3rd EUMMC Workshop

28th-29th May 2015 – Jyäskylä, Finland

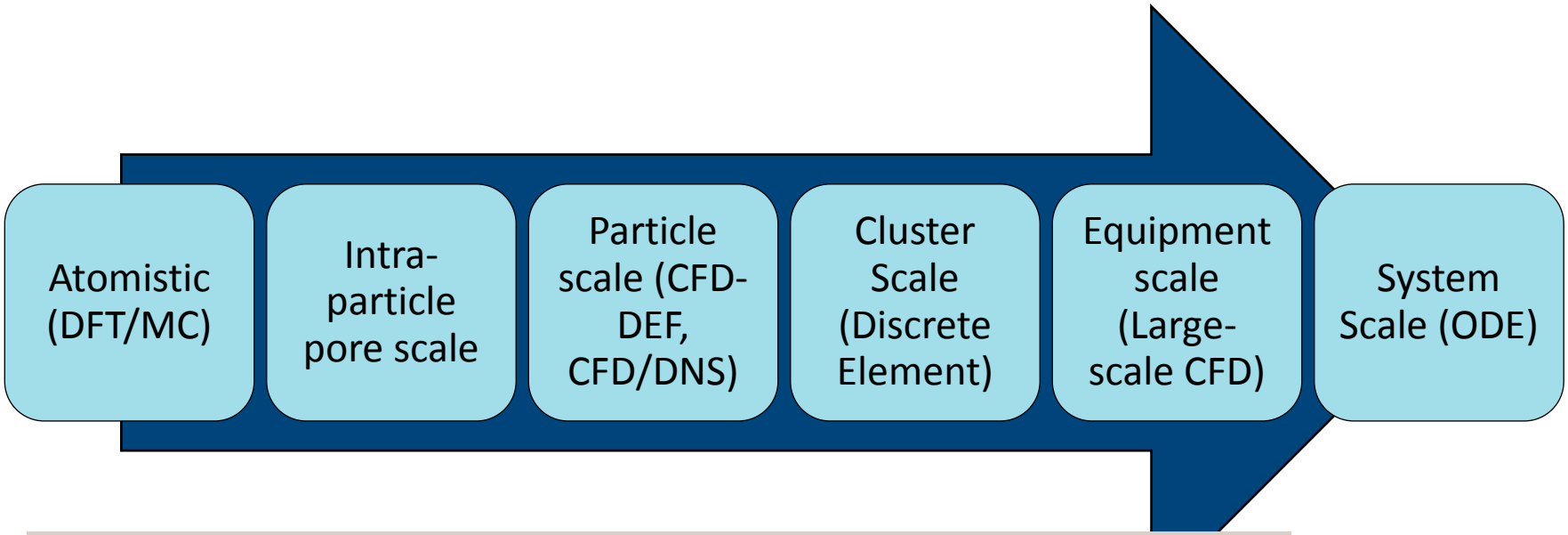
Porto: A framework for information interchange and multi-scale fluid mechanics simulations

T. Hagelien, S. Radl, C. Kloss, C. Goniva, P.I. Dahl, S. M. Nazir, P.Fede, S. Amini

Contents

- Our Multi-Scale Simulation Strategy
- Perspectives on Data, Quality and Connectivity
- Data Centric Architecture and Simulation Framework
- Meta-Data and Structures
- Example - Reading from arbitrary data sources
- Demo – Live Porto-webserver dynamically generating presentation of queries from mongodb using Porto.MVC
- Summary/Conclusions

Multi-Scale Simulation Strategy



- Offline coupling
- Resolve representative elements of the simulation domain
- Feed results into a sub-grid mode on the next larger scale

Software Simulators

In-House

From an Industrial Perspective

Industrial Scale Simulations inherently demand large simulation domains

- To be able to simulate micro scale effects in a large scale environment
- Lowest level of modelling produces highest amount of data
- Low level of modelling is technically infeasible due to computation time

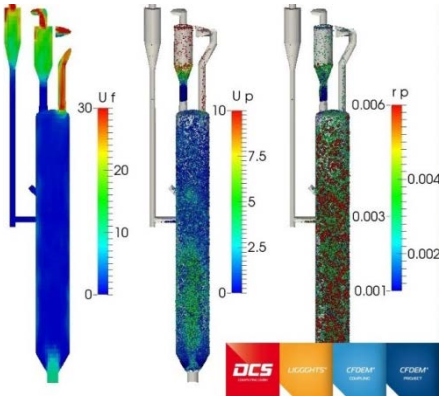
Quality and degree of applied modelling essentially influences the accuracy of the simulation results

- In the past industrial problems were often reduced to a simple cold flow analysis
- Usage of detailed models enables consideration of multi-physics reaction
- Customer's demand implies usage of widely accepted and internationally approved models

Data exchange

- Currently, the majority of applications with significant data exchange have only R&D relevance
- For industrial applications an efficient exchange between platforms is **highly** requested.

From a modelling perspective (CFD-DEM)



Data amount

- ~ 100MB-1GB particle/fluid data per time-step
- ~ 100GB-1TB (O(10e6) particles, CFD cells)

Quality of data

- High temporal and spatial resolution required because of complex physics (physical time-scales (1e-3 sec))

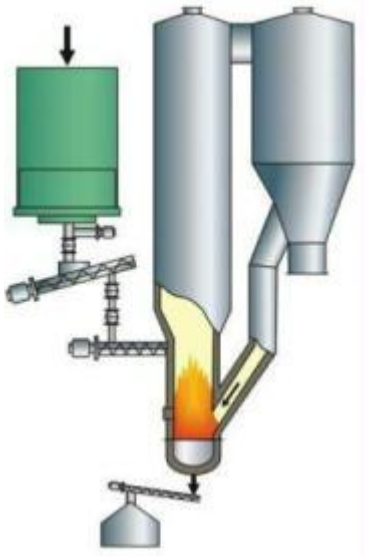
Data exchange between different scales

- Most of particle and fluid data exchanged frequently (exchange time-step (O(1e-3 sec)).
- Needs tight on-line coupling, independent load balancing and efficient, scalable communication schemes. (many-to-many MPI)

Highly-resolved periodic CFB simulation perspective

Data Amount

- For Euler-Euler monodisperse numerical simulation, a “mesh converged” solution requires at least 17 millions of cells. For such a case, the mesh file is about 1Go and for each time the dropped solution (9 variables) files is about 600 Mo. In case of polydisperse or reactive flow the mesh size for “mesh converged” solution is bigger.



From a modelling perspective (CFD-DEM) (continued)

Solutions

Intelligent data management

Separate post-processing data from data needed to restart simulations, stored w/different frequencies

Multi-Scale Modelling

Model selected parts of physics on a continuum level (Finite Volume or Finite Element), either with offline and online coupling.

Need tool-chain for interoperability between simulation packages (Porto)

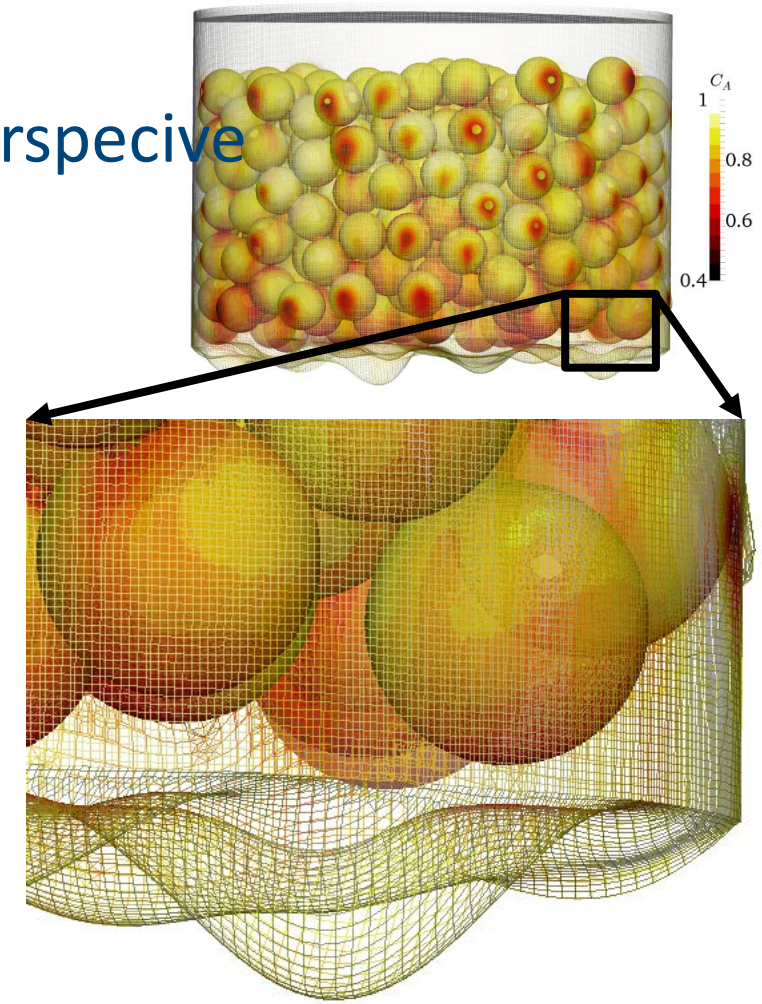
From a particle scale simulator perspecive

Data amount

- Detailed flow simulations require high spatial resolution (> 10 mio. Grid points)
- Unrealistic to save all data (O(100) Gb per simulation, and run parameter variations consisting of O(100) individual simulation runs.
- Standard tools can extract data on the fly – however, they cannot record spatially-filtered statistics (of key relevance for model building i.e. "scale bridging").

Data exchange between scales

- Key issue: standardized input/output format for simulators



Concentration field in a typical heterogeneous reactor (the lines illustrate the required mesh resolution, ©TU Graz, 2015).

Particle scale simulator perspective solution strategy

Data amount

- Established the tool "CPPPO" for on-the-fly **parallel** spatial data filtering (e.g. flow velocity, concentration, reaction rates)
- Monitoring of data statistics during the simulation run is possible

Quality of data

- Correctness of simulator and post processing tool is ensured by an extensive **test harness** hosted by DCS GmbH
- Parallelization enables grid sensitivitiy study

Data exchange between different scales

- JSON-files are widely used
- Simple workflows run in Matlab® / Octave.
- Complex workflows are run in *Porto*

CPPPO is a compilation for the on-the-fly post processing of fluid and particle data, is integrated with OpenFOAM®, and can read CSV or HDF5-style data files.

From a plant simulator perspective

Data exchange

- Commercial software (like Thermoflow), have Excel as a platform to transfer data between different models. Using Excel brings manual intervention to transfer data (which is a challenge)
- Carbon Capture Simulation Initiative (CCSI) group has been actively carrying out research in linking of multi-scale models for CCS processes. NETL, USA developed a modular framework to connect the plant-scale model to reduced order model, optimization module and process synthesis module through Excel interface [1]

[1] Miller, D.C., et al. *A modular framework for the analysis and optimization of power generation systems with CCS*. in *Energy Procedia*. 2011

From the experimentalists perspective

- Data
  - Considerable data related to Chemical-Looping Combustion (CLC) materials, moderate amount related to Reforming (CLR)
  - Limited amount of data concerning successful application of nanoparticles, without coarsening, at elevated temperatures, and at the defined operating conditions for CLR.
- Quality
  - Data need often to be reproduced
  - Testing the same materials, using similar equipment at different locations, is recommended.

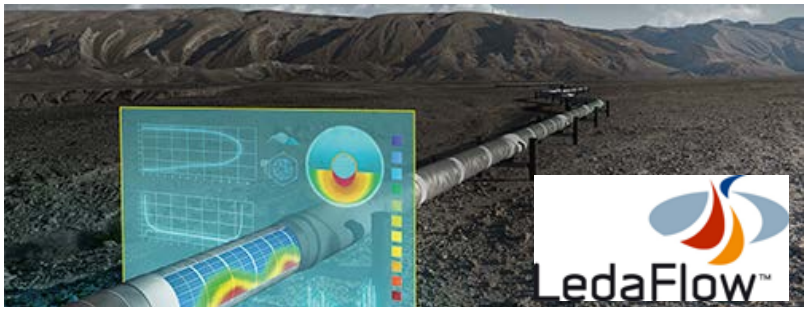
Offline simulator platform perspective

- Utilization of existing data (simulated or experimental)
  - Due to size
  - Formats
  - Quality/Reliability
- Lack of standard schemas and meta-data
- Unavoidable challenge to have to supporting multiple file-formats through readers with a common API.
- Need for access to a scientific data infrastructure – (data warehouses/Hadoop(?)), which supports a rich set of features for search, filtering, data upload, and retrieval.

Some considerations

- Mix of commercial, free/open source and in-house simulators
- Highly heterogeneous simulation environment (geography, hardware, OS)
- The set of simulators may change over time
- Multiple data formats
- Avoid duplication of data
- Completeness of data throughout the simulation workflow
- Allow for analysis/processing of data between workflow steps

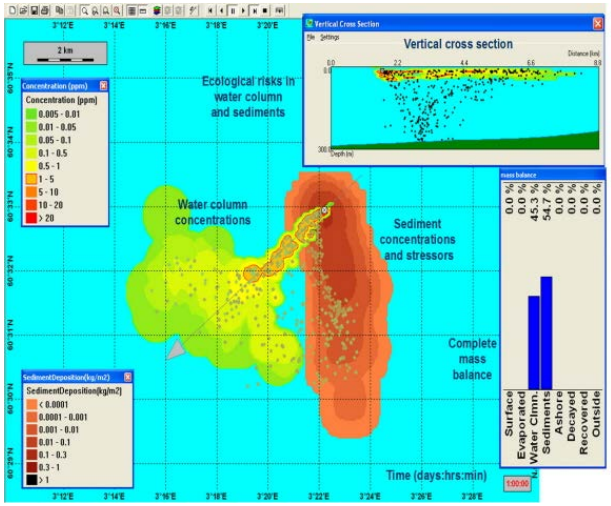
Data Centric Architecture and Simulation Framework



Flow assurance
http://www.kongsberg.com/**ledaflow**

Process control and industrial IT
http://piscada.com/

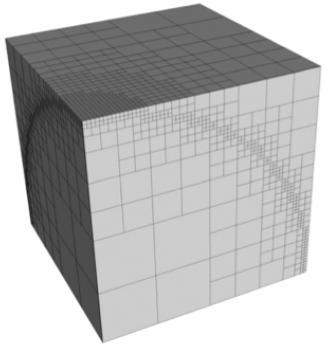
Data Centric Architecture and Simulation Framework



OSCAR – Oil Spill Contingency and Response
http://www.sintef.no/

DREAM - Dose related Risk and Effect Assessment Model
http://www.sintef.no/

Data Centric Architecture and Simulation Framework

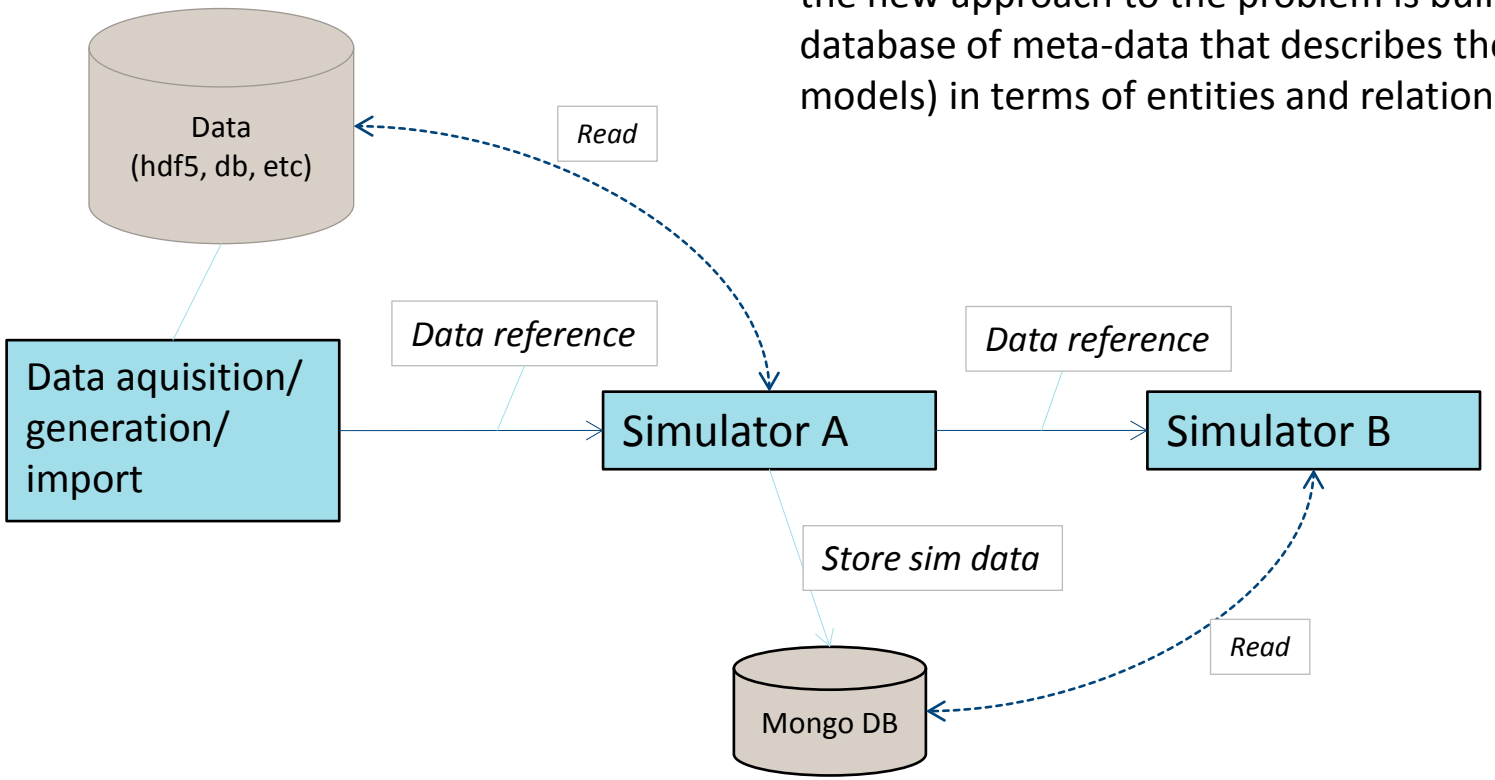


SimCoFlow - A Framework For Complex 3D Multiphase And Multi Physics Flows

Porto

Porto is connecting different simulators and scales. The novelty of the Porto framework lies in the **offline data-centric code-coupling strategy**.

Due to the complexity of (correctly, safely and maintainable) sharing data between multiple in-house and commercial tools (proprietary and open) the new approach to the problem is building a database of meta-data that describes the data (and models) in terms of entities and relationships.



Porto

- Meta-data and data management
- MongoDB database backend
- Configure and run simulator workflows
- Extendable command-line scripting platform based on ECMAScript (QtScript/JavaScript)
- Code generation
  - During development
  - Runtime (Running workflows with automatic code-gen/compile steps)
  - Presentation (Automatic Report Generation/Dynamic Contents)
- New features (beta)
  - HDF5 support (plugin)
  - Integration of the GNU Scientific Library in the scripting environment (early version)
  - Entity translators (automatic input adaptors based on meta-meta)

High Level Tools- and Metadata overview

WP	WP2 COSI	WP3 Atomistic Modelling	WP4 DNS	WP5 Eulerian Modelling	WP6 Phenomenological modelling	WP7 Validation/ Experiments	WP8 Techno Economical Modelling
Tools	• CFDEM • OpenFOAM • LIGGHTS	• REMARC • DFT • SPPARKS	• ParScale • CPPPO	• AnSys Fluent • Neptune CFD	• Phenom	•	• Thermoflow • ASPEN Plus • ASPEN HySys
Data	• LIGGHTS-dump • OpenFOAM Flow Particle	• MD • VASP extend • CHEMKIN-II Data • Surface CHEMIKIN Data • Thermo-Chemistry	• CPPPO Sample • ParScale Sample	• Resolved Flow • Kinetics Input • Reactor Performance	• Mesh • Fluid • Operational • Reactor • Reaction Spec	• Kinetics	• Gas Stream (ASPEN) • GasStream (Thermoflow)

Requirements for metadata

- Formal specification of metadata (critical for sharing!)
- Entities as building blocks for further abstractions
  - External Links
  - Collections of entities (and relationships)
  - Construct and represent metadata semantics from more primive types

"In terms of meta-information, a vector is different from a same-type tuple. For a vector it is implied that vector algebra applies. However, in terms of data storage, a vector and a same-type tuple would be the same."

Dr. Ernst Meese - on the need to be able define a meta-level type-system

Requirements for metadata

- Imperative to separate meta-data from a given storage media
- Complete enough to be able to generate (on the basis of meta-types or instances)
  - Searchable documentation
  - Source code (here are some examples of usage)
    - *Classes/Structs representing internal state*
    - *RPC stubs (Google Protobuf, D-BUS, CORBA IDLs, XML-RPC, ++)*
    - *MPI-dervied data types*
  - Specialized Import/Export file-format converters
- Easy to write/read for humans
- Formal schemas for metadata should be implemented and ***standardized***

Using databases in scientific computing has many benefits

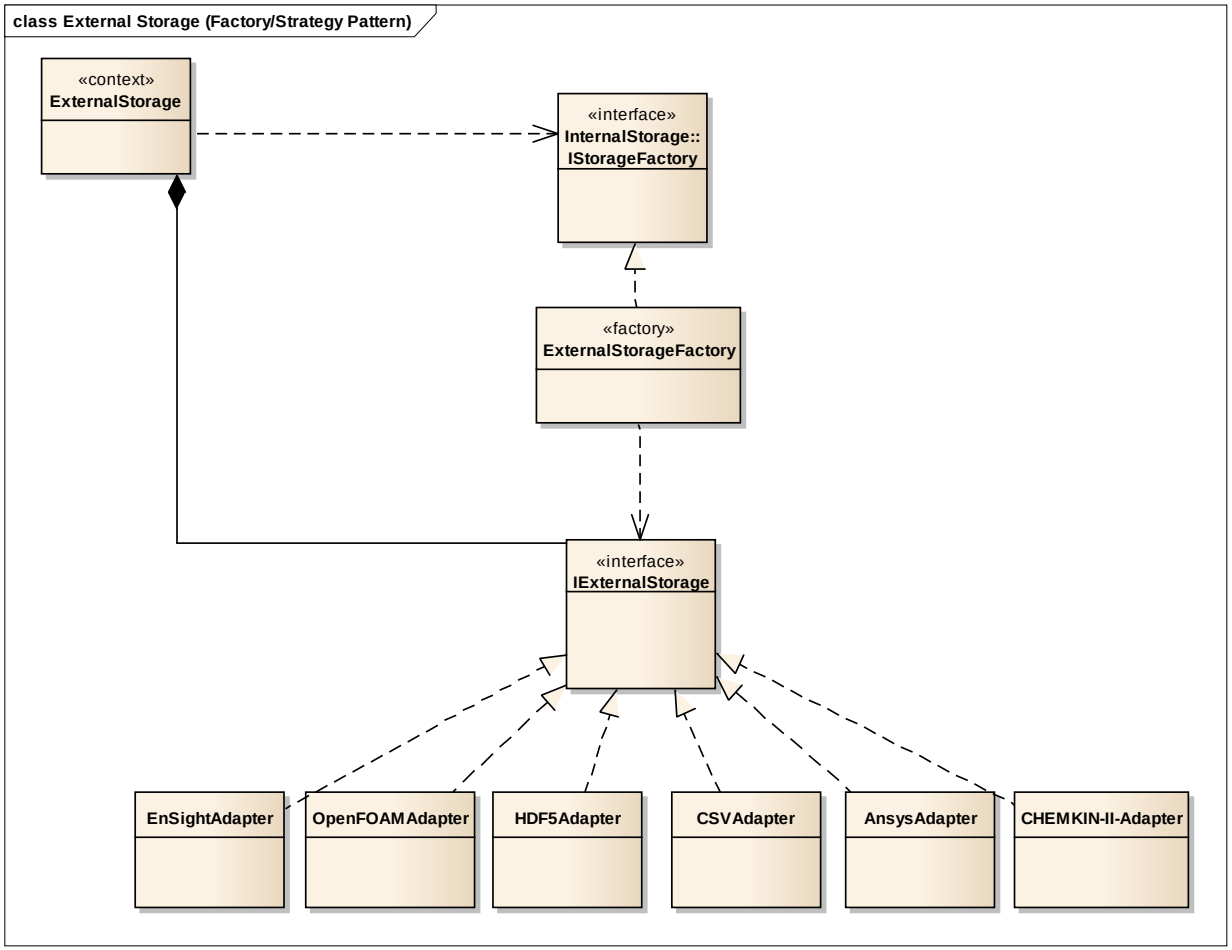
- ✓ Distributed storage
- ✓ Query languages
- ✓ Support transactional data operations
- ✓ Indexed data for fast lookups
- ✓ Concurrent access
- ✓ Data redundancy
- ✓ Scalability
- ✓ Role-Based Access Control
- ✓ Support TLS/SSL encrypted communication between server and clients

Brief about MongoDB

- Currently the World leading NoSQL database

  4th. most popular database (http://db-engines.com/ April/May 2015)
  (only beaten by Oracle and MySQL, Microsoft SQL Server)
- Document Database (not tables identified with keys)
  - The documents are JSON – which maps nicely to programming language data types
- High Performance
  - Fast reads and writes
  - Powerful indexing
- Features that can potentially be utilized:
  - Large data volume aggregation through *Map-Reduce*
  - Storage of very large data sets through *sharding*
  - Connect to a Hadoop Common Scientific Data Warehouse

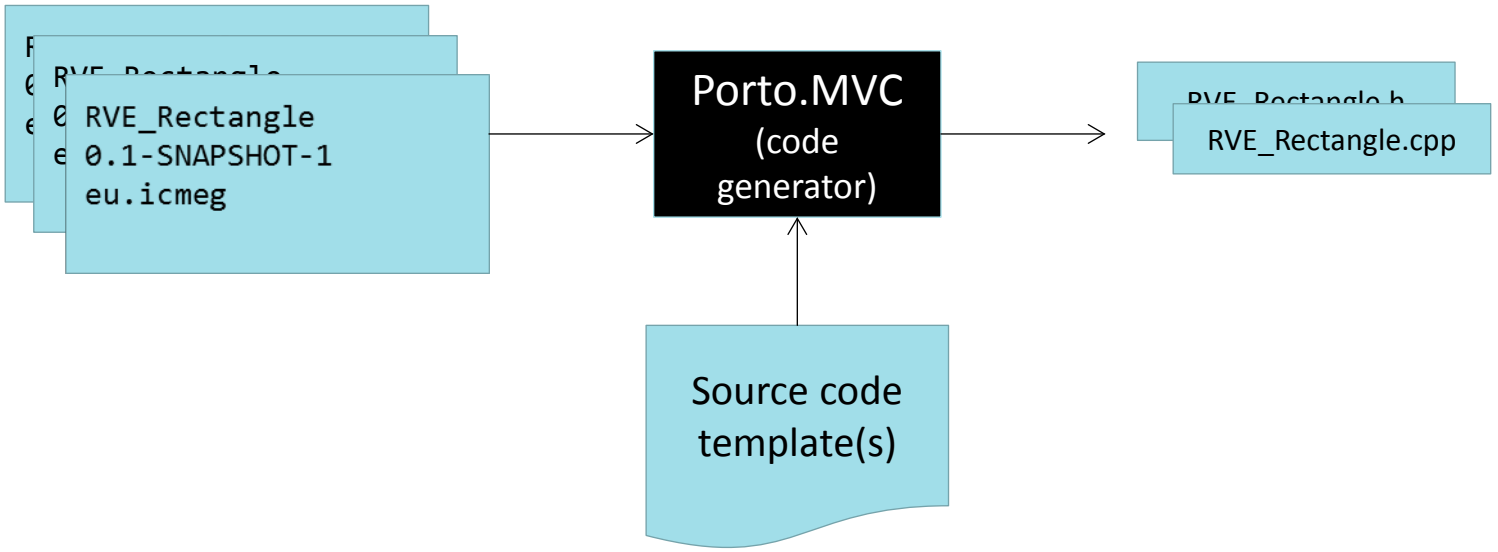
Read Data from an Arbitrary Source using Meta-Data



Define the entity

```
{
    "name": "RVE_Rectangle",
    "version": "0.1-SNAPSHOT-1",
    "namespace": "eu.icmeg",
    "description": "For demonstrational purposes",
    "properties": [
        {
            "name": "position",
            "type": "double",
            "aggregated-type": "vector",
            "unit": "m",
            "description": "Position of the origin of the RVE with respect to a global frame of reference"
        },
        {
            "name": "orientation",
            "type": "double",
            "aggregated-type": "quaternion",
            "description": "Euler angles / quaternions w.r.t global frame of reference"
        },
        {
            "name": "volume",
            "type": "double",
            "aggregated-type": "vector",
            "unit": "mm3",
            "description": "Length in x,y,z axes"
        },
        {
            "name": "NumberCells_x",
            "type": "int",
            "description": "Number of grid cells in x direction"
        },
        {
            "name": "NumberCells_y",
            "type": "int",
            "description": "Number of grid cells in y direction"
```

Generate code for representing the Entity as a class



Generated Code

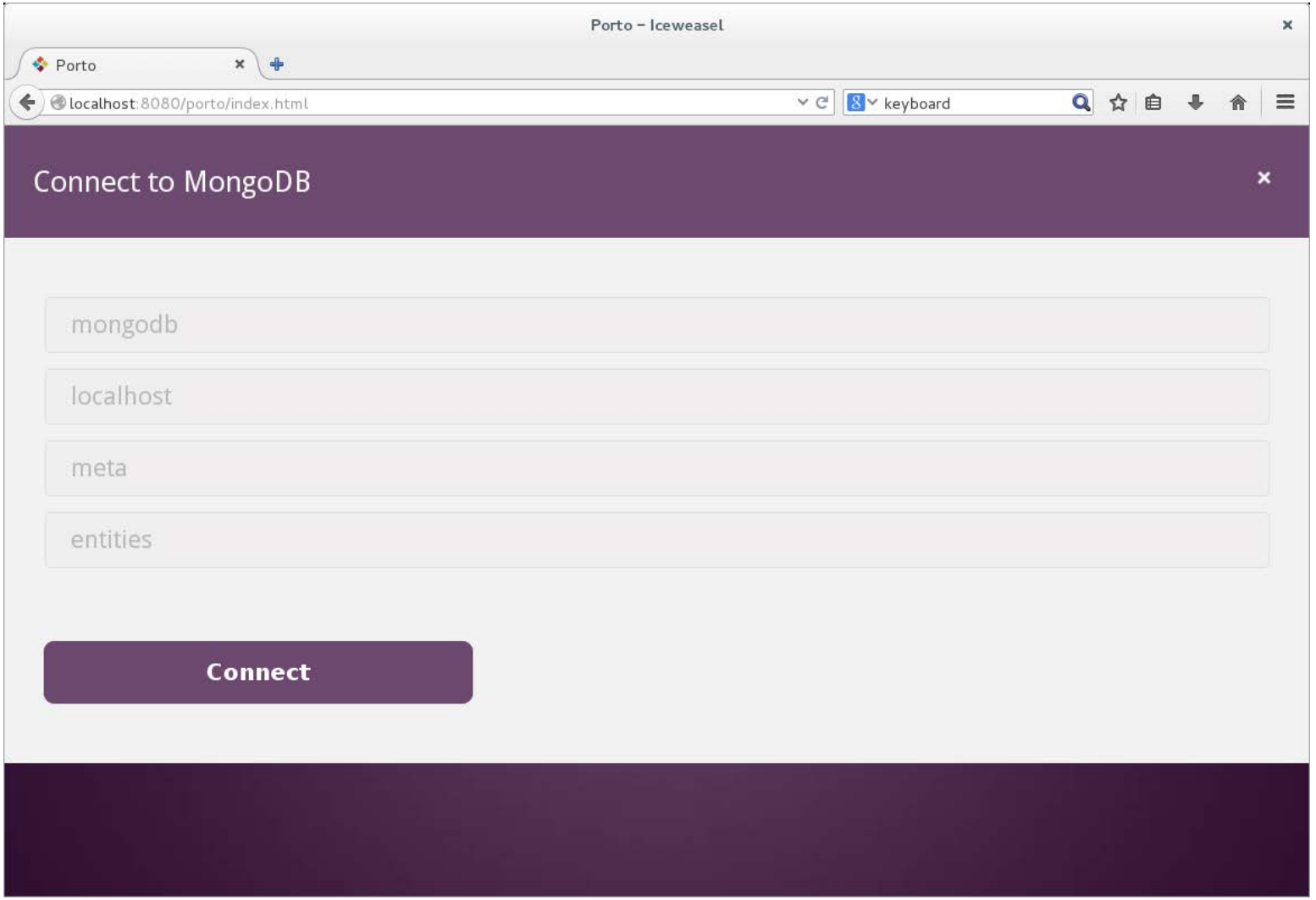
```
// WARNING: This class is autogenerated, please do not edit
#pragma once

#include <ientity.h>
#include <Vector>
#include <Quaternion>

/*! \class RVE_Rectangle
    \brief For demonstrational purposes
 */
namespace eu {
  namespace icmeg {
    class RVE_Rectangle : public porto::IEntity {
    public:
      RVE_Rectangle();
      explicit RVE_Rectangle(const porto::IEntity *entity);
      explicit RVE_Rectangle(const char *id);
      virtual ~RVE_Rectangle();

      static porto::IEntity*   create          (const char *id);
      void                     store           (porto::IDataModel *dataModel)  const;
      void                     load            (porto::IDataModel *dataModel);
      const char *             metaType        ()                              const;
      static const char *      staticMetaType;

      Vector<double> position;            /*!< Position of the origin of the RVE with respect to a ….*/
      Quaternion<double> orientation;     /*!< Euler angles / quaternions w.r.t global frame of reference */
      Vector<double> volume;              /*!< Length in x,y,z axes */
      int NumberCells_x;                  /*!< Number of grid cells in x direction */
      int NumberCells_y;                  /*!< Number of grid cells in y direction */
      int NumberCells_z;                  /*!< Number of grid cells in z direction */

      ...
    };
  } // namespace icmeg
} // namespace eu
```

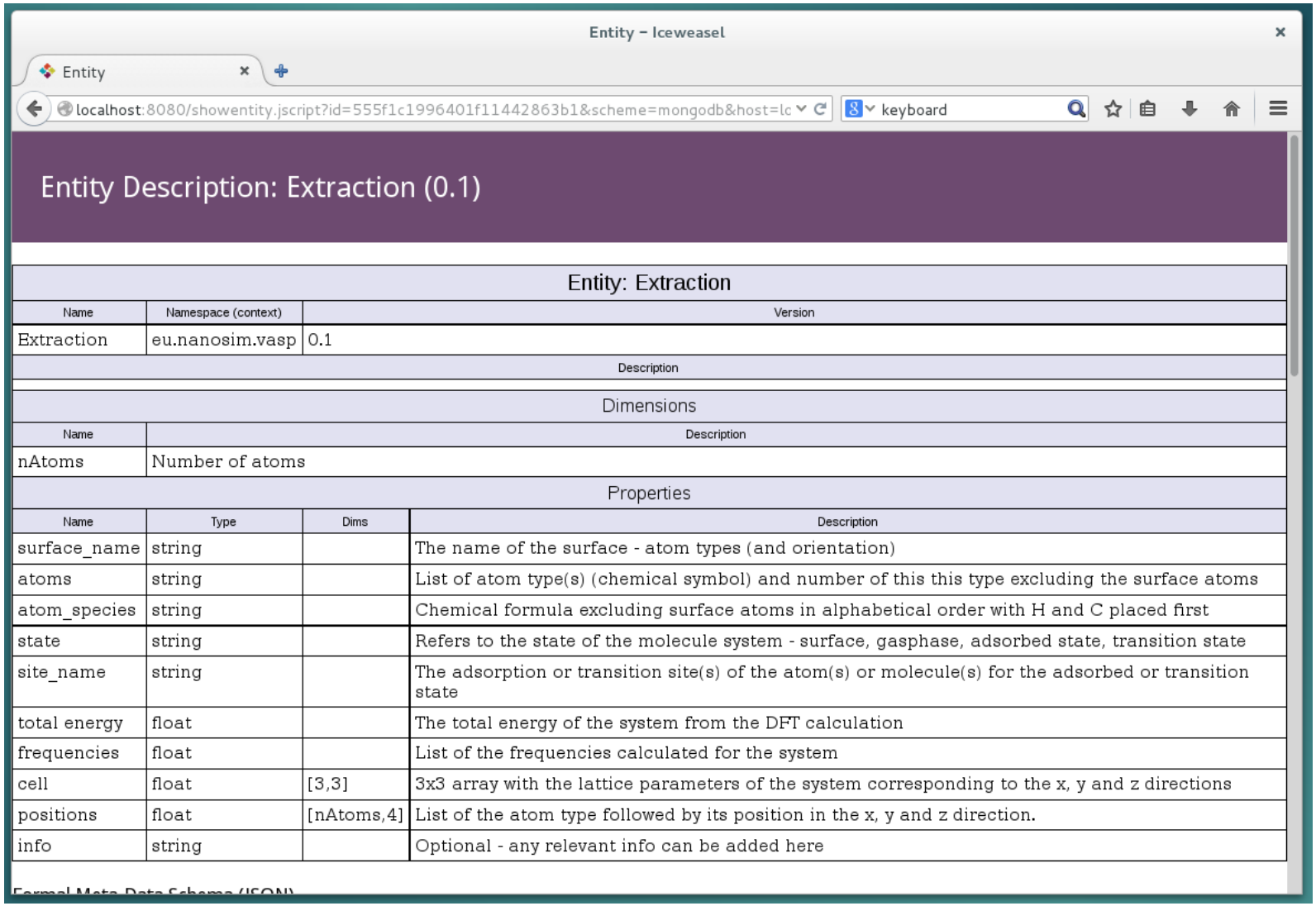
C++ example (external storage)

Client code

```
#include <Porto>
#include "rve_rectangle.h"

using namespace porto::ExternalStorage;

// ..
ExternalStorage storage = ExternalStorage::create()
      .driver("hdf5")
      .uri("file//localhost/path/mycase/data.h5");

auto rveRectangle = new eu::icmeg::RVE_Rectangle();
storage.registerInstance(rveRectangle);
storage->read();
```

Demo


Porto – Iceweasel
Porto
localhost:8080/porto/index.html
keyboard
Connect to MongoDB
mongodb
localhost
meta
entities
Connect



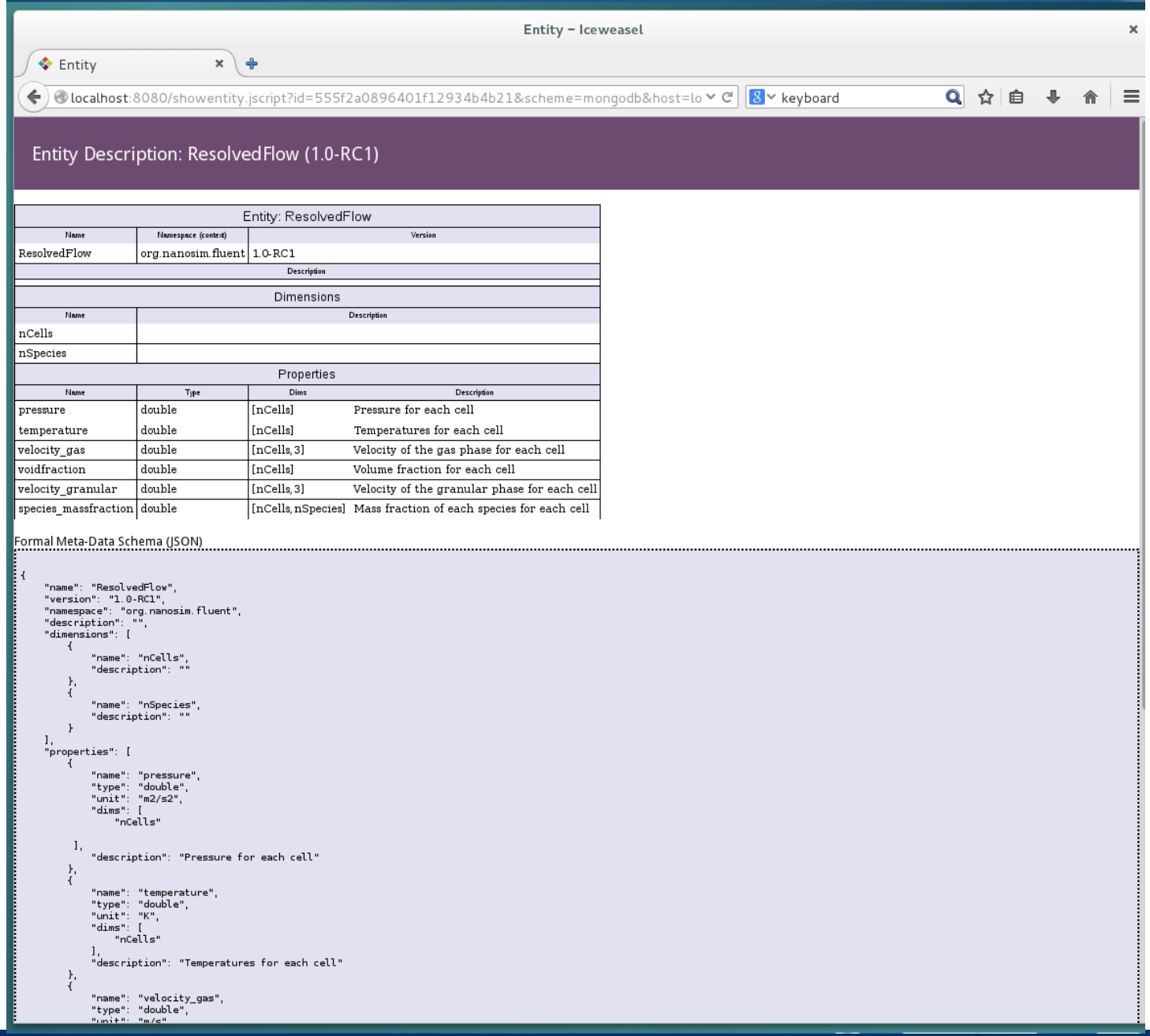
Entity Description: Extraction (0.1)

Entity: Extraction		
Name	Namespace (context)	Version
Extraction	eu.nanosim.vasp	0.1
Description		

Dimensions	
Name	Description
nAtoms	Number of atoms

Properties			
Name	Type	Dims	Description
surface_name	string		The name of the surface - atom types (and orientation)
atoms	string		List of atom type(s) (chemical symbol) and number of this this type excluding the surface atoms
atom_species	string		Chemical formula excluding surface atoms in alphabetical order with H and C placed first
state	string		Refers to the state of the molecule system - surface, gasphase, adsorbed state, transition state
site_name	string		The adsorption or transition site(s) of the atom(s) or molecule(s) for the adsorbed or transition state
total energy	float		The total energy of the system from the DFT calculation
frequencies	float		List of the frequencies calculated for the system
cell	float	[3,3]	3x3 array with the lattice parameters of the system corresponding to the x, y and z directions
positions	float	[nAtoms,4]	List of the atom type followed by its position in the x, y and z direction.
info	string		Optional - any relevant info can be added here

Formal Meta-Data Schema (JSON)



Entity Description: ResolvedFlow (1.0-RC1)

Entity: ResolvedFlow		
Name	Namespace (context)	Version
ResolvedFlow	org.nanosim.fluent	1.0-RC1
Description		

Dimensions	
Name	Description
nCells	
nSpecies	

Properties			
Name	Type	Dims	Description
pressure	double	[nCells]	Pressure for each cell
temperature	double	[nCells]	Temperatures for each cell
velocity_gas	double	[nCells, 3]	Velocity of the gas phase for each cell
voidfraction	double	[nCells]	Volume fraction for each cell
velocity_granular	double	[nCells, 3]	Velocity of the granular phase for each cell
species_massfraction	double	[nCells, nSpecies]	Mass fraction of each species for each cell

Formal Meta-Data Schema (JSON)


```
{
    "name": "ResolvedFlow",
    "version": "1.0-RC1",
    "namespace": "org.nanosim.fluent",
    "description": "",
    "dimensions": [
        {
            "name": "nCells",
            "description": ""
        },
        {
            "name": "nSpecies",
            "description": ""
        }
    ],
    "properties": [
        {
            "name": "pressure",
            "type": "double",
            "unit": "m2/s2",
            "dims": [
                "nCells"

         ],
            "description": "Pressure for each cell"
        },
        {
            "name": "temperature",
            "type": "double",
            "unit": "K",
            "dims": [
                "nCells"
            ],
            "description": "Temperatures for each cell"
        },
        {
            "name": "velocity_gas",
            "type": "double",
            "unit": "m/s"
```

Conclusion

- Hybrid coupling strategy
  - Large data amounts.
  - Frequent exchange/ iterative -> online coupling
  - Offline coupling
- The role of Porto
- Metadata Requirements
- Applications of formal metadata schemas