Session [437] – Computational Methods and Numerical Analysis II

An object-oriented framework for modular chemical process simulation [437a]

*Jing Chen and Raymond A. Adomaitis**
Department of Chemical Engineering and Institute for Systems Research
University of Maryland, College Park, 20742

Abstract

This paper discussed the development of a set of object-oriented modular simulation tools for solving lumped and distributed parameter models generated in process design and simulation. The application of object-oriented design (OOD) and modular approach greatly improves current modeling and simulation capability. Modularized components can be easily integrated/adapted to form a new user-defined system. The system can be solved using a sequential or simultaneous approach depending on the specific applications. Also, solutions to different modules can be computed with separate solver algorithms.

Our simulation framework currently is implemented using MATLAB. Design patterns are used to present the system architecture design and give inexperienced users a better understanding of object-oriented analysis and design. With the proposed design patterns, users can create their object-oriented modular system through object-oriented programming languages of their choice.

* Corresponding author. Email: adomaiti@umd.edu

1. Introduction

Chemical Vapor Deposition (CVD) processes constitute an important unit operation for micro electronic device fabrication in the semiconductor industry. Reactant gases are induced into an atmospheric or low-pressure chamber to react and deposit a layer of thin solid film on a heated wafer surface. A wide variety of materials can be deposited by CVD, such as tungsten, copper, gallium, aluminum, poly-silicon and transition group compounds such as Ta_2O_5. Because the deposited materials generally act as contacts, interconnects, capacitors, transistors, etc in the integrated circuit, the ability to reliably deposit uniform thin films is a critical and essential issue to successful semiconductor device fabrication in semiconductor industry. Because of the complexity of the physical and chemical situation inside the reaction chamber and the difficulty of real-time monitoring and control of reaction conditions, theoretical description and simulation of the deposition process are necessary to understand the transport and reaction situations in the reactor. Simulators of the deposition process are also powerful tools for process optimization and control. However, changing or rewriting CVD simulators to adapt to the various types of CVD coupled with the evolution of CVD reactor designs makes current simulation approaches increasingly costly. A good set of simulation tools featuring reusability and extensibility would facilitate the process modeling, simulation and optimization at a relatively low cost.

The motivation of this research is to develop a set of flexible, expansible and reusable open-structure computational tools for chemical process design, simulation and optimization. By using object-oriented techniques and a modular approach, the proposed framework offers such advantages: code reusability, modular components easily tailored to different applications, and the user-defined system easy to maintain. In the following section, we will introduce the design of the system framework and discuss features of packages developed in the framework. Then a semiconductor processing simulation example is provided to demonstrate the utility of the current framework. Finally, the paper concludes and presents the future work of this research.

2. Framework Architecture

The primary goal of our framework design is to provide a flexible and extensible structure for solving a wide range of chemical process simulation problems. The proposed framework is based on the extensive use of object-oriented programming techniques. Our approach is to use modularized models to encapsulate details from the user, and different users can focus on different model building. Composite modules or large modeling systems can be formed easily by combining the modular models. Information exchange between modules through an interface makes it possible to build a hybrid modeling system with modules having lumped and distributed parameter models. Also, the modular structure makes it easier track the sources of solution divergence or other numerical problems. The module behavior can be studied independently by separating modules from the system. The well-defined modules become reusable parts of modeling library.

As shown in Figure 1, the main packages in the framework include physical property database, modular components, systems, relations and solver tools. The property database package works like a library, looking up species thermal and physical properties and calculating them for a given temperature, pressure or composition. The modular components

package includes stand-alone modules that encapsulate subsystem design information and modeling equations. The function of systems and relations is to integrate independent modules together to form the user-defined systems. This system will be solved by the methods offered in the solver packages.

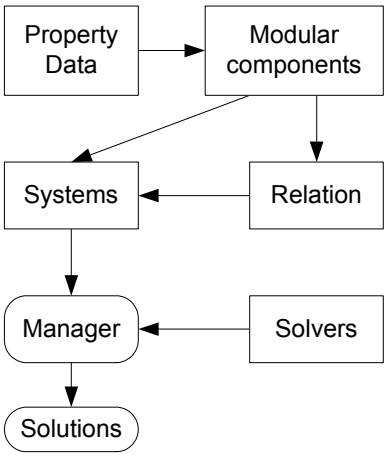

Figure 1. Framework architecture

We present our software design ideas with emphasis on the design patterns methodology. A design pattern is a good solution to a general or commonly recurring design problem and it can be reused again and again. The use of design patterns has multiple benefits: saves designer lots of time and effort; makes the communication between designer and user or other programmers easier and clearer; makes the design more elegant, flexible, extensible and reusable; avoids dealing with the details in the early stage of design by giving the designer a higher-level view on the problems and on the process of design (Gamma, Helm, Johnson and Vlissides, 1995; Cooper, 2000; Shalloway et al., 2002).

We use the *mediator* pattern for the communication among objects of module classes. *Façad*e pattern is used to provide a simple interface to the subsystem of ooMWR tools. We have also used the *strategy* pattern and *adaptor* pattern in the solver package to provide different algorithms and integrate other numerical packages.

3. Framework Implementation

Current database package is developed to provide methods for determining properties of non-polar and polar gases and organic and inorganic gases with emphasis on supporting common semiconductor process gases such as silane, tungsten hexafluoride and trimethylgallium. The thermal physical properties include viscosity, thermal conductivity, diffusivity, heat capacity, molecular weight and density of pure gases and mixtures at ideal gas state. This package is implemented in JAVA for both web applications and local simulations. An interface wrapper class, *GasMixture*, is developed in MATLAB to facilitate data retrieval from JAVA objects.

Based on the properties of modularity, a standalone module class is formulated as follows: constructor method, *residual* or *equation* method, and/or other methods. All process

design, equipment geometry, and related information are stored in the constructor method. If the variables to be found can be expressed by the modeling equations in terms of parameters explicitly, i.e., in the form of $x = g(p)$, then these equations should be put in the *equation* method; if only implicit expressions for variables available, i.e., $f(x, p) = 0$, they should be stored in the *residual* method. Modules can be solved individually to test its convergence behavior.

ooMWR tools is developed for solving boundary value problems (BVPs) in relatively simple geometries using global trial function expansions and weighted residual methods (MWR) (Adomaitis, 2002; Adomaitis et al., 2000; Lin et al., 1999). By using ooMWR tools, a PDE system is discretized and converted to an algebraic system, or ODE/DAE systems (for dynamic cases). The resulting systems then can be solved by the tools developed in solvers classes which will be introduced in the following sections. To integrate the ooMWR tools into the current framework, *façade* pattern is applied to develop the mwrmodel class, which offering an interface to ooMWR class subsystem. The *façade* pattern wraps a complex subsystem together and provides user a simplified interface to access the functionality of the wrapped complex subsystem to makes it easier to use.

To integrate individual module objects together to form a desired modeling system, and still preserve the loose coupling and flexibility, we create a class called *relation*. The communication among objects is administrated through the methods of the *systems* class, which is developed using *mediator* pattern. The *mediator* pattern defines an object that is the only one of knowing and coordinating other classes in the system. These classes communicate with or through the mediator without referring each other explicitly. The *systems* class is mainly used for solving coupled modules simultaneously, while independent modules can be solved sequentially. The current computational framework supports both approaches.

In the solver package, we provide various numerical computational tools for solving AE, ODE or DAE systems. We have a base class, *solvers*, which define data fields, such as, *var*, *param*, *resid*, *solverset*, *currTime*, etc. for subclasses usage, and offer many utility methods, like *unpack*, *display*, *get*, *set*, *columnnae*, *residual* and *equation*, to facilitate derived classes' operations. To solve linear/nonlinear systems, the class *naemodel* is developed using s*trategy* pattern. The *strategy* pattern supports a family of algorithms, which conceptually do the same things but have different implementations. Different numerical algorithms are implemented as member methods of the *naemodel* class. To be able to use existing solvers in MATLAB, we design an interface, class *odemodel*, using the *adapter* pattern to integrate MATLAB built-in ODE/DAE solver packages into our system.

4. Tungsten CVD Simulation

We now consider the problem of simulating tungsten CVD process both at steady state and dynamically over the entire processing operation. The detailed model analysis, heat transfer parameter selections and reactor geometry can be found in the paper (Adomaitis, 2003). To describe one-dimensional thermal dynamics on the surface of wafer, susceptor and guard ring assembly, the energy balance on the wafer assembly surface can be written as:

$$\Delta z \rho \frac{\partial (C_p T)}{\partial t} = \Delta z \frac{1}{r} \frac{\partial}{\partial r} \left[k(T,r) r \frac{\partial T}{\partial r} \right] + Q_{rad} + Q_{conv} + Q_{lamp}$$

with boundary conditions and initial condition,

$$r = 0, \quad \frac{\partial T}{\partial r} = 0; \quad r = R, \quad \frac{\partial T}{\partial r} = 0; \quad t = 0, \quad T = T_0$$

Instead of putting all equations into a single complex module to get the temperature distribution of wafer assembly, several small and simple modules are made up for describing different heat transfer phenomena. Four independent module classes are defined: *wafer*, *lampflux, showerheadflux* and *gasflux*. The class diagram of four module classes is shown in Figure 2.

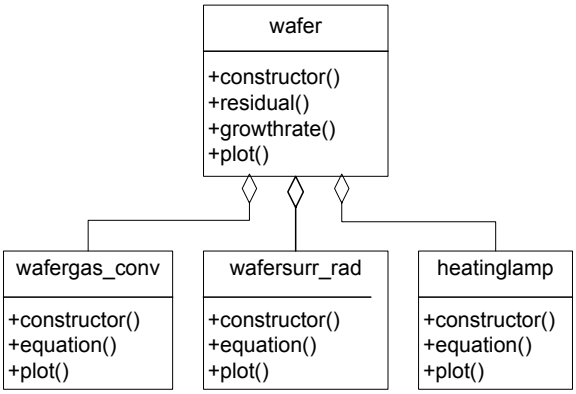

Figure 2. Class diagram of module classes

To obtain the dynamic solution of wafer temperature, we assume the initial temperature at wafer surface is 300K, and the wafer is heated by heating lamp for 10 min and cool down for 10 min. The system assembled by different modules is solved to obtain steady state and dynamic solutions (illustrated in Figure 3).

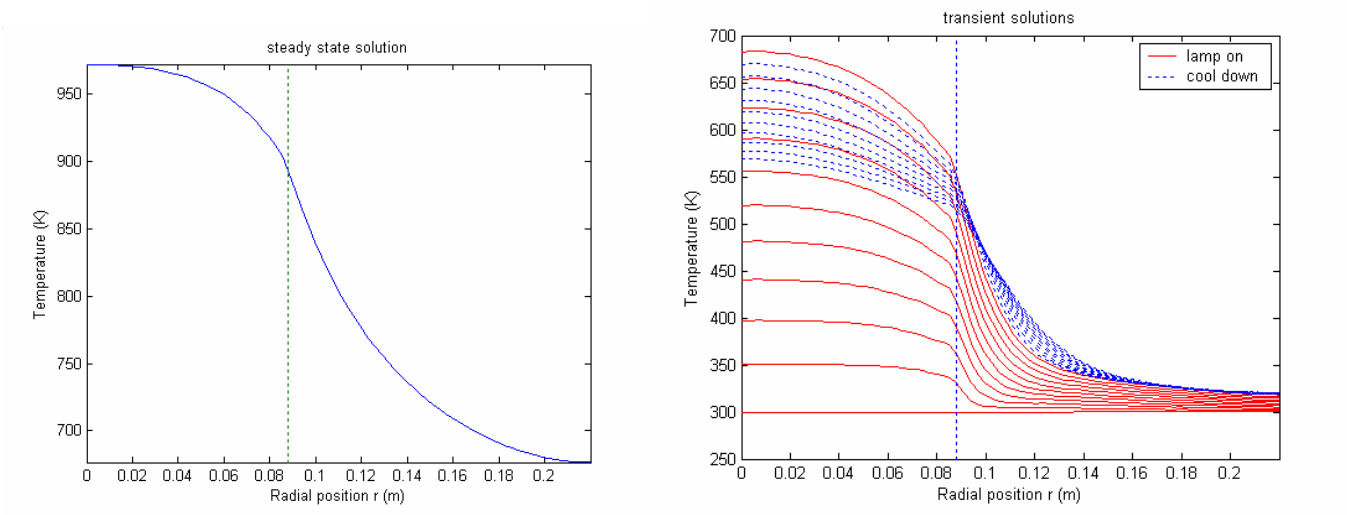

Figure 3. Steady state and dynamic temperature distribution on the wafer assembly

5. Conclusions

The development of this object-oriented computational package offers a set of flexible and adaptable tools for solving large systems consisting of different application models. The application of object-oriented design (OOD) and modular approach greatly improves current modeling and simulation capability. The application of OOD reduces the development cycles of designing new simulators and lowers the costs of process design and simulations. Modularized components can be easily integrated/adapted to form a new user-defined system. Solutions to different modules can be computed with separate solver algorithms.

This simulation framework is implemented using MATLAB. The open architecture of the software makes it possible to integrate other numerical techniques and computational packages, further increasing the software flexibility and reducing development time.

The use of design patterns, which offers a high-level abstract structure to avoid dealing with programming details in the early stage of simulator development, increases simulator design procedure efficiency, and gives inexperienced users a better understanding of object-oriented analysis and design.

References:

Adomaitis, R. A. (2002), Objects for MWR, Computers Chem. Eng., 26, 981-998.

Adomaitis, R. A. (2003), A reduced-basis discretization method for chemical vapor deposition reactor simulation, Mathematical and Computer Modeling, 0, 1-17.

Adomaitis, R. A., Lin, Y. H. and Chang, H. Y. (2000), A computational framework for boundary-value problem based simulations, Simulation, 74 (1), 28-38.

Cooper, J. W. (2000), JAVA design patterns, Addison-Wesley.

Gamma, E., Helm, R., Johnson, R. and Vlissides, J. (1995), Design Patterns: Elements of reusable object-oriented software, Addison-Wesley.

Lin, Y. H., Chang, H. Y. and Adomaitis, R. A. (1999), MWRtools: a library for weighted residual method calculations, Computers Chem. Eng., 23, 1041-1061.

Shalloway, A. and Trott, J. R. (2002), Design patterns explained, Addison-Wesley.