brigman> Is this correct?*
> *User> yes*
> *…*

Now consider the same tasks with Verbosity = Low

> *PVA> What can I do for you*
> *User> Send a message*
> *PVA> To whom?*
> *User> Jim Brigman*
> *PVA> <Jim Brigman> Correct?*
> *User> yes*
> *…*

The second concept was user modeling. Part of the reason the original concept failed was because the user violated our underlying assumptions – a logical progression from Level 1 to Level 4 based on experience. If however we could monitor their usage of the system, we could potentially prevent this – or at least warn them if they tried to move beyond their 'observed' skill level.[2] Though user modeling was not used in this version on the PVA it was used in other products to adjust prompt verbosity, level of help, etc. User modeling is an area on which we are highly focused and which will lead us to where we really need to go…having the PVA readily adapt to the user (vs. the other way around).

---

[1] Hint: This was an engineering driven feature.

[2] No one liked the idea of preventing a user from setting the level as they saw fit so a strict enforcement was somewhat shunned.

7. CONCLUSIONS & WHERE DO YOU GO FROM HERE

There is a lot of fertile ground to be harvested in the area of PVA-like applications. We believe that as speech recognition gets better, as multi-modal devices become more prevalent (yes, we've built a multi-modal version of the PVA) and applied AI techniques improve, PVAs will become more and more useful. Like a good support staff can help monitor and orchestrate a boss's activities and a team of nurses can assist a surgeon, we will we all come to rely on our own Personal Virtual Assistant. It's just a matter of time and **it's all about convenience**.