PB - 178

J. Nielsen: Protocol Model

TRYK: DAIMI/RECAU

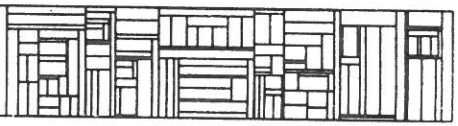
ISSN 0105-8517

A Virtual Protocol Model for Computer-Human Interaction

Jakob Nielsen

DAIMI PB - 178
September 1984

Computer Science Department
AARHUS UNIVERSITY
Ny Munkegade – DK 8000 Aarhus C – DENMARK
Telephone: 06 – 12 83 55



A VIRTUAL PROTOCOL MODEL FOR COMPUTER-HUMAN INTERACTION

Jakob Nielsen
Computer Science Department
Aarhus University
DK-8000 Aarhus C
Denmark

Abstract

A model of computer-human interaction is presented, viewing the interaction as a hierarchy of virtual protocol dialogues. Each virtual protocol realizes the dialogue on the level above itself and is in turn supported by a lower level protocol. This model is inspired by the OSI-model for computer networks from the International Standards Organization.

The virtual dialogue approach enables the separation of the technical features of new devices (e.g. a mouse or a graphical display) from the conceptual features (e.g. menus or windows). Also, it is possible to analyze error messages and other feedback as part of the different protocols.

CR Categories and Subject Descriptors: H.1.2 [Models and Principles]: User/Machine Systems

General Terms: Human Factors

Additional Key Words and Phrases: software ergonomics, virtual dialogues, open systems interconnection

1. INTRODUCTION

Discussions about issues such as software ergonomics or human factors of computer-human interaction tend to be rather imprecise and often focus on minor points such as the use of a mouse versus a touch-sensitive screen as a pointing device. This is of course partly due to the fact that software ergonomics cannot at the present point in time be considered a "hard" science such as e.g. complexity theory. But it is also partly due to the lack of a commonly accepted framework for such discussions, or in other words: A taxonomy of software ergonomics is needed.

Such a taxonomy would consist of performance indexes for "good" ergonomics, of respectable scientific and experimental methods, and much more. But perhaps most important is a model of what it is all about, viz. the interaction between the computer and its human user.

The word "model" has many meanings among software ergonomists [7]. There is the "cognitive model", which is a model (held by the system designer) of the functioning of the user. There are also the "intended conceptual model" and the "actual conceptual model", both of the computer system. The actual conceptual model is the collection of facts and opinions that an individual user actually holds about the functioning of the computer system, and the intended conceptual model is the model of the system that the user ought to have.

Here we present an interaction model. Our model is a dual model of both the user and the computer at the same time. The main purpose is to supply a framework for software ergonomic

discussions. The model is used as such an underlying framework for some design recommendations in section 6.

2. OVERVIEW

A computer-human interaction is a dialogue in which the two participants are sending each other messages according to certain established conventions. Our inspiration in modelling this dialogue has been the reference model for Open Systems Interconnection (OSI) from ISO, the International Standards Organization [4,11], used in the discipline of computer networks. However due to the fact that "humans, unlike machines, are not designed explicitly as parts of man-machine systems" [3] there is a fundamental asymmetry in computer-human interaction not present in computer networks. We are not able to alter drastically the communication methods of one of the communication partners (the user), so the other part (the computer) has to be flexible. That this argument has often been presented the other way around is an indicator of the changing economical ratio between CPU cycles and user confusion and of the low regard of ergonomics held by earlier generations of system designers.

Often the human and the computer use completely different communication channels, e.g. a keyboard for human messages and a VDU-screen for computer messages. This distinction is often made by calling the user's messages input and the computer's messages output. We could also use the terms Command Language and Response Language, respectively.

An overview of the model is given in figure 1. It is seen

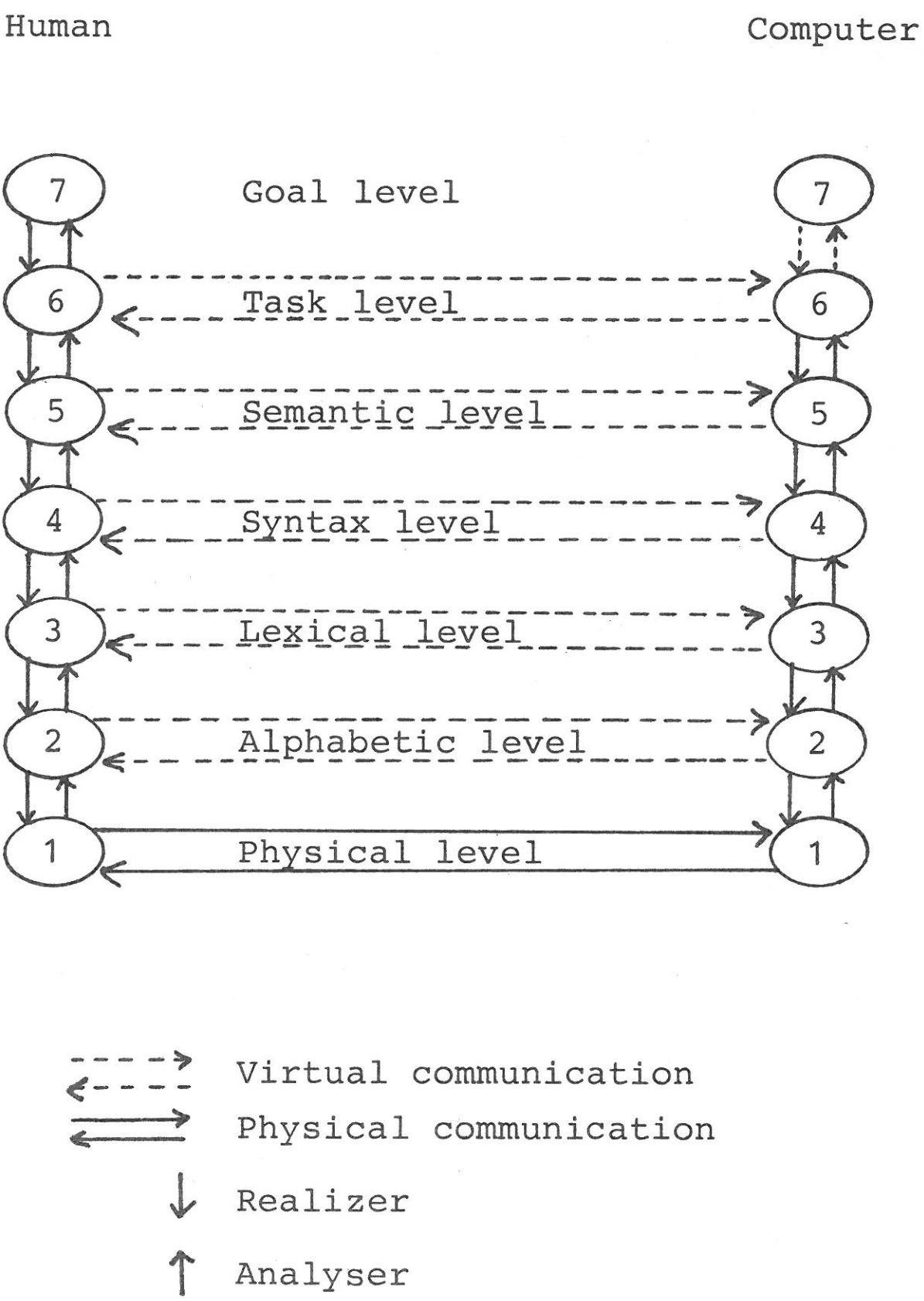


FIGURE 1. Overview of the protocol model.

that the model consists of 7 layers of communication. These are explained more fully in section 3. One might note that the ISO OSI model also consists of 7 layers. This identity in number of levels is only a coincidence, however, as the levels have totally different meanings in the two models.

The levels are numbered from one to seven, with the lowest level having the lowest numbers.

An important aspect of this model is that physical communication is only exchanged at level 1, the physical layer. All other levels realize virtual communication. Messages on level i are exchanged between the actors by way of level i-1. While the level i protocol inside one of the actors (human or computer) "thinks" it is communicating directly with the level i protocol in the opposite actor, in reality it is realizing the communication by sending the messages via its own level i-1. The level i-1 protocol at the opposite actor receives this message, analyses it, and passes it on to its level i. In general one virtual level i message may be realized by several level i-1 messages.

The process translating level i messages to level i-1 messages is called a realizer and the inverse process translating level i-1 messages to level i messages is called an analyser.

3. THE LAYERS

We will now examine the seven layers of the interaction model one by one. We will start with the topmost level and work

our way downwards.

Level 7: Goal Layer

This layer is distinct from all the other layers in the model in that it deals with the real world concepts that the computer system is all about. All lower levels deal with the representations of these concepts inside the given computer system. We assume that there exist such a real world external to the computer in which we have a goal to obtain. Therefore we might say that the computer does not have a goal level, but we will often be pragmatical and say that it has. It may e.g. be the case that our computer has received a letter over some electronic mailing system, so that it now "wants" to tell us that we have a letter waiting. In reality the motivation originates with the sender of the letter, who is presumably a person.

Another example of a level 7 operation would be the goal to delete the last section of the letter (that I am writing on my word processor). In this model we only want to include operational goals. We do not consider such tactical or strategic goals as e.g. why I want to delete that section of the letter. It might be because I realize that it would be impolite to end a letter that way - but such considerations are not part of the design and operation of a computer system. Note that if we invent an AI program to check the politeness of letters, then such issues would be part of the semantic level of the model and we would still not include the question of why I am interested in the politeness of the letter. We would redefine our understanding of the computer system to include the ability to assist us with a larger proportion of our problems, but we would still in

principle have to maintain the distinction between e.g. real world politeness and computer representation of politeness. Some users may have problems maintaining this distinction due to the "user fooling powers" of computers [9].

Level 6: Task Layer

This level deals with general computer-related concepts that are representations of the real world concepts from level 7. The question on this level is more what kinds of objects and operations are in the system and not whether these are directly implemented in the system. They may have to be realized by piecing together a sequence of operations from level 5.

An example could be to delete the last 6 lines of the edited text. This means that our computer system supports the concepts of lines and text (even the concept of the edited text). Note the changes from the same example in level 7: We have changed "section" to "certain lines" and "letter" to "edited text".

Level 5: Semantics Layer

This level determines the detailed functionality of the system: Exactly what each operation does to each object. There are a finite number of concepts in the system and they each have an exact definition. This means that level 5 handles the meaning of the interaction (while the form is the business of the lower levels). One of the things that is included in the semantic level is a definition of what information is necessary for each operation.

One semantic operation could be to remove a line with a given linenumber. And several such commands would then be needed

to realize the task to delete the last 6 lines. Note that even though the system might also support the concept of an interval of linenumbers, this would not be part of the actual communication protocol if the user was unaware of this feature.

Level 4: Syntax Layer

The syntax deals with the sequence of the input and output tokens exchanged on the lexical level. The sequencing can be both in time and in space, including two-dimensional placement of tokens, i.e. including screen layout. The syntax layer includes not only textual syntax but also sequencing of e.g. pointing operations.

One syntactical example could be "DELETE 27" to realize the semantic command "remove a specific line". This example follows the "verb/noun" syntax but we often also see a "noun/verb" syntax such as "point to the wanted line and then hit the DELETE button".

Level 3: Lexical Layer

At the lexical level exchange is going on using the information carrying symbols of the interaction, called tokens. A token may be a word, a special symbol, an icon, a number, a pair of screen coordinates, etc. These symbols are the smallest units with their own system-related meaning.

One example of a lexical token could be the word "DELETE". It could be changed to the Danish word "SLET" having the same meaning without having to change any other part of the interaction, i.e. keeping the same syntax and semantics of the command and the same task and goal levels of the system. The system might

also accept the abbreviation "DEL" as a legal alternative for "DELETE" and still keep all other parts of the interaction unchanged.

Level 2: Alphabetic Layer

Any system contains an alphabet of primitive information carrying units, called lexemes. They do carry information, but they do not have any meaning by themselves. Only when they are put together to form tokens can they be interpreted in system terms. Lexemes may be e.g. letters and digits in the case of textual I/O, lines and colors in the case of graphics, and phonemes in the case of speech I/O.

If screen coordinates are used as a token on level 3, they may be realized in several different ways in the alphabetic layer. One may use pointing devices with different precision of resolution such as a mouse, a touchscreen, or a light pen. Or one may use a sequence of cursor positioning control characters to place the cursor at some point on the screen. Of course each of these alphabetic possibilities will then in turn have to be realized by different hand movements at the physical level and may thus be of varying efficiency in practice. But in principle all support the same interaction on th higher levels.

An example of a lexeme could be the letter "D" which may or may not form part of the token "DELETE".

Level 1: Physical Layer

The physical layer is where there is an observable interchange of information in the form of light, sound, movements, etc.

We may e.g. observe that the user actually presses the D-key on the keyboard, and we may then deduce that it probably should be interpreted as input of a "d". We may also observe that the user simultaneously presses the D-key and the SHIFT-key, in which case we deduce that input of a "D" is intended.

The question of how to design the interaction on the physical level is of course a question of traditional hardware ergonomics and will not be discussed further.

4. IS THE MODEL FOR REAL?

Table I shows an overview of the 7 layers of the model. Note that it is not claimed that the human brain actually follows the layered structure given in figure 1 and table I. It is only claimed that this structure is convenient for discussing and analysing computer-human interaction.

We could call layers 5-7 the "invisible" layers since they are not directly reflected on the computer screen or other I/O-media. And we could similiarly call layers 1-4 the "visible" layers, as they are directly visible (or maybe audible). This distinction should be an obviously useful one, and as shown in section 5, it is made by several other interaction models.

Inside the invisible layers it should also clearly be OK to distinguish between the real world concepts of the goal level and the computer implemented operations and objects of the semantics level. The task level is seen as grouping together concepts and operations from the semantics level with the intention of fulfil-

No.	Name	Exchanged units of information	
7	Goal	real-world concepts	external to the computer system
6	Task	system concepts	what kind objects are in the system, and what can we do with them
5	Sementics	detailed functionality	concrete objetcs in the system and specific operations
4	Syntax	sentences	sequences in time and space (1 or 2 dimensional) of the communicated tokens
3	Lexical	tokens	smallest information-carrying units, e.g. words, numbers, icons, screen-coordinates
2	Alphabetic	lexemes	primitive symbols (hardware dependent), e.g. letters, colors, lines, phonemes
1	Physical	physical information	"hard I/O", e.g. light, sound, movements

TABLE I. Summary of the levels in the protocol model.

ling a goal from the goal level. This is frequently done in practice, and is known under the names of scripts, procedure files, macros, etc. Also users will often form plans of how to combine operations even if they do not always take the trouble of writing them down as a macro. So the task level should also be part of an analysis of an interaction.

The destinction between the different visible layers are if nothing else often explicit inside the computer itself, and should as such be of concern. The novice human user will also be conscious about problems of spelling of keywords, how to use a pointing device, etc. Later on most of these seperate concerns are more or less forgotten as the user "automatically" generates the correct syntax just typing away on the keyboard. But in case of a ccommunication breakdown or just some "noise", the different layers become explicit again. Just consider a user having to read the number 27 and being told that it is an octal number. To decode 27 as twenty-three most users will have to activate an alphabetic to lexical analyser.

5. OTHER MODELS

Several other interaction models exist. Perhaps the most cited models are the Command Language Grammar (CLG) by Moran [6] and the 4-design model by Foley and van Dam [2]. Buxton [1] has modified the Foley/van Dam 4-design model to a 5-design model by adding a so-called pragmatic component. All these models are layered models even though they do not use the virtual protocol scheme advocated in this paper. The Moran model is intended as

a model of how the user may view the computer system, and the Foley/van Dam model is a model of how to design an interface top-down. They are not communication models with explicit representation of both the human user and the computer at the same time.

The three mentioned other models are shown in figure 2 where they are compared with the 7-layer virtual protocol model described in this paper. The figure shows corresponding parts of the models at the same horizontal level. Note that the Foley/van Dam model do not include the real world related concepts from our goal level and not all the hardware and movement oriented details from our physical level. The Moran model includes some issues from the real world that are not part of our goal level that is limited to immediate tactical goals. The way the models are illustrated in figure 2 is of course somewhat biased as the 7-layer model on the figure seems nice and cleancut while the other models seem convoluted. This is of course not necessarily the case.

On the other hand most of the difference between the models is due to the placement of the question of screen layout which is placed in the spatial level by Moran and in the pragmatic component by Buxton. These model components also contain more low-level issues such as I/O-hardware in the Buxton model and the spatial placement of the I/O devices in the Moran model. The argument is that screen layout is a rather hardware related question and should be placed together with other details of the interaction devices used.

It is true that the hardware places certain limits on the

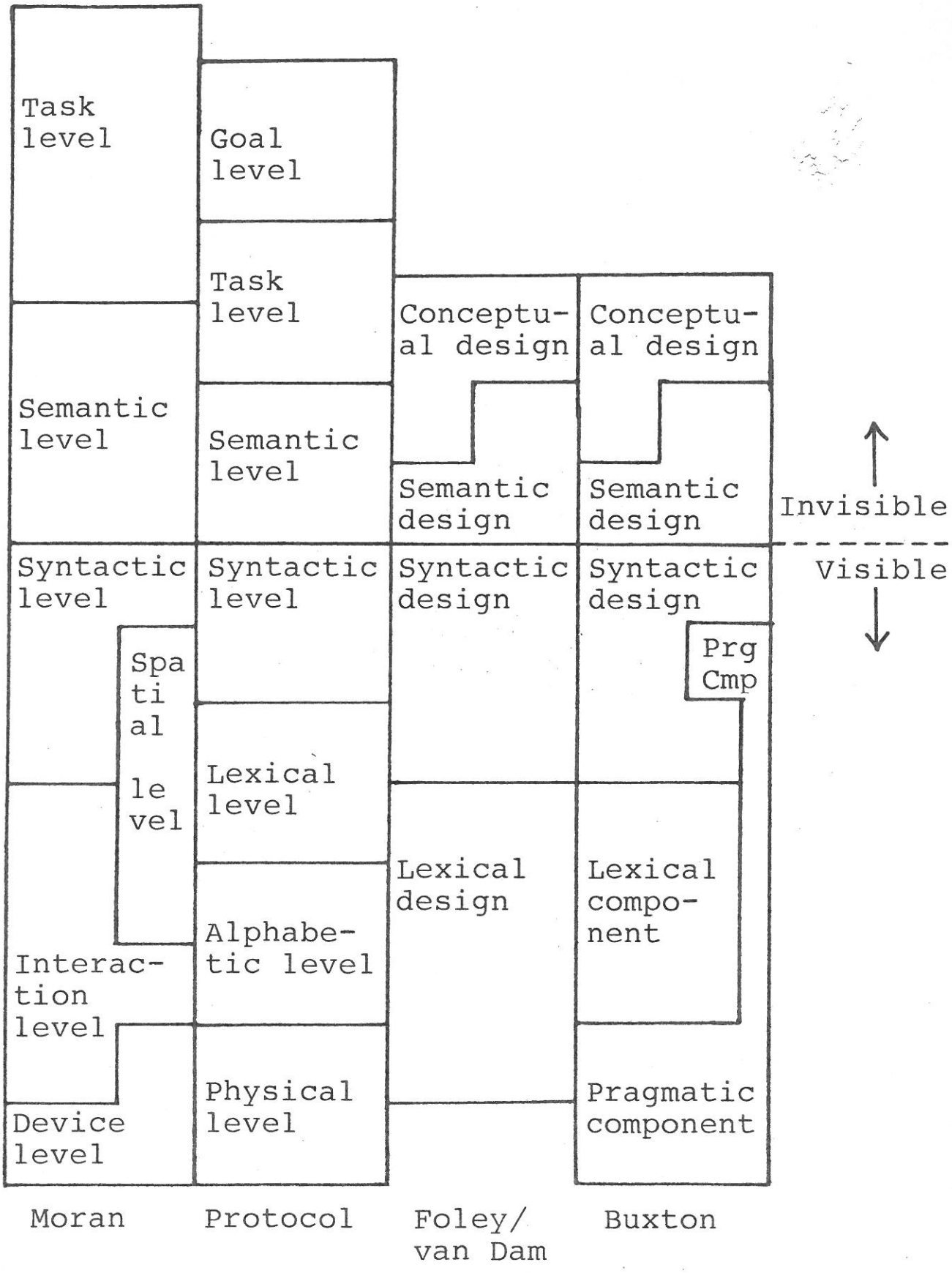


FIGURE 2. Corresponding parts of four interaction models.

possibilities for screen layout. One cannot e.g. use overlapping windows on a hardcopy TTY. But this dependency between layout and hardware is not unique as it will always be necessary to limit the design of the higher interaction levels to those features that are possible to implement efficiently using the existing facilities of the lower levels. Screen layout should be considered just a form of two-dimensional syntax that in principle does not differ from the traditional one-dimensional syntactical issues of grouping and sequence of lexical tokens.

Buxton argues that his pragmatic component has one of the strongest effects on the user's perception of the system, since all interaction takes place using the physical I/O equipment as the primary contact between system and user. This is only true if one does not accept the concept of a virtual communication. In the model presented in this paper it is still true that all contact between user and system takes place at the physical level but that is only in order to realize communication on the other levels in the model. Therefore even though the physical level is clearly still very important, it is mainly a sort of transit station for communication that affects the user more crucially on the higher levels.

One example showing that the actual I/O devices are not of absolute importance is given by Loftus and Loftus [5, p. 69] who describe an expert Pac-Man player who had learned the game at the video arcades. She was then introduced to a home-computer version of the game in which Pac-Man was directed not by a joystick but by control keys on the keyboard. It took her very little time to become just as expert at this version of the game as she could

transfer all her knowledge of the high level interaction, e.g. how to avoid the monsters.

6. DESIGN RECOMMENDATIONS

Such popular interaction concepts as "direct manipulation" and "What-You-See-Is-What-You-Get" (WYSIWYG) may be seen as following the principle that there should be a close connection between the dialogue protocols on the different levels, which would tend to make the work of the analysers and realizers more simple [8]. Direct manipulation could be seen as using the syntax level to closely mirror the semantics level. And WYSIWYG is a close coupling between the goal level and the syntax level.

The model may also be used to give more specific guidelines to designers about how to aim for user friendliness in their programs. One obvious recommendation is to explicitly recognize the layered protocol nature of the interaction and to support this. One important way of supporting a communication protocol is to provide feedback in the form of both acknowledgement and not-acknowledgement messages (the latter are usually called error messages).

Of course the feedback should occur at each level in the model and it should be in terms of the concepts being exchanged at that level. One could imagine a computer system that limits filenames to at most 7 characters. If the user issues the command SAVE,LONGNAME (i.e. giving a filename of 8 characters) the system should recognize this as an error on the lexical level in that an illegal symbol is given to represent a filename. Therefore one

should not use such error messages as "ILLEGAL COMMAND" or "SYNTAX ERROR" in this case since the command as such is well formed syntactically. The following lexical feedback should be given instead: "FILENAME TOO LONG (MAY CONTAIN AT MOST 7 CHARACTERS)".

One could say that this improved error message might be derived from some traditional design guideline such as: "Error messages should be specific and constructive (indicating what must be done)" [10]. This is true, but the point is that the virtual protocol model helps the designer to analyse the problem and decide on how to follow that sort of guidelines in the given situation.

Another general design guideline is that the higher levels in the model are the "important" ones in that problem solving activities take place there. This problem solving should be concerned with the application goal only so we can infer two recommendations: First, the activities not related to that problem solving should be transferred to the lower levels. And second, the lower levels should not themselves take up more of the user's cognitive resources ("brainpower") than absolutely necessary. This last point leads to e.g. the use of menus, meaning that the user does not have to remember the lexical names of commands.

And the point about moving work from the higher levels to the lower could be used in favor of using windows instead of modes to achieve context sensitivity. The natural human way to communicate does not require that you specify your intentions totally unambiguously. When using a text editor I might write

e.g. "FIND RASMUS" which would be interpreted as either some text to insert in the file (if I was in "insert" mode) or as an instruction to find the next occurence of RASMUS (if I was in "command" mode). I do not want to have to write "INSERT:FIND RASMUS" respectively "COMMAND:FIND RASMUS" each time to avoid the context sensitivity of the modes. However the modes do require a special mode-change operation wahih must be invoked before some features from another mode may be used. Therefore the use of modes require advance planning, meaning the involvement of the task level to piece the operation together from the required command as well as mode canges. This plases the burden on one of the highest levels in the model and so the use of modes has become somewhat unfashionable [12].

Instead of modes, different windows on the VDU screen may be used to determine the intended context of user input. Now the change between two windows is a syntactic operation (recall that level 4 includes two-dimensional syntax), so we have moved work downwards from level 6 to level 4 which is an improvement according to the first of the two design recommendations mentioned above. Therefore we have concluded that windows in general are preferrable to modes.

Of course one could argue that it also requires advance planning to decide to move the cursor to another window so that level 6 continues to be involved. But because both windows are visible at the same time one would normally consider movement between the two vindows as such an "immediate" operation that it is achieved by a lexical pointing symbol (perhaps realized by repeated alphabetic cursor positioning or mouse rolling) without

being considered an actual command. But admittedly this is a subjective assessment so at this point the formalism of the model fail: It cannot give an indisputable answer as to what is user friendly. It can only help you structure your thoughts in the matter.

7. CONCLUSION

There are no easy answers in software ergonomics. But this virtual protocol interaction model aims at making it somewhat easier to achieve some answers anyway.

The model has not been verified. Since its purpose is to improve the ergonomics of software design, a validation would consist of supplying some designers with the model and letting them design some systems. A control group of designers would then design the same systems without the benefit of the model. The usefulness of the model would then be considered verified if the designers using the model did better than the control group. The only problem is that the only way to measure the user friendliness of a program is to try it out on some test users. So the verification test would be a two-level experiment and a very expensive project indeed.

Acknowledgement

This paper is partly based on the author's thesis research which was done with Brian H. Mayoh as thesis advisor. Further work was done under research grant 11-4296 from the Danish Natural Science Research Council.

References

[1] Buxton, William. Lexical and pragmatic considerations of input structures. ACM SIGGRAPH Computer Graphics 17, 1 (Jan. 1983), 31-37.

[2] Foley, James D. and van Dam, Andries. Fundamentals of Interactive Computer Graphics. Prentice-Hall, Englewood Cliffs NJ, 1982.

[3] Hollnagel, Erik. What we do not know about man-machine systems. Int. J. Man-Machine Studies 18, 2 (Feb. 1983), 135-143.

[4] International Standards Organization. Draft International Standard, Information Processing Systems: Open System Interconnection - Basic Reference Model. ISO/DIS 7498 (ISO/TC97), 1982.

[5] Loftus, Geoffrey R. and Loftus, Elizabeth F. Mind at Play - The Psychology of Video Games. Basic Books, New York 1983.

[6] Moran, Thomas P. The command language grammar: A representation for the user interface of interactive computer systems. Int. J. Man-Machine Studies 15, 1 (July 1981), 3-50.

[7] Nielsen, Jakob. The Spectrum of Models in Software Ergonomics. Computer Science Department, Aarhus University, 1984.

[8] Nielsen, Jakob. Principles of Isomorfism for User Friendly Programming. Computer Science Department, Aarhus University, 1984.

[9] Plum, Thomas. Fooling the user of a programming language. Software Practice and Experience 7, 2 (March-April 1977), 215-221.

[10] Shneiderman, Ben. Designing computer system messages. Commun. ACM 25, 9 (Sept. 1982), 610-611.

[11] Tanenbaum, Andrew S. Network protocols. Computing Surveys 13, 4 (December 1981), 453-489.

[12] Tesler, Larry. The Smalltalk environment. BYTE 6, 8 (August 1981), 90-147.

transfer all her knowledge of the high level interaction, e.g. how to avoid the monsters.

6. DESIGN RECOMMENDATIONS

Such popular interaction concepts as "direct manipulation" and "What-You-See-Is-What-You-Get" (WYSIWYG) may be seen as following the principle that there should be a close connection between the dialogue protocols on the different levels, which would tend to make the work of the analysers and realizers more simple [8]. Direct manipulation could be seen as using the syntax level to closely mirror the semantics level. And WYSIWYG is a close coupling between the goal level and the syntax level.

The model may also be used to give more specific guidelines to designers about how to aim for user friendliness in their programs. One obvious recommendation is to explicitly recognize the layered protocol nature of the interaction and to support this. One important way of supporting a communication protocol is to provide feedback in the form of both acknowledgement and not-acknowledgement messages (the latter are usually called error messages).

Of course the feedback should occur at each level in the model and it should be in terms of the concepts being exchanged at that level. One could imagine a computer system that limits filenames to at most 7 characters. If the user issues the command SAVE,LONGNAME (i.e. giving a filename of 8 characters) the system should recognize this as an error on the lexical level in that an illegal symbol is given to represent a filename. Therefore one