EFDA
EUROPEAN FUSION DEVELOPMENT AGREEMENT
Task Force
INTEGRATED TOKAMAK MODELLING

Tutorial/demonstration: Kepler for beginners

J. Signoret
& many contributors: Y. Frauel, B. Guillerminet, F.Imbeaux, G.Manduchi…

Introduction

- **Goal : build and execute a Kepler workflow**
- **Pre-requisites :**
  - Create a private database in your account
    - for a given machine (tokamak name) and data structure version :
      - /afs/efda-itm.eu/project/switm/scripts/create_user_itm_dir *TokamakName DataVersion*
      - Example : create a tree for tokamak name test (allowed for testing purposes)
        /afs/efda-itm.eu/project/switm/scripts/create_user_itm_dir test 4.08b
  - Copy KEPLER in your directory :
    - If there is a previous release of Kepler in your directory :
      - mv your_kepler your_kepler_save
      - remove the KEPLER cache : rm -rf ~/.kepler
    - get KEPLER :
      cd ~; tar xvf /afs/efda-itm.eu/project/switm/kepler/*xxxx*/kepler.tar
      where *xxxx* is the current release of the UAL
      This command creates in your home directory a subdirectory named kepler
  - Set the environment variables :
    - put in the configuration file .cshrc:
      source /afs/efda-itm.eu/project/switm/scripts/ITMv1 kepler TokamakName DataVersion
      NB : if this command is not included in the .cshrc file, it is mandatory to type it each time you open a new window

FORTRAN : Create a subroutine

- Identify the CPO in and out
- Create a subroutine with CPOs as argument
  - This example can be found
    ~signoret/public/workflow_examples/fortran

```
subroutine coreprof2mhd(coreprofin,mhdout)

use euITM_schemas
implicit none
integer,parameter :: DP=kind(1.0D0)
!
! Always describe cpo as array
! In case of time slice, the size of the input cpo is 1
!
type (type_coreprof),pointer :: coreprofin(:)
type (type_mhd),pointer :: mhdout(:)
integer :: i,j,k,l
!!!!!!!!!!!!!!!!!! Physics calculations
! The output CPO must be allocated with its number of time slices
! (1 for a single time slice physics module)
allocate(mhdout(size(coreprofin)))
! Fill in the output CPO (Physical data)
do i=1,size(coreprofin)
  ! Time : copy from input CPO
  mhdout(i)%time = coreprofin(i)%time  ! THE TIME FIELD MUST BE FILLED
(MANDATORY)
  ! Psi : copy from input CPO
  allocate(mhdout(i)%psi(size(coreprofin(i)%psi%value)))
  mhdout(i)%psi  = coreprofin(i)%psi%value
! Example of a physics quantity that would have been calculated by the module
  allocate(mhdout(i)%frequency(3))
  mhdout(i)%frequency(1) = 1.1D0
  mhdout(i)%frequency(2) = 1.2D0
  mhdout(i)%frequency(3) = 1.3D0
                          ……

  allocate(mhdout(i)%disp_perp(3,2,2));
  do j=1,3
     do k=1,2
        do l=1,2
           mhdout(i)%disp_perp(j,k,l)=i*1.0+k*(-1.2)+l;
        enddo
    enddo
  enddo
enddo
return
end subroutine
```

FORTRAN : testbed

- To make sure the code handles the CPOs correctly, run it in a “testbed”
  - create a main program which :
    - implements UAL calls to open, read, write, close the shot in the database
    - Calls the subroutine

Example :
~signoret/public/workflow_examples/fortran/standalone.f90

```
program test
use euITM_schemas
use euITM_routines
implicit none
integer,parameter :: DP=kind(1.0D0)

interface
subroutine coreprof2mhd(coreprofin,mhdout)
use euitm_schemas
type (type_coreprof),pointer :: coreprofin(:)
type (type_mhd),pointer :: mhdout(:)
end subroutine
end interface

type (type_coreprof),pointer :: coreprofin(:)
type (type_mhd),pointer :: mhdout(:)

integer :: idxin, idxout, shot, runin, runout, refshot, refrun
character(len=5)::treename

shot = 1983
runin = 1
runout = 2
refshot = 0 ! Dummy, not used
refrun =0   ! Dummy, not used
treename = 'euitm' ! Mandatory, do not change
```

Declare the subroutine

(continued)

```
write(*,*) 'Open shot in MDS !'
call euitm_open(treename,shot,runin,idxin)

write(*,*) 'Reading the input CPO :'
call euitm_get(idxin,"coreprof",coreprofin)

write(*,*) 'Calling the coreprof2mhd subroutine :'
call coreprof2mhd(coreprofin,mhdout)

write(*,*) 'Creating output run :'
! This is the UAL function that creates a shot in the ITM MDS+ tree
call euitm_create(treename,shot,runout,refshot,refrun,idxout)

write(*,*) 'Put result'
call euitm_put(idxout,"mhd",mhdout)

write(*,*) 'Closing Database :'
call euitm_close(idxin,treename,shot,runin)
call euitm_close(idxout,treename,shot,runout)

write(*,*) 'Deallocate CPOs :'
call euitm_deallocate(coreprofin)
call euitm_deallocate(mhdout)
end
```

FORTRAN : compilation and test

- Prepare a makefile to compile both the subroutine as a library and the standalone program

Example :
~signoret/public/workflow_examples/fortran/Makefile

```
F77=pgf90
F90=pgf90
CC=gcc
COPTS = -r8 -fPIC -Mnosecond_underscore -g

LIBS =   -L$(UAL)/lib  -lUALFORTRANInterface_pgi
INCLUDES = -I$(UAL)/include/amd64_pgi

all: coreprof2mhd.o standalone libcpo2cpof

libcpo2cpof: coreprof2mhd.o
		ar -rvs libcpo2cpof.a coreprof2mhd.o

standalone: standalone.f90
		$(F90) $(COPTS) -o $@ $^ coreprof2mhd.o ${INCLUDES} $(LIBS)

coreprof2mhd.o: coreprof2mhd.f90
		$(F90) $(COPTS) -c -o $@ $^ ${INCLUDES} $(LIBS)

clean:
		rm -f *.o  standalone *.a
```

-fPIC flag mandatory to make the library

-g flag for debugging

$UAL = path to the current version of the UAL

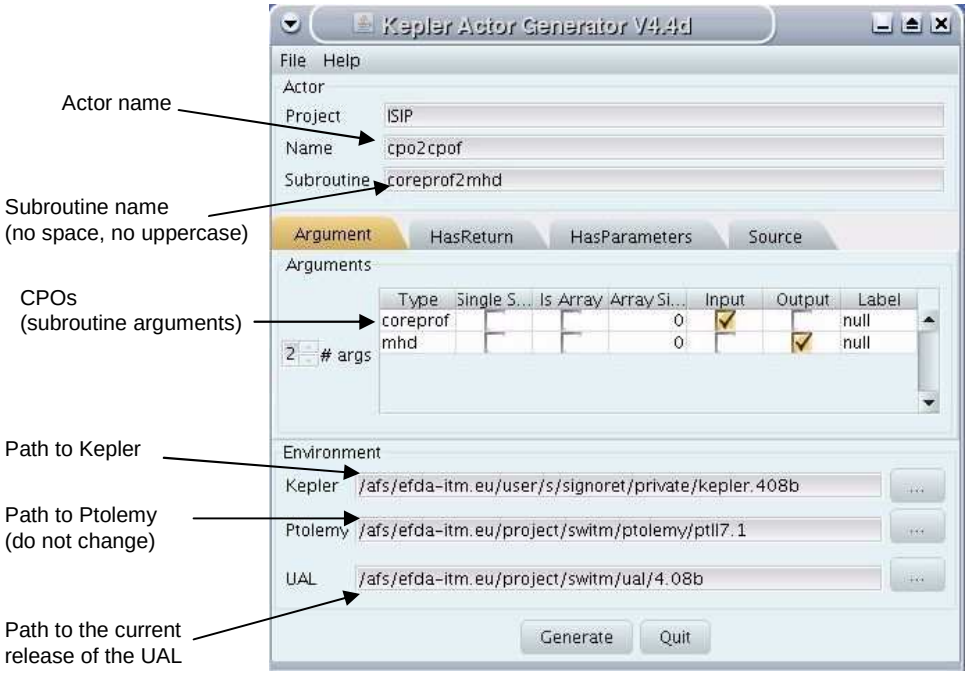
Building the library

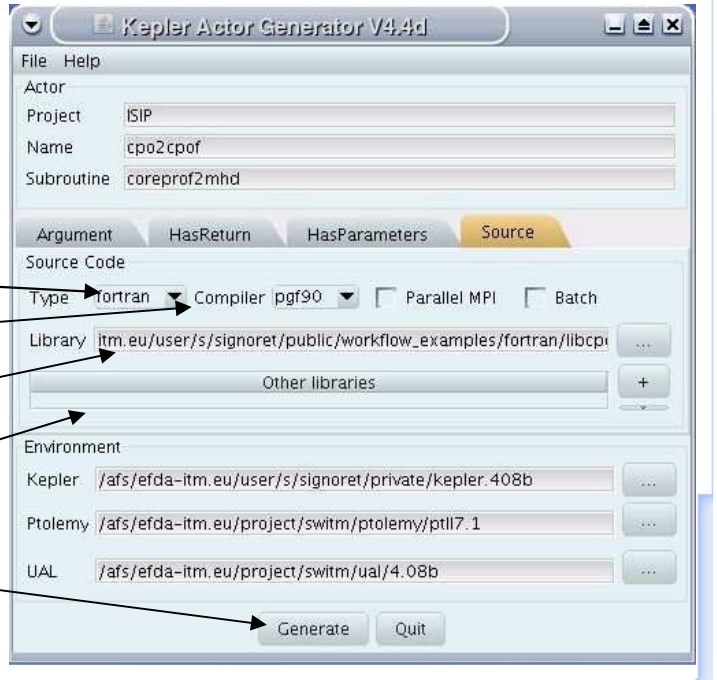
Building the standalone program

- run the standalone program to test the physics code

Run fc2k

- Launch : fc2k &





Automatic generation of the **actor**
(code, component, …)

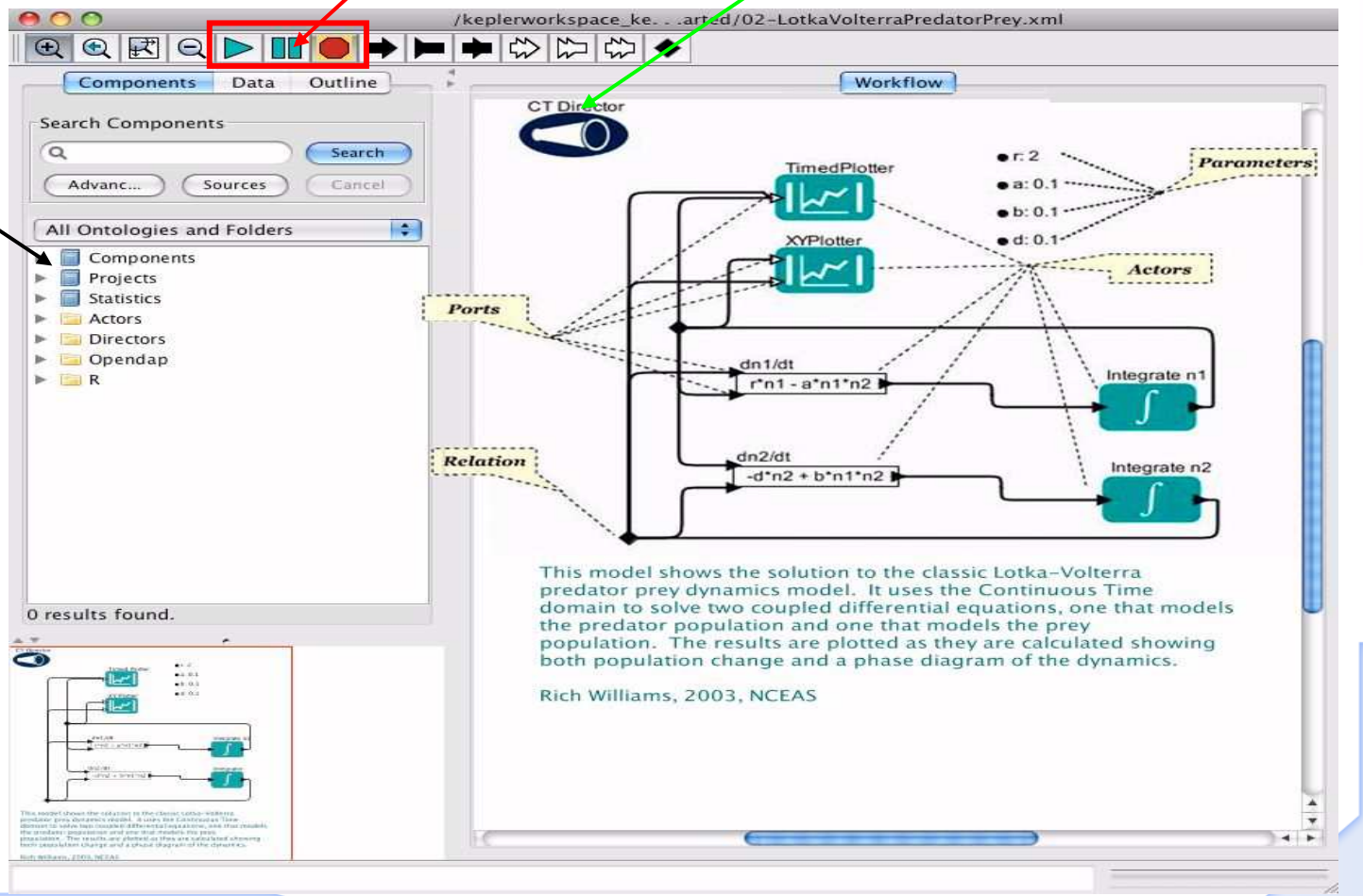
Example : in fc2k load the file
~signoret/public/workflow_examples/fortran/cpo2cpof_fc2k.xml

Workflow: introduction to KEPLER

Run control buttons

Scheduler: time behavior

Catalogues



Many actors:
database, math, display, web service, grid service, R-expression, …

Many directors: DE, CT, SDF, PN, DDF and more in Ptolemy II (FSM, Giotto, DDE)

Design of a workflow (1)

Choose the actors

1. In the Catalogues space, expand 'ISIP', drag and drop to the workspace the actors ualinit, cpo2cpof, ualcollector
   ualinit reads the input shot in the database
   ualcollector writes the resulting cpo in the database
2. Search 'constant' actor : drag and drop 4 constant and 3 String constant actors to the workspace
3. Search 'display' actor; drag and drop it
4. Define the director : choose a SDF director

Customize the constant actor

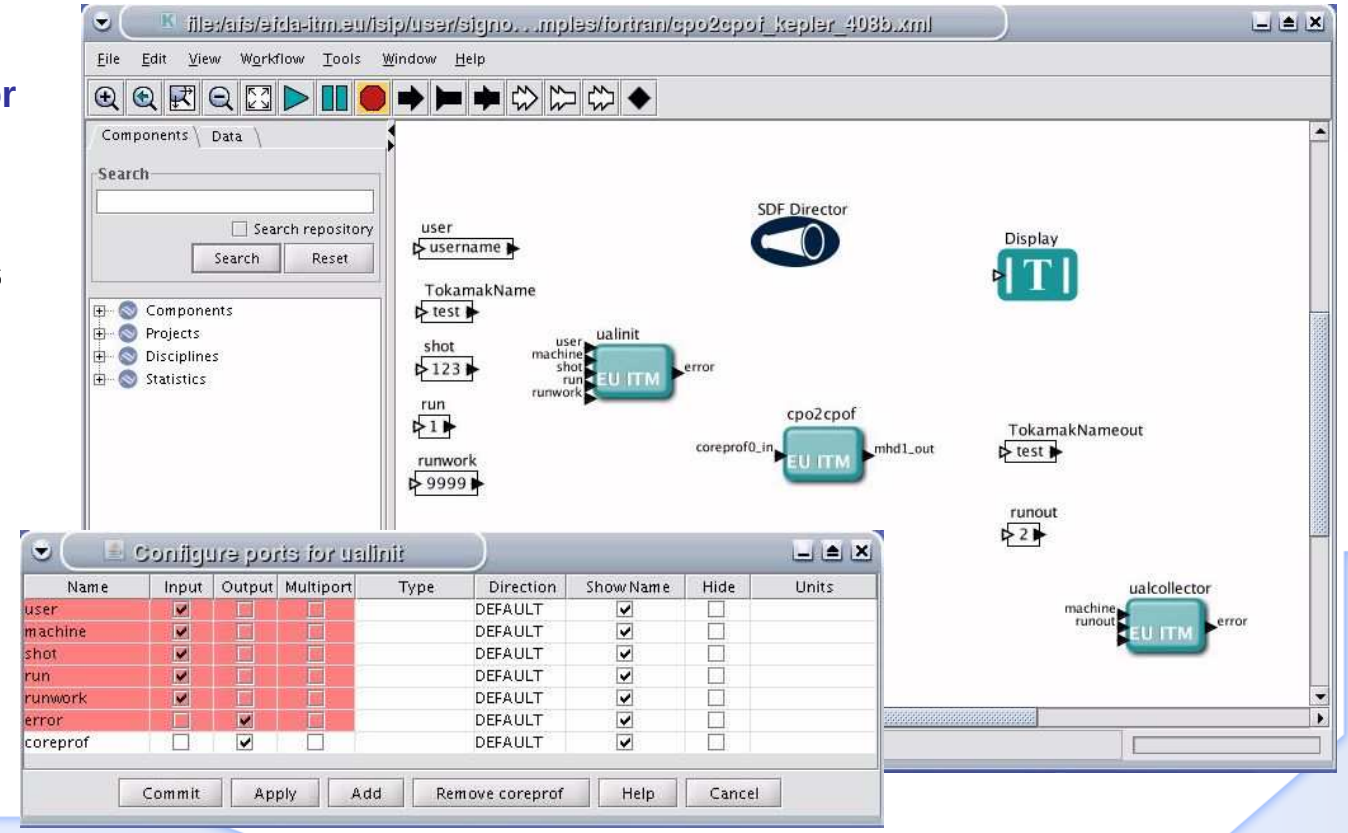
• Change the actors names right click on an actor, choose 'Customize Name')
• Set the values of the actors (right click on an actor, choose 'Configure Actor')

Customize ualinit

1.Add a port for each cpo

Connect the actors

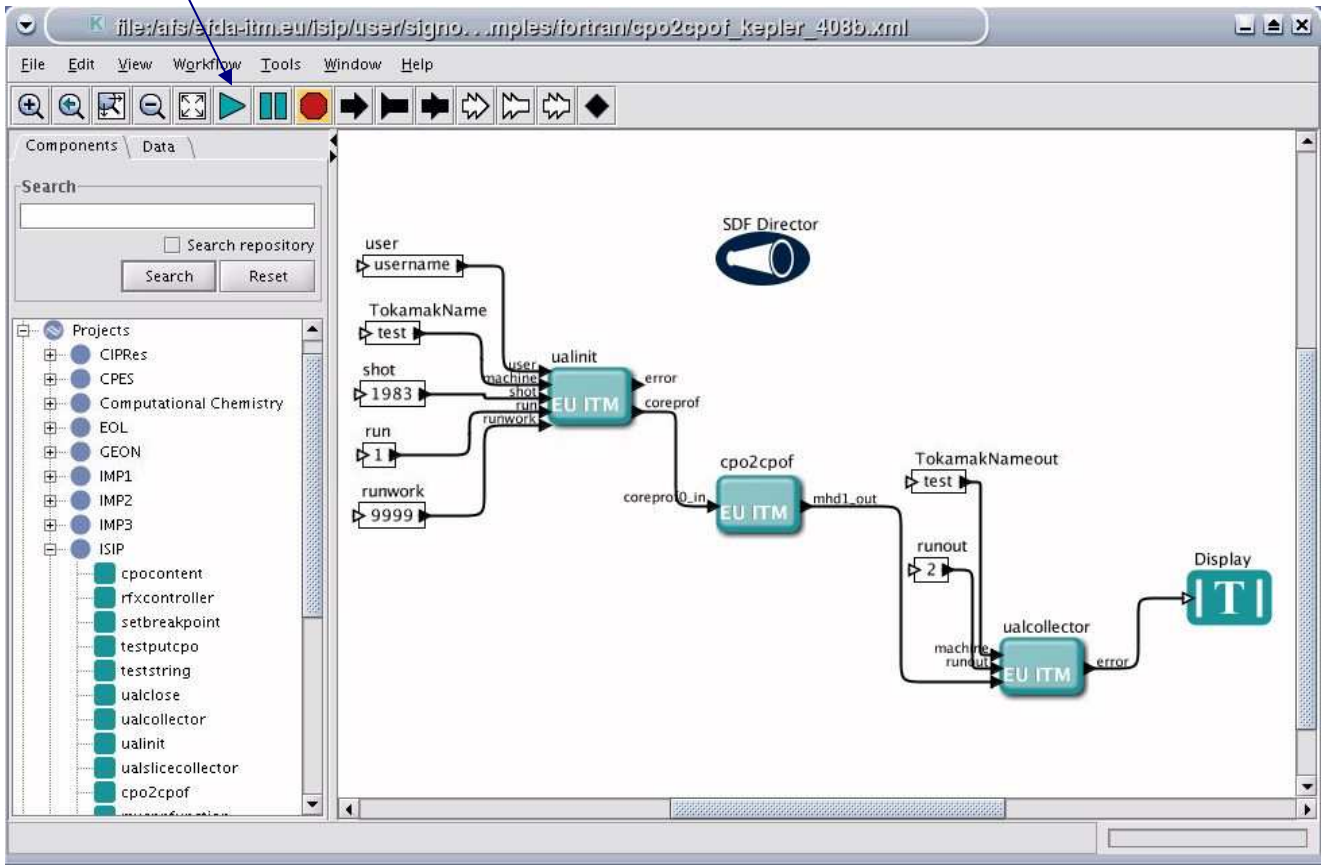
Don't forget to save the workflow!



Execute a workflow

Run the simulation

1. **Optional : Select Tools → Animate at runtime : set a delay in milliseconds to follow the execution of the workflow**
2. **Push the button**



Other examples

NB : the following examples are available for UAL 4.08b

- **Processing a single slice (FORTRAN)**
  - –Parameters for fc2k :
    ~signoret/public/workflow_examples/fortran/cposlice2cposlicef_fc2k.xml
  - –Worklow to open in Kepler :
    ~signoret/public/workflow_examples/fortran/cposlice2cposlicef_kepler_408b.xml

- **Processing a cpo (C++) :**
  - –Parameters for fc2k :
    ~signoret/public/workflow_examples/cpp/cpo2cpocpp_fc2k.xml
  - –Worklow to open in Kepler :
    ~signoret/public/workflow_examples/cpp/cpo2cpocpp_kepler_408b.xml

Useful scripts

- rmactor :
  - Delete an actor from your Kepler installation
  - Usage : rmactor *actorname*
- extract_actor
  - Export all the files from your Kepler installation for a given actor and copy them in a tar file
  - Usage : extract_actor *actorname*
- import_actor
  - Import all the files from an actor tar file into your Kepler installation
  - Usage : import_actor [options] [path/]*actorname*
    - Import_actor –h : display the detailed usage
- NB : users can exchange their own actors using extract_actor and import_actor

References

1. KEPLER:
   - http://www.kepler-project.org/
   - http://users.sdsc.edu/~altintas/KeplerTutorial/
2. ITM: http://www.efda-taskforce-itm.org/
3. Gateway: http://www.efda-itm.eu
4. Euforia: http://www.euforia-project.eu/EUFORIA/
5. Ptolemy II: http://ptolemy.eecs.berkeley.edu/publications/

6. See also :
   - UAL Tutorial
   - ITM Tools Tutorial