3/13/91

> To: CLIDT, CLIPCM Types (I mean human types!)
> From: Ed Greengrass
> Re: Types, Subsetting, Levels of Concern

I have been following with interest the discussion by Ed Barkmeyer and Brian Meek regarding CLIDT types vs. possible CLIPCM subsets, levels of abstraction, levels of concern, etc. I'd like to weigh in by building on (elaborating?, clarifying?, obscuring?) these ideas. Characteristically, I'd like to start with a "for instance".

Since, Brian and Ed have disagreed about State vs. Enumerated (and I did too, in a private communication to Ed), let's consider a case where program A calls procedure (or server) P and one of the operands (OP2) required by P is a value of a type (call it Height) that takes one of the ordered values Tiny, Short, Medium, and Tall. P is in some sense a public resource that advertises its availability, its capabilities and its requirements. The latter includes the datatypes of its operands.

Now, Ed speaks about three "levels of concern": the "user" level (what I mean), the "language" level (what I can write) and the "interface" level (what I can write) and the "interface" level (what I can exchange). Let me examine these levels in terms of my example.

I would argue that P advertises for its potential users. A user may see the advertisement on a PC screen. He may invoke P through a screen form. Or, he may write an application program that calls P. In either case (and others one can easily think of), he is a "user". The advertisement is written at the user level. The user invokes P from a user perspective. What this means is that the advertisement does NOT describe OP2 this way: OP2 is an Integer representing Height where 1 means Tiny, 2 means Short, 3 means Medium, and 4 means Tall. The relationship of Tiny to 1, etc, is exactly the kind of coding detail that a user does not want to be bothered with. Hence, the advertisement (if it uses CLIDT terminology and notation) defines Height as an Enumerated type having values Tiny, Short, Medium and Tall in that order. (Some users may not want to be bothered with terms like Enumerated type either but that's OK. Effectively, that's what it is.) In other words, the user of CLIPCM (whether he writes procedures or calls them) wants the full flexibility of CLIDT, including both State and Enumerated, to express what he means.

Now what happens when a CLIPCM service builds a Protocol Data Unit (PDU) to carry a procedure call to P? How does it represent OP2 of type Height? Let's answer that in two stages.

First, how does one exchange a value of Height in general, i.e., between a general sender and a general receiver? Answer: Logically, one is exchanging one of the values Tiny, Short, Medium or Tall. Hence, one must exchange one of those values tagged with the type (family in CLIDT WD 4) Enumerated and the specific type within the family, Height. Now, of course, one can encode these three units of information, e.g., Enumerated, Height, and say Short for exchange. In particular, one may choose to encode the value as an integer: 1 for Tiny, etc. Note that Height may be encoded As the integer range 0 to 3 within the sender and as the integer range 1 to 4 within the receiver, etc. For exchange purposes, some standard encoding is needed, e.g., integers in ascending order starting with 1. In order for a receiver to make sense of the value 2, i.e., to know that it means Short, it needs to know that the value is family Enumerated, type Height and that the exchange encoding begins with 1, etc. Hence, the general requirement is to exchange Enumerated. Enumerated may be encoded as an integer provided that it is properly tagged with its type.

Second, how does CLIPCM encode its PDU? For the specialized case of a procedure call PDU, encoding shortcuts may be permissible. The PDU will certainly identify the procedure being called. The encoding mechanism knows that P expects the 2nd operand to be a value of Height. Hence, it may be OK to drop the tags for Enumerated and Height. (In the same way, a value of a generated record of type

Personnel_Record might not have to carry the names and types of its fields, if these are well-advertised: it would be sufficient to tag the record as type Personnel_Record of Generator Record. And, if the record value were operand I of a procedure that required a well-advertised type of personnel record for its Ith operand, one could dispense with that tag too.) But, this is just an encoding shortcut. Such shortcuts occur at what Ed calls the "interchange" level. Also, as Ed has noted, this shortcut breaks down in the case of a Choice. Indeed, it breaks down in any general exchange case where the units to be exchanged are not constrained by a relatively fixed protocol, e.g., a user browsing through a complex object-oriented database.

One can view the relationship of CLIDT to the exchange encoding as analogous to the relationship of ASN.1 abstract syntax (ISO 8824) to an encoding such as ISO 8825. The (explicit or implicit) tags Enumerated and Height are just values during the exchange but at the sending and receiving end (outward and inward mappings) they must be "understood" and interpreted. Hence, at both ends it is more than just a label; it has meaning to whoever (or whatever) does the mapping. The more fixed the protocol, the more compact the encoding can be because more of its schema is fixed and understood at both ends in advance.

Best Wishes,
Ed Greengrass