Language Evolution and Human-Computer Interaction*

Jonathan Grudin
Department of Computer Science
Aarhus University
Denmark

Donald A. Norman
Department of Cognitive Science
University of California, San Diego
La Jolla, CA 92093

June 1993

Abstract

Many of the issues that confront designers of interactive computer systems also appear in natural language evolution. Natural languages and humans computer interfaces share as their primary mission the support of extended "dialogues" between responsive entities. Because in each case one participant is a human being, some of the pressures operating on natural languages, causing them to evolve in order to better support such dialogue, also operate on human-computer "languages" or interfaces. This does not necessarily push interfaces in the direction of natural language—since one entity in this dialogue is not a human, this is not to be expected. Nonetheless, by discerning where the pressures that guide natural language evolution also appear in human-computer interaction, we can contribute to the design of computer systems and obtain a new perspective on natural languages.

Introduction
Four Design Characteristics for Language
1. Language Should Be Clear

*To appear in the Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society. Hillsdale, NJ: Lawrence Erlbaum Associates.

Natural language
Human-computer interaction
2. Language Should Be Quick and Easy
Natural language
Human-computer interaction
The "Law of Least Effort" in human performance
3. Language Should Be Expressive
Natural language
Human-computer interaction
4. Language Should Be Processible
Natural language
Human-computer interaction
Contextual Factors in Language Change
1. Gradual Evolutionary Processes
2. Contact with other Languages
3. Creolization
4. Individual Development
Language Evolution
References

Introduction

A "dialogue" does not require natural language, or even words. Animals engage in sustained interactions that can be characterized as dialogues. A mime is engaged in dialogues with real or imaginary objects and with the audience. Two individuals who do not share a common language can work out a means of communication, perhaps as a step to developing a shared "pidgin language."

In this paper, we address extended human-computer interactions that are "dialogues" in this general sense. We include *all* forms of human-computer interaction, not just "conversational" interfaces to computers. Consider, for example, this sequence of events in discarding a document on a Macintosh. As you move the mouse, the arrow or pointer moves across the display. When the pointer is above the icon that represents the document, you press and hold down the button on the mouse. The icon switches to "reverse video" (interchanging black and white), signalling that you have succeeded in selecting it. You move the mouse and an outline image of the icon moves, indicating

your position and telling you that you remain in control of the document. When the pointer and the outline image reach the vicinity of an icon in the shape of a trash can, that image spontaneously switches to reverse video, signalling that when you release the mouse button, the document icon will disappear and effectively be discarded. You do so, and the sides of the trash can bulge slightly, indicating that the document is inside. The bulging sides signal that there are now documents in the trash can that can be retrieved if desired. This entire sequence can be considered to be a dialogue, although no words are used. The system and you monitor one another; you communicate by mouse movements and button presses, the system communicates by moving objects, switching them to reverse video, making them appear and disappear, and changing weir shape.

An interface designer is really a designer of interaction languages. Computer systems are unique among artificial devices in allowing for a substantive, intelligent interaction between person and artifact. The development of interaction techniques is still in its infancy. Certain design guidelines are widely endorsed with little critical examination, such as "build consistent interfaces." The inconsistencies in natural languages—the naturally occurring, continually evolving communication media used for everyday interaction among people—*have* been analyzed, revealing tradeoffs among competing pressures on dialogue. By contrasting the two domains, by finding where analogues exist and where they do not, we may obtain insight into the nature of computer system design and a richer perspective on the constraints on natural languages.

Below, we examine the "design guidelines" proposed by Slobin (1977) for this rich and complex natural system—human language. Obviously, languages have not been designed; they have evolved over thousands of years subject to numerous competing pressures, including political, cultural, and religious factors. Nonetheless, a natural system such as language has much in common with artificially designed computer systems. Many of the requirements are similar. Each must act as a communication medium to transmit intentions, actions, and results among the participants, each must be learnable by beginners, yet efficient for skilled performers. The ability of naturally evolving systems such as language to deal with these conflicting pressures can be revealing for the design of computer systems.

In this paper we examine the changes in both natural and computer lan-

guages, the latter including high-level interface languages, operating systems, and even programming languages. We restrict ourselves to word choice and form and syntactic structure. Thus, we do not deal with speech acts or over subtleties of language.

Four Design Characteristics for Language

Slobs (1977) has analyzed what we might call the "design characteristics" of language, aspects of the usability and functionality of language that lead toward language development and change and that affect the ease of acquisition by children. He identifies four constraints on language:

1. Language should be clear;
2. Language should be quick and easy;
3. Language should be expressive;
4. Language should be processible.

We examine the application of each of these ales boa to natural language and to human-computer interaction.

1. Language Should Be Clear

Natural language. Slobin defines clarity to be a consistent "one-to-one mapping between underlying semantic structures and surface forms." Thus, Slobin's concept of clarity corresponds to consistency as it is generally applied in human-computer interaction. Consistency in a language facilitates learning, both in children and adults. Children not only learn more quickly where it is found, but they enforce consistency by ignoring alternative constructions (using "I will" or "I will not" where adulte would say "I'll" or "I won't") or by using a consistent form even where it is considered to be ungrammatical (using "hitted" rather than "hit" for the past tense).

All natural languages have inconsistencies, the irregularity of verbs being a well-known example. These irregularities cause the language learner great

difficulty, because violations of consistency mean that a single rule no longer applies to a wide class of instances, and instead, many cases have to be learned individually. Although people have created more consistent, artificial languages (e.g., Esperanto), it is significant that none of the thousands of known naturally-forming languages is completely consistent. If consistency were as primary a design rule as some have argued, one might have expected to find a greater appearance of consistency in natural languages.

Human-computer interaction. Computer systems can accrue the same benefits as natural language systems from a clear, consistent mapping between underlying semantic structures (or actions) and surface forms (or commands and system output). Here, too, consistency has been shown to facilitate learning (e.g., Polson, 1988).

However, despite heavy rhetoric advocating consistent design and its prominent place in the standard guidebooks, consistency is often violated. This is not solely due to oversight—in the best of systems, this violation can improve performance (Grudin, 1989). A major point of this paper is to show that some of the same pressures that militate against consistency and an emphasis on clarity in computer systems are found in natural language as well, where they are clearly seen to serve important purposes.

2. Language Should Be Quick and Easy

Natural language. A language principle that often conflicts with consistency and clarity is the desire to be quick and easy. This tendency shows up in numerous ways. Most common words are short and monosyllabic, even in languages that relish long words, such as German. Language is further simplified through abbreviation or other shortening, obtaining efficiency at the expense of learnability, regularity or even clarity. Irregular verbs and plural nouns are generally shorter than their regular counterpart would be—inconsistency is introduced in the service of efficiency.[1]

Often, as a word increases in frequency of use, it is given an abbreviated form:

[1]For example from a list of 173 irregular English verbs, the irregular written form is shorter in 166 cases, the same length in six, and only longer in one ("brought" is longer than "buyed" would be, although equally "quick and easy" to pronounce).

"automobile" becomes "auto," "television" becomes "TV", "picture element" becomes "pixel". Pronouns shorten utterances, but at the cost of introducing ambiguity, reducing clarity. Entire phrases may be eliminated in the cause of efficiency. Although such utterances can technically be ambiguous, usually, when interpreted in context, they are not.

Note that irregular constructions that simplify and shorten will work only if everyone is familiar with them. Therefore, irregularity is most often found with frequently occurring constructions—it is the most frequently occurring verbs that tend to be irregular.

Irregularities cause difficulty during learning, but once learned, they simplify the language process, making the constructions quick and easy to use. As long as the irregularities are frequently encountered, they stay learned. Thus, the mature native speaker seldom has difficulties with irregularities: It is only the learner or the novice user who has trouble.

An interesting demonstration of the relationship between irregular language forms and frequency of usage occurs as language evolves and words change in their frequency of usage. When the frequency of usage of an irregular verb drops, the verb also drops its irregularities and reverts to a regular form (Bybee, 1988). Thus, speakers are not burdened with the task of keeping track of language exceptions that rarely occur.

Human-computer interaction. Do we find the same push toward non-standard, abbreviated structures in computer interactions? Yes, a frequent user's desire for quick and easy means to carry out operations results in simplification, abbreviation—and, therefore, inconsistency. Many computer systems allow their users to create short keystroke sequences as substitutes for longer command sequences: Some systems even provide these "short-cuts" as standard features: shell commands, aliases, scripts, macros, dedicated function keys, option-key equivalences, or "power-keys". Much as the shorter constructions in natural language tend to be those that are used with higher frequency, shortcuts in computer systems are used primarily for high-frequency operations.

In the Macintosh computer, users wanted a quick way to eject a diskette from the drive and to free the memory that the system had reserved for it. Initially, two operations were required: an "eject" command and the

action of moving its remaining, "greyed-out" icon into the trash can. In a triumph of usage over consistency, an imaginative programmer combined these into one operation, carried out by moving the diskette icon to the trash can icon. The operation violated many people's notions of consistency and confused first-time users, but due to its overwhelming efficiency it became widely accepted.[2]

Computer users who create their own shortcuts often produce namesets that are efficient, but so inconsistent that they themselves subsequently forget the names that they devised (Grudin & Barnard, 1985). They may misjudge the frequency with which these terms will be accessed. Other users, of course, are likely to find these personal shortcuts to be incomprehensible. Natural language handles the corresponding problem through several mechanisms.

With a computer system, if a user invents a new command name or other shortcut, this innovation is kept relatively private: Only the user and the computer system need how. Similarly, if a computer designer creates a poor name or shortcut, a user may be able to fix it with an alias, but again this remains a private adjustment.[3] With natural language, however, a neologism is only effective if it is used with others. This shared social use provides for a natural evolutionary process. Successful innovations are those that are kept alive through usage within a language community—we see examples in the way that some slang terms maintain their existence trough frequent usage, whereas others die natural deaths. In language evolution, one natural tendency is towards consistency, and only frequently used constructions maintain an inconsistent form. Today's computer systems provide neither the extensive shared social use of innovations nor an equivalent evolutionary process that will rescue users from poorly devised names or procedures.[4]

The "Law of Least Effort" in human performance. The pressure to increase efficiency is observed in many domains of human skill. Zipf (1949,

[2]The inconsistency has always bothered the design team, however, who plan to phase out this "slang" shortcut if a more consistent but equally efficient solution is found.

[3]A major use of customization features the "undoing" of designer innovations in new releases (Mackay, 1990).

[4]The best analogy between linguistic and computational neologisms may be private abbreviations used in personal diaries or notebooks. Here there are no social interactions, and here the analogy holds well: Much as in the computer world, a diarist's clever abbreviations and phrases may prove undecipherable to the writer turned reader.

1965) postulated that a general "law of least effort" applied to much of human behavior. Zipf showed that a power law applies between the length or size of an instance and its relative ran of frequency occurrence.[5] Ellis and Hitchcock (1986) have found that experienced computer users create command abbreviations ("aliases") that follow Zipf's Law, with shorter terms used for higher-frequency commands. As expertise develops, people modify the task, system, language or method of operation in order to produce smooth, effortless, and efficient performance (Agre & Shrager, 1990; Newell & Rosenbloom, 1981).

3. Language Should Be Expressive

Natural language. Natural languages must have powerful expressive capability. "... to communicate effectively, engagingly, appropriately, and so forth. The speaker must be able to direct the listener's attention, to take account of his knowledge or expectations" (Slobin, 1977, p. 187). The central point here is that language must function in a wide range of contexts, requiring a versatility that often comes into conflict with the other constraints. In order to be both expressive and efficient, language must be compressed—thereby sacrificing a clear, consistent mapping between form and function. Slobin writes, "it is the charge to be expressive which introduces much of the complexity into language."

Miller (1951) observed that "the social pressure for a common vocabulary and the convenience of monosyllabic words tend to restrict the variety of our responses, whereas the attempt to differentiate between similar statements expands the vocabulary and leads to the occasional use of polysyllabic words" (p. 94). This captures the opposing pressures of Slobin's maxims "be expressive" and "be quick and easy." The various tensions push the solutions in opposing ways.

Human-computer interaction. The range of expression is narrower in

[5]This abbreviated description fails to do justice to Zipf's numerous observations of the relationship between ranking, size, and relative frequency. His observations, largely forgotten, may be worth reexamination as indicative of some general principles of human action.

computer interaction than in language, but as applications mature the demands for a wider range of expressiveness grow, and we find analogous sources of inconsistency. Information retrieval systems provide a wide range of search capabilities, whereas a simple string search is sufficient for a word processor; a professional typographer requires a degree of layout precision not needed for most document preparation. The result is often inconsistent interaction languages of varying complexity. In general, large applications may have hundreds of commands to satisfy the requirements of thousands of different users, who oftentimes require very different system performance. We expect this issue to be of increasing importance as computer systems become richer and more powerful.

4. Language Should Be Processible

Natural language. The rate at which the speaker and listener can accurately encode and decode language utterances must be comparable. If they proceeded at rates that were too discrepant or led to too much error, communication would suffer[6]. This is a particular challenge in spoken language, because of the non-persistence of sound—the listener has a limited ability to review what has been spoken, and thus must process it in "real time".

Human-computer interaction. Computer communication is persistent: The computer can preserve a record of input and can provide persistent output by means of a static visual display or by allowing ready repetition of an otherwise transient auditory or visual signal. Even so, a general constraint to be humanly processible operates in the visual medium as well as the acoustic. For example in the design of visual icons, the relative size difference of a trash can and a document in the real world is not mapped onto the interface (it would make one icon too large or the other too small); similarly, one may

[6]The issue is not simply that the listener be able to keep up with the speaker, but that the processing rates be comparable. If the "speaker" were forced to handwrite instead of speak, the mismatch of rates would disrupt communication: Normal reading rates are about twenty times as fast as normal writing rates. We handle this discrepancy by having most writing take place "off-line", asynchronously with reading. In normal spoken conversation, the mismatch between speaking and listening rates is not sufficiently great to affect communication, although especially slow speakers can tax especially quick listeners.

ehance the users' ability to distinguish among objects by exaggerating differences (Hollan Hutchins, McCandless, Rosenstein, and Weitzman, 1987). Thus, if it is crucial for the users of a system that controls an industrial process to distinguish between 200-gallon and 220-gallon boilers, a designer might use icons that vary in size by 50% rather than a precisely-mapped 10%. This violates a clear mapping of semantic information onto surface form, but provides greater human processibility.

Contextual Factors in Language Change

In addition to the design ales, Slobin (1977) discussed four different means by which natural pressures can change natural languages:

1. Gradual evolutional processes;
2. Contact with over languages;
3. Creolization;
4. Individual development.

In this section we briefly examine these four avenues of language change and the way similar factors influence computer interaction design.

1. Gradual Evolutionary Processes

Broad shifts in a language occur that are independent of specific external pressure on it. Slobin presents evidence that these primarily improve how well language can be processed: "At each point in its history the language has apparently been strongly constrained by the charge to conform to perceptual strategies." He also discusses constraints that facilitate production (speech).

These language changes correspond in a sense to broad changes in computer interactions that also have moved toward conformance with perceptual-motor abilities. One step in this direction is the shift from simple "glass teletype" interactions—single line statements displayed on terminals, and typewriter

keyboard input—to full-screen graphical interfaces with input through pointing and gesture. Future computer systems promise to enhance the perceptual mapping through increasing use of graphical displays, including large screens, color, and three-dimensions, and the use of motion and sound. Change in production is manifested in the proliferation of input mechanisms including pointing devices, gesture recognition, and even voice recognition and eye-tracking. Interestingly, Slobin notes that language shifts are accompanied by an initial focus on increasing consistency, a pattern also found in computer system design.

However, there is generally little evolutionary force upon specific computer systems apart from slow pressures of the market and innovation that lead to new releases. These artificial systems are relatively immutable: Once designed, one is unchanged until a new system takes its place. A major exception is in the evolution of inherently extensible computer language systems such as Lisp and Unix, in which new constructions or commands that are added by any user become relatively indistinguishable from the original language primitives. But as noted earlier, computer systems lack a feedback or "natural selection" mechanism. The result of evolution for both Lisp and Unix has been an amazing proliferation of commands and structures, so that a new user faces daunting sight of manuals and documentation whose size is measured in meters. Instead of simplifying a user's task, this form of evolution has increased the learning burden.

But signs of evolution are indeed there. Some of the original constructs of Unix and Lisp are no longer taught to newcomers and are replaced instead with more efficient and useful evolutionary appendages. But we suspect that computer systems suffer from the lack of social interaction and communication. Children learn a language by existing and interacting within a community, and what these new learners acquire then determines what they will pass on to their children. The related process in the acquisition of computer languages and systems has a much different character.

2. Contact with other Languages

When two societies that speak different languages come into contact, the languages change, in part to make communication between the language groups

more efficient. Over time, each language may import elements of the other language: Individual words are the first to cross over, but eventually whole syntactic structures can be incorporated to allow quicker and easier speech (Slobin, 1977). Part of the price of this merger of the two languages and the overall improvement in communication is the introduction of inconsistencies.

A clear analog is found in the computer domain. Operating systems, applications, and application domains can be thought of as independent language families. Contact among these language groups takes place as users move among them or when a single computer comes to support several systems (e.g., as the stand-alone word processor, personal computer, transaction processing, and other worlds come together). Different names are suddenly being used for the same thing or the same name has different meanings in different contexts. This seems a particularly promising topic for further exploration.

In computer programming languages, as with other human-computer interfaces, the clash of different cultures has meant changes to all languages. Thus, elements of structured programming have come to even the least structured languages of all: Basic and Fortran; and algebraic languages have made their impact upon such deviant structures as Lisp and Prolog, which in turn, have led to changes in the algebraic languages.

3. Creolization

The term "Creolization" refers to the creation of a new language by the expansion of a "pidgin" or barter language. Pidgins are communicative systems developed to make it possible for groups that use widely different language systems to interact. These are used primarily for bartering and they tend to be simple and not very expressive. When children acquire the pidgin as a first language, this starts its evolution into a full-fledged language—a Creole. Children first make the language more regular, then expand it to apply it to all situations, adding vocabulary, verb tense, and so forth.

Erickson (1990) notes parallels between pidgin languages and many of today's simple computer interaction languages. As functionality is added, a point is reached where the language form cannot support the desired functions: It is time for the pidgin to become a full-fledged language. Erickson notes that

the lack of tense—our restricted ability to refer to past and future events—is shared by pidgins and computer languages. Such limitations are often most apparent to new users of a system who may feel that the existing structures are needlessly complex yet insufficiently expressive for their needs. New users provide the pressure to develop a full-fledged language—Creolization.

4. Individual Development

Slobin notes that the language learner is first most conceded that language be consistent and processible. Later, the language leaner is willing to sacrifice consistency for expressiveness and efficiency. Speakers of natural languages share their knowledge of the language by propagating their innovations to other speakers.

In the computer world, one finds similar processes. Consistency is of most importance for learners, whereas advanced computer users may welcome or develop shortcuts, even at the expense of consistency. Advanced users do tend to share their special knowledge with others, trading macros, scripts, hints, and shortcuts (Mackay, 1990). Computer magazines usually have columns devoted to hints for the use of specialized systems. And informal tutoring networks develop.

Even so, there is far less sharing in computer usage than in languages because most dialogues involve only one person, and the computer does not learn from the experience. Innovations in speech are immediately passed on to the people with whom we speak, but innovations in computer use only affects one computer system's interaction with the innovator. We have to make a special effort and use a special forum to communicate this innovation to others. To complete the analogy with language, it is the computer that needs to change: As we develop shortcuts, the computer system must make them available to other users of similar computer systems.

Language Evolution and the Design of Computer Systems

The analyses of natural languages and the design of interactive computer systems reveal many of the same pressures. In both communication media, these pressures lead to innovations in the structure of the medium, inconsistencies, and a continual tension between expressiveness, ease of use, ease of understanding, and ease of learning.

Computer systems lack the human ability to interpret context and are thus unable to take full advantage of mechanisms for promoting efficiency. Computer systems and designers could make better use of contextual effects to interpret people's actions, allowing simplification of the actions required of the user. A good example of the use of context is in the specification of Unix files. The full name of a file includes its compete "path" (the entire directory hierarchy), but Unix allows for considerable abbreviation by using the current location of a user in the file hierarchy as the default context. There may be many files in the computer system named "notes," but a user who types just the name "notes" is assumed to be referring only to files in the current directory. Unfortunately, this nice use of context is more the exception than the rule in current system design.

Spoken human communication inevitably contains errors. Listeners often do not even notice these errors because the context makes the utterance interpretable even when ambiguous or erroneous. When listeners do have troubles, the speaker can often detect this through the listener's nonverbal and verbal reactions. Language is an example of a system that seems designed for error—it tolerates a good deal of imprecision and it provides error-correcting mechanisms that are so effective that, after the fact, sometimes neither listener nor speaker is aware of the error. Computer systems' general lack of sensitivity to context means that developers must take the initiative by building in safeguards and confirmation steps to prevent catastrophic errors (Lewis and Norman, 1986)—which can, of course, add complexity or inconsistency to the dialogue.

Today, the proper analogy with computers is perhaps not full-fledged natural languages, but rather pidgins. Like pidgins, human-computer interaction deals with exchanges between users and system that are restricted in domain.

Pidgins are restricted in expressive power. But the "pidgin" used for human-computer interaction must develop toward a Creole as greater range is sought, thus bringing into play all the issues discussed in this paper.

Computer systems are still small and limited. Unlike natural language systems, they do not last for multiple generations of users, and they do not provide mechanisms for the sharing of developments among the user community. Unlike human listeners, they do not evaluate innovations and propagate good ones. More important, perhaps, is that there is none of the richness of natural language that allows for heavy use of context, a relative insensitivity to error, and efficient error correcting mechanisms.

An understanding of how naturally evolving, intensely social systems such as languages cope with conficting pressures can help the designers of artificial systems. But if we are to adapt some of the lessons, we must move beyond today's systems which have relatively limited capabilities and limited lifetimes and that are static and unresponsive. Instead, we must learn to develop systems that have long lifetimes of gradual evolution, and that are adaptive, flexible, and robust.

Acknowledgment

We thank Karen Courtenay, Rod Owen, Larry Parsons, and Steve Pinker for helpful discussions and literature references on the role of language change and irregularity. Tom Erickson, Don Gentner, John Paulin Hansen, Jim Hollan, Phyllis Reisner, Hank Strub and John Sullivan also provided helpful comments. Norman's research was supported by grant NCC 2-591 to Donald Norman and Edwin Hutches from the NASA Ames Research Center in the Aviation Safety/Automation Program. Everett Palmer served as technical monitor. Additional support was provided by funds from the Apple Computer Company and the Digital Equipment Coloration to the Affiliates of Cognitive Science at UCSD.

References

[1] Agre, P. and Shrager, J. 1990. Routine evolution as the microgenetic basis of skill acquisition. *Proceedings of the Conference of the Cognitive Science Society.* Hillsdale, NJ: Lawrence Erlbaum.

[2] Bybee, J. 1988. Morhology as lexical organization. In M. Hammond and M. Noonan (Eds.), *Theoretical morphology.* New York: Academic.

[3] Ellis, S.R. and Hitchcock, R.J. 1986. The emergence of Zipf's Law: Spontaneous encoding optimization by users of a command language. *Transactions on systems, man, and cybernetics, 16*, 3,423–427.

[4] Erickson, T. 1990. Interface and evolution of pidgins: Creative design for the analytically inclined. In B. Laurel (Ed.), *The art of human-computer interface design.* Reading, MA: Addison-Wesley.

[5] Grudin, J. 1989. The case against user interface consistency. *Communications ACM, 32*, 1164-1173.

[6] Grudin, J. and Barnard, P. 1985. When does an abbreviation become a word? and related questions. In *Proceedings of CHI '85.* New York: ACM.

[7] Hollan, J.D., Hutchins, E.L., McCandless, T.P., Rosenstein, M., and Weitzmann, L. 1987. Graphic interfaces for simulation. In W.B. Rouse (Ed.), *Advances in man-machine systems research, Vol. 3*, Greenwich, CT: JAI, 129–163.

[8] Lewis, C. and Norman, D. A. 1986. Designing for error. In D. A. Norman and S. W. Draper (Eds.), *User centered system design.* Hillsdale, NJ: Lawrence Erlbaum.

[9] Mackay, W.E. 1990. Patterns of sharing customizable software. *Proceedings of CSCW '90.* NY: ACM.

[10] Miller, G. A. 1951. *Language and communication.* New York: McGraw-Hill.

[11] Newell, A. and Rosenbloom, P. S. 1981. Mechanisms of skill acquisition and the law of practice. In J. R. Anderson (Ed.), *Cognitive skills and their acquisition.* Hillsdale, NJ: Lawrence Erlbaum.

[12] Polson, P. (1988). The consequences of consistent and inconsistent user interfaces. In R. Guindon (Ed.), *Cognitive science and its applications for human-computer interaction.* Hillsdale, NJ: Lawrence Erlbaum.

[13] Slobin, D. I. 1977. Language change in childhood and history. In J. Macnamara (Ed.), *Language learning and thought.* NY: Academic, 185–214.

[14] Zipf, G. K. 1949. *Human behavior and the principle of least effort; an introduction to human ecology.* Cambridge, MA: Addison-Wesley.

[15] Zipf, G. K. 1965. *The psycho-biology of language; an introduction to dynamic philology.* (Introduction by George A. Miller.) Cambridge, MA: MIT.