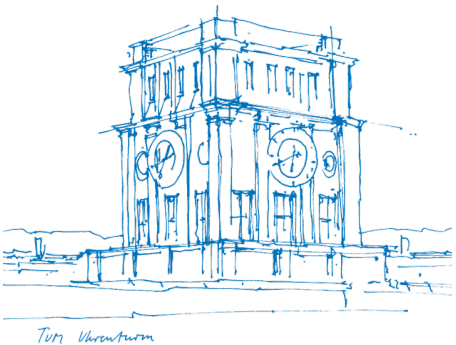
# Changes in preCICE Version 3

preCICE Workshop 2023

**Frédéric Simonis**
Technical University of Munich

15. February 2023



# Progress

## Version 3.0.0

No due date **63%** complete

72 open 121 closed

**Discussion in progress** means feedback welcome!

# No more SolverInterface

```
# preCICE setup
interface = precice.Interface("Solid", "../precice-config.xml", 0, 1)
```

# No more SolverInterface

```python
# preCICE setup
interface = precice.Interface("Solid", "../precice-config.xml", 0, 1)
```

## V3: Work in progress

```python
# preCICE setup
participant = precice.Participant("Solid", "../precice-config.xml", 0, 1)
```

# IDs and names

```python
# define coupling mesh
mesh_name = "Solid-Mesh"
mesh_id = interface.get_mesh_id(mesh_name)
vertices = coupling_sample.eval(ns.x)
vertex_ids = interface.set_mesh_vertices(mesh_id, vertices)

# coupling data
temperature_id = interface.get_data_id("Temperature", mesh_id)

temperature_values = interface.read_block_scalar_data(temperature_id, vertex_ids)
```

# IDs and names

## V3: Work in progress

```
# define coupling mesh
mesh_name = "Solid-Mesh"
vertices = coupling_sample.eval(ns.x)
vertex_ids = participant.set_mesh_vertices(mesh_name, vertices)

# coupling data
temperature_values = participant.read_block_scalar_data(
    mesh_name, "Temperature", vertex_ids
)
```

# Simpler connectivity

```python
# define coupling mesh
vertex_ids = interface.set_mesh_vertices(mesh_id, vertices)
a, b, c = vertex_ids[:3]

if interface.is_mesh_connectivity_required(mesh_id):
    ab_id = interface.set_mesh_edge(mesh_id, a, b)
    bc_id = interface.set_mesh_edge(mesh_id, b, c)
    ac_id = interface.set_mesh_edge(mesh_id, a, c)
    interface.set_mesh_triangle(mesh_id, ab_id, bc_id, ac_id)
```

# Simpler connectivity

```python
# define coupling mesh
vertex_ids = interface.set_mesh_vertices(mesh_id, vertices)
a, b, c = vertex_ids[:3]

if interface.is_mesh_connectivity_required(mesh_id):
    interface.set_mesh_triangle_with_edges(mesh_id, a, b, c)
```

# Simpler connectivity

## V3: Implemented

```
# define coupling mesh
vertex_ids = participant.set_mesh_vertices(mesh_name, vertices)
a, b, c = vertex_ids[:3]

if participant.requires_mesh_connectivity_for(mesh_name):
    participant.set_mesh_triangle(mesh_id, a, b, c)
```

# Simpler initial data

```
# define coupling mesh
mesh_id = interface.get_mesh_id("Solid-Mesh")
vertex_ids = interface.set_mesh_vertices(mesh_id, vertices)

precice_dt = interface.initialize()

# define coupling data
flux_id = interface.get_data_id("Heat-Flux", mesh_id)
if interface.is_action_required(precice.action_write_initial_data()):
    interface.write_block_scalar_data(flux_id, vertex_ids, initial_flux_values)
    interface.mark_action_fulfilled(precice.action_write_iteration_checkpoint())

interface.initializeData()
```

# Simpler initial data

## V3: Implemented

```
# define coupling mesh
mesh_name = "Solid-Mesh"
vertex_ids = participant.set_mesh_vertices(mesh_name, vertices)

# define coupling data
flux_name = "Heat-Flux"
if participant.requires_initial_data():
    participant.write_block_scalar_data(
      mesh_name, flux_name, vertex_ids, initial_flux_values
    )

precice_dt = participant.initialize()
```

# From actions to requirements

```
while interface.is_coupling_ongoing():
    # save checkpoint
    if interface.is_action_required(precice.action_write_iteration_checkpoint()):
        lhs_checkpoint = lhs0
        timestep_checkpoint = timestep
        interface.mark_action_fulfilled(precice.action_write_iteration_checkpoint())
    # ... solve and advance ...
    # read checkpoint if required
    if interface.is_action_required(precice.action_read_iteration_checkpoint()):
        lhs0 = lhs_checkpoint
        timestep = timestep_checkpoint
        interface.mark_action_fulfilled(precice.action_read_iteration_checkpoint())
    else:
        # visualize
```

# From actions to requirements

## V3: Implemented

```
while participant.is_coupling_ongoing():
    # save checkpoint
    if participant.requires_writing_checkpoint():
        lhs_checkpoint = lhs0
        timestep_checkpoint = timestep
    # ... solve and advance ...
    # read checkpoint if required
    if participant.requires_reading_checkpoint():
        lhs0 = lhs_checkpoint
        timestep = timestep_checkpoint
    else:
        # visualize
```

# Moving towards waveform interpolation

## V3: Implemented

```
while interface.is_coupling_ongoing():
    # read temperature from interface
    if interface.is_read_data_available()
        T_values = interface.read_block_scalar_data(temperature_id, vertex_ids)
        # apply T_values to nutils

    # solve nutils timestep

    # write heat fluxes to interface
    if interface.is_write_data_required(dt):
        # extract flux_values from nutils
        interface.write_block_scalar_data(flux_id, vertex_ids, flux_values)
```

# Simplify access to timestep sizes

```
precice_dt = interface.initialize()

while interface.is_coupling_ongoing():
  # Read data
  dt = min(solver_dt, precice_dt)
  # Solve timestep
  # Write data
  precice_dt = interface.advance(dt)
```

# Simplify access to timestep sizes

## V3: Discussion in progress

```
participant.initialize()

while participant.is_coupling_ongoing():
  # Read data
  precice_dt = participant.get_max_time_step_size()
  dt = min(solver_dt, precice_dt)
  # Solve timestep
  # Write data
  participant.advance(dt)
```

# Unified data access

```
while interface.is_coupling_ongoing():
    # read temperature from preCICE
    T_values = interface.read_block_scalar_data(temperature_id, vertex_ids)
    # apply T_values to nutils

    # solve nutils timestep

    # write heat fluxes to preCICE
    # extract flux_values from nutils
    interface.write_block_scalar_data(flux_id, vertex_ids, flux_values)
```

# Unified data access

## V3: Discussion in progress

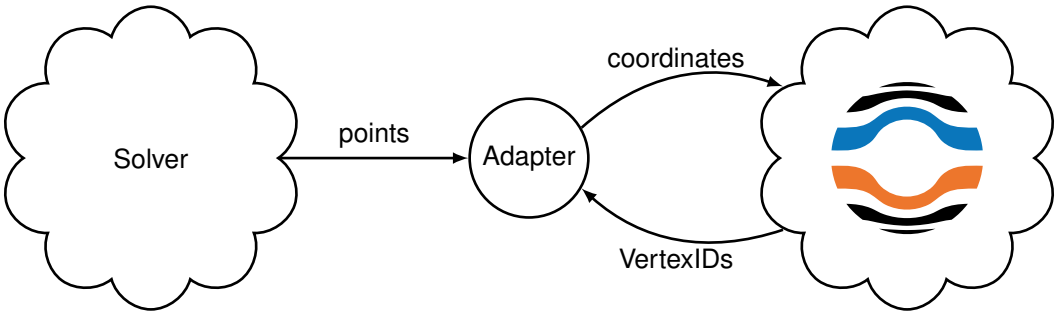
```
while participant.is_coupling_ongoing():
    # read temperature from preCICE
    T_values = participant.read_data(mesh_name, temperature_name, vertex_ids)
    # apply T_values to nutils

    # solve nutils timestep

    # write heat fluxes to preCICE
    # extract flux_values from nutils
    participant.write_data(mesh_name, flux_name, vertex_ids, flux_values)
```

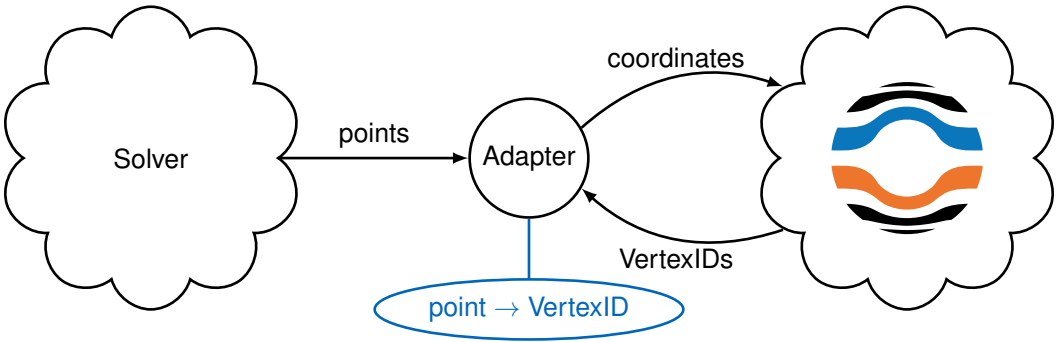
# Removing coordinate lookups

## V3: Implemented



# Removing coordinate lookups

## V3: Implemented



# Clarifying some tags

```
<use-mesh name="{string}" from="" safety-factor="0.5" provide="false"
          geometric-filter="on-secondary-ranks"  direct-access="false" />

<coupling-scheme:multi>
  <participant name="MySolver1" control="yes"/>
  <participant name="MySolver2" />
  <participant name="MySolver3" />
</coupling-scheme:multi>
```

# Clarifying some tags

## V3: Work in progress

```
<provide-mesh name="{string}" />
<receive-mesh name="{string}" from="" safety-factor="0.5"
              geometric-filter="on-secondary-ranks" direct-access="false"/>

<coupling-scheme:multi>
  <controller  name="MySolver1" />
  <participant name="MySolver2" />
  <participant name="MySolver3" />
</coupling-scheme:multi>
```

# New Wendland CX-polynomial RBF

| | |
|---|---|
| | thin-plate-splines |
| | multiquadrics |
| | inverse-multiquadrics |
| | volume-splines |
| | gaussian |
| | compact-tps-c2 |
| | compact-polynomial-c0 |
| **New** | compact-polynomial-c2 |
| **New** | compact-polynomial-c4 |
| | compact-polynomial-c6 |

# Defaults for acceleration

```
<acceleration:IQN-ILS>
    <data name="Displacement" mesh="Solid-Mesh" />
    <data name="Stress" mesh="Fluid-Mesh-Centers" />

    <preconditioner type="residual-sum" />
    <filter type="QR2" limit="1.2e-3" />
    <initial-relaxation value="0.1" />
    <max-used-iterations value="60" />
    <time-windows-reused value="15" />
</acceleration:IQN-ILS>
```

# Defaults for acceleration

## V3: Work in progress

```
<acceleration:IQN-ILS>
    <data name="Displacement" mesh="Solid-Mesh" />
    <data name="Stress" mesh="Fluid-Mesh-Centers" />
</acceleration:IQN-ILS>
```

# Order of coupling schemes

## V3: Implemented

```
<coupling-scheme:parallel-explicit>
  <participants first="This" second="OtherA" />
</coupling-scheme:parallel-explicit>

<coupling-scheme:serial-implicit>
  <participants first="This" second="OtherB" />
</coupling-scheme:serial-explicit>

<coupling-scheme:parallel-explicit>
  <participants first="This" second="OtherC" />
</coupling-scheme:parallel-explicit>
```

# Multiple implicit coupling schemes

```
<coupling-scheme:parallel-explicit>
  <participants first="This" second="OtherA" />
</coupling-scheme:serial-explicit>

<coupling-scheme:serial-implicit>
  <participants first="This" second="OtherB" />
</coupling-scheme:serial-explicit>

<coupling-scheme:serial-implicit>
  <participants first="This" second="OtherC" />
</coupling-scheme:serial-implicit>
```

# Multiple implicit coupling schemes

## V3: Implemented

```
<coupling-scheme:parallel-explicit>
  <participants first="This" second="OtherA" />
</coupling-scheme:parallel-explicit>

<coupling-scheme:multi>
  <controller  name="This" />
  <participant name="OtherB" />
  <participant name="OtherC" />
</coupling-scheme:multi>
```

H. Bungartz, F. Linder, M. Mehl, B. Uekermann. A plug-and-play coupling approach for parallel multi-field simulations. Comput Mech 55, 1119-1129 (2015). https://doi.org/10.1007/s00466-014-1113-2

# Dynamic meshes

## V3: Work in progress

```
<participant name="Solid">
  <provide-mesh name="SolidMesh" dynamic="true"/>
  <write-data mesh="SolidMesh" name="Temperature"/>
</participant>

while participant.is_coupling_ongoing():
  if i_need_to_remesh():
    new_coordinates = remesh()
    participant.reset_mesh("SolidMesh")
    vertex_ids = participant.set_mesh_vertices("SolidMesh", new_coordinates)

  participant.write_data("SolidMesh", "Temperature", vertex_ids, temperature_values)
  participant.advance(dt)
```

# Thank you for listening!

**Frédéric Simonis**
Chair of Scientific Computing
Technical University of Munich
**Mail** simonis@in.tum.de
**GitHub** fsimonis
**GitLab** fsimonis