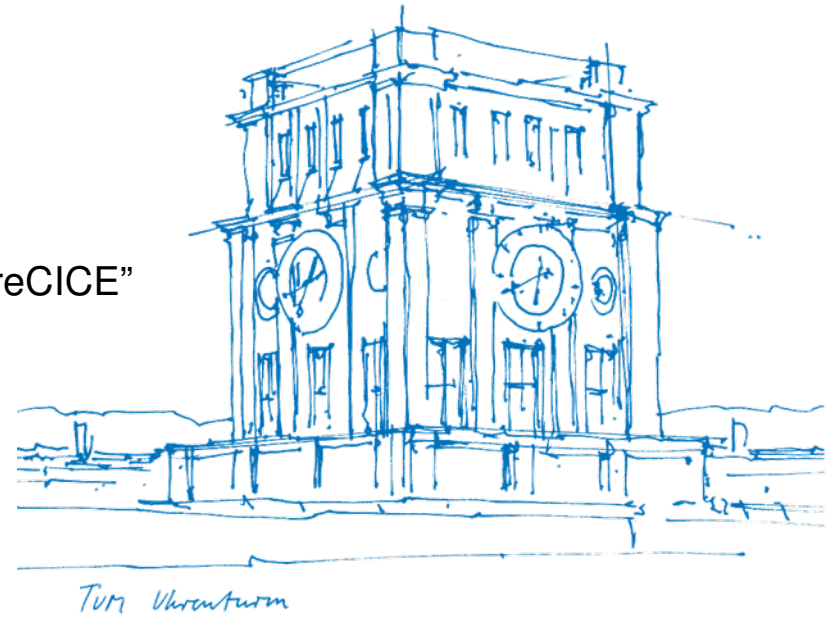
Can my OpenFOAM solver easily be coupled with preCICE?

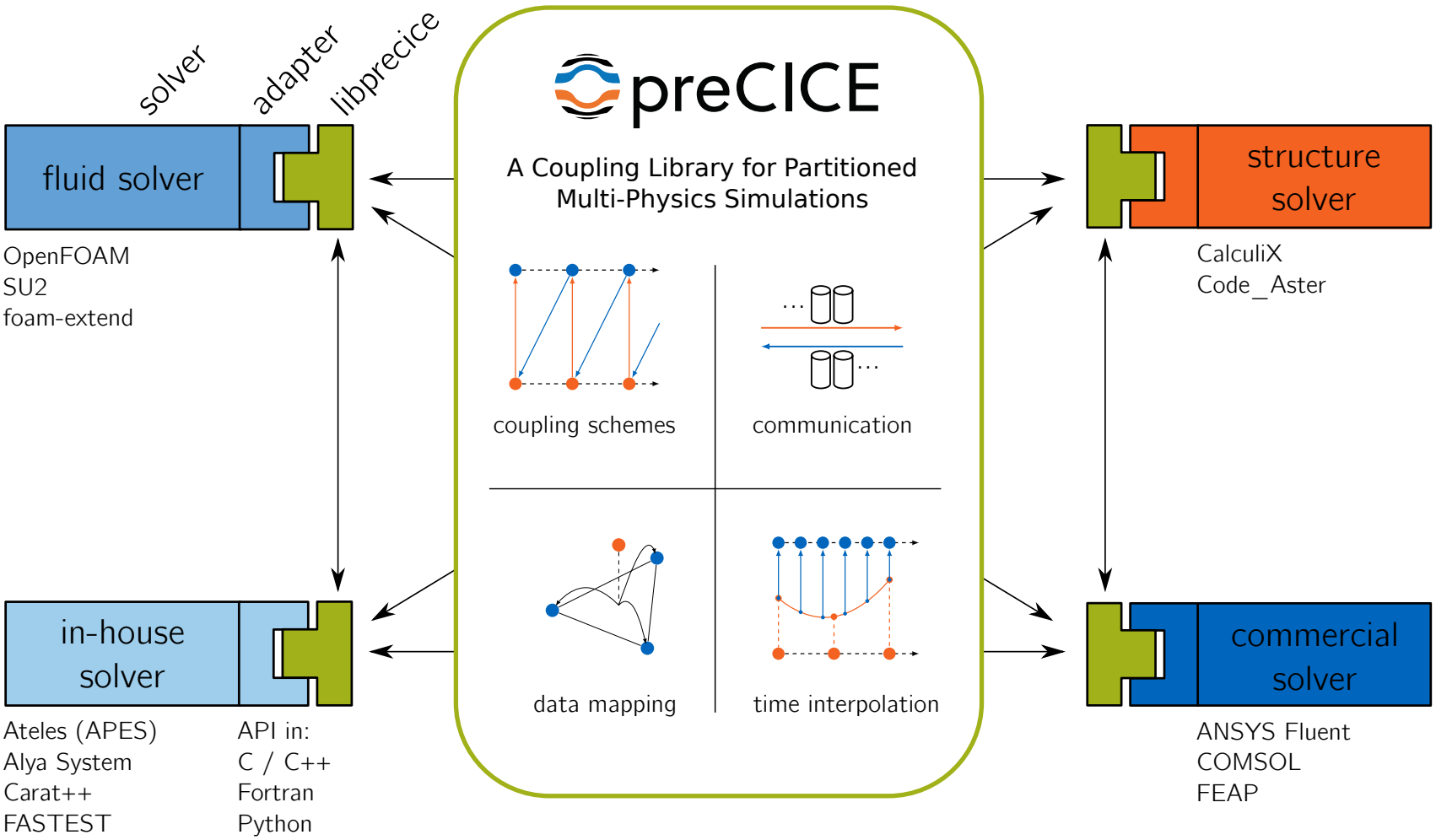
Gerasimos Chourdakis, Benjamin Uekermann, Hans-Joachim Bungartz

Technical University of Munich
Department of Informatics
Chair of Scientific Computing in Computer Science

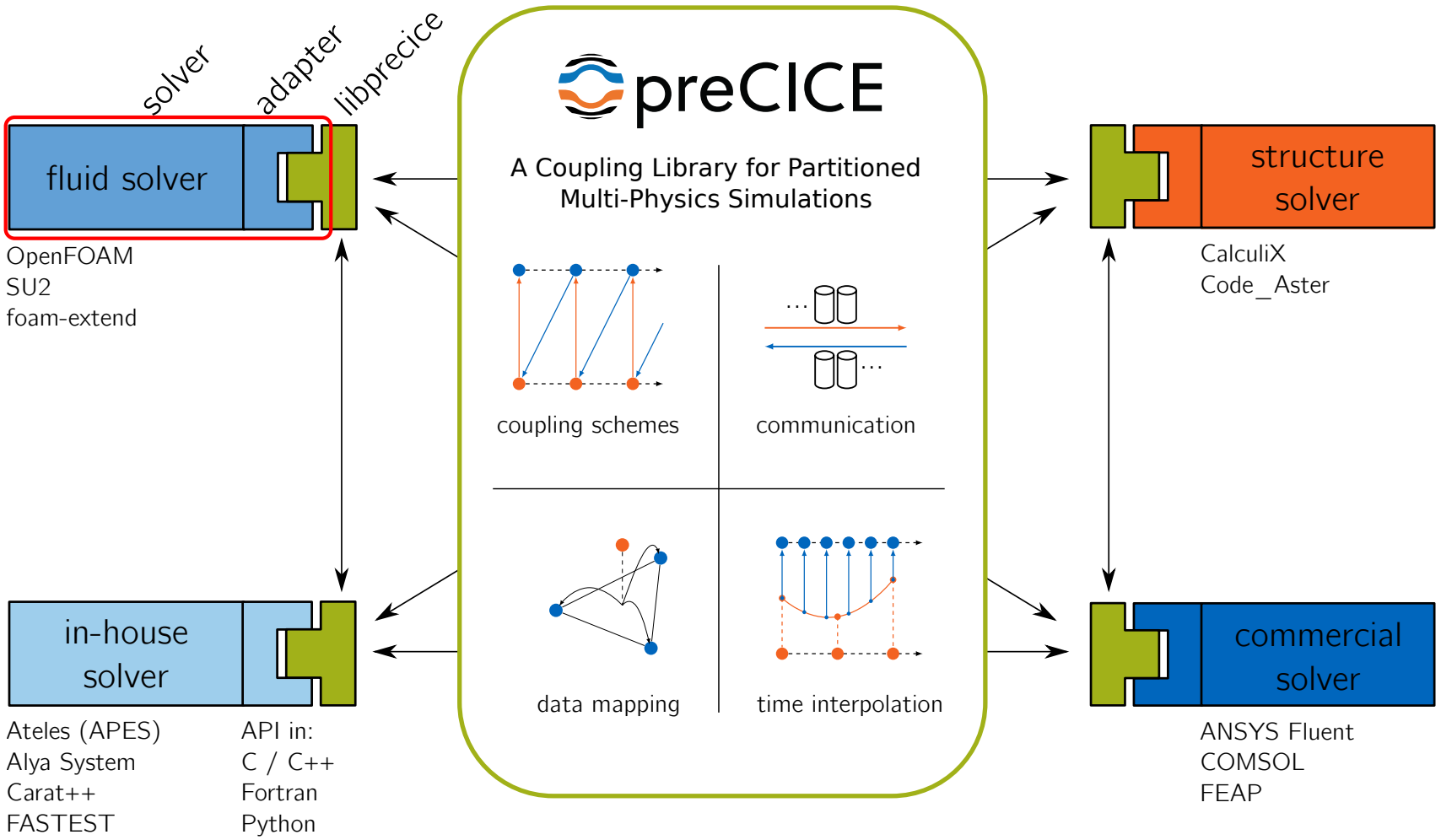
ECCOMAS ECCM/ECFD 2018, Glasgow
Minisymposium MS127A
“Multi-Physics Simulations with the Coupling Library preCICE”
June 14, 2018



Overview


solver
adapter
libprecice
fluid solver
OpenFOAM
SU2
foam-extend
in-house solver
Ateles (APES)
Alya System
Carat++
FASTEST
API in:
C / C++
Fortran
Python
preCICE
A Coupling Library for Partitioned Multi-Physics Simulations
coupling schemes
communication
data mapping
time interpolation
structure solver
CalculiX
Code_Aster
commercial solver
ANSYS Fluent
COMSOL
FEAP

Overview


solver
adapter
libprecice
fluid solver
OpenFOAM
SU2
foam-extend
preCICE
A Coupling Library for Partitioned Multi-Physics Simulations
coupling schemes
communication
data mapping
time interpolation
structure solver
CalculiX
Code_Aster
in-house solver
Ateles (APES)
Alya System
Carat++
FASTEST
API in:
C / C++
Fortran
Python
commercial solver
ANSYS Fluent
COMSOL
FEAP

How to couple my own solver?

```
1  precice::SolverInterface precice("FluidSolver",rank,size);
2  precice.configure("precice-config.xml");
3  precice.precice.setMeshVertices();
4  precice.precice.initialize();
5
6  while (precice.isCouplingOngoing()) { // main time loop
7      solve();
8
9      precice.writeBlockVectorData();
10     precice.advance();
11     precice.readBlockVectorData();
12
13     endTimeStep(); // e.g. write results, increase time
14 }
15
16 precice.finalize();
```

Timesteps, most arguments and less important methods omitted. Full example in the wiki.

Adapting an OpenFOAM solver

```
/* Start the solver */
```

```
Info<< "\nStarting time loop\n" << endl;
while (runTime.run()) {
  #include "readTimeControls.H"
  #include "compressibleCourantNo.H"
  #include "setDeltaT.H"
```

```
  runTime++;
```

```
  /* continue --> */
```

```
  /* solve the equations */
  #include "rhoEqn.H"
  while (pimple.loop())
  {
      ...
  }
```

```
  runTime.write();
}
```

```
/* Finalize */
```

Adapting an OpenFOAM solver

```
/* Start the solver */
/* Adapter: Initialize coupling
      calls precice->initialize() */
adapter.initialize();

Info<< "\nStarting time loop\n" << endl;
while (runTime.run()) {
  #include "readTimeControls.H"
  #include "compressibleCourantNo.H"
  #include "setDeltaT.H"




  runTime++;


  /* continue --> */
```

```
  /* solve the equations */
  #include "rhoEqn.H"
  while (pimple.loop())
  {
      ...
  }




  runTime.write();
}


/* Finalize */
```

Adapting an OpenFOAM solver

```
/* Start the solver */
/* Adapter: Initialize coupling
        calls precice->initialize() */
adapter.initialize();

Info<< "\nStarting time loop\n" << endl;
while (adapter.isCouplingOngoing()) {
    #include "readTimeControls.H"
    #include "compressibleCourantNo.H"
    #include "setDeltaT.H"




    runTime++;


    /* continue --> */
    /* solve the equations */
    #include "rhoEqn.H"
    while (pimple.loop())
    {
        ...
    }




    runTime.write();
}


/* Finalize */
```

Adapting an OpenFOAM solver

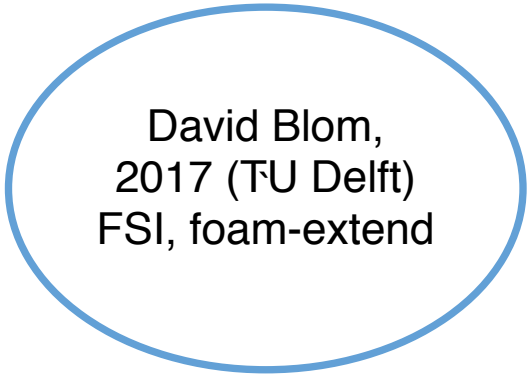
```
/* Start the solver */
/* Adapter: Initialize coupling
      calls precice->initialize() */
adapter.initialize();

Info<< "\nStarting time loop\n" << endl;
while (adapter.isCouplingOngoing()) {
  #include "readTimeControls.H"
  #include "compressibleCourantNo.H"
  #include "setDeltaT.H"




  runTime++;

  /* Adapter: Receive coupling data */
  adapter.readCouplingData();
  /* continue --> */
  /* solve the equations */
  #include "rhoEqn.H"
  while (pimple.loop())
  {
      ...
  }

  /* Adapter: Write in buffers */
  adapter.writeCouplingData();

  /* Adapter: advance the coupling
        calls precice->advnace()  */
  adapter.advance();

  runTime.write();
}


/* Finalize */
```

An adapted OpenFOAM solver

```
/* Start the solver */
/* Adapter: Initialize coupling
      calls precice->initialize() */
adapter.initialize();

Info<< "\nStarting time loop\n" << endl;
while (adapter.isCouplingOngoing()) {
  #include "readTimeControls.H"
  #include "compressibleCourantNo.H"
  #include "setDeltaT.H"

  /* Adapter: Adjust solver time */
  adapter.adjustSolverTimeStep();

  runTime++;

  /* Adapter: Receive coupling data */
  adapter.readCouplingData();
  /* continue --> */
  /* solve the equations */
  #include "rhoEqn.H"
  while (pimple.loop())
  {
      ...
  }

  /* Adapter: Write in buffers */
  adapter.writeCouplingData();

  /* Adapter: advance the coupling
        calls precice->advnace()  */
  adapter.advance();

  runTime.write();
}


/* Finalize */
```

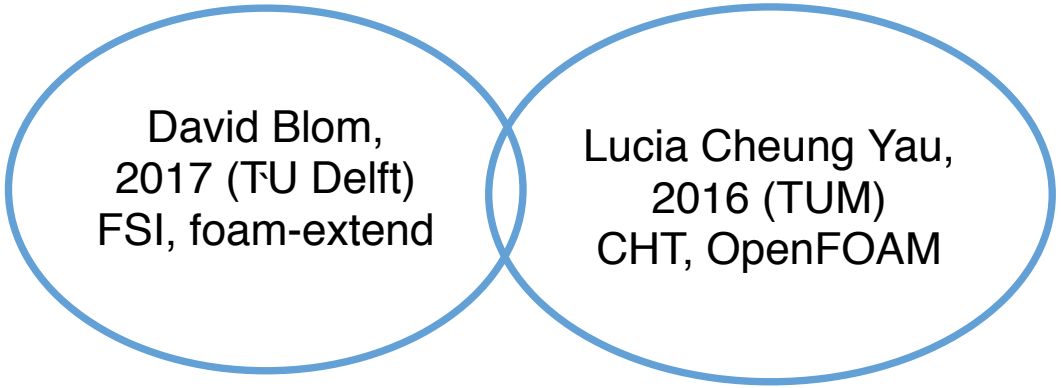
Duplicated development effort

OpenFOAM (and family) adapters for preCICE

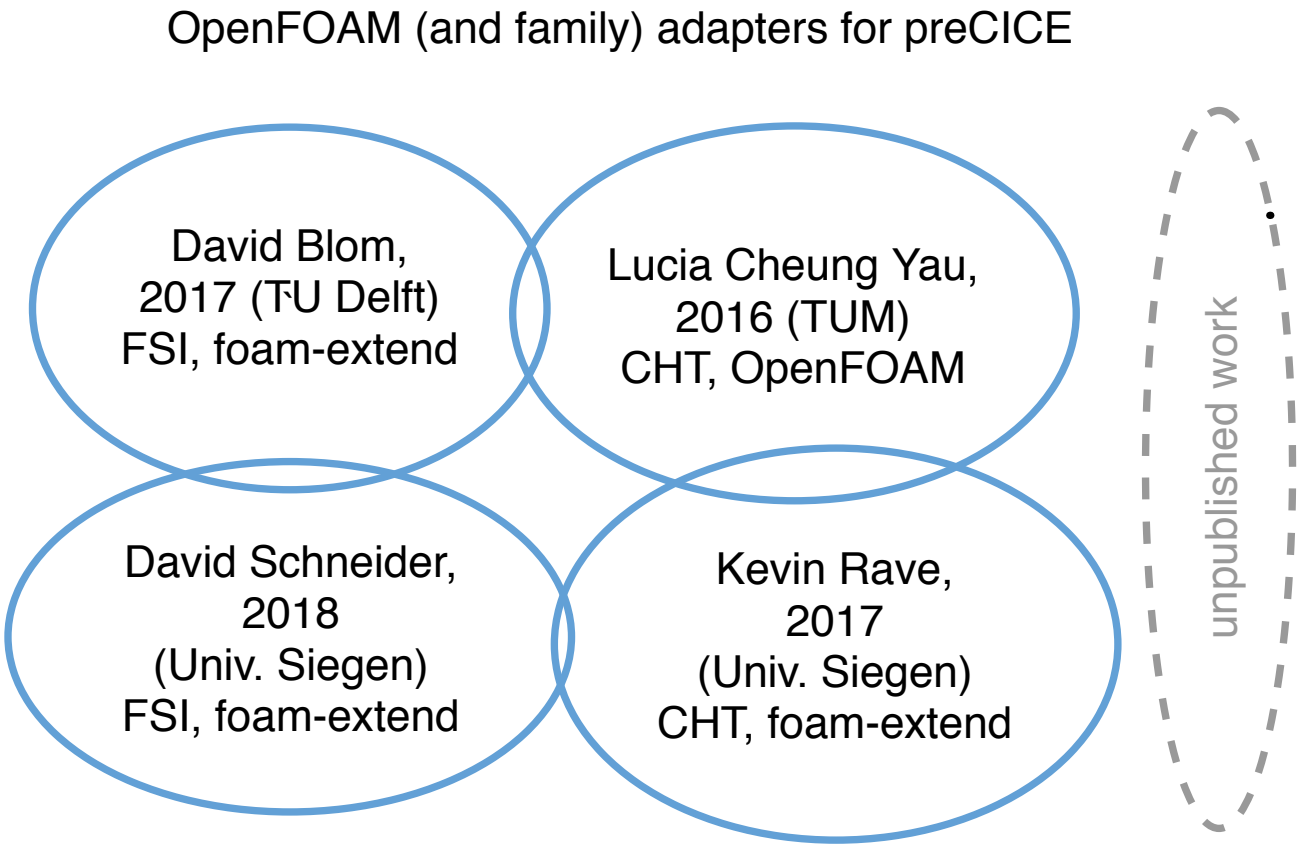


Duplicated development effort

OpenFOAM (and family) adapters for preCICE

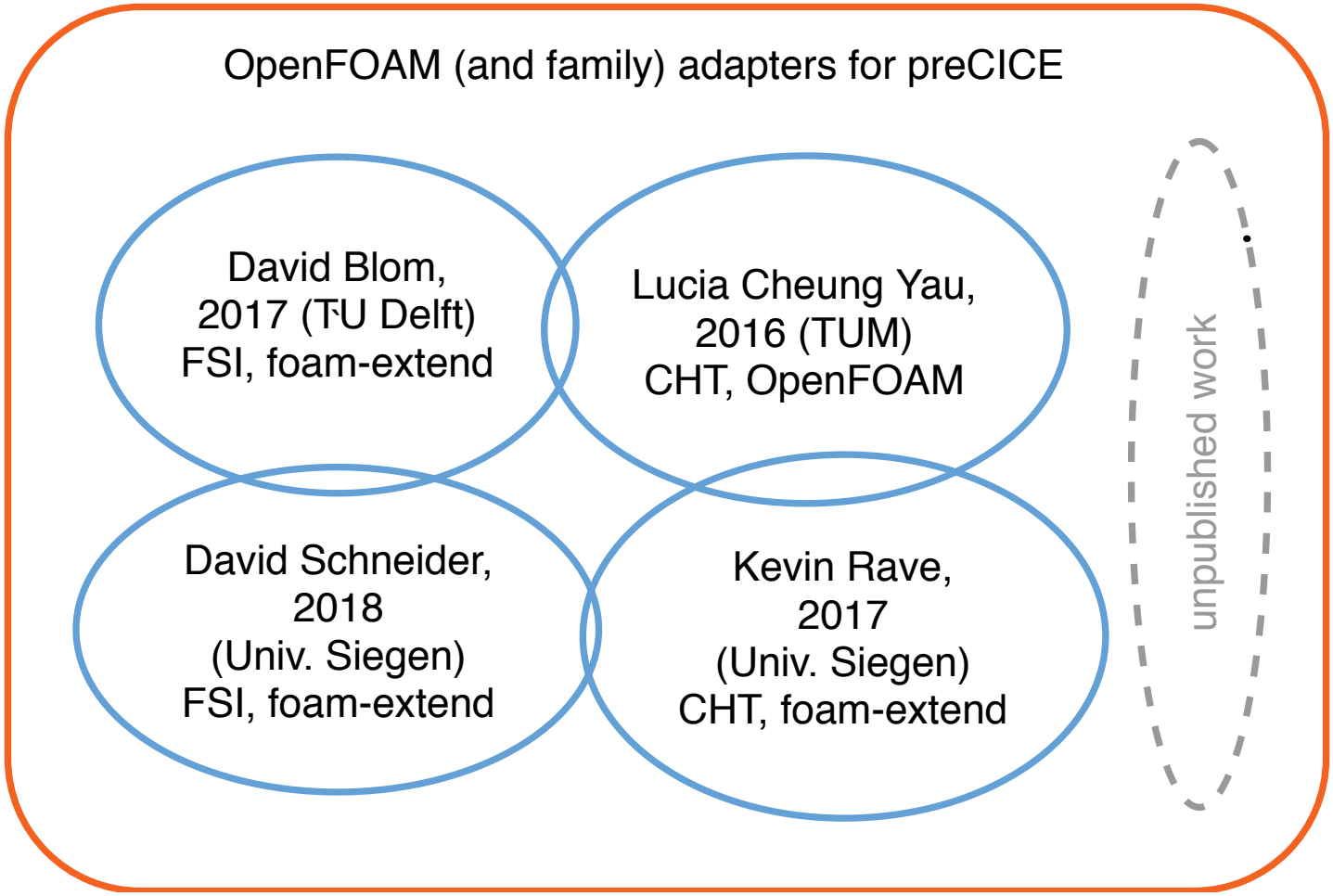


Duplicated development effort



All these adapters are **bound to specific solvers**!

Duplicated development effort



All these adapters are **bound to specific solvers**!
→ We need an official, general adapter!

Before: Working and validated prototypes

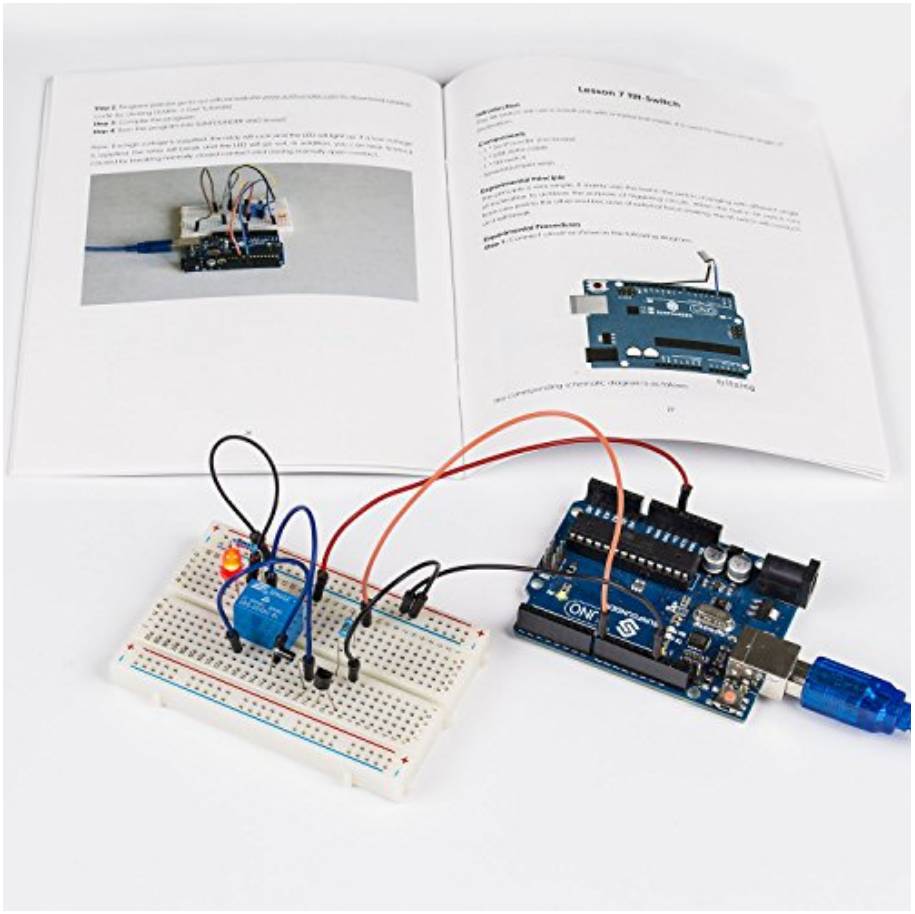


Image from desertcart.ae.

Before: Working and validated prototypes

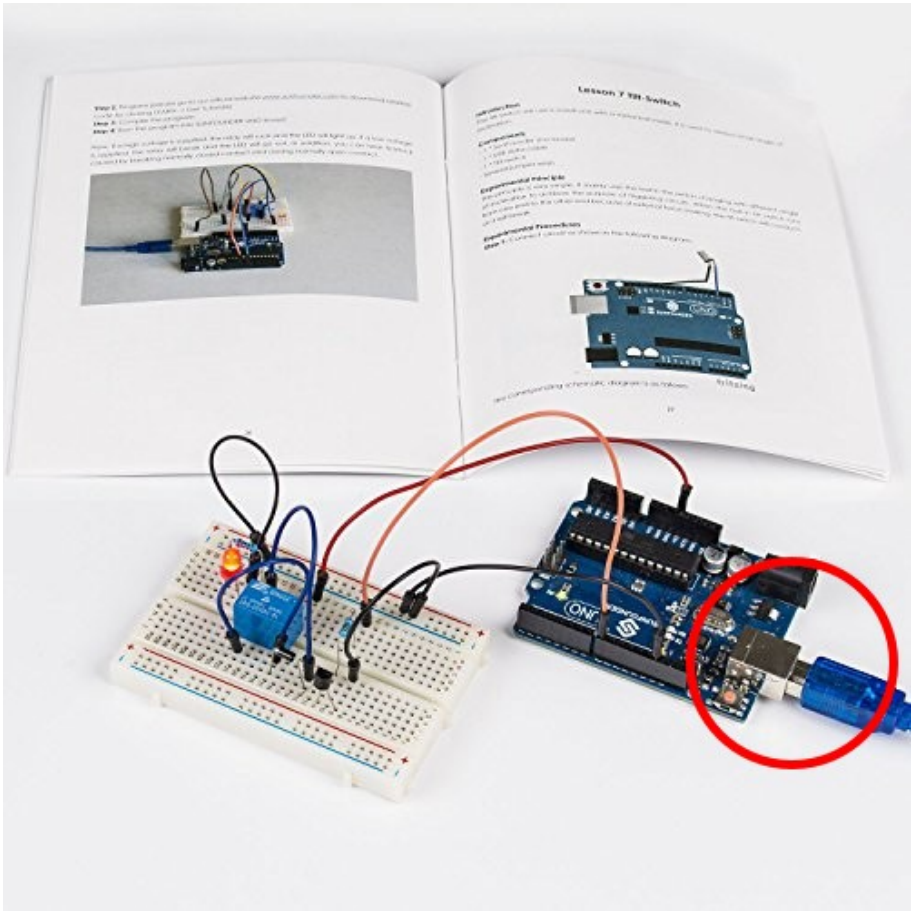
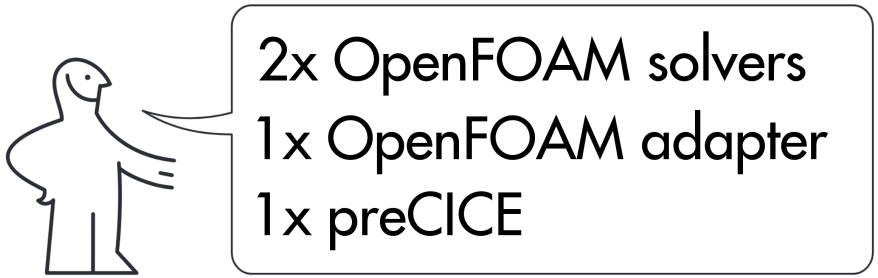


Image from desertcart.ae.

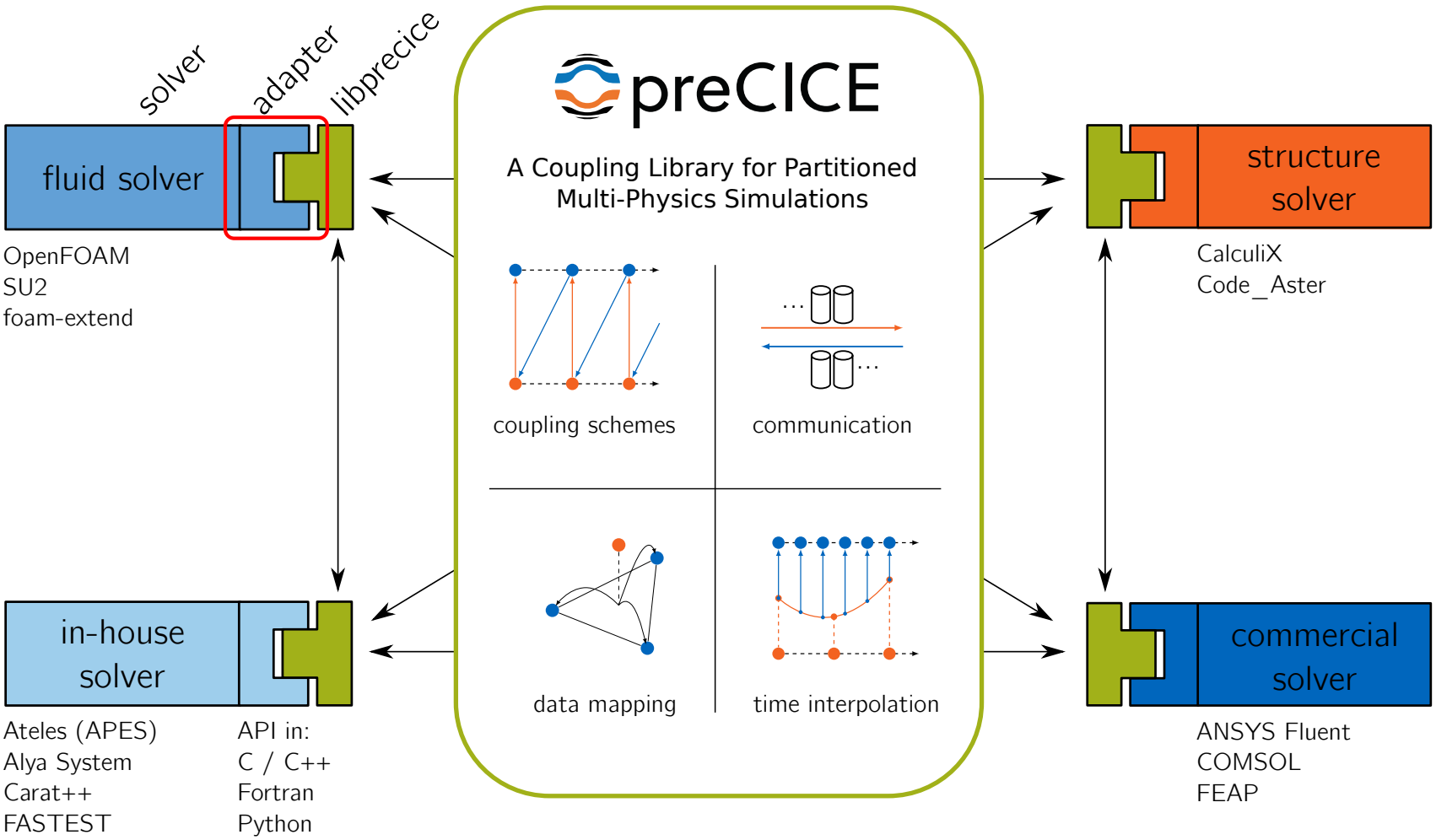
Now: A user-friendly, plug-and-play adapter

KOPPLAD



The human-like figure is a property of ikea.com.

Overview


solver
adapter
libprecice
fluid solver
OpenFOAM
SU2
foam-extend
preCICE
A Coupling Library for Partitioned Multi-Physics Simulations
coupling schemes
communication
data mapping
time interpolation
structure solver
CalculiX
Code_Aster
in-house solver
Ateles (APES)
Alya System
Carat++
FASTEST
API in:
C / C++
Fortran
Python
commercial solver
ANSYS Fluent
COMSOL
FEAP

Isolating the adapter: Function Objects

```
/* Start the solver */

Info<<"\nStarting time loop\n"<< endl;
while (runTime.run()) {
  #include "readTimeControls.H"
  #include "compressibleCourantNo.H"
  #include "setDeltaT.H"

  runTime++;

  /* solve the equations */
  #include "rhoEqn.H"
  while (pimple.loop())
  {
      ...
  }

  runTime.write();
}

/* Finalize */
```

Isolating the adapter: Function Objects

```
/* Start the solver */

Info<<"\nStarting time loop\n"<< endl;
while (runTime.run()) {
  #include "readTimeControls.H"
  #include "compressibleCourantNo.H"
  #include "setDeltaT.H"

  runTime++;

  /* solve the equations */
  #include "rhoEqn.H"
  while (pimple.loop())
  {
      ...
  }

  runTime.write();
}

/* Finalize */
```

```
// system/controlDict OpenFOAM config file
functions
{
   preCICE_Adapter
   {
     type preciceAdapterFunctionObject;
     libs ("libpreciceAdapterFunctObj.so");
   }
}
```

Isolating the adapter: Function Objects

```
/* Start the solver */

Info<<"\nStarting time loop\n"<< endl;
while (runTime.run()) {
  #include "readTimeControls.H"
  #include "compressibleCourantNo.H"
  #include "setDeltaT.H"

  runTime++;

  /* solve the equations */
  #include "rhoEqn.H"
  while (pimple.loop())
  {
      ...
  }

  runTime.write();
}

/* Finalize */
```

```
// system/controlDict OpenFOAM config file
functions
{
   preCICE_Adapter
   {
     type preciceAdapterFunctionObject;
     libs ("libpreciceAdapterFunctObj.so");
   }
}
```

```
// 0/T OpenFOAM config file
interface
{
    type            fixedValue;
    value           uniform 300;
}
// other types: fixedGradient, mixed
```

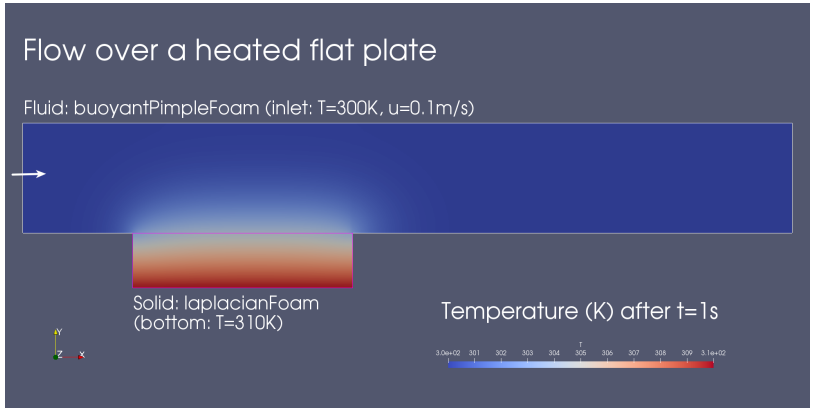
Coupling boundary patches, problem & solver type: `precice-adapter-config.yml`

Tutorials

On `www.precice.org/resources` (step-by-step):

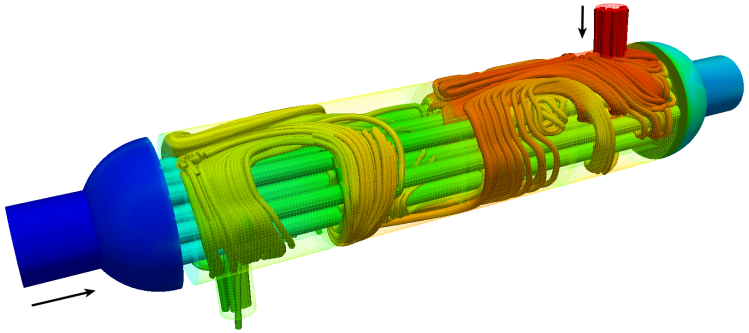
Flow above a heated plate

- Demo case, bundled with the adapter
- buoyantPimpleFoam + laplacianFoam
- Learn how to use the OpenFOAM adapter



Shell-and-Tube Heat Exchanger

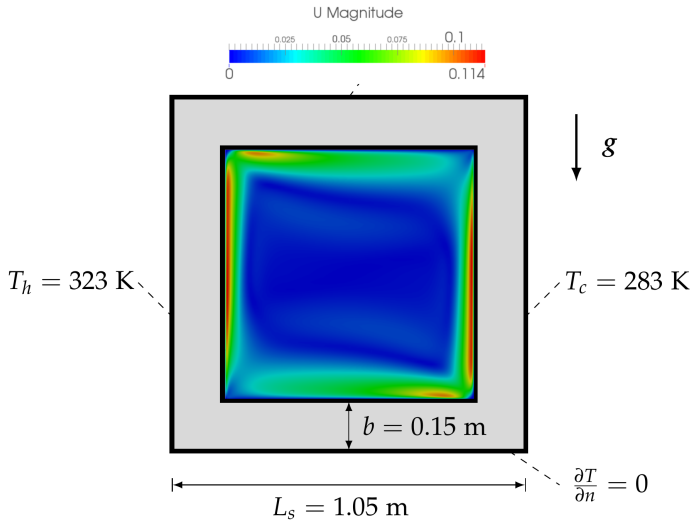
- Larger case, in `precice/tutorials`
- buoyantSimpleFoam (x2) + CalculiX
- Learn how to do multi-coupling

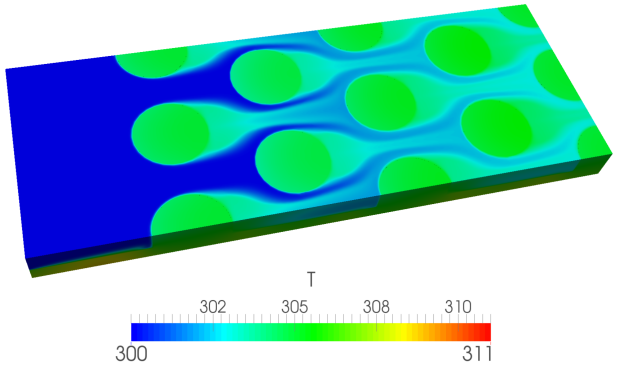


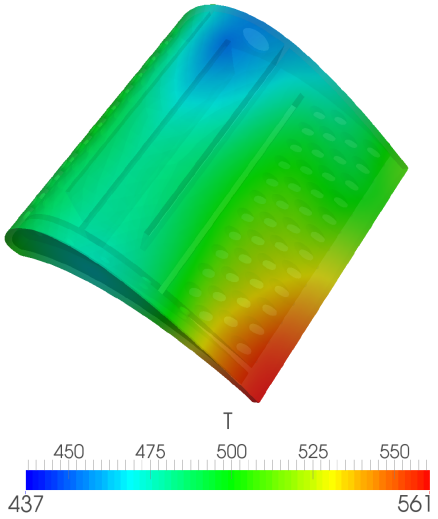
More examples with OpenFOAM

- Natural convection cavity
  - OpenFOAM + CalculiX
  - Transient
  - Robin-Robin serial-implicit coupling, IQN-ILS
- Pin-Fin cooling system
  - OpenFOAM + CalculiX
  - Steady-state
  - Robin-Robin parallel-implicit coupling, IQN-ILS
- Cooling of a turbine blade
  - OpenFOAM + CalculiX (or Code_Aster)
  - Steady-state
  - Robin-Robin parallel-explicit coupling

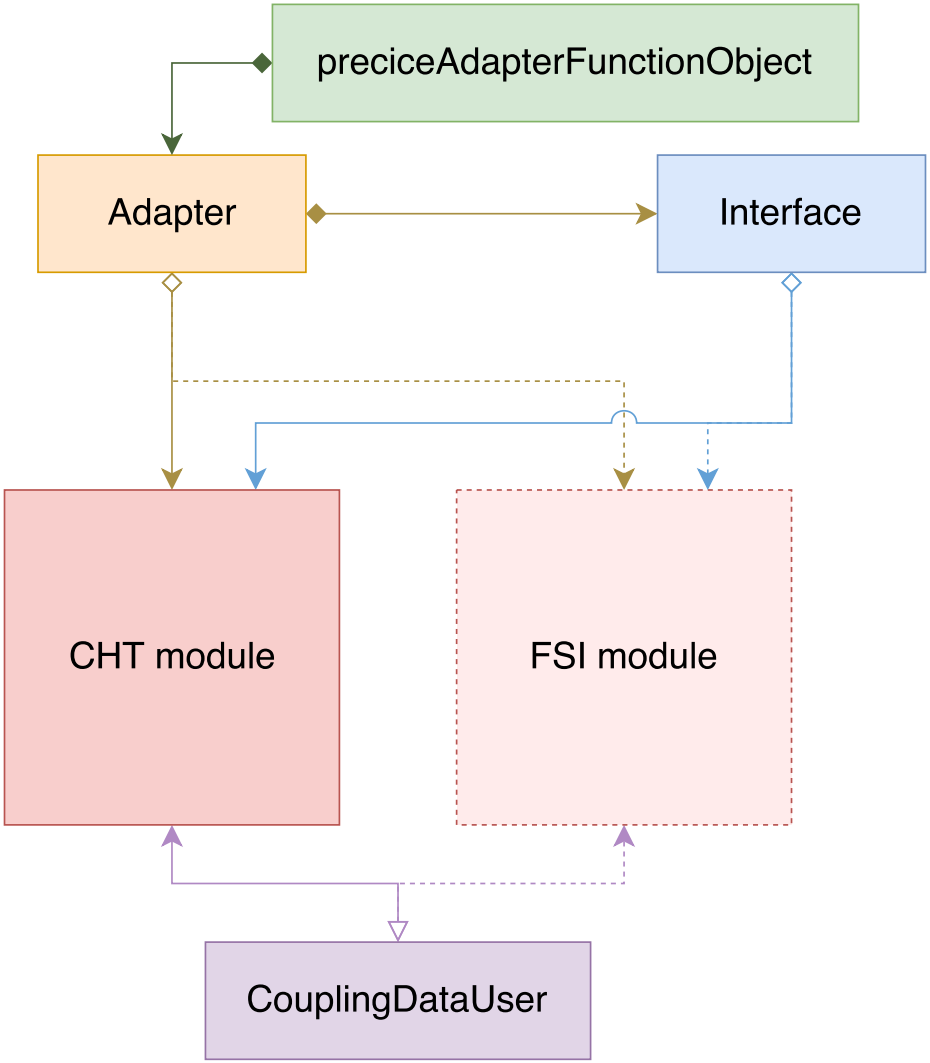
(simulations by L. Cheung Yau, 2016)







A modular design



Adding new features: Fluid-Structure Interaction

```
|openfoam-adapter/
|- preciceAdapterFunctionObject.C
|- Adapter.C
|- CouplingDataUser.C
|- CHT/
|   |- CHT.C
|   |- HeatFlux.C
|   |- HeatTransferCoefficient.C
|   |- KappaEffective.C
|   |- SinkTemperature.C
|   |- Temperature.C
|   |- ...
|- FSI/
|   |- Displacement.C
|   |- Force.C
|   |- FSI.C
|   |- ...
|- Interface.C
|- Utilities.C
...
```

Create a module for fluid-structure interaction #7

Open **MakisH** opened this issue on Nov 27, 2017 · 2 comments

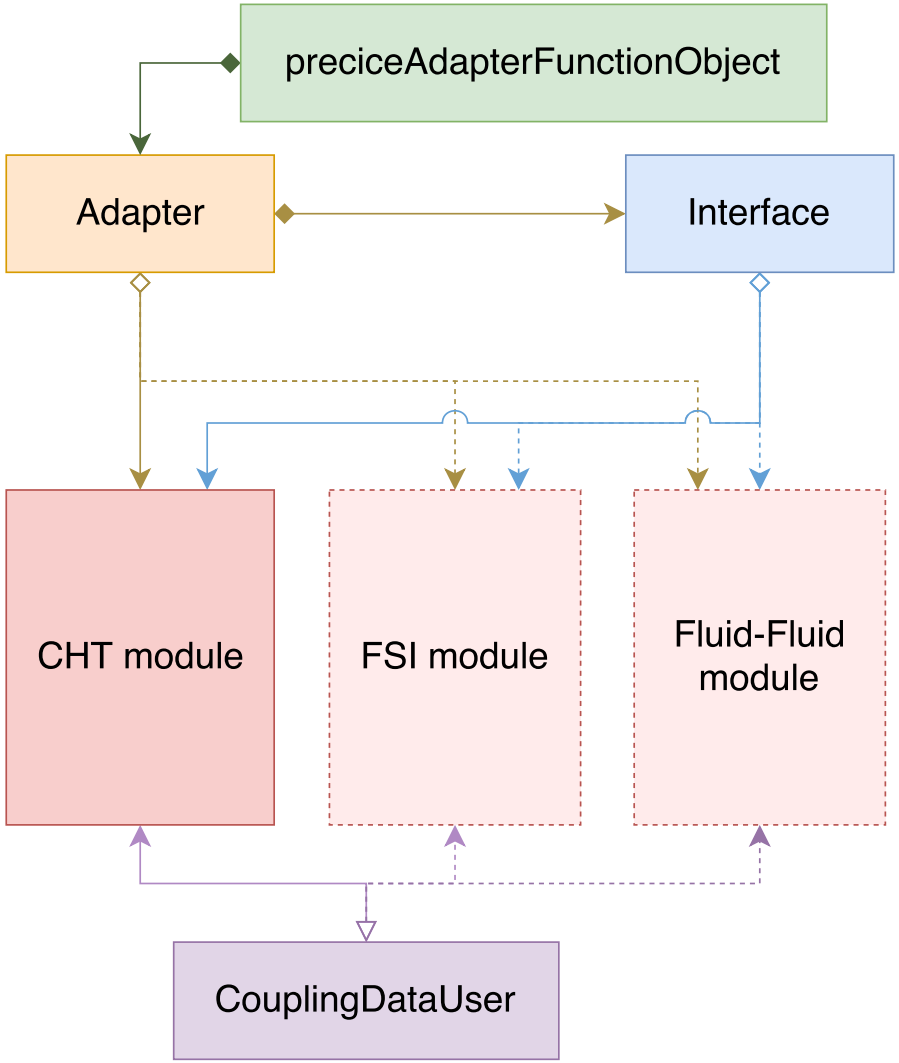
MakisH commented on Nov 27, 2017 • edited — Collaborator

In order to support mechanical fluid-structure interaction, we need a module similar to the one for conjugate heat transfer. The adapter also needs a few additions that can also be tested in this type of problem.

Roughly, the following sub-tasks are required:

- [x] Resize the data buffers for vector data (see the methods `preciceAdapter::Interface::addCouplingDataWriter` and `preciceAdapter::Interface::addCouplingDataReader` in the Interface.C.
- [x] Create the files `FSI.H` and `FSI.C`, similarly to the CHT.H and CHT.C. They should declare and define the methods `configure(const YAML::Node adapterConfig)`, `addWriters(std::string dataName, Interface * interface)`, and `addReaders(std::string dataName, Interface * interface)`. These methods must be called in the Adapter.C in two places (see comments with `NOTE`). A distinction among different solver types may need to be defined (most probably different than the one for CHT). Everything should be in the `FSI` namespace.
- [x] Create dummies of the new boundary conditions or *coupling data users*. These classes need to inherit from the `CouplingDataUser` class and to define the `write(double * buffer)` and `read(double * buffer)` methods. They should be in the namespace `FSI`.
- [ ] Implement the new coupling data users: Force.
- [ ] Implement the new coupling data users: Displacement.
- [x] Create objects of the new coupling data users, according to the adapter's configuration file.
- [x] Add an option to enable the FSI module in the `preciceAdapter::Adapter::configFileRead()`.
- [ ] If any other types need to be checkpointed, add them in the `preciceAdapter::Adapter::setupCheckpointing`, `preciceAdapter::Adapter::readCheckpoint`, and `preciceAdapter::Adapter::writeCheckpoint()` methods and create the respective `preciceAdapter::Adapter::addCheckpointField(...)`.
- [ ] Declare and create dummies of the virtual methods `updateMesh(const mapPolyMesh& mpm)` and `movePoints(const polyMesh& mesh)` in the preciceAdapterFunctionObject.H and

Planned: Fluid-Fluid coupling



Does it work with “chocolate” OpenFOAM?

Known to work with:

The OpenFOAM Foundation: 4.0 – dev

ESI - OpenCFD: v1706

Currently does not work with:

The OpenFOAM Foundation: $\leq$ 3.0

ESI - OpenCFD: $\leq$ v1606+

foam-extend: any version

+

OpenFOAM

References

preCICE **preCICE – A Fully Parallel Library for Multi-Physics Surface Coupling**
H.-J. Bungartz, B. Gatzhammer, F. Lindner, M. Mehl, K. Scheufele, A. Shukaev, B. Uekermann, 2016
In Computers and Fluids, Volume 141, p. 250—258. Elsevier.

Adapters **Official preCICE Adapters for Standard Open-Source Solvers**
B. Uekermann, H.-J. Bungartz, L. Cheung Yau, G. Chourdakis, A. Rusch, 2017
7th GACM Colloquium on Computational Mechanics for Young Scientists from Academia

Thesis 2 **A general OpenFOAM adapter for the coupling library preCICE**
G. Chourdakis, 2017
Master's thesis, Institut für Informatik, Technische Universität München

Thesis 1 **Conjugate Heat Transfer with the Multiphysics Coupling Library preCICE**
L. Cheung Yau, 2016
Master's thesis, Institut für Informatik, Technische Universität München

FOAM-FSI **Efficient numerical methods for partitioned fluid-structure interaction simulations**
D. Blom, 2017
Dissertation, Delft University of Technology

foam-extend + deal.II **Kopplung von OpenFOAM und deal.II Gleichungslösern mit preCICE zur Simulation multiphysikalischer Probleme**
K. Rave, 2017
Master's thesis, Lehrstuhl für Strömungsmechanik, Universität Siegen

Summary

Can my OpenFOAM solver easily be coupled with preCICE?

- Conjugate Heat Transfer with standard solvers: Yes!
- Conjugate Heat Transfer with custom solvers: Yes, maybe with a few additions.
- Fluid-Structure Interaction: Coming soon...
- Fluid-Fluid coupling: Planned.

Summary

Can my OpenFOAM solver easily be coupled with preCICE?

- Conjugate Heat Transfer with standard solvers: Yes!
- Conjugate Heat Transfer with custom solvers: Yes, maybe with a few additions.
- Fluid-Structure Interaction: Coming soon...
- Fluid-Fluid coupling: Planned.

Take-away:

- Isolating the adapter code allows for flexibility.
- We would like to have similar callback functionality also in other solvers.
- Modular design allows for easy extension.

Summary

Can my OpenFOAM solver easily be coupled with preCICE?

- Conjugate Heat Transfer with standard solvers: Yes!
- Conjugate Heat Transfer with custom solvers: Yes, maybe with a few additions.
- Fluid-Structure Interaction: Coming soon...
- Fluid-Fluid coupling: Planned.

Take-away:

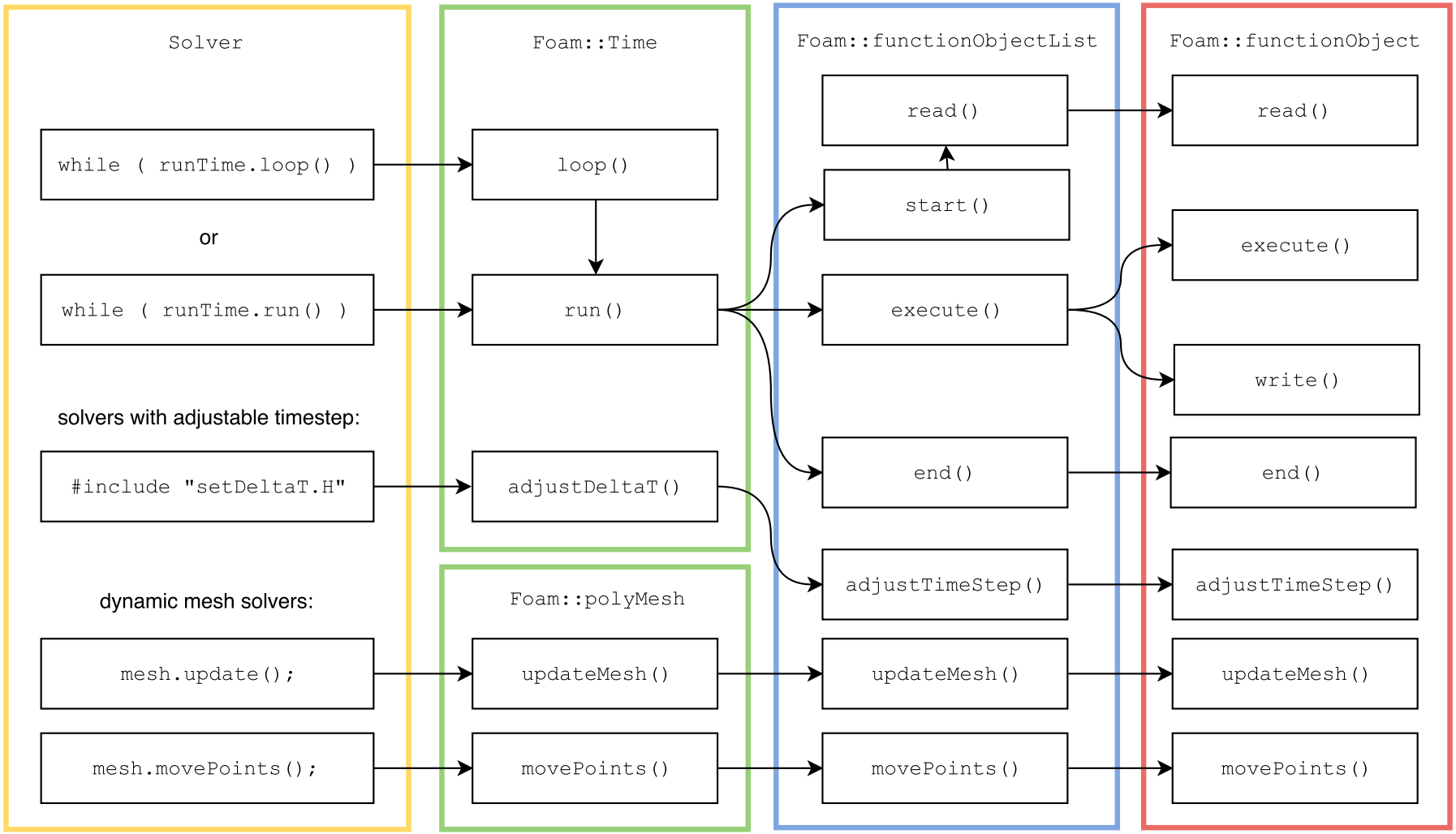
- Isolating the adapter code allows for flexibility.
- We would like to have similar callback functionality also in other solvers.
- Modular design allows for easy extension.

preCICE + OpenFOAM The OpenFOAM Foundation / OpenFOAM

Source/Wiki: `github.com/precice/openfoam-adapter` ★

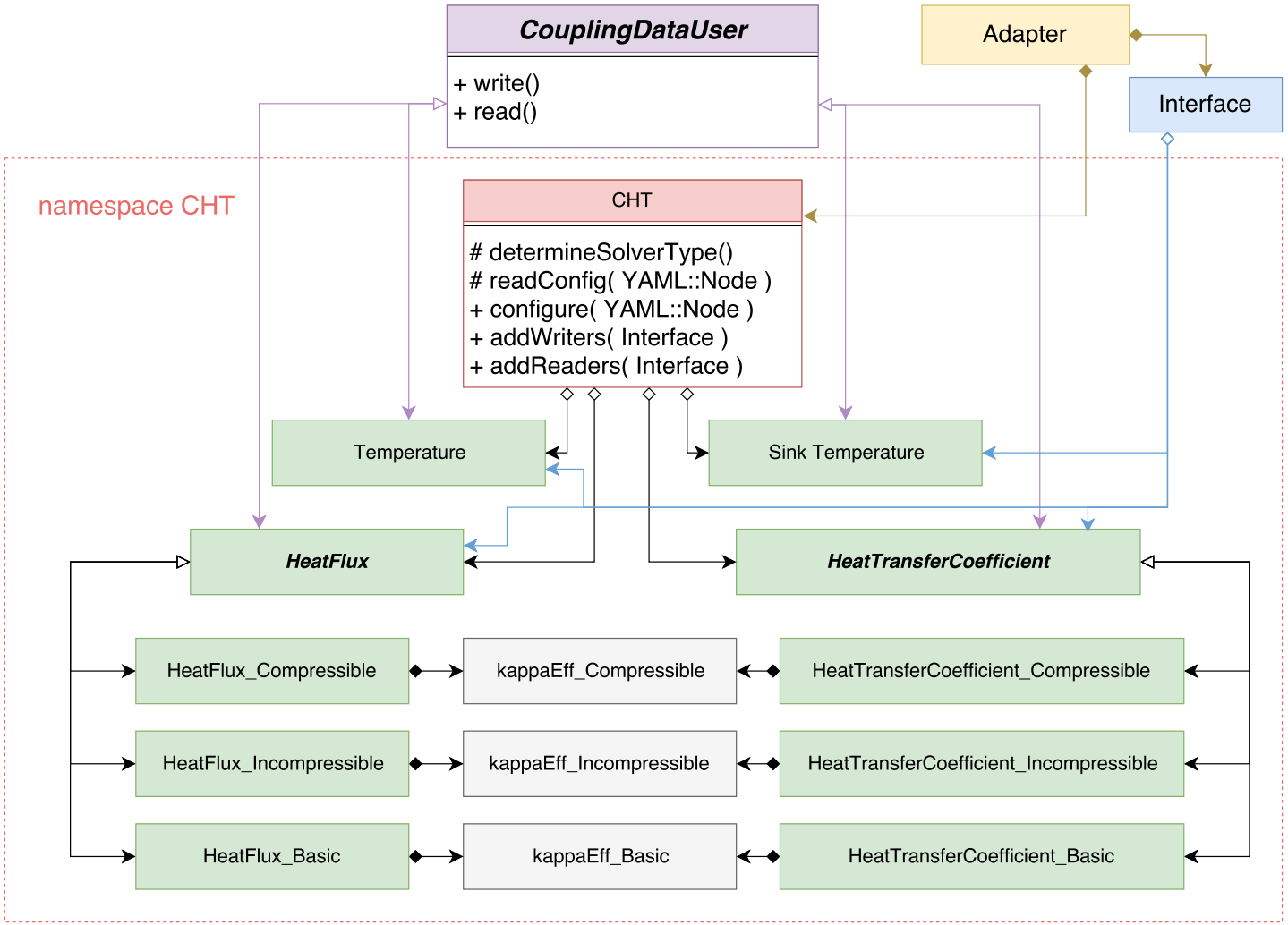
My e-mail: `chourdak@in.tum.de`

Additional slide: Function Objects



Callbacks in OpenFOAM function objects

Additional slide: The CHT Module



The Conjugate Heat Transfer module