17th International Conference on the Application of Computer
Science and Mathematics in Architecture and Civil Engineering
K. Gürlebeck and C. Könke (eds.)
Weimar, Germany, 12–14 July 2006

A FRAMEWORK FOR THE INTERACTIVE VISUALIZATION OF ENGINEERING MODELS

M. Baitsch[*], **and D. Hartmann**

[*]*Institute for Computational Engineering*
Ruhr-University of Bochum
Bochum, Germany
E-mail: matthias.baitsch@ruhr-uni-bochum.de

Keywords: Visualization, Engineering Models, Object-Oriented Framework, Java, VTK.

Abstract. *This contribution introduces a framework that can be used as a basis for the development of interactive applications for the visualization of engineering models. Implemented in Java, the framework defines the application logic needed to translate model and simulation data into a three-dimensional graphical representation. Several high-level classes are provided that can be used to visualize e.g. structural elements, two- and three-dimensional continua and boundary conditions just by providing the data as required. New classes can easily be added if some required functionality is missing.*

1 INTRODUCTION

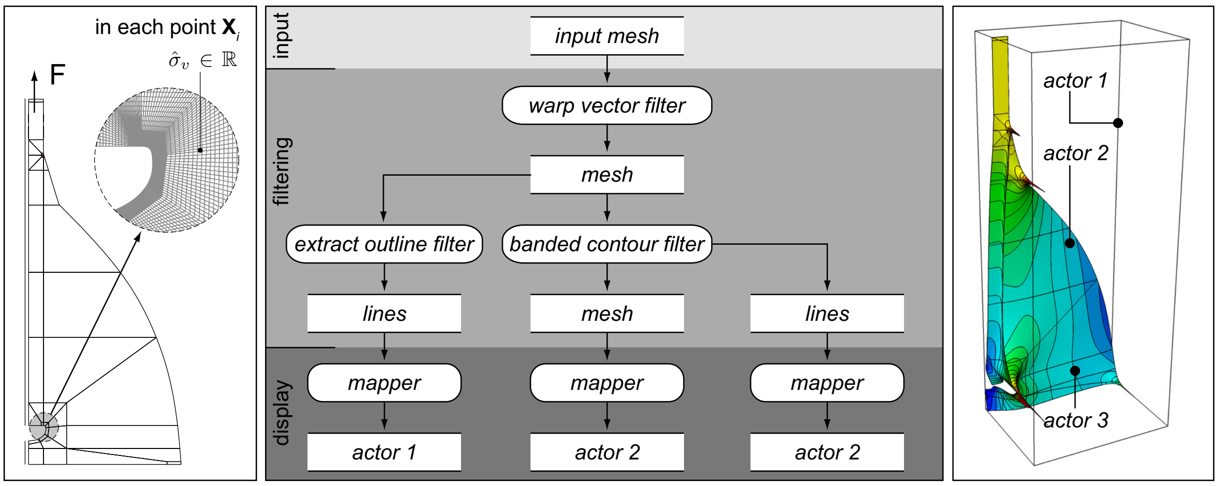
Interactive visualization based on 3D computer graphics nowadays is an indispensable part of any simulation software used in engineering. Although ready made solutions like Ensight, Tecplot or OpenDX provide a huge functionality, they can hardly be incorporated into own applications. On the other side, general purpose libraries for scientific data visualization require lots of programming before engineering models can be adequately visualized because they provide only basic functionality.

The presented Java framework for the interactive visualization of engineering models is built on top of the open source visualization toolkit VTK [7] which is widely used in various fields of scientific data visualization. VTK is a collection of more than 2000 C++ classes. Wrappers of the C++ classes exist for various other programming languages including Java.

Our framework tailors VTK to engineering applications on two levels. On the first level it adds new – engineering specific – filter classes to VTK. For instance, a class for the visualization of truss and beam elements has been developed. On the second level, ready made pipeline layouts for certain aspects of engineering models are provided.

2 THE VISUALIZATION TOOLKIT VTK

For the reader who is not familiar with VTK, this section briefly introduces the most basic ideas behind it. In VTK, data is visualized by so called visualization pipelines which basically connect data objects, filters, mappers and actors. Data objects contain geometrical (points), topological (cells) and attribute (scalars, vectors, tensors) data. Using a filter, one data object is transformed into another data object by applying a certain algorithm. Many of the VTK classes are filters including e.g. a classes to generate deformed geometry, to extract contour lines, etc. At the end of each thread in a pipeline, a mapper converts a data object into graphics primitives that are displayed on screen by an actor.



(a) input mesh (b) visualization pipeline (c) actors on screen

Figure 1: A pipeline for the visualization of equivalent stress in a plate

An example of a pipeline visualizing the equivalent stress of a connection plate computed with a high order finite element analysis is shown in Figure 1. On the left hand side is the mesh serving as input data. Using the pipeline displayed in the middle, the image on the right hand side is generated. In the diagram, oval blocks represent filters that operate on data, and the rectangular blocks represent data stores. Inside of the pipeline, first a *warp vector filter* is used to translate the mesh points along the z-axes in order to generate the 3D plot of the stress. The *extract outline filter* is responsible for computing the outlining box. With the *banded contour filter*, the uniformly colored areas and the contour lines are provided. Each resulting data object is rendered to screen by a mapper-actor pair.

Despite the large number of filters available in VTK, one-dimensional structural elements which are typical for many engineering applications are not well supported.

3 SPECIALIZED FILTERS FOR ENGINEERING MODELS

In order to support the visualization of engineering models, several new filter classes have been added to VTK. New filters can be implemented in Java although VTK is C++ software. In VTK this is realized by invoking the Java method that contains the visualization algorithm through the Java Native Interface (JNI).

For many structures it is suitable to idealize the original three-dimensional continuum with one-dimensional elements whereas the two missing dimensions are represented by a cross-section. For this purpose, the line elements filter has been developed. This filter takes as input the nodes (representing the geometry of the system) and the elements which connect nodes (representing the topology). Each element also has an associated local y-axes which is used to orient the sections in space. In addition, the outlines of the cross-sections have to be provided. Using that information, the filter extrudes the correctly oriented outline along the elements. In order to visualize forces, moments or stresses in one-dimensional elements, the line plot filter is used. Based on the mesh, the geometry and scalar quantities associated with the cells, this filter generates a surface along a certain direction. Finally, a third new filter is used to generate symbols representing attributes of points like displacement constraints, forces or moments. The application of the new filter classes is shown in Figure 2.

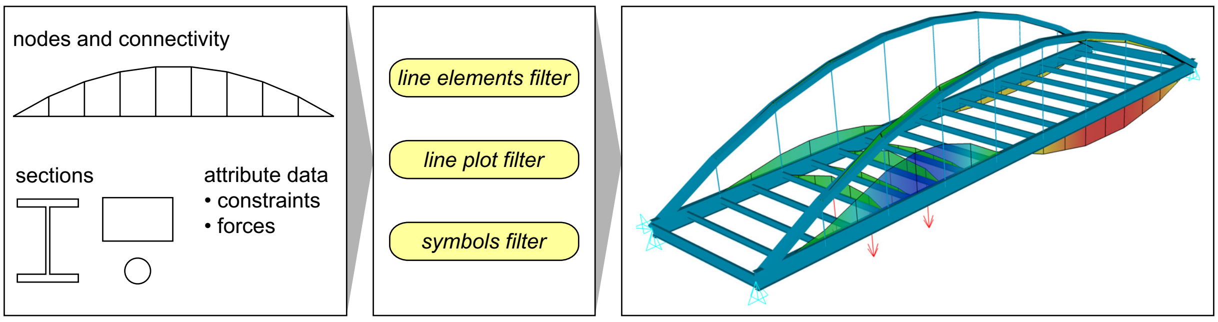


Figure 2: Additional filter classes for engineering models

With the existing and the new filter classes, visualization pipelines can be established that efficiently visualize a wide variety of engineering models. Often, such pipelines involve ten or more filter objects and are difficult to design, implement and test. Reusing pipeline layouts therefore is highly desirable.

4 MANAGING VISUALIZATION PIPELINES

Engineering models like finite element models, multibody systems or computational fluid dynamics share many similar features. This is mainly because the underlying concepts are based on partial differential equations in combination with some discretization technique. Typically, such models involve nodes whereas at the nodes there are several types of attributes like forces or constraints defined. If a model consists of line-like elements, these are usually described by a center line and a cross section. Two-dimensional and three-dimensional continua are conveniently represented by the surface. Simulation results are visualized by coloring or deforming the surface or by placing symbols on the surface if vector or tensor quantities should be visualized.

The basic idea of the present contribution is a framework to define visualization pipelines that each visualize one aspect of the actual model. Such pipelines can then be used in different applications for different kinds of models. Various ready-made pipelines for common aspects are already incorporated and furthermore, the framework can easily be extended with additional pipeline classes for special purposes. A pipeline represents a certain view on the model that visualizes one specific aspect. Data provider objects serve as adapter between the actual model and a pipeline.

Figure 3 shows the most important classes of the framework. For clarity, method signatures are omitted. In the developed concept, the most fundamental abstraction is the *Pipeline* interface which specifies operations for objects that encapsulate a VTK visualization pipeline like the one in Figure 1(b). As input, the model for which a certain aspect should be visualized is specified using the *setModel* method. Because no assumptions about the model can be made in advance, the parameter is of type *Object*. A pipeline object provides several actors that are individually accessible and can be collectively added to a render window. The *PipelineCollection* class is based on the composite pattern [3] and conveniently collects an arbitrary number

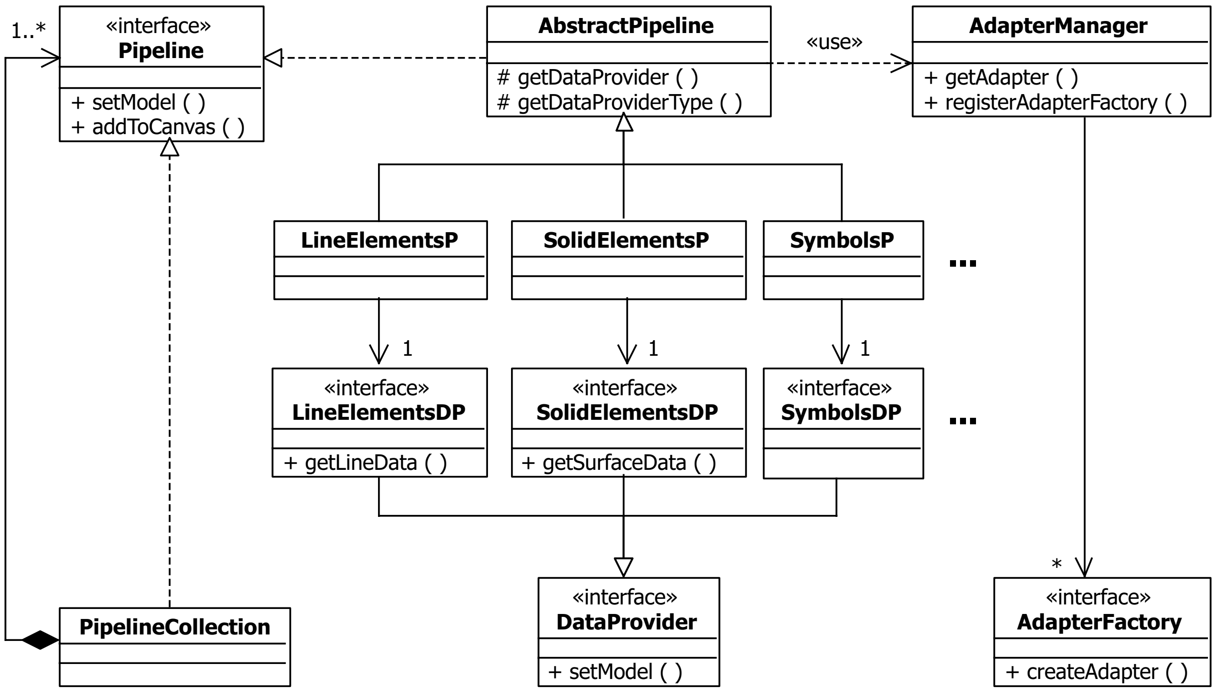


Figure 3: UML class diagram of pipeline package

of pipelines in one single object. A *PipelineCollection* is constructed by specifying an instance of the *PipelineFactory* (not shown in the diagram) class providing the pipeline objects required for a specific model.

The flexibility of the presented approach lies in the *AbstractPipeline* class. It is based on the idea, that for each of its subclasses, there exists a corresponding subinterface of *DataProvider* which specifies the format of the data that is needed by the pipeline. The three concrete classes *LineElementsP*, *SolidElementsP* and *SymbolsP* in the center of the diagram are examples for such *Pipeline* classes. Below each class, the corresponding *DataProvider* interface is shown. In order to get an instance of the *DataProvider* interfaces for the actual model, the *AdapterManager* class is used. Here, classes for a specific model type register an *AdapterFactory* object for a certain pair of a model type and a data provider type. At runtime, the *AbstractPipeline* uses this information to obtain the object that provides the data currently needed. Not explicitly shown in the diagram is that the observer pattern is applied to the pipeline as well as data provider. While observing the data provider, a pipeline chains corresponding events to observers of the pipeline.

In addition, ready made Graphical User Interface (GUI) components are provided in the framework. With these components, an application that allows the user to determine the appearance of the model interactively can quickly be established.

Using the framework for a specific type of model requires the following steps. First, additional filters or pipeline classes that may be necessary to visualize special aspects have to be implemented. Next, a *PipelineFactory* is specified that creates the pipeline objects needed. For each pipeline, an adapter has to be established that generates the data representing the actual model in the format specified by the interfaces derived from *DataProvider*. Finally, the interactive application is created using the ready-made GUI components.

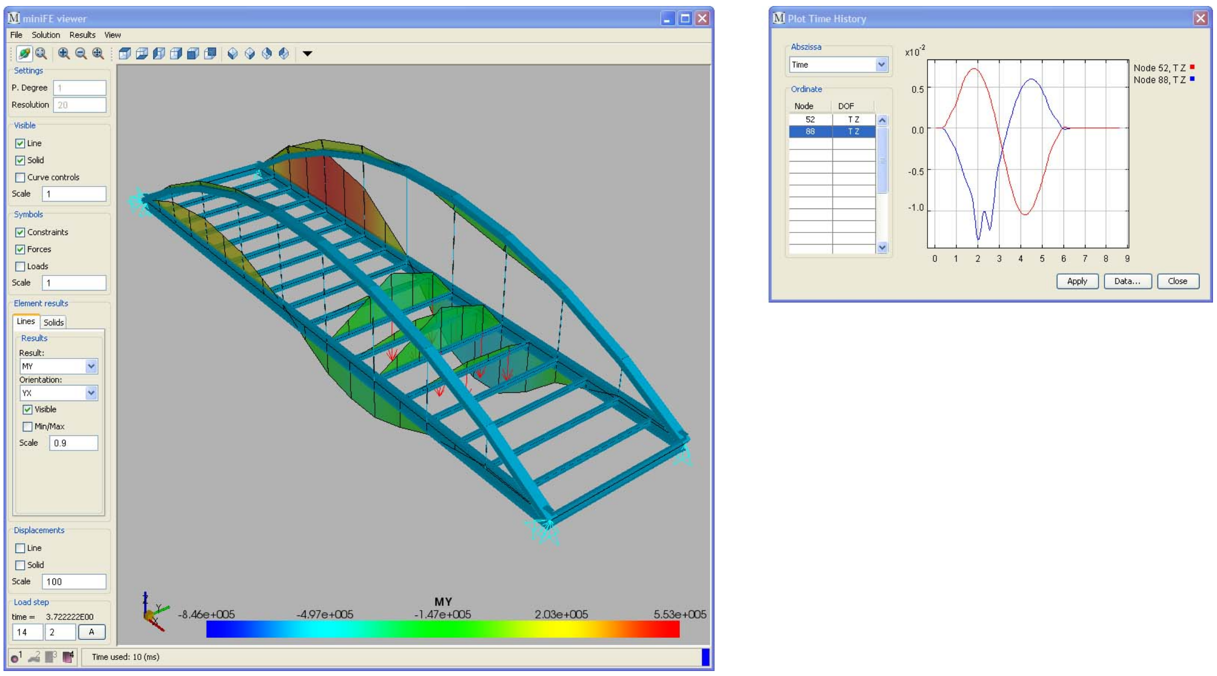
5 APPLICATIONS OF THE FRAMEWORK

The presented visualization framework is used in various research projects at the Institute for Computational Engineering at the Ruhr-University of Bochum. For each project, we briefly explain how the software is incorporated into the overall concepts.

5.1 Object-oriented finite element system

The framework emerged firstly in the context of an object-oriented finite element system developed by the authors [1]. The system supports both the h-version and the p-version of FEM. It can be used for static and dynamic analysis taking into account non-linear effects. Applications include research in lifetime-oriented design optimization [2] and teaching.

Based on the presented framework, a software component has been developed that is used to visualize various types of finite element models (truss and beam structures, two-dimensional and three-dimensional continua) and to monitor optimization processes. The visualization component consists of several packages. A set of adapters provide data for the pipelines that each visualize a certain aspect of the model. In order to generate the surface data for the different types of continuous elements (plane, plate, solid), an adapter mechanism similar to the one described above is used. Other adapters generate data representing nodes, forces or constraints etc. The graphical interface for the user consists of a main frame that displays the structure and directly gives access to the most important visualization options. Additional dialogs provide more detailed access or information.

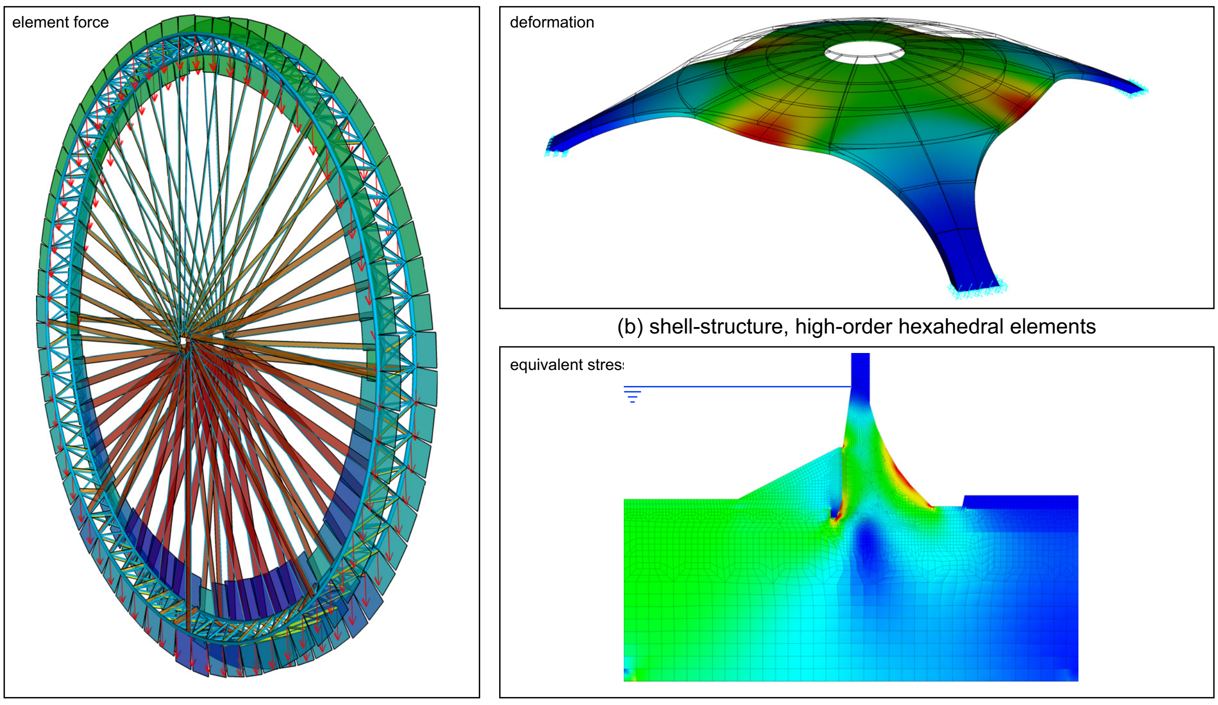


Settings
P. Degree 1
Resolution 20
Visible
Line ✓
Solid ✓
Curve controls
Scale 1
Symbols
Constraints ✓
Forces ✓
Loads
Scale 1
Element results
Lines Solids
Results
Result: MY
Orientation: YX
Visible ✓
Min/Max
Scale 0.9
Displacements
Line
Solid
Scale 100
Load step
time = 3.722222E00
14 2 A
Time used: 10 (ms)

Abszissa
Time
Ordinate

Node	DOF
52	T Z
88	T Z

Figure 4: Viewer application in the transient analysis of an arch-bridge



(a) ferris-weel, prestressed cable elements

(b) shell-structure, high-order hexahedral elements

(c) two-dimensional model of a masonry dam

Figure 5: Visualization examples

An application of the viewer for the transient analysis of an arch-bridge is shown in Figure 4. The three-dimensional view in the center shows the structure along with the bending moments. Additionally, the time variant behavior of the structure could be displayed in an animation. The dialog on the right shows the time-variant displacement of two selected nodes. Further examples for the visualization of different types of structures are shown in Figure 5.

5.2 Agent based monitoring of dam structures

The second application example stems from a research project on an agent-oriented approach for the computer-based monitoring of dam structures [6]. In this project, the multi-agent system approach is chosen because, by that, the monitoring problem can be modeled in a natural fashion and implemented adequately on a computer network. Furthermore, existing conventional solution methods (e.g. computation programs) can efficiently be integrated through so-called wrapper agents. Figure 6 shows how existing finite-element applications are incorporated into the application using a wrapper-agent.

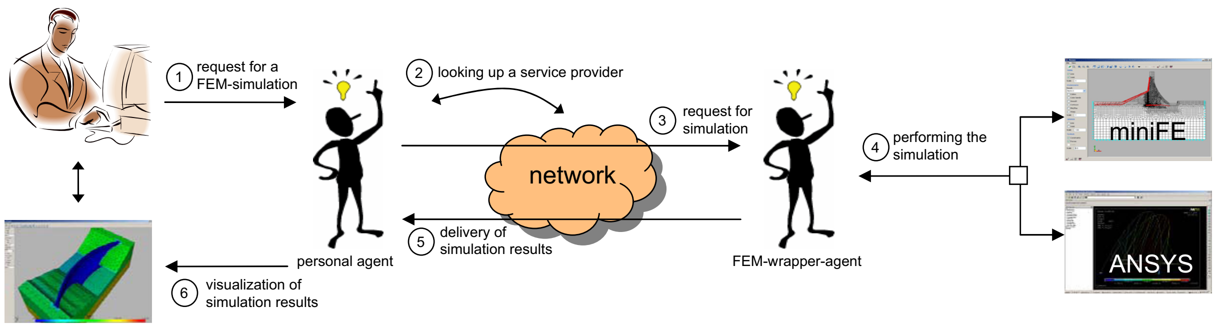

1 request for a FEM-simulation
2 looking up a service provider
3 request for simulation
network
4 performing the simulation
miniFE
ANSYS
5 delivery of simulation results
personal agent
FEM-wrapper-agent
6 visualization of simulation results

Figure 6: Agent based FEM-simulation (from [5])

In this project, the visualization framework is used to visualize the structure analyzed by the FEM-wrapper agent on the workstation of the user. Here, adapters that read the result data returned by the agent are used in order to provide the data for the individual pipeline objects. The result of a simulation using a three-dimensional model based on quadratic tetrahedral elements is shown in Figure 7.

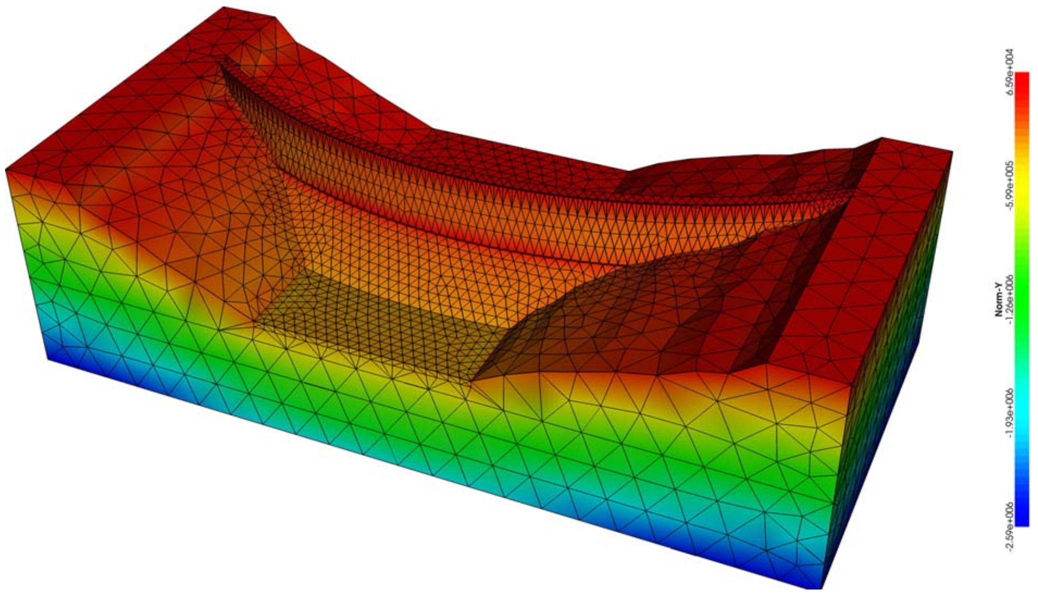
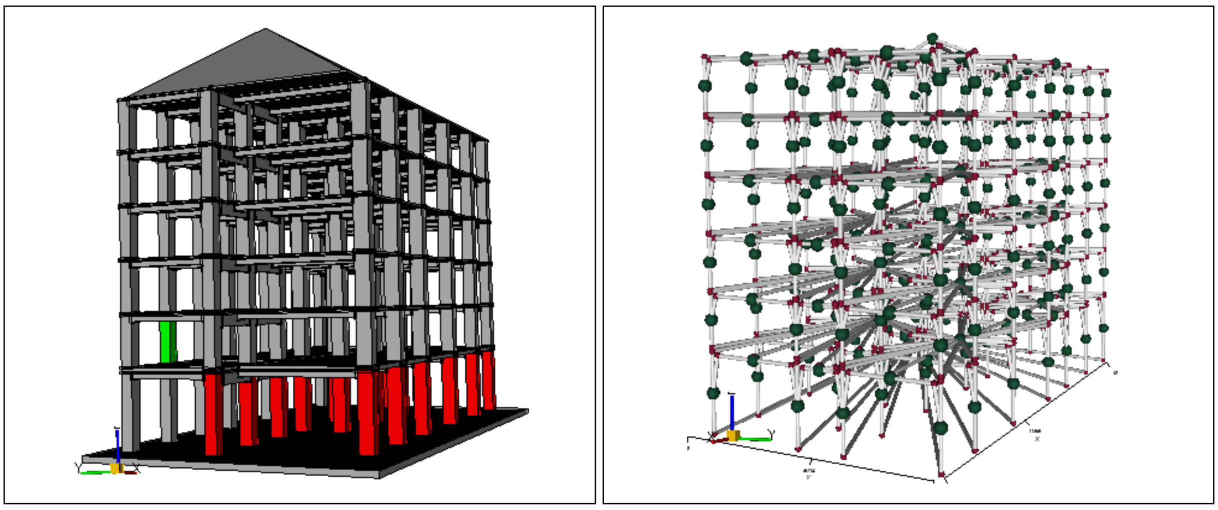


Figure 7: Visualizing the results of a simulation with ANSYS

5.3 Simulation of blasting processes based on multibody dynamics

The presented framework is also used in the simulation of blasting processes [4]. Here, a realistic and efficient simulation requires a powerful simulation model that covers the entire complex dynamic process that is evoked by the ignition of the loads and ends with the collapse of the building. The concept of the simulation model is therefore based on a multi-level model of the blasting process that comprises three main levels: On the first level (local level) the effects of the exploding charges are modeled such that the volitional damages can be captured and described. On the second level (near field level) the effects of the local damages on the adjacent structure are analyzed. Based on the first and second level, finally the collapse of the entire structure is modeled on the third level (global level) including fracture processes and relevant contact mechanisms.



(a) blasting-oriented product model

(b) multibody model

Figure 8: Visualization of various submodels involved in blasting simulation (from [8])

The developed simulation model on the global level introduces different submodels, each responsible for different aspects of the simulation procedure. The blasting-oriented product model (Figure 8(a)) serves as a database and contains all relevant data needed for the global level simulation. The core physical model for the simulation model on the global level is based on a special multibody system that is created adaptively during the simulation process (Figure 8(b)). In this project, the presented framework is used to visualize all the involved submodels.

6 CONCLUSION

The framework presented in this contribution provides the application logic needed to implement an interactive application for the visualization of engineering models. Much of the achieved flexibility is based on the concept of pipeline objects visualizing a certain model aspect in conjunction with data provider interfaces that specify the format of the required data. Because the adapters to the data providers are determined at runtime, the system can also be used to build generic visualization applications for which the user just has to supply the adapters for the actual model. The variety of the application examples show the versatility of the framework.

REFERENCES

[1] M. Baitsch and D. Hartmann. Object-oriented finite element analysis for structural optimization using p-elements. In K. Beucke, B. Firmenich, D. Donath, R. Fruchter, and K. Roddis, editors, *X. ICCCBE-Digital conference proceedings*, Weimar, 2004.

[2] M. Baitsch and D. Hartmann. Towards lifetime optimization of hanger connection plates for steel arch bridges. In K.J. Bathe, editor, *Third MIT Conference on Computational Fluid and Solid Mechanics*, pages 1213–1217, Cambridge, 2005. Elsevier Science.

[3] M. Grand. *Patterns in Java*. Wiley, 2002.

[4] S. Mattern, G. Blankenhorn, M. Breidt, V. Nguyen, S. Höhler, K. Schweizerhof, D. Hartmann and F. Stangenberg. Comparison of building collapse simulation results from finite element and rigid body models. In *Multiscale Problems in Multibody System Contacts*, Stuttgart, February 20. - 23. 2006. IUTAM Symposium.

[5] A. Marx. Entwicklung eines Software-Agenten zur Integration von FEM-Software in ein agentenbasiertes Talsperren-Monitoring-System. Diplomarbeit, Lehrstuhl für Ingenieurinformatik im Bauwesen, Ruhr-Universität Bochum, 2006.

[6] I. Mittrup. Structural monitoring of dams with software agents. In *Int. Conference on Computing in Civil Engineering, Cancun 2005*. ASCE (American Society of Civil Engineers), 7 2005.

[7] W. Schroeder, K. Martin, and B. Lorensen. *The Visualization Toolkit*. Prentice-Hall, 1998.

[8] R. Schütz. Entwicklung interaktiver Visualisierungskomponenten zur Darstellung von Ingenieurmodellen mit vtk und deren Integration in ein Eclipse-basiertes Simulationsframework zur Analyse komplexer Bauwerk-Sprengungen. Diplomarbeit, Lehrstuhl für Ingenieurinformatik im Bauwesen, Ruhr-Universität Bochum, 2006.