From: Frank Farance
Organization: Farance Inc.
Telephone: +1 212 486 4700
Fax: +1 212 759 1605
E-mail: frank@farance.com
Date: 1995-12-22
Document Number: WG14/N529 X3J11/95-130
Subject: Issues on C Binding for LID

In the previous WG14/N463 document, ''Impact of adding WG11's LIA-1, LID, and LIPC features'', I had described adding LID (Language Independent Datatypes) as infeasible for C9X. Since then, WG14 had a presentation from Craig Schaffert of WG11 and I have had discussed these issues with him.

Briefly, LID specifies several datatype and datatype generation mechanisms. Each is described by its domain (a set of acceptable values), its operational properties (e.g., integers are: ordered, each, numeric, unbounded), and its characterizing operations (methods in object-oriented terminology).

The following datatypes are specified in LID:

> Primitive datatypes: boolean, state, enumerated, character, ordinal, date-and-time, integer, rational, scaled, real, complex, void.
>
> Subtypes: range, selecting, excluding, extended, size, explicit subtypes.
>
> Generated datatypes: choice, pointer, procedure.
>
> Aggregate datatypes: record, set, bag, sequence, array, table.
>
> Defined datatypes.

The following are the remaining outstanding issues for continuing work on a C binding for LID:

> (1) There is much commonality with these datatypes and the C++ class libraries being specified within WG21. In some cases retaining compatibility with C++ means using templates. Since C9X is unlikely to include templates, what will our strategy be: eliminate the functionality or build a different interface?
>
> (2) Even if we chose to take C++ class libraries (including templates), how do we resolve interfaces that are incompatible or incomplete with LID?
>
> (3) LID doesn't address promotion issues among different types (e.g., converting boolean to integer

or vice versa). How do we determine appropriate, if any, conversions?

(4) LID doesn't address promotion issues among different implementations of the same type (e.g., a date-time that specifies month, day, hour, minute, second vs. a date-time that includes the year, month, day). Do we leave this completely implementation-defined?

(5) How does the programmer learn of the limitations of each of the implementations of the datatypes. This would serve a purpose similar to what LIA-1 does for arithmetic types: determining the implementation characteristics of the operands and the operators.

While it's attractive to jump into the details of specifying these new types, we should resolve these issues first so that we can focus our efforts properly.