
Babel Structs Discussion

Tom Epperly

Center for Applied Scientific Computing

Common Component Architecture Working Group

January 22, 2004



This work was performed under the auspices of the U.S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under Contract No. W-7405-Eng-48.

UCRL-PRES-201938



Adding structs to SIDL/Babel

- Give Babel something analogous to C structs or Fortran 90 derived types

```
struct date_t {  
    int month;  
    int day;  
    int year;  
}
```

- Simple grouping of data – no methods

SIDL Class Equivalent (de facto approach)

```
class Date{
    int getMonth();
    void setMonth(in int month);
    int getDay();
    void setDay(in int day);
    int getYear();
    void setYear(in int year);
}
```

- Requires 10-20 times as many hand written lines as the same thing in C
- Multi-language method overhead for each access
- Data hiding (positive)

Objectives

- **Performance**
 - **avoid copying and multi-language overhead (assumes most of the struct fields are needed)**
- **Development**
 - **avoid 10-20x penalty for the de facto approach**
- **Completeness**
 - **support a common programming construct**
- **Compatibility**
 - **CORBA & WSDL have struct concepts**

Easy & hard types for structs

- **Easy types**

- int
- long
- float
- double
- fcomplex
- dcomplex

- **Predictable size & encoding**

- **Hard**

- bool
- string
- object/interface
- array
- opaque
- struct

- **Variable size & encoding**

Four alternatives

- **Do nothing (de facto approach)**
- **Generate impls**
- **Copy & marshal**
- **Minimal copy & marshal**

Do nothing (de facto approach)

- **Nothing added to SIDL/Babel**
- **Document & promote the class approach**

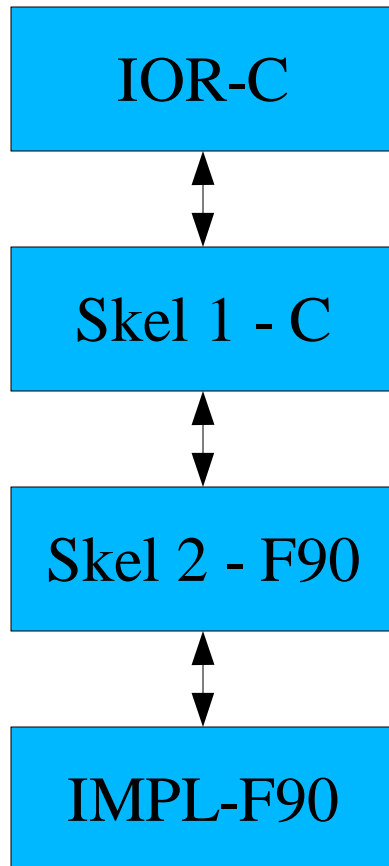
Automatically generate IMPLs

- **Add struct to SIDL as a short cut method for making a class**
- **Automatically generate IMPL files and serialization code (for RMI)**
- **A SIDL struct is treated just like a class with get/set methods for each data element**

Copy and marshal approach

- **Add struct to SIDL**
- **C & C++ access struct with no marshalling for easy types and marshalling for hard types**
- **F90 marshals everything by expanding argument list**
- **structs look native in C, C++, Python, Java (presumably) and F90**

Passing a struct to F90



















- **SIDL**
`void f(in Date d)`
- **Skel 1**
`void f_s(struct Date_t *d)`
- **Skel 2**
`subroutine f_m(d_day, &d_month, d_year)`
- **IMPL**
`subroutine f_i(d)`
`type(Date_t) :: d`

Minimal copy & marshal approach

- **Assume C structs & F90 derived types map to memory identically for easy types**
- **Copy & marshal for F90 only when hard types are used**

Evaluating alternatives

	Nothing	Auto impl	Copy/marshal	Min copy/marshal
Performance				
Development				
Completeness				
Compatibility				
Babel Dev	