# Universal Human-Machine Speech Interface: A White paper

Roni Rosenfeld, Carnegie Mellon University

Dan Olsen, Brigham Young University

Alex Rudnicky, Carnegie Mellon University

CMU-CS-00-114

March 2000

(This is a revised version of a May 1998 unpublished manuscript by the same authors, written while the second author was at Carnegie Mellon University)

## Abstract

We call for investigation and evaluation of universal paradigms for human-machine speech communication.

The vision driving us is ubiquitous human-machine interactivity via speech, and increased accessibility to technology for larger segments of the population. Speech recognition technology has made spoken interaction with machines feasible; simple applications have enjoyed commercial success. However, no suitable universal interaction paradigm has yet been proposed for humans to effectively, efficiently and effortlessly communicate by voice with machines.

On one hand, systems based on natural language interaction have been successfully demonstrated in very narrow domains. But such systems require a lengthy development phase which is data and labor intensive, and heavy involvement by experts who meticulously craft the vocabulary, grammar and semantics for the specific domain. The need for such specialized knowledge engineering continues to hamper the adoption of natural language interfaces. Perhaps more importantly, unconstrained natural language severely strains recognition technology, and fails to delineate the functional limitations of the machine.

On the other hand, telephone-based IVR systems use carefully crafted hierarchical menus navigated by DTMF tones or short spoken phrases. These systems are commercially viable for some applications, but are typically loathed due to their inefficiency, rigidity, incompleteness and high cognitive demand. These shortcomings prevent them from being deployed more widely.

These two interaction styles are extremes along a continuum. Natural language is the most effortless and flexible communication method for humans. For machines, however, it is challenging in limited domains and altogether infeasible otherwise. Menu systems are easy for computers and assure the best speech recognition performance due to their low branch-out factor. However, they are too cumbersome, rigid and inefficient to be widely accepted by humans

The optimal style, or paradigm, for human-machine communication arguably lies somewhere in between: more regular than natural language, yet more flexible than simple hierarchical menus. The key problem is to understand the desired properties of such a style.

We have analyzed human communication with a variety of machines, appliances, information servers and database managers, and plan to propose and evaluate a universal interface style. Such a style consists of a metaphor (similar to the desktop metaphor in graphical interfaces), a set of universal interaction primitives (help request, navigation, confirmation, correction etc.), and a graphical component for applications afforded a display. Extensive user studies will be conducted to evaluate the habitability of the proposed interface and the transference of user skills across applications. In addition, a toolkit will be created to facilitate rapid development of compliant applications, and its usefulness will be empirically assessed.

# 1 Introduction

## 1.1 Vision

Our vision is ubiquitous speech-based human-machine interactivity for the entire population in a 5-10 year timeframe.

By "machine" we mean not only computers in the lay sense of the word, but also any gadget, appliance or automated service which, in order to be fully utilized, must be reconfigured, controlled, queried or otherwise communicated with. We are surrounded by dozens of such machines today, and hundreds more will undoubtedly be developed in the near future. Examples of such interactivity include:

- Configuring and using home appliances (VCRs, microwave and convection ovens, radios, alarms…)
- Configuring and using office machines (fax machines, copiers, telephones…)
- Retrieving public information (e.g. weather, news, flight schedule, stock quotes…).
- Retrieving and manipulating private information (e.g. bank or other accounts, personal scheduler, contact manager, other private databases).
- Handling asynchronous communication (voice, email, fax).
- Controlling miscellaneous user and consumer applications (map following, form filling, web navigation).

We focus in this document on "simple machines", where the user can, at least in principle, possess a mental model of the machine's capabilities and of the machine's rough state. Further, the user is assumed to know ahead of time what they want to do, although they do not have to know *how* to get it done. Under this paradigm, high-level intelligent problem solving is done by the human; the machine is only a tool for getting needed information, modifying it, and/or issuing instructions to the back-end.

In particular, the approach we are proposing is not aimed at applications requiring truly intelligent communication. Thus it is not meant for man-machine collaborative problem solving or, more generally, for intelligent agents. We view these applications as very important and promising. However, they require significant research on "deep NLP" and other AI-hard problems, which is already carried out in many places. Our goal here is to only address communication with "simple machines", which we consider the proverbial low-lying fruit. Thus, for example, an air travel reservation system will fall under our purview only if the machine is used to consult flight schedules and fares and to book flights, while actual planning and decision making is done by the user. The machine plays the role of a passive travel agent, who does not do much thinking on their own, but mostly carries out the explicit requests of the user. The more intelligent travel agent, however desirable, is outside the scope of this discussion.

All the examples above are, according to our definition, "simple machines". However, in all these examples, the capabilities that were designed (or could be designed) into the machine far exceed our current ability to take advantage of them, for all but the most sophisticated and experienced users. For ubiquitous interactivity to become a reality for the entire population, our method of interacting with machines must be fundamentally re-engineered. Any solution must address the following:

1. **Reduce the cognitive load on the user**. VCRs' myriad features are not currently being used because they require significant cognitive effort to configure and activate. Some people cannot master VCR programming no matter how hard they try. Others are able to do so, but at a significant investment of time and effort. Such effort cannot reasonably be expended on each of the dozens of machines we need to interact with in our daily lives. The total sum of cognitive effort required to interact with tools in our environment must be matched to our abilities as humans. Such abilities cannot be expected to grow significantly in our lifetime.

2. **Reach out to the entire population.** The information revolution increased the gap between the haves and have-nots, who are increasingly becoming the *knows* and *know-nots.* For disenfranchised groups such as inner city youth, and for others who were "passed by" the information revolution, being able to use the growing number of automated tools is a last chance to keep up with a technology literate society.

3. **Interactive form factor.** A solution to universal and ubiquitous interactivity must scale to any size or shape of device. This includes anything from houses to wristwatches. Physical widgets for interactivity such as buttons, keyboards and screens cannot be arbitrarily scaled down in size because of human form factors. In addition, many machines are used in mobile settings, where they are hard to reach, or where the user is already engaged in other activities (e.g. driving). In many such settings, a screen and keyboard are impossible to install and/or use.

4. **Technology must be cheap.** For commercial viability, the cost of a solution must be considered. In a mass market dominated by manufacturing cost, the cost of the physical hardware necessary for achieving interactivity must constitute only a small fraction of the total cost of the device. Here again, physical widgets such as buttons, keyboards and screens are not expected to come down in price to the same extent as computing power and memory. Moore's law simply does not apply to such hardware. Simpler hardware will be cheaper, and thus preferable. Further reduction in cost can be achieved via uniformity of hardware. Similarly, design costs can be reduced via uniformity of both hardware and software: with thousands of applications, extensive design and development efforts cannot be supported separately for each application.

We submit that interactivity via speech is uniquely suited to addressing the above requirements, and is thus uniquely positioned for achieving our vision.

## *1.2 Why Speech?*

When considering technologies that might meet the needs of ubiquitous interactivity, the issues of situation, cost, breadth of application and the physical capabilities of human beings must be considered. When we talk of embedding interactivity into a wide variety of devices and situations, the keyboard and mouse are not acceptable interactive devices. In any situation where the user is not seated at a flat surface, a keyboard or mouse will not work. In any application with any richness of information, a bank of buttons is unacceptably restrictive. In a large number of cases only speech provides information rich interaction while meeting the form factor needs of the situation.

If we consider the exponential growth in processing and memory capacity, it is clear that any interactive solution relying primarily on these technologies will over time become very small and very very cheap. Speech interaction requires only audio I/O devices which are already quite small and cheap, coupled with significant processing power which is expected to become cheap. No such projections hold for keyboards, buttons and screens. Visual displays have made only modest gains in pixels per dollar over the last 20 years and no order of magnitude breakthroughs are expected. Visual displays are also hampered by the size requirements for discernable images and by the power required to generate sufficient light energy. Buttons are cheap but are also restricted as to size and range of expression. Any richness of interaction through the fingers quickly becomes too large and too expensive for ubiquitous use. Only speech will scale along with the progress in digital technology.

Spoken language dominates the set of human faculties for information-rich expression. Normal human beings, without a great deal of training, can express themselves in a wide variety of domains. As a technology for expression, speech works for a much wider range of people than typing, drawing or gesture because it is a natural part of human existence. This breadth of application is very important to ubiquitous interactivity.

## *1.3 State of the art in speech interfaces*

Speech interfaces are only beginning to make an impact on computer use and information access. The impact thus far has been limited to those places where current technology, even with its limitations, provides an advantage over existing interfaces. One such example is telephone-based information-access systems, because they provide a high input-bandwidth in an environment where the alternative input mode, DTMF, is inadequate. We believe that speech would achieve much higher penetration as an interface technology if certain fundamental limitations were addressed. In particular:

- Recognition performance
- Accessible language (for the users)
- Ease of development (for the implementers)

That is, it should be possible for users to verbally address any novel application or artifact they encounter and expect to shortly be engaged in a constructive interaction. At the same time it should be possible to dramatically reduce the cost of implementing a speech interface to a new artifact such that the choice to add speech is not constrained by the cost of development.

We consider three current approaches to the creation of usable speech systems. First, we can address the problem of accessible language by allowing the user to use unconstrained natural language in interacting with an application. That is, given that the user understands the capabilities of the application and understands the properties of the domain in which it operates, he or she is able to address the system in spontaneously formulated utterances. While this removes the onus on the user to learn the language supported by the application, it places it instead on the developer. The developer needs both to capture the actual observed language for a domain (principally through Wizard of Oz collection and subsequent field trials) and to create an interpreting component to the application that maps user inputs into unambiguous statements interpretable by a back-end. Examples in the literature of this approach include ATIS [Price 90] and Jupiter [Zue 97] and it is commonly used in commercial development of public speech-based services.

In addition to placing the burden of ensuring usability on the developer, this approach also guarantees the creation of a one-off solution that does not produce benefits for the implementation of a subsequent application (by this we focus on the language and interpretation components of the system). At the same time it does not appear to produce any transferable benefit for the user. Even if there is learning in the context of a particular application (say through the modeling of spoken utterance structure [Zoltan-Ford 91]) there is no expectation that this learning will transfer across applications, since development efforts are not systematically related. A second problem with this approach is that it does not systematically address the issue of constraining language with a view to improving recognition performance; the only improvement in recognition performance is through the accumulation of a domain-specific corpus.

As an alternative to allowing the user to speak freely (and compensating for this in the parsing and understanding component of the system) we can constrain what the user can say and exploit this constraint to enhance system performance. We consider two such approaches, *dialog-tree* systems and *command and control* systems.

Dialog-tree systems reduce the complexity of recognition by breaking down activity in a domain into a sequence of choice points at which a user either selects from a set of alternatives or speaks a response to a specific prompt (such as for a name or a quantity). The drawbacks of such systems, from the user's perspective, center on the inability to directly access those parts of the domain that are of immediate interest, to otherwise short-circuit the sequence of interactions designed by the developer. A space with many alternatives necessarily requires the traversal of a many-layered dialog tree, as the number of choices at any one node will necessarily be restricted. From the designer's perspective such systems are difficult to build, as it requires being able to break an activity down into the form of a dialog graph; maintenance is difficult as it may require the re-balancing the entire tree as new functionality is incorporated. While dialog-tree systems may be frustrating to use and difficult to maintain, they do simplify the interaction as well as minimize the need for user training. What this means is that the user's contribution to the dialog is effectively channeled by the combination of directed prompts and the restricted range of responses that can be given at any one point.

Command and control interfaces reduce complexity by defining a rigid syntax that constrains possible inputs. The language consists of a set of fixed syntactic frames with variable substitution (for example, "TURN VALVE <valve-id> TO POSITION <position-value>"). In a restricted domain, such a language provides the user with sufficient power to express all needed inputs. At the same time the recognition problem is simplified since the language is predictable and can be designed to avoid confusion. The utility of such interfaces depends on the willingness of users to spend the time to learn the language for such a system. It is further predicated on the assumption that once learned, the language will be used on a regular basis as part of daily activity, for example to control a desktop environment on a computer or for process control in an industrial setting. The drawback of command and control systems stems from the investment that the user makes in learning the language. Being able to communicate with additional applications requires that this effort be duplicated. This may be feasible for a few applications but it is unlikely to scale to tens or hundreds of applications, which is what would be required of an interface that permits ubiquitous access to machines. It also suffers in comparison with dialog-graph based systems in that this interaction style is not inherently self-explanatory; the user is not guided into the correct input style by the structure of the interface.

## 1.4 Lessons from the GUI revolution

In seeking a solution for effective development of spoken language interfaces we can learn from the history of graphical user interfaces. The current state of research into speech interaction and its supporting tools is very similar to the state of research into graphical user interfaces in the early 1980s. At that time it was assumed that tools for constructing graphical user interfaces must remain very general in their capabilities. A primary goal was to provide maximum flexibility for designers to create and implement any interactive dialog. It was clearly stated that tools should not introduce bias as to the types of interactive dialogs that should be developed [Newman 68, Jacob 82].

The commercial advent of the Apple Macintosh changed all of that thinking. The Macintosh clearly showed that wide variability in interactive dialog was harmful. A fundamental maxim of the Macintosh was its style guide and the toolkit that supported it. The shift was away from all possible interactive styles to uniformity of interactive style. When menus, scrollbars, buttons, dragging and double-clicking all work in the same way, the ease of use is much greater than when these items are uniquely crafted for each situation. Some major keys to the Macintosh success are: 1) once a user learns the basic alphabet of interactive behaviors, those behaviors can be transferred to almost any Macintosh application, and 2) when faced with a new Macintosh application the set of things to try is very clear. This transference of experience and the presentation of a clear space of exploration as a way to master new applications has been wildly successful in graphical user interfaces, and has come to dominate virtually all interactive computing.

The uniformity of the Macintosh style also has impact on the way in which interactive applications are developed. Instead of creating tools that supported general dialogs of any sort, a widget-based strategy dominated. In the widget strategy the paradigm is prebuilt, pretested pieces that are assembled to address the needs of a given application. These pieces are parameterized so designers can fill in the application-specific information. The widget strategy for development has been extended to the notion of interactive frameworks. In such frameworks, developers are provided with a complete skeleton application with all standard behaviors predefined. The particular needs of the application are then fitted into this framework. This tool and implementation strategy not only simplified development but also reinforced the uniformity of applications.

The key insight of the Xerox Star/Macintosh revolution is that usability is a global rather than a local phenomenon. Users do not use one or two applications in the course of their work. They use many different applications and are repeatedly faced with learning new ones. Furthermore those applications must continually work together to create a productive whole. Uniformity in the basic interactive alphabet increases the transference of learning from one application to another and enhances the overall usability of the entire interactive environment.

Speech interaction can learn further from our experience since the introduction of GUI interface standardization. When originally introduced, a standardized “look and feel” was presented as the key to learning transference between applications. Over time, however, we have learned that a uniform look (visual appearance) is not as important as a uniform feel (input behavior). Users readily adapt to scroll bars or buttons that have different stylistic appearances. They do not adapt as readily, however, to differences in feel or interactive inputs. A key to making Macintosh users comfortable in the Windows environment was not by changing the appearance of menus but by changing the mouse inputs for menus so that they supported the Macintosh menu behavior. Uniform input behavior is important because the input behavior is invisible and must be remembered. This insight is critical in spoken language interfaces because all interaction is invisible. Uniform structure, terminology and input behavior are critical for an invisible interactive environment to be successful.

## *1.5 Solution: a universal speech interface*

Our solution is thus a universal speech interface (USI) style. By this we mean a standardized "look and feel" (or, rather, "say and sound") across varied applications and domains. There are several components to such a style:

- **A universal metaphor**. A metaphor allows the user to map concepts from a familiar conceptual landscape into a new, less familiar domain. The success of the GUI revolution depended to a great extent on the desktop metaphor. A metaphor for human machine speech communication is even more important because the limited bandwidth of speech severely restricts the cues that can be provided at runtime --- it is important that the user have a good mental model of the application and of how to accomplish various goals within it. Ideally, a single, universal metaphor should be used to cover a wide range of application classes.

- **Universal user primitives**. Many aspects of dialog interaction are universal, i.e. they recur in many applications. The list of such building blocks is quite long: recognition error detection and recovery, barge-in, taking or relinquishing the floor, answering multiple-choice questions, asking for help, navigating etc. There must be standardized ways for the user to express each of these. For a given application, idiosyncratic ways of expressing these primitives may be better. However, the uniformity and the usability that uniformity brings dominate other issues. It is very important that the interactive scaffolding that supports users be uniform. There must be a reliably consistent way in which users obtain help and find out what a particular application can do at any point in time. Language for orientation and navigation must be consistent. Users will need to learn where they can go and what they can do in a new application, but how to get there and how they invoke the possible must be uniform.

- **Universal machine primitives.** Machine primitives are similar to user primitives in that a small set of machine prompt and response types recur in many applications, and in fact constitute the great majority of turns in these applications. Standardizing these machine-side universals will allow the machine turns to be shorter and clearer. This could be accomplished by using appropriate common phrases, by making use of prosody, and by packing the audio channel with non-verbal information (beeps, melodies and other earcons). For work in this direction, see [Kamm et. al., 1997].

- **A universal display style.** Human machine speech communication is appropriate for displayless environments as well as environments where a display, however small, may be feasible. Even a small display has significant advantages over voice output, and should be used whenever possible. In these cases, voice output can both complement and reinforce the visual output. Ideally, the style of voice interaction should be the same in displayed and displayless applications, and especially in the displayed and displayless versions of the same application. Doing this will reinforce the universal metaphor and reduce the cognitive load on the user. In addition, mutually unified audio-visual output can help users to quickly develop mental models of specific applications, and build associations between the visual and audio output. The user can then move to rely less and less on the display, until it is completely eliminated.

It is important to note that *these components by themselves do not dictate the entire syntax of the interaction*. It is possible for different applications to conform to the above universals, while still having different syntax, or even specifying no particular syntax at all, thus allowing otherwise unconstrained language.  It is an open and important question whether the syntax of interaction should also be standardized, and if so, to what extent.

The goal should thus be the design of a USI whose help structure, explanation facility, navigation and confirmation/clarification dialogs are uniform across all applications. We believe this approach will lead to sharply reduced development effort in creating new speech-based applications and will simplify transference of learning among applications. Such an approach will also reduce or eliminate the ambiguity in what the users intend. The development of a universal style also leads directly to the development of tools for creating interfaces within that style.

One of the primary contentions in favor of unconstrained language interaction is that there is zero learning time because all human experience transfers to an application that understands natural language. In practice there is no natural language experience that has zero learning time. All human-human situations require time to acquire terminology, shared history, shared goals and relationships of trust. Human beings regularly find themselves in situations where they do not understand and cannot communicate effectively even though all parties are speaking the same language.

More importantly, unconstrained language does not inherently eliminate communication difficulties.  Because the knowledge and capacities of human beings are so rich, we require rich language structures to build up the shared understandings that support natural language. When communicating with “dumb machines”, such rich mutual understandings with their shades and nuances of meaning are not only not required but in many cases are an impediment to getting service. Simplifying the language and unifying its structure will reduce these problems.  Simple machines as defined earlier are fundamentally limited in the services that they can perform. We want a system that will readily and efficiently communicate those limits, so that users can make their own judgements about effective usage. We expect predictable, efficient subservience from our simple machines, not independent behavior and thought.

The use of a universal interface will greatly reduce the development time of spoken language interfaces. Much of the interaction can be standardized and included in an application framework. This limits the difficulty of producing new speech applications. To the extent that we reduce costs and shorten the time required to develop a new application, we empower the creation of thousands of speech applications rather than tens.

The use of the universal interface also simplifies the speech recognition problem. Even if users are free to use a variety of linguistic constructs, the uniform treatment of the universal primitives will result in a substantially lower expected branch-out factor for the user’s language, with a direct impact on recognition accuracy.

We believe that such a USI is possible, and will be effective precisely because of the restricted nature of computing technology. Devices and applications function in terms of their information state. Communicating that state to the user and allowing that user to modify the state is fundamental to such interactive situations. Any interaction style that does this clearly, effectively and uniformly will to some level be successful as a USI.

It should be noted that the USI approach to spoken language interfaces is a departure from the natural language-based approaches taken by others. The USI approach assumes that, at least for simple machines, the advantage of speech lies not so much in its "naturalness" but in the interactive contexts where screens, buttons and other more traditional interaction forms do not apply. There is a contrary hypothesis to ours, which states that natural language-based speech will be more effective than a uniform style. The design and evaluation of USIs will help answer that question empirically rather than by hypothetical debate.

## 2 General approach

The challenge is thus to develop, test and disseminate a Universal Speech Interface for human-machine communication. Such a USI must be effective and efficient. It must impose low cognitive demands on the user. Finally, it must support all the application types we listed, and enable transference across them. In developing such a style, one may rely on the following principles:

- **Moderate and incremental user training.** Training time and effort will of course be amortized across all applications. Even several hours' worth of training will therefore be well justified in the long run (this is still less than the amount of training needed for keyboard typing). For mass dissemination, though, the barrier to entry must be substantially lower. In order to entice novice users to start using USI-compliant applications, learning the USI core should take only a few minutes. This core should make the user immediately productive. Further training will be accomplished incrementally, as the need arises, using the dynamic self-help facility described below. To achieve this goal, the design must be very intuitive, conforming to people's prior expectations. In other words, it should resemble natural language.

A good example of this approach can be found in the Graffiti™ character writing system currently deployed with the PalmPilot PDA. Graffiti is an invented "language", which was engineered to optimize automatic handwriting recognition. It bears strong resemblance to the way people naturally write letters, digits and punctuation marks. The original, natural style of writing was modified minimally, only when it was prudent to do so for ease of recognition. As a result, it takes only a few minutes to become a productive Graffiti writer. Then, a built-in help facility ensures that productivity continues to improve with use.

Once a user understands the most basic USI concepts, a comprehensive built-in help becomes the main tool for speeding up learning. Another tool to facilitate incremental on-the-job learning is a "cheat sheet", namely a compact and concise written summary of the main features of the USI. Such a summary could be printed on a small sticker and placed in a wallet or on the device. It could then be consulted as needed, the intention being that it be consulted less and less with time. When a display is available, the coordinated visual output mentioned earlier is an even better aid.

- **A standardized way to accomplish each task.** A compact design will eliminate possible confusion and enable a clear path from the user's intention to the system's action. Taking example from Graffiti again, there is a standardized "correct" way to write each character. In contrast, Apple Computers' Newton™ handwriting recognition system attempted to handle each user's idiosyncratic writing styles. Instead of leading the user to high accuracy uniformity, the system tried to adapt to the user – an ostensibly more difficult task. As a consequence, the Newton's recognition performance was inadequate.

- **Factor in the limitations of the Speech Recognition (SR) technology.** SR technology has made dramatic progress in the last decade, and more progress will no doubt be achieved in the near future. Nonetheless, recognition errors will continue to occur. In designing the USI, One must minimize such errors in several ways. Among comparable lexical alternatives, one should favor the more acoustically distinct ones (interestingly, Graffiti's designers used a similar strategy – character strokes were chosen to maximize distance in feature space). In addition, the USI itself should include the facility to explicitly deal with recognition errors (more on this later).

## 2.1 An Ontology of Interaction Primitives

In considering the form the USI might take we have identified 4 categories of interface discourse which must be accounted for. They include:

- Help/Orientation
- Speech/Audio
- Discourse
- Application

The help and orientation primitives are a key part of the self revealing nature of USI-based applications. These primitives standardize how a user can discover the possibilities and limits of the machine being controlled. When faced with a new graphical user interface a user can explore the menu, investigate any dialog boxes and read the tool-tip messages attached to icons. These strategies for discovery are standard for a given windowing system and are fundamental to the learning of a new system. Knowing standard interface discovery strategies empowers new users of automated services.

The speech and audio primitives deal with the vagaries of machine-recognized speech. There will always be recognition errors and discourse problems having to do with interruption, noise and extraneous speech. The user must have clear expectations of the capacity of the machine and familiar standard strategies for detecting and repairing mistakes.

The discourse primitives deal with the dynamics of an interactive exchange. Users must understand when closure is reached, changes confirmed and actions taken. They must also have clear expectations of how to recover from erroneous commands. Not all errors are the fault of the machine. This is a particular problem if users tend to "think out loud" without having worked out their ultimate goals. A key lesson learned in graphical user interfaces is that the safety of a reliable UNDO and/or CANCEL facility encourages user exploration and thus speeds learning of new systems. The discourse primitives of a USI must satisfy those same needs in spoken language interfaces.

The application primitives are specialized to the actual semantics of the machine the user is attempting to use. These must have a clear and consistent structure. On a Macintosh, users expect to select one or more objects and then perform manipulations on them. The approach may be different for spoken language interfaces, but as with graphical systems the model used must be uniform across applications.

## *2.2 Design Principles for the Machine's Prompts/Responses*

As mentioned in the introduction, most of the machine's prompts and responses can also be classified into a small set of primitives. These include various error conditions, yes/no questions, multiple-choice questions, confirmations, clarifications, etc. A hierarchy can be developed similar to the one mentioned above for the user side. Here we will only offer a few principles for the structure of the machine's output.

- **Standardized universal phrases.** The cognitive effort required of the human to understand the machine's utterances can be minimized by using standardized, universal phrases. These must be terse, yet they must convey all useful information. As users become more and more familiar with these phrases, many of them can be replaced by earcons. A standard "narrowing" process of increasing brevity will speed interaction without losing the user. However, such brevity is only possible if the speaker (machine) and the hearer (user) have similar expectations. Such expectations can only be formed by a uniform structure across many applications.

- **Barge-in consideration.** The user may cut the machine off at any point. The machine's utterances should therefore be designed to convey the most important information up front. Uniform expectations about what the machine may say, and why, can lead to more aggressive barge-in when users know that the information to follow is not what they want. Again this can produce more succinct and effective interfaces.

- **Increased effective bandwidth.** The audio channel has a drastically lower bandwidth as compared with a visual display. Also, unlike a display, audio is a temporally sequential medium: one cannot "browse" an utterance back and forth, random-access it, or review it again after an interruption. For these reasons, making optimal use of the medium's limited bandwidth is crucial. Information can be conveyed by prosody, speaking rate, or the quality of the synthesized voice. In fact, varying pitch and even multiple voices can be used to convey semantic information. For example, all confirmation/clarification dialogs can be carried out using a distinct pitch or distinct voice, different than the one used in the application itself. Also, using the audio channel for speech only is sub-optimal: additional information can be packed into the system's output by using non-speech messages, such as beeps, earcons, music etc.

- **Entrainment.** The machine's prompts should be used to "lexically entrain" the user [Boyce et. al, 1996]. Namely, lexical priming should be exploited by selecting the terms in the machine's prompt to be those that the user should ideally be using. In addition, "syntactic entraining" should also be used. For example, suppose that in a flight information application, the airline, date, time, origin and destination need to be prompted for. Instead of the machine prompting for them one by one, or asking "WHAT ARE THE AIRLINE, DATE, TIME, ORIGIN AND DESTINATION?", a more efficient prompt could be "FOR AIRLINE *** ON DATE *** DEPARTING TIME *** FROM *** ARRIVING AT ***". The "***" symbol denotes an appropriate earcon, perhaps in the form of de-lexicalized speech, which stands for the missing information to be filled in by the user. This style suggests to the user how to phrase their next utterance, exploiting short-term audio memory. If the list is too long, it can be truncated, and ended with "AND MORE". If the user only remembers a few of the fragments, they can specify those and then request additional assistance.

# 3 Scientific Challenges

The scientific challenges in the USI approach fall into three areas: design of the interaction style, learnability and development tools. The design challenge is the development of a USI structure, which subsumes a large space of spoken language applications. The learnability challenge involves basic user experimentation with the system to prove the hypothesis that the universal style paradigm is truly effective. The development tools' goal is to capture the design and experimentation knowledge in working reusable software.

## *3.1 Universal Interfaces Design*

In designing a USI it is necessary to discover and/or invent an interactive style and a logical structure that will be invariant across a large class of applications. In a trivial sense, natural language could fill this role. The key challenge is to identify a subset of natural language that can be parameterized to specific applications. Designing this subset involves the study of a large number of applications and how people might talk about those applications. From this broad sample one must distill down an "inner structure", or core of language capabilities, that will cover the interactive needs of a broad class of applications. This design is also problematic in that the use of generalized style should not make some applications unduly awkward. One could try to get at these issues by user experiments with a USI across a variety of applications. This spoken interface design is a new combination of natural language and artificial specification languages. The universal part must be precise and regular, but the rest of the interaction may remain flexible, perhaps even unconstrained. A USI design would be judged successful if its learnability across several applications is shown experimentally and if it readily lends itself to the construction of development tools for such interfaces.

## *3.2 Learnability*

Key to the success of this approach is testing the learnability of applications embedded in the USI. There are four important aspects of learnability that must be considered. The first is for anyone using the USI for the very first time with no knowledge of it and its constructs. This barrier must be as low as possible. The minimum knowledge required to start using the interface must not stop users from making the initial trial and must afford first-time success. A fundamental question here is whether existing knowledge and intuition will be sufficient for first time users of the USI. This question can only be resolved by experimentation with actual users. One can test such a first time experience by comparing USI-based interfaces with those found on existing appliances such as lawn sprinkler timers, "smart" home thermostats, or home security systems. These are all home devices, which the public has difficulty learning and using. If USI-based interfaces are more easily learned than the existing physical interfaces then we can count a success. Similarly, one can perform analogous comparisons with existing over-the-phone information servers (movie information, weather, flight schedule etc.).

The second learnability question is whether users continue to develop their USI expertise over time and eventually acquire all of the features, or plateau on a very primitive set of capabilities. The user experiments to address this issue will suggest techniques for enticing users to ever-increasing skill levels. The USI must offer a gentle and continuous slope for incrementally acquiring universal skills. User experimentation coupled with redesign will yield answers to this question. The key scientific goal here is to capture the resulting knowledge about user behavior and encode that knowledge into a toolkit and interface design. With this, developers of individual applications need not repeat the expensive experimentation process. Success in this area can be measured by posing tasks that require successively more sophisticated knowledge of USI capabilities. If users over successive trials gain facility with the machine and use more of its capabilities, then these enticement goals have been met. If, however, users consistently miss some capabilities even though the task may call for them, then this portion of the system must be redefined.

The third and most critical learnability challenge is that of transference of interaction skill. The fundamental hypothesis of this approach is that once a user acquires skill in one USI-based application, the acquisition of skill in the second and third applications will be vastly improved. Experimentation on this issue is very important. The Palm Pilot forced all users to learn a new way of writing and yielded user satisfaction that was better than the Newton, which would train to the user and attempted to recognize any reasonable handwriting. The hypothesis here is that a modest amount of training to acquire USI skill, which then transfers to hundreds of other applications, would be worth the effort. User experiments should be conducted which will test this hypothesis in the context of specific USI designs. Such transference of USI knowledge can be tested using successive trials over time with different USI-based tasks.

A fourth challenge is to determine if USI-based applications can be accessed by broad classes of individuals. In particular it is important that USI-based applications be accessible to segments of the population that currently are not well served by information technology. It is obvious that telephone technology is accessible to many more people than Internet access devices. The key research question is whether a learned, universal interaction style imposes educational or intellectual barriers that preclude access by members of disenfranchised groups, or whether it actually entices and facilitates such access.  A separate interesting question is how easy it will be for children to learn such a style. One way to test this aspect of the design is to create telephone-based services using USI. Advertising the services broadly and then monitoring usage and errors will provide the necessary data to validate claims of learnability by broad populations.

## *3.3 Development Tools*

If spoken language interfaces rely on a universal style, then it becomes possible to create a more powerful set of application development tools. These tools would automatically provide the scaffolding to handle the universal primitives and provide the self-explanation facilities that these interfaces should have. It is also key that the results of the extensive user testing and experimentation involved in this research be encoded into the toolkit. The goal is not to produce extensive guidelines for USI-compliant design, but rather to generate such interfaces automatically for new applications. Technologies for generating languages for graphical user interfaces were explored and discarded with the dominance of direct manipulation [Lieberman 85, Olsen 86, Olsen 89, Wiecha 90]. Such ideas can be reapplied in this new context. Using such a toolkit, developers need not perform all of the user experimentation on grammar, help systems, or learning of the interface structure. These features will be automatically supported in the toolkit. The application developers can focus exclusively on the functionality and usability of their services rather than on the design of spoken languages. The embodiment of experimental results in a toolkit which is extensively distributed is key to their wide use and acceptance.