Towards computer-aided multiscale modelling: A generic supporting environment for model realization and execution

Yang Zhao, Cheng Jiang, Aidong Yang[1]

Division of Civil, Chemical and Environmental Engineering, Faculty of Engineering and Physical Sciences, University of Surrey, Guildford GU2 7XH, UK

Abstract

Computer-aided multiscale modelling (CAMM) may be carried out in three consecutive stages, namely conceptual modelling, model realization, and model execution. Following earlier work on a conceptual modelling tool which aims to support the first stage of CAMM, prototypical tools for realizing conceptual models and for the execution of simulation are developed in this work, with the assumption that a multiscale simulation is to be carried out by means of integrating existing single-scale models. More specifically, the tool that supports model realisation helps modellers generate information required for executing the multiscale model. The model execution stage is in turn supported by a component-based simulation environment. Two different multiscale simulation modes, namely "coordinator driven" and "master tool driven", are identified and supported separately. Details of the design and implementation of these tools are provided. Two reactor modelling examples are used to validate these tools and to demonstrate their application.

Keywords: multiscale modelling, computer-aided modelling, tool integration

[1] Corresponding author. Tel.: +44 1483 686577; fax: +44 1483 6581. E-mail address: a.yang@surrey.ac.uk (A. Yang).

1. Introduction

Multiscale modelling combines models of different scales of a system in order to obtain an overall model of desired quality or computational efficiency which is difficult to achieve by a single scale model. This modelling paradigm is widely regarded as a promising and powerful tool in various disciplines, including process engineering (Charpentier, 2002, 2009; Braatz et al., 2004; Vlachos, 2005; Lucia, 2010; Jaworski et al., 2011), material science (Karakasidis and Charitidis, 2007; Maroudas, 2000), computational mechanics (Liu et al., 2007; O'Connell and Thompson, 1995), and system biology (Southern et al., 2008), to name a few.

Compared to a single-scale model, multiscale model is usually much more difficult to construct due to a range of conceptual, numerical and software implementation challenges (Yang and Zhao, 2009; Zhao et al., 2012). A number of efforts have been made to address these challenges. One of the notable outcomes is classification of multiscale modelling approaches, schemes or frameworks which define the generic ways how submodels of different scales of a system can be introduced and coupled to form an integral model (Pantelides, 2001; Ingram et al., 2004; Li et al. 2005; Vlachos, 2005). Another category of effort is on the support of integration between specific types of modelling techniques and tools each of which addresses one particular scale of a multiscale system (Bezzo et al., 2004; Kougoulos et al., 2005; Kulikov et al., 2005; Morales-Rodríguez & Gani, 2009). The software integration aspect of multiscale modelling may be addressed using approaches comparable to those developed previously for the tool integration in (single-scale) modelling and simulation. In this regard, the component-based approach adopted by the process engineering community via the CAPE-OPEN initiative (Braunschweig et al., 2000) is particularly relevant. The result of CAPE-OPEN has been vital for the improvement of openness in commercial process

modelling software, and for the development of open simulation platforms such as CHEOPS (Schopfer et al., 2004). A more recent application of CAPE-OPEN standards is reported by Zitney (2010), where an advanced process engineering software tool, termed a "co-simulator", has been developed which supports the integration of steady-state process simulation with other multiphysics-based equipment simulations (e.g. CFD) in order to get high-fidelity simulation results for designing and analysing energy systems. In some other areas such as biological systems modelling (e.g. Hetherington et al., 2007; Hunter et al., 2005) and multi-physics modelling (e.g. Smirnov, 2004), there have been similar developments in supporting the combined use of different models and simulation tools. Senin et al. (2003) developed a distributed object modelling environment (DOME), which supports the integration of different modelling tools involved in various aspects or stages of a (mechanical) engineering design project. In the modelling of building systems, coupling strategies between building simulation and CFD were explored towards better predictions of indoor thermal environment (Wang et al., 2009).

Despite the advances in tool integration for modelling and simulation including multiscale modelling, there is still a lack of *generic* supporting mechanism for realising multiscale simulations based on a tool integration approach. In an ongoing effort which explores the feasibility of developing generic tools for Computer-Aided Multiscale Modelling (CAMM), a methodology has been proposed (Yang and Zhao, 2009; Zhao et al., 2012). This methodology applies a three-stage strategy by which computer based support of various degrees of "automation" is offered to conceptual modelling, model realization, and model execution. Following this methodology, a conceptual modelling tool has been prototyped (Zhao et al., 2012). Termed a conceptual model, the output of the tool is a specification of what scales are involved, how each scale should be modelled

(in terms of components, phenomena, properties, laws and connections among components), and how different scales are connected. A conceptual model may be used as a systematic documentation of a multiscale model. Moreover, the three-stage methodology for CAMM suggests that a conceptual model can be taken as the input to the subsequent modelling stages eventually leading to the successful execution of the intended multiscale simulation.

This present contribution focuses on computer-based support to multiscale model realisation and execution. In the rest of the paper, Section 2 presents the overview of the methodology of CAMM. Sections 3 and 4 address the stages of model realisation and execution respectively by introducing the design and implementation of a set of software tools. In Section 5, two case studies are presented to demonstrate these tools, before concluding remarks are given in Section 6.

2. Overview of the overarching methodology for CAMM

The previously developed methodology for CAMM, mentioned in Section 1, has two important aspects. As the first aspect, a unified, hierarchical framework of conceptualisation is developed and taken as the basis for all CAMM tools, in order to maximise the generality of these tools. This hierarchical conceptualisation consists of (i) a fundamental level which offers a unified theory for general multiscale systems and (ii) a domain-specific level which is on top of the fundamental level to provide domain specific conceptualization. The central idea is to develop CAMM tools solely based on the unified theory as far as possible so that the solutions will be most generic. When applying these tools on specific domains, domain-specific conceptualisation can be taken as explicit input without having to modify the tools themselves. It has been proposed to use ontologies to represent the conceptualisation at these two levels. In particular, a generic ontology for multiscale systems (Zhao et al., 2012; see also Yang and Marquardt, 2009) and an adapted part of OntoCAPE

(Morbach et al., 2007; Marquardt et al., 2010) as a domain ontology for chemical process engineering have been adopted so far for evaluating the methodology. Both ontologies are currently coded using the ontology modelling language OWL (http://www.w3.org/TR/owl-ref/).

The second aspect of the methodology is to apply a three-stage strategy to maximise computer-based support to modellers at different modelling stages. As briefly outlined in Section 1, it is considered that developing a multiscale model generally starts with conceptual modelling through the description of a multiscale system by means of physical concepts. Following this stage, the model realisation and execution stages deal with *(i)* the realisation of the conceptual model and *(ii)* the actual execution of model, respectively. As the second stage, model realization is responsible for transforming the conceptual model generated by the conceptual modelling tool into a form which is ready to be executed. One possible approach to model realisation is automatic code generation (Yang et al., 2004). The essence of this approach is that a mathematical model is composed by selecting and customizing elements from a library of basic building blocks according to the conceptual model. This approach would result in a "single" set of equations that can be solved collectively to carry out multiscale simulation. However, the applicability of this approach is limited in that, in contrast to building a model completely from scratch, a more realistic way of multiscale simulation is by the integration of existing modelling tools (e.g. gPROMS, Fluent, Aspen Plus, etc.), each simulating one particular scale of a multiscale system. Figure 1 illustrates the computer-aided mutiscale modelling process based on realizing and solving models by the tool-integration approach.

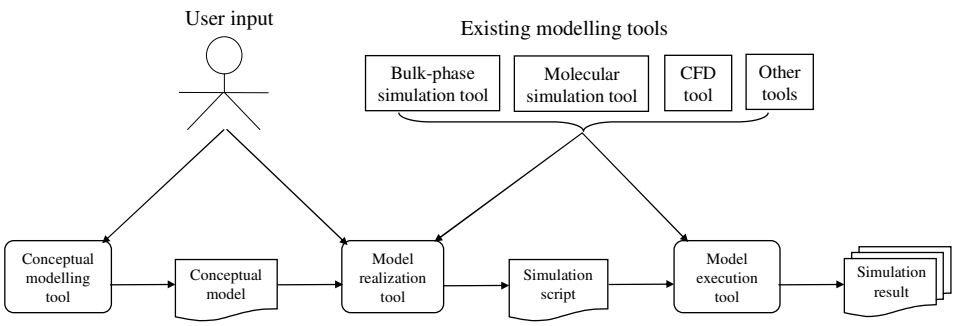


Figure 1. The workflow of simulating multiscale models based on the three stage strategy.

For a particular modelling problem, the conceptual modelling tool takes user input, including a domain ontology, and generates a conceptual model as mentioned above. The conceptual model, in the form of an OWL file, is subsequently taken as the input by the model realization tool to produce a simulation script which is a collection of information required for executing the entire model. The model execution tool in turn runs the simulation by invoking and coordinating the models for individual scales according to the simulation script.

Details of the overarching methodology and the conceptual modelling tool have been reported by Zhao et al. (2012). The present contribution will subsequently focus on the tools for model realisation and execution.

3. Model realization

To support the tool integration based approach to model realization, a simulation script generator (SSG) is developed in this work.

3.1 Design of simulation script generator (SSG)

As shown by Figure 2, the SSG is constructed upon the same ontology for general multiscale systems as used by the conceptual modelling tool developed earlier (Zhao et al., 2012) so that the SSG is able to correctly interpret conceptual models. The objective of SSG is to take a conceptual

model as input, which should have defined the scales to be modelled and inter-scale links (e.g. aggregation and disaggregation of data), and to produce a simulate script that specifies which modelling tool is used for simulating a scale defined in the conceptual model, what software component should be executed to realise a pre-defined inter-scale link, and what is the order of running these tools and components. In the sequel, a modelling tool which is used for simulating a specific scale is referred to as a "scale solver", a "scale component" or a "single scale tool" depending on the context. A software component that realises an inter-scale link, on the other hand, is referred to as an "inter-scale solver" or an "inter-scale software component".

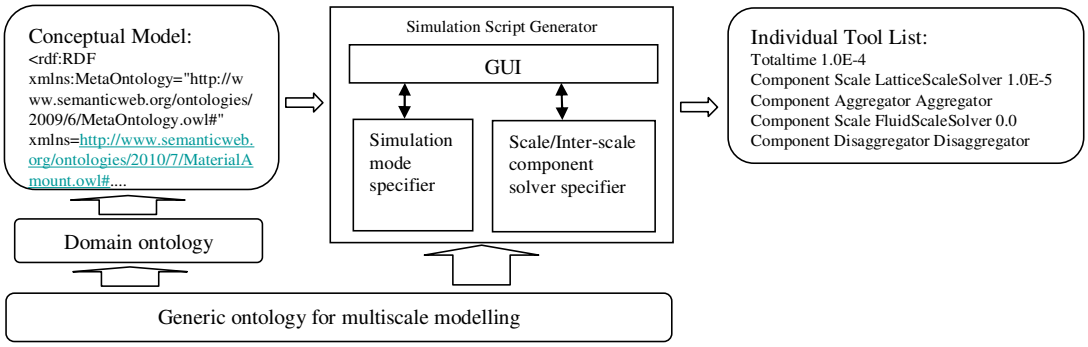
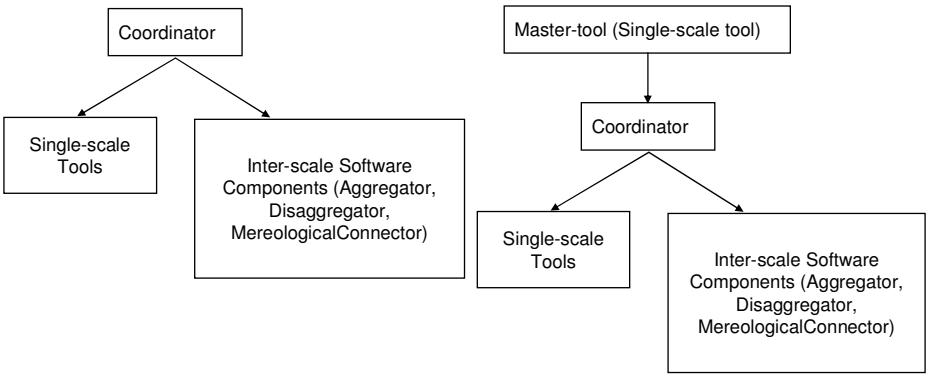


Figure 2. Design of the Simulation Script Generator (SSG)

3.1.1 Specification of simulation mode

Two different modes can be distinguished for tool-integration based multiscale simulation, namely the *coordinator driven* mode and the *master tool driven* mode as illustrated in Figure 3. Both modes involve a simulation coordinator, which is an important component in the model execution system which is explained in Section 4. In the *coordinator driven* mode, the coordinator plays a dominant role in the process of model execution while other components, including single-scale tools and inter-scale software components such as aggregators and disaggregators, are driven by the coordinator. The coordinator controls the total length of the simulation duration, the order, timing

and duration (in terms of simulated physical time) of execution of each single-scale tool, as well as the exchange of data between single-scale tools and inter-scale software components. In the *master tool driven* mode, one of the single-scale simulation tools is selected as the "master tool" which controls the simulation process by driving the coordinator in order to obtain data from the tools for other scales, when and as necessary. With the *master tool driven* mode, the role of the coordinator is to coordinate the execution of individual tools other than the master tool. For a particular modelling problem, the SSG allows a user to select one of the two simulation modes depending on whether the simulation is to be driven by a "master tool" or not. For each of the two simulation modes, the SSG allows the user to specify respective details. As already indicated above, the details for the former mode include the total time of the simulation, the order of executing the single-scale tools, as well as the length of duration for each step of execution of these tools (which is comparable to the time step size for modular dynamic simulation of process flowsheets, see e.g. Kulikov et al., 2005). When the latter mode is selected, time specification function is disabled since the simulation process is controlled by the "master tool"; there is no need for the coordinator to know of the simulation time. However, the SSG will in this case allow the user to choose which particular scale is to be solved by the "master tool".



(a) Coordinator-driven mode (b) Mastertool-driven mode

Figure 3. Two modes for tool integration based multiscale simulation.

3.1.2 Specification of scale/inter-scale software component

This function allows a modeller to specify a modelling software tool for solving each of the scales defined in the conceptual model and also an inter-scale software component (provided by the model execution system which is explained in Section 4) for each instance of the "InterscaleLaw" adopted in the conceptual model (Zhao et al., 2012) which characterises a connection between two different scales.

It should be noted that an inter-scale software component is meant to be generically applicable as opposed to being valid only for a specific simulation case. It implements an inter-scale law as defined in the generic ontology for multiscale modelling (Zhao et al., 2012). A simple example is the "averaging aggregation law", which mathematically can be represented as:

$$y = \frac{\sum_i x_i w_i}{\sum_i w_i} \tag{1}$$

where y is a property of a higher (i.e. coarser) scale, x_i is a property of part i of a lower (i.e. finer) scale, w_i is a weighting factor which itself may be also a property of part i (e.g. volume, mass, etc.). If this law is adopted in the conceptual model, the user is expected to select a corresponding inter-scale (software) component when the simulation script is composed. At the stage of model execution (to be detailed in Section 4), the tool used there will need to know concretely what physical quantities the variables in Eq.(1) (i.e. y, x_i, and w_i) refer to in this particular simulation in order to carry out the rather generic inter-scale operation. Those involved physical quantities will be case-specific. It is proposed to use a configuration file as input to the generic inter-scale software component to customise its operation; examples are provided in Section 5.

A general workflow for generating an executable simulation script of a multiscale system is

schematically shown in Figure 4. Note that the current design assumes the simulation involves only two different scales. This is perhaps the most commonly encountered case in multiscale modelling, although in principle extension can be made to handle more scales. Furthermore, it is considered that the coordinator driven mode most likely applies to an application involving dynamic simulation of each involved scale, whereas in the master tool driven mode the mission of a "slave" scale is to perform (quasi) steady-state simulation to provide the master tool with static quantities (e.g. certain physical properties).

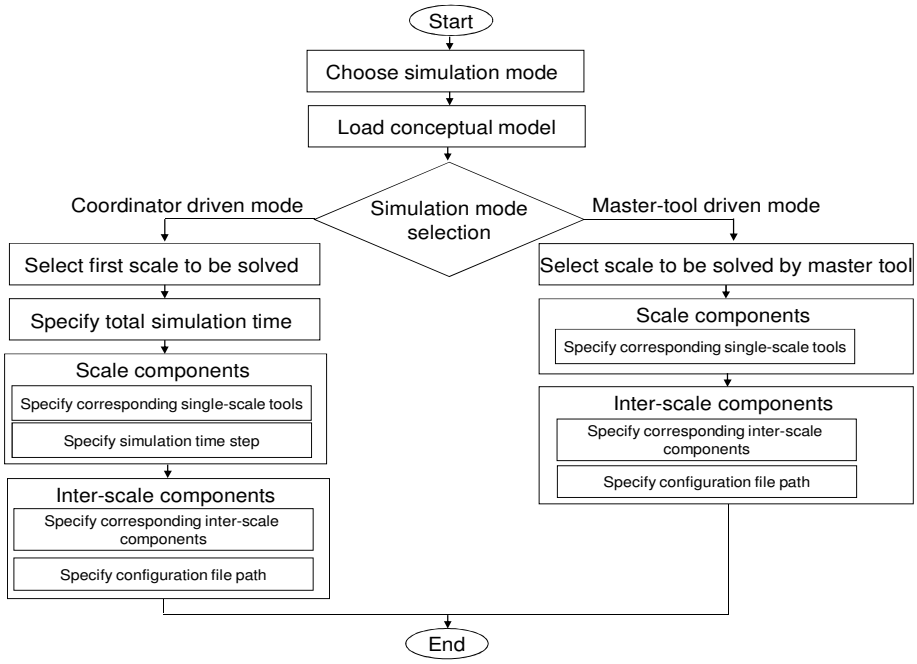


Figure 4. The workflow for model realization.

3.2 Implementation of SSG

A prototype of SSG is implemented using Java and the popular ontology processing package Jena (http://jena.sourceforge.net/ontology/). Through its GUI, the SSG presents modellers a scale structure according to the conceptual model. Based on this scale structure, the modeller can select appropriate modelling tools and software components implementing inter-scale links. S/he can also define the global simulation time and simulation step length under the coordinator driven mode.

4. Model execution

Within the three-stage strategy for CAMM, model execution as the third stage is about solving a multiscale model based on the output of the model realization step. As aforementioned in Section 3, there are two approaches to transforming a conceptual model into a form which is ready to be executed, namely automatic code generation and integration of existing tools. For the first case, the model execution stage could be implemented by applying a numerical solver to the mathematical model generated by the model realisation stage. Again, the present work focuses on the second case which in essence leads to a problem of tool integration. In this case, model execution is about executing simulation scripts generated by the model realization tool described in Section 3 to run simulations through the coordinated execution of single-scale modelling tools and inter-scale software components such as aggregators and disaggregators.

4.1 Design of the model execution system (MES)

Dealing with cases where a multiscale model is realised by tool integration, a model execution system (MES) is developed. The overall software design of MES has adopted a component-based approach to integrate software possibly developed by different vendors and even run on different computational platforms. This has been the approach of CAPE-OPEN (Braunschweig et al., 2000) and CHEOPS (Schopfer et al., 2004): A set of standard software interfaces is defined so that tools to be integrated either support the interfaces "natively" or are linked to the MES by wrappers that implement the interfaces. Similarly, standard interfaces are also defined for the "native" components of the MES, namely the coordinator and the inter-scale software components to achieve the interchangibility between different implementation options.

As mentioned in Section 3, two different simulation modes may be specified in a simulation script

to be taken as the input by the MES. Consequently, two different options for model execution are supported as illustrated in Figure 5. Note that although in Figure 5 "coordinator driven model execution" (the dashed box) appears to be part of "master tool driven model execution" (the solid box), these two options actually represent two separate and rather different ways of controlling the simulation process, as explained below.

4.1.1 Coordinator driven model execution

As illustrated by Figure 5 (the dashed box), Coordinator Driven Model Execution (CDME) makes use of a simulation coordinator to connect individual model realization components (including single-scale tools and inter-scale software components) by means of predefined interfaces. To execute a specific simulation run, the simulation coordinator starts with loading the simulation script produced by the aforementioned SSG. The coordinator then calls corresponding components in the predefined order, controls the simulation time for each of the single-scale tools, and carries out the exchange of data between different components until the total simulation time is reached. Note that the connection between "simulation coordinator interface" and "simulation coordinator" shown in Figure 5 does not exist in this case.

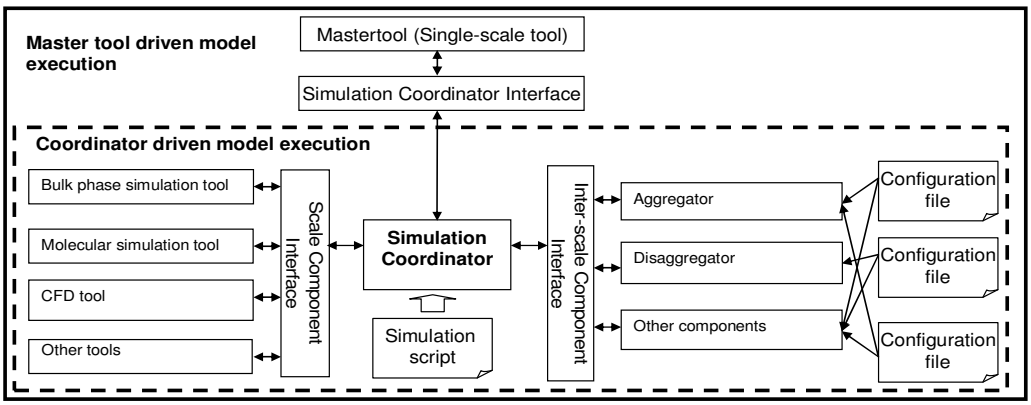


Figure 5. Design of the Model Execution System.

4.1.2 Master tool driven model execution

Figure 5 (the solid box) illustrates the structure of Master tool Driven Model Execution (MDME). Unlike CDME, MDME utilises a simulation coordinator which has a predefined interface through which the "master tool" connects to the coordinator. Since the "master tool" controls the simulation process, the coordinator only manages the data flows among other single-scale tools and the inter-scale software components. Apart from the above difference, the design of MDME is identical to that of CDME.

4.2 Implementation of MES

Like the other CAMM tools developed in this work, the MES is implemented using Java language. CORBA (http://www.corba.org/), a mature technology for developing component-based systems has been adopted to serve as the middleware between the simulation coordinator and individual modelling tools. Figure 6 shows the interface definition for scale components (i.e. those simulating individual scales), inter-scale software components (including aggregator, disaggregator and mereological connector) as well as the simulation coordinator (required by MDME) by CORBA interface definition language (IDL) (http://www.omg.org). The CORBA implementation provided by the Java 2 platform has been used.

```
module ModelExecutionSystem {
        struct TransportData {
          sequence<string> name;
          sequence<double> value;
        };
        interface Coordinator {
                void setInputData(in TransportData tdata);
                void initialize();
                void compute();
                TransportData getResult();
        };
        interface ScaleSolver {
          void setInputData(in TransportData tdata);
          void setSimulationTime(in double time);
          double getSimluationTime();
          void initialize();
          void compute();
          TransportData getResult();
        };
        interface Aggregator {
          void setInputData(in TransportData tdata);
          void setConfigurationFilePath(in string cfpath);
          void initialize();
          void aggregate();
          TransportData getResult();
        };
        interface Disaggregator {
          void setInputData(in TransportData tdata);
          void setConfigurationFilePath(in string cfpath);
          void initialize();
          void disaggregate();
          TransportData getResult();
        };
        interface MereologicalConnector {
          void setInputData(in TransportData tdata);
          void setConfigurationFilePath(in string cfpath);
          void initialize();
          void mereologicalconnect();
          TransportData getResult();};};
```

Figure 6. MES Component Interfaces defined using CORBA IDL.

5. Case studies

This section provides two case studies to illustrate the application of the CAMM tools developed in this work, particularly those of model realization and model execution with the two different simulation modes as introduced in Section 3.1.

5.1 Modelling of a homogeneous-heterogeneous reactor

In this example, a homogeneous-heterogeneous chemical reactor model is utilized to demonstrate how to construct and execute a multiscale simulation using the coordinator driven mode.

5.1.1. The modelling problem

This problem has originally been presented by Vlachos (1997) and used recently for illustrating the conceptual modelling tool as part of the three-stage strategy for CAMM (Zhao et al., 2012). It is

outlined here again to support the subsequent presentation of how this modelling problem is tackled in the model realisation and execution stages. The reactor involves a homogenous bulk fluid phase and a heterogeneous solid catalyst surface; the latter is to be characterised at the continuum level and additionally by a molecular-lattice. For the bulk fluid phase, a continuum transport model is adopted to describe the diffusion of reactants towards the surface whilst a mass conservation model incorporating the kinetics rates of the surface catalytic reaction is applied to characterize the solid surface. Moreover, the solid surface is further represented by a molecular lattice comprising a number of sites. On each site a set of adsorption, desorption or reaction phenomena may occur. Monte-Carlo (MC) method is applied to construct the non-continuum model for these three micro-processes. Therefore, this reactor can be modelled as a two scale system. Figure 7 presents the structure of this model. At Scale_0, the homogeneous bulk fluid phase connects to the macroscopic, continuum reaction surface. At Scale_1, a microscopic view is given to the reaction surface, in the form of a molecular lattice comprising a number of sites. The connection between these two scales is presented by data aggregation and disaggregation. As for the continuum surface model in Scale_0, a specific vector property, **q**, ,which is required for calculating the particular kinetics rate related to the lateral interactions of species at the surface, is determined by the occupation-site function δ_i and the local coordination lc_i for each site i of the molecular-lattice. Thus, Scale_0 is linked to Scale_1 by means of aggregation. On the other hand, for each site i of the lattice, two properties should be characterized, i.e. temperature at the site T_i (K) and the concentration of the reactant A at the fluid phase boundary layer adjacent to the surface $C_{A0,i}$ (g/L). These properties of the site are determined by the corresponding properties of the entire solid surface, namely T and C_{A0} respectively, by means of disaggregation relations.

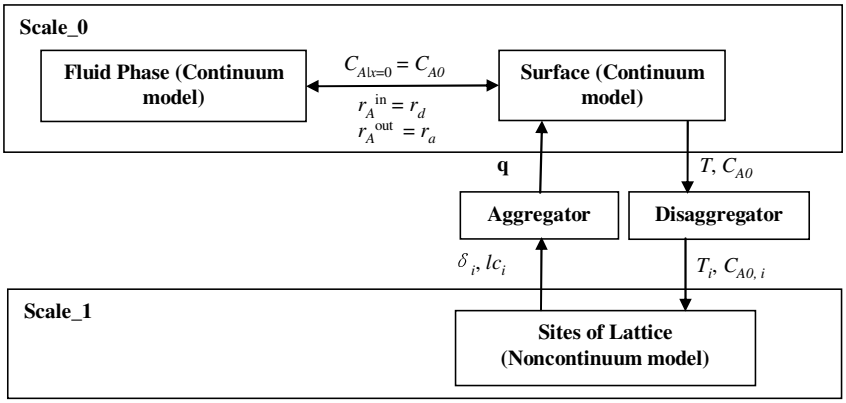


Figure 7. Structure of the homogeneous-heterogeneous reactor model.

A conceptual model of this reactor representing the aforementioned multiscale structure has been developed earlier; details of the building steps are reported in Zhao et al. (2012).

5.1.2. Model realisation

To use the SSG to transform the conceptual model into a simulation script to be executed using the coordinator driven mode, the modeller starts with specifying the simulation mode for this task. After that, the modeller loads a pre-generated conceptual model. The SSG will analyse the conceptual model and presents its scale structure, as shown in Figure 8. In the conceptual model, an instance of a particular inter-scale law as defined in the generic ontology for multiscale modelling (e.g. the averaging aggregation law) has been created for each inter-scale link specified in the conceptual model (e.g. the aggregation link for obtaining the fraction of sites of a specific coordination from the microscopic configuration).

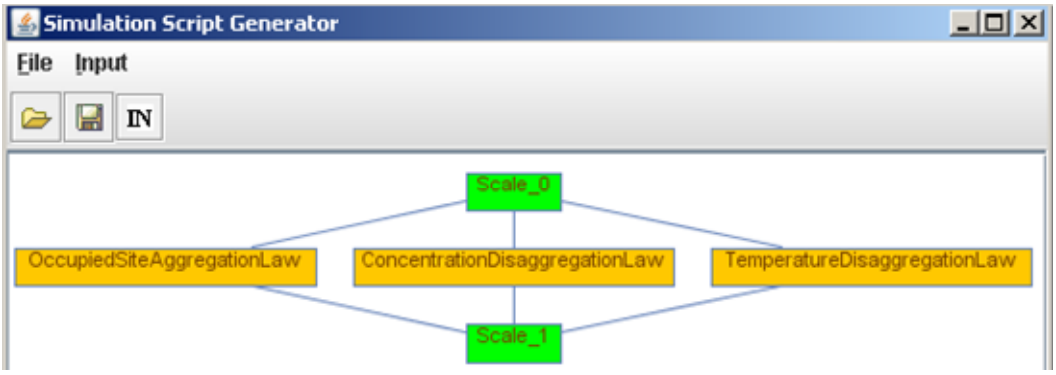


Figure 8. Scale structure of the conceptual model presented by SSG.

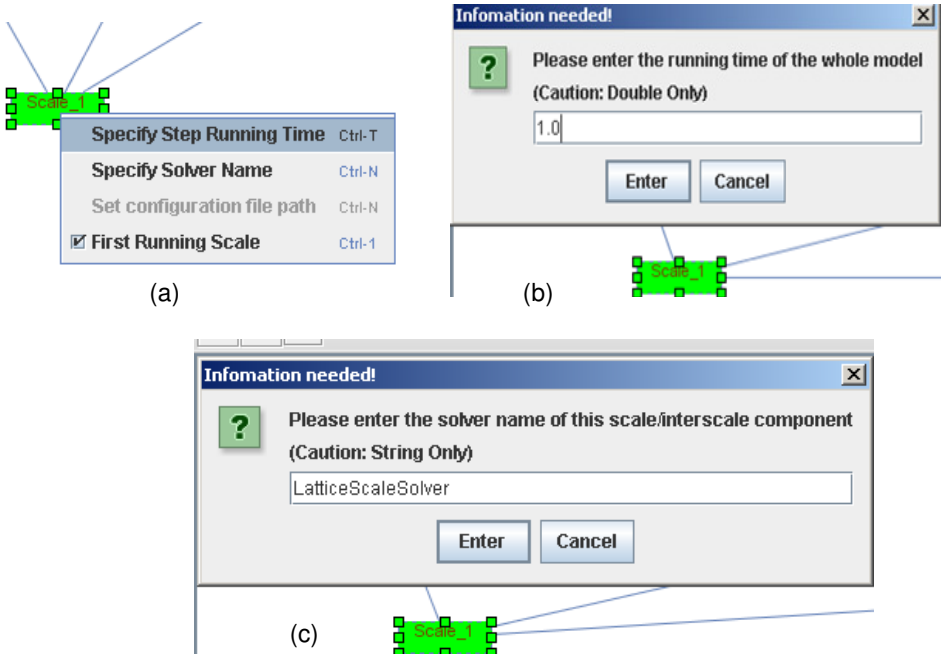


Figure 9. Screenshots of the SSG. (a) Selecting the first running scale; (b) specifying the total simulation time; (c) specifying scale solver.

After assigning the inter-scale software components, the modeller needs to choose the first running scale (cf. Figure 9a). This information is necessary since the simulation coordinator will need to decide the calling sequence of existing modelling tools. As aforementioned, the SSG currently supports only 2-scale systems which means the simulation order will be determined once the first called scale is specified. The modeller also needs to specify a total simulation time, as shown in Figure 9b, which is required by the coordinator to determine when a simulation should be terminated. Subsequently, the modeller specifies an existing single-scale modelling tool for each scale, referred to as a scale component (as opposed to an inter-scale software component). The time step size of these scale components also needs to be defined by the modeller. Figure 9c illustrates the step of specifying a scale component for “Scale_1”. The “lattice scale solver” appearing in

Figure 9c is a CORBA component prepared in this work which implements the Monte-Carlo model and offers the "scale solver" interface shown in Figure 6. In a way similar to assigning scale components, the modeller also selects a particular software component for each of the inter-scale laws specified in the conceptual model for modelling the inter-scale links involved in the modelling case. In this current example, an averaging aggregator implementing Eq.(1) (for aggregating site occupation functions to fraction of sites) and two equality disaggregators (for passing temperature and concentration) are chosen to handle the data flows between Scale_0 and Scale_1. An equality disaggregator assigns the value of a quantity at the coarser scale directly to the corresponding quantity of the relevant parts at the finer scale. To customize these generic inter-scale software components, a configuration file is created for each of them following the principle explained in Section 3.1. Figure 10a shows the file for the aggregator, which indicates that x in eq (1) refers to the site occupation function with coordination, y to fraction of sites (with specific coordination), and w to "identical weight" which is interpreted by the aggregator as "1". The aggregation law, after customisation, reads

$$q_j = \frac{1}{n}\sum_{i=1}^{n} x_{j,i},\ j = 0,1,2,3,4 \qquad (2)$$

where q_j is the jth component of the vector **q** shown in Figure 7 with j denoting a possible local coordination, n is the number of sites on the lattice. This suggests that, in this example, y and x in Eq.(1) actually become a vector and a matrix respectively. Linking to the original microscopic quantities (δ_i and lc_i) presented in Figure 7, $x_{j,i}$, referred to as "site occupation function with coordination", is a derived quantify computed by the Scale_1 model for the aggregator:

$$x_{j,i} = \begin{cases} 0, & \delta_i = 0 \vee lc_i \neq j \\ 1, & otherwise \end{cases} \qquad (3)$$

In the current implementation, a configuration file is manually created by the user. However, it is possible to generate it automatically based on the part of the conceptual model where the inter-scale connections are described (cf. Zhao et al., 2012). It should be noted that a variable from a particular scale, if involved in an inter-scale link (aggregation in the case of eq 2), is always expressed by a scalar (cf. "struct transportdata" in Figure 6). Furthermore, its name is composed of a prefix and a suffix. The prefix will be identical to the name of x or y as labelled in the configuration file, while the suffix contains the index information if this variable is actually an element of a vector or a matrix. The aggregator will then be able to establish the correspondence between scalar(s) forming y and those forming x based on the prefixes and the index information extracted from the suffixes.

Finally, a simulation script is generated, as illustrated in Figure 10b. The first line of the script records the total simulation time to be executed. Note that this line would not be required in the case of MDME. Each of the other lines has four terms. The first term demotes that this is a line about a (scale or inter-scale) software component. The second term of the line denotes the type of a software component: "Scale" denotes a single-scale solver; "Aggregator", "Disaggregator" and "MereologicalConnector" are used to indicate different types of inter-scale software components. The third term is the name of the specific scale/inter-scale software components chosen by the user. For example, in the third line of Figure 10b, this term is "averaging aggregator". The forth and final term has multiple possible usages. It denotes the simulation step size for a scale component if the component type is "Scale" (not applicable in the master tool driven mode) or the path of a configuration file if it is an inter-scale software component.

```
x SiteOccupationFunctionWithCoordination
w IdenticalWeight
y FractionOfSite
```

(a)

```
Totaltime 1.0
Component Scale LatticeScaleSolver 1.0E-5
Component Aggregator AveragingAggregator D:/ModelScriptGenerator/Case1/cf1.txt
Component Scale FluidScaleSolver 1.0E-5
Component Disaggregator EqualityDisaggregator D:/ModelScriptGenerator/Case1/cf2.txt
Component Disaggregator EqualityDisaggregator D:/ModelScriptGenerator/Case1/cf3.txt
```

(b)

Figure 10. Input files generated for model execution of Case Study 1. (a) Configuration file for the aggregator; (b) Simulation Script.

5.1.3. Model execution

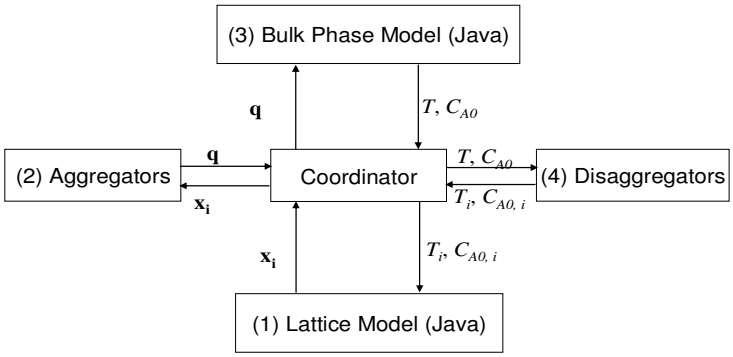


Figure 11. Simulation structure of the homogeneous-heterogeneous reactor model.

As the simulation is to be conducted in the coordinator-drive mode, the CDME option of the MES (cf. section 4.1.1) is applied. Recall that the MES contains a coordinator (implemented as a CORBA client) as well as inter-scale software components (implemented as CORBA server components following the respective interfaces defined in Figure 6). As for the scale components, in addition to the "lattice scale solver" already mentioned above, a continuum model for Scale_0 was implemented which supports the same software interface, "ScaleSolver" (cf. Figure 6). Figure 11 shows the computational procedure of this case, as an instance of the generic simulation structure enforced by CDME. The coordinator begins with loading the simulation script. It analyzes the script

and extracts the total simulation time. After that, the coordinator calls the named scale/inter-scale solvers according to the order specified in the simulation script. Specifically, the coordinator first calls the scale solver for “Lattice Model” since “Scale_1” is selected as the first running scale. The coordinator runs the “LatticeScaleSolver” until the specified time step size is reached. The generated status of sites is then returned to the coordinator which in turn forwards these data to the averaging aggregator to compute the value which is to be subsequently used by the “Bulk Phase Model”. The coordinator then calls the solver for “Bulk Phase Model” and then passes the generated result to the equality disaggregator which is responsible for transforming these data into a form which can be recognized by “LatticeScaleSolver”. The coordinator repeats calling these solvers and passes the data among them until the simulation is completed.

The result of the simulation conducted using the MES is shown by Figure 12. In this figure, plot “a” presents trajectory of the concentration of reactant A at the fluid phase boundary layer adjacent to the surface; plot “b” displays the trend of q_i ($0 \leqslant i \leqslant 4$) which denotes the fraction of the occupied sites with a coordination i. In this case study, it had been possible to use an alternative implementation of the multiscale simulation with a single C++ code to enable a comparison with the simulation realised by the proposed approach, i.e. via the MES. In terms of the simulation results, virtually no difference was observed between these two approaches, which offer an initial validation of the mechanism of the MES. In terms of computational time, the two simulations were also comparable with each other in this particular case. Generally speaking, tools integration based simulation runs can be more time consuming than those realised by a single modelling tool due to the computational overhead associated with the integration. However, integration of multiple tools should be viewed as a means of realizing a multiscale model which would otherwise be impractical

to develop; in that case the computational burdens would become a secondary concern.

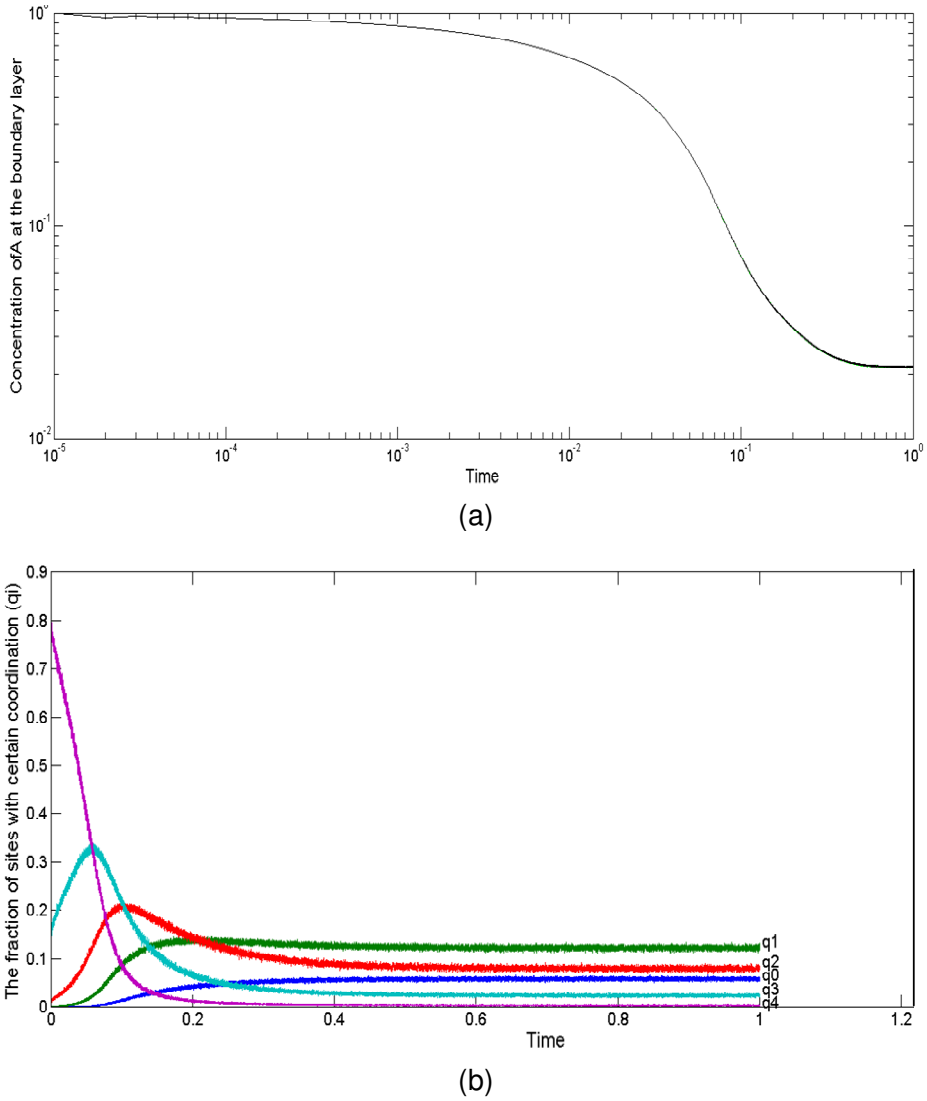


Figure 12. Simulation results of the homogeneous-heterogeneous reactor model. (a) reactant concentration; (b) fraction of sites with specific coordination, as aggregated information.

5.2 Modelling of a stirred bio-reactor

The second example deals with the modelling of a stirred bio-reactor model by the combined multizonal/computational fluid dynamics (CFD) approach originally presented by Bezzo et al. (2003). In contrast to the first example, the *master tool driven* mode is adopted in this second example.

5.2.1. The modelling problem and the conceptual model

The bio-reactor model comprises six perfectly mixed internal zones which are separated according to the flow pattern in the reactor. The dynamic behaviour of these zones is represented by a multizonal model. These zones are coupled by the mass flows between them. Each of these zones is further modelled by CFD which captures the fluid-mechanical phenomena with a large number of cells. As such, the whole reactor can be modelled by two scales, the first scale describes the characteristics of the perfectly mixed zones as well as their interactions, whilst the detailed fluid mechanics is modelled via the CFD cells at the second scale (cf. Figure 13).

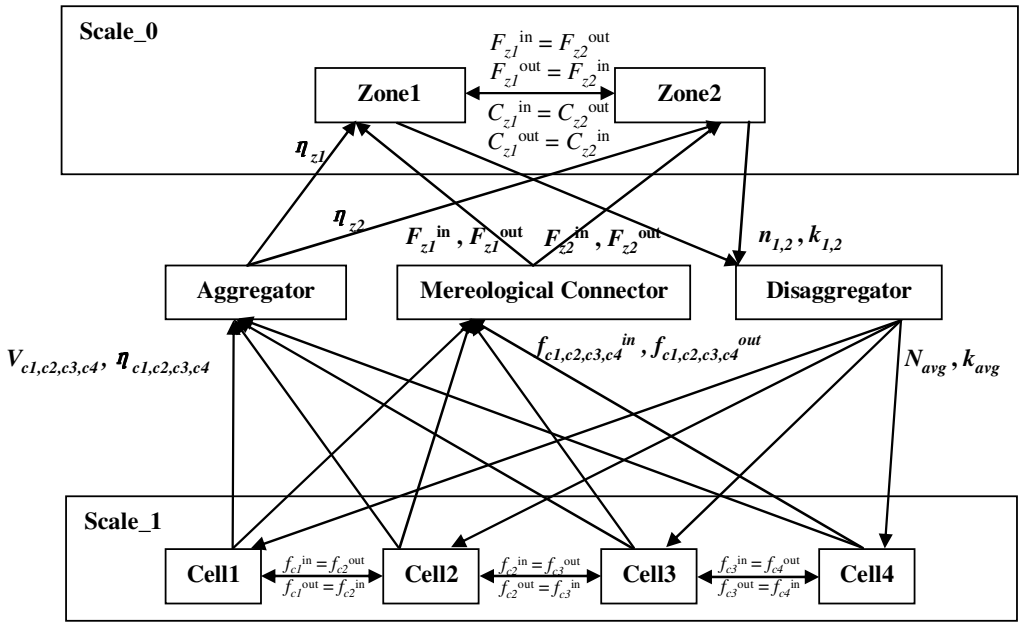


Figure 13. Multiscale structure of the bio-reactor model.

At scale_0, the perfectly mixed zones (represented schematically by Zone1 and Zone2) connect to their adjacent neighbours. The inlet and outlet mass flow rates of zone *i* are denoted as F_{zi}^{in} and F_{zi}^{out} (kg/s) and their inlet and outlet concentrations are denoted as C_{zi}^{in} and C_{zi}^{out} (g/L). At Scale_1, a microscopic view is given to each zone by means of a number of cells which are inter-connected. f_{cj}^{in} and f_{cj}^{out} (kg/s) represent the inlet and outlet mass flow rates of a given cell *j* respectively. The

connection between these two scales comprises data aggregation, mereological connection and disaggregation. In Scale_0, the effective viscosity, η_{zi} (Pa•s), of zone *i* is required to compute the mass-transfer coefficient which is determined by the viscosity, η_{cj} (Pa•s), of the individual cells that belong to this zone. In addition to η_{cj}, the volume of zone (V_{zi} (L)) and that of each of cells belonging to it (V_{cj} (L)) are needed to compute the effective viscosity for zone *i* through aggregation. The mass flow rates between the zones, F_{zi}^{in} and F_{zi}^{out}, are required for the mass balances at Scale_0, and their values directly relate to those of the mass flow rates between the cells located at the boundary of the neighbouring zones (f_{cj}^{in} and f_{cj}^{out}). Mereological connection is one type of inter-scale links defined in the generic ontology to denote the relationship between the couplings at one scale and those at a neighbouring scale. The software implementation of a mereological connection is called a "mereological connector" (cf. Figure 13) and is adopted here to carry out the inter-scale mapping of mass flows. For each cell at Scale_1, coefficients including the consistency index (denoted as *k)* and the flow behavior index (denoted as *n*) are needed to compute viscosity η_{cj}. These two quantities are provided by the model of Scale_0. Therefore, disaggregation relations are established between these scales to pass these two quantities from the zones at Scale_0 to the cells at Scale_1. The conceptual model of this bio-reactor is constructed to capture the above multiscale structure, following the steps reported in Zhao et al. (2012). Figure 14 shows the screenshot of the graphical representation of this conceptual model as displayed in the conceptual modelling tool.

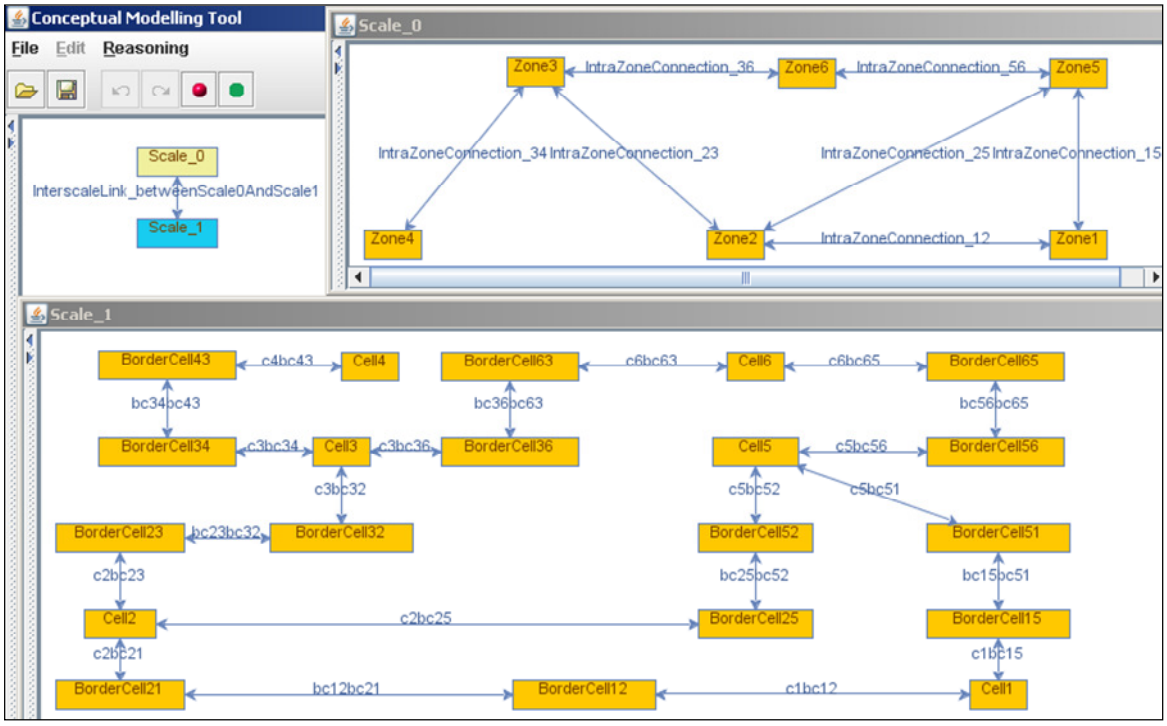


Figure 14. Screenshot of conceptual model of the bio-reactor.

5.2.2. Model realisation

Same as the previous example, the first step of generating the simulation script is to specify the simulation mode. In this example, the multiscale simulation will be driven by the multi-zone model in gPROMS (Process Systems Enterprise, 2001), which requires the CFD model in Fluent (ANSYS, 2006) to be executed to provide viscosity and inter-zone flow rates when the multizone model is being solved. In other words, when Fluent is to be called is completely determined by gPROMS. Therefore, it is a typical case of master tool driven simulation. After the simulation mode is determined, the conceptual model of the bio-reactor is loaded and its scale structure is presented. The modeller then needs to select a scale to be solved by the "master tool", which in this case is Scale_0. The modeller also needs to specify software components for implementing each scale/inter-scale software component and to set configuration file paths for the inter-scale software components. For the inter-scale software components, the averaging aggregator and the equality

disaggregator adopted in the former case are also selected in this case but with different configuration files. As an example comparable to that created for the previous case (cf. Figure 10a), the configuration file for customising the averaging aggregator is considered (cf. Figure 15a). In this case, the x_i variable in Eq.(1) is substituted by the value of the viscosity of each CFD cell belonging to a certain zone, w_i is substituted by the value of the volume of each cell. The output of the aggregator, y, refers to the effective viscosity of the zone these cells belong to. As such the aggregation essentially provides volume-averaged viscosity. A new inter-scale software component exclusively involved in this case is a mereological connector which implements the mereological connection mentioned earlier in Section 5.2.1. Since the execution stage is fully controlled by the "master tool", there is no need to specify the total simulation time or the time step for each scale component. The produced simulation script, of the structure explained earlier in Section 5.1.2, is shown in Figure 15b.

```
x ViscosityOfCell
w VolumeOfCell
y EffectiveViscosityOfZone
```

(a)

```
Component Disaggregator EqualityDisaggregator D:/ModelScriptGenerator/Case2/cf3.txt
Component Disaggregator EqualityDisaggregator D:/ModelScriptGenerator/Case2/cf4.txt
Component Scale Fluent
Component Aggregator AveragingAggregator D:/ModelScriptGenerator/Case2/cf1.txt
Component MereologicalConnector EqualityMereologicalConnector D:/ModelScriptGenerator/Case2/cf2.txt
```

(b)

Figure 15. Input files generated for model execution of Case Study 2. (a) Configuration file for the aggregator; (b) Simulation Script.

5.2.3. Model execution

In this example, the simulation mode has been set as "master tool driven", so the MDME option described in Section 4.1.2 is used to execute the simulation. Recall that the coordinator for MDME not only plays the role of managing the interactions of the scale/inter-scale software components, it

also acts as a server to response to the requests from the "master tool." The multizonal model for Scale_0 was implemented by an existing modelling tool gPROMS and the CFD model for Scale_1 were implemented by Fluent. gPROMS is taken as the master tool, while the CFD model is connected to the MES by a wrapper that implements the "ScaleSolver" interface (cf. Figure 6). The simulation structure is illustrated by Figure 16. The "master tool" gPROMS triggers the simulation and calls for the coordinator when it needs data from the CFD model. The coordinator then calls the disaggregator to pass the values of *n* and *k* received from gPROMS to Fluent via the coordinator. Fluent subsequently computes the quantities of each cell and returns their values to the coordinator. The coordinator in turn forwards them to the aggregator and the mereological connector. The former component computes the effective viscosity of each zone while the latter provides mass inlet and outlet flow rates at the boundary of these zones. This procedure will continue until gPROMS completes the simulation. Figure 17 finally shows the trend of the product concentration as part of the results produced by gPROMS in co-operation with the coordinator and other components in the MES. In this example, it was not practical to compare the simulation result with that of an implementation of the multiscale model by a single tool (as was possible in the first example). However, the outcome has proved the feasibility of the tool integration based approach.

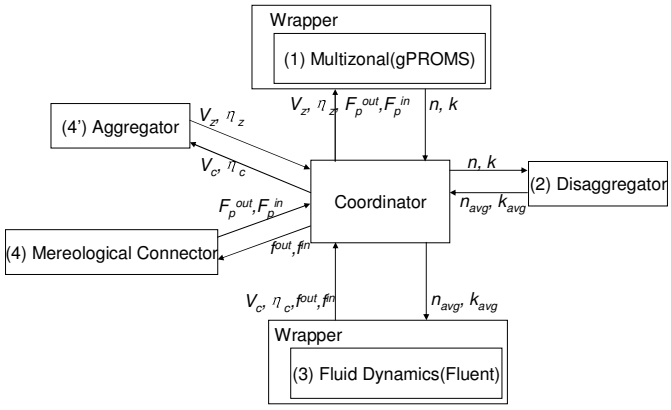


Figure 16. Simulation structure of the bio-reactor model.

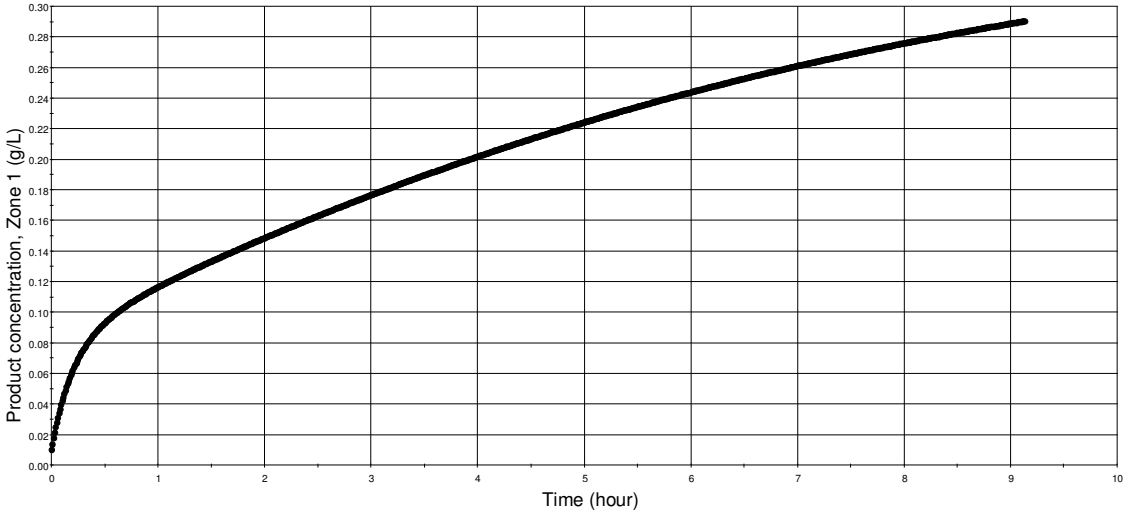


Figure 17. Simulation result of the bio-reactor model.

6. Conclusions

Based on the previously proposed three-stage strategy for CAMM, this work developed proof-of-concept tools for the model realization and execution steps to deal with cases where multiscale simulation is to be carried out by integrating existing single-scale simulators. For model realization, a tool called Simulation Script Generator (SSG) is prototyped, which is able to generate two different types of simulation scripts according to the simulation mode (master tool-driven or coordinator driven) selected for a given application. A Model Execution System (MES), supporting model execution through integrating existing modelling tools, takes a script as input and carries out the simulation accordingly. Two options of the MES were developed, each using a simulation coordinator suitable for one of the two different simulation modes. The application of these tools has been demonstrated by two examples of reactor modelling.

Together with what was reported earlier with regard to the overall methodology for CAMM and the conceptual modelling tool (Zhao et al., 2012), the experience gained in this study gives a preliminary indication that it is feasible to develop generic CAMM tools without assuming

case-specific details. This is enabled fundamentally by building these tools on a common conceptualisation of multiscale systems (particularly in a form of a generic ontology) as the basis for implementing the "general logic" for multiscale modelling. This approach has the potential to allow for the separation of the general logic from case-specific information and knowledge; the latter can be treated as the case-specific, changeable input to CAMM tools. This has been the case of a domain ontology, defined on top of the generic ontology for multiscale systems and used as the input to the generic conceptual modelling tool (Zhao et al., 2012). Similarly as reported in this current paper, a simulation script, generated using the Simulation Script Generator, is taken as the input to a generic Model Execution System in order to perform a specific simulation run.

This initiative as a whole has suggested positively that the concepts and methods developed in computer aided process modelling (e.g. phenomena-based modelling, use of ontologies, component-based approach) can be extended to support multiscale modelling. Following the framework developed in this work, future work may refine and extend the software interfaces defined for scale and inter-scale software components towards a standard. Furthermore, inter-scale software components implementing more advanced methods may be introduced for aggregation and disaggregation, with an aim to improve the numerical performance of multiscale simulation.

Acknowledgement

This work is supported by the UK Engineering and Physical Science Research Council (EPSRC) under Grant No EP/G008361/1.

References

Bezzo, F., Macchietto, S., & Pantelides, C. C. (2003). A general hybrid multizonal/CFD approach for bioreactor modelling. *AIChE Journal*, 49, 2133-2148.

Bezzo, F., Macchietto, S., & Pantelides, C. C. (2004). A general methodology for hybrid multizonal/CFD models. Part I. Theoretical framework. *Computers & Chemical Engineering*, 28, 501-511.

Braatz, R. D., Alkire, R. C., Rusli, E., & Drews, T. O. (2004). Multiscale systems engineering with applications to chemical reaction processes. *Chemical Engineering Science*, 59, 5623-5628.

Braunschweig, B. L., Pantelides, C. C., Britt, H. I., & Sama, S. (2000). Process modelling: The promise of open software architectures. *Chemical Engineering Progress* , 96, 65-76.

Charpentier, J. C. (2002). The triplet "molecular processes–product–process" engineering: the future of chemical engineering? *Chemical Engineering Science*, 57, 4667-4690.

Charpentier, J. C. (2009). Perspective on multiscale methodology for product design and engineering. *Computers & Chemical Engineering*, 33,936-946.

Hetherington, J., Bogle, I. D. L., & Saffrey, P. (2007) Addressing the challenges of multiscale model management in systems biology. *Computers & Chemical Engineering*, 31, 962-979.

Hunter, P. J., Burrowes, K., Fernandez, J., Nielsen, P., Smith, N., & Tawhai, M. (2005). The IUPS Physiome Project: Progress and Plans. In: Kriete, A., Eils, R., editors. Chapter 17 in Computational Systems Biology. Oxford: Elsevier, 383-394.

Ingram, G. D., Cameron, I. T., & Hangos, K. M. (2004). Classi cation and analysis of integrating frameworks in multiscale modelling. *Chemical Engineering Science*, 59, 2171-2187.

Jaworski, Z. & Zakrzewska, B. (2011). Towards multiscale modelling in product engineering. *Computers & Chemical Engineering*, 35, 434-445.

Karakasidis, T. E., & Charitidis, C. (2007). A. Multiscale modelling in nanomaterials science. *Materials Science and Engineering*, 27, 1082-1089.

Kougoulos, E., Jones, A.G., & Wood-Kaczmar, M. (2005) CFD modelling of mixing and heat transfer in batch cooling crystallizers: Aiding the development of a hybrid predictive compartmental model. *Chemical Engineering Research and Design*, 83, 30-39.

Kulikov, V., Briesen, H., Grosch, R., Yang, A. Wedel, L. & Marquardt, W. (2005). Modular dynamic simulation of integrated process flowsheets by means of tool integration. *Chemical Engineering Science*, 60, 2069-2083.

Kulikov, V., Briesen, H., & Marquardt, W. (2005). Scale integration for the coupled simulation of crystallization and fluid dynamics. *Chemical Engineering Research and Design*, 83, 706-717.

Li, J., Ge, W., Zhang, E. J., & Kwauk, M. (2005). Multi-scale compromise and multi-level correlation in complex systems. *Chemical Engineering Research and Design*, 83, 574-582.

Liu, A., Rugonyi, S., Pentecost, J. O., Thornburg, K. L. (2007). Finite element modelling of blood flow-induced mechanical forces in the outflow tract of chick embryonic hearts. *Computers & Structures*, 85, 727-738.

Lucia, A. (2010). Multi-scale methods and complex processes: A survey and look ahead. *Computers & Chemical Engineering*, 34, 1467-1475.

Maroudas, D. (2000). Multiscale modelling of hard materials: Challenges and opportunities for chemical engineering. *AIChE Journal*, 46, 878-882.

Marquardt, W., Morbach, J., Wiesner, A. & Yang, A. (2010). *OntoCAPE: A Re-Usable Ontology for Chemical Process Engineering*, Berlin:Springer.

Morbach, J., Yang, A., & Marquardt, W. (2007). OntoCAPE - a large-scale ontology for chemical process engineering. *Engineering Applications of Artificial Intelligence*, 20, 147-161.

Morales-Rodríguez, R., & Gani, R. (2007). Computer-aided multiscale modelling for chemical process engineering. *Computer Aided Chemical Engineering*, 24, 207-212.

O'Connell, S. T., & Thompson, P. A. (1995). Molecular dynamics–continuum hybrid computations: A tool for studying complex fluid flows. *Physical Review E*, 52, 5792-5795.

Pantelides, C. C. (2001). New challenges and opportunities for process modelling. *Computer Aided Chemical Engineering*, 9, 15-26.

Scharwaechter, H. , Wedel, L. v. , Yang, A., Marquardt, W. (2002). A Tool Integration Framework for Dynamic Simulation in Process Engineering. *15th IFAC World Congress on Automatic Control*, Barcelona, 21-26.7.2002.

Schopfer, G., Yang, A., Wedel, L. v., Marquardt, W. (2004). CHEOPS: A Tool-Integration Platform for Chemical Process Modelling and Simulation. *International J. on Software Tools for Technology Transfer*, **6**, 186-202.

Senin, N., Wallace, D. R., Borland, N. (2003). Distributed Object-Based Modeling in Design Simulation Marketplace. *ASME Journal of Mechanical Design*, 125, 2-13.

Smirnov, A. V. (2004). Multi-physics modelling environment for continuum and discrete dynamics. *International Journal of Modelling and Simulation*, 24, 190-197.

Southern, J., Pitt-Francis, J., Whiteley, J., Stokeley, D., Kobashi, H., Nobes, R., et al. (2008). Multi-scale computational modelling in biology and physiology. *Progress in Biophysics and Molecular Biology*, 96, 60-89.

Vlachos, D. G. (1997). Multiscale integration hybrid algorithms for homogeneous–heterogeneous reactors. *AIChE Journal*, 43, 3031-3041.

Vlachos, D. G. (2005). A review of multiscale analysis: Examples from systems biology, materials engineering, and other fluid-surface interacting systems. *Advanves in Chemical Engineering*, 30, 1-61.

Wang, L. & Wong, N.H. (2009). Coupled simulations for naturally ventilated rooms between building simulation (BS) and computational fluid dynamics (CFD) for better prediction of indoor thermal environment. *Building and Environment*, 44, 95-112.

Yang, A., Morbach, J., & Marquardt, W. (2004). From conceptualization to model generation: the roles of ontologies in process modelling. In: Floudas CA, Agrarwal R , editors. Proceedings of FOCAPD 2004; 2004 July 11-16; New Jersey, USA; New York: American Institute of Chemical Engineers, 591-594.

Yang, A., & Marquardt, W. (2009). An ontological conceptualization of multiscale models. *Computers & Chemical Engineering*, 33, 822-837.

Yang, A., & Zhao, Y. (2009). From a generic idea to a generic tool set: exploring computer-aided multiscale modelling. International Symposium on Process Systems Engineering, August 2009, Brazil.

Zhao, Y., Jiang, C., & Yang, A. (2012). Towards computer-aided multiscale modelling: an overarching methodology and support of conceptual modelling. *Computers & Chemical Engineering*, 36, 10-21.

Zitney, S. E. (2010). Process/equipment co-simulation for design and analysis of advanced energy systems. *Computers & Chemical Engineering*, 34, 1532-1542.