LINGUA MACHINAE - An Unorthodox Proposal

Florian Schiel, Christoph Draxler, Marion Libossek

Bavarian Archive for Speech Signals
Ludwig-Maximilians-Universität, München, Deutschland
{schiel, draxler, marion.libossek}@bas.uni-muenchen.de

Abstract

Voice User Interfaces (VUI) are slowly emerging into today's IT applications. A decade ago, members of the speech science community (as well as market analysts) predicted a much faster growth of commercial VUI usage and are now wondering why the acceptance was so moderate during the last five years. One reasons for this is certainly the difficult economic situation since the crash of the new technology market. Another reason is the fact that people are slow, especially in the light of an uncertain economy, to leave well-trodden paths for new shores. In this paper we raise the unorthodox question whether VUIs might work more efficiently, be better accepted by users and have a greater commercial impact on future IT markets if there existed a widely agreed communication paradigm on how to use VUIs. For the moment we'd like to call this paradigm 'Lingua Machinae'[1].

Index Terms: Lingua Machinae, dialog system, VUI design, communication paradigm

1. A Hypothesis and its Counter-Arguments

"Humans learn spoken language implicitly at a very young age rather than through explicit education. In contrast, most other user interfaces depend on specific learned actions designed to accomplish the task at hand (e.g. choosing operations from a toolbar, dragging and dropping icons). *Therefore, the VUI designer must work on the user's terms, with an understanding of the user's conversational conventions. As designers, we don't get to create the underlying elements of conversation.*" (from M.H. Cohen, J.P. Giangola & J. Balogh, "Voice user interface design" [1], p. 8, emphasis added by the authors)

If Cohen et al. were right, a 'Lingua Machinae' (LM) that is a paradigm of how to use a Human Machine Interface (HMI), would per definition have to be the human way to interact. But our question to Cohen et al. is quite simply: 'Pray: what human way would that be?'

We challenge that the above formulated hypothesis of Cohen et al and of some other authors (e.g. [2], [5]) is probably not true. Here are our counter-arguments:

1. If the statement of Cohen et al is true, why is it that most people interacting with a machine interface instinctively start to use a special register which they think is appropriate for interacting with a machine? The main problem of very well designed HMI systems nowadays is that users do not communicate with it in a 'natural' way, thus thwarting all the tremendous efforts that designers have put in the development process. On the contrary: speakers start using all kinds of 'unnatural' registers which they had experiences with in other situations. Some users start talking in single commands only, some suddenly speak with more emphasis and with more precise pronunciation than usual, some people adopt a strongly reduced set of syntactic rules (speaking only in infinite verbs for instance) and some people even try baby talk.

2. If the statement of Cohen et al were true, we would expect that speakers behave extremely conservative and stick to one speaking style ('register') in all different situations of communication. Everybody who has studied real human communication knows that this is not true: speakers change their registers all the time and completely effortlessly, sometimes even in mid-sentence. The average speaker has 4-5 different registers in daily life: business talk, flirting, speaking to infants, speaking to a customer, small talk to strangers, small talk to friends, buddy talk (in most cases strongly dialectal), chick talk, all kinds of hobbies, talking to animals, body language to foreigners to name just a few. Humans are champions in creating and maintaining a special new register, so why should they have problems with adopting a new one to communicate with their car?

3. Humans not only use conventions, they love them! In every form of communication humans feel better if they know exactly which communication rules are appropriate for this particular situation. Average speakers become very stressed if they are put into a situation where they do not know how to interact. So, why should it be wrong to conventionalize the communication with an IT system?

4. If the statement of Cohen et al were true, the only satisfying spoken HMI interface would be a system that passes the Turing Test. Is this really what we want? Systems that do not show signs of being a non-human artificial system? In informal talks colleagues in the field of VUI often stress the importance of being aware all the time of speaking to a machine for ethical reasons (Cohen et al state a similar opinion in [1]).

5. There are experiences from other HMI interfaces we can learn from: In the 90s Apple developed the Newton, a very advanced handheld PDA exclusively using pen input. Apple put a lot of effort into the Newton to design the hand writing recognition engine as good as possible. The specifications required that users may use their personal hand writing without any restrictions. The arguments for such

[1] Although we'd like to point out here that concepts of Lingua Machinae need not to be restricted to linguistic contents.

a difficult task were surprisingly similar to the arguments given by Cohen et al. Experts argued that users were not willing to adapt to the machine. The Newton was one of the biggest product failures of Apple. Market analysts and designers gave a number of different reasons: Too advanced, not working yet, too early for the market etc. Five years later low cost PDAs like the Palm were developed using a predefined hand writing character set and a much more robust and simpler recognition engine. The prognosis of the experts was that the Palm will be an even bigger failure than the Newton before, because users would not accept to adapt their writing style. It turned out that, quite on the contrary, the simpler paradigm was a great success. Users did not mind using a learned communication paradigm, if the technology works with it.

2. Some more Arguments in Favor of an LM

Most of the counter-arguments above already make a point for a 'Lingua Machinae'. Here are some more:

1. Conventions like an LM make communication more effective, thus speeding up the process. Conventions are in some way nothing more than to communicate with certain code books and redundancy protocols. Both methods are used in modern IT techniques simply because they make communication over a disturbed small band channel faster and safer.
2. Conventions like an LM that can be transferred to all kinds of HMI systems lower the 'first-usage-angst-threshold'. We often observe that especially older potential users refuse to use a VUI because they 'are afraid to talk to a machine', 'don't know what to say', 'don't know what the machine understands' etc. A clear and easy to learn convention – possible even structured into several layers of complexity which each form a sufficient and consistent communication protocol – might alleviate these problems.
3. Every HMI system is necessarily limited in its capabilities of communication. Therefore every HMI system has to be 'learned' in some way by its users, and if it is just that the user must explore the possible limits of its capabilities[2]. This exploration phase would probably be much shorter if the user already knew the basic capabilities (= LM) of every HMI system beforehand.
4. Other established communication interfaces did not have their breakthrough[3] before certain conventions were established and widely known. For instance, most people are not aware that the telephone started out with a very different design in the beginning. Nowadays computer users are mostly unaware that there were several different paradigms to interact with a Personal Computer (some of them still exist), before Xerox's desktop paradigm was adopted by Apple and later copied by Microsoft. Granted, these were automatic, 'evolutionary' developments, and one serious advice to somebody babbling about Lingua Machinae might be to sit still and just wait. Of course we could do that – but what might have happened, if 30 years ago the colleagues at Xerox had thought that way?

   We are not pleading to invent *the* Lingua Machinae right here on the spot; we think that this is not possible. But we also think that the coming of the LM is unavoidable. We would like to get people – especially the designers that build current and future HMI interfaces – to think about the possibility to shorten the long and cumbersome process by starting to adopt certain conventions right now and thus maybe to boost the usage of VUIs along the way.

3. How many LMs?

Clearly, one LM will not be enough to cover all aspects of voice communication with machines. Instead, one should consider a layered approach to LM, each layer specifying a context in which the particular LM is used. We suggest the following onion-skin model together with a context specification by interface modalities already available.

The innermost layer consists of conventions for a voice-only user interface, e.g. telephone without even DTMF, the next layer consists of voice-only output and voice input plus a simple clickable interface, e.g. DTMF. The third layer adds a graphical display and programmable buttons or menus. The fourth layer adds pointer movement and dragging, the fifth layer adds mimics and emotion recognition, etc.

All features of an inner LM layer are available to all the outer LM layers. The additional features of the outer LM layers either provide a unique access path, e.g. dragging a graphical element, or an additional access path to features available in inner LM layers, e.g. 'OK' and 'Cancel' keys for the affirmative and negative voice commands.

4. Some Ideas for an Innermost LM Layer

The following list of ideas represents the boiled down essence of an extended brain storming session (5 days) during a workshop at Venice International University in Spring 2005 organized by Florian Schiel. This collection is intended merely as a starting point for a more thorough discussion with members of the speech science community in the future. We are quite aware that many items postulated in the following are debatable, some might simply not work or have been tried out by others already. Also, you might notice that about half of the suggestions here are not linked to a certain command word but a rather design guidelines. The reason for this is that during our extended discussion it turned out that LM cannot be treated independently from such general design principles. Hence, the concept of an LM in the sense of a communication protocol is causing technical requirements in the HMI system.

In the following literal LM keywords are underlined. For the sake of this initial discussion all keywords are formulated for a hypothetical English LM; for other languages the appropriate translations will have to be chosen[4]. To simplify the discussion we took the liberty to enumerate the following LM rules. However, the numbering does not reflect any ranking of the rules per se.

[2] In the SmartKom project, we sometimes observed users who solved the given tasks very quickly and then used their spare time to explore the limits of the system. For instance, one user tried to talk the system into a date this evening. Not being discouraged by the monotone 'Sorry, I do not understand your input.' he tried it seven times until the time of the experiment was over.

[3] Breakthrough in the sense of 'broadly used'.

[4] Please keep in mind that the actual chosen words here are not of great importance; if you have a preferable alternative in mind, please do not hesitate to inform us about it.

4.1. How to initialize a dialog?

Although many VUI designers are talking about keyword initialization, most current commercial systems are started by Push-to-Talk (PTT). Even for such a simple concept we can think of some useful conventions:

PTT1: The user needs to push the PTT only once to initiate the dialog, not for every input.

PTT2: The PTT signal must be acknowledged by the HMI system by a signal (e.g. an earcon or icon).

PTT3: The system must signal that the PTT is again ready to be used by an earcon.

PTT4: Pressing the PTT during the dialog causes the system to leave the dialog state.

However, the optimal (hands free) way to initiate a dialog would be a keyword instead of a PTT:

KEY1: The initialization keyword should have a minimum of four syllables and should unambiguously address the specific system.

KEY2: The reception of the keyword must be acknowledged by the HMI system by a signal (e.g. an earcon or icon).

KEY3: The system must signal in any way that it is ready to receive another keyword.

KEY4: At any time during the dialog uttering a certain 'break' keyword causes the system to leave the dialog state.

4.2. Simple Input

Basic input functions which are deployed by every HMI system should be conventionalized.

- *Confirmation/Rejection* don't necessarily have to be expressed in the most common terms 'yes/no' and their many, many derivatives found in daily spoken language. On the contrary, it might be useful that a system requires rather infrequent terms to indicate the special register that the user is using while communicating with a machine. Also, it would be very helpful to use words with more than one syllable to minimize recognition errors as well as confusion with digits on that basic level (which can be disastrous) even under bad conditions (like heavy noise). We therefore propose the somewhat exotic:

  INP1: affirmative for a confirmation

  INP2: negative for a rejection[5]

  for HMI systems *in all languages*. For practical reasons, we think that these two basic command words should be applied in parallel to the 'yes/no' version of the respective language.

- *Input of numbers*[6]. This is a very basic function and many attempts have been made to make it more safe and simple to use. In our opinion, the best and most robust method can be found, for example, in the current BMW HMI system and is represented by the following rules[7]:

  INP3: **0.)** System clears input register.

  1.) Allow the user to utter as many single digits as he wants (two four nine three ...).

  2.) When there is a silence interval of more than 250 msec the system acknowledges the recognized digits by repeating them followed by the words 'and then?'

  3.1.) The user utters new digits: the system adds the last input digit string into its input register and continues with 1.) or

  3.2.) The user says 'negative': the system deletes the last input digit string and continues with 1.) or

  3.3.) The user says 'affirmative': the system finishes the digit input sequence and processes the input register.

- *Spelling*. Most HMI systems must account for the possibility that the user wants to input a name that is not represented in the database. In this case, the user is usually ask to spell out the word to the system. There are probably several thousand ways to spell an 8-character word in the world languages. Unfortunately, the most basic form of spelling, the use of single characters ('a', 'bee', 'zee', ...) shows the highest confusion errors. Anything else is better than this! Since misspellings can have dramatic effects in certain situations, the military long ago developed standard spelling systems[8]. VUI designers have put much effort into spelling systems to cover the most frequently used way to spell, but soon you'll find yourself at a point where the coverage of the nineteenth way to spell the letter 's' is simply not cost-effective any more. We therefore propose the following rather radical spelling rule:

  INP4: For single character input follow the procedure of INP3. For input allow only the most common spelling system for the respective language (for instance a defined set of first names) or alternatively - in all languages - the NATO spelling alphabet.

4.3. Selections

All HMI systems will present alternative data to the user and ask for the user's choice. In the simplest case, this will be a choice between two possibilities, in the worst case it is a selection from a list whose length is unknown at the time of the VUI design. Depending on the complexity and length of the list items the human short-term memory prohibits longer list presentations.

SEL1: The length of the list must be announced by the system before the list is presented.

SEL2: The selection may be done via the item position in the list ('the third', 'number three', 'the last one', ...), or by repeating the prompted list item itself, or by barge in with the word 'stop'.

SEL3: Silence longer than 2 secs indicates that the user does not want to select any of the presented items.[9]

[5] The common syllables 'ative' may cause possible confusions between the two terms. Any proposals for a better term are very welcome.

[6] Pre-formatted digit strings (e.g. date, time, currency expressions) are not considered as simple input and must be described by a grammar.

[7] Please note: this is not a 'natural' convention; it must be learned!

[8] E.g. the NATO spelling system: Alpha, Bravo, Charlie, Delta, Echo, Foxtrot, ...

[9] The reaction of the system depends on the particular VUI design; it is definitely not recommended to repeat the list.

SEL4: The system may not present any other output after presenting list elements[10]

SEL5: Lists with fewer elements than 5 are presented completely (short list)

SEL6: Lists with more than 4 elements are partitioned into chunks of not more than 4 elements.

SEL7: The last element of a list/chunk is uttered with a raised f0 at the end indicating that the system expects a selection.

SEL8: Longer lists than 3 chunks (16 elements) must be announced by the system and an alternative way to reduce the list size must be offered.

SEL9: The following list commands can be used to navigate in a list:

’pause’: the presentation of the list is paused at least 4 secs

’resume’: the list presentation is continued, either after a ’pause’ or after a chunk.

’repeat’: either the short list is repeated (anytime) or the last chunk is repeated (anytime) or the long list is started again (if the last chunk was completed)

’top’: go back to the top of the entire list.

4.4. Navigation

The following LM rules should provide the user with a simple way to navigate within the dialog system.

NAV1: ’where am I’: the system explains in which state the system is, which information is in the input cache, which information is missing.

NAV2: ’say again’: the last prompt/output of the system is repeated. Note that this is different from:

NAV3: ’undo’: the last (successfully processed) dialog step is repeated; the input cache for that step is deleted/overwritten

NAV4: ’operator’: the user is transferred to a human operator, or the system explains why this is not possible and what other options exist.

NAV5: ’push it’: the current state of the system is stored in memory. The user might perform another dialog and then utter: ’pull it’ and return to the stored dialog state. The system automatically issues a NAV1 command to indicate the restored dialog.[11]

NAV6: ’pause’: puts the system into a waiting state. And ’resume’ terminates the waiting state.

4.5. Help

Most contemporary VUI designs include context-sensitive help at any time of the dialog, often in combination with an increasing level of detail before the user is finally passed to a human operator. Since this is a fundamental part of the individual VUI design, it is probably impractical to enforce certain conventions here. However, as a minimal requirement every system should be able to provide the user with just enough information so that the user may continue the dialog successfully.

HLP1: ’help’: the system explains the current dialog state, how the user can get into the current state (important for ‚undo’ operations), and what options are available.

4.6. Feedback / Turn Taking

Turn taking is a fundamental part of human communication. Most users feel uncomfortable with HMI systems because of the missing feedback. Therefore, the system must at a minimum signal the following state information to the user; preferably with standardized earcons or prosodic gestures (suggestions in square brackets). In general ’short’ (250 msec), pitch-rising, harmonic earcons signal positive feedback, while ’long’ (800 msec), pitch-falling, disharmonic earcons signal problems.

TUR1: System expects input [short earcon, rising f0 in last word of prompt]

TUR2: System estimates that it will take longer than 2 secs to answer/next prompt [permanent monotone earcon]

TUR3: The last user input cannot be processed for technical reasons (no speech input, ASR confidence is too low, out-of-vocabulary words) [long earcon]

These proposals are intended to form a base from where a more thorough discussion might start from. We urge anybody interested in the topic to contact us and participate in the Lingua Machinae mailing list for further discussions (*linguamachinae@bas-services.de*).

5. Acknowledgments

We thank the following colleagues and students for their valuable input and discussions: Dr. Florian Gallwitz (SYMPALOG), Dr. Ludwig Hitzenberger (University of Regensburg), Steffen Alexander, Matthias Mahrhofer, Meral Akyol, Felix Weide, Dr. Elke Kachelrieß (VUI Design), Hannes Mögele, Prof. Hans G. Tillmann (University of Munich), Joanna Rymarska, Thomas Barth, Jan Gorisch, Sarah Himmelsbach, Tanja Ellbogen, Karin Högner, Stefanie Schatz, Klaus Härtel (3SOFT), Chantal Ackermann, Markus Heller, Conny Keinitz, Christoph Ringlstetter, Mariya Vitusevych.

6. References

[1] M.H. Cohen, J.P. Giangola & J. Balogh: ”Voice user interface design”. Addison-Wesley, 2004.

[2] Randy A. Harris: ”Voice Interaction Design”. Morgan Kaufman Series in Interactive Technologies, 2005.

[3] Mike Cohen: ”VUI Design for Advanced Natural Language Understanding: The Art and the Science”. In: VUI Visions - Expert views on Effective Voice User Interface Design; William Meisel, Editor; TMA Associates, California, USA, 2006.

[4] Ryan Bush & Lisa Guerra: ”Yes/No Questions are Simple, Right?”. In: VUI Visions - Expert views on Effective Voice User Interface Design; William Meisel, Editor; TMA Associates, California, USA, 2006.

[5] Blade Kotelly: The Art and Business of Speech Recognition; Addison Wesley, 2003.

[10] To prevent redundant information loaded to the short-term memory of the user.

[11] This is probably hard to implement in existing VUIs.