Grant Agreement No.: 604656

Project acronym: NanoSim

Project title: A Multiscale Simulation-Based Design Platform for Cost-Effective CO_2 Capture Processes using Nano-Structured Materials (NanoSim)

Funding scheme: Collaborative Project

Thematic Priority: NMP

THEME: [NMP.2013.1.4-1] Development of an integrated multi-scale modelling environment for nanomaterials and systems by design

Starting date of project: 1st of January , 2014

Duration: 48 months

WP N°	Del. N°	Title	Version	Lead beneficiary	Nature	Dissemin. level	Delivery date from Annex I	Actual delivery date dd/mm/yyyy
1	4	NanoSim Meta-data	0.1	SINTEF	Report	PU	01/12/2014	13/03/2015

1 Executive Summary

Multi-scale modeling is the discipline of utilizing multiple models at different scales to describe a system. The data flows through the different scales, where simulations are connected steps of data transformations. The data itself don't have meaning unless it is put into a context and becomes **information**. The sharing and transformation of information requires that there exists some kind of information interchange mechanism. This is one of the key purposes of the common platform *Porto*.

The extra information that tells us what the data represents is often called meta-data (or data describing data). The rationale behind the meta-data schema used in Porto is given in the initial chapter, while the collected meta-data from the different work-packages are given in chapter 3.

The meta-data of NanoSim is also input to the European Materials Modeling Council (EMMC) in context of the Multiscale Cluster/ICMEg consortium for defining a standardized structure for metadata interchange. For this work the section on "Values, data and meta-data" is our suggestions for how the formal schemas should be implemented and standardized, motivated by numerous other projects that are using this very same schema.

2 Values, data and meta-data

A *value* can be defined as a numerical amount denoted by an algebraic term, such as a magnitude, quantity a number or even something complex. Values can also be seen as a something that describes the state of an entity at a point in time. In computing this is often referred to just as **data**. An *entity* is a physical or imaginary "thing" that is logically self-contained and independent. In order to interpret, share and utilize data, we need to apply meaning and context to the data. This is what we often call **information** (Figure 2) In this chapter we will look at how we can define information in terms of *meta-data*. Furthermore, we look at how entities can be built from these primitives, and give an abstract context to a set of properties.

Figure 1 Property

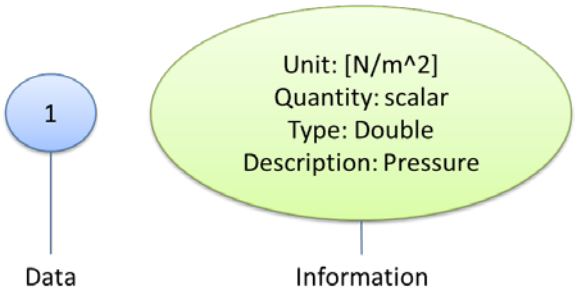

Figure 2 Data vs Information

2.1 From Data to Information

To go from data to information, we need to describe what the data is. In order to do this we define the concept of a **property (**Figure 1**).** A property represents a primitive type (see Table 1 Primitive types) in a data set and gives it meaning.

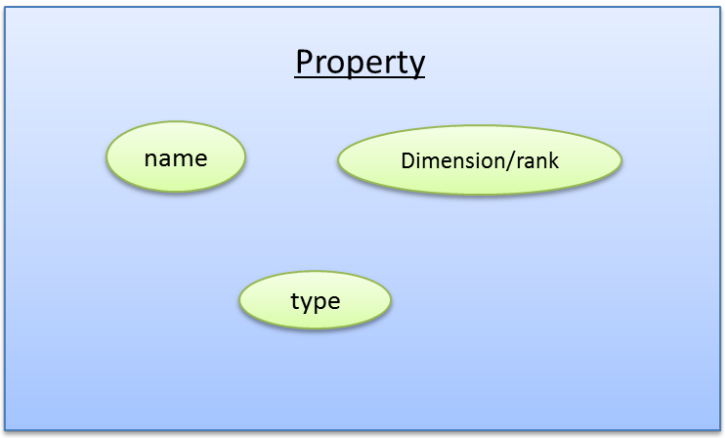

Figure 3 - Property defined with a name, type and dim/rank

Table 1 Primitive types

Property Type	Equvalent C type
Boolean	bool
Character	unsigned char
Integer	int
String	
BLOB	
Int8	int8_t
Int16	int16_t
Int32	int32_t
UInt8	
UInt16	
UInt32	
Float	float
Double	double
LongDouble	long double
enumerated type[1]	

In a computer programming language, a property is often declared as a variable or a constant. In programming languages that supports static typing, a constant or variables is declared as defined *type*. If the property is not a scalar, the dimensionality and sizes is declared. Figure 3 illustrates how a property can be defined to give enough meaning for a computer program to declare an instance of the data.

[1] Enumerated types are locally defined data types consisting of a set of named values (enumerators). These names are to be considered as constant identifiers. For example, an enumerated type to describe a solver type for incompressible flow in OpenFOAM could consist of the enumerators BOUNDARY_FOAM, CHANNEL_FOAM, ICO_FOAM, MRF_SIMPLE_FOAM etc.

This is, however, not enough information to understand what the data really represents. For that we need to add more metadata. In scientific computing the data is worthless without knowing unit of measurement. To extend the information further, a textual description is often useful for humans (See Figure 4). Other attributes of a property can be applied, but these are often dependent on the context in which the property is used, and should therefore be avoided in a standardized generic description of the metadata.

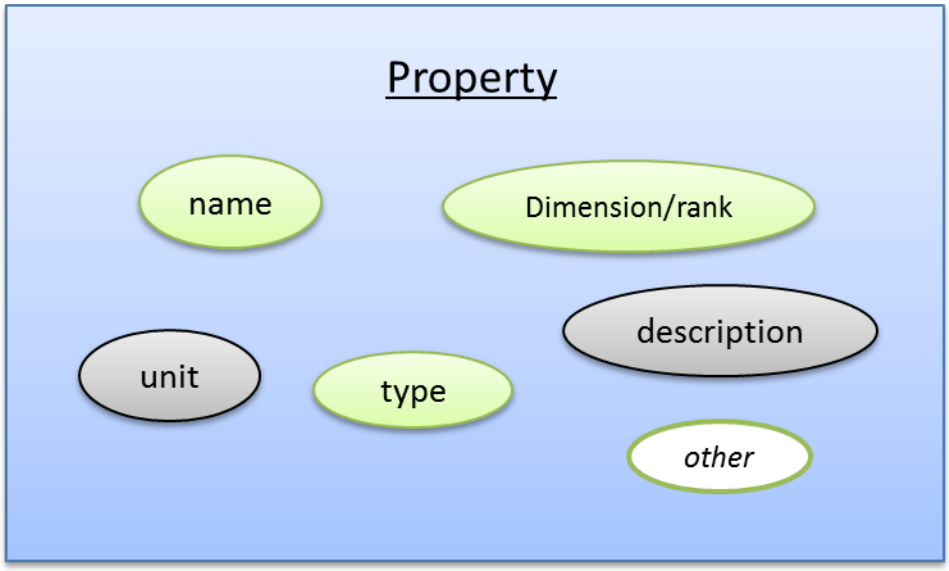

Figure 4 A property defined with context information

2.2 Defining entities

Once we have the lowest level primitives covered we need to put them in a context. An entity iøs an abstract context that is self-contained and independent (of other entities). This allows the entity to be used in any context without causing dependency problems. To be able to do this in practice, we need to define a standardized schema for how the entities are defined. Any software that wants to apply the entity information should therefore comply with this standard. There are numerous options for choosing a language to describe the schema. We have chosen JSON (JavaScript Object Notation - ECMA-404 The JSON Data Interchange Standard.) because of the widespread support of the standard in popular programming languages, its ease of use and readability for both humans and computers.

2.2.1 Contents of the entity

The entity (Figure 5) contains a set of properties. In addition to this it should be uniquely identified in order to avoid conflicting versions of entity definitions. This can be achieved by giving it a name and version number. It might even be necessary apply additional information to ensure uniqueness. A textual description is useful for humans to understand the context and purpose of an entity.

The way we want to define dimensionality in a property is by giving it a named value (for instance NX and NY could be the number of cells in a grid in x and y-direction, and a pressure field could be

defined by a property named 'p' with the dimension ["NX","NY"]. Because these named values can be shared by multiple properties, it makes sense to declare the dimensions as part of the entity.

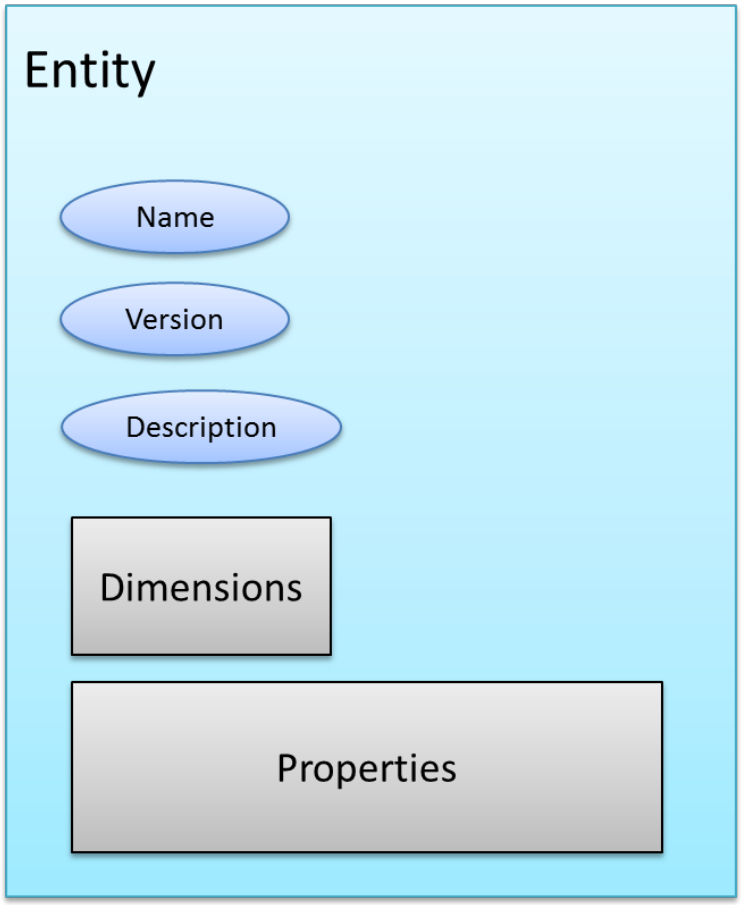

Figure 5 - The Entity

2.2.2 JSON Schema

The JSON Schema standard is still at a draft stage. However, we are able to use this for defining the formal specification of an entity:

Figure 6 JSON Schema - The Entity

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "id": "http://schema.sintef.no",
  "type": "object",
  "properties": {
    "name": {
      "id": "http://schema.sintef.no/name",
      "type": "string"
    },
```

```
"version": {
  "id": "http://schema.sintef.no/version",
  "type": "string"
},
"description": {
  "id": "http://schema.sintef.no/description",
  "type": "string"
},
"enums": {
  "id": "http://schema.sintef.no/enums",
  "type": "array",
  "items": {
    "id": "http://schema.sintef.no/enums/0",
    "type": "object",
    "properties": {
      "name": {
        "id": "http://schema.sintef.no/enums/0/name",
        "type": "string"
      },
      "values": {
        "id": "http://schema.sintef.no/enums/0/values",
        "type": "array",
        "items": {
          "id": "http://schema.sintef.no/enums/0/values/1",
          "type": "string"
        }
      },
      "description": {
        "id": "http://schema.sintef.no/enums/0/description",
        "type": "string"
      }
    }
  }
},
"dimensions": {
  "id": "http://schema.sintef.no/dimensions",
  "type": "array",
  "items": {
    "id": "http://schema.sintef.no/dimensions/0",
```

```
        "type": "object",
        "properties": {
          "name": {
            "id": "http://schema.sintef.no/dimensions/0/name",
            "type": "string"
          },
          "description": {
            "id": "http://schema.sintef.no/dimensions/0/description",
            "type": "string"
          }
        }
      }
    },
    "properties": {
      "id": "http://schema.sintef.no/properties",
      "type": "array",
      "items": {
        "id": "http://schema.sintef.no/properties/1",
        "type": "object",
        "properties": {
          "name": {
            "id": "http://schema.sintef.no/properties/1/name",
            "type": "string"
          }
        }
      }
    }
  },
  "required": [
    "name",
    "version",
    "dimensions",
    "properties"
  ]
}
```

3 NanoSim Meta-data

One ambition of the Porto framework in NanoSim is to collect or access data from all scales regardless of the simulation tool or file formats being used. (Figure 7) This requires two things: 1) The data must be described (in terms of meta-data) to make the information available outside the scope of the original context (or simply, create a common language such that all data can be interpreted). The framework to do this is described in the previous chapter where the anatomy of the entity schema is defined. 2) The data must be accessible by common Data Access Interfaces (Data Access API's). Where and how the data is actually stored is then irrelevant for any application that wants to use the information.

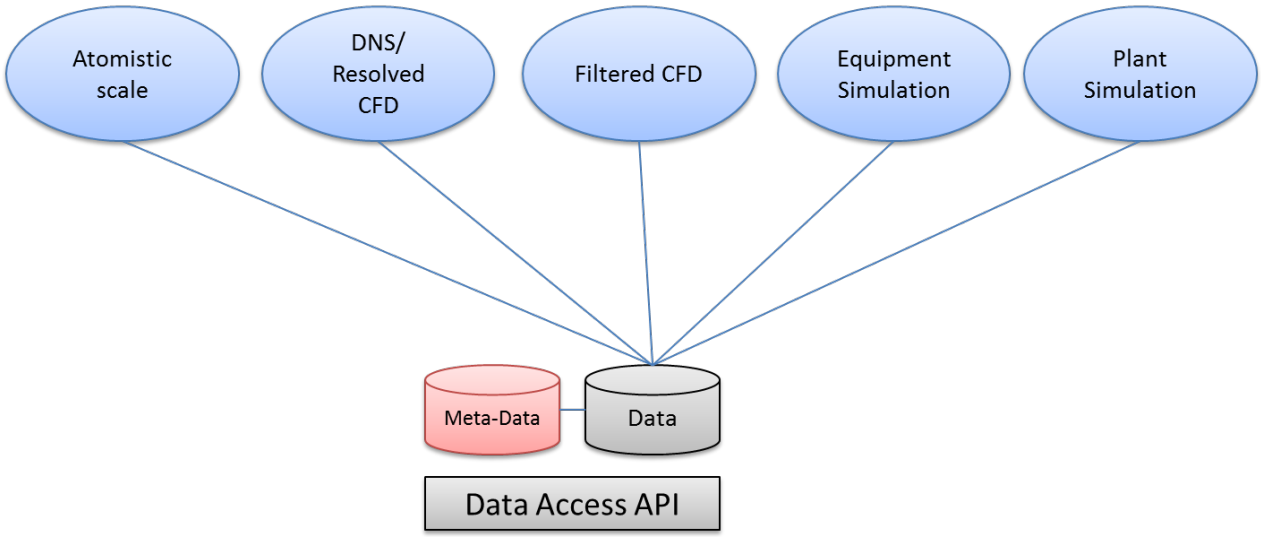

Figure 7 Meta-data and Data from all scales

3.1 Scale connectivity

The different work-packages in NanoSim have defined a set of simulation tools that can be regarded as data transformations. Each transformation step has a well-defined set of inputs and outputs. In the case where the output from one transformation is the input to another, the data must match semantically. In cases where the data does not match, an in-between transformation must be defined (adapter) in order for the workflow to be complete. In NanoSim there are 10-14 different simulation tools (transformations) in addition to experimental activities where data used for validation, as and input to simulators.

Defining meta-data for each work package has the following advantages:

- The flow of information can be documented
- Missing data in the information flow can be identifies (or other sources of data must be given)
- Through file-format adaptors the data stored in different files and formats can be accessed through a common API which has no dependencies on the original format and structure of the data.

- Enable the possibility of performing post-processing on data for the entire offline coupled workflow (for instance, observing non-linear effects of parameters at the atomic scale on the equipment or plan simulations).
- Introducing new simulation tools that can work within the workflow, without having to make changes elsewhere in the framework. (extendibility/scalability)
- Reduce the risk of introducing semantic errors in the coupling of models.

3.2 High Level Tools and Meta-Data Overview

The following table illustrates the different tools involved in the work-packages. In addition, the currently defined meta-data for each work-packages is listed in the row labeled *Data*.

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	• PaScal • 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 • PaScal Sample	• Resolved Flow • Kinetics Input • Reactor Performance	• Mesh • Fluid • Operational • Reactor • Reaction Spec	• Kinetics	• Gas Stream (ASPEN) • GasStream (Thermoflow)

The formal meta-data schemas reside in the NanoSim/Porto repository on GitHub. A tabulated form of the same data is given in the next chapters. The formal primitive types of the properties is not used in the following sections, but they are part of the formal entity declarations in JSON.

3.3 WP 2 Meta-Data

Model/Simulator	Entity/Group	Propertyname	Type	Unit/Symbol	Rank/Dimensions	Description
LIGGHTS	dump	timestep	int	s	-	Time step the snapshot was collected"
LIGGHTS	dump	nmberOfAtoms	int	-	-	The number of particles within the simulation
LIGGHTS	dump	boxBounds	double	-	-	Boundaries applied in the simulation, as well as bounds of the simulation box
LIGGHTS	dump	boxBoundsDescription	string	-	-	Description of the boundaries
LIGGHTS	dump	atoms	double	-	2 [nparticles,nproperties]	array of the per-particle properties
LIGGHTS	dump	atomsDescription	string	-	2 [nparticles,nproperties]	String containing description of the per-particle properties
OpenFOAM	Flow	pressure	double	m^2/s^2	1 [nCells]	List of scalars containing the actual pressure for each cell
OpenFOAM	Flow	rho	double	kg/m^3	1 [nCells]	List of scalars containing the actual fluid density for each cell
OpenFOAM	Flow	velocity	double	m/s	1 [nCells]	List of scalars containing the actual fluid velocity for each cell
OpenFOAM	Flow	voidfraction	double		1 [nCells]	List of scalars containing the actual voidfraction for each cell
OpenFOAM	Particle	position	double	m	2 [nparticles,3]	List of vectors representing particle positions
OpenFOAM	Particle	radii	double	m	1 [nparticles]	List of scalars representing particle radii
OpenFOAM	Particle velocity	velocity	double	m/s	2 [nparticles, 3]	List of vectors representing particle velocities

3.4 WP3 Meta-Data

Model/Simulator	Entity/Group	Propertyname	Type	Unit/Symbol	Rank/Dimensions	Description
VASP	VASP_extraction	surface_name	string	-	[]	The name of the surface - atom types (and orientation)
VASP	VASP_extraction	atoms	string	-	[]	List of atom type(s) (chemical symbol) and number of this this type excluding the surface atoms
VASP	VASP_extraction	atom_species	string	-	[]	Chemical formula excluding surface atoms in alphabetical order with H and C placed first
VASP	VASP_extraction	state	string	-	[]	Refers to the state of the molecule system - surface, gasphase, adsorbed state, transition state
VASP	VASP_extraction	site_name	string	-	[]	The adsorption or transition site(s) of the atom(s) or molecule(s) for the adsorbed or transition state
VASP	VASP_extraction	total energy	float	eV	[]	The total energy of the system from the DFT calculation
VASP	VASP_extraction	frequencies	float	cm^(-1)	[]	List of the frequencies calculated for the system
VASP	VASP_extraction	cell	float	Å	[3,3]	3x3 array with the lattice parameters of the system corresponding to the x, y and z directions
VASP	VASP_extraction	positions	float	- / Å	[nAtoms,4]	List of the atom type followed by its position in the x, y and z direction.
VASP	VASP_extraction	info	string	-		Optional - any relevant info can be added here
	Thermochemistry	Temperatures	float	K	[]	Temperatures for which thermodynamics data are given
	Thermochemistry	reaction	char / float	-	[nReactions]	Chemical reaction in default CHEMKIN format
	Thermochemistry	DeltaH	float	kJ/mol	[nReactions,nTemperatures]	Enthalpy change of reaction
	Thermochemistry	DeltaS	float	J/mol K	[nReactions,nTemperatures]	Entropy change of reaction
	Thermochemistry	DeltaG	float	kJ/mol	[nReactions,nTemperatures]	Free energy change of reaction
REMARC	Surface Chemkin data	comment	char	-	[]	Comment that explains a particular reaction scheme
REMARC	Surface Chemkin data	site_name	char	-	[]	Name of surface
REMARC	Surface Chemkin data	site_density	float	mol/cm^2	[]	Density of surface sites
REMARC	Surface Chemkin data	surface_species	char	-	[nSurfSpecies]	Name of species at surface
REMARC	Surface Chemkin data	thermo	char / float	-	[nSpecies,20+2*nElements]	Array containing names and composition chemical species along with parameters to calculate its thermodynamic quantities in given temperature ranges (default CHEMKIN format)
REMARC	Surface Chemkin data	energy_unit	char	-	[]	Unit of activation energy used in calculation of rate constants
REMARC	Surface Chemkin data	reaction	char / float	-	[nReactions,5]	Chemical reaction in default CHEMKIN format with corresponding Arrhenius parameters for the calculation of rate constants and comment on source and applicability of data
REMARC	Surface Chemkin data	forward_reaction_order	char / float	-	[nReactions,3]	Modified forward reaction order with respect to one species in the reaction specified as (Y/N, surface_species, reaction order)

REMARC	Surface Chemkin data	reverse_reaction_order	char / float	-	[nReactions,3]	Modified reverse reaction order with respect to one species in the reaction specified as (Y/N, surface_species, reaction order)
REMARC	Surface Chemkin data	duplicate	char	-	[nReactions]	Specifies whether reaction is duplicate of previous reaction (as Y/N)
REMARC	Surface Chemkin data	stick	char	-	[nReactions]	Specifies whether reaction is a sticking process (as Y/N)
REMARC	Surface Chemkin data	coverage_dependence	char / float	-	[nReactions,5]	Specifies if rate constant is modified as a function of surface coverage of give species as (Y/N, surface_species, parameter1, parameter2, parameter3)
	CHEMKIN-II data	species	string	-	[]	Comment that explains a particular reaction scheme
	CHEMKIN-II data	element	string	-	[nElements]	Name of chemical elements
	CHEMKIN-II data	species	string	-	[nSpecies]	Name of chemical species
	CHEMKIN-II data	thermo	float	-	[nSpecies,20+2*nElements]	Array containing names and composition chemical species along with parameters to calculate its thermodynamic quantities in given temperature ranges (default CHEMKIN format)
	CHEMKIN-II data	energy_unit	string	-	[]	Unit of activation energy used in calculation of rate constants
	CHEMKIN-II data	number_unit	string	-	[]	Unit of particle number to determine concentration in calculation of rate constants (molecules, moles)
	CHEMKIN-II data	reaction	float	-	[nReactions,5]	Chemical reaction in default CHEMKIN format with corresponding Arrhenius parameters for the calculation of rate constants and comment on source and applicability of data

3.5 WP4 Meta-Data

Model/Simulator	Entity/Group	Propertyname	Type	Unit/Symbol	Rank/Dimensions	Description
PASCAL	Particle	temperature	double	K	[nparticles]	The actual (volume-averaged) particle temperature
PASCAL	Particle	headFlux	double	W	[nparticles]	The actual total heat flux
PASCAL	Particle	intraParticleT	double	K	[nparticles,ngrid]	The actual intra-particle temperature (at each grid point)
PASCAL	Particle	position	double	m	[nparticles,3]	The actual particle position
PASCAL	Particle	radius	double	m	[nparticles]	The actual particle radius
C3PO	Sample	data	double	N	2 [nsamples, 3]	A list of vectors representing the force representative for the samples
C3PO	Sample	data	double	K	1 [nsamples]	A list of scalars representing the fluid temperature experienced by each sampled particle
C3PO	Sample	data	double	K	1 [nsamples]	A list of scalars representing the particle average temperature
C3PO	Sample	data	double	K	1 [nsamples]	A list of scalars representing the particle surface temperature
C3PO	Sample	data	double	m/s	2 [nsamples, 3]	A list of vectors representing the fluid-particle relative velocity within each sample
C3PO	Sample	data	double	-	1 [nsamples]	A list of scalars representing the voidage experienced by each sampled particle

3.6 WP5 Meta-Data

Model/Simulator	Entity/Group	Propertyname	Type	Unit/Symbol	Rank/Dimensions	Description
FLUENT	Resolved_Flow	pressure	double	m^2/s^2	1 [nCells]	Pressure for each cell
FLUENT	Resolved_Flow	temperature	double	K	1 [nCells]	Temperatures for each cell
FLUENT	Resolved_Flow	velocity_gas	double	m/s	2 [nCells, 3]	Velocity of the gas phase for each cell
FLUENT	Resolved_Flow	voidfraction	double		1 [nCells]	Volume fraction for each cell
FLUENT	Resolved_Flow	velocity_granular	double	m/s	2 [nCells, 3]	Velocity of the granular phase for each cell
FLUENT	Resolved_Flow	species_massfraction	double		2 [nCells, nSpecies]	Mass fraction of each species for each cell
WP7	Kinetics_input	particle type	String		1 (nCases)	Type of particle (eg. components and composition)
WP7	Kinetics_input	particle_size	double	m	1 (nCases)	The particle size used in each setup considered
WP7	Kinetics_input	reaction_type	string		1 (nReactions)	Format of rate expression
WP7	Kinetics_input	reaction_rate	double	depends on parameter	2 (nReactions, nParameters)	Parameters used in the rate expression
FLUENT	Reactor_Performance_output (for WP8 input and for WP6 comparison)	particle_size	double	m	1 (nCases)	The particle size used in each setup considered
FLUENT	Reactor_Performance_output (for WP8 input and for WP6 comparison)	geometry	double	m	2 (nCases,2)	Includes the geometry size in 2 dimensions (height & diameter). For more complex geometries more complex descriptions might be necessary
FLUENT	Reactor_Performance_output (for WP8 input and for WP6 comparison)	Bed loading	double	kg	1 (nCases)	Mass of particles used
FLUENT	Reactor_Performance_output (for WP8 input and for WP6 comparison)	Particle type	String		1 (nCases)	Type of particle (eg. components and composition)
FLUENT	Reactor_Performance_output (for WP8 input and for WP6 comparison)	temperature_in	double	K	1 (nCases)	Inlet temperature
FLUENT	Reactor_Performance_output (for WP8 input and for WP6 comparison)	temperature_out	double	K	1 (nCases)	Outlet temperature
FLUENT	Reactor_Performance_output (for WP8 input and for WP6 comparison)	Pressure	double	Pa	1 (nCases)	Operating pressure
FLUENT	Reactor_Performance_output (for WP8 input and for WP6 comparison)	Inlet_flow_rate	double	mol/s	2 (nCases, nSpecies)	Average molar flow rate into the reactor for each species

FLUENT	Reactor_Performance_output (for WP8 input and for WP6 comparison)	Outlet_flow_rate	double	mol/s	2 (nCases, nSpecies)	Average molar flow rate out of the reactor for each species

3.7 WP6 Meta-Data

Model/Simulator	Entity/Group	Propertyname	Type	Unit/Symbol	Rank/Dimensions	Description
Phenom	Mesh	ncell	int	-	-	Number of cells along z direction
Phenom	ReactionSpecifications	nspecies	int	-	-	Number of species in the system
Phenom	ReactionSpecifications	nRx	int	-	-	Number of reactions taking place in the reactor (input to the model)
Phenom	Fluid	velocity_in	double	[m/s]	[ncells,1]	Initial superficial gas velocity (input of the model)
Phenom	Fluid	massflowrate_in	double	[kg/s]	-	Mass flow rate of the gaseous stream at the reactor at the inlet (input of the model)
Phenom	Fluid	massflowrate_out	double	[kg/s]	-	Mass flow rate of the gaseous stream at the reactor at the outlet (to WP8)
Phenom	Fluid	massfraction_in	double	-	[nspecies,1]	Species Mass fractions at the inlet (input of the model)
Phenom	Fluid	massfraction_out	double	-	[nspecies,1]	Species Mass fractions at the outlet (to WP8)
Phenom	Operation	pressure_in	double	[Pa]	-	Pressure at the reactor inlet (initial pressure)
Phenom	Operation	temperature_in	double	[K]	-	Temperature at the reactor inlet (initial temperature)
Phenom	Operation	pressure_out	double	[Pa]	-	Pressure at the reactor outlet (to WP 8)
Phenom	Operation	temperature_out	double	[K]	-	Temperature at the reactor outlet (to WP8)
Phenom	Reactor	height reactor	double	[m]	-	Reactor height (to WP8)
Phenom	Reactor	reactor_di	double	[m]	-	Reactor inner diameter (to WP8)
Phenom	Reactor	reactor_do	double	[m]	-	Reactor outer diameter (to WP8)

3.8 WP7 Meta-Data

Model/Simulator	Entity/Group	Propertyname	Type	Unit/Symbol	Rank/Dimensions	Description
ASPEN Plus / Aspen Hysys -	Gas Stream Data - Input	Temperature	Float (excel)	C	1 [1 cell]	Temperature of Reduction Reactor

WP8						Product Gas Stream from Phenom Model of WP6
	Gas Stream Data - Input	Mass Fraction	Float (excel)		1 [n components]	Mass fraction of n components of Reduction Reactor Product Gas Stream from Phenom Model of WP6
	Gas Stream Data - Input	Pressure	Float (excel)	bar	1 [1 cell]	Pressure of Reduction Reactor Product Gas Stream from Phenom Model of WP6
	Gas Stream Data - Output	Temperature	Float (excel)	C	1 [1 cell]	Temperature of Product Gas Stream (GT Fuel after Water Gas Shift and CO2 Separation)
	Gas Stream Data - Output	Mass Fraction	Float (excel)		1 [n components]	Mass fraction of n components of Product Gas Stream (GT Fuel after Water Gas Shift and CO2 Separation)
	Gas Stream Data - Output	Pressure	Float (excel)	bar	1 [1 cell]	Pressure of Product Gas Stream (GT Fuel after Water Gas Shift and CO2 Separation)
Thermoflow - WP8	Gas Stream Data - Input	Temperature	Float (excel)	C	1 [1 cell]	Temperature - GT Fuel (output from Aspen Model)
	Gas Stream Data - Input	Volume Percent	Float (excel)	%	1 [n components]	Volume fraction of n components of GT Fuel (output from Aspen Model)
	Gas Stream Data - Input	Pressure	Float (excel)	bar	1 [1 cell]	Pressure of GT Fuel (output from Aspen model)
	Gas Stream Data - Input	Temperature	Float (excel)	C	1 [1 cell]	Temperature of Oxidation Reactor Product Gas Stream from Phenom Model of WP6
	Gas Stream Data - Input	Volume Percent	Float (excel)	%	1 [n components]	Mass fraction of n components of Oxidation Reactor Product Gas Stream from Phenom Model of WP6
	Gas Stream Data - Input	Pressure	Float (excel)	bar	1 [1 cell]	Pressure of Oxidation Reactor Product Gas Stream from Phenom Model of WP6