Introduction to ORCAN

M. Kellner

Department of Materials Science 6, University of Erlangen-Nuremberg

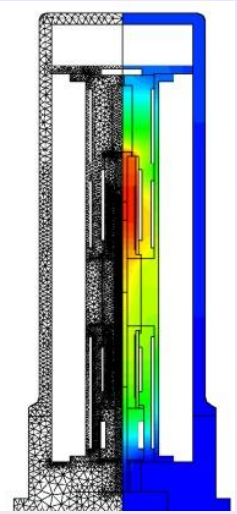
ORCAN Workshop, 26. April 2005

Outline

Software for Simulation of Crystal Growth

Crystal Growth Laboratory

- Development of simulation software since years.
- CrysVUn, STHAMAS/-3D licensed to industry.
- 2D or axisymmetric complex geometries.
- User friendly interface.



Research Project (CrysVUn3D)

- Fully 3D simulation in complex geometries.
- Focus on realistic modelling of thermal radiation.
- Completely new development of simulation software.

Why ORCAN?

Example (Thermal Radiation)

- Geometry import and management.
- Mesh generation and management.
- Heat conduction simulation.
- Linear equation system solver.
- Coupling between (different) meshes.
- Visualization.
- Graphical user interface.
- Parallelization.

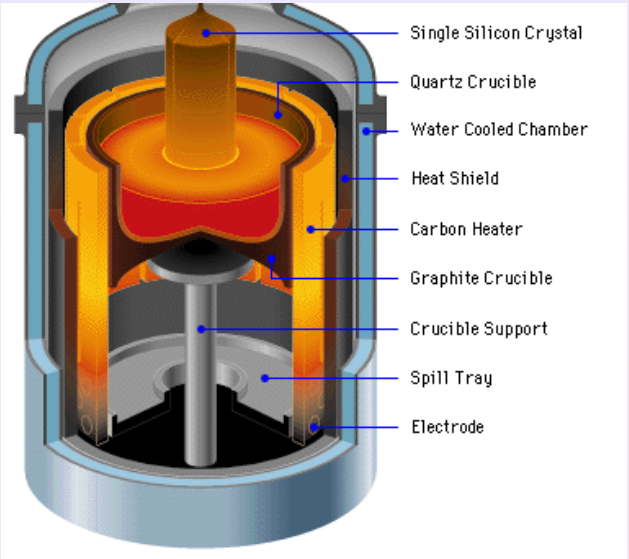


Figure: Czochralski furnace.

Drawback

A good deal of the time must be spend in preparation for the real task.

Why ORCAN?

Example (Thermal Radiation)

- Geometry import and management.
- Mesh generation and management.
- Heat conduction simulation.
- Linear equation system solver.
- Coupling between (different) meshes.
- Visualization.
- Graphical user interface.
- Parallelization.

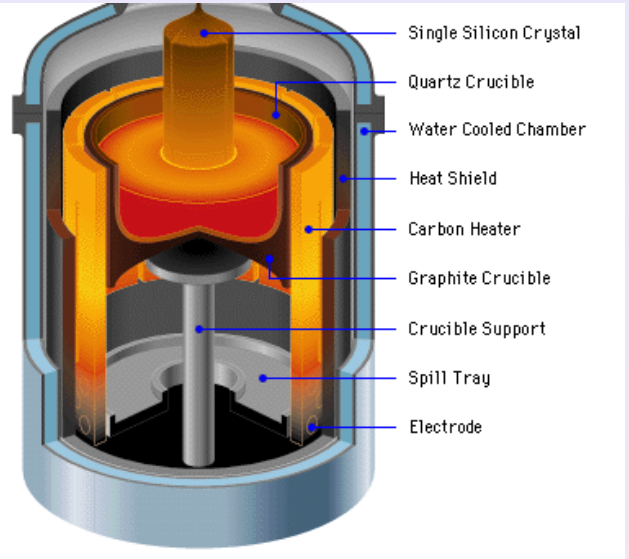


Figure: Czochralski furnace.

Drawback

A good deal of the time must be spend in preparation for the real task.

Why ORCAN?

Example (Large-Scale Software)

- Not all can be developed anew.
- Reuse of existing software packages.
- Integration of a specific product is critical.

- Highly interwoven application modules.
- Increasing efforts on modifications.
- Replacements more and more expensive.

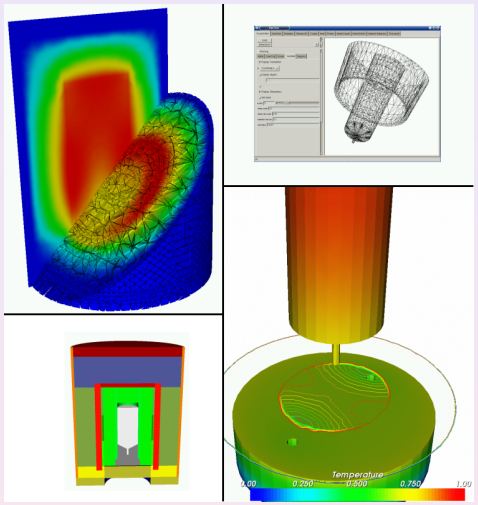


Figure: CrysVUn3D.

Drawback

Large application may fail if they reach a specific level of complexity.

Why ORCAN?

Example (Large-Scale Software)

- Not all can be developed anew.
- Reuse of existing software packages.
- Integration of a specific product is critical.

- Highly interwoven application modules.
- Increasing efforts on modifications.
- Replacements more and more expensive.

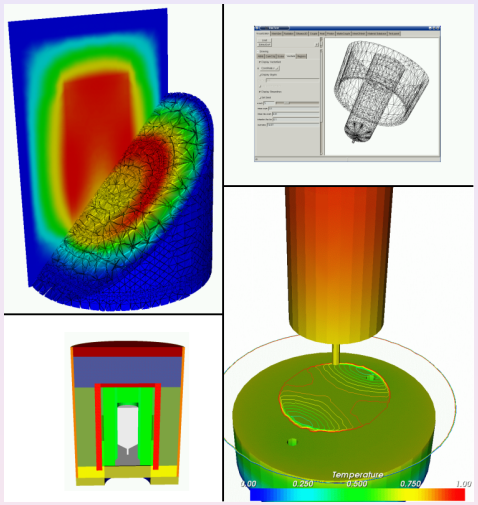


Figure: CrysVUn3D.

Drawback

Large application may fail if they reach a specific level of complexity.

Goals

We are looking for a framework which

- allows the decomposition of software in manageable modules.
  $\Rightarrow$ *components*
- allows a clear functional specification of a module.
  $\Rightarrow$ *interfaces*
- forces the developer to use the specification of a module.
  $\Rightarrow$ *data hiding*
- allows replacement of a module with a minimum of effort.
  $\Rightarrow$ *3rd party products*
- allows easy extension of the software by new modules.
  $\Rightarrow$ *plug-in mechanism*

Middleware

CORBA, (D)COM, *Python*, ...

Goals

We are looking for a framework which

- allows the decomposition of software in manageable modules.
  $\Rightarrow$ *components*
- allows a clear functional specification of a module.
  $\Rightarrow$ *interfaces*
- forces the developer to use the specification of a module.
  $\Rightarrow$ *data hiding*
- allows replacement of a module with a minimum of effort.
  $\Rightarrow$ *3rd party products*
- allows easy extension of the software by new modules.
  $\Rightarrow$ *plug-in mechanism*

Middleware

CORBA, (D)COM, *Python*, ...

Idea

```
void * module;
string name;

Methods
```

Idea

```
void * module;
string name;

Methods
```

Idea

```
void * module;
string name;
```

Methods

Idea

```
void * module;
string name;
```

Methods

Idea

```
void * module;
string name;
```

Methods

Representative

Idea

Components

with

Interfaces

and a

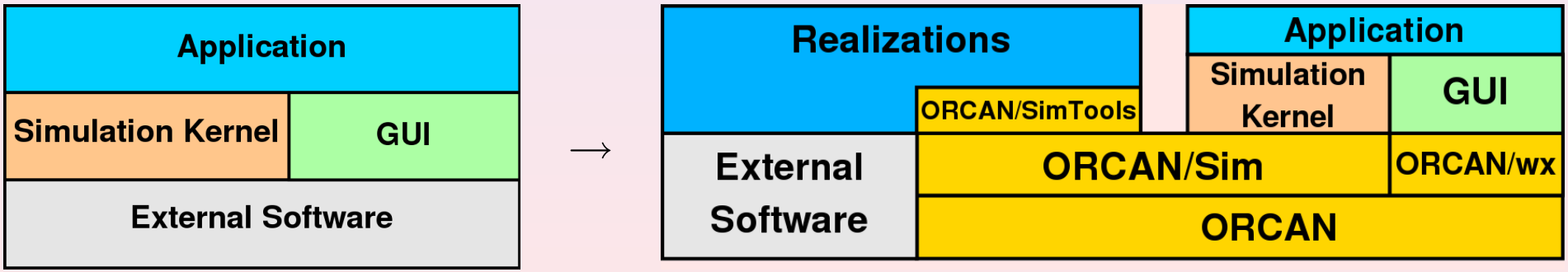
PropertyMap

Outline

What is ORCAN?

Traits

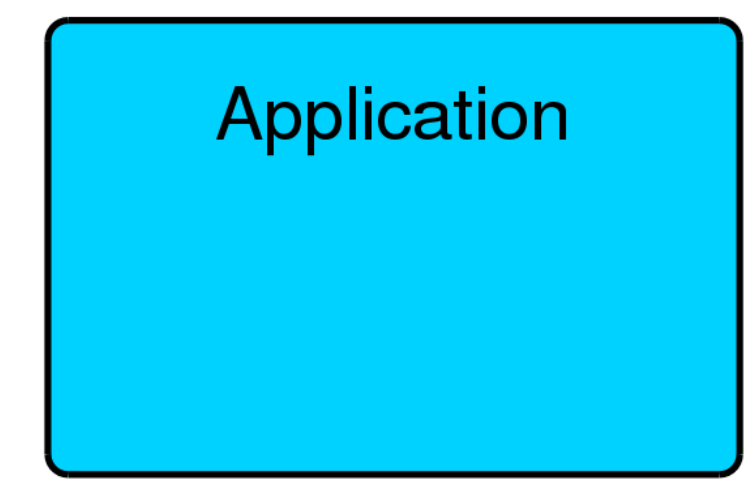
- Collection of C++ libraries.
- Self contained.
- Middleware: Service between application modules.
- Decomposition of software complexity.

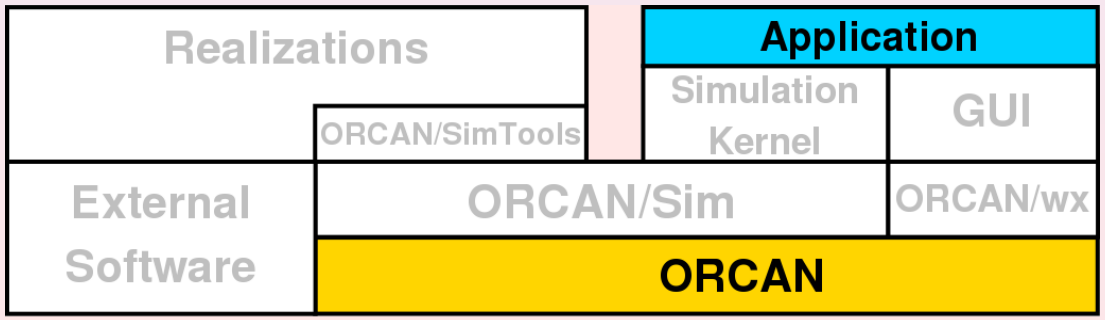


ORCAN

Specification

- Component model (Placeholder for implementation)
- Interfaces
- Load implementation on demand

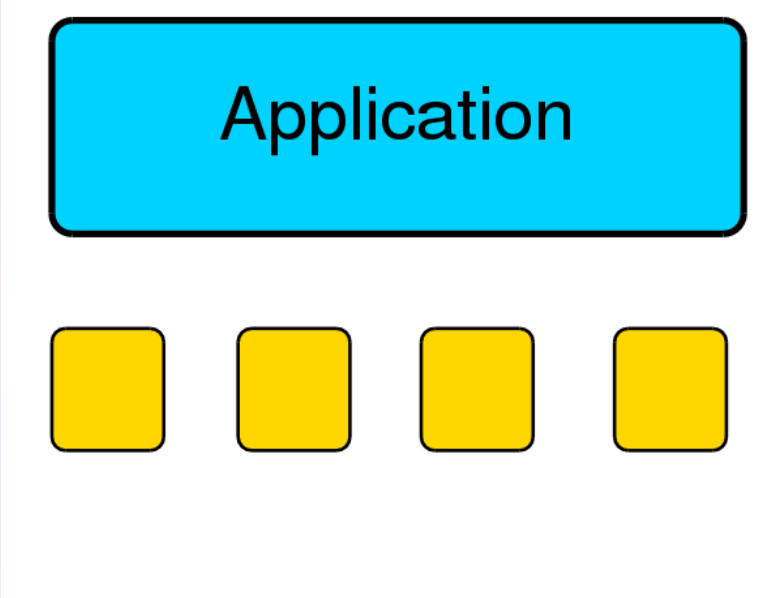


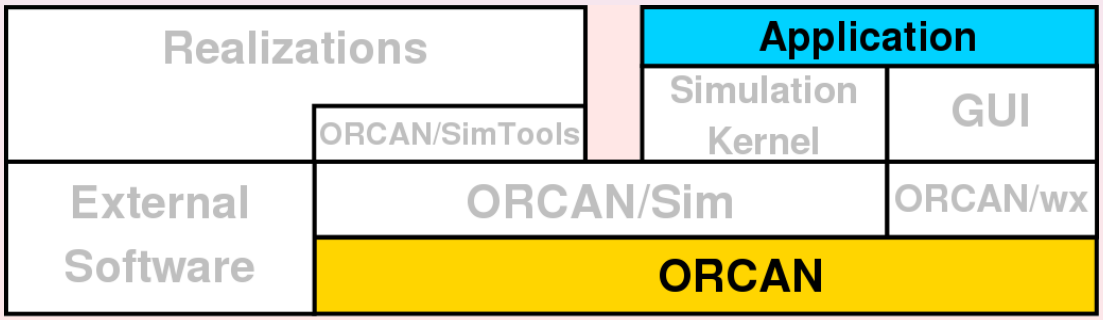


ORCAN

Specification

- Component model (Placeholder for implementation)
- Interfaces
- Load implementation on demand

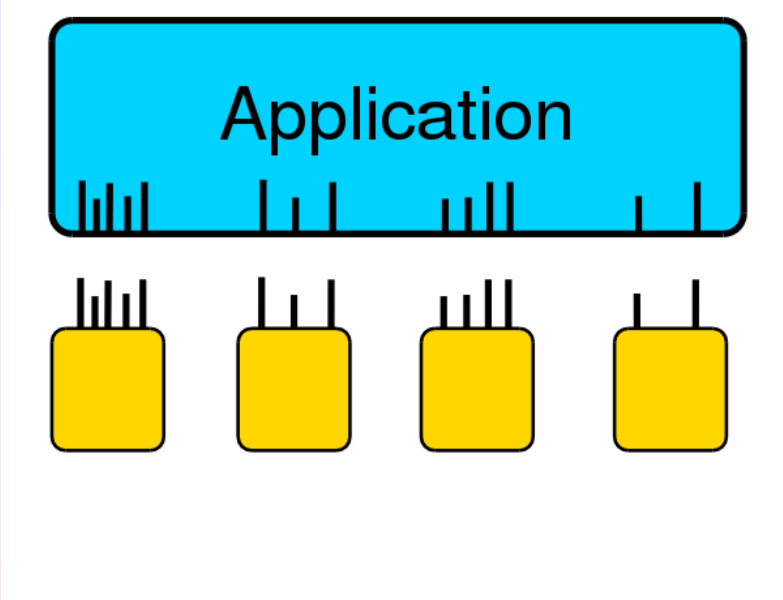


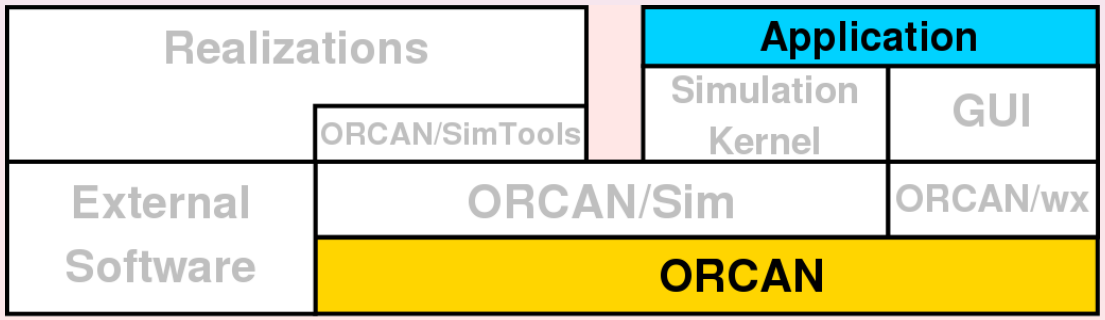


ORCAN

Specification

- Component model (Placeholder for implementation)
- Interfaces
- Load implementation on demand

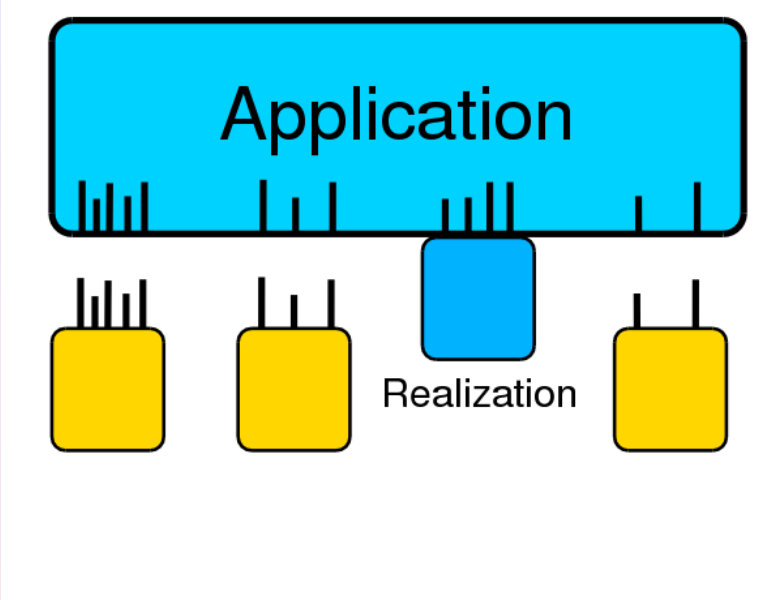


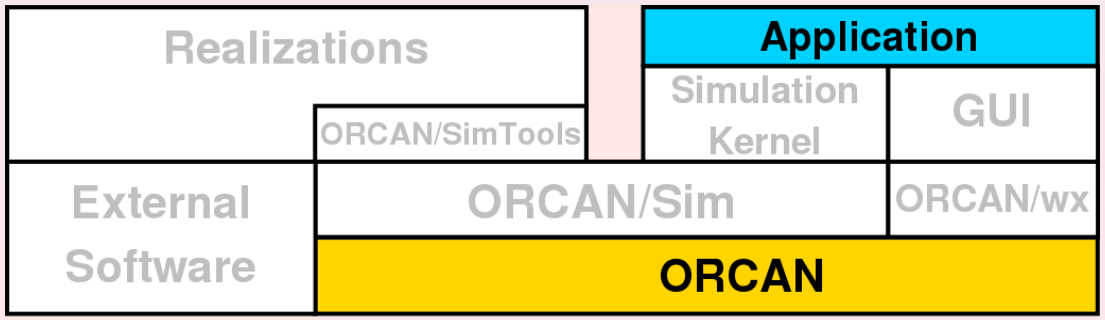


ORCAN

Specification

- Component model (Placeholder for implementation)
- Interfaces
- Load implementation on demand





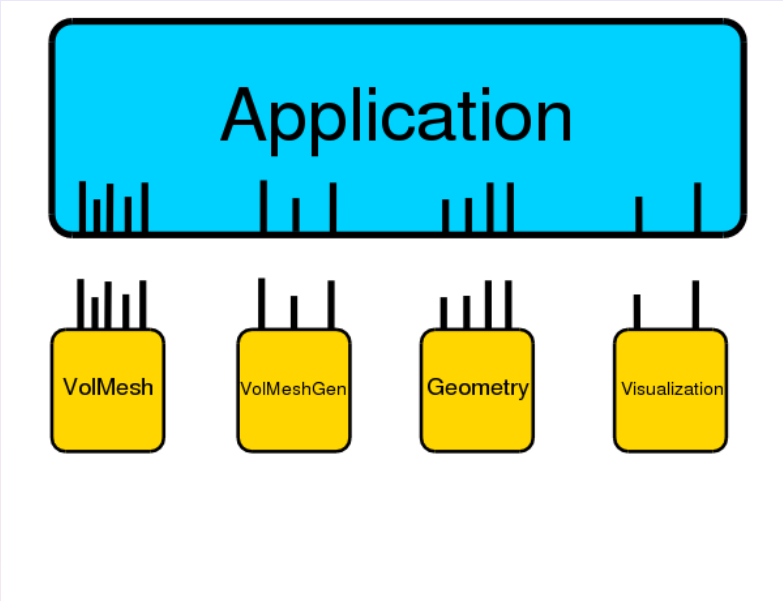
ORCAN/Sim

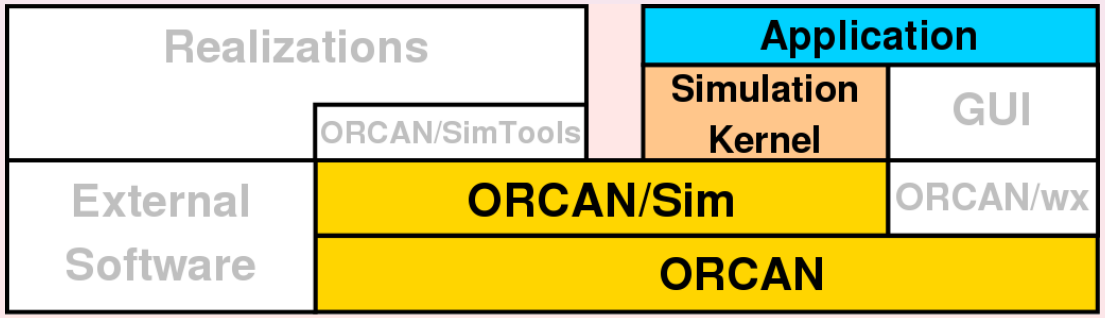
Specification

- Based on ORCAN.
- Simulation related components.

Example

- VolMesh
- VolMeshGen
- Geometry
- Visualization





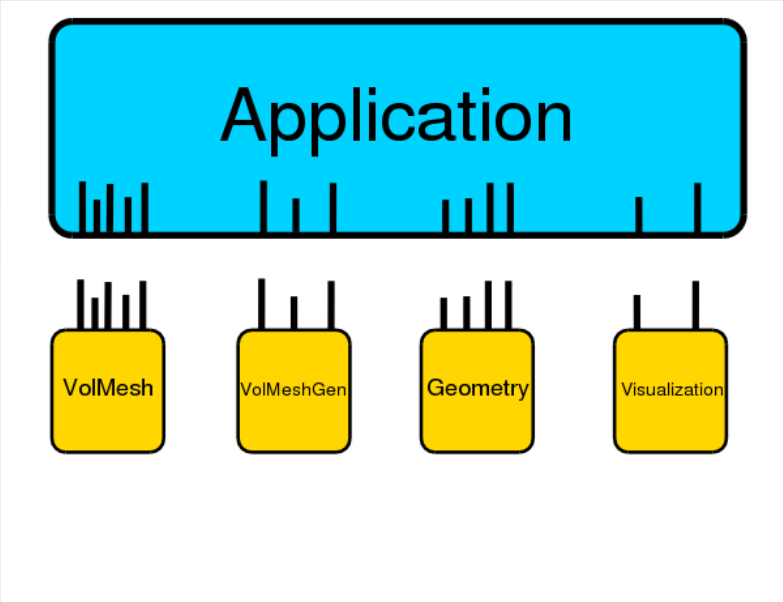
ORCAN/Sim

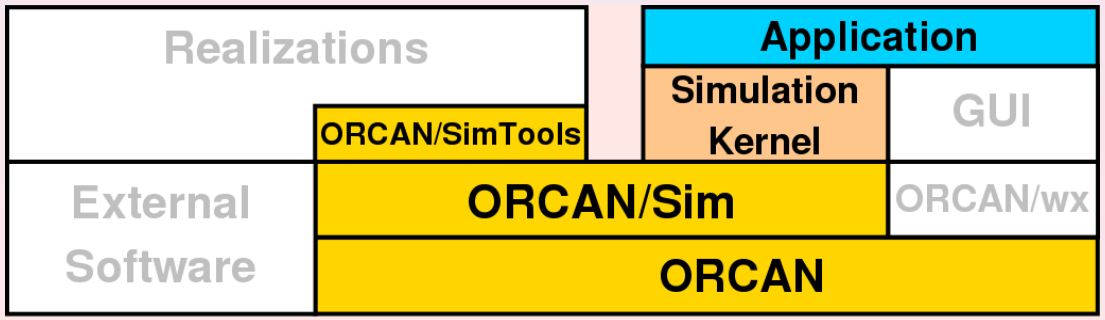
Specification

- Based on ORCAN.
- Simulation related components.

ORCAN/SimTools

- Based on ORCAN/Sim.
- Collection of useful tools.

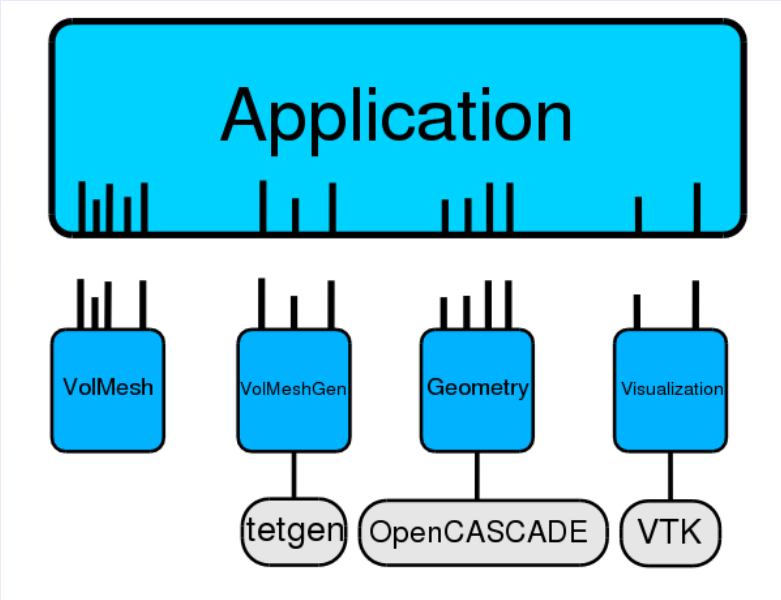


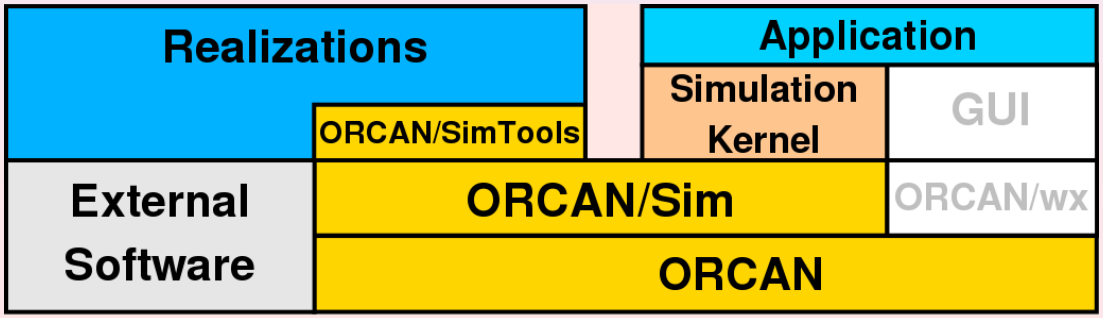


Realizations

Specification

- Based on ORCAN/Sim.
- Implementation of components.
- Not all interfaces necessary.
- May use external software.
- Loaded on demand.
- Exchangeable, even at runtime.

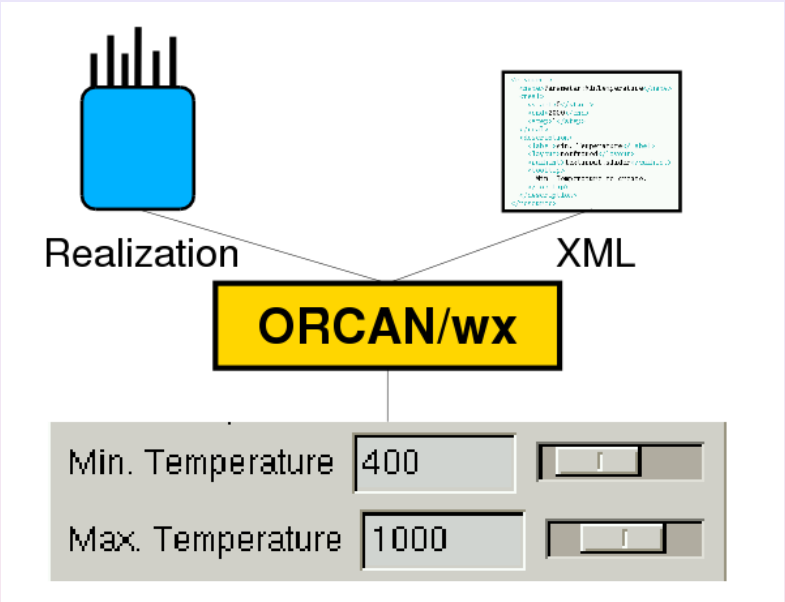


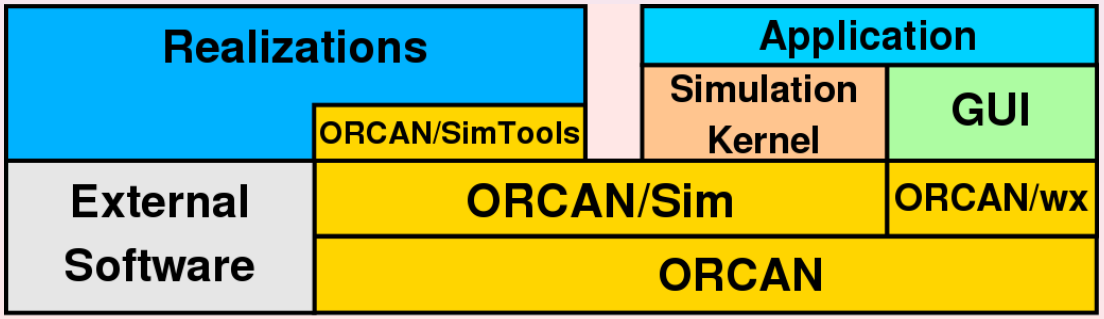


ORCAN/wx

Specification

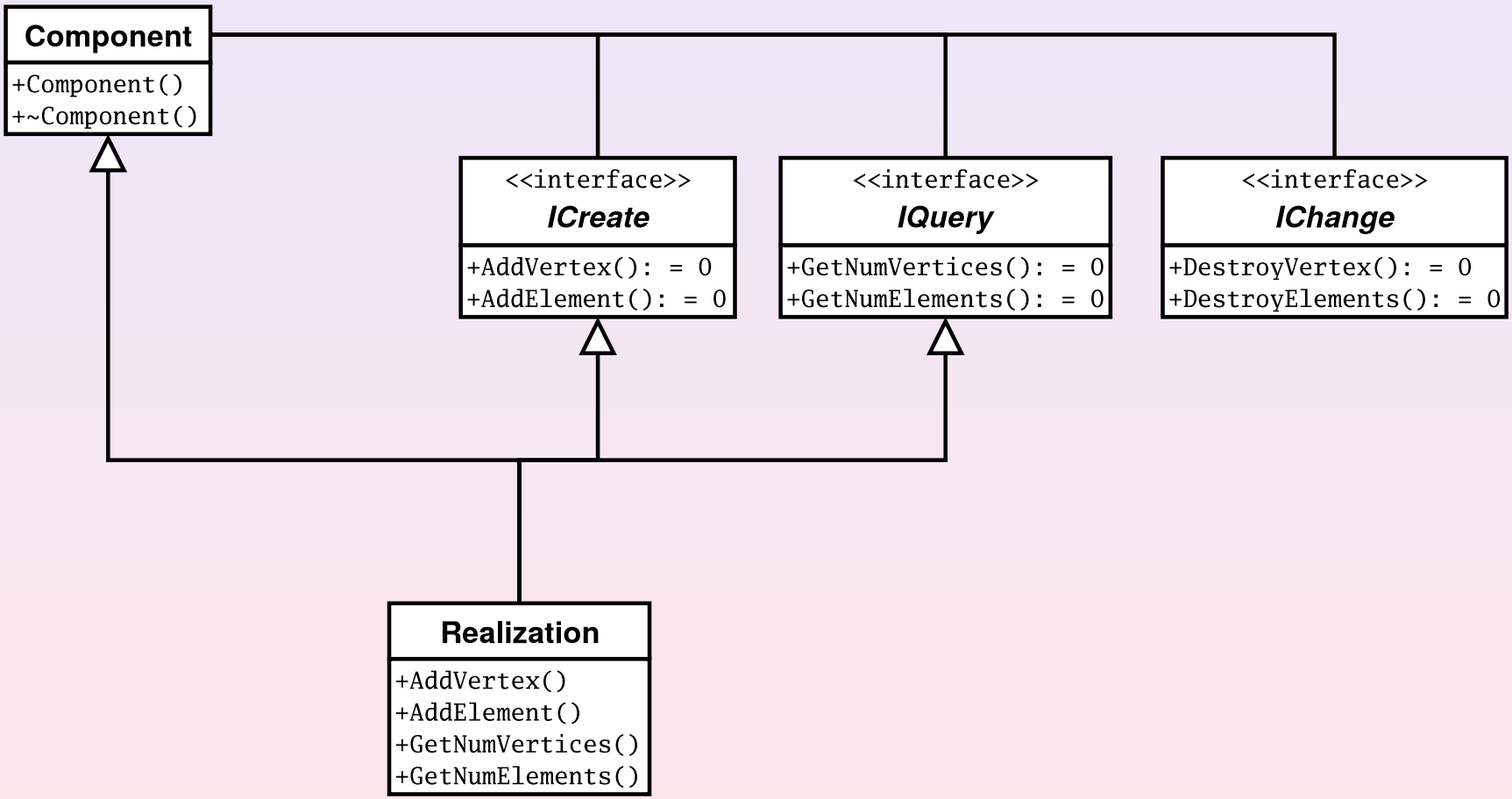
- Based on ORCAN.
- Automatic GUI generation.
- Display parameters of realization.
- XML description of GUI.
- Based on *wxWidgets*.





Outline

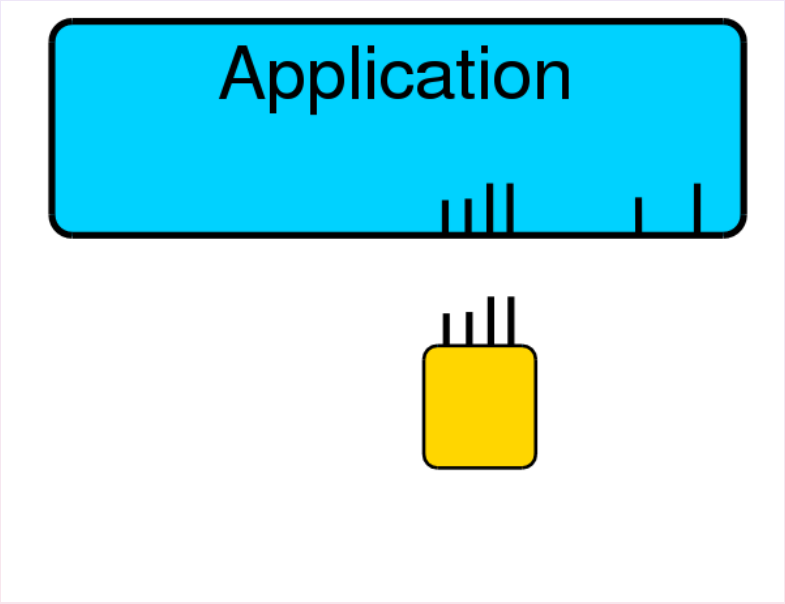
Component, Interfaces, Realization



Object Creation Process

ObjectBroker

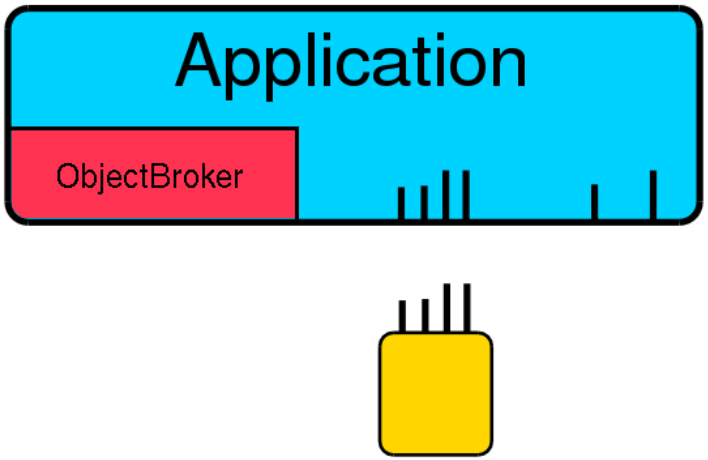
- Application needs new instance. `VolMesh::New()`
- Calls *ObjectBroker* internally.
- Load appropriated library file.
- Instantiates realization.
- Returns reference to application. `VolMeshRef`



Object Creation Process

ObjectBroker

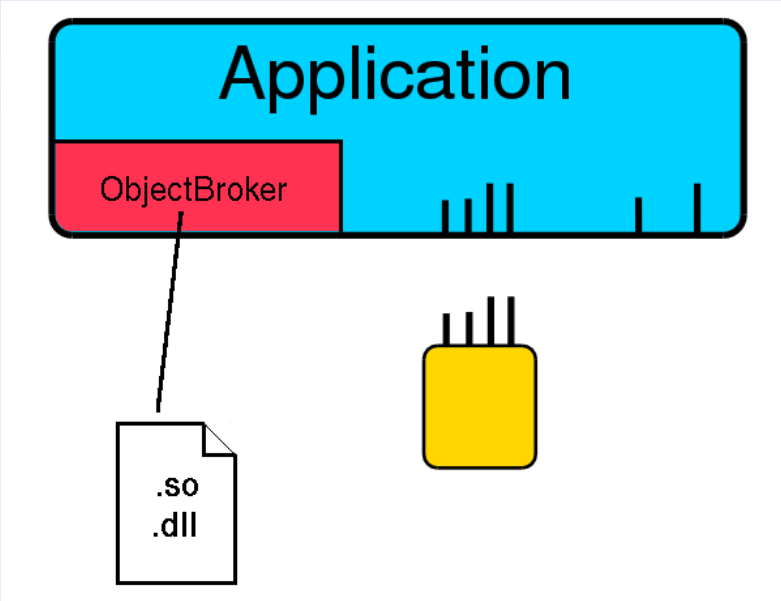
- Application needs new instance. `VolMesh::New()`
- Calls *ObjectBroker* internally.
- Load appropriated library file.
- Instantiates realization.
- Returns reference to application. `VolMeshRef`



Object Creation Process

ObjectBroker

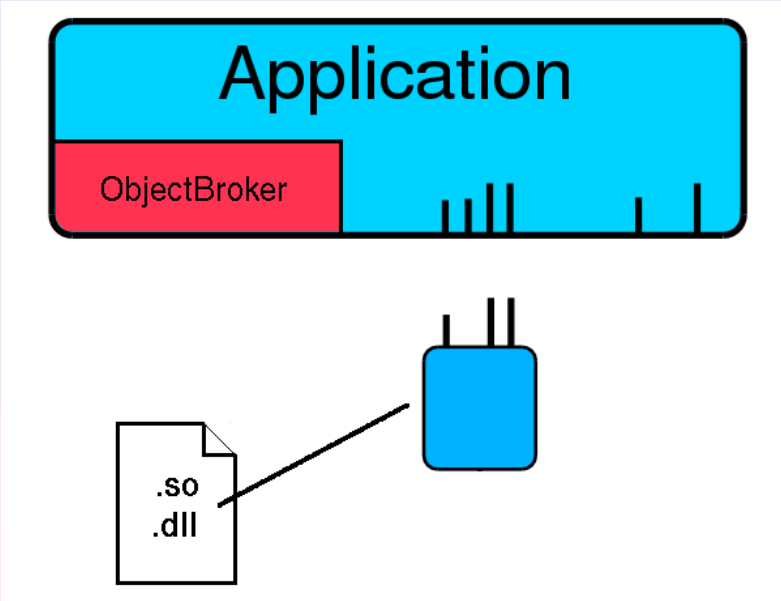
- Application needs new instance. `VolMesh::New()`
- Calls *ObjectBroker* internally.
- Load appropriated library file.
- Instantiates realization.
- Returns reference to application. `VolMeshRef`



Object Creation Process

ObjectBroker

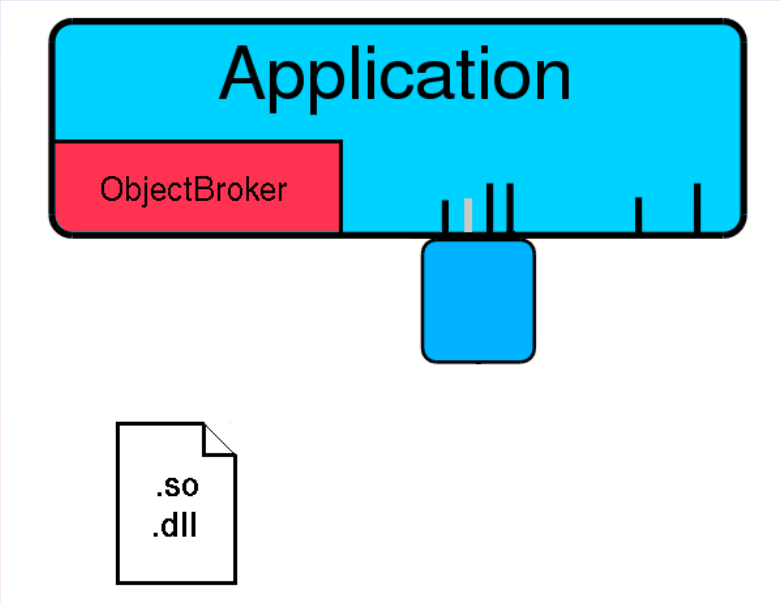
- Application needs new instance. `VolMesh::New()`
- Calls *ObjectBroker* internally.
- Load appropriated library file.
- Instantiates realization.
- Returns reference to application. `VolMeshRef`



Object Creation Process

ObjectBroker

- Application needs new instance. `VolMesh::New()`
- Calls *ObjectBroker* internally.
- Load appropriated library file.
- Instantiates realization.
- Returns reference to application. `VolMeshRef`



Object Reference

- Access point to realization.
- Query and usage of component interfaces.
- Destroy of realization.

Example (Query interface)

Component with query interface IQuery:

```
VolMeshRef ref = VolMesh::New();

if( ref && ref.I.QueryPtr ) {
  int num = ref.I.QueryPtr->GetNumVertices();
}

ref.Delete();
```

Reflectivity

Problem

Realizations of the same component may have different parameters.

Example (VolMeshGen)

- Gmsh
```
float characteristic_length;
float max_element_size;
```
- Tetgen
```
float global_max_volume;
float min_angle;
```

Reflectivity support

Ability to query a component for its intrinsic parameters at runtime.

Reflectivity

Problem

Realizations of the same component may have different parameters.

Example (VolMeshGen)

- Gmsh
```
float characteristic_length;
float max_element_size;
```
- Tetgen
```
float global_max_volume;
float min_angle;
```

Reflectivity support

Ability to query a component for its intrinsic parameters at runtime.

PropertyMap

A parameter of a realization is encapsulated in a property.

A property consists of a key/value pair:

string <name>, *type* <value>

Example

“MinAngle”	*(float)* 15.3
“MaxVolume”	*(float)* 0.12
“SubdivideBdry”	*(bool)* true

- All properties are stored in a map: **PropertyMap**
- The property map can be queried by
  `HasProperty()`
  `GetProperty()`

ORCAN/Sim

Components for Simulation