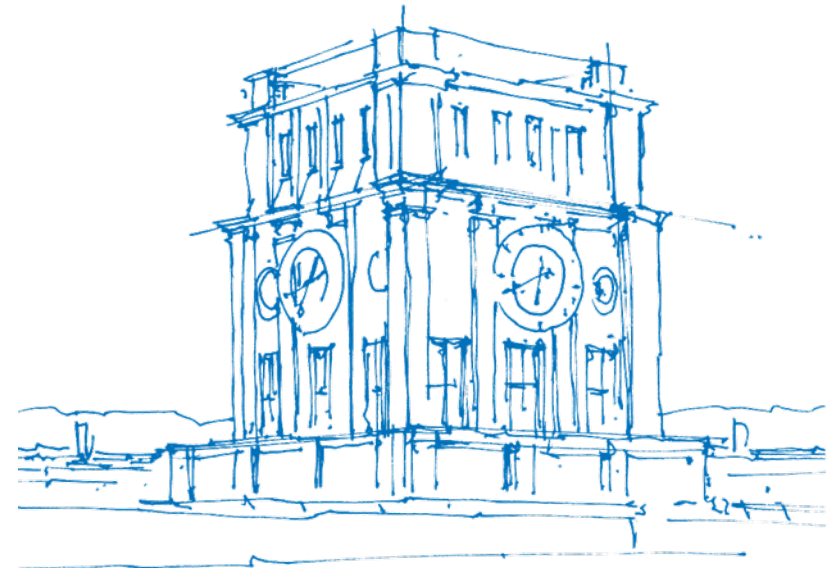


preCICE: multi-physics simulations reusing existing single-physics solvers

Gerasimos Chourdakis, Benjamin Uekermann, Benjamin R uth, Florian Lindner,
Miriam Mehl, Hans-Joachim Bungartz

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

PDESofT 2018 - Quality Assurance workshop
Lygra, Norway
May 31, 2018



TUM Uhrenturm

Multi-physics problems

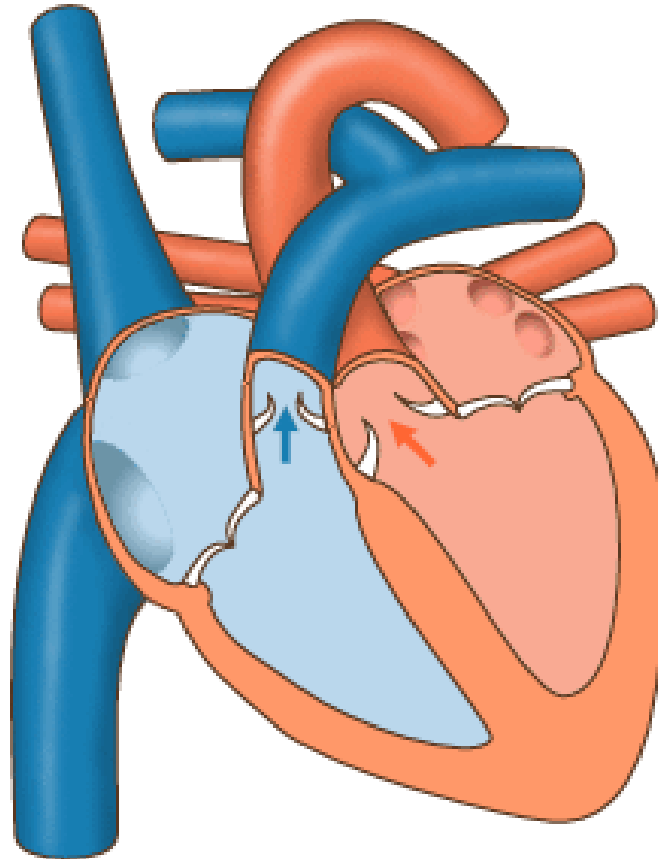


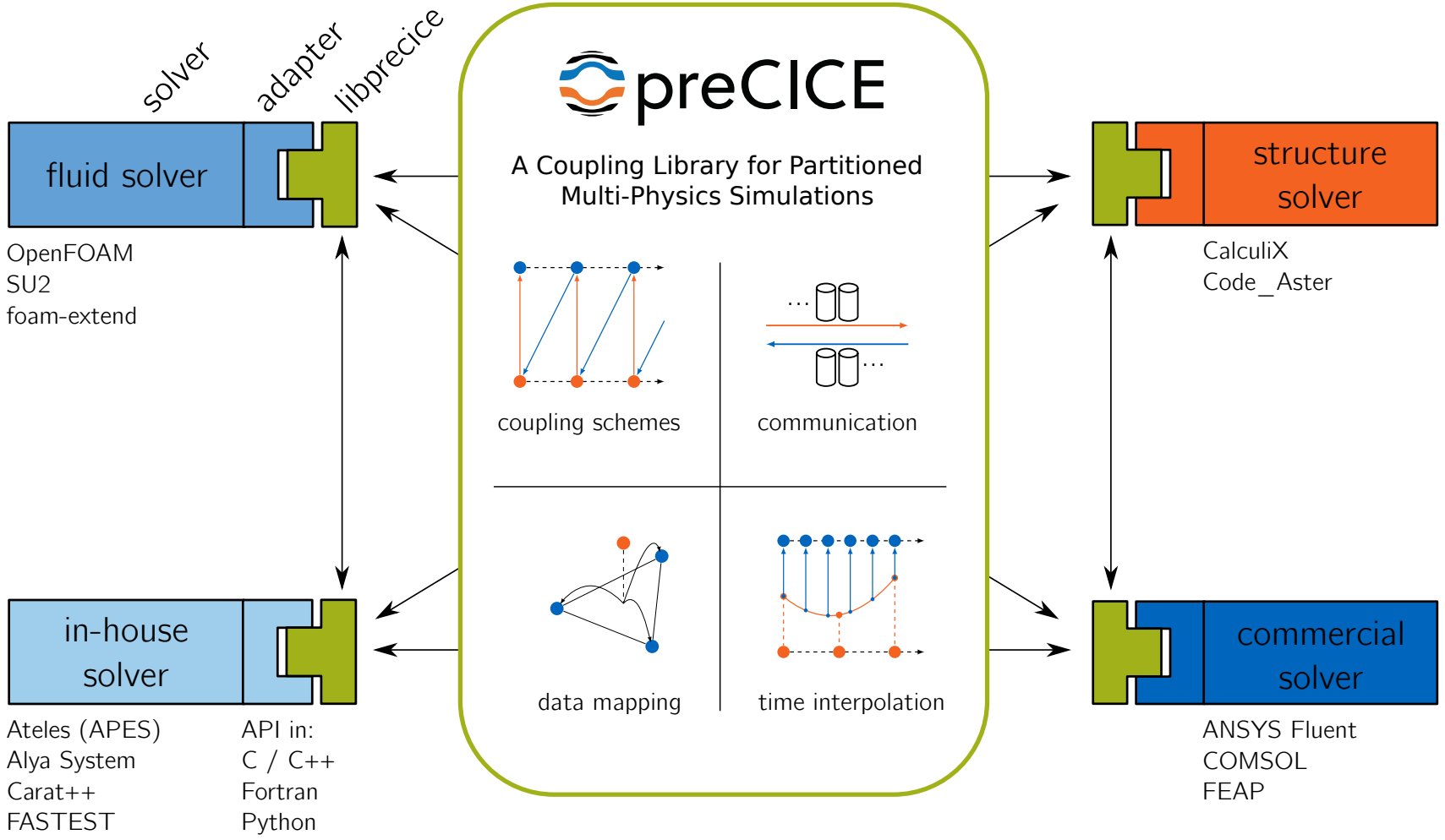
Image by josiño (Wikimedia Commons, public domain)

Agenda

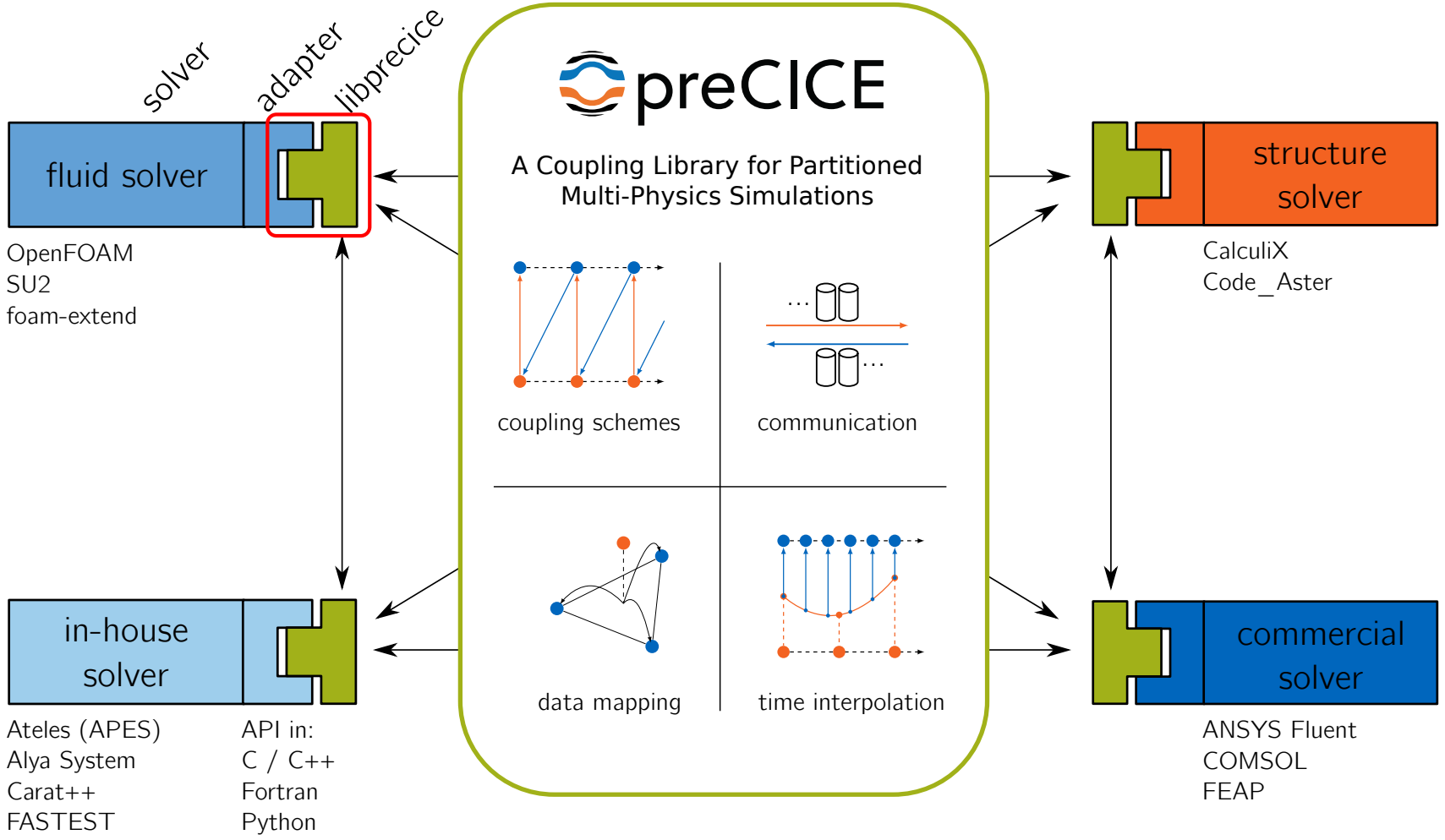
Part I preCICE overview

Part II Making preCICE sustainable

preCICE: Couple single-physics solvers



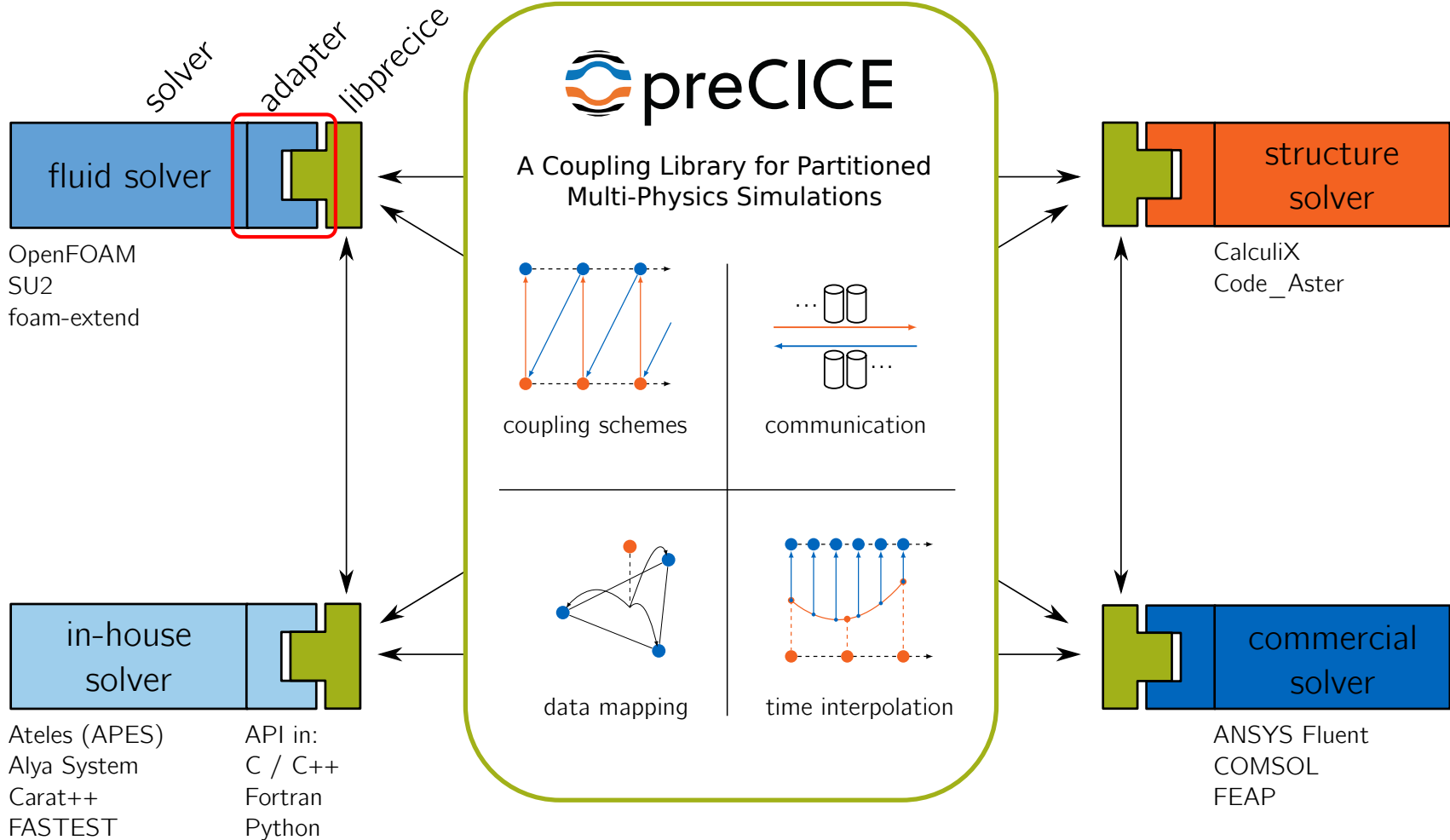
preCICE: Couple single-physics solvers



The preCICE Solver Interface (a glimpse)

```
1 int setMeshVertex (
2     int          meshID,
3     const double* position );
4
5 void writeScalarData (
6     int    dataID,
7     int    valueIndex,
8     double value );
9
10 double advance ( double computedTimestepLength );
11
12 void readScalarData (
13     int    dataID,
14     int    valueIndex,
15     double& value );
```

preCICE: Couple single-physics solvers



How to couple your own solver

```

1  /* Adapter: Initialize coupling
2     calls precice->initialize() */
3  adapter.initialize();
4
5  Info<< "\nStarting time loop\n" << endl;
6  while (adapter.isCouplingOngoing()) {
7     #include "readTimeControls.H"
8     #include "compressibleCourantNo.H"
9     #include "setDeltaT.H"
10
11     /* Adapter: Adjust solver time */
12     adapter.adjustSolverTimeStep();
13
14     runTime++;
15
16     /* Adapter: Receive coupling data */
17     adapter.readCouplingData();

```

```

18     /* solve the equations */
19     #include "rhoEqn.H"
20     while (pimple.loop())
21     {
22         ...
23     }
24
25     /* Adapter: Write in buffers */
26     adapter.writeCouplingData();
27
28     /* Adapter: advance the coupling
29        calls precice->advnace() */
30     adapter.advance();
31
32     if(adapter.isCouplTimeStepComplete())
33         runTime.write();
34 }

```


How to couple your own solver - better

```

1 // Don't modify the source code!
2 // Provide callbacks to inject code
3 // at runtime (plugins).
4 // OpenFOAM: "function objects"
5 Info<< "\nStarting time loop\n" << endl;
6 while (runTime.run()) {
7     #include "readTimeControls.H"
8     #include "compressibleCourantNo.H"
9     #include "setDeltaT.H"
10    //
11    //
12    //
13    //
14    runTime++;
15    //
16    //
17    //

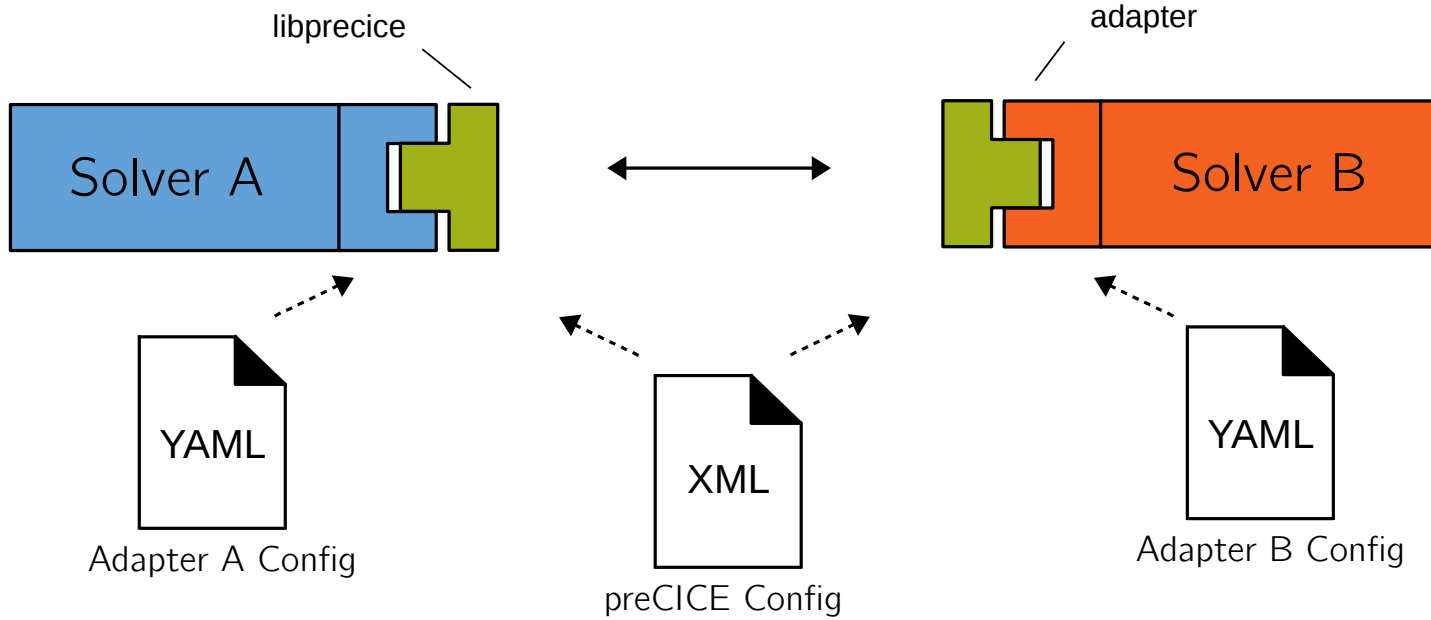
```

```

18 /* solve the equations */
19 #include "rhoEqn.H"
20 while (pimple.loop())
21 {
22     ...
23 }
24 //
25 //
26 //
27 //
28 //
29 //
30 //
31 //
32 //
33     runTime.write();
34 }

```

How to run simulations?



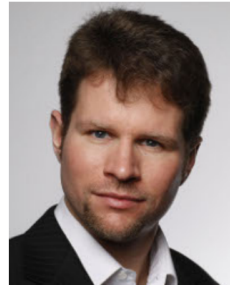
To run the simulation, just execute the solvers as usual.

Past & Present

Based on FSI*ce. 10+ years and 3 PhD generations later:



Miriam Mehl
U Stuttgart



Florian Lindner
U Stuttgart



Amin
Totounferoush
U Stuttgart



Alexander Rusch
ETH/COPLON



Hans Bungartz
TUM



Benjamin R uth
TUM



Gerasimos
Chourdakis
TUM



Benjamin
Uekermann
TUM/COPLON

Previous contributors: Bernhard Gatzhammer, Klaudius Scheufele, Lucia Cheung, Alexander Shukaev, Peter Vollmer, Georg Abrams, ...

Part II

Making [preCICE | *anotherProjectName*] sustainable

“Doctor, I have sustainability problems...”

- Does making good software pay off in academia?
 - “How many publications?” - vs - “How many users?”
- Problems:
 - Scientific software is only a by-product of research
 - Researchers finish and leave
 - Exa-scale: more complex software needed, we cannot build it from scratch

“Doctor, I have sustainability problems...”

- Does making good software pay off in academia?
 - “How many publications?” - vs - “How many users?”
- Problems:
 - Scientific software is only a by-product of research
 - Researchers finish and leave
 - Exa-scale: more complex software needed, we cannot build it from scratch
- Why bother?
 - Better software invites more people to invest time on it
 - Bigger community → more testers and contributors, more citations
 - Your work is being used → higher motivation

“Doctor, I have sustainability problems...”

- Does making good software pay off in academia?
 - “How many publications?” - vs - “How many users?”
- Problems:
 - Scientific software is only a by-product of research
 - Researchers finish and leave
 - Exa-scale: more complex software needed, we cannot build it from scratch
- Why bother?
 - Better software invites more people to invest time on it
 - Bigger community → more testers and contributors, more citations
 - Your work is being used → higher motivation

Question: How to advocate better scientific software?

“Doctor, I have sustainability problems...”

- Does making good software pay off in academia?
 - “How many publications?” - vs - “How many users?”
- Problems:
 - Scientific software is only a by-product of research
 - Researchers finish and leave
 - Exa-scale: more complex software needed, we cannot build it from scratch
- Why bother?
 - Better software invites more people to invest time on it
 - Bigger community → more testers and contributors, more citations
 - Your work is being used → higher motivation

Question: How to advocate better scientific software?

An answer from the USA:

“Better Scientific Software” (BSSw),

“Extreme-scale Scientific Software Development Kit” (xSDK),

permanent developer positions

What are we? What do we want?

We are **developers!** We want to:

1. Collaborate effectively
2. Understand the code & usage
3. Contribute easily
4. Focus on features
5. Feel useful (and buy beer) → get users!

What are we? What do we want?

We are **developers**! We want to:

1. Collaborate effectively
2. Understand the code & usage
3. Contribute easily
4. Focus on features
5. Feel useful (and buy beer) → get users!

They are **users**! They want to:

1. Install it easily
2. Get support
3. Learn it easily
4. Trust their results
5. Trust that the project will stay
6. Know that such a project exists

What are we? What do we want?

We are **developers**! We want to:

1. Collaborate effectively
2. Understand the code & usage
3. Contribute easily
4. Focus on features
5. Feel useful (and buy beer) → get users!

They are **users**! They want to:

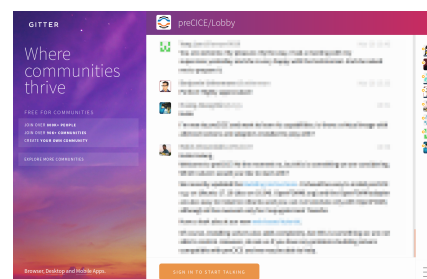
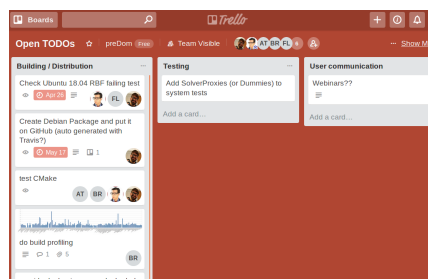
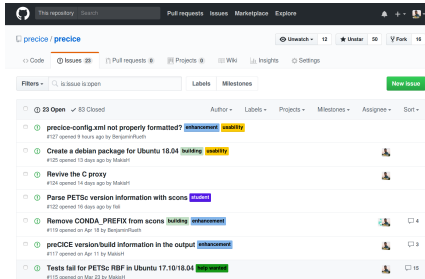
1. Install it easily
2. Get support
3. Learn it easily
4. Trust their results
5. Trust that the project will stay
6. Know that such a project exists

The DFG project preDOM: *Domesticating preCICE*

DFG call: “Research Software Sustainability” (2017)

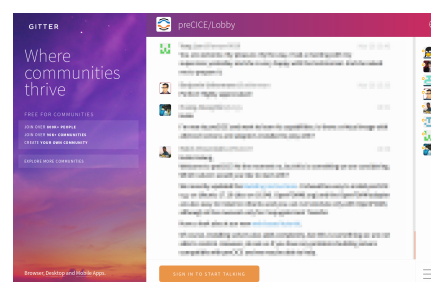
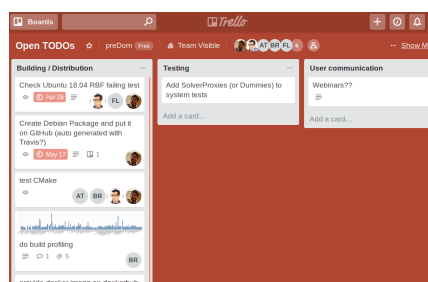
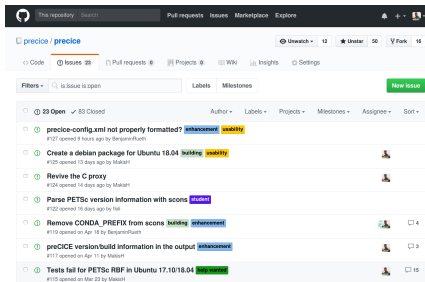
Dev wish 1: Collaborate effectively

- GitHub issues: “Can be closed with a commit”
- Trello: General TODOs and ideas
- Gitter: Quick questions (developers, users)
- Mailing list: Longer discussions
- TelCo: Weekly phone conferences with agenda and moderator in round-robin
- “Coding days”: On-site, once per three months
- preCICE user conferences (e.g. ECCOMAS ECFD 2018)



Dev wish 1: Collaborate effectively

- GitHub issues: “Can be closed with a commit”
- Trello: General TODOs and ideas
- Gitter: Quick questions (developers, users)
- Mailing list: Longer discussions
- TelCo: Weekly phone conferences with agenda and moderator in round-robin
- “Coding days”: On-site, once per three months
- preCICE user conferences (e.g. ECCOMAS ECFD 2018)



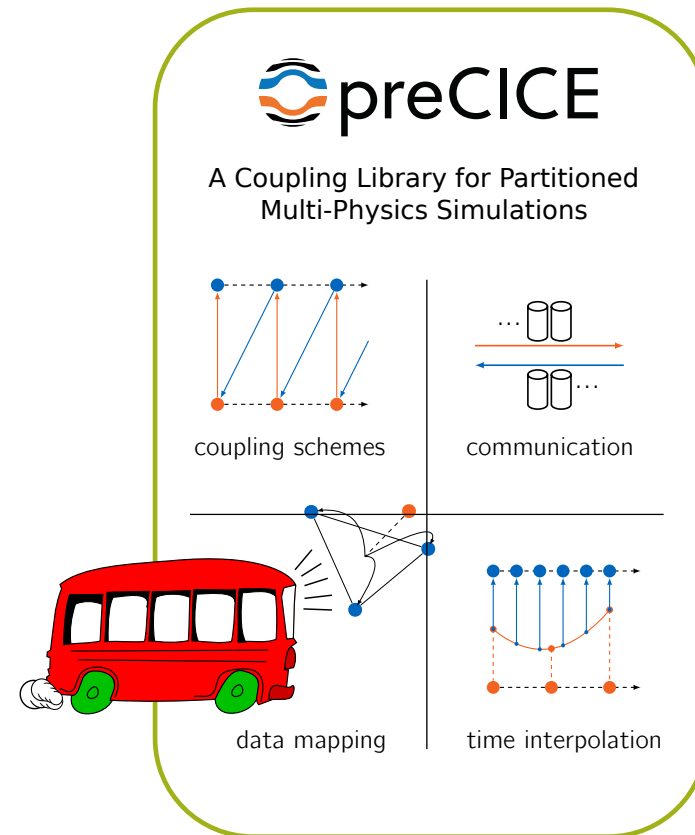
Question: How to collaborate with many universities? How to make decisions?

Dev wish 2: Understand the code & usage

- User documentation: GitHub Wiki
 - Step-by-step tutorials (also web-based)
 - Immediately improved after user feedback
- Developer documentation: Doxygen

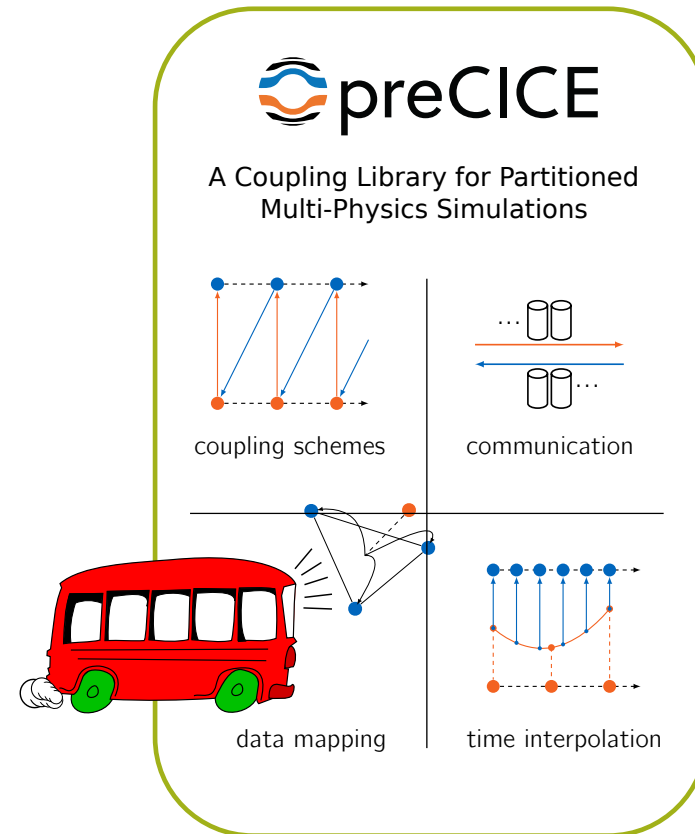
Dev wish 2: Understand the code & usage

- User documentation: GitHub Wiki
 - Step-by-step tutorials (also web-based)
 - Immediately improved after user feedback
- Developer documentation: Doxygen
- Bus factor: who knows what?
 - Share the knowledge via non-expert talks.
 - Push for open discussions (avoid private, 1-to-1 communication).



Dev wish 2: Understand the code & usage

- User documentation: GitHub Wiki
 - Step-by-step tutorials (also web-based)
 - Immediately improved after user feedback
- Developer documentation: Doxygen
- Bus factor: who knows what?
 - Share the knowledge via non-expert talks.
 - Push for open discussions (avoid private, 1-to-1 communication).



Question: How to increase the bus factor?

Dev wish 3: Contribute easily

- Git: Moved from private GitLab to public GitHub
- Website: GitHub Pages (Jekyll)
- User documentation: GitHub Wiki

Testing: contribute with confidence

- Travis CI builds and tests every commit
- Unit and integration tests with Boost
- Experimental: System tests with Docker
 - Special situation: test with several solvers, multiple languages
 - New solver versions may break compatibility

preCICE [Features](#) [FAQ](#) [Coupled Codes](#) [Resources](#) [Testimonials](#) [Publications](#) [About](#)

Welcome to preCICE

News: preCICE release v1.1.1 available since Apr 19, 2018

Give us your feedback: [please use this form](#). It's important for us!

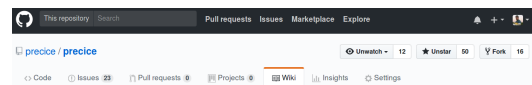
preCICE is now on Twitter: Follow us at [@preCICE_org](#) to always stay up-to-date!

preCICE in the media: Feature on [Science Node](#)

preCICE (Precise Code Interaction Coupling Environment) is a coupling library for partitioned multi-physics simulations, including, but not restricted to fluid-structure interaction and conjugate heat transfer simulations. Partitioned means that preCICE couples existing programs (solvers) capable of simulating a subpart of the complete physics involved in a simulation. This allows for the high flexibility that is needed to keep a decent time-to-solution for complex multi-physics scenarios.

The software offers methods for transient equation coupling, communication means, and data mapping schemes. **Ready-to-use adapters** for well-known commercial and open-source solvers, such as OpenFOAM, SU2, or CalculiX, are available. Adapters for in-house codes can be implemented and validated in only a few weeks.

preCICE is an open-source software under the LGPL3 license and available on [GitHub](#).



Home

Makis Chourdakis edited this page 13 days ago · 89 revisions

Welcome to the preCICE Wiki! Here, we collect information for users. Information only relevant for developers is given in our [Doxygen documentation](#).

If you want to contact us or if have any question, please write to our [mailing list](#) or in our [Gitter chat room](#). Please include at least the `precice-config.xml` file whenever you ask for support! Also, [give us some feedback!](#)

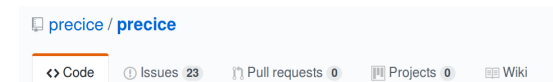
You don't know yet what preCICE is? Have a look at our [website](#) or our [reference paper](#).

Missing a feature? Maybe it is already on our [Roadmap!](#) If not, tell us!

1. Building preCICE

- [Building on Linux/macOS](#)
- [Building on Windows \(experimental\)](#)

If you want to use preCICE, you are currently required to build preCICE yourself (binary packages are [expected soon](#)). To do this, please follow our [general building guidelines](#). Most users and also most developers use preCICE under Linux. Some also use macOS. Both systems are, therefore, actively tested and should work smoothly. We also had already Windows users. Building on Windows is possible, but requires a little bit more effort. A successful Windows user documented his steps [here](#).



Branch: **develop**

Commits on May 18, 2018

Move sending/receiving/broadcasting of vector<int> to Communication

ifloil committed 4 days ago ✓

Commits on May 16, 2018

Fix uninitialized variable warning.

ifloil committed 6 days ago ✓

Dev wish 3: Contribute easily

- Git: Moved from private GitLab to public GitHub
- Website: GitHub Pages (Jekyll)
- User documentation: GitHub Wiki

Testing: contribute with confidence

- Travis CI builds and tests every commit
- Unit and integration tests with Boost
- Experimental: System tests with Docker
 - Special situation: test with several solvers, multiple languages
 - New solver versions may break compatibility

preCICE | Features | FAQ | Coupled Codes | Resources | Testimonials | Publications | About

Welcome to preCICE

News: preCICE release v1.1.1 available since Apr 19, 2018

Give us your feedback: please use this form. It's important for us!

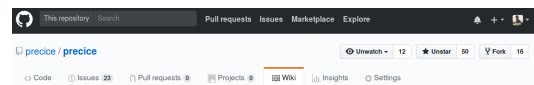
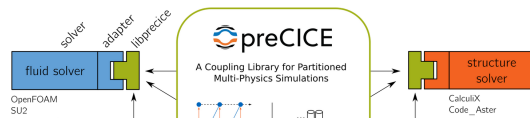
preCICE is now on Twitter: Follow us at @preCICE_org to always stay up-to-date!

preCICE in the media: Feature on Science Node

preCICE (Precise Code Interaction Coupling Environment) is a coupling library for partitioned multi-physics simulations, including, but not restricted to fluid-structure interaction and conjugate heat transfer simulations. Partitioned means that preCICE couples existing programs (solvers) capable of simulating a subpart of the complete physics involved in a simulation. This allows for the high flexibility that is needed to keep a decent time-to-solution for complex multi-physics scenarios.

The software offers methods for transient equation coupling, communication means, and data mapping schemes. Ready-to-use adapters for well-known commercial and open-source solvers, such as OpenFOAM, SU2, or CalculiX, are available. Adapters for in-house codes can be implemented and validated in only a few weeks.

preCICE is an open-source software under the LGPL3 license and available on GitHub.



Home

Makis Chourdakis edited this page 13 days ago · 89 revisions

Welcome to the preCICE Wiki! Here, we collect information for users. Information only relevant for developers is given in our [Doxygen documentation](#).

If you want to contact us or if have any question, please write to our [mailing list](#) or in our [Gitter chat room](#). Please include at least the `preCICE-config.xml` file whenever you ask for support! Also, [give us some feedback!](#)

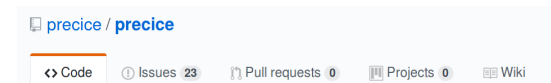
You don't know yet what preCICE is? Have a look at our [website](#) or our [reference paper](#).

Missing a feature? Maybe it is already on our [Roadmap!](#) If not, tell us!

1. Building preCICE

- [Building on Linux/macOS](#)
- [Building on Windows \(experimental\)](#)

If you want to use preCICE, you are currently required to build preCICE yourself (binary packages are expected soon). To do this, please follow our [general building guidelines](#). Most users and also most developers use preCICE under Linux. Some also use macOS. Both systems are, therefore, actively tested and should work smoothly. We also had already Windows users. Building on Windows is possible, but requires a little bit more effort. A successful Windows user documented his steps [here](#).



Branch: **develop**

Commits on May 18, 2018

Move sending/receiving/broadcasting of vector<int> to Communication

Itoli committed 4 days ago ✓

Commits on May 16, 2018

Fix uninitialized variable warning.

Itoli committed 6 days ago ✓

Question: How to do big system tests?

Question: How to convince scientists to write tests?

Dev wish 4: Focus on features

Reuse third-party implementations

Required:

- C++11, Boost
- Own testing framework → Boost tests
- Own XML-parser → libxml2
- Eigen for linear algebra

Optional:

- PETSc for parallel RBF interpolation
- MPI
- Python to inject code

Problem: newest feature set - vs - compatibility:

- Try to keep the currently maintained Ubuntu LTS compatible
- Complication: preCICE + PETSc + solvers need MPI

Dev wish 4: Focus on features

Reuse third-party implementations

Required:

- C++11, Boost
- Own testing framework → Boost tests
- Own XML-parser → libxml2
- Eigen for linear algebra

Optional:

- PETSc for parallel RBF interpolation
- MPI
- Python to inject code

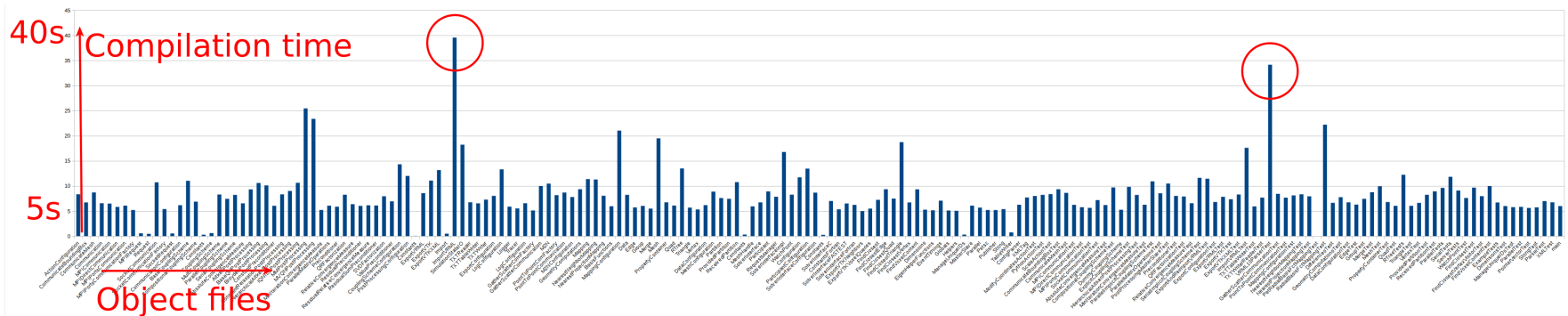
Problem: newest feature set - vs - compatibility:

- Try to keep the currently maintained Ubuntu LTS compatible
- Complication: preCICE + PETSc + solvers need MPI

Question: How to support multiple platforms? What about MPI?

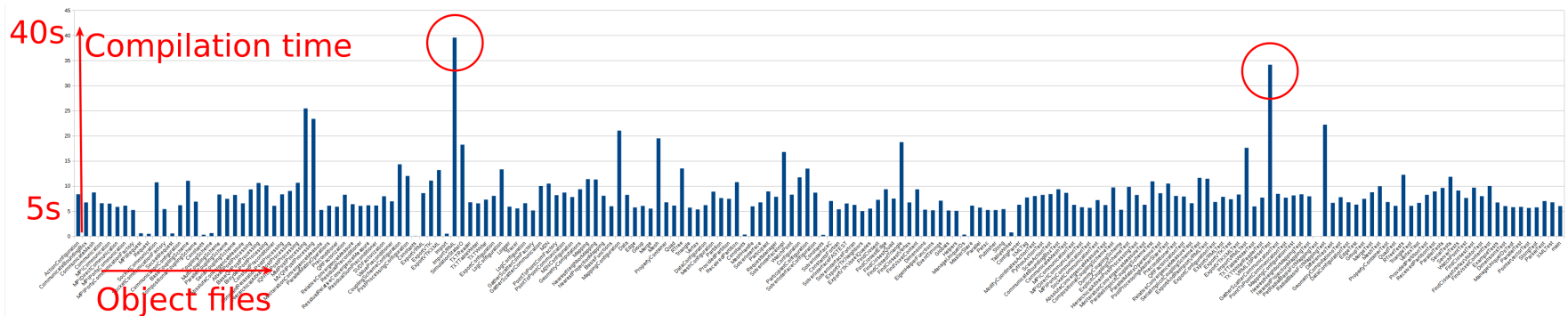
User wish 1: Install it easily

- Building:
 - Currently using SCons (Python)
 - Considering CMake (own language)
 - More standard
 - Easier for versioning
 - Faster? In our case, 1.5× speedup. (why?)
 - Profile building (GCC: `-ftime-report`)
 - Dead code: Seek & destroy
- Distribution:
 - Build from source (instructions for Ubuntu)
 - Experimental: Debian packages, Conda, Docker images
 - Also interesting: Spack, EasyBuild (for HPC)



User wish 1: Install it easily

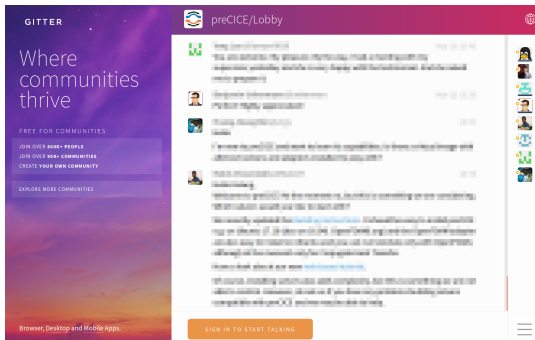
- Building:
 - Currently using SCons (Python)
 - Considering CMake (own language)
 - More standard
 - Easier for versioning
 - Faster? In our case, 1.5× speedup. (why?)
 - Profile building (GCC: `-ftime-report`)
 - Dead code: Seek & destroy
- Distribution:
 - Build from source (instructions for Ubuntu)
 - Experimental: Debian packages, Conda, Docker images
 - Also interesting: Spack, EasyBuild (for HPC)



Question: How to build and distribute (for HPC)?

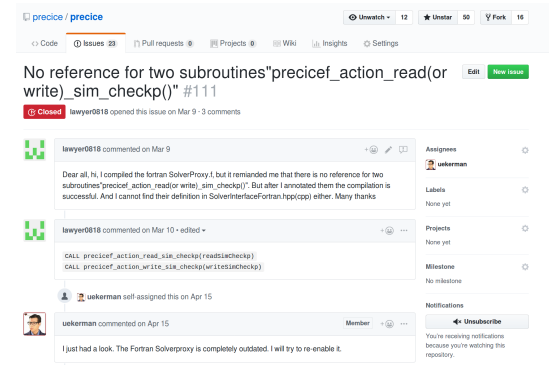
User wish 2: Get support

- Gitter - vs - Mailing list (our experience)
 - Gitter is an easier starting point, more interactive
 - More elaborate questions on the mailing list
 - Archiving: mailing list is crawled by search engines
 - Gitter reduces the non-interesting questions via e-mail
 - Non-developers also write in the mailing list
- Flow: Gitter → Mailing list → Issue



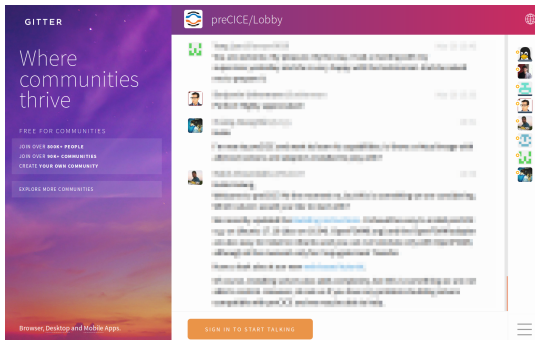
2018 Archives by thread

- Messages sorted by: [\[subject \]](#) [\[author \]](#) [\[date \]](#)
 - [More info on this list...](#)
- Starting:** Thu Jan 11 13:35:50 CET 2018
Ending: Wed May 16 13:53:06 CEST 2018
Messages: 85
- [Installation of preCICE on Ubuntu 16.04 and Calculix adapter](#) Claudio Caccia
 - [Installation of preCICE on Ubuntu 16.04 and Calculix adapter](#) Makis Chourdakis
 - [Installation of preCICE on Ubuntu 16.04 and Calculix adapter](#) Claudio Caccia
 - [Installation of preCICE on Ubuntu 16.04 and Calculix adapter](#) Rusch, Alexander
 - [Installation of preCICE on Ubuntu 16.04 and Calculix adapter](#) Florian Lindner
 - [CFD Time step](#) Cinquegrana Davide
 - [R: CFD Time step](#) Cinquegrana Davide
 - [CFD Time step](#) Benjamin Uekermann
 - [R: CFD Time step](#) Cinquegrana Davide
 - [R: CFD Time step](#) Cinquegrana Davide
 - [CFD Time step](#) Benjamin Uekermann
 - [R: CFD Time step](#) Cinquegrana Davide
 - [CFD Time step](#) Benjamin Uekermann
 - [PreCICE Tutorial with SU2 and CalculiX](#) Claudio Caccia
 - [PreCICE Tutorial with SU2 and CalculiX](#) Rusch, Alexander



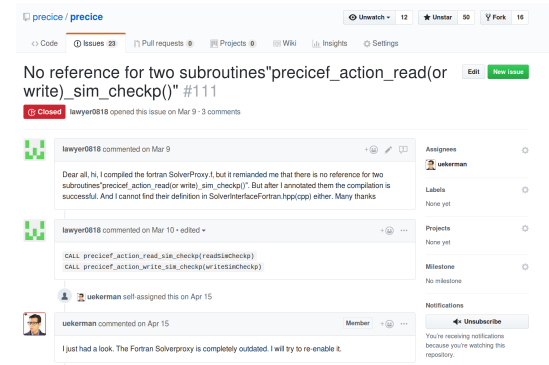
User wish 2: Get support

- Gitter - vs - Mailing list (our experience)
 - Gitter is an easier starting point, more interactive
 - More elaborate questions on the mailing list
 - Archiving: mailing list is crawled by search engines
 - Gitter reduces the non-interesting questions via e-mail
 - Non-developers also write in the mailing list
- Flow: Gitter → Mailing list → Issue



2018 Archives by thread

- Messages sorted by: [\[subject \]](#) [\[author \]](#) [\[date \]](#)
 - [More info on this list...](#)
- Starting:** Thu Jan 11 13:35:50 CET 2018
Ending: Wed May 16 13:53:06 CEST 2018
Messages: 85
- [Installation of preCICE on Ubuntu 16.04 and Calculix adapter](#) Claudio Caccia
 - [Installation of preCICE on Ubuntu 16.04 and Calculix adapter](#) Makis Chourdakis
 - [Installation of preCICE on Ubuntu 16.04 and Calculix adapter](#) Claudio Caccia
 - [Installation of preCICE on Ubuntu 16.04 and Calculix adapter](#) Rusch, Alexander
 - [Installation of preCICE on Ubuntu 16.04 and Calculix adapter](#) Florian Lindner
 - **CFD Time step** Cinquegrana Davide
 - **CFD Time step** Benjamin Uekermann
 - **R: CFD Time step** Cinquegrana Davide
 - **CFD Time step** Benjamin Uekermann
 - **R: CFD Time step** Cinquegrana Davide
 - **R: CFD Time step** Benjamin Uekermann
 - **CFD Time step** Benjamin Uekermann
 - **R: CFD Time step** Cinquegrana Davide
 - **CFD Time step** Benjamin Uekermann
 - [PreCICE Tutorial with SU2 and CalculiX](#) Claudio Caccia
 - [PreCICE Tutorial with SU2 and CalculiX](#) Rusch, Alexander



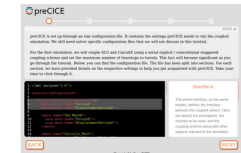
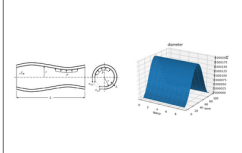
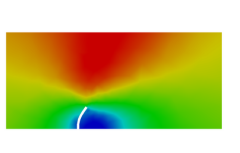
Question: How to support users? Does it scale?

Other user aspects

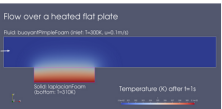
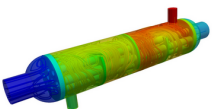
Wanted:

- Who are our users? (important for us!) → feedback form
- Smooth learning curve → tutorials (also online, maybe video)
- User trust → activity, responds to support requests, GitHub
- Tighter integration with solvers → ?

Fluid-Structure Interaction

Web-based tutorial	1D FSI Example	FSI with SU2 and CalculiX
Flow in a channel with an elastic flap	Flow through a deformable tube	Flow in a channel with an elastic flap
		

Conjugate Heat Transfer

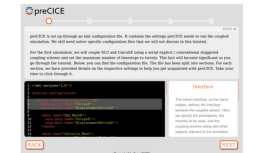
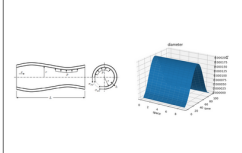
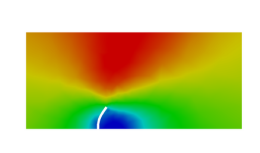
CHT with OpenFOAM	CHT with OpenFOAM and CalculiX
Flow over a heated flat plate	Shell-and-tube heat exchanger
	

Other user aspects

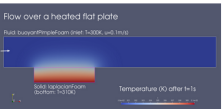
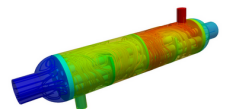
Wanted:

- Who are our users? (important for us!) → feedback form
- Smooth learning curve → tutorials (also online, maybe video)
- User trust → activity, responds to support requests, GitHub
- Tighter integration with solvers → ?

Fluid-Structure Interaction

Web-based tutorial	1D FSI Example	FSI with SU2 and CalculiX
Flow in a channel with an elastic flap	Flow through a deformable tube	Flow in a channel with an elastic flap
		

Conjugate Heat Transfer

CHT with OpenFOAM	CHT with OpenFOAM and CalculiX
Flow over a heated flat plate	Shell-and-tube heat exchanger
	

Questions:

- How to reach users?
- What conferences to present to?
- Where to organise satellite workshops?
- How to become part of solvers?

The big question

Best practices for any scientific software project?

- Get and analyze user feedback
- Discuss openly (i.e. in conferences such as this)

The big question

Best practices for any scientific software project?

- Get and analyze user feedback
- Discuss openly (i.e. in conferences such as this)

Question: How to communicate experiences and results?

Questions? (and recap of our own)

1. How to advocate better scientific software?
2. How to collaborate with many universities? How to make decisions?
3. How to build and distribute (for HPC)?
4. How to increase the bus factor?
5. How to support multiple platforms? What about MPI?
6. How to support users? Does it scale?
7. How to do big system tests?
8. How to convince scientists to write tests?
9. How to reach users? What type of conferences to present to?
10. Where to organise satellite workshops?
11. How to become part of solvers?
12. How to communicate experiences and results? (best practices)



Website: precice.org

Source/Wiki: github.com/precice ★

Contact us: precice.org/resources

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