An extensible framework for smart engineering simulation software: its architecture, implementation and applications

Chunyu ZHANG[a]* and Liwei MA[b]
[a]Sino-French Institute of Nuclear Engineering & Technology, Sun Yat-Sen University (Zhuhai Campus), 519082, Guangdong, China
[b]Tech-star Software Co. Ltd, Nanjing, 210000, Jiangsu, China

ABSTRACT
The industries are seeking smarter simulation tools to automatize the modeling work to alleviate the difficulties of using the numerical simulation techniques. The present work designs and implements an extensible framework to facilitate the development and deployment of the smart application-oriented simulation softwares. The kernel of the framework is the data management module which treats the various data in a unified approach.The geometry modeling module, the meshing module, the analysis module and the visualization module manipulate the data and the graphical user interface module assists in manipulating and displaying the data interactively. Two typical applications, i.e., the software for computing and analyzing the load-carrying capabilities of slew bearings and the software for designing the transmission tower, are presented for demonstrations.

Keywords: Computer-aided engineering, Numerical simulation, Extensible, Smart software

1. INTRODUCTION

Numerical simulation is playing an increasing role in both engineering designs and scientific researches. The 2006 National Science Foundation (NSF) Report on "Simulation-based Engineering Science" [1] showed the potential of using simulation technology and methods to revolutionize the engineering science. In recent years, the numerical simulation codes have advanced significantly in their modeling capabilities and user-interfaces. Nevertheless, using the general-purpose simulation codes well is still a challenging task: on one hand, the user must be familiar with the large number of commands and the fine-tuning options of the codes; one another hand, the user must have a clear understanding in the underling algorithms (for example, the finite element method) in order to properly model the physical problem. Sometimes, the user has to strive to feed and debug user-defined subroutines to the bulk code in order to define complex loadings or to implement new material models.

In order to alleviate the difficulties of using the numerical simulation technique, industry or application-oriented simulation codes, such as the plastic injection molding design software [2], the die-casting simulation software [3, 4], the sheet forming simulation software [5, 6], the turbine blade design software [7] and the composite material design software [8], have been developed into which knowledge of the industries and optimized modeling practices are incorporated. These smart codes greatly improve the efficiency and the

*Corresponding Author: E-mail: zhangchy5@mail.sysu.edu.cn Tel: 0756-3668967 reliability of the modeling work. Although designed for different applications, these smart codes share many common functionalities, such as geometric modeling, mesh generation and results visualization just as the general-purpose codes. Considering the huge cost and the long cycle to develop these codes by using various techniques, an extensible framework with these common functionalities would obviously speed up the development and deployment of smart simulation codes with less cost.

In recent years, many industries are seeking even smarter simulation tools to automatize the tedious modeling work which are prone to errors, such as parametrizing commonly used components and meshes, updating the the definition of groups of elements from the CAD data to apply boundary conditions, defining the complex interaction between components in huge assembled structures and so on [9–11]. Although it is quite challenging to develop these smarter simulation softwares considering the interoperation between CAD modeling and subsequent discretization and calculation, a framework of a general-purpose simulation software and an access to the core functions of the framework (for example, manipulation of the geometric data and the mesh data, supervision of a computation task) would facilitate the implementation of the chaining and the coupling between the different modules [12].

The first modular, extensible and general-purpose framework for numerical simulation is Salomé [13,14], an open-sourced code which itself makes good use of other open-sourced codes, such as OpenCASCADE [15] for geometry modeling, Netgen [16] for meshing in addition to its own meshers, Paraview [17] for data visualization and so on. Several application-oriented simulation platforms have been successfully built upon Salomé, such as the design platform for superconducting magnets [12] and the platform for nanoscale device simulation and visualization [18]. The goal of Salomé is to provide a platform for embedding different simulation codes into a single framework and to achieve this, advanced but complex programming models (for example, the distributed object model) and rules are adopted.

Motivated by developing smarter simulation softwares and inspired by the success of the Salomé platform, the present work designed and implemented a simpler but more flexible framework of which the kernel is the data management module. Other modules, i.e., the geometry modeling module, the meshing module, the analysis module and the visualization module manipulate (create, edit, save and load) the data and the graphical user interface (GUI) module assists in manipulating and displaying the data interactively. Pure object-oriented programming with C++ was adopted and based on the framework, industry-oriented applications can be developed within a short cycle. Two typical applications, i.e., the software for computing and analyzing the load-carrying capabilities of slew bearings and the software for designing the transmission tower, are presented for demonstrations.

2. THE ARCHITECHTURE OF THE FRAMEWORK

The general architecture of the extensible framework (Fig. 1) is similar to that of the Salomé platform and consists of a set of generic modules of which the data management module (the data module hereafter) is the kernel.

Different from other types of codes, the numerical simulation softwares usually have to manipulate very complex data such as the geometry data, the mesh data, the scalar or vector field data and various parameters which store the material properties, the boundary conditions and other informations. The data module treats these different kinds of data in a unified approach by using a single database (see 3.1). All the data are declared as objects within the framework of object-oriented programming. One of the prominent advantages of this single-database approach is that the data can be accessed and operated very conveniently. And in addition, interrelationships can be defined between different data objects, for example, by

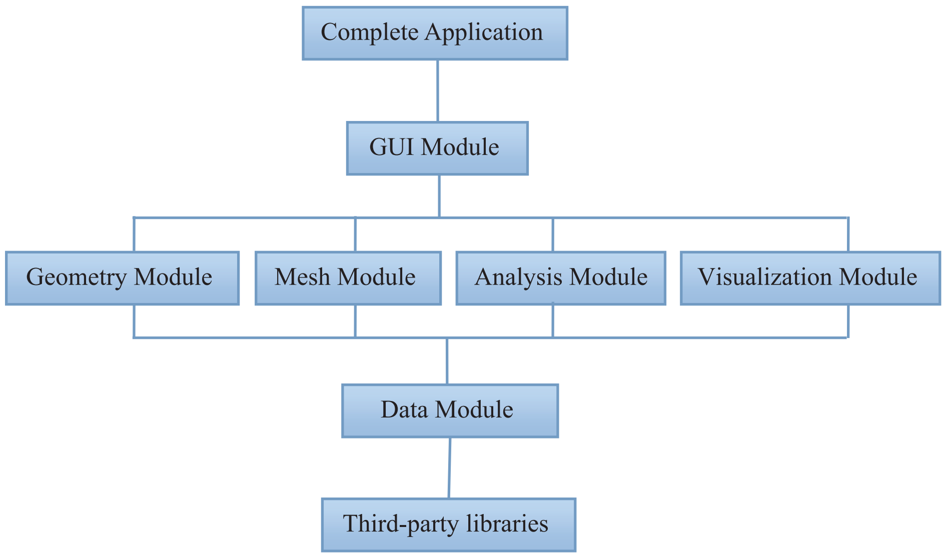


Figure 1: The general architecture of the extensible framework

storing the identification numbers of the dependent objects, to achieve a synchronization in the data. Automatic update of the whole numerical model from the original CAD data is an important step toward intelligentizing the simulation codes.

The geometry module provides basic functionalities to create, import (CAD models stored in standard data exchange formats such as STEP, IGES and BREP), edit and access the geometric entities. The module also provides some "smart" functionalities to identify and select desired geometric entities, for example, to recognize bolt holes, to generate middle surfaces from thin solid CAD models, to select the closest geometric entities of a reference point and so on. And in addition, standard parts such as bolts and flanges can be defined as parametric models (Fig. 2). With a full access to the basic functionalities, the advanced functionalities can be extended continuously and the geometric modeling and operations will become smarter and smarter.

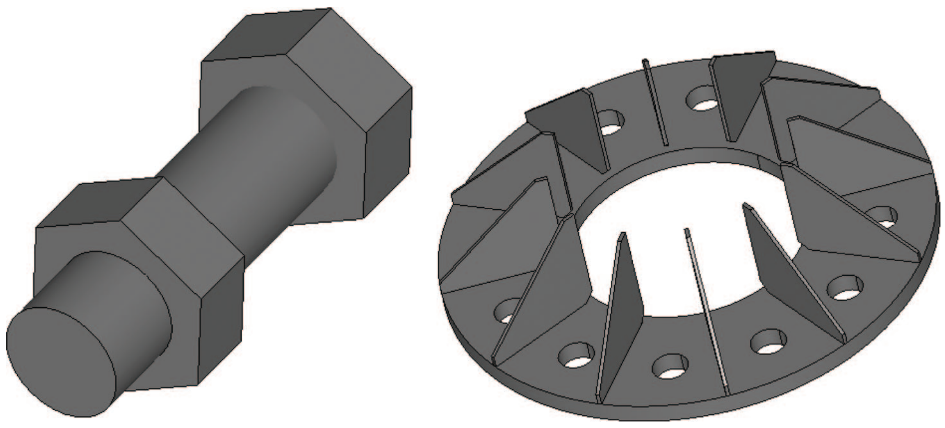


Figure 2: Demonstration of the parametric modeling of bolts (left) and flanges (right)

The mesh modules provides capabilities to discretize geometric entities with primitive elements such as line segments, triangles, tetrahedrons or hexahedrons. For simple geometric entities such as blocks, spheres and cylinders, parametric meshing templates are provided with using the best meshing practices (Fig. 3). Meshes as well as mesh groups can be defined to be dependent on geometric objects and thus a smart meshing process can be achieved [10].

The analysis module provides functionalities to define boundary conditions and to create and supervise analysis jobs. With a full access to the geometric entities and the meshes, some boundary conditions, especially those prone to errors such as the bolt connections or the interactions between the parts of a complex assembly [11], can be defined automatically by following prescribed rules. The materials properties of grouped meshes and some other supplementary information such as the thicknesses of shell meshes can be assigned automatically as well. The complete input decks which are fed into the solvers can be generated based on predefined templates which follow the syntax of the solvers. At present stage, the analysis module only supervises local analysis jobs. Through some standard communication interfaces, it would be straightforward to monitor the jobs running on other computers.

The visualization module provides rendering engines for the different kinds of data, i.e., the geometric entities, the meshes, the bulk field data and the analysis reports. A rendering engine is called a view and multiple views can be created at a time. The concept of ‘view’ here is different from that often used in the conventional document/view programming architecture [19] where a view refers to a window displaying the data. An equivalent concept of ‘viewport’ is adopted by the present framework to refer to the graphical window.

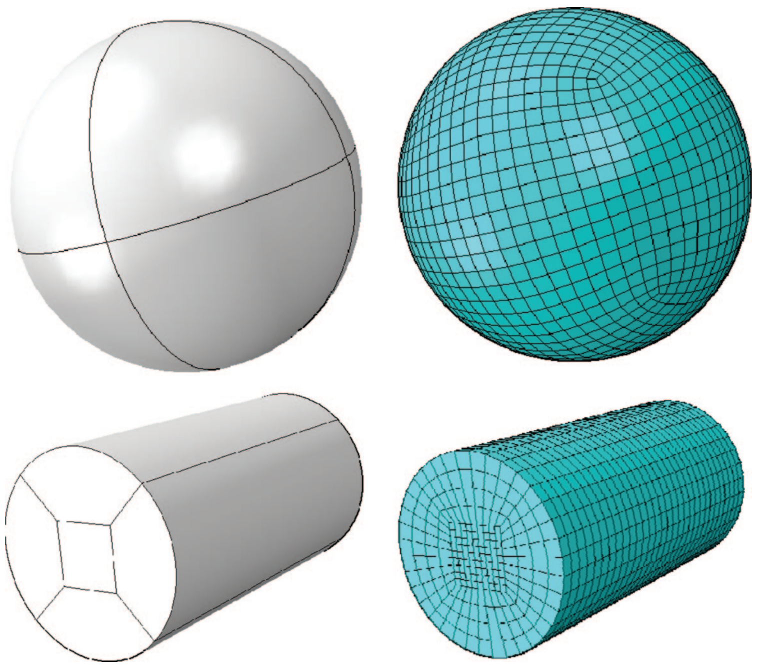
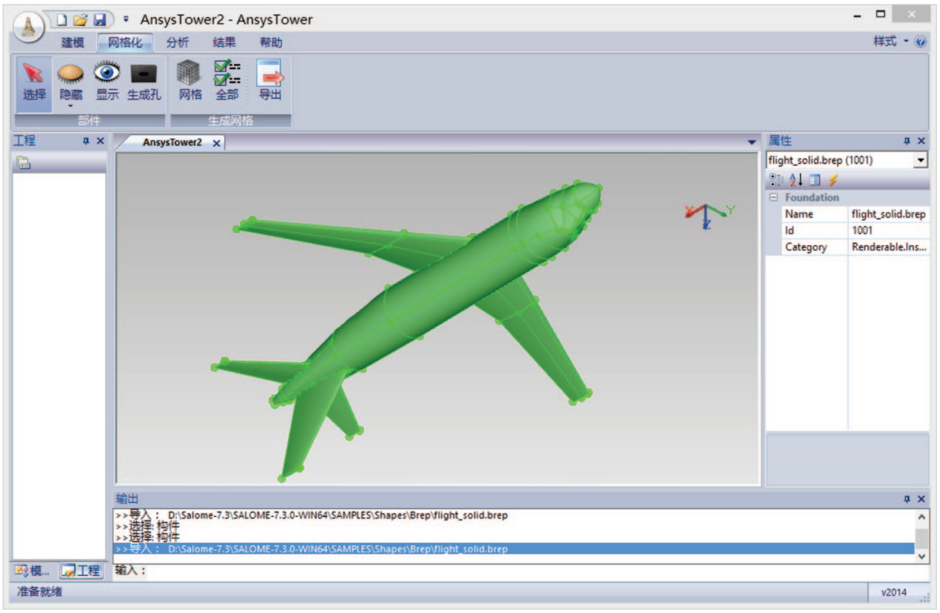


Figure 3: Meshing templates for spheres and cylinders. Partitioning is performed on the shapes to ensure high quality of the generated elements



flight_solid.brep (1001)

Name	flight_solid.brep
Id	1001
Category	Renderable.Ins...

```
>>导入：  D:\Salome-7.3\SALOME-7.3.0-WIN64\SAMPLES\Shapes\Brep\flight_solid.brep
>>选择: 构件
>>选择: 构件
>>导入：  D:\Salome-7.3\SALOME-7.3.0-WIN64\SAMPLES\Shapes\Brep\flight_solid.brep
```

Figure 4: Illustration of the graphical user interface

The GUI module in fact is a complete skeleton of generic simulation softwares which assists in manipulating and visualizing the various data in an intuitive and interactive way. The module contains essential GUI elements such as menu, toolbar, status bar, properties panel, view windows, model trees and so on (Fig. 4). It is a template to develop other smart application-oriented simulation codes.

3. IMPLEMENTATION

Object-oriented programming with the language C++ was adopted to code the extensile framework described above. Robust open-sourced packages were exploited where it is possible.

The unified data model consists of the kernel of the present software framework and each kind of data (various parameters, geometric entities, meshes, field results) is described by a class which is derived from the same root class, i.e., *AObject* (Fig. 5). The root class defines the common members and methods for all the derived classes, i.e., an identifier to distinguish the subclass and a serialization interface to save and restore the objects instantiated from the class. Basic types of parameters such as string, float number, integral number, Boolean number, vector and even color (ternary number) are derived directly from the root class with additional members to do math operations and type conversions. Usually these basic types of parameters are not manipulated directly but used to define the properties of the more advanced objects such as the geometric entities and the meshes. The advanced objects are managed by a single database (a document) and share the same parent class, i.e., *AElementWithProperties* (Fig. 5), where a unique identification number, a name, a category, a list of parameters of basic types (the properties) and a pointer to the database are declared. The geometric part is described by the derived class *APartInstance* (Fig. 5) where the topological representation of the part is declared and manipulated through interfacing to the open-sourced 3D modeling engine, i.e.,

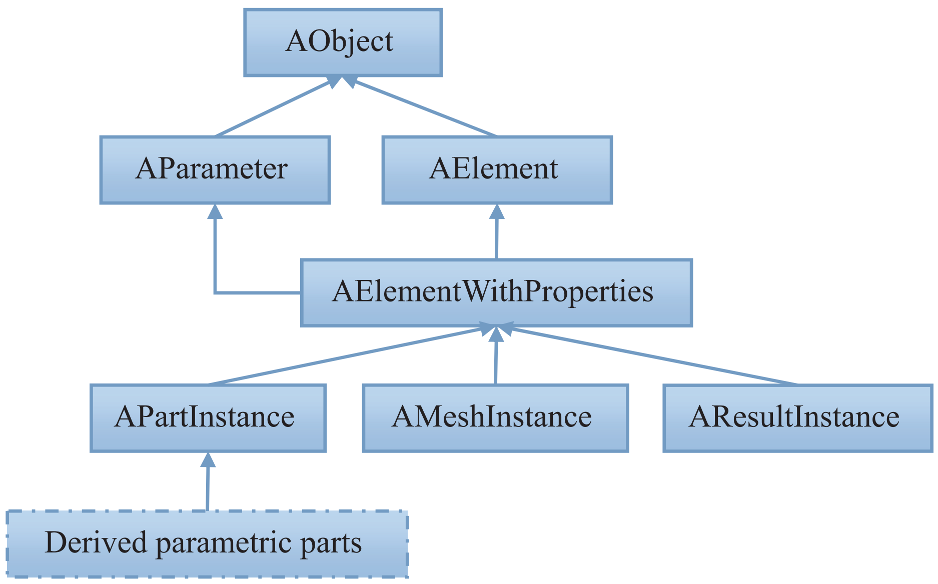


Figure 5: Hierarchical structure of the unified data model

OpenCASCADE [15]. Other properties such as the material, the color, the position and the orientation are also defined in the class and the parent class. The mesh data is described by the derived class *AMeshInstance* (Fig. 5) where the MED data format [20] is used to manage the nodes and elements of the mesh. In additions, two pointers are declared to preserve the dependent geometries, one pointing to the geometric part and the other pointing to the geometry used to generate the mesh (geometric cleaning and healing are usually needed before mesh generation and therefor the second geometry is often different from the original one). The result field data is described by the derived class *AResultInstance* (Fig. 5) where the MED data format is used again to manage the scalar or vector fields defined on the mesh.

The geometry module focuses on the advanced manipulation of the geometric data through interfacing to the OpenCASCADE modeling engine. One the request of practical applications, several functionalities have been incorporated into the module:

- Manipulation of a linear or a circular group of parts (Fig. 6);
- Parametric modeling of several kinds of standard parts such as bolts, flanges, tubes, angle steels and roller bearings (Fig. 7);
- Smart recognition of typical topologies of shapes such as holes, cylinders, spheres, revolved shapes and extruded shapes;
- Searching utilities to identify geometric entities, such as finding the closest hole to a reference point and finding the closest shape to a specified shape.

These advanced functionalities can be extended continuously and the geometric modeling and operations will become smarter and smarter.

In the mesh module, all the meshing algorithms are derived from the root class, i.e., *AMeshAlgo*, where the member function to generate the mesh is declared. The pointer to the database, the identification number of the underlying geometry, the dimension of the mesh and the name of the meshing algorithm are all passed to the member function by

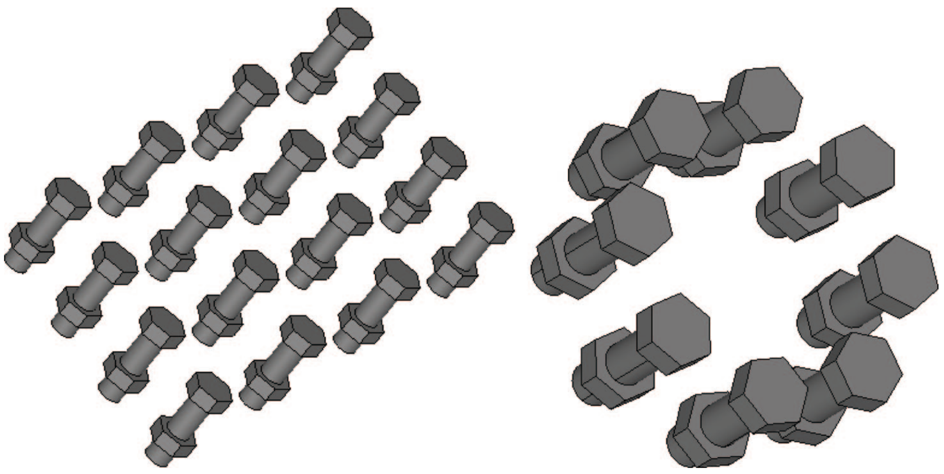


Figure 6: Demonstration of a linear (left) and a circular (right) group of bolts

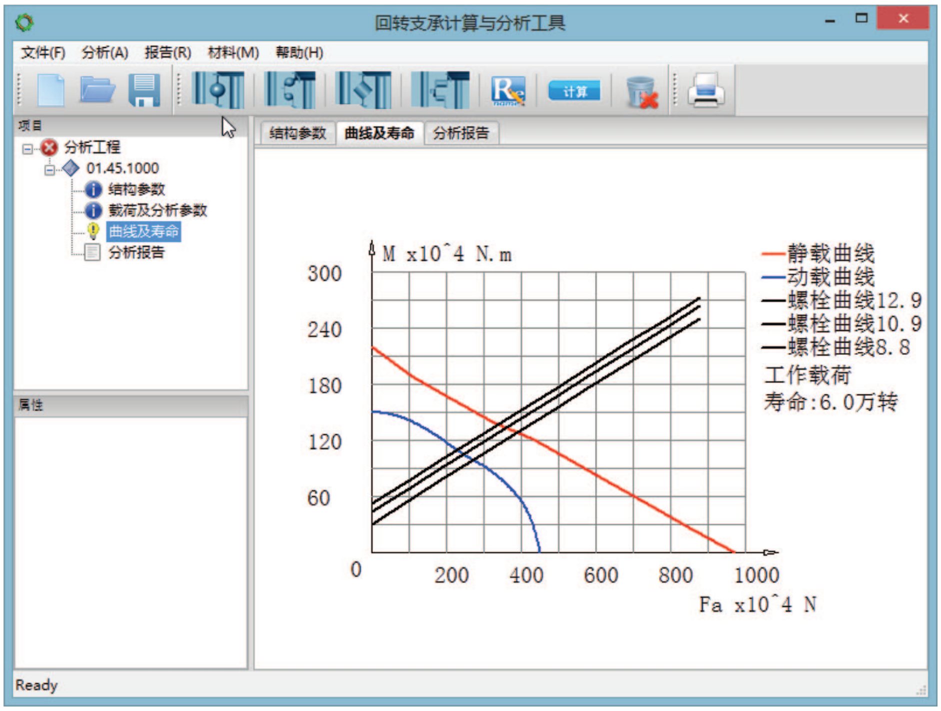


Figure 7: GUI of the bearing code

the structure data named *AMeshContext*. Two meshing algorithms have been implemented:

– Generation of 2D and 3D structured mesh by solving elliptic equations [21, 22];
– Generation of triangles and tetrahedrons by the advancing front method through interfacing to Netgen [16];

A mesh can be created as the following piece of code demonstrates,

```
AMeshContext context;
AMeshInstancePtr pMesh;
context.pDocument = pDocument;
context.partId = idPart;
context.type = AFEAElement::FEA_2D;
context.algoName = AMeshAlgo::NETGEN;
pMesh = AMeshAlgo::Create(context);
```

Utilities to define element groups on selected edges or faces, to combine multiples meshes and to convert the MED data format to other data formats (for example, the format of ABAQUS [23]) are also implemented in the class *AMeshUtil*.

The analysis module defines four kernel classes to create and monitor an analysis task, i.e., *ADataCollector* to scan the unified database and collect the different categories of information (materials, boundary conditions, mesh properties, contact pairs and so on) in the predefined lists, *ASolver* to generate input decks for the solver and to set environmental variables for the solver, *AMonitor* to monitor the status of the computation and *AReporter* to generate an analysis report in the rich text format [24].

In the the visualization modules, viewports for displaying the different kinds of data are implemented, i.e., *A3DViewport* for displaying the geometric data, *AVTKViewport* for displaying the mesh and the field data through interfacing to the open-sourced visualization toolkit (VTK) [25] and *AReportViewport* for displaying the analysis reports. To ensure the platform-independence of these graphical windows, the cross-platform GUI library, i.e., wxWidgets [26] was chosen for its more native looking interface and its wide use in many sectors.

The GUI module integrates the above modules and forms a generic skeleton of generic simulation softwares. The conventional document/view programming architecture was adopted and support of multiple analysis was ensured by the multiple document interface (MDI). Predefined lists and maps (called 'tables') are created in each document to store the different data as the following piece of code demonstrates,

```
//Member function: OnNewDocument()
ADocument* pDocument = GetDocument();
AUndoSetter disableUndo(pDocument);
pDocument->AddTable(AElementTablePtr(new APartInstanceTable()));
pDocument->AddTable(AElementTablePtr(new AMeshInstanceTable()));
pDocument->AddTable(AElementTablePtr(new AResultInstanceTable()));
```

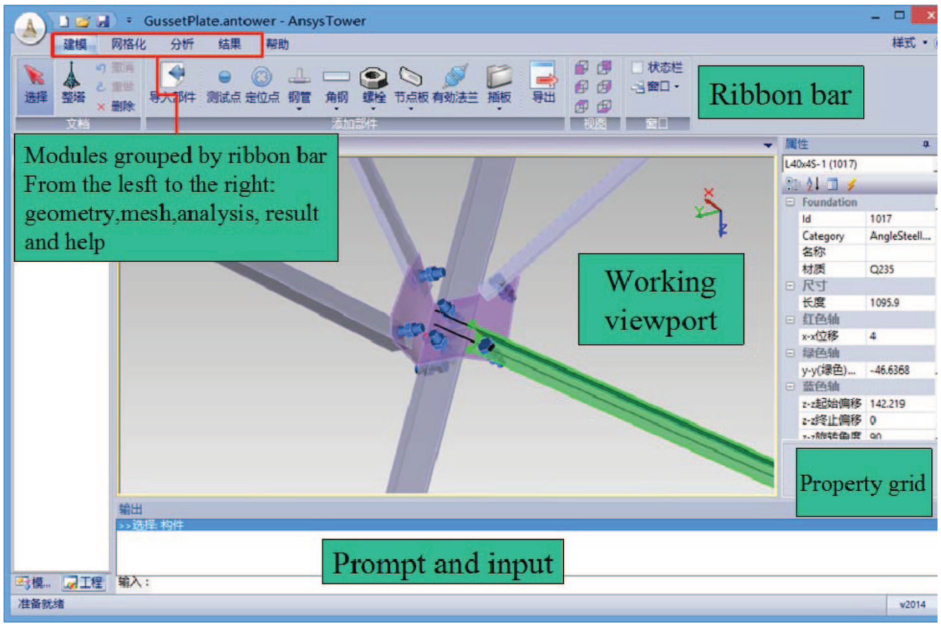
The wxWidgets library was used to create the GUI. Besides the conventional GUI elements, advanced GUI controls such as the ribbon bar, the property grid and the floating/docking frames can be conveniently integrated. A property grid manager is developed to automatically link the properties of the selected objects with the property grid.

4. CASES OF APPLICATIONS

Two applications have been successfully built upon the above modular and extensible framework. The first one is a computation tool to analyze the load-carrying capabilities of slew bearings (Fig. 7, the bearing code hereafter) and the second is a smart CAE software to analyze the mechanical behavior of transmission towers by using the finite element method (Fig. 8, the tower code hereafter).

The bearing code uses an efficient semi-analytical method to solve the complex problem of contact mechanics [27]. Neither geometry modeling nor subsequent meshing is needed and therefore, only the GUI module, the data module, the analysis module and the visualization module are integrated into the code. A slew bearing is declared to be an object instantiated from the class *APartInstance* and the geometric parameters such as the diameter of the bearing, the number of rollers and the radius of the grooves, are defined as the properties of the object. The end-user inputs the geometric parameters, the load levels and the analysis options either by following a step-by-step wizard or through the property grids (Fig. 9). These parameters and options are directly fed into the in-house developed solver and an analysis job immediately begins. When the computation completes, the static loading cure, the dynamic loading curve, and the bolts loading curves are displayed in a viewport (Fig. 7). According to a predefined template, an analysis report is automatically generated and displayed in another viewport (*AReportViewport*) which supports printing and editing of the report.

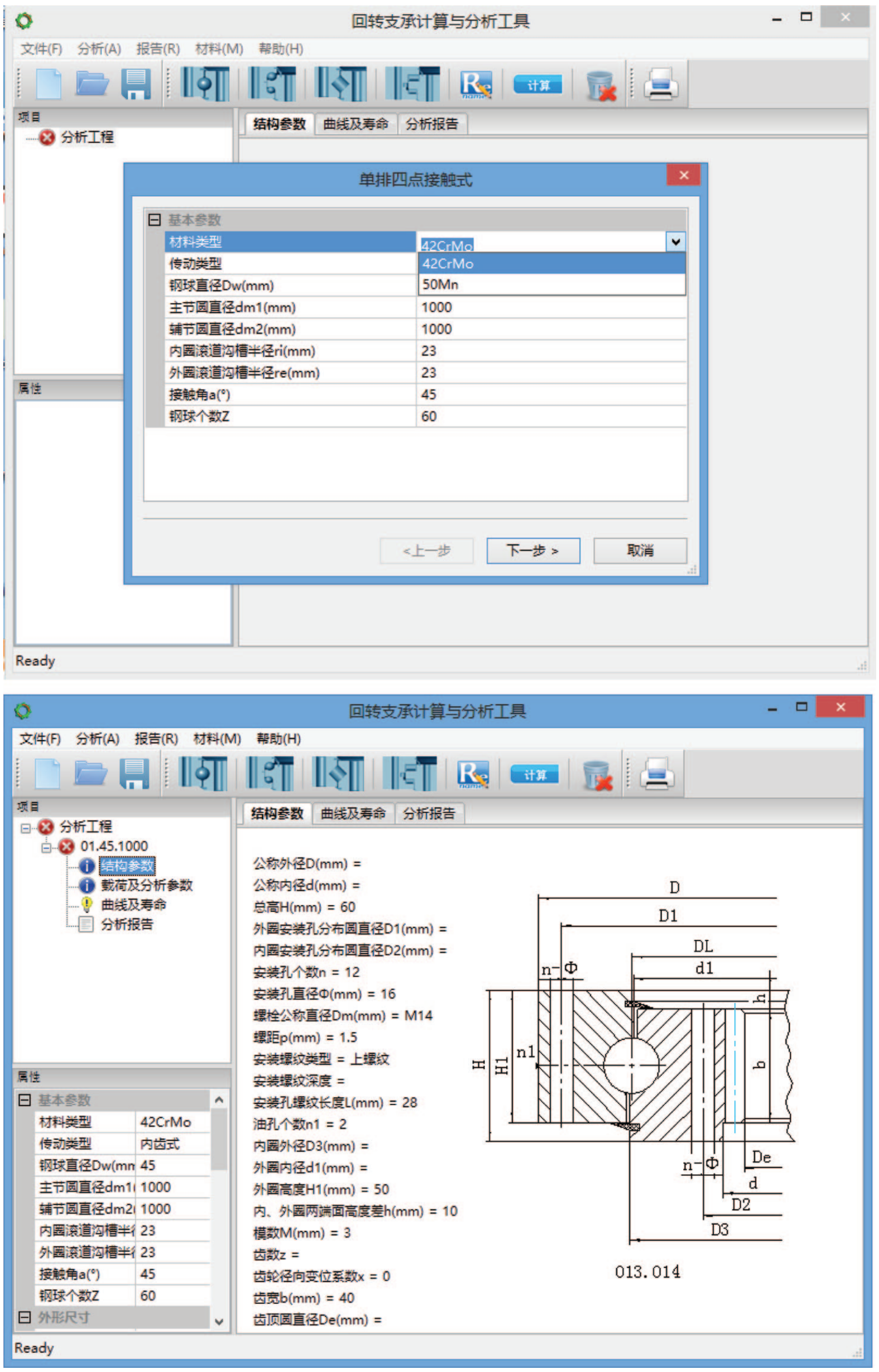
The tower code is a typical industry-oriented simulation software. It is quite tedious to set up a finite element model to calculate the mechanical response of transmission towers considering the complex bolt connections, the complex weldments and the complex part interactions have to be properly modeled. Therefore, the industry has long been seeking


GussetPlate.antower - AnsysTower
建模 网格化 分析 结果 帮助
样式
选择 塑塔 撤消 重做 删除
文档
导入部件 测试点 定位点 钢管 角钢 螺栓 节点板 有効法兰 插板 导出
添加部件
视图
状态栏
窗口
Ribbon bar
Modules grouped by ribbon bar From the lesft to the right: geometry,mesh,analysis, result and help
Working viewport
Property grid
属性
输出
>>选择 构件
Prompt and input
输入：
模... 工程
准备就绪
v2014

L40x4S-1 (1017)

Foundation	
Id	1017
Category	AngleSteell...
名称	
材质	Q235
尺寸	
长度	1095.9
红色轴	
x-x位移	4
绿色轴	
y-y(绿色)...	-46.6368
蓝色轴	
z-z起始偏移	142.219
z-z终止偏移	0
z-z旋转角度	90

Figure 8: GUI of the tower code



基本参数	
材料类型	42CrMo
传动类型	42CrMo
钢球直径Dw(mm)	50Mn
主节圆直径dm1(mm)	1000
辅节圆直径dm2(mm)	1000
内圈滚道沟槽半径ri(mm)	23
外圈滚道沟槽半径re(mm)	23
接触角a(°)	45
钢球个数Z	60

基本参数	
材料类型	42CrMo
传动类型	内齿式
钢球直径Dw(mn	45
主节圆直径dm1(	1000
辅节圆直径dm2(	1000
内圈滚道沟槽半(	23
外圈滚道沟槽半(	23
接触角a(°)	45
钢球个数Z	60
外形尺寸	

公称外径D(mm) =
公称内径d(mm) =
总高H(mm) = 60
外圈安装孔分布圆直径D1(mm) =
内圈安装孔分布圆直径D2(mm) =
安装孔个数n = 12
安装孔直径Φ(mm) = 16
螺栓公称直径Dm(mm) = M14
螺距p(mm) = 1.5
安装螺纹类型 = 上螺纹
安装螺纹深度 =
安装孔螺纹长度L(mm) = 28
油孔个数n1 = 2
内圈外径D3(mm) =
外圈内径d1(mm) =
外圈高度H1(mm) = 50
内、外圈两端面高度差h(mm) = 10
模数M(mm) = 3
齿数z =
齿轮径向变位系数x = 0
齿宽b(mm) = 40
齿顶圆直径De(mm) =

Figure 9: The step-by-step wizard (upper) and the property grid (lower)

a smart CAE software to automatize the modeling and analyzing process. The tower code integrates all the modules and additional functionalities specific to transmission towers have been developed. All the functionalities are grouped by the ribbon bar (Fig. 8). The preliminary three dimensional model of a joint can be automatically generated by reading the structural parameters from a one dimensional line model (Fig. 10). Bolts, flanges, tubes, angle steels and additional connecting plates can be created by parametric modeling (Fig. 2). The properties of the parts are listed in the property grids and can be edited in an interactive way (Fig. 8). For each kind of the parts, a parametric meshing algorithm is derived from the root class *AMeshAlgo* and the common meshing strategy is overridden by an improved one which is more suitable with the topology of the part. By iterating over all the parts, the mesh of the assembly can be generated by one click (Fig. 11). Within the analysis module, a subclass, i.e., *ACodeAsterSolver* is derived from the class *ASolver* and generates input decks for the finite element solver, i.e., Code-Aster [28] based on predefined templates. A piece of the generated input file is demonstrated in Fig. 12. Environmental variables for the solver are set through a dialog. When the computation complete, the result file can be imported and displayed in a viewport (*AVTKViewport,* Fig. 13).

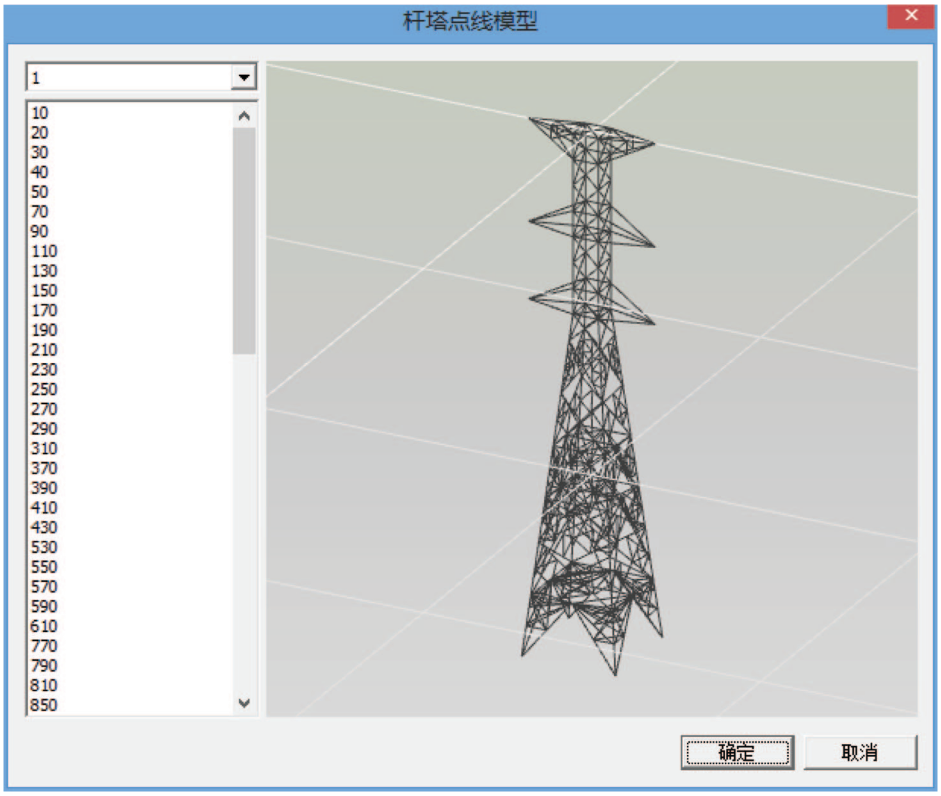
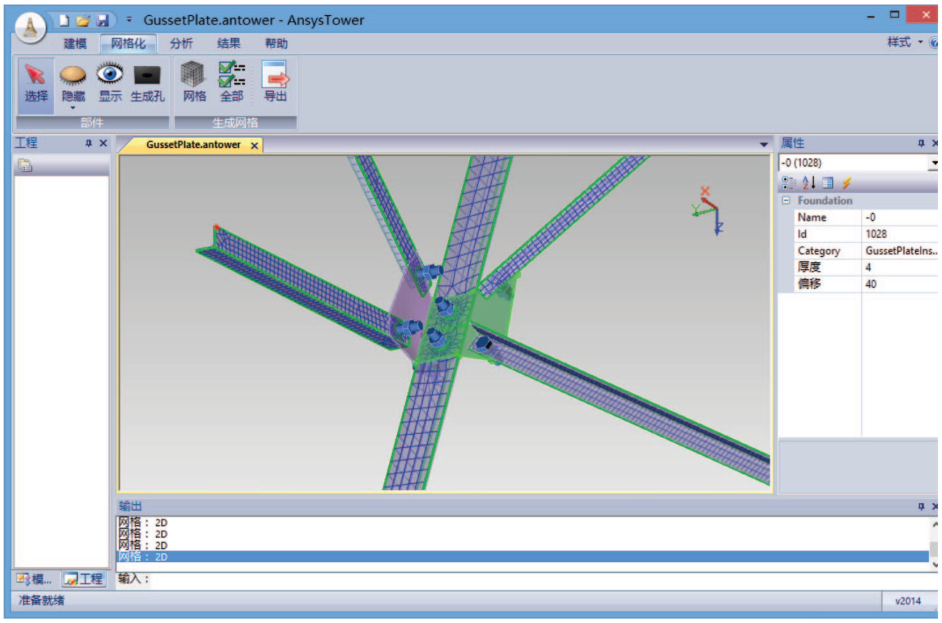


Figure 10: One dimensional model of a transmission tower



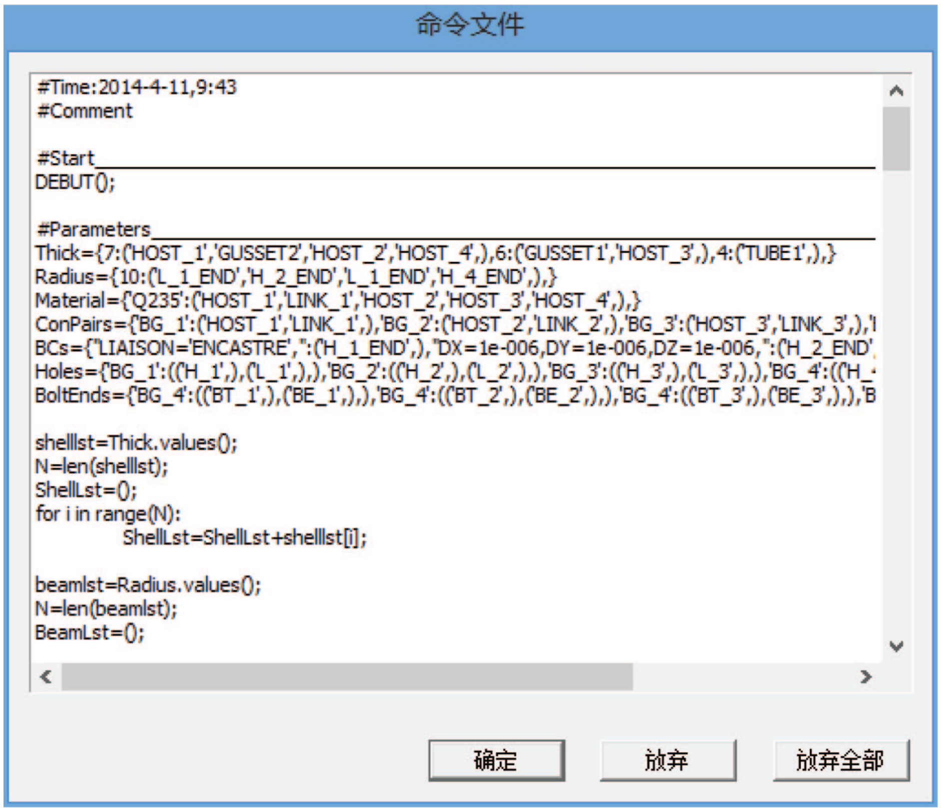
-0 (1028)

Foundation

Name	-0
Id	1028
Category	GussetPlateIns..
厚度	4
偏移	40

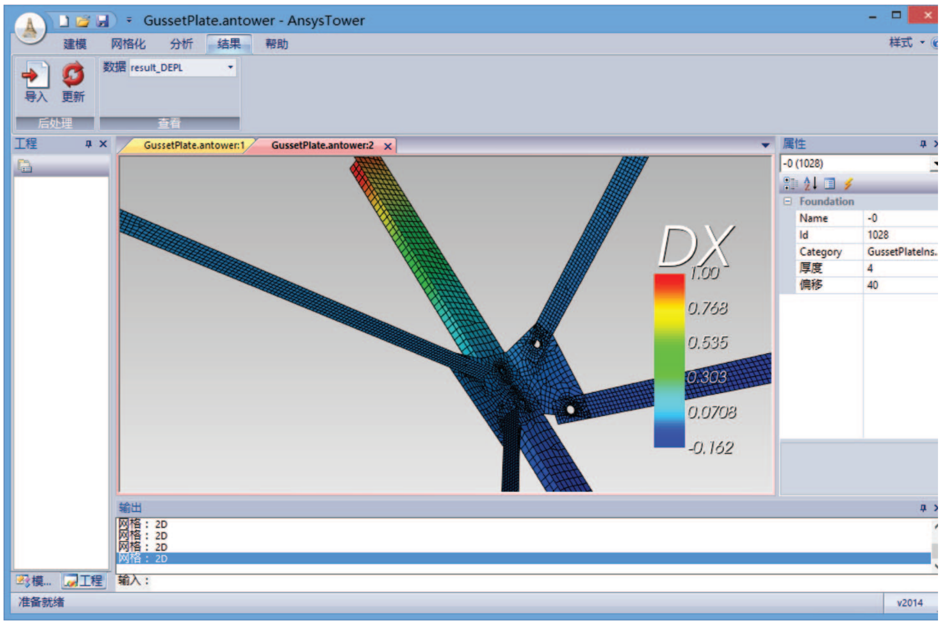
```
网格：2D
网格：2D
网格：2D
网格：2D
```

Figure 11: The generated shell mesh of an assembled joint of a transmission tower



```
#Time:2014-4-11,9:43
#Comment

#Start_____________________________________________
DEBUT();

#Parameters________________________________________
Thick={7:('HOST_1','GUSSET2','HOST_2','HOST_4',),6:('GUSSET1','HOST_3',),4:('TUBE1',),}
Radius={10:('L_1_END','H_2_END','L_1_END','H_4_END',),}
Material={'Q235':('HOST_1','LINK_1','HOST_2','HOST_3','HOST_4',),}
ConPairs={'BG_1':('HOST_1','LINK_1',),'BG_2':('HOST_2','LINK_2',),'BG_3':('HOST_3','LINK_3',),'I
BCs={"LIAISON='ENCASTRE'',":('H_1_END',),"DX=1e-006,DY=1e-006,DZ=1e-006,":('H_2_END',
Holes={'BG_1':(('H_1',),('L_1',),),'BG_2':(('H_2',),('L_2',),),'BG_3':(('H_3',),('L_3',),),'BG_4':(('H_
BoltEnds={'BG_4':(('BT_1',),('BE_1',),),'BG_4':(('BT_2',),('BE_2',),),'BG_4':(('BT_3',),('BE_3',),),'B

shelllst=Thick.values();
N=len(shelllst);
ShellLst=();
for i in range(N):
        ShellLst=ShellLst+shelllst[i];

beamlst=Radius.values();
N=len(beamlst);
BeamLst=();
```

Figure 12: A piece of the generated input file



-0 (1028)

Foundation

Name	-0
Id	1028
Category	GussetPlateIns.
厚度	4
偏移	40

```
网格：2D
网格：2D
网格：2D
网格：2D
```

Figure 13: The displacement field of the assembly displayed in a viewport

5. PRESPECTIVES AND FUTURE DEVELOPMENTS

Benefiting much from open-sourced resources, the extensile framework will be released as another open-sourced code in the future when the documentation is compiled and the application programming interfaces (APIs) are further standardized. We believe an open-source strategy will speed up the development and deployment of the framework in the community of numerical simulations.

The request for smart simulation software is evolving rapidly and many new developments are planned for the near future: advanced boundary-layer capabilities for computational fluid dynamics (CFD) applications, parallel meshing, more advanced feature recognition, remote job monitoring—to name a few. In virtue of the good modularity of the framework, these new functionalities can be integrated into the framework straightforwardly.

REFERENCES

[1] National Science Foundation (NSF) Blue Ribbon Panel (2006). Report on simulation-based engineering science: revolutionizing engineering science through simulation. NSF Press, May.

[2] Moldflow, http://www.autodesk.com/moldflow

[3] ProCAST, https://www.esi-group.com/software-services/virtual-manufacturing/casting

[4] MAGMA, http://www.magmasoft.com

[5] Cai,Y., Li, G.Y., Wang, Zheng, H.G. and Lin, S. Development of parallel explicit finite element sheet forming simulation system based on GPU architecture. Advances in Engineering Software 45 (1), 370–379 (2012).

[6] Autoform, http://www.autoform.com

[7] Wang, C. G. Integrated aerodynamic design and analysis of turbine blades. Advances in Engineering Software 68, 9–18 (2014).

[8] Lin, H., Louise, P. B. and Andrew, C. L. Modelling and simulating textile structures using TexGen. Advanced Materials Research 331, 44–47 (2011).

[9] Williams, R. K., Amberiadis, K. and Angst, D. Smart-power devices seek wiser CAE tools. IEEE Circuits and Devices Magazine 7, 20–25 (1991).

[10] Wang, H., Patil, V., Resh, W., Insalaco, P., Flesher, D. and Lanski, S. Smart meshing template process with CAD/CAE link, SAE Technical Paper 2013-01-0637.

[11] Yamada, T., Kushida, N., Araya, F., Nishida, A. and Nakajima, N. Component-wise meshing approach and evaluation of bonding strategy on the interface of components for assembled finite element analysis of structures. Key Engineering Materials, 452–453, 701–704 (2011).

[12] Nunio, F. and Manil, P. SALOME as a platform for magneto-mechanical simulation. IEEE Transactions on Applied Superconductivity 22, 4904904 (2012).

[13] Salomé, The open source integration platform for numerical simulation, http://www.salome-platform.org

[14] Ribes, A. and Caremoli, C. Salomé platform component model for numerical simulation. in COMPSAC'07: Proceedings of the 31st Annual International Computer Software and Applications Conference, (Washington, DC, USA), pp. 553–564, IEEE Computer Society, 2007.

[15] OpenCASCADE, http://www.opencascade.org/

[16] Schöberl, J. NETGEN, an Advancing front 2D/3D-mesh generator based on abstract rules. Computing and Visualization in Science 1, 41–52 (1997).

[17] Paraview, Open Source Scientific Visualization, http://www.paraview.org/

[18] Gayer, M. and Iannaccone, G. A Software platform for nanoscale device simulation and visualization. ACTEA 2009 (Zouk Mosbeh, Lebanon), pp. 432–437.

[19] Document/View Architecture, http://msdn.microsoft.com/en-us/library/4x1xy43a.aspx

[20] MED data model. http://www.code-aster.org/outils/med/

[21] Thompson, J. F., Thames, F. C. and Mastin, C. W. Automatic numerical generation of body-fitted curvilinear cordinate system for field containing any number of arbitrary two-dimensional bodies. Journal of Computational Physics 15, 299–319 (1974).

[22] Thompson, J. F. A general three-dimensional elliptic grid generation system on a composite block structure. Computer Methods in Applied Mechanics and Engineering 64, 377–411 (1987).

[23] Dassault Systèmes Simulia Corp. 2012. Abaqus Analysis User's Manual, V6.12.

[24] Rich Text Format, http://en.wikipedia.org/wiki/Rich_Text_Format

[25] VTK-The Visualization Toolkit. http://www.vtk.org/

[26] wxWidgets, cross-platform GUI library. http://wxwidgets.org/

[27] Harris, T. A. Rolling bearing analysis, 3rd ed. Wiley: New York, 1991.

[28] Code-Aster, A multi-physics finite element code for structural mechanics, http://www.code-aster.org/, V10.3, 2012.