CCA Components in Fortran 90 Using Babel

Tammy Dahlgren, Tom Epperly, and Gary Kumfert

Center for Applied Scientific Computing

Common Component Architecture Working Group

April 10, 2003

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-152699

Outline

- Objectives
- Steps to writing a CCA component in F90
  - Get cca.sidl
  - Write your SIDL file
  - Build CCA client-side library
  - Write your server-side library
  - Build your server-side library
- Feedback

Objectives

- Demonstrate Fortran 90 bindings
- Elicit feedback on Fortran 90 bindings
- Elicit feedback on Fortran 90 component writing
- Provide hints on building Fortran 90 bindings

SIDL files

- cca.sidl is distributed with babel
  babel-0.8.4/examples/cca/cca.sidl
- Write your SIDL file
  - Using tutorial example

Driver SIDL file

```
package tutorial version 1.0 {

  class Driver implements-all gov.cca.ports.GoPort,
                              gov.cca.Component
  {
  }

}
```

Function SIDL file

```
package functions version 1.0 {

  interface Function extends gov.cca.Port
  {
    double evaluate(in double x);
  }

  class LinearFunction implements-all Function,
                                      gov.cca.Component
  {
    double evaluate(in double x);
  }

  class NonlinearFunction implements-all Function,
                                         gov.cca.Component
  {
  }

  class PiFunction implements-all Function,
                                   gov.cca.Component
  {
  }

}
```

Integrators SIDL file

```
package integrators version 1.0 {

  interface Integrator extends gov.cca.Port
  {
    double integrate(in double lowBound, in double upBound, in int count);
  }

  class MonteCarloIntegrator implements-all Integrator,
                                            gov.cca.Component
  {
  }

  class MidpointIntegrator implements-all Integrator,
                                          gov.cca.Component
  {
  }

  class ParallelIntegrator implements-all Integrator,
                                          gov.cca.Component
  {
  }

}
```

Build CCA client-side library

- mkdir cca-client ; cd cca-client
- babel –client=f90 ../cca.sidl
- Build libcca.a or libcca.so
- Note: this library also includes the SIDL F90 bindings

CCA client lib GNU Makefile

```
# GNU Makefile
include babel.make
.SUFFIXES: .F90 .c .o

F90=pgf90
INCLUDES=-I/usr/casc/babel/babel-0.8.4/casc-linux-gcc2.96/include
F90FLAGS=-shared
CFLAGS=-shared $(INCLUDES)

IOROBJS  = $(IORSRCS:.c=.o)
STUBOBJS = $(STUBSRCS:.c=.o)
STUBMODULEOBJS = $(STUBMODULESRCS:.F90=.o)
ARRAYMODULEOBJS = $(ARRAYMODULESRCS:.F90=.o)
TYPEMODULEOBJS = $(TYPEMODULESRCS:.F90=.o)
SKELOBJS = $(SKELSRCS:.c=.o)
IMPLMODULEOBJS = $(IMPLMODULESRCS:.F90=.o)
IMPLOBJS = $(IMPLSRCS:.F90=.o)

libCCA.a : libCCA.a($(STUBOBJS) $(STUBMODULEOBJS) $(ARRAYMODULEOBJS) $(TYPEMODULEOBJS))

$(ARRAYMODULEOBJS) : $(TYPEMODULEOBJS)
$(STUBMODULEOBJS): $(TYPEMODULEOBJS) $(ARRAYMODULEOBJS)
$(IMPLOBJS) : $(STUBMODULEOBJS) $(IMPLMODULEOBJS)

.F90.o:
        $(CPP) -traditional $(INCLUDES) -P -o $(@:.o=.f90) -x c $<
        $(F90) $(F90FLAGS) $(INCLUDES) -c -o $@ $(@:.o=.f90); rm -f $(@:.o=.f90)
clean:
        rm -f *.f90 *.o *.mod libCCA.a
```

Write server-side library

- babel --server=f90 --exclude=”^SIDL.*” --exclude=”^gov.*” ../cca.sidl ../driver.sidl
- Edit tutorial_Driver_Mod.F90 (private object state)
- Edit tutorial_Driver_Impl.F90 (implementation)
  - Add #include's
  - Write constructor
  - Write destructor
  - Write setServices
  - Write go method

tutorial_Driver_Mod.F90

```
#include"tutorial_Driver_fAbbrev.h"
module tutorial_Driver_impl

! DO-NOT-DELETE splicer.begin(tutorial.Driver.use)
#include "gov_cca_Services_fAbbrev.h"
  use gov_cca_Services_type
! DO-NOT-DELETE splicer.end(tutorial.Driver.use)

type tutorial_Driver_private
  sequence
  ! DO-NOT-DELETE splicer.begin(tutorial.Driver.private_data)
  type(gov_cca_Services_t) :: d_services
  ! DO-NOT-DELETE splicer.end(tutorial.Driver.private_data)
end type tutorial_Driver_private

type tutorial_Driver_wrap
  sequence
  type(tutorial_Driver_private), pointer :: d_private_data
end type tutorial_Driver_wrap

end module tutorial_Driver_impl
```

tutorial_Driver_Impl.F90 #include

```
! DO-NOT-DELETE splicer.begin(_miscellaneous_code_start)
#include "integrators_Integrator_fAbbrev.h"
#include "SIDL_BaseException_fAbbrev.h"
! DO-NOT-DELETE splicer.end(_miscellaneous_code_start)
```

tutorial_Driver_Impl.F90 constructor

```
recursive subroutine tutorial_Driver__ctor_mi(self)
  use tutorial_Driver
  use tutorial_Driver_impl
  ! DO-NOT-DELETE splicer.begin(tutorial.Driver._ctor.use)
  use gov_cca_Services
  ! DO-NOT-DELETE splicer.end(tutorial.Driver._ctor.use)
  implicit none
  type(tutorial_Driver_t) :: self

! DO-NOT-DELETE splicer.begin(tutorial.Driver._ctor)
  type(tutorial_Driver_wrap) :: pd
  external tutorial_Driver__set_data_m
  allocate(pd%d_private_data)
  call set_null(pd%d_private_data%d_services)
  call tutorial_Driver__set_data_m(self, pd)
! DO-NOT-DELETE splicer.end(tutorial.Driver._ctor)
end subroutine tutorial_Driver__ctor_mi
```

tutorial_Driver_Impl.F90 destructor

```
recursive subroutine tutorial_Driver__dtor_mi(self)
  use tutorial_Driver
  use tutorial_Driver_impl
  ! DO-NOT-DELETE splicer.begin(tutorial.Driver._dtor.use)
  use gov_cca_Services
  ! DO-NOT-DELETE splicer.end(tutorial.Driver._dtor.use)
  implicit none
  type(tutorial_Driver_t) :: self

! DO-NOT-DELETE splicer.begin(tutorial.Driver._dtor)
  type(tutorial_Driver_wrap) :: pd
  external tutorial_Driver__get_data_m
  call tutorial_Driver__get_data_m(self, pd)
  if (not_null(pd%d_private_data%d_services)) then
     call deleteRef(pd%d_private_data%d_services)
     call set_null(pd%d_private_data%d_services)
  end if
  deallocate(pd%d_private_data)
! DO-NOT-DELETE splicer.end(tutorial.Driver._dtor)
end subroutine tutorial_Driver__dtor_mi
```

tutorial_Driver_Impl.F90 setServices

```
recursive subroutine tutorial_Driver_setServices_mi(self, services)
  use gov_cca_Services
  use tutorial_Driver
  use tutorial_Driver_impl
  ! DO-NOT-DELETE splicer.begin(tutorial.Driver.setServices.use)
  use gov_cca_Services
  use gov_cca_Port
  use SIDL_BaseException
  ! DO-NOT-DELETE splicer.end(tutorial.Driver.setServices.use)
  implicit none
  type(tutorial_Driver_t) :: self
  type(gov_cca_Services_t) :: services

! DO-NOT-DELETE splicer.begin(tutorial.Driver.setServices)
  type(tutorial_Driver_wrap) :: pd
  type(gov_cca_Port_t) :: myPort
  type(gov_cca_TypeMap_t) :: tm
  type(SIDL_BaseException_t) :: excpt
  external tutorial_Driver__get_data_m
  call tutorial_Driver__get_data_m(self, pd)
  if (not_null(pd%d_private_data%d_services)) then
     call deleteRef(pd%d_private_data%d_services)
  end if
  pd%d_private_data%d_services = services
```

tutorial_Driver_Impl.F90 setServices

```
  if (not_null(services)) then
     call addRef(services)
     call createTypeMap(services, tm, excpt)
     if (not_null(excpt)) then
        write(*,*) 'createTypeMap threw exception'
     end if
     call cast(self, myPort)
     call addProvidesPort(services, myPort, 'GoPort', &
          'gov.cca.ports.GoPort', tm, excpt)
     if (not_null(excpt)) then
        write(*,*) 'addProvidesPort threw exception'
     endif
     call registerUsesPort(services, 'IntegratorPort', &
          'integrators.Integrator', tm, excpt)
     if (not_null(excpt)) then
        write(*,*) 'registerUsesPort threw exception'
     endif
  endif
! DO-NOT-DELETE splicer.end(tutorial.Driver.setServices)
end subroutine tutorial_Driver_setServices_mi
```

tutorial_Driver_Impl.F90 go (part 1)

```
recursive subroutine tutorial_Driver_go_mi(self, retval)
  use tutorial_Driver
  use tutorial_Driver_impl
  ! DO-NOT-DELETE splicer.begin(tutorial.Driver.go.use)
  use gov_cca_Port
  use gov_cca_Services
  use integrators_Integrator
  use SIDL_BaseException
  ! DO-NOT-DELETE splicer.end(tutorial.Driver.go.use)
  implicit none
  type(tutorial_Driver_t) :: self
  integer (selected_int_kind(9)) :: retval
```

tutorial_Driver_Impl.F90 go (part 2)

```
! DO-NOT-DELETE splicer.begin(tutorial.Driver.go)
  real(selected_real_kind(15,307)) :: lowerBnd, upperBnd, result
  type(tutorial_Driver_wrap) :: pd
  type(gov_cca_Port_t) :: port
  type(integrators_Integrator_t) :: intPort
  type(SIDL_BaseException_t) :: excpt
  external tutorial_Driver__get_data_m
  call tutorial_Driver__get_data_m(self, pd)
  retval = -1
    call getPort(pd%d_private_data%d_services, 'IntegratorPort', port, excpt)
  if (is_null(port)) then
     print *, 'Driver: getPort() returns null port'
     return
  end if
  call cast(port, intPort)
  if (is_null(intPort)) then
     print *, 'Driver: cast() returns null intPort'
     return
  end if
  lowerBnd = 0.0
  upperBnd = 1.0
  call integrate(intPort, lowerBnd, upperBnd, 1000, result)
  print *, 'Value = ', result
  retval = 0
  call deleteRef(port)
  call releasePort(pd%d_private_data%d_services, 'IntegratorPort', excpt)
! DO-NOT-DELETE splicer.end(tutorial.Driver.go)
end subroutine tutorial_Driver_go_mi
```

Last step: build your server-side library

```
include babel.make
.SUFFIXES: .F90 .c .o
F90=pgf90
INCLUDES=-I../cca-client -I../integrator-f90 \
         -I/usr/casc/babel/babel-0.8.4/casc-linux-gcc2.96/include
F90FLAGS=-shared
CFLAGS=-shared $(INCLUDES)
LIBS=../cca-client/libCCA.a
IOROBJS  = $(IORSRCS:.c=.o)
STUBOBJS = $(STUBSRCS:.c=.o)
STUBMODULEOBJS = $(STUBMODULESRCS:.F90=.o)
ARRAYMODULEOBJS = $(ARRAYMODULESRCS:.F90=.o)
TYPEMODULEOBJS = $(TYPEMODULESRCS:.F90=.o)
SKELOBJS = $(SKELSRCS:.c=.o)
IMPLMODULEOBJS = $(IMPLMODULESRCS:.F90=.o)
IMPLOBJS = $(IMPLSRCS:.F90=.o)
libDriver.so : $(STUBOBJS) $(STUBMODULEOBJS) $(ARRAYMODULEOBJS) \
          $(TYPEMODULEOBJS) $(IMPLOBJS) $(IMPLMODULEOBJS) \
          $(SKELOBJS) $(IOROBJS)
         $(F90) -o $@ -shared $^ $(LIBS)
$(ARRAYMODULEOBJS) : $(TYPEMODULEOBJS)
$(STUBMODULEOBJS): $(TYPEMODULEOBJS) $(ARRAYMODULEOBJS)
$(IMPLOBJS) : $(STUBMODULEOBJS) $(IMPLMODULEOBJS)
.F90.o:
         $(CPP) -traditional $(INCLUDES) -P -o $(@:.o=.f90) -x c $<
         $(F90) $(F90FLAGS) $(INCLUDES) -c -o $@ $(@:.o=.f90)
         rm -f $(@:.o=.f90)
clean:
         rm -f *.f90 *.o *.mod libCCA.a
```

What do you think?

- Will this style be “natural enough” for F90 programmers?
- Do you have any improvements?