A Modelica Library Organization Method Supporting Online Modeling and Simulation

Xiong Tifan[1] Zhou Zhiming[1] Wan Li[1] Li Yongchao[1]

[1]CAD Center, Huazhong Univ. of Sci &Tech, China,
{xiongtf ,wanli}@hust.edu.cn, {zhouzm, liyc}@comodel.net

Abstract

Today, the trend of achieving networked collaborative innovation and design of complex product based on Modelica is predictable in the industrial field. However, the existing file-based Modelica library organization method designed for single-machine environment does not satisfy the model management requirements for dynamic collaborative modeling and sharing under the network environment. Aiming at this problem, a new organization method of Modelica library based on database is proposed. The main principle of this method is that the organization objects are models rather than files. Through interacting with database storing metadata describing models, it is available to achieve model management based on the granularity of single model. Finally, a network-based multi-domain unified modeling and simulation platform is developed on the basis of the model management architecture.

Keywords: Modelica, organization method, online modeling and simulation

1 Introduction

In recent years, with the increasing complexity of the products in the field of engineering, it is difficult to construct all sub-models from different areas for one person or one team (Zha, 2006), and separate subsystem simulation in different fields cannot meet the requirement of the design innovation. The process of modeling and simulation for complex products is moving in the direction of integrated multi-domain modeling. Modelica, as a unified object-oriented modeling language, can solve this problem properly well. However, the existing Modelica softwares of single-machine environment do not support sharing and reuse of models, which seriously slows down the process of product innovation. So in order to meet the challenges of collaborative management and sharing of models in the internet-distributed environment, it is extremely necessary to build an online platform to provide services for different user roles, such as one person, one team or one enterprise, as shown in Figure 1. A variety of intelligent solutions have been proposed to explore the network-based system.

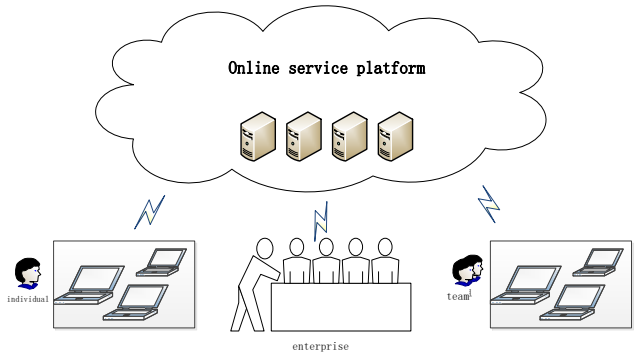

Figure 1. Online service platform

Eissen et al. discussed different realization alternatives for Web-based simulation services and presented the prototypic implementation of a web service which is built on the proposed W3C Web interface stack (Eissen, 2006). This research allows for the analysis and execution of technical models described in the well-known Modelica modeling language. Shi et al. presented an Internet-based electrical engineering virtual lab (IEEVL) using Modelica for unified modeling. It uses XML to represent and exchange information and is only capable of modeling and simulating for electrical engineering domain (Shi, 2011). Mohsen et al. constructed a web-based teaching environment, OMWeb, which is part of the open source platform Open-Modelica. It can be used both in engineering courses as well as for teaching programming languages (Mohsen, 2011). Oscar built the UN-VirtualLab, this web platform offers a free web simulation environment for educational purposes. Users can simulate models that have been stored on the server though setting parameters (Oscar, 2011). Zhang et al. researched and developed a web platform called Proteus. This platform is designed for education and academic research, and provides a place where students, educators and academic researchers can easily create and share their models of physical systems described using Modelica (Zhang, 2013).

However, though these research and platforms could help us to solve some problems in a certain extent, there exist serious obstacles. They do not break *through* the traditional static unstructured organization method of Modelica library, but just developed as

remote presentation systems using the mechanism of compiler and solving in the single-machine environment. The functions are too incomplete to get models reused and shared through the internet and do not support multi-domain collaborative online modeling.

This paper mainly aims to propose a decoupled structured dynamic model library organization method supporting control on single model by combining the database with file storage. The method adopts a way of one file corresponding to one model, to eliminate the coupling relationship among multiple models within a single file. So, through managing model metadata by database and organizing model files by file storage, it is available to restore the stand-alone environment on the server for the existing Modelica compiler and meet the functional requirement of re-use and sharing under the network environment.

2 Analysis of traditional Modelica library organization

In the Modelica language specification, the main compositions are consist of eight types, package, model, type, connector, block, function, record, and class (Fritzson,1998). In order to discuss conveniently, here we just define two types, *package* and *model*. The package of specification is our defined *package*, and the remaining types are collectively called as *model* type, because their operation and organization in the database and file storage are same in this paper.

As shown in Figure 2, all Modelica model libraries, both standard and private, are constructed in the form of *package.mo* files. The physical file organization of Modelica library is based on the file directory, including folders and *mo* files, and the logical file organization takes *Package* and *Model* as its objects. Modelica complier is the core engine of a Modelica-based multi-domain physical modeling and simulation platform. When the compiler works, it needs to search and analyze the referenced or inherited models of current model, and referenced or inherited models in a deeper level (Zhao, 2011). The model library organization based on file directory can well support the existing compiler's searching and compiling function.

As to the traditional organization of model library, there are two methods, structured organization and unstructured organization (Modelica Language Specification Version 3.0). If the folder directory contain *package.mo* file, the organization is structured. In unstructured method, there is no *package.mo* file, package classes and sub-classes exist in the same mo file. For example, as shown in Figure 2, if the Rotational library is organized by the unstructured way, there is only a *Rotational.mo* file, and then it contains all the information, including sub-models, Spring and Damper, also including models in sub-library, AngleSensor and SpeedSensor. A combination of two ways is applied to the existing software in the single-machine environment, which will lead to a situation that one mo file contains multiple packages or multiple models， as shown in Figure 3.

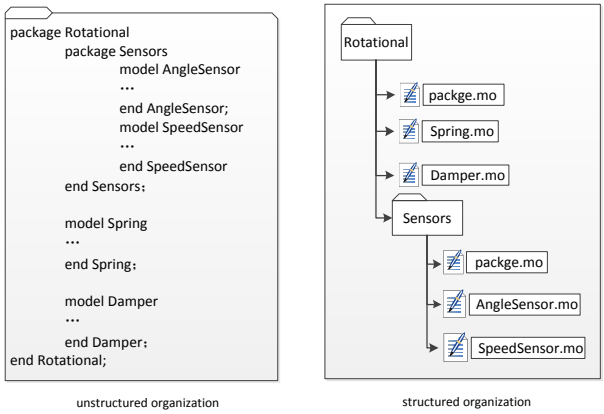

Figure 2. Traditional Organization of Model Library

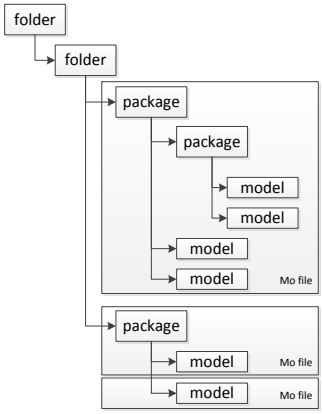

Figure 3. Model organization based on file index

According to the functional requirements of users, the online platform should supply the modeling service, simulation service and model management service at least. Modeling service provides visual modeling to help to create basic components and models. For completed models, users can run simulations by setting the parameters of components through the web-based simulation service. And the model management service mainly enables users manage their models conveniently, including uploading models from local environment, sharing models, renting models, and so on. So the current model management organization is too extensive to satisfy these demands. It mainly displays in two aspects:

1. Simple file directory mode. As to traditional Modelica model libraries, there are only mo files in the file storage, the mo files are taken as control objects. Though the mode meets the requirements well for the model-loading of the compiler, it is negative to achieve the control on single model in a unified way. To realize the sharing and control of single model and make it easy to query and

DOI 10.3384/ecp15118817

manage, we need to change the organization of the model library and organize it orderly based on its object feature attributes, such as the model's name, owner, creation time, release time, the release states, etc.

2. Unstructured model organization method. There can be coupling relationships in a single file, if the library is organized by this method. And you may change other models in a same file when you change one model, which is not suitable to manage models based on the granularity of single model.

3 Model Management Architecture

According to the analysis above, in order to achieve granularity management of single model in the network environment, it's required that the organization objects of library are models rather than files, and the feature attributes and parameters of single model and relationships between models, here collectively called as metadata of models, should be stored on the server. A relational database can help achieve this purpose. Under this premise, in order to obtain compiler's support to realize the online modeling and simulation, the single model in the matching relational database needs to get recognition to search corresponding mo file from a file system. So one to one mapping relationship between database and file base is required.

As shown in Figure 4, in the B / S architecture, we use a combination of model database and model file base to provide data support for modeling and simulation, users can interact with the browser to get the modeling and simulation service and model management service.

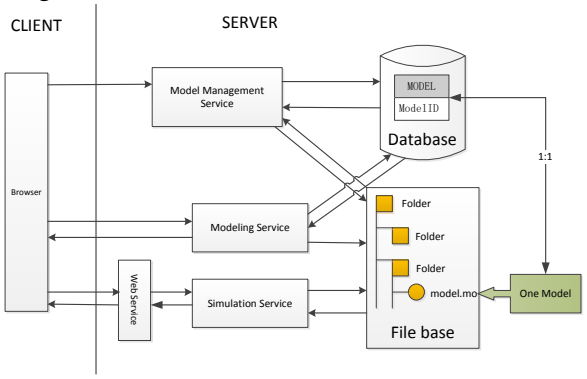

Figure 4. Model management framework

Database is used to store the basic properties (metadata) of model and the relationships (reference, inheritance, etc.) between the models. And file base is organized by *mo* files based on file directory to support the online compiling and simulating service. The model management service can be achieved on the basis of attributes of models in the database. For example, the display and renting authority of model could be accessed via simply changing publishing state. If this model has been published, it can be rented and referenced by other users to acquire re-use. Instead, others cannot search it on the Internet. The modeling service saves the models' mo file in the file base and the models' metadata in the database. Besides, solving results can be stored in the file base by simulation service. In this framework, one model's descriptive information (metadata) in the database corresponds to one mo file in the file base. So, the management of model resources based on the granularity of single model can be reached.

4 Detailed design of management framework

The framework involves the database design, the file base design and the dealing with the relationship between the database and the file base. In addition, importing the existing libraries, managing library version and collaborative modeling are also worth researching. We will discuss these aspects in this part.

4.1 Database design

As shown in Figure 5, we design the metadata of model object which includes name (Name), creator (Creator), create date (CreateDate), release date (PublishDate), status (Status), text information (Text), model's parameters (Parameter), industry information (Industry), major information (Major), model price (Price), model description (Description) and so on. By the designing of these attributes, any model can be acquired, and any operation can be executed. The text information contains four parts: icon, diagram, information and used, which is used to displaying models in different views. Inheritance (Extend) and reference (Used) relations between models can be used to searching models. Package object's attributes are similar to model, which is included in package.

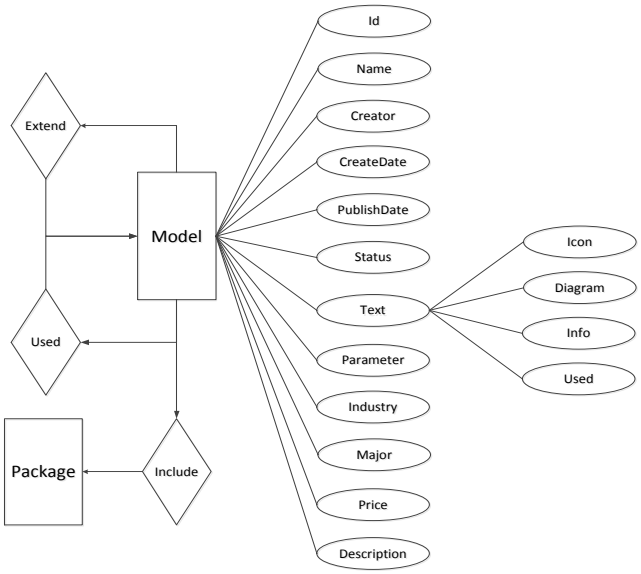

Figure 5. Design of model object

In order to achieve structured display in the client interface (an internet browser) and management toward

single model, it is necessary to use the database to manage the model library. In accordance with the model object, we separate sub-models' details and relationship from coarse-grained *mo* files into the database.

The database structure is shown in Figure 6, Model table, File table and SVG table are designed to save the basic information, text information and svg information (Scalable Vector Graphics, an XML-based language describing two-dimensional vector graphics in a graphical format for graphical modeling and interface icons) of the fine-grained models after initializing from the model library. And user attributes are added to achieve the user's management and control on models. The Component table stores components of all models. The Version table helps to achieve model version management. The Industry table and Major table can separate models into different industries and majors. The Parameter table saves parameter information o models. The Compile Task table records information of compile task and Simulation table records information of simulation task.

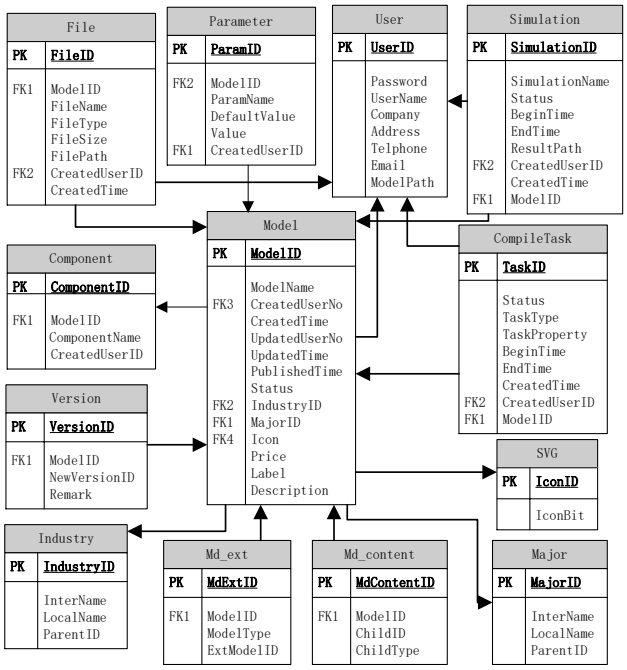

Figure 6. Database structure

Inheritance and reference information of models are critical. These information are included in mo file in the traditional organization method, and compiler can get these information by loading the file into the memory to have a complete model. There will be an error for server to compile a model's mo file if the information is incomplete, which will lead to an unfriendly user experience. For this reason, inheritance relational table (Md_Content) and reference relational table (Md_Ext) are added to store the inheritance and reference relationship between models. When a user loads a specific model, the server will load all the models associated with this model from the relational database.

Through the designing of database, one user can get all the corresponding information of himself or herself, including user information, models, compile tasks, simulation tasks, and so on. The authority control can be achieved conveniently.

4.2 File Base Design

The modified model library organization mode is shown in Figure 7. The physical organizational mode is still represented by folder and mo file, and also the logical organizational mode is still organized by package and model object. But the mapping relationships between models and packages have been changed. We abandon the former coupling relationship between models or packages in mo file, by using a single mo text to represent a single model or package.

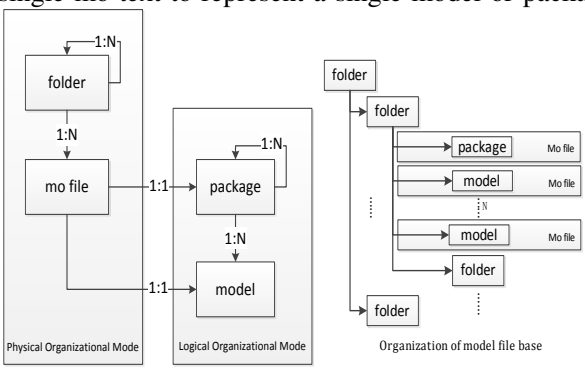

Figure 7. Mapping between models and packages

Compared to the existing model library organization mode, all the models separated from an original package will be stored in a folder named by the package's name, and the *package.mo* file will be the loading index of the compiler, which do not change the way of loading models based on file directory. So the existing compiler can be used in the internet-distributed environment.

In the database, model is the basic unit of management. And mo file is the basic organization object in the file base. After improving organization of Modelica library, one record of model in the database can be matched with one mo file in the file base, as shown in Figure 8. Then any operation to one model through database can respond to the mo file, like adding components, modifying parameters and creating equations. In this way, it is easy to realize synchronous management of database and file base in the process of online modeling.

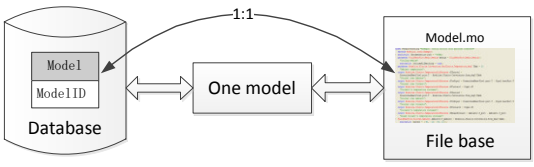

Figure 8. The relationship between database and file base

4.3 Dealing with existing libraries

There have accumulated so many model libraries, since the softwares in the single-machine environment, such

DOI 10.3384/ecp15118817

as Dymola (Dag, 2002), MWorks (Wu, 2006) and Openmodelica (Fritzson, 2005), having been used in machinery, electric, aerospace and other fields for more than a decade. The network platform should make full use of these libraries and import them to expand the library resources, which would be beneficial to achieving re-use, cutting costs and shortening the cycle of product development.

For the existing libraries, there are only mo files and unstructured coupling relationships are common. Then research on how to split them into single-type mode, and storing them in the database, would be crucial. The detailed process of dealing is shown in Figure 9. After uploading a library file, the splitter of the online platform analyzes every mo file, splits them into lower-level mo file one by one, until it just contains one package or one model, and creates metadata in the database about corresponding package or model at the same time. By this way, the existing Modelica libraries can be stored and reused well on the online platform.

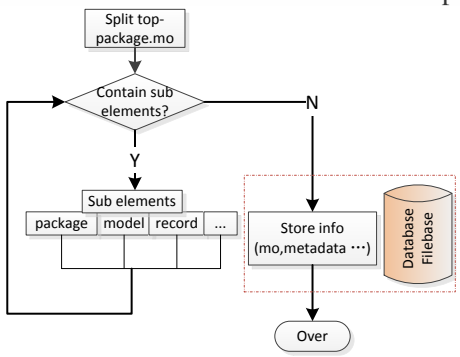

Figure 9. Process of splitting mo file

4.4 Collaborative modeling and version management

The version of one library is labeled as a number separated by '.', like "3.0". When a library is created, the version is defined as 1.0 by default. While one library is published, a copy of this library will be added into database and file base, whose state will be set as published, and meanwhile, the version of original library would be modified as 2.0, which are still editable for creator. This published library can be referenced by others, so the re-use of models is reached easily.

For one huge project, it is necessary to create a group or team on the online platform. All members of this group can edit this library, but different member would get different task according to their professions, and also they would have different authority of task. The complex model can be implemented through integrating sub-models of group members by a charge leader. The progress of collaboration is as shown in Figure 10, multi-domain collaborative modeling can be achieved through the division of task.

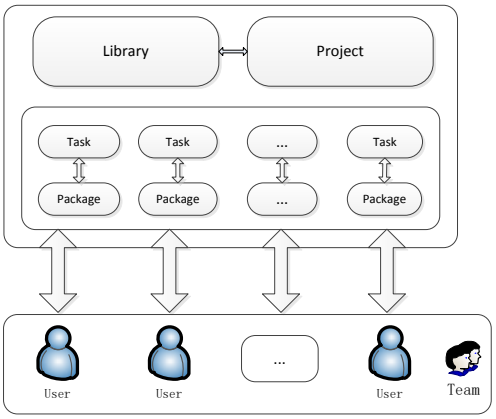

Figure 10. The progress of collaborative modeling

5 Application Verification

An online service platform supporting multi-domain physical modeling and simulation in the web environment - CoModel (http://www.comodel.net.), as shown in Figure 11, has been researched and built based on this organization method of Modelica library.

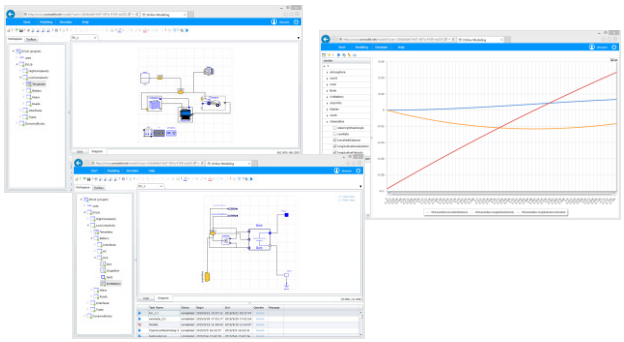

Figure 11. Online modeling page of CoModel

The architecture of data supporting and modeling service of CoModel platform is shown in Figure 12.

In the collaborative visualization modeling and simulation platform, users can interact with the browser to get the modeling and simulation service and model managing service. Modeling and simulation functions are on the basis of file directory in the file storage, which do not change the way of searching and loading models of existing compiler, can be implemented effectively. Solving results are also stored in file storage, users can download them easily. Model management can be achieved through contacting with database and file storage. The platform builds four kinds of libraries, including standard libraries, individual libraries, group libraries, and public libraries. Standard libraries are provided by Modelica Association (http://www.modelica.org.). Individual libraries are created by a user. They store users' private models and nobody can get the models' information except the models' owner. Group libraries are developed by a team, aiming at complex and collaborative product designing. Once individual libraries and group libraries are uploaded and

published on the server, they become public libraries, and then every user in the platform can rent them to establish their own models. All the libraries are organized by a combination of database and file storage. Database stores all the metadata about information of model and package, and file storage stores all mo files by file path. They would respond to any operation on any model in the process of modeling. The fact that at present, this data supporting method based on the decoupled structured dynamic model library organization shows good performance on the platform, proves that this method is advanced and effective to support online modeling and simulation.

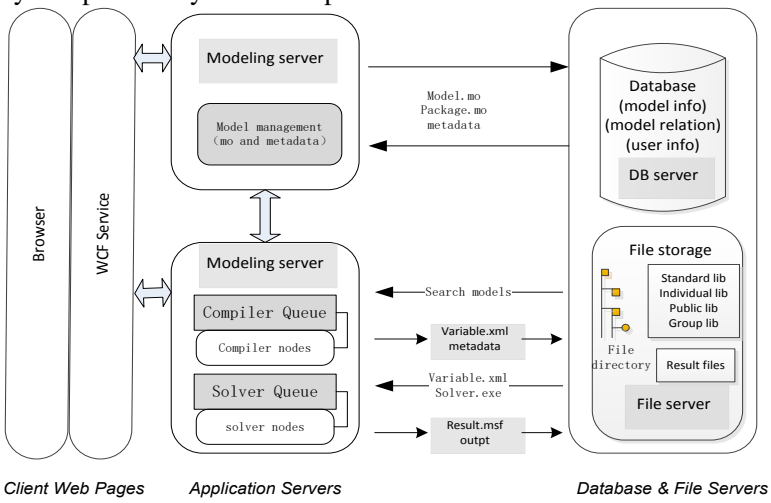

Figure 12. The architecture of CoModel

6 Conclusion

The library organization method acting as the data support of the web-based general multi-domain physical modeling and simulation can effectively achieve dynamic collaborative management and sharing of models under the network environment. And based on this method, the technical feasibility of the reuse and redeveloping of model for further development can be increased. Of course, this work is just an initial study of the web platform supporting online modeling and simulation, and just a compromised way for existing compiler in stand-alone environment. The future work involves the development of networked compiler and improving the efficiency and performance of online platform.

References

Modelica Association. Modelica – A Unified Object-Oriented Language for Physical Systems Modeling Language Specification Version 3.0. http://www.modelica.org.

Brück Dag, et al. Dymola for multi-engineering modeling and simulation[C]. *Proceedings of Modelica,* 2002.

Peter Fritzson, Vadim Engelson. Modelica – A Unified Object-Oriented Language for System Modeling and Simulation[J]. *Lecture Notes in Computer Science,* 1998:67-90. doi: 10.1007/bfb0054087.

Peter Fritzson, Peter Aronsson, Lundvall Håkan, et al. The OpenModelica modeling, simulation, and development environment[J]. *Simulation News Europe,* 2005.

Torabzadeh-Tari Mohsen, et al. OMWeb – Virtual Web-based Remote Laboratory for Modelica in Engineering Courses[J]. *Tari,* 2011. doi：10.1109/ICCSN.2011.6013894.

Duarte Oscar. UN-VirtualLab: A Web simulation environment of OpenModelica models for educational purposes[C]. *Proc 8th Modelica Conf .* Dresden，Germany: Link ping University Electronic Press，2011: 30-31. doi：10.3384/ecp11063773.

Eissen S M Z, Stein B. Realization of web-based simulation services[J]. *Computers in Industry,* 2006, 57(3): 261-271. doi: 10.1109/VPPC.2006.364294.

Zhengyin Shi, Shenglin Zhao, Shan-an Zhu. An Internet-based electrical engineering virtual lab: Using Modelica for unified modeling[J]. *IEEE International Conference on Communication Software & Networks,* 2011:555 - 559. doi：10.1109/ICCSN.2011.6013894.

Yizhong Wu. Development of hybrid modeling platform for multi-domain physical system[J]. *Journal of Computer-Aided Design & Computer Graphics,* 2006, 18(1):120-124.

Xuan F. Zha, H. Du. Knowledge intensive collaborative design modeling and support Part I: Review, distributed models and framework. *Computer in Industry,* 57(1): 39-55, 2006. doi: 10.1016/j.compind.2005.04.007.

Yanshan Zhang, et al. A knowledge-based web platform for collaborative physical system modeling and simulation[J]. *Computer Applications in Engineering Education,* 2013. doi: 10.1002/cae.21572

Jianjun Zhao, Zijun Wu. Multi-domain modeling and co-simulation based on Modelica[J]. *Computer Aided Engineering,* 2011.