From: Ed Greengrass

To: WG11

Subject: Response to Turba's Comments on CLIDT WD 5 (WG11/N262), part 2 of 3

Date: Mon, 15 Jul 91 11:36:34 -0400

****

(6) Turba: None of the datatype concepts of object-oriented programming, e.g., subclassing and inheritance, are addressed in this document. This is extremely unfortunate, especially when common terms such as "subtype" are used in this area but have a definition that is more inclusive.

****

I agree about the importance of object-oriented (OO) concepts. How does this apply to CLIDT? In several papers, SC22/WG11 N232 (X3T2/91-072), SC22/WG11 N247, and SC22/WG11 N258, I have discussed and advocated an Object datatype where the term "Object" is used in the OO sense. Objects are distinguished from simple values of CLIDT datatypes by the fact that they possess unique identities. (There is only one Integer of value "3" but there may be many distinct Integer objects of value 3.) Thus, an OO "Class" is a collection of objects of some type T. It follows that there is a fundamental difference between the CLIDT concept of "subtype" and the OO concept of "subclass". Subtypes partition a domain of values. Subclasses partition a domain of objects. (In N247 and N258, I discuss the idea of a class of objects as a Table whose key is Unique ID (UID). In terms of this idea, a subclass is a subset of the rows of the given Table.)

Since generators are parameterized, they provide one kind of inheritance. For example, if Free Form Character String = List of Character, then Free Form Character String inherits the properties of List. More generally, if class B inherits from class A, then B may have the characterizing operations of A plus additional characterizing operations. Or, B may have characterizing operations that refine those of A. This is a trickier concept. But, for "strict" inheritance, the effect must be that the operations of B generate no objects that are not also in the class A. In other words, the additional properties that distinguish objects of class B should not contradict its properties as an object of class A. A List of Characters is still a List. (Inheritance as defined here can apply to values as well as objects.)

****

(7) Turba: Requiring that Choice be a required datatype generator seems a bit strong, especially for languages like FORTRAN that only indirectly support the concept. What is the rationale for requiring it? Also, why are Lists a required datatype generator? They are not nearly as essential as records or arrays, and many languages do not directly support them.

****

Answer:

These questions presuppose that CLIDT is intended to provide only a least common denominator for all widely used languages. On the contrary, CLIDT should provide support for all the types of all of these languages. The fact that, e.g., Ada, supports Choice, and e.g., LISP supports List, are sufficient reasons for supporting those types or something equivalent. Moreover, CLIDT should support the ability for diverse languages to communicate. Mappings between diverse languages need not be one-one. A List might be supported directly in one language, e.g., LISP, and indirectly through Pointers in another. A Character String (more precisely, a free form Character String may be defined as List of Character. And so on.

One should also bear in mind the important trend toward languages in which new datatypes can be defined. And, in practice, applications like Language-Independent Procedure Calls, and Remote Procedure Calls require the ability of procedure definers (more generally, server definers) to advertise their services in terms of application-specific datatypes erected on top of a common datatype base. Hence, CLIDT should support fundamental types and generators (Set and List among them) whose generic properties lend themselves to defining other less generic types.

Indeed, CLIDT should provide support for a common vocabulary of datatypes in terms of which the datatypes of each language can be defined and explained to all the others.

Note that I have discussed Goals for CLIDT in a little more detail in X3T2/90-293.