TOWARDS A GENERIC SUPPORTING ENVIRONMENT FOR MULTISCALE MODELLING

Yang Zhao, Cheng Jiang, Aidong Yang
Chemical and Process Engineering
Faculty of Engineering and Physical Sciences
University of Surrey
Guildford, UK
E-mail: {y.zhao, c.jiang, a.yang}@surrey.ac.uk

KEYWORDS

multiscale modeling, conceptual modeling, tool integration, computer-aided modelling

ABSTRACT

Multiscale modelling as an emerging modelling paradigm is now widely regarded as a promising and powerful tool in various disciplines. However, a multiscale model is usually much more difficult to develop than a single-scale model due to a range of challenges. This work presents a methodology to facilitate the development of multiscale models, which comprises three main modelling steps, namely conceptual modelling, model realization and model execution. A set of proof-of-concept tools have been developed to realize the proposed methodology. A case study on the modelling of a heterogeneous chemical reactor is presented to demonstrate these tools and to illustrate the key concepts.

INTRODUCTION

Multiscale modelling as an emerging modelling paradigm is now widely regarded as a promising and powerful tool in various disciplines, including process engineering (Charpentier, 2002; Braatz et al., 2004; Vlachos, 2005), material science (Karaksidis and Charitidis, 2007), computational mechanics (Liu et al., 2007) and many other areas. Through combining the models of different resolution scales of a complex system, multiscale modelling is able to provide higher quality characterization or higher computational efficiency than traditional (single-scale) modelling. Compare to developing single-scale model, multiscale model is usually much more difficult to construct due to a range of challenges (Yang and Zhao, 2009).

To tackle the difficulties, several efforts have been dedicated to facilitate multiscale modelling, especially in the field of modelling methodology formulation which aims to provide certain generic guidance to the construction of multiscale models. One of the most notable activities is the classification of multiscale modelling approaches, which defines the generic ways of multiscale model development by suggesting what scales should be introduced in a multiscale model and how their connections should be established (Pantelides, 2001; Ingram et al., 2004; Vlachos, 2005). Although such general conceptual guidance assists modellers to some extent, what can provide more substantial support is computer based tools (Marquardt et al., 2000; Ingram et al., 2004). However, progress in this direction has been very insignificant so far; the application of multiscale modelling is thus still remaining as a special privilege of highly skilled modelling experts (Fraga et al., 2006).

This paper presents explorative research in an ongoing project which is aimed to explore the way to a computer-based, generic and open supporting framework for multiscale modelling. This paper outlines an overarching methodology for developing a support environment for Computer-aided Multiscale Modelling (CAMM) and reports the implementation details of its three successive modelling steps. The rest of the paper

Proceedings 25th European Conference on Modelling and Simulation ©ECMS Tadeusz Burczynski, Joanna Kolodziej Aleksander Byrski, Marco Carvalho (Editors)
ISBN: 978-0-9564944-2-9 / ISBN: 978-0-9564944-3-6 (CD)

is structured as follows. In Section 2, an overall methodology of CAMM will be briefly reviewed. Section 3 presents the design and development of a conceptual modelling tool. The implementation details of model realization and execution will be given in Section 4 and Section 5. A case study which is used to validate these tools is shown in Section 6.

OVERALL METHODOLOGY

There are two sets of challenging issues which must be addressed in the process of marching towards CAMM. The first set of challenges consists of conceptual, numerical and software issues while the second set involves the maximisation of computer-based support as well as the generality of the CAMM tools. In this section, a methodology comprising two important components is proposed for addressing the above issues. (cf. Fig. 1)

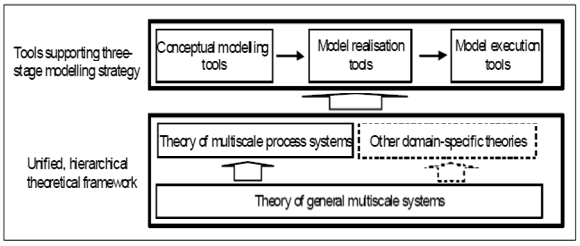

Figure 1: A Methodology for CAMM

The first component is to establish a theoretical framework of multiscale systems as the basis for CAMM. To achieve the maximisation of the generality that a generic solution of CAMM should support, a unified theoretical framework is needed. This extends the method proposed by Yang et al. (2004) for general systems modelling. More specifically, this framework should be constructed by a hierarchical structure which consists of two levels, namely a fundamental level and a domain-specific level. The fundamental level should provide a unified theory for general multiscale systems, allowing the formulation of explicit and rigorous definitions of fundamental concepts in multiscale modelling. A generic ontology (expressed using OWL, http://www.w3.org/TR/owl-ref/) based on the conceptualization proposed by Yang and Marquardt (2009) has been developed to play the role of the fundamental level. Domain-specific level is on top of the fundamental level, which aims at providing domain specific conceptualization. For particular applications, domain specific concepts can be achieved either by concretising the generic ontology or reusing the existing theories with modifications to conform to the fundamental concepts. Part of OntoCAPE (Morbach et al., 2007) is reused as the domain ontology for the case study to be reported later in this paper. It is expected that, by building upon this unified, hierarchical theoretical framework, the generality of the CAMM solutions can be maximised.

The second component of the proposed methodology is to apply a three-stage strategy to maximise computer-based support. Motivated by the need of addressing different types of modelling issues while maximising computer-based support to modellers, it would be helpful to separate the most creative modelling tasks such as those of conceptual modelling, which inevitably require the input/intervention from the modellers, from other tasks such as those of model realization and model execution which can be handled more "automatically" by software tools. For a given modelling problem, the conceptual modelling stage generates a conceptual model which dictates ("by words") what scales to be included in the model and what kind of linkages should be established between these scales. Following this stage, the model realization and execution stages deal with (i) the realization of the conceptual model (in a form which is ready to execute) and (ii) the actual execution of model, respectively. The implementation details of the tools facilitating these modelling steps will be given in the following sections

CONCEPTUAL MODELLING

As aforementioned, the main task of conceptual modelling is to describe what a multiscale model should contain in terms of the scales involved, the composition of each scale, and the connections between different scales. Both modeller's creative effort and computer-based supports are needed to construct conceptual models. A computer-aided conceptual modelling tool (CCMT) has been designed to facilitate modellers for completing this purpose.

As shown by Figure 2, the tool assumes the conceptual

description of any multiscale system will conform at the abstract level to what is defined in the generic ontology. On the other hand, it is the domain-specific concepts that should be directly used for constructing a specific conceptual model for an application of a particular domain. This requirement is met by introducing a domain ontology as a specialisation of the generic ontology. Within the CCMT, three types of functions are provided through a graphical user interface (GUI) and by leveraging generic tools for processing ontologies and rules. A detailed introduction of these functions can be found in Zhao et al. (2010).

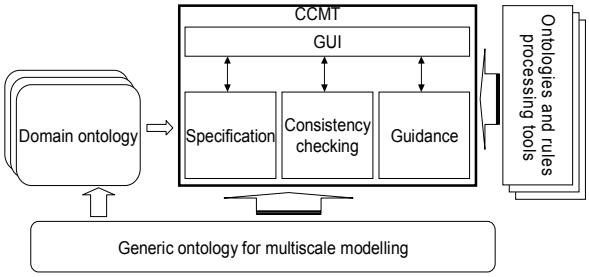

Figure 2: Design of the CCMT

A prototype of the CCMT has been implemented using Java and based on Jena a widely used OWL ontology processor (http://jena.sourceforge.net/ontology/). Pellet (http://clarkparsia.com/pellet/) as the reasoner for consistency checking and for modelling paradigm classification is embedded into the CCMT. The generic ontology has been constructed using Protégé (http://protege.stanford.edu/) with the combination of the general multiscale system concepts and SWRL rules (http://www.w3.org/Submission/SWRL/) which are adopted to strengthen the expressiveness of OWL to define complex concepts.

MODEL REALIZATION

Following the overarching methodology presented earlier, model realization is responsible for translating the conceptual model into a form which is ready to be executed. The general way of realizing conceptual model 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 realize a multiscale simulation. However, the applicability of this approach is limited given the fact 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, each simulating one particular scale of a multiscale system.

To support the tool integration based approach to model realization, a simulation script generator (SSG) is developed in this work for constructing multiscale simulation based on the combination of several existing single-scale modelling tools, such as gPROMS, Fluent and Aspen Plus. As shown by Figure 3, the generic ontology provides the theoretical basis for SSG which analyzes conceptual models and generates simulation scripts. The simulation scripts contain the modeller specified single-scale tools and inter-scale components to realize inter-scale links as well as the running time of each of these tools. All these tools/components in this list are arranged in a modeller specified order to from the execution sequence of a multiscale model. SSG possesses two functions, namely running time specifier and scale/inter-scale component solver specifier. The former supports specifying the time span of the model execution stage as well as that of each single-scale modelling tools while the later allows for designating appropriate solvers for each scale/inter-scale components for a particular modelling task. So far a prototypical SSG has been developed by Java and Jena.

MODEL EXECUTION

In general, model execution is responsible for solving a multiscale model based on the simulation scripts generated by the model realization tool. For situations where a multiscale model is realized by integrating existing tools, a model execution system (MES) is proposed.

As illustrated by Figure 4, the core of MES is a simulation coordinator which connects individual model realization components, comprising single-scale tools and inter-scale components, through predefined interfaces. To execute a specific simulation, the

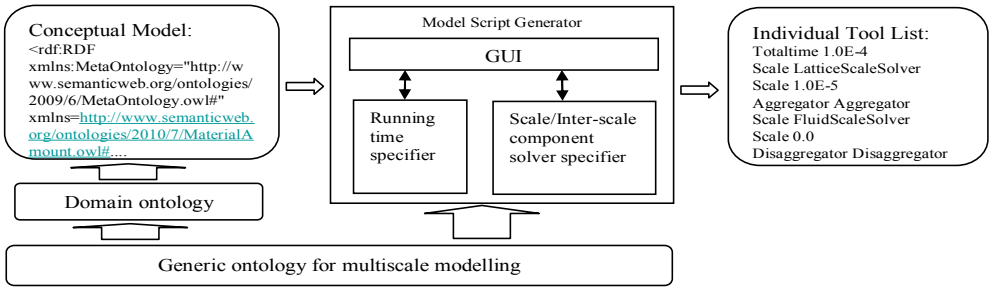

Figure 3: Design of the Simulation Script Generator (SSG)

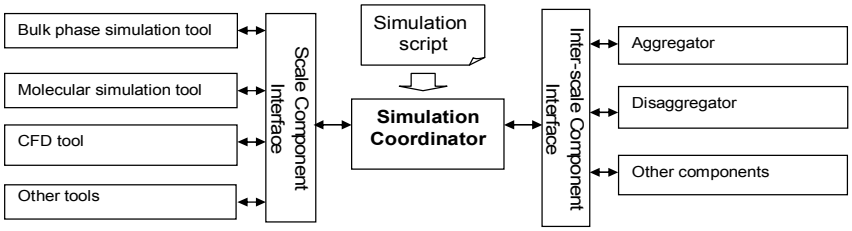

Figure 4: Design of the Model Execution System (MES).

simulation coordinator reads the simulation script produced by SSG, calls modeller named components one by one according to the sequence specified by the script, and carries out the exchange of data between them until the simulation task is completed. As for the integration of existing modelling tools, the component-based philosophy similar to that of CAPE-OPEN (Braunschweig et al., 2000) and CHEOPS (Schopfer et al., 2004) is followed: a set of standard software interfaces is to be provided so that the tools to be integrated either support the interfaces "natively" or are linked to the MES by wrappers that implement the interfaces.

Like the other CAMM tools developed in this project, the MES is being 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 5 shows the interface for scale components (i.e. those simulating individual scales) and inter-scale components (including aggregator and disaggregator) by CORBA interface definition language (IDL).

CASE STUDY

A homogeneous-heterogeneous chemical reactor (Vlachos, 1997) is used to validate the aforementioned tools. The reactor to be modelled comprises a homogenous bulk fluid phase and a heterogeneous solid catalyst surface. In 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. Additionally, 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. The Monte-Carlo (MC) method is involved to construct the non-continuum model for these three micro-processes.

To use the CCMT to construct such a conceptual model, the modeller starts with loading the domain ontology. After that, the modeller represents the intended scale structure by creating two scale instances (bulk fluid scale & molecular-lattice scale) and linking them with an instance of inter-scale link. When the overall scale structure is created, the modeller continues to specify each scale involved as well as the link between the two scales. Figure 6 shows a screen snapshot at the step where the components at molecular-lattice scale are

specified.

```
module SolverComponent {
            struct Data {
               sequence<string> name;
               sequence<double> value;};
            interface ScaleSolver {
               void resolveInitialData(in Data data);
               void setComputationTime(in double time);
               double getComputationTime();
               void initialize();
               void compute();
               Data getResult();
               Data getUnknownData();
               void setUnknownData(in Data data);};
            interface Aggregator {
               void getUpperScaleParameter(in Data data);
               void getLowerScaleParameter(in Data data);
               void setComponentNumberOfLowerScale(in double number);
               void aggregate();
               Data getResult();};
            interface Disaggregator {
               void getUpperScaleParameter(in Data data);
               void getLowerScaleParameter(in Data data);
               void setComponentNumberOfLowerScale(in double number);
               void disaggregate();
               Data getResult();};};
```

Figure 5: Scale/Inter-scale Component Interface by CORBA IDL.

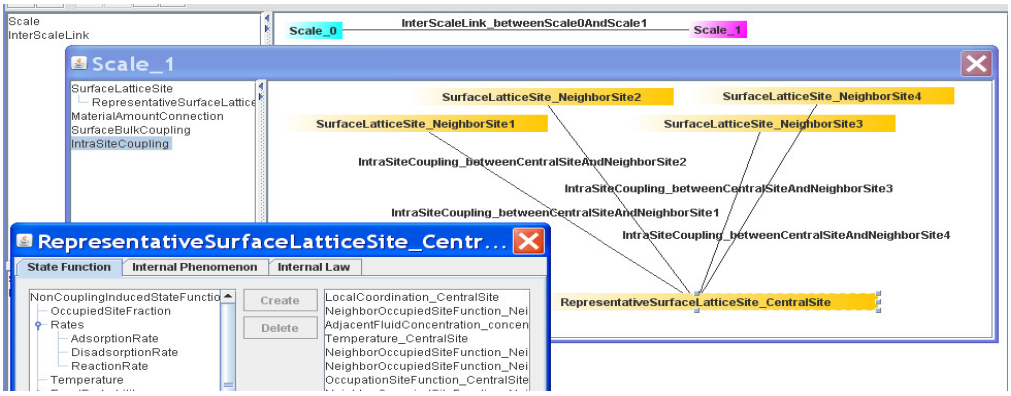

Figure 6: Specification of Scale 1 and its components.

After building up the conceptual model, the SSG is called to produce a simulation script that specifies which modelling tool is used for simulating a scale defined in the conceptual model, what software component should be executed to realize a pre-defined inter-scale link, and what the order of running these tools and components is. (cf. Figure 7)

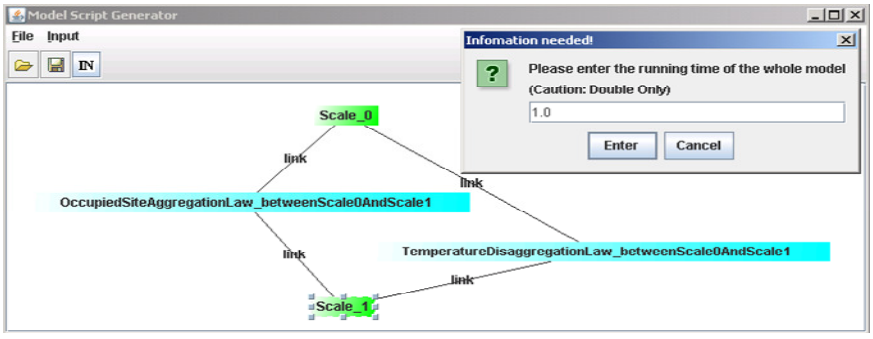

Figure 7: Screenshot of Using the Simulation Script Generator.

As shown in Figure 8, scale_0 (bulk fluid scale) involves the continuum models for both fluid phase and surface as well as the coupling between them whilst scale_1 (molecular-lattice scale) includes the non-continuum model for the sites of molecular-lattice. 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 are needed to be characterized, namely the temperature at the site T_i and the concentration of the reactant A at the fluid phase boundary layer adjacent to the surface $C_{A0,i}$. 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.

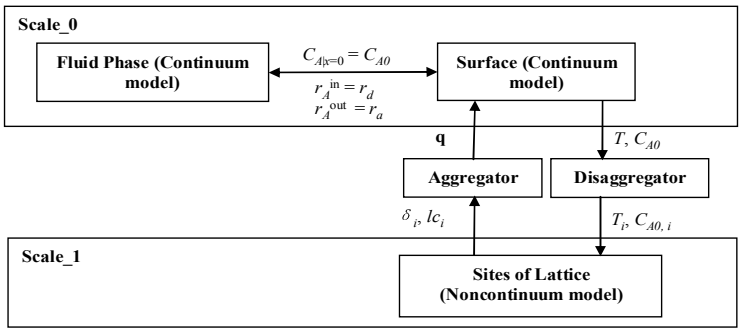

Figure 8: Structure of the heterogeneous reactor model.

In this case study, a continuum model for scale_0 and a Monte-Carlo model for scale_1 were implemented separately, each providing the software interface "ScaleSolver". Furthermore an aggregator and a disaggregator were also implemented offering the corresponding interfaces. The result of the simulation conducted using the MES was found identical to that of an implementation of the entire multiscale model in a single C++ code. In Figure 9, the left plot presents the logarithm of the concentration of reactant A at the fluid phase boundary layer adjacent to the surface versus the logarithm of simulation time; the right plot is the coverage of A at the surface versus the logarithm of simulation time. Note that no difference between the result from MES and that of the single-trunk model is visible. In terms of computational time, the two simulations were also comparable to 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 realising a multiscale model which would otherwise be impractical to develop; in that case the computational burdens would become a secondary concern.

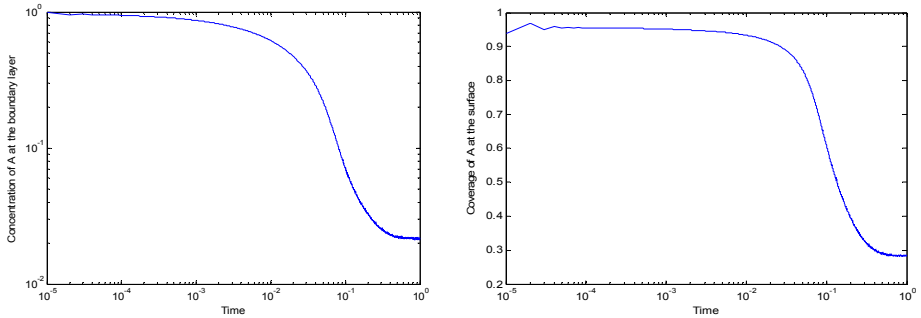

Figure 9: Simulation results of the multiscale reactor model.

CONCLUSIONS

The difficulty and challenges confronted by multiscale modelling call for computer-based support to improve the efficiency of constructing multiscale models. This paper presents a generic methodology for computer-aided multiscale modelling (CAMM) and presents the implementations of several tools that support conceptual modelling, model realization and model execution. A proof-of-concept case study has been presented to demonstrate the developed CAMM tools. Future work will further evaluate the methodology, extend the prototypical tools, and carry out more sophisticated case studies.

ACKNOWLEDGEMENT

This work is supported by the EPSRC (UK) under Grant No EP/G008361/1.

REFERENCES

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.

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

Fraga, E.S., Wills, G., Fairweather, M., Perris, T. 2006. "'Smart Models' - a framework for adaptive multi-scale modelling." *Computer Aided Chemical Engineering*, **21**,457-462.

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.

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

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.

Marquardt, W., Wedel, L.V. and Bayer, B. 2000. "Perspectives on lifecycle process modelling." *Foundations of computer-aided process design*, **96**, 192–214.

Morbach, J., Yang, A., Marquardt, W. 2007. "OntoCAPE - a large-scale ontology for chemical process engineering." *Journal of Engineering Applications of Artificial Intelligence*, **20 (2)**, 147-161.

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

Schopfer, G., Wyes, J., Marquardt, W., Wedel L. 2005. "A library for equation system processing based on the CAPE-OPEN ESO Interface." *Computer Aided Chemical Engineering*, **20**, 1573-1578.

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." *Advances in Chemical Engineering*, **30**, 1.

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. 2010. "Conceptual and numerical support to the development of multiscale models." *Computer Aided Chemical Engineering*, **28**, 1667-1672.