**WORKING NOTE AC97**

Alan Creak
7 August 1996

# INTERACTORS IN REHABILITATION SYSTEM DESIGN

*I've been exploring some ideas of interactors as they might be applied to previous notions of a rehabilitation system design aid (€me, unpublished, recently ). Here I collect some of my observations.*

## BRIEF BACKGROUND.

For some years, I've been thinking about approaches to the design of computer-based rehabilitation systems, with particular concern for the computing components of systems used in communication aids[5]. This has led me to suggest a design discipline[2] which could perhaps be thought of as a member of the "walkthrough" class[4], intended to be used as an informal guide to system design.

A major guiding principle of my explorations has been to ensure that all relevant factors are taken into account at all stages of the design. An important characteristic of rehabilitation communication systems is that they are directly concerned with communication between people; though HCI is obviously important, it is used as a means of achieving HHI in a more direct sense than is common in more familiar examples of computer interaction. Even in programmes such as word processors, which are certainly concerned with HHI in the long run, the main concern in interface design is on HCI, with emphasis on facilities required by the writer to control the printing device in such a way as to produce the effects which will convey the intended meaning to the eventual reader of the document produced. The writer is primarily controlling a printer rather than engaging in communication. I shall later use the example of emphasis; in the context of the word processor, it doesn't exist. Instead the operator must work at the machine level, perhaps selecting italic or bold text as a means of conveying the emphasis required. Notations such as HTML, in which emphasis can be explicitly requested, are unusual, and seem clumsy in comparison.

In order to provide for all the information carried in the communication signal, I have based my discussion on a stratified model of communication derived from some earlier work[1], with the top level potentially including all the components of human interaction. By describing the information to be conveyed by the system in these terms, and checking at all internal interfaces that all components are still present, it is possible in principle to determine whether the system meets its initial specification – or, alternatively, to show that, within the constraints of machinery available or cost or whatever, the specification cannot be satisfied.

The final paragraphs of a short account of this proposed design method[2] read thus :

> In this note, I have shown that a certain way of describing the flow of information through communication systems can assist in drawing up system specifications, and throw some light on the performance of such systems in practice. I can add, from my own experience in following this approach in other systems, that it has given me helpful insights into the relationships and interactions between the many activities which must operate at different stages of the communication process and at different levels of complexity.
>
> The approach therefore appears to have some value as an informal descriptive technique. It is less obvious that it can be developed as a useful method for system design (€which was the original intention ) as well as for description. For such a development to be possible, it would be necessary to conduct the analysis as a much more formal process, with standards established for the identification of the steps in the communication, and for their description.
>
> Further work in this direction will begin with careful analysis of a set of cases covering a wide range of communication techniques. The aim will be to establish standards as described above, and thereby to develop the method into a useful tool for designing communication systems.

When listening to a seminar on interactors given by David Duke, I was struck by what appeared to be a very strong resemblance between his descriptions and the ideas with which I had been experimenting. I have since spent some time on exploring the applicability of the ideas of interactors to my design method, and they seem to fit uncommonly well. My major reference has been an informal paper produced by David for the seminar[3], referred to hereafter as "David's paper".

I am fairly optimistic that a theory based on interactors, or something quite like them, might turn out to be the more formal approach which I wanted.

## HOW IT WORKS.

To illustrate my discussion, I shall use the example[2] of extending a simple single-switch scanner and communicator system so that, as words are entered one by one, any word can be marked as "to be emphasised", and will then be expressed with some appropriate form of emphasis in the system's output. The aim is to make it possible to utter (€in whatever output medium ) phrases equivalent to "Mary had a little lamb", "*Mary* had a little lamb", "Mary had a *little* lamb", etc. There is no special significance to the choice of emphasis as the example, except that it's easy to think about and to write down, it justifies the title of this note, and it's an example which I used in my previous discussion; apart from that, most of the discussion would apply equally well to a keyboard. ( Indeed, much of it does. )

A scanner is a device for encoding characters ( or anything else, for that matter ) in terms of their coordinates on a grid. ( – or, more precisely, in terms of an ordered pair of time intervals, but the grid description will suffice to describe the operation. ) A screen displays a set of characters arranged in a rectangular matrix ( typically 6×6 ). In its resting state, groups of characters – I'll assume rows, but columns can obviously be used in the same way – are highlighted one by one, and the scanning process continues cyclically. Receipt of a signal – typically a key depression, though other means are common – selects the row which is currently highlighted, and switches the scanner into its character-select mode, in which the cyclic highlighting is repeated, but this time with the individual characters of the selected row. A second signal selects the currently highlighted character, and the system reverts to its resting mode. It's slow, but it needs only a single switch, and it works. Given the coordinates selected by this process, software can then produce a corresponding ASCII or other signal straightforwardly. ( It occurs to me that it might be interesting to develop a careful analysis of the scanner itself using the interactor techniques; here, I've used the notation in a more descriptive way over a larger fraction of the basic system. )

The basic machine is composed of an input device which selects one character at a time ( in this case, the scanner ), some internal software which deals with the characters, and an output device which presents the material entered to a dialogue partner. I haven't defined the output device. I assume only that there is some way to present a phrase with the desired emphasis; it could be a speech synthesiser with control of volume or pitch, or a terminal screen with flashing characters, or a simple printer with capital letters. The analysis I've presented here doesn't get as far as the output device, but can certainly be extended to deal with these cases.

Using a scanner of conventional design, with no specific provision for emphasis, it is straightforward to enter the sentence "Mary had a little lamb", but much more difficult to convey the sense of the emphasised variants. To do so, you can resort to circumlocution ( "Mary had a little lamb, but John didn't" ), or perhaps – if you're willing to learn more technical details of your system – find ways to exploit specific characteristics of the output device you are using. Neither of these alternatives is entirely satisfactory. Circumlocution will work with the simple system, but takes longer, and generally requires more sophisticated use of language; to control the behaviour of the device when your access to it is restricted to the simple text channel available might be difficult or impossible, and in any case would be unlikely to transfer easily to a different device. ( The control signals used to flash characters on a screen are rather unlikely to be the same as those which alter the volume of a speech synthesiser. )

In order to do better, emphasis must be explicitly designed into the system. This is where the design technique becomes useful, for it offers a systematic way to check that the new feature is introduced uniformly and consistently throughout the system. The procedure is to introduce the new feature at some point in the system, and to follow the implications of the change from component to component. There is no prescribed starting point; in this example, the obvious place to begin is at the input interface where the person using the system wants to mark a word as emphasised. My first step is therefore to imagine three ways of handling the emphasis at the scanner interface :

- by **markup**, incorporating the control information in the same stream as the characters : "Mary had a <em>little</em> lamb";
- by **tagging the input**, retaining the single stream but introducing new signals for control : use a double key operation instead of a single operation at some point while entering "little";
- using new signals carried by a **separate channel** : supply a second switch which implies emphasis if depressed at any time while "little" is being entered.

These are to be regarded as three separate possibilities, and the intention of the analysis is to compare them in detail with a view to choosing the most appropriate.

In all cases, the process can be described in terms of interactions between components : person – scanner switch – scanner software – screen – output device driver – .... I have not worked through the whole system in any detail (€it's extraordinarily tedious, and I didn't want to spend a lot of time barking up wrong trees ), but I have done enough to convince myself that the interactor treatment is very effective, and corresponds closely to my informal descriptive approach.

## LUDICROUSLY ABBREVIATED SUMMARY.

Here is a collection of not necessarily coherent bits and pieces from my attempts to describe my ideas in terms of interactors. The translation into interactorese went smoothly in a sense, but I had to backtrack several times as I understood more about either interactors or the system I was describing to repeat sections. Some of the backtracks lead to comments which I've collected in the next section.

The increasing understanding of interactors was welcome and expected ( and almost certainly yet to be completed ); the increasing understanding of the system was not expected, and therefore more welcome, and one of the reasons for my emthisiasm.

I'll describe various components of the process whereby some notion ( N ) is passed from one person ( A ) to another ( B ). I insist that the end points of the communication are both people, because I don't want to introduce arbitrary limitations on the nature of the notion. Communication between people is certainly far richer and more complex than any of our communications with entities such as animals or machines, and forms such as literature and poetry have grown as part of our efforts to achieve fully effective communication through the medium of words. I want my discussion, at least in principle, to include such encoding systems.

In order to communicate N, A is restricted to the use of a communication system ( C ), which for present purposes is composed of a scanner ( S ), somewhat augmented as described above, for input, a device capable of expressing emphasis, but otherwise undefined, for output, and anything else that turns out to be necessary. A's first task is therefore to express N in terms of vocabulary acceptable to C. To fix ideas, I'll define N as the notion embodied in the sentence "Mary had a little lamb, but has one no longer". How can this be expressed in terms acceptable to the scanner ?

The scanner's vocabulary is very limited, and different in each of the three cases described. The lists of possibilities in the three cases are :

- **markup** : do nothing, operate the key;
- **tagging the input** : do nothing, operate the key once, operate the key twice;
- **separate channel** :
  for the scanner : do nothing, operate the key;
  for the second switch : do nothing, operate the switch.

The scanner actions can be described using David's notation. Supposing the scanner to begin in its initial state with row 1 highlighted ( $S_1$ ) :

do nothing :

> A $\oplus$ [ S.donothing ] $\rightarrow$ A $\rhd$ [ S.donothing ];
> [ S.donothing ] $\oplus$ $S_1$ ### [ S.donothing ] $\nabla$ $S_1$;
> [ clock ] $\oplus$ ( [ S.donothing ] $\nabla$ $S_1$ ) $\rightarrow$ $S_2$

operate the key :

> A $\oplus$ [ S.press ] ### A $\rhd$ [ S.press ];
> [ S.press ] $\oplus$ $S_1$ $\rightarrow$ [ S.press ] $\nabla$ $S_1$ ### Srow1

where Srow1 represents the scanner in its alternative state ready to scan along row 1. Corresponding sequence apply to the other possible states. "A $\oplus$ [ S.donothing ] denotes A having formed the intention of doing nothing in the S vocabulary. "The significance of "[ S.donothing ] $\nabla$ $S_1$" is that S has received no input while in state 1; in consequence, as shown by the next step, it changes its state to 2 at the next clock signal. In effect, the result is a description of a simple finite-state machine which moves the scanner through its states as the clock signals are received according to the action ( or inaction ) perceived during the time interval. The current state must be communicated to A by feedback through the screen display, and this must be interpreted by A as he decides what to do :

> $S_1$ ### $S_1$ $\rhd$ [ $S_1$display ];
> [ $S_1$display ] $\oplus$ ( [ want to send X $\notin$ $S_1$ ] $\nabla$ A ) ### A $\oplus$ [ S.donothing ].

or, alternatively :

> [ $S_1$display ] $\oplus$ ( [ want to send X $\in$ $S_1$ ] $\nabla$ A ) ### A $\oplus$ [ S.press ].

( A complete description would undoubtedly be much more complicated, but working it out is not a fit job for a human. That's why the potential automatic generation of the details is attractive. )

Clearly, there must be intervening coding stages performed by A between N and its coded form, as the only vestige of N which appears in that description is "want to send X". In the case of the scanner interface, A knows ( because he knows the vocabulary of S ) that X cannot be any more complicated than a character or emphasis signal. A must therefore convert the original representation of N into a sequence of such items, and will therefore recast N in terms of words and emphasis, then reduce it to letters with emphasis as appropriate to the interface type. ( I have explored the application of the "walkthrough" method to these internal encoding operations a little[2], but really I doubt whether the mental processes are sufficiently clearly defined to justify using interactors – or anything else formal – in this part of the system. I shall nevertheless do it, just a bit, but at a descriptive level which I think commits me to practically nothing in psychological terms. )

I'll suppose ( in order to give me an excuse to introduce emphasis later ) that the first step, A's decision on how to express his desired notion, can be written :

> A $\oplus$ [ "Mary had a little lamb, but has one no longer" ] ###
> A $\oplus$ [ Mary_*had*_a_little_lamb ]

Notice that even this apparently innocuous transformation depends, albeit remotely, on the vocabulary offered by S. A must know the vocabulary, and express his notions in its terms. Some communication systems, particularly those used by very seriously disabled people, offer only a set of sentences; in such circumstances, if the idea you want to communicate doesn't fit in with any of the sentences, you cannot even take this first step.

This decision amounts to a plan for the subsequent communication. To put the plan into effect, A must convert it into a sequence of the available interface operations. It is reasonable to think of this process as one of stepwise decomposition, first into words then into characters then into physical interface operations. Other schemes could in principle be used, but at least the "words" level makes sense because A knows – again from his knowledge of S – that emphasis is associated with words. The result is a sequence of communication actions at the word level which we can describe like this :

A ⧐ [ lamb ] ⧐ [ little_ ] ⧐ [ a_ ] ⧐ [ had_.emphasis ] ⧐ [ Mary_ ]

In all but one case, the actions will be implemented as a sequence of character communications ( because that's all that A can do with the scanner – vocabulary once again ); but what do we do with [ had_.emphasis ] ?

I've written that in a form which reflects the structure of all three interfaces in that emphasis is associated with a word; there is no sense in which one can send individual italic characters. The next step, though, depends on the details of the interface design.

- **markup** :

  A ⧐ [ had_.emphasis ] ### A ⧐ [ _ ] ⧐ [ /em ] ⧐ [ had ] ⧐ [ em ]

  This is implemented as a simple expansion of the text; no new sorts of action are necessary, but the amount of text to be transmitted is increased. The further expansion of this text continues as for any other text. ( You could emphasise the final space if you wanted without coming to harm; I've been a bit fussy in the interests of pedantry. It would probably be better to regard spaces as separate items in the sentence expansion, but I was too lazy to do that. )

- **tagging the input** : operate the key once

  A ⧐ [ had_.emphasis ] ###
      A ⧐ [ _ ] ⧐ [ d ] ⧐ [ a ] ⧐ [ h.emphasis ] ###
      A ⧐ .... ⧐ [ operate the key once ] ⧐ [ operate the key twice ] ⧐ [ donothing ]

  The last line is schematic rather than precise; the intention is to carry down to the keypress level the requirement to "double click" at some point within the word "had". A more precise description would incorporate an element of indeterminacy, though anyone with any sense would in fact put in the emphasis as soon as possible. The sequence of key operations is determined by the clock, as described earlier.

- **separate channel** :

  A ⧐ [ had_.emphasis ] ###
      A ⧐ S.[ _ ] ⧐ ( ( S.[ d ] ⧐ S.[ a ] ⧐ S.[ h ] ) ⊕ ( B.[ press ] ) )

  This time, it is necessary to distinguish beteeen two output channels, S ( the scanner switch ) and B ( the emphasis button ). It is essential that the button be pressed while the word is being transmitted through the scanner proper; I have tried to suggest that by the grouping used in the last line, but in general some sort of temporal operators are probably desirable.

In each case, given a degree of goodwill, the operation is easy to translate into a more formal expression, though the proviso made earlier still holds : to do it properly, you need a machine.

One could then continue to describe how the scanner software dealt with the keypresses received, and how the emphasis signal should be handled in the different cases. Though the final steps require synthesis rather than analysis, I don't think that any new principles are involved, and I think the description given above is sufficient to illustrate the nature of the treatment which I have in mind.

## SOME NOTES.

These are observations on my ( wholly gedanken ) experiments. They are approximately in chronological order, which is of very little significance.

- **Vocabularies :** each channel of communication is associated with its own vocabulary. This is well modelled in the interactor notation by the named objects used to describe private interaction, with the name interpreted as the name of the vocabulary used. In effect, a private communication is one which is expressed in a vocabulary known only to the parties concerned.

  A contrary argument is that a vocabulary, being very much something "that can be known, stored, manipulated, or combined" ( David's paper, page 5 ) should be regarded as an object with properties of its own, qualified by a name when necessary to identify the particular vocabulary required in a context. One would then have ( for example ) scanner.vocabulary, human.vocabulary, typewriter.vocabulary, etc.

- **Manuals :** How does the sender acquire the receiver's vocabulary ? – in this example, through a "users' manual", which is a special example of a communication from system to user. (€More generally, and at a more immediate and informal level, such indications of "how to do it" which pass from system to user are affordances. ) The receiving interactor must therefore have (€at least ) two components, the implementation and the description. The description in turn has (€at least ) three components :

  – what can be received, which defines the vocabulary available (€e.g., any text );
  – modes in which it can be presented to the interface (€e.g., normal or emphasised ); and
  – how to present it, which gives operating instructions (€e.g., press this key then this key ... ).

  Without this information, the sender cannot decide whether the interface is capable of performing the required functions, nor work out how his message should be encoded, nor plan the actions necessary for the presentation.

  Whether such interactions as manuals come into the scope of interactors as they're discussed in David's paper, I'm not sure – though I don't see why they shouldn't, and affordances surely must. It would certainly be good to include them in a design technique to draw attention to the need for both documentation and affordances. There's a hint of an example in the next note.

- **Presentation :** To present a message at an interface, the sender must first acquire and interpret the description received from the receiver, and then encode the desired message in terms of the interface vocabulary. The overall process for a typewriter ( chosen only because it's easier than a scanner ) is described by ( something like ) :

  sender $\oplus$ typewriter.description ###
  sender $\nabla$ typewriter.description = typist;

  ( That is, the sender receives the typewriter documentation, and, having interpreted it, becomes a typist who can now use the information about the typewriter to decide on further action. )

typist ⊕ (€Mary_had_a_*little*_lamb ) ###
        typist ⊕ [ description.(€Mary_had_a_*little*_lamb ) ] ###
        typist ⊳ [ description.(€Mary_had_a_*little*_lamb ) ]

The translation from the original object, Mary_had_a_*little*_lamb, into the equivalent form encoded for the typewriter must proceed in several stages. The sequence for a typewriter is very partially described in the scheme below ( with the details of emphasis carefully omitted ); for the scanner, there would be at least one more level reducing the operation to presses of a single button and waits for feedback events from the receiver. There is some uncertainly in just what I mean by "description" in the scheme, for, while the first item is clearly an object proper to the human sender, the resulting action in step 6, while still expressed in human terms, is designed to conform to the interface's requirements. Questions of this nature would have to be addressed in any attempts at a thorough treatment.

1 : description.(€Mary_had_a_*little*_lamb. ) ###
2 : description.( ( Mary ) o⊳(€had ) o⊳(€a ) o⊳(€*little* ) o⊳(€lamb. ) ) ###
3 : description.(€(€M ) o⊳(€a ) o⊳(€r ) o⊳(€y ) ..... ) ###
4 : description.(€(€m.uppercase ) o⊳(€a.lowercase ) ..... ) ###
5 : description.(€(€m.lowercase + shift ) ..... ) ###

– while the act of presentation involves the further step :

6 : description.(€(€(€move left hand onto shift ) + (€move right hand above m ) ) o⊳ (€press ) ..... )

(€That's an illustration, not an exhaustive account of the operation. ) Notice that information from the receiver guides each step of the transformation, and that all the transformations are governed by rewriting rules. The information provided by the receiver can be presented before the transmission begins ( the manual ) or as part of the interaction ( the scanner feedback ). Notice too that all the objects in that scheme are to be regarded as incrementally associated with the sender object, resulting in the object which really performs the interaction, denoted earlier as "typist ⊕ [ description.(€Mary_had_a_*little*_lamb ) ]".

"o⊳" is a sequence operator, which I think is essential. (€There doesn't seem to be anything which does quite that job in David's notation. "⊕" doesn't imply sequence – indeed, it's explicitly stated to be commutative – while "∘" produces an interactor; ⊳ can manage sequential presentation, but I wanted something which would describe a sequential message being planned, not necessarily presented. ) It might well be possible to define o⊳ in terms of existing notions, perhaps by using the nr(€• ) operator.

- **Hierarchy :** What is the sender "really" presenting in that example ? The sender thinks that it's the original message (€1 ), but the interface thinks it's at the lowest level of detail (€6 ). This is the root of my problem : the stupid communication system must be made to transmit an intelligent message which it cannot comprehend. It seems best to regard all these levels as being presented simultaneously by the sender ( that's why I used the ### operator in the scheme above ), and received selectively according to the capability of the receiver.

    My more descriptive approach was intended to address the problems raised by this selectivity. In effect, it is a way of keeping track of all the levels through the system components in which only the lower levels can be represented, and ensuring that enough information is carried in one form or another to reconstruct the higher levels when required.

    Whether or not the stratified view makes sense depends on the structure of the component concerned. For a human-computer interface, by definition, it's all right for the people, but not necessarily for the mechanical partner to the interaction. The nature of a typewriter is such that it cannot "perceive" more than a single character at a time, because its internal state is not sufficiently complex to be able to represent more than one character. The hierarchy is not simply a matter of representation, though; a buffer might be able to represent a sequence of characters such as (€M ) o>(€a ) o>(€r ) o>(€y ), but that is not the same as (€Mary ), because there is nothing in the system to which one could attach attributes of (€Mary ). In other words, there is no way for any system object to be identified with the object ( Mary ). A symbol table can represent (€Mary ). A parser is an interactor which uses the receiver's vocabulary description to interpret (€M ) o> (€a ) o>(€r ) o>(€y ) as ( Mary ). A simple symbol table would not represent "little" and "*little*" as different entities, though – given the emphasis information – it is a simple matter to devise a symbol table which would preserve the difference; alternatively, and probably more practicably, with the addition of an attribute list emphasis could be recorded as an attribute of an *instance* of "little".

There is a notion here of a link between complexity of some sort and interpretation. A typewriter will never be able to represent "Mary"; a buffer can represent, but not interpret, "Mary", but a parser can interpret "Mary". It follows (€for example ) that a speech synthesiser at the other end of the system must be at least as complex as a parser in order to associate the pronunciation of "Mary" with the whole word, and must do rather better adequately to deal with emphasis.

- **Agents :** Agents appear briefly (€David's paper, page 6 ) as a subclass of objects. An agent can become an interactor by receiving an object. It seems to me that this overloads the notion of "object", and that the natural interpretation of "interactor" doesn't fit well with the definition of object : "anything that can be known, stored, manipulated, or combined into larger structures" ( page 5 ). The definition seems to imply that an object is cognitive rather than physical in nature, and passive rather than active; agents need not be physical things (€they could be software – parsers, for example ), but they are active rather than passive.

  Because of this, some aspects of the notation seem to be strained. For example, the receptor operator ° does not work consistently. Consider the expression Q ° (€P ° A ) (€page 6 ), where Q and P are percepts, and A is an agent – all are objects. P ° A is {€A, having received P€}; it is not unreasonable that the resulting modified agent should be able to receive Q. (€The agent *is* modified, so the implied question isn't silly : an agent like a typewriter can't accommodate two percepts – it can only handle one percept at a time, and cannot receive a second until it has forgotten the first. ) But, though Q and P are both objects, Q ° P has no very obvious interpretation. While that doesn't make the notation faulty in principle, as there is no need for the operator to be defined for all pairs of objects, it does seem that it only makes sense when the operands are an agent and a percept, suggesting a significant difference.

  If interactors (€agents ) are to be identified with any of the basic concepts of the theory, it could be argued that they partake of some of the properties of names. The essence of communication is to transfer an idea from one agent to another. In terms of the notation in David's paper :

  ( [idea] ⊕ A ) ⊕ (€B ) ### (€[idea] ⊕ A ) ⊕ ( [idea] ⊕ B ).

  Alternatively, regarding A and B as names, and naming the two instances of [idea] to ensure that they are distinct :

  A.[idea] ### A.[idea] ⊕ B.[idea].

  Even in the previous formulation, the two [idea]s on the right-hand side should perhaps be distinguished in some way, so at least A and B should have names :

  (€*<u>A</u>*.[idea] ⊕ A ) ⊕ (€B ) ### (€*<u>A</u>*.[idea] ⊕ A ) ⊕ (€*<u>B</u>*.[idea] ⊕ B ), where I have written $\underline{A}$ for the name of A. This is unconstrained speculation, and is not necessarily productive, but it illustrates my unease about the status of interactors. Is it necessary to introduce an explicit notion of "agent" into the theory ?

Another reason for doing so is to provide for the complexity of an interactor. If my discussion is to work, I have to be able to associate certain properties with interactors. Each interactor has the two components implementation and description, which includes the language specification. I also want to be able to state that a typewriter has a capacity of one low-level percept, a buffer can hold several low-level percepts, a symbol table can hold several low-level percepts and a parser can associate them into higher-level percepts, and so on. On the other hand, it seems possible that these more complex entities could be constructed from the simpler notions, or perhaps regarded as specialisations thereof, which would be preferable.

## NOT SO MUCH A CONCLUSION, MORE A SORT OF GUESS.

In working through these ideas, I've been impressed at how well and how smoothly the ideas of interactors, as I understand them, have fitted in with my previous descriptions of the systems and processes concerned.

I've introduced a few notions which seem to me to help to link the basic descriptions in David's paper with interfaces as I know them. I would have to review these carefully if I wanted to proceed with this development ( which I think I do ). It's possible that they can be reduced to the basic interactor calculus as described by David.

I haven't ventured on a formal definition of the scanner system, because, while I can usually make sense of pronouncements expressed in formal languages such as Z, I'm not sufficiently confident ( yet ) to try to speak it. In general terms, though, and assuming that some of my earlier comments can be resolved, I would expect it to be straightforward. Assuming that's so, I think it would be well worth investigating as the basis of a design technique for rehabilitation systems, where matters such as matching people's abilities to devices and getting optimum performance out of limited channels are of importance. A formal approach to this problem would be far superior to my informal method[2], as it would be much harder to leave out important factors.

It is clear from the fragments I've shown that a complete description of a communication in the system would be exceedingly cumbersome. It is equally clear, though, that that doesn't matter, because the details are fully defined by the processes of translation from one vocabulary to another. A full account of what happens in a communication can therefore be elaborated in full quite automatically.

In the long term, one could foresee several further developments. For example, the potential reduction of all interactions to their basic components raises the prospect of direct predictions of effective performance through a GOMS-like approach, and this would certainly be a valuable tool in practice.

## REFERENCES.

1 : G.A. Creak, R. Sheehan : *The representation of information in rehabilitation computing*, Auckland Computer Science Report #54, Auckland University Computer Science Department, July 1991.

2 : G.A. Creak : *Reaching Beyond Words In Rehabilitation Computer Systems*, unpublished Working Note AC96 ( May, 1996 ).

3 : David Duke : *A calculus for interactors*, notes for a seminar, 25 March 1996.

4 : J. Preece, Y. Rogers, H. Sharp, D. Benyon, S. Holland, T. Carey : *Human-Computer Interaction* ( Addison-Wesley,1994 ), page 679.

5 : G.A. Creak : *A view of rehabilitation computing*, Auckland Computer Science Report #46, Auckland University Computer Science Department, August, 1990.