Persona: An Architecture for Animated Agent Interfaces

David Kurlander
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052 USA

djk@microsoft.com

ABSTRACT
Several years ago we started a project called Persona, with the goal of building a framework for animated, conversational, agent-based interfaces. The project stitched together state-of-the-art speech recognition and natural language understanding components, with a 3D animation and control system of our own design. The first interface that we built with this framework was a 3D animated parrot named "Peedy", that controlled a musical jukebox. Users could ask Peedy to play various musical selections, and Peedy would do his best to comply, while engaging the user with entertaining sound and graphics.

Persona is an interface architecture, and Peedy was our first test of that architecture. Certain aspects of Peedy worked well, while other aspects were inadequate for the task. However, we learned a significant amount by building Peedy, which will influence future modifications to the agent architecture. Lessons that we learned from building Persona and Peedy will be useful to others building animated, agent-based interfaces.

Keywords
Animated interfaces, intelligent interfaces, agents, natural language, dialog modeling, creatures.

INTRODUCTION
Although computer interfaces are becoming easier to use, they are still too complex and unnatural for many people. Command line interfaces have largely given way to graphical user interfaces, but both of these forms of interfaces are languages that users need to learn. Conversational interfaces, in which users talk to a computer just as they would talk to a person, would be accessible to far larger populations. Such interfaces could be supplemented by a smoothly-animated graphical representation, to give the sense that the user is speaking to a real entity.

We have built such an agent-based interface, called "Peedy". Peedy is a three-dimensional computer-animated parrot that responds to user commands. In order to build Peedy, we first created an architecture for animated agent interfaces. This architecture is called "Persona", and Peedy was the first agent-based interface built with Persona [1]. Peedy also served as the first test of the Persona architecture, and by building Peedy, we learned of a number of things that worked very well and worked very poorly in the Persona architecture. Here we document some of the lessons that we learned with Peedy the Parrot and the Persona architecture. Hopefully they will be useful to others building agent-based interfaces.

In the next section, we list 11 lessons learned from the Persona architecture, and the Peedy test-bed. Following this, an additional section discusses various dialog management requirements that would need to be met for Peedy to perform acceptably under real use. These dialog requirements were determined through Wizard of Oz experiments, and will inform the design of future systems.

LESSONS LEARNED
We learned a number of things by building the Peedy agent, about creating lifelike characters, about implementing conversational interfaces, and about deploying agents. This section describes many of these lessons. Some were learned by making mistakes during the project, others by evaluating what worked well. But all of these lessons should be valuable to people building lifelike character agents and conversational interfaces.

LESSON 1: **Use professional designers and animators**
Professional designers and character animators are skilled at creating the semblance of life through animation. They can create appealing characters that people enjoy. From the very beginning of the project, we had a talented designer and animator, Tim Skelly, develop the character of Peedy the Parrot, and the ancillary characters used in our interfaces. David Thiel, an experienced sound developer, added to Peedy's realism by manufacturing rich soundscapes. Peedy the Parrot is shown in Figure 1, and a demo of Peedy is available on the web at http://www.research.microsoft.com/ui/djk/planner/djk_fg0.mov.

Figure 1: Peedy the Parrot

LESSON 2: **The illusion is fragile**

One of the reasons that professional designers and animators are so important on a project like this, is that the illusion of a living creature, produced through computer animation, is so hard to maintain. At one point in our project, we decided to give Peedy a voice by incorporating a computerized text-to-speech engine into the Persona system. Having a robotic voice for Peedy, destroyed the illusion of a friendly cartoonish bird, and we had to replace the speech synthesizer with recorded, sampled speech from human voice talent. This restored the believability of Peedy.

Similarly, when the frame rate was poor, we needed to refine the graphics engine, so that the animation appeared less computerized. Initially the audio in the Peedy system was poorly synchronized with the video, which violated the realism. This was later fixed.

LESSON 3: **Add variety**

In animation, part of the illusion of life is generated by irregularly occurring and unpredictable behavior. If an action occurs too regularly and predictably, it appears robotic. For example, the action of Peedy blinking follows a stochastic model, as does his head motions and glances away from the user. If Peedy blinked every second, it would look unnatural. Perlin [5] achieves a great deal of realism in his character animations by having almost all of his computer actors' actions governed by layered stochastic models.

Peedy's various autonomous actions had a great degree of variety and richness, in part due to simple stochastic models. However, Peedy lacked variety on a larger scale. As will be discussed later, Peedy could only perform a few different functions, and understand only a small number of requests. This lack of higher-level variety really limited the utility of Peedy.

Furthermore, while Peedy's lower-level actions had a great degree of randomness, his higher level animations did not. For example, there were only one or two ways, that Peedy would animate adding another song to the play-list. This made using Peedy repetitively tedious at best. The latter problem could be improved by adding stochastic elements and more variety throughout the animation.

LESSON 4: **Agents can be engaging**

Even though Peedy was not truly adequate as a musical assistant, people enjoyed interacting with the prototype. They found Peedy to be fun and novel, and seemed eager to have this interface tied to a more functional application. People that were not experienced with traditional interfaces commented that they would enjoy using such an agent-based interface. We were surprised ourselves at how engaging the interface was, and the sense of presence and believability that the agent evoked. Others have noted this with different creature-based interfaces as well. The Catz and Dogz computer desktop-based pets products have proven popular [6], and more recently the Tamagotchi wristwatch-based virtual creatures as well [2].

LESSON 5: **Lifelike presence suggests lifelike intelligence**

Although having an agent with a lifelike presence is appealing, it can also generate problems. For example, since people could speak to Peedy, and Peedy would speak back, they assumed that the system would understand unconstrained natural language. This was far from the truth. In fact, Peedy understood only about 30 sentences, with variables for musical titles, artists, and genres. If someone were placed in front of Peedy and handed the microphone, there would be little chance that Peedy would respond adequately to any request. This violates an important principle of user interface design – that functionality be discoverable.

We did take some steps to limit the assumption that people would make of lifelike intelligence. First, instead of creating a human agent, we created a parrot agent. Parrots, though capable of talking, have little intelligence. We also had Peedy make it clear to the user when he did not understand an utterance.

In truth, it will be a number of years before conversational interfaces are a reality. We distinguish between conversational interfaces, in which the user participates in a dialog exchange with the computer, similar to a dialog that they might have with another person, from simple speech-based interfaces, in which the computer recognizes isolated words and phrases. Part of the challenge in building conversational interfaces is *dialog modeling*, following and contributing to the structure and state of the conversation. After building Peedy, we did an experiment to find out what dialog structures and features people would really expect from a musical assistant. These are presented in the latter half of this paper. However, no existing computer interface can deal with all complex dialog management issues. Due to the great potential of conversational interfaces, this is an important area for research.

LESSON 6: **Constrain the input as much as possible**

Given the difficulty of building broad-coverage conversational interfaces with today's natural language technology, it is important to constrain what the user might say to the system as much as possible. Two different kinds of input constraints can be helpful. The first is domain constraints. The user needs to be told in advance what topics the conversational interface can address. The second is language constraints. The user often must be told what words and constructions the system can understand. This can be done in advance, or as part of the runtime interface. In the latter case, acceptable words or constructions might appear on the screen, or the agent itself might use words and sentences that it understands, hoping that these will be adopted by the user. Maulsby's experiments suggest that users do adopt language employed by a conversational agent [4].

LESSON 7: **Have tight integration between speech and natural language**

Speech recognition is the technology that maps audio waveforms to words. *Natural language recognition* attempts to assign meaning to collections of words. In Persona, the speech and natural language recognition components were weakly coupled. The output of the speech recognition system went to the natural language recognition system, and there was no reverse feedback. This simplistic approach can produce mediocre results at best.

In an ideal system, the speech recognition system would adjust its probabilities according to hints from the natural language system. These hints would ideally be dependent upon the dialogue state. The natural language system would make it clear which candidate word recognitions are valid in the context of the likely sentence, and which are not. Together with the speech recognition system, the natural language component should help to identify disfluencies in the utterances, and prune them out.

LESSON 8: **Authoring animations can be extremely hard**

In Persona, the animations are controlled by a state machine. The state machine receives events from the dialogue management system (itself a state machine), triggering possible animations, and possibly transitioning the state machine to a new state. However, authoring such state machines can be extremely difficult.

For the first implementation of Peedy, we authored these state machines by hand, and learned first-hand how difficult it was. Fixing a bug in one part of the machine often caused new problems elsewhere. We needed a mechanism for making the state machine easier to specify and more manageable.

Later we developed a means of specifying the animation using a planning-based approach. However, traditional AI-type planning can be too slow for real-time animation with precise timing requirements, so we developed a method for pre-compiling these planning-based specifications into a state machines that can be quickly executed in real-time. The planning-based animation specification takes the form of a set of operators with preconditions and postconditions, and a set of goal states to be achieved when a given graphics controller event is received. Using the planning approach, it was far easier to specify animation state machines, and the specifications were smaller and simpler. A complete description of the planning-based animation specification appears in a [3].

LESSON 9: **Database applications are challenging**

The musical assistant domain that we chose for our Peedy prototype was a database application. The program consulted a database of musical selections that Peedy could play. The database contained the titles, performers, genres, and durations of the music, among other information. Theoretically, the application should be extensible – simply add a new song or compact disc to the database, and Peedy should be able to play and answer questions about the new music.

However things were not that simple in practice. The speech system required that it be given the precise phonetics of every word that it could recognize. So artist names, song titles, and album names had to be encoded into a special phonetics language. Hence anybody adding new music to the system had to be taught the phonetics encoding.

On the output side, recall that we chose to use sampled speech so that Peedy's voice would not appear robotic. The sampled speech was in the form of words and phrases that were stitched together at runtime. Since Peedy needed to be able talk about the artists, song titles, and album names, every time that we added new music to the system, we had to have the voice talent participate in another recording session. The problem was exacerbated by the fact that the additional speech output samples and phonetic encodings were not stored in the central music database, but in other data stores.

These same problems would be encountered in nearly any database application. By choosing a non-database application for Peedy, we could have made our jobs much simpler, and our system much more easily extensible.

LESSON 10: **Successful systems will be built by cross-disciplinary teams**

On the Peedy project, we had animators, sound designers, user interface researchers, graphics experts, and software developers all working together to make a compelling agent-based interface. Without any one of these specialists, the project would have suffered and the quality of the system would be less than what it was.

If we were to do this again, it would be helpful to have specialists in speech recognition, natural language, and machine learning participating as part of the project too. Although we did work with such specialists and used general-purpose components that they had built previously,

it would have been helpful to have these specialists working side-by-side with us, on a moment by moment basis, helping to specialize their components for our domain, and making their components interoperate better with each other and with the Persona architecture.

LESSON 11: **Use your system for real work, and distribute system to others**

In building novel technologies, such as animated, conversational agents, it is important to use the technology every day, see what needs to be fixed or improved upon, and address that. The Peedy system was highly ambitious, and attempted to put a number of very novel components together in a working system. We succeeded at creating a vision for future interfaces, that pointed out numerous technologies that needed to be better integrated and developed further. However, we did not succeed at building an interface that any of us could realistically use on a daily basis. Without constant use, certain deficiencies are never addressed.

Furthermore, such systems should be distributed to others to use. This would have been difficult with Peedy, since it required several machines, including a costly graphics workstation, to be networked together, dedicated to running the Persona architecture. However, distributing Peedy would have given more people a sense of the promise of such animated, conversational agents, and the challenges that need to be solved in order to make these agents a reality. The lesson that research systems should be used for real work and distributed to others, applies not only to agent-based interfaces, but to all software research.

DIALOG MANAGEMENT

As mentioned in the last section, one of the great challenges in creating conversational interfaces is building a dialog management component that interprets the user's statements appropriately in the current context, and guides the user in a productive direction through additional dialog. One of the lessons that we learned in building Persona and Peedy is how hard this really is.

Persona has a simple dialog management system, designed as a state machine. Whenever the user made a statement to an agent built with Persona, the system would evaluate the statement in the context of the current dialog state, and choose a subsequent state for the dialog manager.

The dialog state machine that we built for Peedy was extremely simple, and not at all robust enough for unconstrained conversation. Yet users of Peedy had no means for determining the limits of Peedy's conversational ability, and constantly encountered its limits.

Later, in order to help us scope out necessary revisions to Peedy's dialog model, we performed a set of ten Wizard of Oz studies, where real people interacted with another person taking on the role of an automated musical assistant. Wizard of Oz experiments are frequently used in the field of human-computer interaction, since they allow system designers to spec out the necessary capabilities of an interface, without going to the trouble of actually building a real system [4].

By analyzing the transcripts, we identified eight different types of dialog challenges that we feel a truly compelling conversational agent-based interface would have to address in some way. In this section, we describe these dialog challenges, and give examples from real human subjects and testers in our musical assistant Wizard of Oz mock-up. Lines of dialog preceded by "A" are from the human assistant, and lines of dialog beginning with an "S" are from the human subject.

Challenge 1: *Handling complex conversational features, like repetition, simultaneous speech, irrelevant utterances, and utterances intended for someone else.*

Often if the musical assistant did not respond quickly enough, the subject repeated a request. The assistant needs to be smart enough not to respond multiple times. An example of this is below.

S: What do you have?
S: What do you have?
A: We have Luka Bloom, we have Nancy Griffith.

In other cases, the subject and the assistant spoke at the same time. Typically, the subject interrupted the assistant as in the example below.

A: There's also Miracle Man, No Dancing, Blame it on Caine, Alison, Sneaky...
S: Oh, I'd like to hear Alison.
A: ...Feelings. OK.

For an agent, it is important to be able to detect an interruption, and act upon the request.

In some cases, the subject made utterances not intended for the musical assistant at all. In one case, the subject, while listening to music, made comments relating to a recent news event. The agent-based interface should be able to distinguish comments directed towards it from comments directed to someone else.

Even comments directed towards the assistant were occasionally irrelevant with respect to its task, and computer agents, like human agents, should be able to handle this.

A: Would you like to hear the Ray Charles version?
S: Yes I would actually.
A: One moment.
S: It's funny, since I was just reading about Ray Charles. (Subject holds up the newspaper.)

In the exchange above, the final comment is irrelevant to the agent's task of playing the Ray Charles music, but the assistant hears it anyway. A good dialog agent would be able to detect such situations and not let it affect its actions inappropriately.

Challenge 2: *Inaudible speech*

People can tell when they are unable to hear a word that is said. Computers have more trouble with this. The problem of inaudible speech is particularly tricky for a music assistant, in that the assistant must be able to hear the user through the music, which not only can mask the words, but also create an ambiguity over what was said by the user and what was sung by a vocalist.

The Peedy system partially dealt with this problem by using a directional microphone with a push-to-talk button. Thus the user could explicitly express when they were making a statement for the agent, and the music did not interfere very much with Peedy's speech recognition capabilities. When Peedy could not recognize a request with a reasonable confidence, it would say "Huh?" to prompt the user again. However, Peedy still made recognition errors.

Challenge 3: *Understanding corrections, including the user misspeaking, making an incorrect statement, or changing his or her mind.*

People frequently make mistakes in their own utterances, and sometimes correct these mistakes themselves. A good conversational agent would be able to handle these mistakes, and recover from them in a natural manner. In the example below, the subject corrects a misconception of the assistant. The assistant made an incorrect assumption in reaction to the subject's first response, and the subject finally corrects the assistant.

A: Would you like to hear something else from the same album or are you interested in other female vocalists from a different album?
S: Yes.
A: We have a different version of Summertime, with a different singer, on the Porgy and Bess CD.
S: Oh, no, I meant I'd like to hear more by Ella Fitzgerald.

However, users can also make mistakes that need to be corrected by the agent. An example of this is in the exchange below:

A: Would you like to hear something else from the same album?
S: Eh, yes. Ride On.
A: I'm sorry but that's not on this album.

In a third kind of correction, the subject made a self-correction.

S: What other classical… No, I'm sorry, what other classic rock do you have?

When presented with such an utterance, a good conversational assistant would be able to understand the correction, and act only upon the user's final request.

Challenge 4: *Different world models*

Several conversational challenges occur when the parties participating in the conversation have different world models. One type of model mismatch is when the user's classifications don't match those of the interface. For example, in one Wizard of Oz experiment, the subject expressed surprise upon learning that k.d. lang was classified as a country singer (in addition to being a member of other genres too).

The participant's expectations of how to interact with the assistant can also differ from the assistant's true interaction model. This is apparent in the following dialog from our Wizard of Oz study:

A: OK, would you like to hear another rock album?
S: Yes, please.
A: <long pause> Would you suggest one?
S: Sorry, I thought you were going to give me a choice.

In the exchange above, the subject expected to be presented with a list of rock albums, while the agent waited for the subject to present a particular request. Human beings can easily recover from such mismatched expectations, and such flexibility needs to be incorporated into agent-based interfaces too.

When the user ignores the current dialog state, and asks a totally different question, this is another case of differing world models. In the example below, the subject ignored the assistant's question in posing his final request:

S: You know, I'm in the mood for something else.
A: What genre?
S: Do you have anything by the Eagles?

If an agent-based interface is to handle such interactions, it needs to be able to transcend its momentary expectations, and recognize a reasonable request at any time.

Another way in which the subject's world model may differ from the assistant's is in the belief of whether the constraints established earlier by the subject still hold. This is evidenced in the exchange below:

S: OK, anything from the 40's or 50's?
A: We have… (many selections listed)
S: Do you have anything by women?
A: Certainly. In the 40s and 50s?
S: Uh.

Here, the assistant was not sure whether or not the earlier request for music from the 40's or 50's also applies to the subsequent request for music from female artists. In this case, the subject seemed to be confused about this too!

There are many other cases where the world models of the subject and assistant could differ, leading to dialog difficulties. For example, sometimes the subject thought there was a unique response to its request when there was not (for example, when there were multiple recordings of a

particular title). Other times the subject made a request that was outside of the capabilities of the assistant. Human assistants are good at resolving the different world models through additional dialog. Good agents should be capable of this as well.

Challenge 5: *Ambiguities*

Frequently, ambiguities in the subjects' utterances needed to be resolved. For example, one subject asked to fast-forward beyond "this part", but the assistant did not know whether "this part" referred to part of the song or the entire song. In some cases, a participant requested a title that was both the name of a song and the name of an album. It was unclear to the assistant which should be played, but the human assistant resolved this ambiguity by asking an additional question. One of our favorite exchanges in the experiment appears below.

A: OK, <pause>, we have Peter Gabriel.
S: So?

Here, the subject's response to the assistant's statement could be construed in two ways: as perhaps a statement of disinterest in Peter Gabriel's music, or a query about a particular album (the album, *So*.). Here, the latter interpretation was accurate, but fortunately the assistant, through further conversation could disambiguate this.

Numerous other ambiguities were found in the Wizard of Oz dialogs. One person requested music from Chicago. He was talking about the city rather than the band. The musical domain is so challenging in part because band names and song titles can be just about anything. In the dialog below, the song title could have been a point of confusion.

S: Um...I'd like to hear...uh, I, uh...America? I like it here in America?
A: One moment.

Here the user was requesting a song from *West Side Story* ("I Like It Here in America"), and given the context that *West Side Story* was mentioned earlier in the conversation, the assistant knew what the subject really wanted. However, without any domain knowledge or context, the subject's utterance could have been interpreted as a simple statement. Of course, building this kind of common sense into computers is a grand challenge.

Challenge 6: *Non-verbal information*

People communicate with each other verbally, but they also communicate with each other without words. In our experiment, subjects sometimes communicated with the assistant (who was behind a one way mirror) by shaking their head or nodding, and by giving proposed selections a thumbs-down or thumbs-up. One participant used non-verbal information in the following exchange:

A: We have a very good album, Ella Fitzgerald Live. We have Stan Getz. We've got Miles Davis…
S: Mmmm, mmmm! <and gestures>
A: Miles Davis?
S: Yes, please.

Here, the subject grunted and pointed in response to hearing Miles Davis' name. In this case, the grunting and pointing was due to the subject having a mouth full of soda. But a human assistant was able to interpret the odd form of communication correctly, and when the subject was queried, he verified this to be so.

Most forms of non-verbal communication exhibited in the experiments suggest that computer vision must be an important capability of future conversational agents, if such agents will truly mimic human communications.

Challenge 7: *Disfluencies and ungrammatical utterances*

Spoken speech is often not entirely grammatical, and hence can be harder to understand than written language. People often litter their speech with "uh"s and "um"s, which can be difficult for speech recognition software to recognize as being distinct from more meaningful words in a sentence.

Often people will put *false starts* in their speech, which are the beginnings of utterances that remain uncompleted, and are overridden by the remainder of the sentence. Examples of this appear in the first and last utterances below.

S: Put on the last one, and then … and then let … surprise me with two more things.
A: One moment.
S: This sounds like the last one. What's this?
A: I thought you asked for the last one.
S: Oh. No. I wanted to … I wanted you to just make a list, and I wanted you to play something new.

Here the false start actually led to a misunderstanding on the part of the assistant. The subject intended to correct his false start, and ask for two surprise musical selections. However, the assistant did not realize that the subject was correcting or altering his request in the middle.

Occasionally the words that the subjects strung together were totally ungrammatical. Yet, in most cases the human assistant was intelligent enough to understand the intended meaning, or through additional dialog with the subject, determine what the subject was requesting.

S: Do you have... uh.. it's the new age song. George Winston. uh, I forget the name of it.

The request above is highly ungrammatical, yet it did not bring a halt to the dialog, and the assistant was able to continue to help the subject with his musical requests.

Challenge 8: *Partial information*

Often the subjects did not provide all of the information necessary to satisfy their requests – at least not initially. It is the assistant's job to find out the rest of the information through additional dialog. In the case of conversational interfaces, the dialog manager component must determine which information is missing, and through conversation, fill in the additional pieces.

S: Ummm I think I'm in the mood for something classical.
A: OK, anything in particular?

In the dialog above, the user requests some classical music, but the assistant has numerous classical pieces that he could play. So, the assistant prompts for a particular selection.

This particular pattern of receiving partial information, and turning it into a complete request, is one of the most necessary skills of a conversational interface. Over and over again in our Wizard of Oz experiments we saw the subject and the assistant working together to piece together a complete musical request.

S: Well, I'd like to hear umm that song that was from the Beatles. That Joe Cocker sang.
A: Which song?
S: Let's see. He sang more than one? Which songs did he sing that were by the Beatles? Was there another one?
A: She Came in through the Bathroom Window, and With a Little Help from My Friends.
S: Oh, yeah, She Came in through the Bathroom Window.

In the above dialog, the subject thinks he is asking for a specific song. The assistant knows that multiple songs satisfy the request, and works with the subject in order to track down the desired one.

In this section we discussed eight challenges for dialog management, as determined from Wizard of Oz experiments with human subjects and assistants. Although the experiments were in the domain of music selection, these challenges are universal problems for dialog management in conversational interfaces. While the Peedy prototype did have limited support for resolving inaudible speech, and handling partial information, its dialog management was much too primitive for general use. Dialog management is a difficult problem, and one of the grand challenges in building conversational interfaces. Progress in this area will enable a major step forward in human-computer interaction.

ACKNOWLEDGMENTS

Tim Skelly created the character of Peedy, and through his animation skills, brought Peedy to life. David Thiel gave Peedy a voice and an audible environment. Gene Ball designed much of the Persona architecture. David Pugh, Andy Stankosky, and Maarten van Dantzich built Peedy's animation environment. Dan Ling directed the project. Microsoft Research's Natural Language Group and Speech Recognition Group also contributed components and expertise to the Persona system.

REFERENCES

1. Ball, G., Ling, D., Kurlander, D., Miller, J., Pugh, D., Skelly, T., Stankosky, A., Thiel, D., van Dantzich, M., and Wax, T. Lifelike Computer Characters: The Persona Project at Microsoft Research. http://www.research.microsoft.com/ui/peedycha.doc.
2. Bandai Corporation. Tamagotchi. http://www.bandai.com/tamagotchiaria.index.shtml.
3. Kurlander, D., and Ling, D.T. Planning-Based Control of Interface Animation. *CHI '95 Conference Proceedings*. May 1995. pp. 472-479.
4. Maulsby, D. The Turvy Experience: Simulating an Instructible Interface. Chapter 11. In *Watch What I Do: Programming by Demonstration*. Allen Cypher (ed.). MIT Press. 1993.
5. Perlin, K., and Goldberg, A. Improv: A System for Scripting Interactive Actors in Virtual Worlds. *Proc. SIGGRAPH '96*. pp. 205-216.
6. PF Magic. Welcome to Petz. http://www.petz.com.