SOME NOTES ON THE REPRESENTATION OF THE USER'S CONCEPTUAL MODEL

Micheal Mac an Airchinnigh
Trinity College
University of Dublin, Ireland

NOTE 1. The M-R-C Notation

> "There seems to be no reason to believe that we are ever acquainted with other people's minds, seeing that these are not directly perceived." - Bertrand Russell (1956) p.42

In November 1983 a Workshop on User Interface Management Systems was held under the auspices of IFIP WG5.2 and EUROGRAPHICS at Seeheim, FR Germany. At that meeting a working group studied the User's Conceptual Model. As a basis for discussion a preliminary version of the M-R-C Notation was proposed. The following is a revised account of that notation (Seeheim 1983).

Let u denote some user and let e denote some entity. Then M(u,e) denotes the mental model that u has of e. M(u,e) is purely a mental or conceptual model and, as such, cannot be directly perceived. In order to discuss M(u,e) we must find a way to externalise it. Let L denote some language. Then the externalisation of M is given by the representation R(M,L). Possible candidates for L are (1) the user's native natural language, (2) some other natural language known to the user, (3) the language of mathematics, (4) the language of first order predicate calculus, (5) etc. The Seeheim working group chose L to be conceptual graph notation (Sowa 1983). To be useful in a 'computable' sense, we must be able to encode this representation R in some programming language P. We denote this encoding by C(R,P). Again there are many possible candidates for P ranging from assembly programming language to, say, Ada (R). It is envisaged that the mapping from M to R and the mapping from R to C will ever only be approximate. This is expressed by the descending chain:

$$\begin{array}{ccccc} M & \sqsupseteq & R & \sqsupseteq & C \\ \| & & \| & & \| \\ M(u,e) & & R(M,L) & & C(R,P) \end{array}$$

Figure 1

where the symbol $\sqsupseteq$ denotes 'is approximated by'. Considering M(u,c), M(u,u), and M(u,w), where c denotes a computer system and w denotes the world, we can see that the specification of the range of entities e needs careful analysis. This is currently under consideration. An observation on the concept of 'is approximated by' is also in order. For example, the notion of real number is expressible in mathematical notation. However, it is impossible to capture that notion completely in any encoding. In this case the discipline of numerical analysis is used to measure the degree of approximation between R and C. An obvious need exists for a general discipline to measure the degree of approximation between the representation of arbitrary notions expressible in R and which are encoded

in C. The theory of abstract data types is a possible basis for such a discipline. That R is an approximation of M seems intuitively obvious. As to the measure of that approximation, one is faced with a much more difficult task.

NOTE 2. Representations

> "First and foremost, conceptual graphs support a more direct mapping to and from natural language. Peano-Russell notation has no standard mapping to English, and textbooks on logic seldom do more than give a few examples and expect the readers to invent their own conventions." - John F. Sowa (1983) p.149

R(M,L) denotes the representation R of the user's conceptual model M using some language L. In note 1 I have suggested some possible candidates for L. Let us look at the concept of 'natural number'. To indicate that u has a mental model of such a concept we will write M(u,'natural number'). Immediately, we see the difficulty in discussing mental models. It is impossible to be specific about any concepts without giving them expression. Thus, when I write 'natural number', one is supposed to unbind the concept from its expression. To emphasise such an unbinding I introduced the woolly diagram (1983). For 'natural number' the following woolly diagram is appropriate:

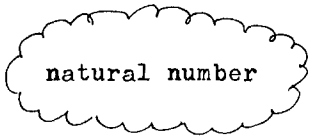

Figure 2

In conceptual graph notation the concept of 'natural number' is denoted by [NATURAL_NUMBER] . The intent is the same. Although Sowa uses English exclusively for conceptual graphs throughout his book, he does suggest that any other suitable language may be chosen. Explicit mention of which language is used for the representation of the concept is equivalent to the tagging of the woolly diagrams. The following concepts are all deemed to be equivalent: [NATURAL_NUMBER/English], [UIMHIR_AICEANTA/Irish] , [NOMBRE_NATUREL/French] , [NATURLICHE_ZAHL/German] , etc. To elaborate upon the concept of 'natural number' in, say, English, without using some form of mathematical notation is not easy. But of course, we all know what is a natural number and what is not a natural number. The use of mathematical notation 'formalises' the concept for us:

> 0 is a natural number
>
> Let n be a natural number.
> Then n + 1 is a natural number.

Figure 3

Ada is a registered trademark of the U.S. Department of Defense, ALO

However, this is not very rigorous. Following Peano, one can give an axiomatic specification of natural number:

Natural numbers is a Peano System

$$\langle N, ', 0 \rangle$$

where

P1. For all $x,y \in N$, if $x' = y'$, then $x = y$.

P2. For all $x \in N$, $x' \neq 0$.

P3. If S is any subset of N such that

(i) $0 \in S$ and
(ii) whenever $x \in S$ then $x' \in S$,

then $S = N$.

Figure 4

Naturally, very few users u would be familiar with such a representation. However, we can say that there is a relationship between R(M(u,'natural number'),English) and R(M(u,'natural number'),Mathematics), for some u. The Peano axiomatisation is not completely formal. Interspersed with the mathematical notation are some English keywords. To achieve total formality one might turn to a first-order language. In such a system, P1 might read

$$(\forall x)(\forall y)(=(S(x),S(y)) \rightarrow =(x,y))$$

Figure 5

where S (') denotes the successor function. Such a representation implies a relationship between R(M(u,'natural number'),Mathematics) and R(M(u, 'natural number'),First-order Language). We may continue in this fashion by considering other possible representations with different Ls. The important point is that there exists the concept of 'natural number' which is conceptually invariant under these different representations.

NOTE 3. Encodings

> "By means of detailed combinatorial studies ... the proposed characterizations of Turing and of Kleene, as well as those of Church, Post, Markov, and certain others, were all shown to be equivalent; that is to say, exactly the same class of partial functions (and hence of total functions) is obtained in each case." - Hartley Rogers, Jr. (1967) p.18

The Basic Result, Part I, taken from Rogers and quoted above forms the foundation for Church's Thesis. Applying it in a slightly different way, one can say that no matter what programming language is chosen, the class of representable concepts that can be encoded in such a programming language is always the same. The choice of programming language P is determined, not by its computational power, but rather by its expressive power. The history of the development of programming languages may be regarded as the history for the search of a computable expressive

notation that more clearly captures representable concepts. To illustrate some of the ideas developed thus far I will confine my attention to encodings of natural number in the programming language Ada (ALRM 1983). For each encoding presented I will show which aspects of the corresponding representation are involved.

A first approximation to the encoding of the concept of natural number may be obtained by using a straight forward type definition:

```
type NATURAL_NUMBER is new INTEGER range 0..SYSTEM.MAX_INT;

-- Annotation:
-- 1. The natural numbers form a subset of the integers;
-- 2. All of the operations on integers apply to natural numbers;
-- 3. There are only a finite number of natural numbers;
```

Figure 6

Those familiar with Pascal should recognise the similarity between the above Ada encoding and the comparable Pascal encoding. It is important to note that this approach entails an understanding of the concept of integers upon which the concept of natural numbers is derived. Secondly, we are relying on the INTEGER type predefined in Ada for our definition of NATURAL_NUMBER.

A second and better approximation may be obtained by using Ada's package structure:

```
package NATURAL_NUMBER_MODEL is
   type NATURAL_NUMBER is private;
   function ZERO return NATURAL_NUMBER;
   function SUCCESSOR(X: NATURAL_NUMBER) return NATURAL_NUMBER;
   -- ...
private
   -- FOR IMPLEMENTOR'S EYES ONLY!
end NATURAL_NUMBER_MODEL;
-- Annotation
-- 1. The implementation details of the type NATURAL_NUMBER
--    are not available to the user of the package;
-- 2. There is a unique NATURAL_NUMBER called ZERO;
-- 3. NATURAL_NUMBERs may be generated by the SUCCESSOR function;
-- 4. The equality predicate is supplied by the Ada language;
-- 5. The encoding presented is pure syntax;
```

Figure 7

In order that the Ada package may be used as an incarnation of an abstract data type we use 'limited private' instead of 'private'. Now the inbuilt equality predicate is no longer available and we must supply our own. The third approximation becomes:

```
package NATURAL_NUMBER_MODEL is
   type NATURAL_NUMBER is limited private;
   -- ZERO and SUCCESSOR functions as above;
   function EQUAL(X,Y: NATURAL_NUMBER) return BOOLEAN;
   -- ...
end NATURAL_NUMBER_MODEL;
```

Figure 8

Finally, we would like to have addition and subtraction operators for NATURAL_NUMBERS. Thus we would include the following functions:

```
function "+"(X,Y: NATURAL_NUMBER) return NATURAL_NUMBER;
function "-"(X,Y: NATURAL_NUMBER) return NATURAL_NUMBER;
```

Figure 9

As noted above, the Ada encodings are pure syntax. To augment the Ada code we must supply some form of semantics. One possible approach is to use the actual representations themselves as comments or documentation.

Though seemingly harmless, the previous sentence has important implications! The use of the natural language representation for this purpose has been commonplace in the past. But such a specification of the semantics of a piece of programming code by, say, an English text has been criticised on the grounds of being ambiguous or incomplete or informal. Then, how are the semantics of the English text to be given? Similarly, if we use the mathematical representation to characterise the semantics of the code, what are the semantics of the mathematical representation itself? Finally, if we use the first-order language representation to supply the semantics, we are assured that, at least, there is a mechanism by which we can supply that representation's semantics: "In sentential logic we had truth assignments to tell us which sentence symbols were to be interpreted as being true and which as false. In first-order logic the analogous role is played by structures, which can be thought of as providing translations from the formal language into English." (Enderton 1972) p.79. The issue of the semantics of the representations themselves is a thorny one and is currently under consideration. For the present, we may consider the semantics of the encoding to be given by one or more of the corresponding representations.

For the NATURAL_NUMBER_MODEL, a first attempt at providing semantics might be:

```
-- for all X,Y: NATURAL_NUMBER
--      1. NOT EQUAL(SUCCESSOR(X),ZERO)
--      2. NOT EQUAL(SUCCESSOR(X),X)
--      3. IMPLIES(EQUAL(SUCCESSOR(X),SUCCESSOR(Y)),EQUAL(X,Y))
```

Figure 10

Note that 1 above corresponds to Peano axiom P2 and 3 above corresponds to Peano axiom P1. 2 is extra and perhaps unnecessary. One also ought to include axioms for "+" and "-".

```
--      4. EQUAL(X+ZERO,X)
--      5. EQUAL(X+SUCCESSOR(Y),SUCCESSOR(X+Y))

--      6. IMPLIES(LESSTHAN(X,Y),EQUAL(X-Y,ZERO))
--      7. IMPLIES(NOT LESSTHAN(X,Y),
--              EQUAL(X-SUCCESSOR(Y),PREDECESSOR(X-Y)))
-- where
--      8. IMPLIES(EQUAL(X,ZERO),ZERO)
--      9. IMPLIES(NOT EQUAL(X,ZERO),
--              EQUAL(PREDECESSOR(SUCCESSOR(X)),X))
```

Figure 11

Observe that the axioms for "+" and "-" are not at all obvious. I have taken them from their primitive recursive definitions (Hermes 1969) p.64. Very few programmers will be in a position to read and understand such semantics, never mind write them themselves. Thus the importance of my earlier remark that the semantics ought to be provided by one or more of the corresponding representations. Furthermore, there would seem to be a natural order in the degree of formality of the said representations which would indicate the order in which the semantics would be provided. In other words, start with the natural language representation. If applicable, provide the mathematical representation and finally, the first-order language representation. Ambiguities, etc., that may arise at a particular level of representation ought to be resolved wherever possible at the next level of formality.

NOTE 4. Models and Language

> "A CLG representation of a system is a description of its user interface - the system as the user sees it and understands it. The psychological hypothesis behind CLG is that CLG describes the user's conceptual model of the system. Since this model is exactly what the designer of the user interface should be working with, CLG is also structured to be useful during the system design process." Thomas Moran (1981) p.5

I have used the term model for that which is conceptual and thus not directly perceived. Customarily, the term model is used by researchers in other disciplines for that which resides at the representational level. In an earlier paper I used the term model in both senses (1983). In other words, one can say that a given representation is, in a sense, a model of the actual user's conceptual model. Thus Moran's statement quoted above may be paraphrased to read "CLG provides a representation (model) of the user's conceptual model of the system". The essence of the hypothesis resides in the implication that the construction of a representation somehow forces the corresponding mental model into existence. The reality is that the mental model is constructed from many different representations. As well as the CLG representation, there is the user's prior experience, previous education, etc., all of which influence the building up of the mental model. Furthermore, the final syntax derived from the representation also plays its part in mental model formation. One ought not to underestimate the role played by syntax in this respect. It is through language that we perceive our world and form our concepts. For a natural language such as English, the meaning of a sentence is inextricably linked with the grammar of English. In this sense, one can speak about English as being a rich language in so far as it is self-sufficient with respect to its own semantics. But, of course, this richness does not prevent ambiguities arising. Resolution of the same is provided by dialogue. Artificial languages on the other hand, while having a grammar, do not contain within themselves their own semantics. These must be supplied in some other way. Ultimately, any such semantics will be traceable back to the user's natural language so that she can achieve understanding.

Given any command language whatsoever a user will, with practice, learn how to use it efficiently. Naturally, efficiency is directly correlated with frequency of use. Why then is there the concern for providing 'friendly' user interfaces? I argue that the issue depends completely on the notion of conceptual invariance and expressive power. I have applied these ideas already to the area of programming languages. Now I generalise to any (artificial) language. If the language syntax is alien or strange then a certain amount of translation must always be provided by the user. Of course, there is nothing new in this. We have learned

how to read English efficiently. In fact, we do it so well that we scarcely ever think of a translation process taking place. Moreover, the written form of the language corresponds exactly with the spoken form. With artificial languages there is the extra overhead of extracting the meaning from the alien form. A word of caution is necessary here. I am not making a case for the use of the user's natural language wherever possible. For example, it is very unlikely that we would ever have attained our current mathematical culture were it not for the special mathematical languages that have been developed. Similarly, I do not believe that we can make much progress in the development of user-computer interaction without comparable progress in the development of special user-interaction notations. The current research in interactive computer graphics clearly points in this direction. What we do require is a way of understanding the relation between the concepts of conceptual invariance and expressive power and any proposed language notation. Certainly, Iverson understood this principle and explains to some extent the continued existence and popularity of the programming language APL. Similar cases can be made for Lisp and Prolog. With the widespread use of voice input, I would not expect to see a return of natural language as either the primary or most 'friendly' language for communication with a computer system. On the other hand, for the so-called 'computer-naive' or 'casual' users, voice input may well attain that position.

As well as conceptual invariance and expressive power, the choice of language clearly depends on the application domain in question. In other words, there is a direct correspondence between the problems that a user wants to solve and the notation at her disposal. Again, this point is clearly illustrated by the use of mathematics. Task analysis is the phrase used to address 'problem solving' specification with respect to user-computer interaction. Finally, all task analysis is directly related to the user's conceptual model and so we come back to our point of departure.

Conclusion

These few short notes on the representation of the user's conceptual model suggest the direction of my research in this area since the publication of the two working papers that have appeared already in the SIGCHI Bulletin (1982, 1983). The distinction between model, representation, and encoding, seems to be a fruitful one. The three notions of conceptual invariance, expressive power, and task analysis are central. One problem that clearly stands out is that of measuring the approximation relation between the model and the representation and between the representation and the encoding.

Acknowledgement

I am indebted to the members of the working group on the user's conceptual model at the IFIP WG5.2/EUROGRAPHICS Workshop on User Interface Management Systems for the provision of the M-R-C notation which now supercedes my earlier M notation. They are Richard Guedj, Jan Matthys, John Sibert, Henning Hanusa, and Dimiter Novatchev. The improved M-R-C notation presented in this paper results from a critical review by my colleague Hans-Juergen Kugler.

References

1. Reference Manual for the Ada Programming Language, ANSI/MIL-STD 1815A, Alsys (La Celle-Saint-Cloud, France) 1983.
2. Enderton, Herbert B., A Mathematical Introduction to Logic, Academic Press (New York and London) 1972.
3. Hermes, Hans, Enumerability, Decidability, Computability - An Introduction to the Theory of Recursive Functions, (trans.) Hermann, G.T. and Plassmann, O., Springer-Verlag (Berlin, Heidelberg, New York) 1969.
4. Mac an Airchinnigh, Mícheál, Classifying the User, SIGCHI Bulletin, vol.14 no.2 (October 1982) p.3-8.
5. Mac an Airchinnigh, Mícheál, The $\mathfrak{L}$-concept, SIGCHI Bulletin, vol.14 no.4 (April 1983) p.8-15.
6. Mac an Airchinnigh, Mícheál, The Model of the User's Conceptual Model of ..., in: Proc. of the IFIP WG5.2/EUROGRAPHICS Workshop on User Interface Management Systems, Seeheim, FR Germany, November 1983, to appear.
7. Moran, Thomas P., The Command Language Grammar: a representation for the user interface of interactive computer systems, Int. J. Man-Machine Studies 15 (1981) p.3-50.
8. Rogers Jr., Hartley, Theory of Recursive Functions and Effective Computability, Mc-Graw Hill Book Company (New York) 1967.
9. Russell, Bertrand, On Denoting, in: Marsh R.C. (ed.), Logic and Knowledge, Allen and Unwin (London) 1956.
10. Proc. of the IFIP WG5.2/EUROGRAPHICS Workshop on User Interface Management Systems, Seeheim, FR Germany, November 1983, to appear.
11. Sowa, John F., Conceptual Structures - Information Processing in Mind and Machine, Addison-Wesley Publishing Company (London) 1983.