Technical University of Denmark

ITM integration, parallelization of ESEL, HDF 5

Nielsen, Anders Henry

Publication date:
2009

The Integrated Tokamak Modeling Task Force (ITM-TF)

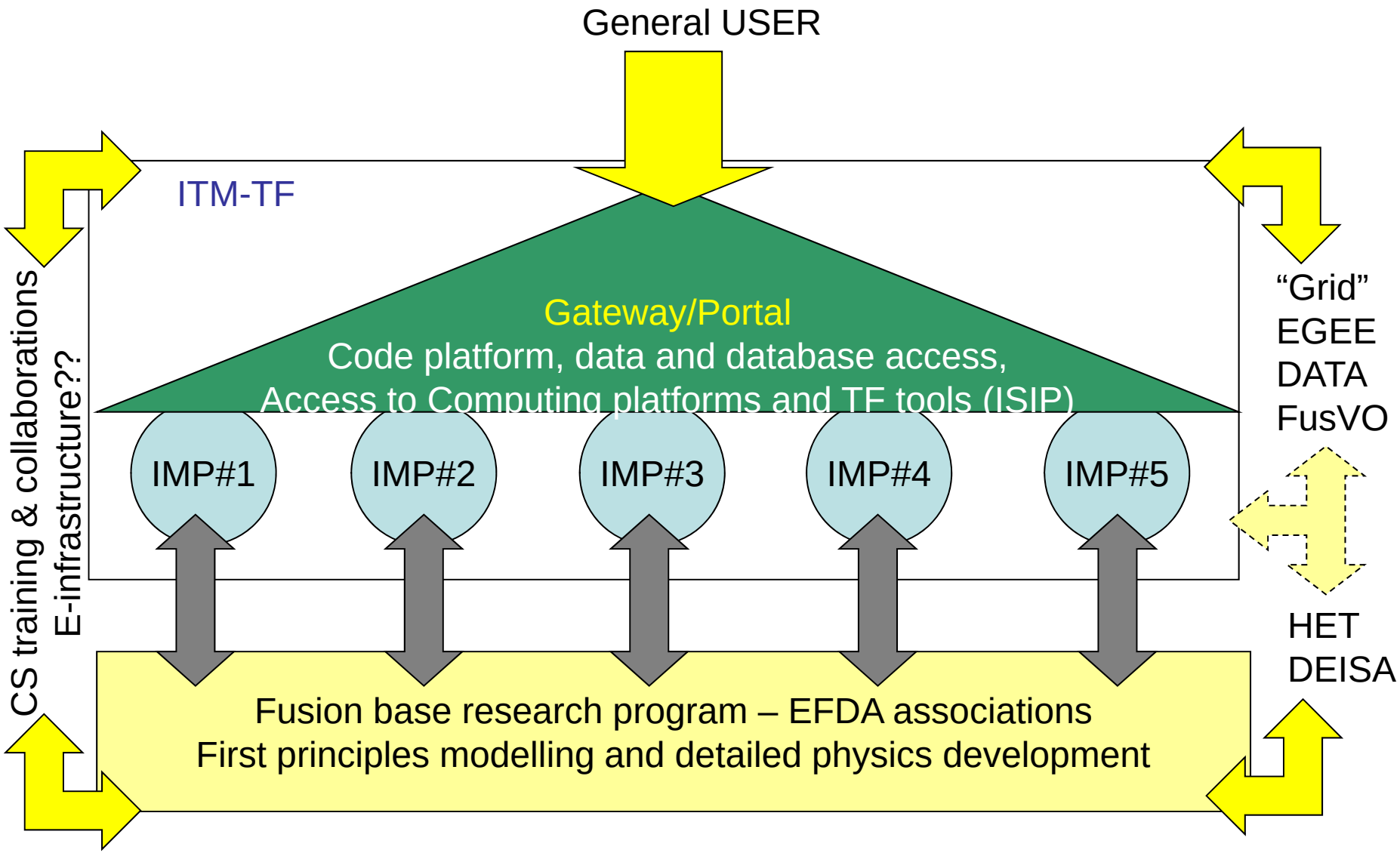
The Integrated Tokamak Modeling Task Force (ITM-TF) has the long-term aim to provide the EU with suite of codes necessary for preparing and analyzing future ITER discharges, with the highest degree of flexibility, confidence and reliability.

Taskforce Leader: Paar Strand

Five Integrated Modeling Projects (IMP)

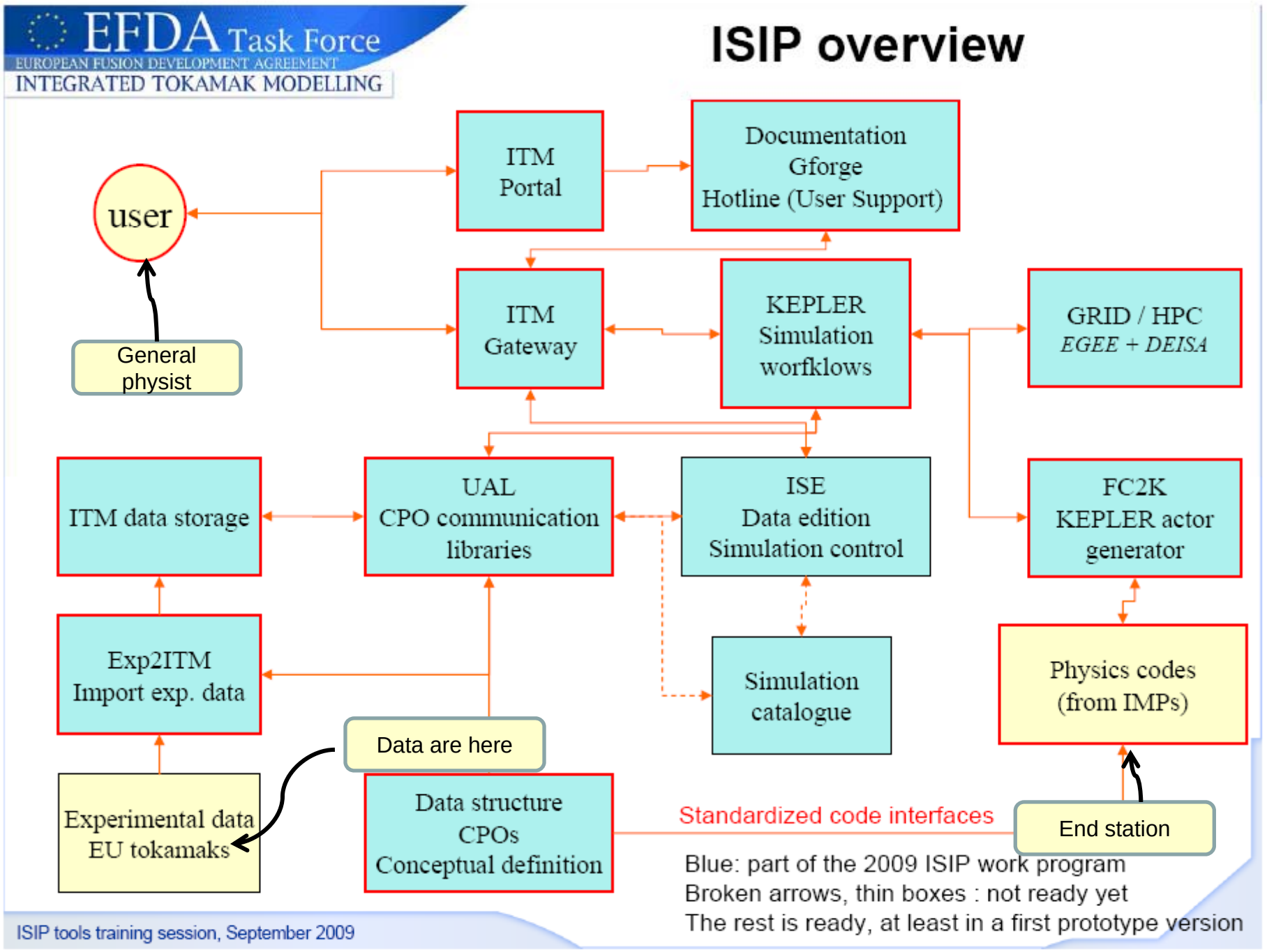
- IMP#1 Equilibrium and MHD
- IMP#2 Non linear MHD, sawtooth and ELMs
- IMP#3 Energy and particle transport
- IMP#4 First principle transport and turbulence (Bruce Scott)
- IMP#5 Fast particles and heating
- ISIP Infrastructure Support Project (UAL, Kepler, ESE,..) (F. Imbeaux)

Schematics


General USER
ITM-TF
Gateway/Portal
Code platform, data and database access,
Access to Computing platforms and TF tools (ISIP)
IMP#1
IMP#2
IMP#3
IMP#4
IMP#5
CS training & collaborations
E-infrastructure??
"Grid"
EGEE
DATA
FusVO
HET
DEISA
Fusion base research program – EFDA associations
First principles modelling and detailed physics development

EFDA Task Force
EUROPEAN FUSION DEVELOPMENT AGREEMENT
INTEGRATED TOKAMAK MODELLING

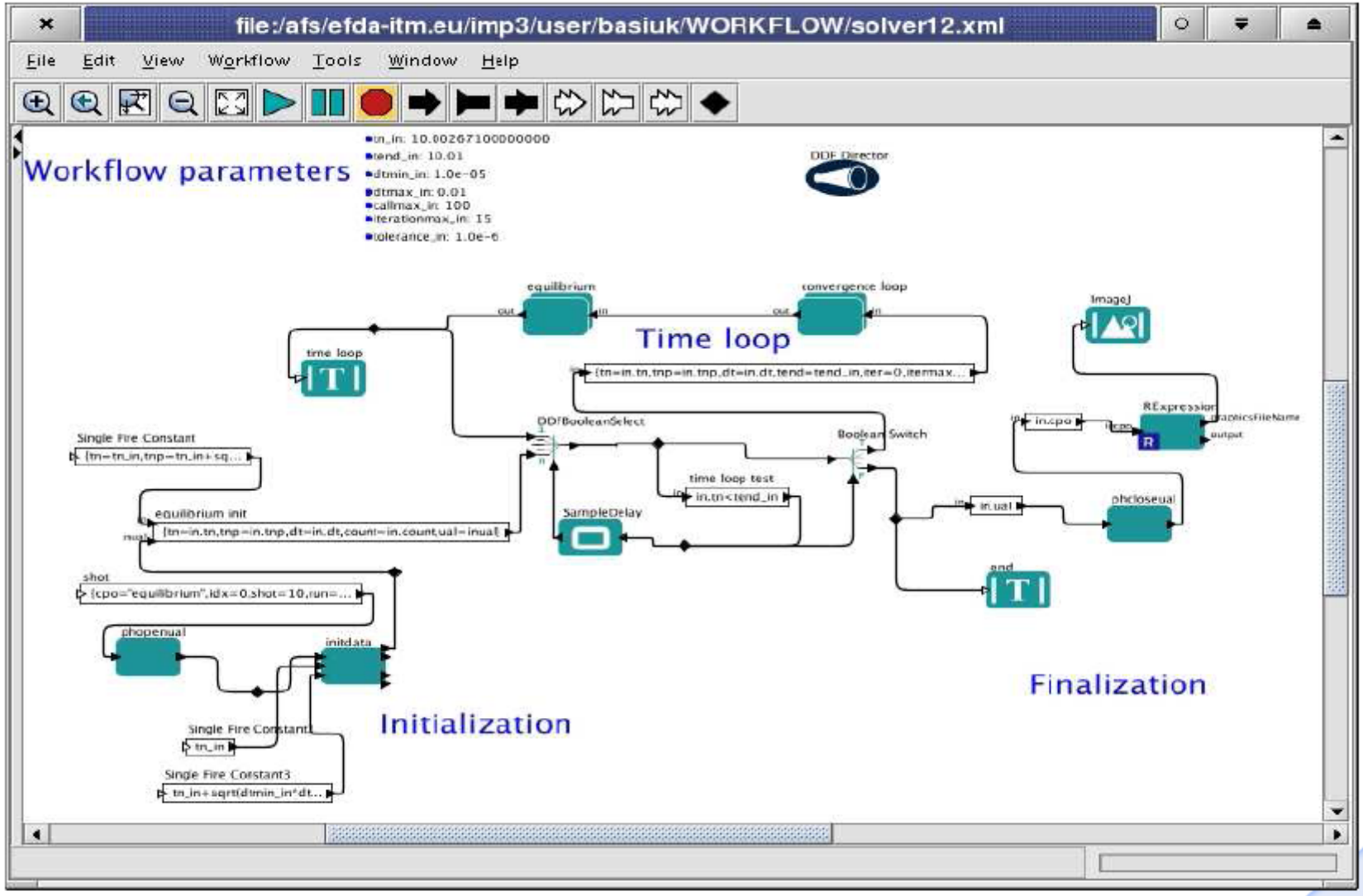
ISIP overview



Overview

- Data Structure : Physics data organised in Consistent Physical Objects (CPO) : standard of communication between physics modules. Thought in view of modular and flexible workflow design
  - coded as XML schemas, a series of XSL transforms turn them into HTML documentation, language specific libraries, machine description file, …
- UAL : Communication library between physics modules : allows to exchange CPOs
  - Available in Fortran, C++, Java, Matlab, Python
  - Generated dynamically from the data structure
  - Uses MDS+ as backend (HDF5 option exists but through MDS+ software)
  - Read/Write from/to disk (default) or memory
- KEPLER : design and runs workflows, with wrapped physics modules as elementary "acting units"
  - Integrated platform developed in San Diego, used by some of the US FSP
  - Drag and drop physical modules « actors » to create workflows

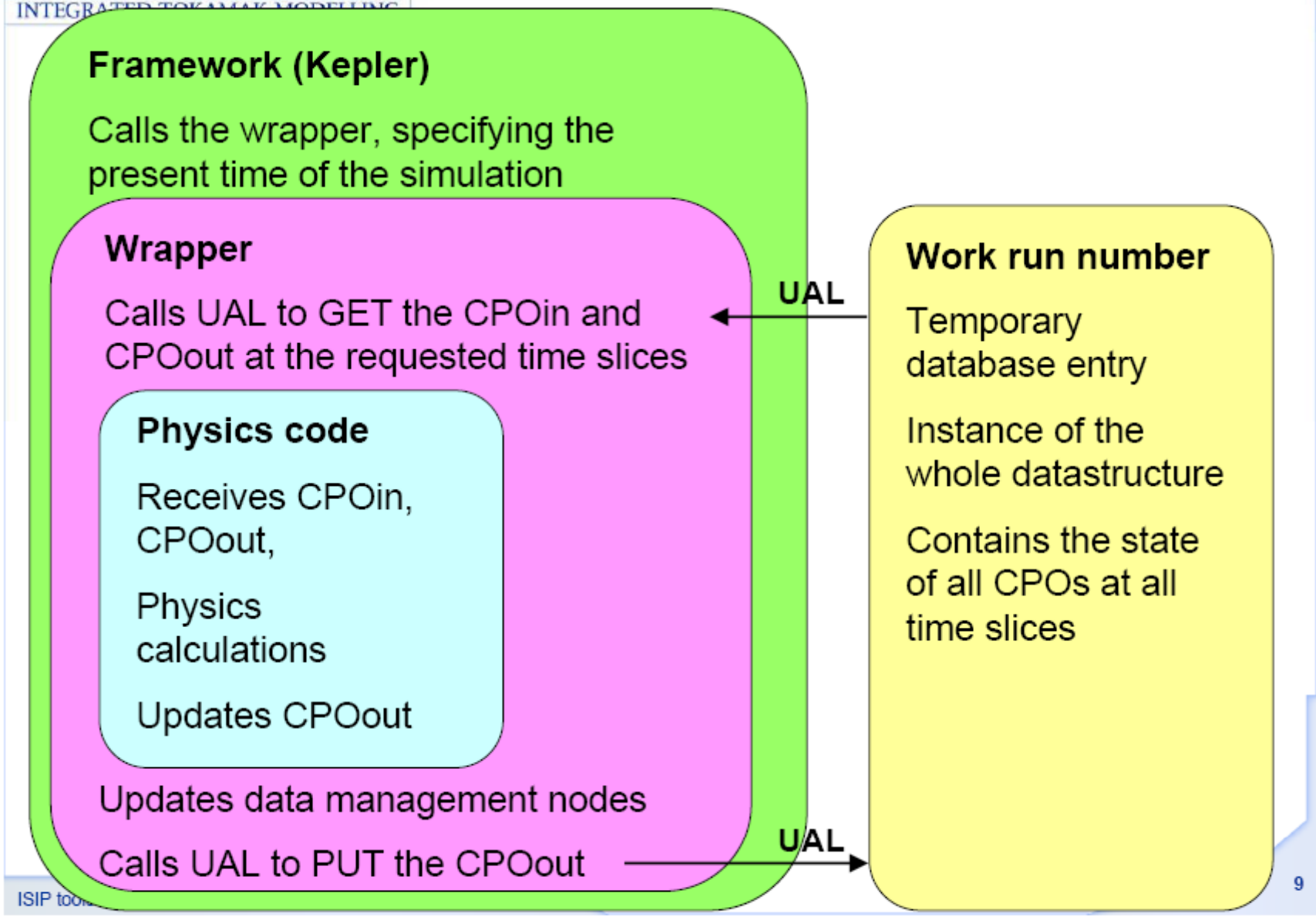
KEPLER


file:/afs/efda-itm.eu/imp3/user/basiuk/WORKFLOW/solver12.xml
File Edit View Workflow Tools Window Help
Workflow parameters
tn_in: 10.00267100000000
tend_in: 10.01
dtmin_in: 1.0e-05
dtmax_in: 0.01
callmax_in: 100
iterationmax_in: 15
tolerance_in: 1.0e-6
DDF Director
equilibrium
convergence loop
Time loop
ImageJ
time loop
{tn=in.tn,tnp=in.tnp,dt=in.dt,tend=tend_in,ier=0,itemax...
DDFBooleanSelect
Boolean Switch
RExpression
in.cpo
graphicsFileName
output
Single Fire Constant
{tn=tn_in,tnp=tn_in+sq...
time loop test
in.tn<tend_in
equilibrium init
{tn=in.tn,tnp=in.tnp,dt=in.dt,count=in.count,ual=inual}
SampleDelay
in.ual
phcloseual
shot
{cpo="equilibrium",idx=0,shot=10,run=...
and
phopenual
initdata
Finalization
Initialization
Single Fire Constant2
tn_in
Single Fire Constant3
tn_in+sqrt(dtmin_in*dt...

CPO transfer in KEPLER

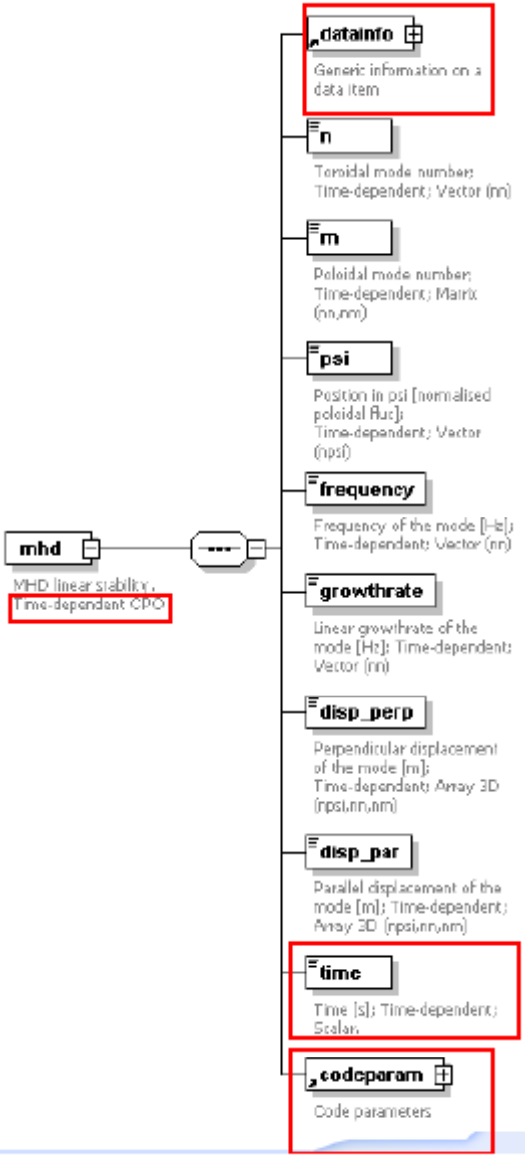
- Physics modules are subroutines expecting CPOs as input and output
  - No need to use the UAL in a physics module, except for testing purposes
  - Physics modules must be wrapped in a layer that deals with the UAL calls
  - The wrapping is done automatically by the FC2K interface
  - Wrapped physics modules are called « actors » and are usable by KEPLER
- CPO transfer in KEPLER is done by linking actors with an « arrow »
  - This arrow may correspond to a CPO or a workflow parameter (e.g. time)
  - When the arrow represents a time-dependent CPO, all time slices are potentially transfered (default mode)
    - Exception : for CPOs declared in « single time slice » mode at the actor generation (FC2K), the wrapper will GET the latest time slice of the CPO before the current time and PUT the output at the current time. The current time is given to the actor as an explicit KEPLER variable.
  - CPOs are not native KEPLER variables. The CPO transfer is done in fact by the « wrapper », not by KEPLER

Outside the Physics module

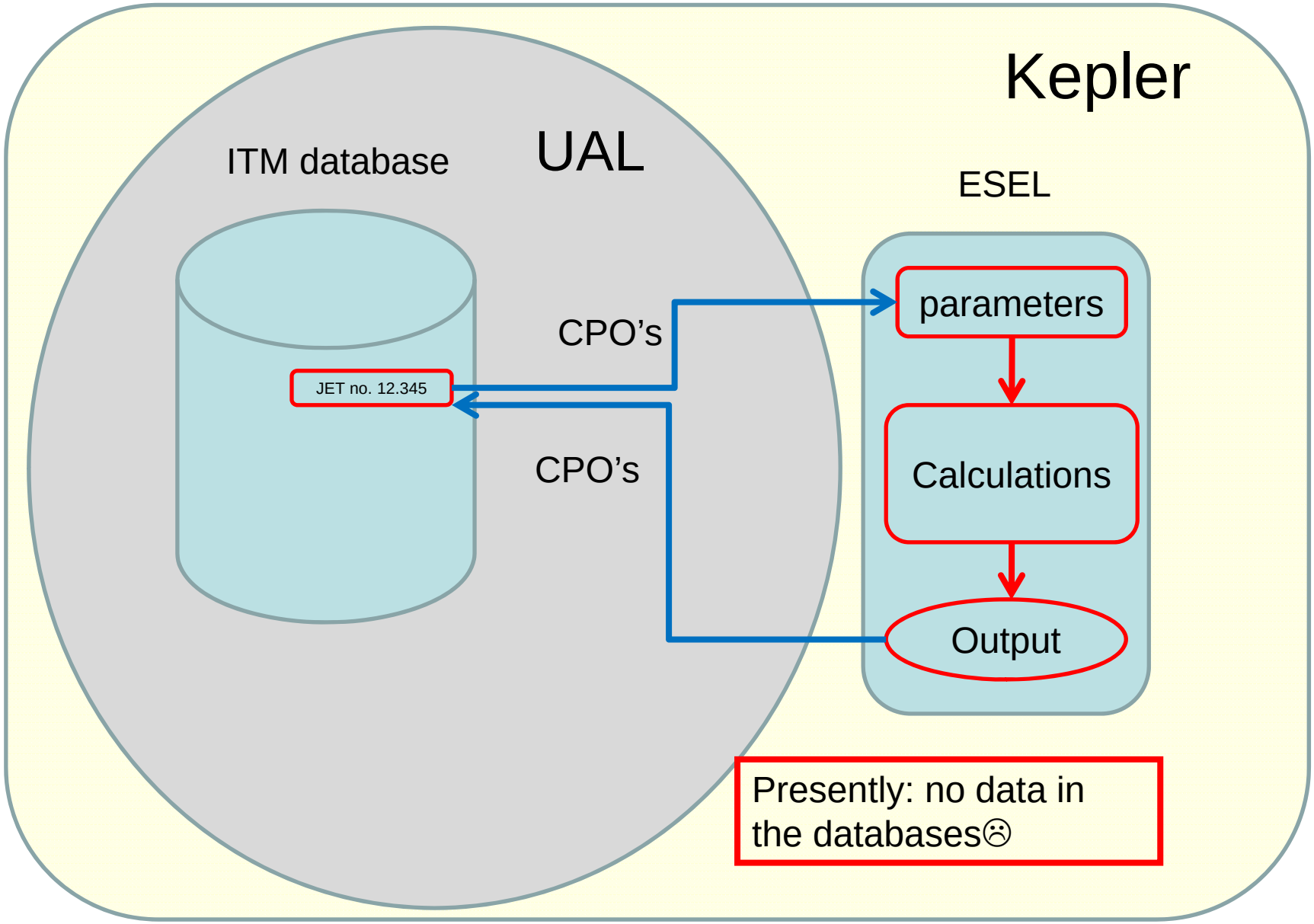


CPO structure

- CPOs have a structure with many signals below
  - Substructures are frequent
  - Fine structure depends on the physics
  - All physics signals related to a CPO are there
- Each CPO has its own time array (if time-dependent)
- Each CPO has a bookkeeping sub-structure (datainfo)
- Code-specific parameters are in codeparam



ITM: my interpretation



Example of standalone wrapper program

```
program test

use euITM_schemas
use euITM_routines

implicit none

integer,parameter :: DP=kind(1.0D0)

interface
subroutine physics(coreprofin,mhdout)
use euitm_schemas
!use euitm_routines
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
integer :: numDims,dim1,dim2,dim3

character(len=5)::treename
shot = 4
runin = 1
runout = 2
refshot = 0 ! Dummy, not used
refrun =0   ! Dummy, not used
treename = 'euitm' ! Mandatory, do not change

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 physics subroutine :'
call ESEL(coreprofin,mhdout)

write(*,*) 'Creating output run :'
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
```

Note: The other IMP's will write data to the databases through the CPO's but IMP4 will (mostly) store HDF in HDF5 data files

HDF5
(**Hierarchical Data Format)**

The official data format for IMP4

One simulation, one file:
A single file containing all fields at all time slices, including probes (compressed) (~0.5 GB)

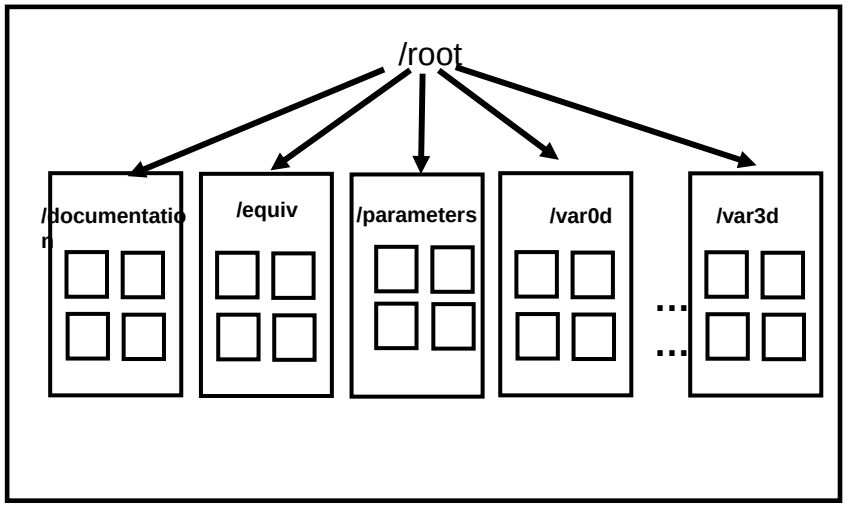
FUTILS: FORTRAN wrapper developed and maintained by CRPP Lausanne:
http://pleiades1.epfl.ch/~tmt/pub

Does NOT support parallel read/write☹

Matlab and graphics programs
interface for reading/writting HDF5 files

HDF5 is a data model, library, and file format for storing and managing data. It supports an unlimited variety of data types, and is designed for flexible and efficient I/O and for high volume and complex data. HDF5 is portable and is extensible, allowing applications to evolve in their use of HDF5.

From: http://www.hdfgroup.org/HDF5/



Parallelization of ESEL

- ESEL is presently being parallelized at *Åbo Akademi University, Department of Information Technologies, Finland*.
- Domain decomposition
- Information about boundary values need to be communicated between neighboring domains/CPUs every time step
- A parallel Poisson/Helmholtz solver are needed
- Presently 16 – 32 CPU are the upper limits due to parallel write bottlenecks. Hopefully 100 CPUs should be achieved.