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	Contributors	Version	Lead beneficiary	Nature	Dissemin. level	Delivery date from Annex I	Actual delivery date dd/mm/yyyy
1	D1.1	Functional Requirements Document	Author: Thomas F. Hagelien Checked by: Stefan Radl (TUG) and Christoph Kloss (DCS)	1	SINTEF	Other	PU	31/06/2014	18/08/2014

1 Introduction

1.1 Purpose

The purpose of this document is to give as detailed descriptions of the requirements for the common environment "Porto" as possible. This document is a live document in the sense that requirements, as they are introduced and/or changed, will be reflected through revisions of the specification document.

1.2 Scope

The software to be implemented (Porto) will enable offline-coupling of physical processes that occurs on different scales. Porto will have a role both in the development of the process simulators, as well as in the execution of the simulation. Porto will also be an extendable framework on which custom features can be added and new simulators integrated.

1.3 List of definitions and abbreviations

1.3.1 Definitions

Actor	(UML) A specified role played by a user/system that interacts with the subject
Application	A software program, or group of programs, that is designed for the end user
ECMAScript	A programming language standardized by ECMA International in the ECMA-262 specification and ISO/IEC 16262. Also known by implementations such as JavaScript, Jscript and ActionScript
Entity	A type that contains a collection of properties that describes the properties of itself
Model	A software implementation of a simulator or prediction tool
Offline simulation	A loosely coupled simulation that doesn't depend on other simulations to be running concurrently, but may depend on data produced by another simulation
Plugin	A software component that adds a specific feature to an existing software application. Usually plugins can be loaded run-time (dynamically linked libraries/shared objects).
Relationship	A description of a dependency between two entities in a given context

1.3.2 Abbreviations

URI	Uniform Resource Identifier (RFC1630) – A string identifying a name and/or location of a resource.
API	Application Programming Interface – A formal (software) specification of the interaction between software components
TCP	Transmission Control Protocol - A reliable, stream-oriented, connection-oriented protocol
UDP	Use Datagram Protocol - a lightweight datagram-oriented connectionless protocol
UML	Unified Modeling Language

1.4 Overview

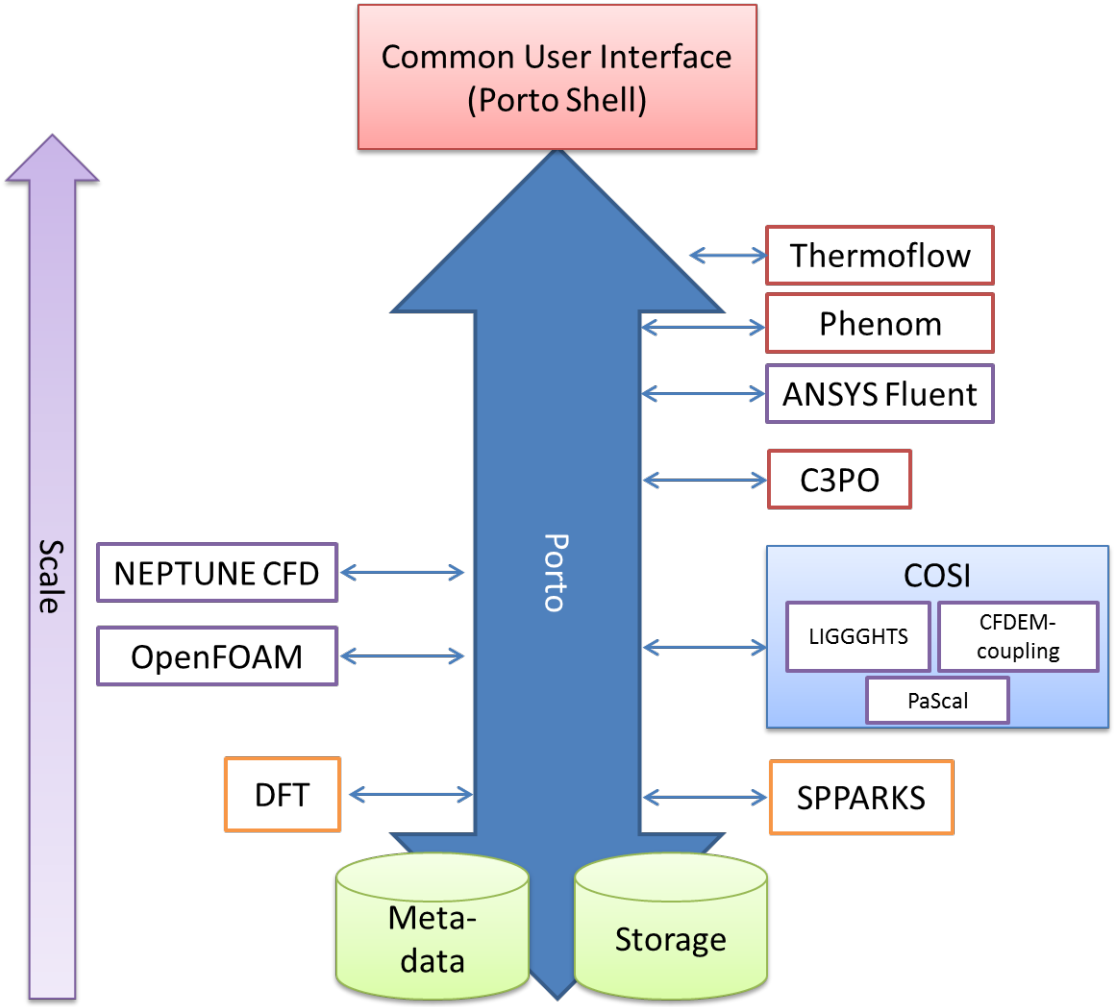
In chapter two of this document a description of the Porto system and its environment is given. In Chapter three the Use Cases collected from other work packages that are necessary for the requirements analysis are given. Chapter four contains the software requirements.

2 Overall Description

2.1 Product Perspective

The objective of the NanoSim EU-project is to create a multi-scale simulation platform that connects models spanning a wide range of scales from the atomic scale through the particle and cluster scales, industrial equipment scales and up to full system scale. The offline exchange of data and information between different simulation software will be facilitated by the NanoSim common framework 'Porto'.

Porto will primarily be a tool for data exchange and connecting different computational codes performing computations on different scales. This will be achieved by building adopters that utilize a common data description format for metadata. Porto will be able to store huge amounts of data by employing MongoDB, an open-source document database designed to support massive data volumes through horizontal scaling. Porto will provide a powerful scripting engine for performing tasks such as code-generation, running workflows, automating testing, report generation and performing analysis. Porto will also provide interfaces for creating custom I/O driver and scripting extensions.



2.1.1 User Interfaces

The main user interface to Porto is a scripting shell based on ECMAScript. It will be possible to use the scripting both as batch processes and as an interactive process. Different Porto-modules and extensions can be loaded at run-time.

2.1.2 Hardware Interfaces

None.

2.1.3 Software Interfaces

Porto will provide interfaces for creating application extensions and custom I/O drivers. For connections to the MongoDB database, Porto will use the officially supported MongoDB C driver (written in C89 compatible C and distributed under the Apache License v2.0). Porto will also make use of the Qt cross-platform application framework available under the open-source LGPL v2.1 license terms.

2.1.4 Communication Interfaces

Porto will provide interfaces for accessing offline data, and as such does not need to support any communication interfaces on the application/model level – however, for running workflows and performing automated tasks it is useful to have mechanisms to exchange information between processes. For this purposes there will be scripting extensions that supports asynchronous TCP and/or UDP communication.

2.1.5 Memory Constraints

There are currently no known memory constrains.

2.2 User Characteristics

Users of Porto will be divided into three categories.

1. Developer (personnel extending the Porto framework itself).
2. Advanced users/scientists (in charge of running and integrating simulators into the modeling framework of NanoSim or another modeling workflow)
3. Industrial users (engineers or scientists)

2.2.1 Developer

People with this profile have a thorough understanding of the design and implementation of the Porto framework, and are able to change/extend and adopt new modules and interfaces.

2.2.2 Advanced users/scientists

In this category we have simulator/model developers with a deep understanding of the requirements for interfacing their own software.

2.2.3 Industrial users

In this category we find domain experts with the ability to use the software for investigation and analysis purposes.

3 User Scenarios / Use Cases

In this chapter different Use Cases defined by consortium partners are summarized. The Use Cases illustrate how different simulation tools will be used. Based on these Use Cases, as well as input that is generated during workshops organized by the consortium members, the complete offline workflow and data exchange requirements will be mapped out. [Ref to workflow document]

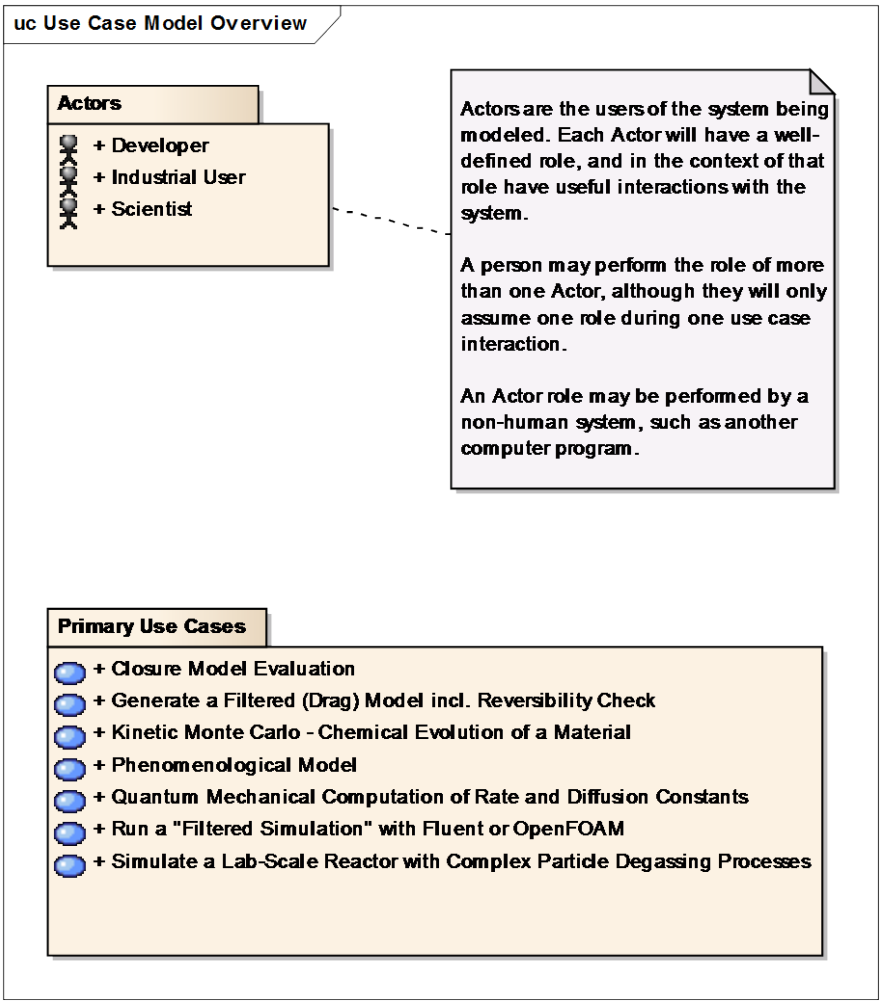


Figure 1 Overview - Use Case Model

3.1 Use Case UC1: Generate a Filtered (Drag) Model incl. Reversibility Check

Primary Actor: Scientist

Stakeholders and Interests:

- The NanoSim Consortium: Wants a reliable drag model for use with FLUENT & OpenFOAM.
- Scientific members of NanoSim: wants that filtered models are accepted by industry to strengthen cooperation with industry. Wants to have a clear workflow that makes generation of filtered drag models less cumbersome.

- EC: Wants to see that we (i.e., the NanoSim consortium) publish new models that help in predicting full-scale performance of reactors using nano structured materials during the project. Wants that these models & the corresponding workflow can be re-used by other (nano)-focused groups of researchers (outside & inside of the NanoSim project).
- Industrial members of NanoSim: Wants to see that the filtered drag model is "better" than the other currently available, wants to see the advantage over competitors when using filtered models (what can we do better in case we use these models?). Typically, no interest in model building, but understanding of the model usage and limitations (want to see reference results, documentation).
- CFDEM & OpenFOAM user community & consulting companies: want to use a predefined (& functional) workflow to generate customized filtered drag (or heat & mass transfer) models to be (i) published, or (ii) sold to customer. Want to re-use the workflow for similar tasks (e.g., build filtered models for other reactive systems).

Preconditions: Relevant domain-averaged particle volume factions, particle parameters (density ratio, as well as a number of dimensionless quantities like the particle Froude number Fr_p), physical and chemical parameters (e.g., the Schmidt, *Sc*, or the Damkoehler number, *Da*, etc.), as well as sub models (e.g., from direct numerical simulation, or correlations from literature) are available.

Success Guarantee (Postcondition):

- A function that predicts the effective drag (and other effective transfer coefficients) with an accuracy of better than 10 % when compared to fully resolved Euler-Euler simulations.
- A function that is universal (i.e., contains only dimensionless parameters), robust (has been tested for a variety of parameter combinations), and corresponds to known theoretical limits. It is demonstrated that the model is consistent with fully resolved Euler-Euler simulations (i.e., the model switches itself off in case it is not needed, i.e., it is user friendly).
- A function that significantly improves industrial-scale flow (& heat / mass transfer) predictions. This requires that the (i) the model yields significantly better predictions compared to a standard model (deviation to experimental data is reduced by a factor of 2 or more), and (ii) that the model predicts (without tuning of parameters) experimental data (e.g., hold up, domain-averaged reaction and heat transfer rates, gas/particle flow rates) with an accuracy better than 20% over a wide parameter range.

Main Success Scenario:

1. Setup
    a. Simulations with various domain-averaged volume fractions, particle parameters (density ratio, Fr_p, *Sc*, *Da*, etc.)
    → generate input scripts (e.g., for mesh generation, setup of simulation parameters, etc.), start scripts for use with the simulator (e.g., OpenFOAM).
2. Run Simulation
    a. Run OpenFOAM simulations on a remote cluster
3. Post process data

   a. Collect samples for slip velocities, forces, pressure gradient, etc. for various filter sizes (ca. 10 x 4 x 10^8 data samples for each run).
   b. Bin Data (Calculate mean, variance, etc.)
   c. The system is asked to generate a report that includes comparisons with previous results, known analytical limits and collapsed results with different filter sizes.
   d. Find collapse of data, and fit with some function (user input required).
4. Reversibility checks
   a. Construct model (user input required).
   b. Run semi-automated reversibility check (i.e., compare model prediction with fully resolved simulation). This requires fully resolved and filtered simulations to be run at the same time (but independently, of course)
   c. Iterate over 2. / 3. / 4. to find the correct model structure and parameters.
5. Validation
   a. Compare model prediction with experimental data and predictions of standard models.

Extensions (Alternative Flows)

1. If the resulting data from the simulation in Main Success Scenario step 2 does not look good, run the case for a longer time.
2. If the reversibility check fails, one has to rethink the model structure (modify the model in Porto), propose a new model structure, export the model to one or more simulation codes, and re-run the filtered model, and possible also run some new fully resolved Euler-Euler simulations.

Special Requirements:

None.

Technology and Data Variations List:

2a: during project: run FLUENT and NEPTUNE_CFDE on a cluster. After project: run OpenFOAM on a Linux Cluster.
3b: Porto is set up with mongoDB as the database backend
3d: Could be realized in Octave (or similar)

3.2 Use Case UC2: Quantum Mechanical Computation of Rate and Diffusion Constants

Main Success Scenario:

- Set up structure (bulk or surface), and run to find low energy stable structures
- Find transition state (i.e. NEB algorithm)
  - Get activation energy (ΔH, difference in energy between initial state and transition state)

- Do vibrational analysis
    - Obtain entropic prefactor
- Calculate rate constant of transition
- (Related) Do molecular dynamics of carriers, obtain diffusion constant

3.3 Use Case UC3: Kinetic Monte Carlo – Chemical evolution of a material

3.3.1 Introduction

If one is interested in the long-time scale physical and chemical evolution of a solid material on a 1-100 nm length scale and a time scale of a microsecond or much longer, Kinetic Monte Carlo (KMC) offers an attractive way of simulating this. A KMC-based computation requires as input (i) the initial structure and (ii) the rates of fundamental diffusion and reaction steps. The structure might be resolved into individual atoms or larger units such as molecules or clusters. These basic structural units can lie on a lattice of given symmetry or be distributed without any spatial bias. In the latter case, the interactions between atoms are most often provided through some force calculator, as a user-provided function or taken from a quantum mechanical calculation, such as DFT.

3.3.2 Procedure

1) Setup simulation (on-lattice KMC calculation)

- Define the geometry of the system (for a crystalline solid it is natural to choose a lattice with the same symmetry as the experimental structure). Apart from structural units sitting at lattice points, there are no restrictions on the connectivity of the structure such that it is possible to build in steps, kinks and pores
- Define the rates for all possible transitions in the system, e.g., diffusion jumps, chemical reactions, adsorption, and desorption. The rates can be estimated or derived from simulations or experiments

2) Run simulation (on-lattice KMC calculation)

- The simulation progresses through a number of simulation steps tracing a trajectory of the entire system. In each step 2 random numbers are drawn, one to determine the next step to take, based on the relative rates of all possible transition, and the other to determine the time it takes for the transition to take place, based on the total rate of escape out of the current state (sum of individual rates of each fundamental step)
- The simulation is run for a given number of steps that are sufficient to study the process of interest
- To obtain proper statistics it is often necessary to run a larger sample of KMC trajectories

3.3.3 Analyze simulation (on-lattice KMC calculation)

- Follow the property of interest, e.g., chemical composition, morphology or displacement, and output its characteristics, e.g., concentration, as function of time

- Derive the overall rate of change of interesting property as slope of characteristic parameter (concentration/displacement/other) versus time at long times

3.4 Use Case UC 4: Phenomenological Model

3.4.1 Introduction

Phenomenological modelling sacrifices some generality for a very large reduction in computational cost and complexity. The primary objective is therefore to maximize generality and establish clear bounds on the applicability of different models. Many closures for the phenomenological modelling of fluidized bed reactors are already available. NanoSim will therefore build on this available data source, and extend and/or refine the current capabilities of phenomenological models.

3.4.2 The Foundation

This well-established base and further improve the generality and fidelity of these models. In order to maximize generality, the phenomenological model should have the following features:

- Run transient or steady state simulations
- Offer easy access to a wide range of closure laws
- Handle a single reactor or groups of interconnected reactors
- Include user-friendly pre- and post-processing

This foundation must be built from scratch or compiled from existing codes

3.4.3 Closure Model Evaluation

The performance of existing closure models must be evaluated against results from filtered CFD modelling (WP5). Comparisons must be carried out over a wide range of parameters and over the spatial extension (i.e., the height) of the reactor. Ranges of good comparison and ranges of poor comparison must be carefully noted.

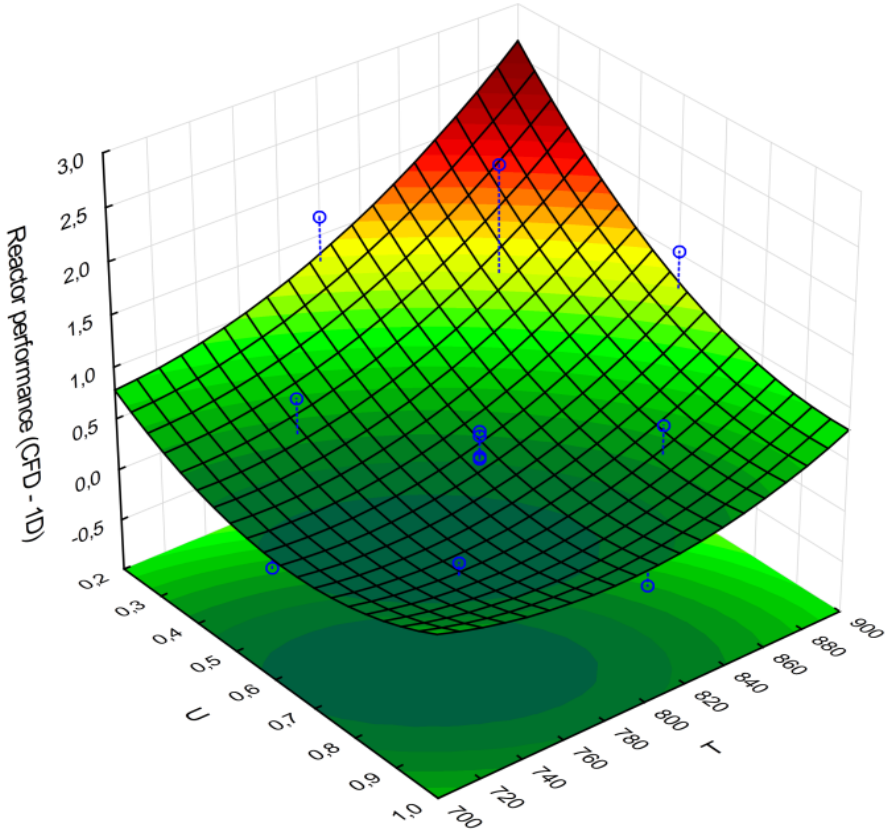


Figure 2 - Difference between CFD and 1D predictions of reactor performance as a function of fluidization velocity (U) and reactor temperature (T).

3.4.4 Improved Closures

Ranges of poor comparison must be analyzed in more detail in order to identify the defective closure law and make the necessary improvements. This step may require some more expensive resolved simulations to extract the information required for closure development (WP5 resolved simulations). New closures must then be evaluated again.

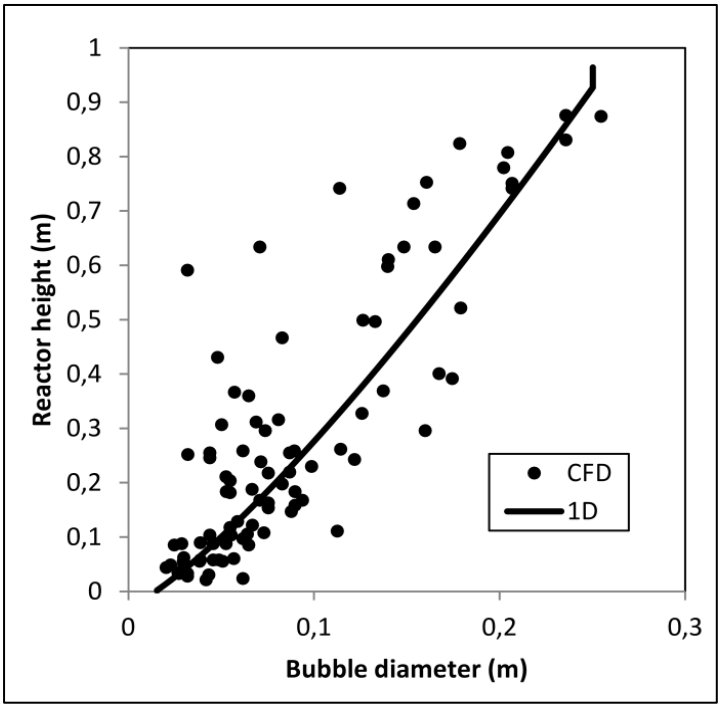


Figure 3 Comparison of bubble diameters predicted by CFD and 1D models

3.4.5 The Final Package

The phenomenological tool will be developed especially for industrial application. Participating industrial partners will receive a fast and user-friendly modelling tool programmed in Matlab or Octave with well-documented guidelines for closure selection.

3.5 Use Case UC5: Run a "Filtered Simulation" with Fluent or OpenFOAM

3.5.1 Introduction

This use case details the procedure how to run a filtered CFD simulation of a fluidized bed using the NanoSim platform. For the current project, this is relevant for industrial end users (e.g., Andritz), as well as partners that will implement and test filtered drag laws (i.e., TUG and DCS).

Up to now, filtered models have been used in OpenFOAM (Euler-Euler) and CFDEM (Euler-Lagrange) simulations via dedicated software modules directly implemented into these software packages. This use case should illustrate how NanoSim will interface with any of the CFD software packages (e.g., OpenFOAM, FLUENT, NEPTUNE_CFD) used within this project.

3.5.2 Setup simulation

1. Define the **geometry** (often complex), and generate the **mesh** (check mesh quality)**.**
2. **Define system parameters and model** (pick an appropriate filtered model from a database).
3. Generate **UDF for Fluent**, or select model and set parameters in an **input file for OpenFOAM**
4. (alternatively: compile a library to be called at run time, this can also be an option for interfacing with NEPTUNE_CFD in future extensions)

A web-based solution might bring benefit to industrial end-users.

3.5.3 Run simulation

1. Run Fluent / OpenFOAM simulations and save statistics: instantaneous slip velocity, bed expansion, fuel conversion, etc.
2. Visualize results online to check status of the simulation (e.g. via remote access and a **web-based solution)**.

3.5.4 Postprocess

1. Post-process simulation results with Paraview (or FLUENT GUI). Assemble & Call OpenFOAM post-processing routines if required for more sophisticated post-processing.
2. Compare to reference data & generate reports (e.g., to support design decisions, etc.)

3.6 Use Case UC6: Simulate a Lab-Scale Reactor with Complex Particle Degassing processes

3.6.1 Introduction

This use case details how to embed a particle-scale simulation tool from the NanoSim module „PaScal“ into a CFD code. For the current project, this is relevant for WP „Lagrangian Modelling“, where fully resolved simulation of flow, concentration and temperature fields surrounding a small ensemble of particles will be performed.

Up to now, complex particle processes often have been accounted for by embedding UDFs into FLUENT. This use case should illustrate how NanoSim will auto-generate code that can be used in any CFD software packages (e.g., OpenFOAM, FLUENT, NEPTUNE_CFD) to account for complex particle-scale processes.

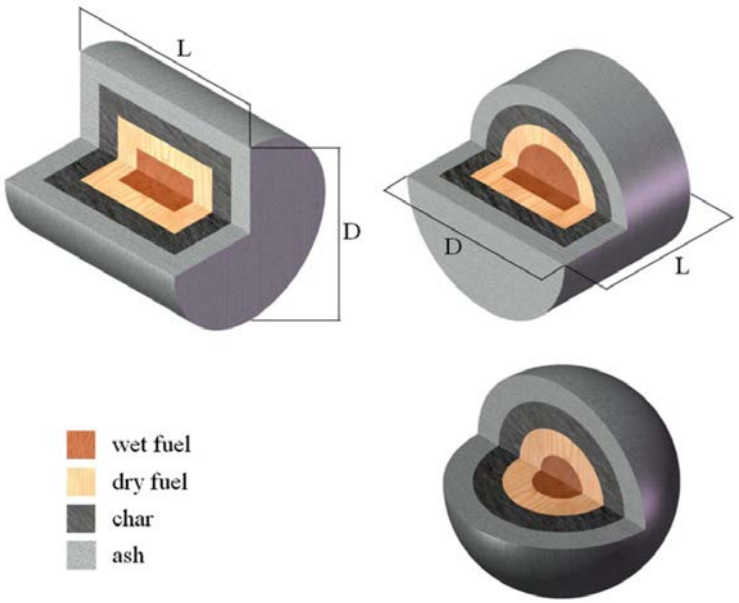


Figure 4 “Thunman” 4-layer particle often used for biomass particles (Mehrabian et al., 2012)

3.6.2 Procedure

- Define the **geometry** (often complex), and **mesh** (check mesh quality)**.**

- **Define system parameters and model**
  (pick an appropriate particle scale model with sub-models, e.g., for pyrolisis, char gasification, etc.). This will often be a **system of ODEs**!
- Generate **UDF code for Fluent**; or **generate top-level code for *PaScal***, compile, and link to OpenFOAM

A web-based solution could be considered as an educational resource.

3.6.3 Run Simulation

- Run Fluent / OpenFOAM simulations and save statistics: particle temperatures, char burnout, etc.
- Visualize results online to check status of the simulation. Remote access via **web-based solution**

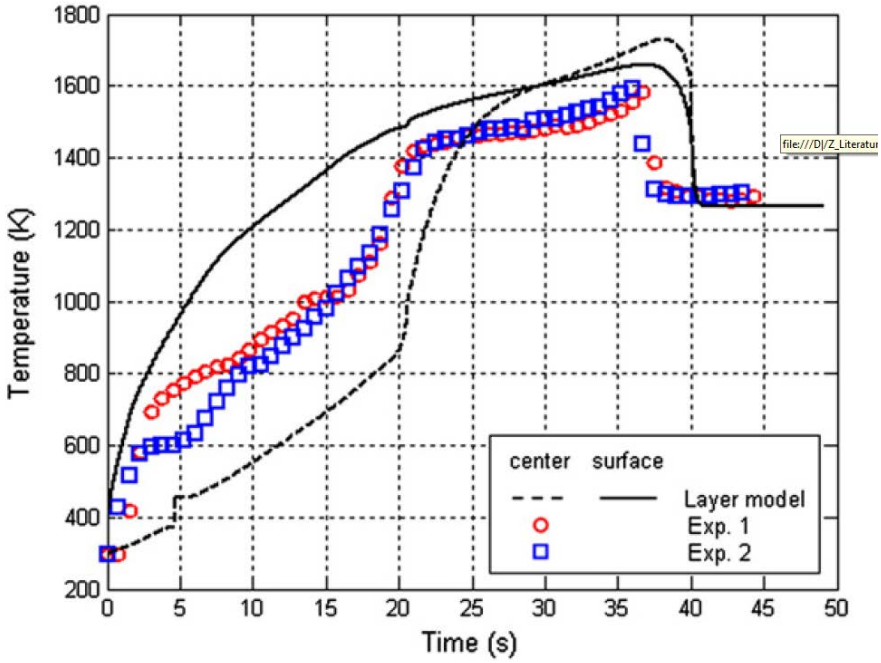


Figure 5 Temperature at the center and the surface of a wood particle (Mehrabian et al., 2012)

3.6.4 Post Process

- Compare to reference data & generate reports (validation).

3.7 Use Case UC7: Parameterize PaScal to Predict the Metal Distribution in a Porous Particle

3.7.1 Introduction

This use case details how to parameterize PaScal in order to predict the distribution of the active (i.e., the carrier metal, or a catalyst) in a porous particle. The description is based on the model of Liu et al. (CES 79:187-199; Ind. Eng. Chem. Res. 49:2649–2657), which has been applied to wet impregnation of porous alumina particles with nickel nitrate hexahydrate. Similarly, this use case can be applied to, e.g., other impregnation processes of support particles, or (with further adaptation) to

parameterize the particle scale simulation tool "*PaScal*" based on TGA experiments for the prediction of processes in the fuel and air reactor. Thus, this use case should illustrate a general approach where experimental data and atomistic simulation data from the Porto database is combined to establish a model.

3.7.2 Setup Simulation

1. Define the **geometry and discretization** of the particle (spherical, cylindrical, or sheet; particle size; discretization scheme, numerical solution parameters, fitting algorithm), and **boundary and initial conditions** to reflect the influence of the surrounding fluid (e.g., constant gas-phase composition and fixed heat transfer coefficient; temperatures, initial concentrations).
2. **Define the particle parameters**: the key experimental data required here is the **pore size distribution** (e.g., measured via mercury intrusion), which is used to estimate the permeability (both for the gas and liquid phase; simple models can be used based on the porosity; thus, a sub-model for permeability needs to be picked from a database).
3. **Define the transport properties**: key parameters like diffusion coefficients, thermal heat conductivities and capacities, viscosity, density as well as the surface tension needs to be modelled (for both the liquid and the gas phase; models need to as a function of temperature, and in case of concentrated solutions of the salt concentration). The user needs to pick **these parameters from a database, or select an appropriate model for its prediction**. For parameters which are connected to uncertainty, there should be an option to perform a sensitivity analysis (i.e., **select parameters for later analysis)**.

3.7.3 Run Experiments

1. Conduct simple experiments to determine unknown model parameters (e.g., adsorption parameters).
2. Conduct impregnation & drying experiments; record drying rate and temperature profiles, as well as metal distribution via micro-X-ray fluorescence spectroscopy

3.7.4 Run Simulation & Fit

1. Pick a certain set of experimental data (fit data set), run PaScal (stand-alone) to fit certain model parameters with fit data set.
2. Run PaScal model with optimized parameters and save results (base case & sensitivity study; quality of fit)

4 Specific Requirements

In this chapter all requirements and constraints of the software to be developed are given. The product will adhere to these requirements.

4.1 External Interface Requirements

4.2 Functional Requirements

Requirements in this chapter are formulated as Use Cases using this form:

"As a [*role*] I want/need [*goal/desire*] so that [*benefit]*" - where [*role*] will be replaced by the primary actor of the Use Case, and the [goal/desire*]* will be replaced by the required functionality.

4.2.1 Running jobs from shell

Use Case

Title	Comment
Name	Running jobs from shell
Purpose	As a scientist I need to be able to execute external processes from a scripting shell so that I can perform post-processing operations from the interactive shell.
Main Success Criteria	A process can observably be executed from the scripting shell

4.2.2 Working with files from shell

Use Case

Title	Comment
Name	Working with files and standard IO routines from shell
Purpose	As a scientist I need to load, store and create files and directories from the shell so that I can navigate around and work with the file system.
Main Success Criteria	Being able to create a directory, change current directory to the newly created one. The store text as a new file in the current directory, and read it back in

4.2.3 Communicate between script processes

Use Case

Title	Comment
Name	Simple script application communication protocol
Purpose	As a scientist/developer, I want to be able to send and receive asynchronous messages to other scripting processes in order to perform operations that depend on the status of other running processes, i.e., start a post processing operation job when a given workflow is signaling that data is ready
Main Success Criteria	A message is passed though the asynchronous communication protocol from one script job to another and verified

4.2.4 Adding a new set of entities

Use Case

Title	Comment
Name	Registering a new dataset (entity)
Purpose	As a developer I need to be able register the information about how a certain data point should be interpreted, in order for other programs to be able to read the data and ensure the correct semantics.
Main Success Criteria	A file describing the contents of an entity is registered into a shared database

4.2.5 Defining relationships between entities

Use Case

Title	Comment
Name	Entity relationships
Purpose	As a developer I need to define relationships between entities in such a way that this can be exploited during the searching/restructuring of data, and during code generation
Main Success Criteria	A schema for defining relationships is defined and user provided contents can be validated against this schema.

4.2.6 Resolve version dependencies and transform internal data

Use Case

Title	Comment
Name	Version dependency
Purpose	As a user I need to be able to assure that data read into an application is consistent, such that my models are consistently defined even though the external source might change.
Main Success Criteria	An internal data representation (entity) can be restored from the state of a different entity (or different version of the entity)

4.2.7 Setting up Workflows

Use Case

Title	Comment
Name	Workflow setup
Purpose	As an industrial user I want to define a set of execution steps and automate the running of a batch of coupled simulations
Main Success Criteria	A setup file is created which defines the execution order and other conditions. The file is executed in the scripting environment and causes the applications to be run in the defined sequence, producing a verifiable result.

4.2.8 Writing and validating data descriptions

Use Case

Title	Comment
Name	Writing and validating data description files
Purpose	As a scientist/developer I want to describe a data point (entity) and validate the file against the formal specification such that I can verify the correctness of the data before it is stored in the metadata-database
Main Success Criteria	An entity is specified in the formal schema. By running a validation script that takes the entity description file as a parameter, the user will receive a report indicating whether the specification is valid or not.

4.2.9 Adding a new external driver

Use Case

Title	Comment
Name	Adding a new external driver
Purpose	As a developer I need to integrate a new file format into the Porto platform such that data from an external file can be used as input for different simulators in the NanoSim environment.
Main Success Criteria	By implementing a driver connector using the API specifications from Porto, it will be possible to instantiate an entity with data from an external source.

4.2.10 Adding a new script plugin

Use Case

Title	Comment
Name	Adding a script plugin
Purpose	As a developer I want to extend the scripting functionality with a custom plugin such that I can perform different data analysis tasks in the interactive shell
Main Success Criteria	By implementing a Porto plugin based on the specified Porto Plugin API, and placing the plugin in the correct location on the file system, the scripting environment will load the functionality and make the new functions available for scripting.

4.2.11 Plotting data

Use Case

Title	Comment
Name	Plotting data contents from entities
Purpose	As a scientist/industrial user I need to validate simulation results by creating various plots for different entities, regardless of their origin, such that data from different sources can be analyzed and processed to develop a model on the next level.
Main Success Criteria	By specifying a specific instance of an entity, its data is available for plotting.

4.2.12 Importing data from external sources

Use Case

Title	Comment
Name	Importing data from external source
Purpose	As a scientist I need to be able to connect to an external data source and read the contents such that the information can be processed further.
Main Success Criteria	By pointing the data source to a specified external driver type and an address (URI) the software should be able to read the contents into an entity.

4.2.13 MongoDB driver

Use Case

Title	Comment
Name	Using MongoDB as data storage
Purpose	As a developer I want to use MongoDB to store the contents of my entities in order to exploit the MongoDB facilities such as auto-sharding, full index search, map/reduce and huge data support.
Main Success Criteria	By specifying a database - and collection name, an entity should be serialized to the given storage location, and read back in again.

4.2.14 Storing 'internal' data

Use Case

Title	Comment
Name	Storing the current state of an entity
Purpose	As a developer/scientist I want to store the current state of an entity in my preferred programming language, with the ability to retrieve the contents from a different application written in the same, or different, language
Main Success Criteria	An entity is instantiated in a software application and stored in the 'internal' database (i.e., MongoDB).

4.2.15 Retrieving 'internal' data'

Use Case

Title	Comment
Name	Retrieving a stored state of an entity
Purpose	As a developer/scientist I want to retrieve a stored state of an entity
Main Success Criteria	See [UC: MongoDB driver]

4.2.16 Importing 'external' data to 'internal' data

Use Case

Title	Comment
Name	Importing external data to internal data
Purpose	As a scientist/developer I want to be able to import data from an external source, and store the contents in the internal storage
Main Success Criteria	Data stored in an external file is identified through the appropriate metadata and read into Porto. Then the entities are written back to the 'internal' storage (MongoDB).

4.2.17 Exporting 'internal' data to an external (proprietary) data format

Use Case

Title	Comment
Name	Exporting data
Purpose	As a scientist I want to export internal data into an external data format such that proprietary software can import the data.
Main Success Criteria	By importing data stored in the Porto database and exporting it through a specified storage device the data can be read by a third party application.

4.2.18 Generating source code from JSON

Use Case

Title	Comment
Name	Generate source code from JSON
Purpose	As a developer I want to generate source code by combining a template file with a JSON file, and produce compile-able output code such that the tedious and possibly error prone task of implementing large amounts of similar code can be fully automated.
Main Success Criteria	A template file for producing source code is combined with a data source specified in JSON. By processing these files an output file is produced that can be compiled, linked and run.

4.2.19 Generating source code from internal data

Use Case

Title	Comment
Name	Generate source code from internal data
Purpose	As a developer I want to generate an OpenFOAM model (i.e., source-code) based on the state of an entity stored in the internal database, such that new models can be produced and compiled automatically as part of a process workflow.
Main Success Criteria	A template file for producing source-code is combined with an instance of one or more entities. By running the generator an output file is produced that can be compiled, linked and run as part of the simulation environment (UDF or OpenFOAM model)

4.3 Performance Requirements

As Porto is a framework for offline coupling of models, it is clear that performance does not play a major role.

4.4 Design Constraints

As Porto is based on the SINTEF Open Framework and Tools (SOFT 5), it is inherently a data centric framework. Models/simulators revolve around data in a way that restricts the developers to adhere

to standard ways of specifying the meta-data for information that is exchanged. In addition, one idea behind the data centric framework is to decouple structure (hierarches) from the data. This is done by organizing the data into *entities* (i.e., a set of properties) that are linked with other *entities* through a separate specification of their *relationships* (see Entity-Relationship modeling, ER-modelling, as discussed in D.1.2, Detailed Workflow Document).

4.5 Software System Attributes

4.5.1 Reliability

While failure is not a critical issue in the offline coupling of simulators in NanoSim, Porto will need to be able to safely store, keep and retrieve large amounts of data (<1 Tb) and support mechanisms for providing safe, and fault tolerant handling of data.

4.5.2 Availability

There are no availability requirements for the Porto framework.

4.5.3 Security

Security is currently not a requirement of the Porto framework

4.5.4 Maintainability

It is imperative that Porto is implemented in such way that the software can be extended, adapted to changes, improved and corrected. Key principles for designing the software are loose coupling and high cohesion. Loose coupling refers to having few interdependencies. Module interdependencies can often cause problems when changing one part of the code which has effects on other parts, which also might have to be changed, and so on. High cohesion is the principle of creating classes and modules that does *one specific job*, in contrast to low cohesion when a class or module does many unrelated tasks.

Another principle of designing Porto to be maintainable is identifying and implementing known design patterns where applicable. Design patterns are proven "best practices" for different reusable solutions to common design challenges, and are known to improve code readability (for coders) and architects familiar with the patterns.

4.5.5 Portability

Porto will be used on different computing platforms from desktops to clusters. Because of this the code needs to run on multiple operating systems. Specific target operating systems are Windows and Linux (32 and 64 bits).

4.6 Other Requirements

None