Education in Finite Element Analysis and Applications in Computational Grid Networks

Rimantas BARAUSKAS
Kaunas university of technology, Dept. System analysis,
Studentu 50-407, 3031 KAUNAS, Lithuania
rimantas.barauskas@ktu.lt

Finite element analysis (FEA) is the most popular technique for presenting the behavior of physical systems in "virtual reality". At present time it is commonly realized as a *technology of the physically based modeling* rather than the numerical finite element method carrying the name and comprising the mathematical kernel of the technology. The distinguishing feature of education in FEA lies in its interdisciplinary nature. Computational grids are offering essential profits and opportunities in teaching, research, as well as, performing practical computations and by integrating FE applications into the design and development of real engineering objects and systems. The advance in unification of educational course materials, visualized examples, easy access to up-to-date software systems and libraries can be regarded as several expected goals of the above mentioned prospective efforts. This work presents the basic ideas of structure and implementation of educational tools of FEA into computational grid environments.

Finite element method, computational grid, education

1. INTRODUCTION

Grids are "super Internets" for high-performance computing: worldwide collections of high-end resources such as supercomputers, storage, advanced instruments and immersive environments. These resources and their users are often separated by great distances and connected by high-speed networks. Grids bring together geographically and organisationally dispersed computational resources, such as CPUs, storage systems, communication systems, real-time data sources and instruments, human collaborators [1].

According to IBM's definition, "a Grid is a collection of distributed computing resources available over a local or wide area network that appear to an end user or application as one large virtual computing system. The vision is to create virtual dynamic organizations through secure, coordinated resource-sharing among individuals, institutions, and resources. Grid computing is an approach to distributed computing that spans not only locations but also organizations, machine architectures and software boundaries to provide unlimited power, collaboration and information access to everyone connected to a Grid." [2]. Grids are new: many of the enabling technologies have not yet been invented.

Finite element analysis (FEA) is the most popular techniques for presenting the behavior of physical systems in "virtual reality". At present time it is commonly realized as a *technology of the physically based modeling* rather than the numerical finite element method carrying the name and still comprising the mathematical kernel of the technology. At present time the FEM codes are able to cope with wide spectra of problems (nonlinear, coupled, multi-scale, multi-physics). However, the codes involving adaptive meshes, multi-level solvers, parallel and distributed execution are rather complex and difficult to interfere and to modify. Learning curves for users are steep and generally the finite element modeling is close to a professional activity based on deep knowledge of the FEA mathematical background, thorough understanding of the engineering essence of the problem as well as, experience in using FEA codes.

As FEA has already achieved high level of perfection, many commercial systems are available. At present time in Lithuanian universities ANSYS, ALGOR, COSMOS, ABAQUS, DIANA, LSDYNA, AUTODYN, FEMLAB are used. Of importance are the data interfaces between the FEA systems and various pre- and post processing tools (TrueGrid, PATARAN) and CAD systems (CATIA, SolidWorks, AutoCAD, CADKEY). Generally, they are not simply alternative programs used for solving the same tasks. Though designed basing upon the same basic finite element scheme, they provide many unique features and are oriented for different classes of engineering problems. At present time, data interfaces are elaborated and, in principle, the data can be "transported" among different programs. However,

the interaction level is still far below the real functional unification, and only the analyst is responsible to decide if the whole computation process has been organized correctly. The more advanced implementations have been performed in mechanical design systems (e.g., CATIA, Microstation, etc.), where linear elasticity and field problem solution processes (meshing-> FE model generation –> loading –> solution -> post-processing) nowadays are fully integrated into corresponding CAD systems.

This work regards the finite element method (FEM) code as a grid application. Grid environment from application developer prospective offers such attractive features as open environment (coupling with external modules via standardization of interfaces), high performance communication libraries, simpler code development, cooperation with other developer groups and broad user base [3].

2. MAIN FEATURES OF EDUCATION IN FEA

The Finite Element Analysis (FEA) is one of the main disciplines of Computational Science and Engineering (SCE) study program taught at many universities. CSE is the multi-disciplinary field of computer-based modeling and simulation for studying scientific phenomena and engineering designs [4]. It requires methods from computer science, applied mathematics, and the respective application fields. Moreover, it often demands the use of high performance parallel computers able to cope with the difficulty and size of the problems. It is *different from Computer Science.* Computer Science focuses on topics that are essential to build computer hardware and software, as well as, complete information systems. Typical topics of computer science include:

- computer architecture and network architecture;
- operating systems, data bases, software engineering, etc.;
- finding fast algorithms for core problems (sorting, searching, etc.) .

Thus, computer science could be described as the art of building and using computers and information systems. In contrast, Computational Science and Engineering means *using computational methods to solve specific problems in science and engineering*. It provides a basic education in the field of CSE to be able to follow an academic or professional career in this field.

The distinguishing feature of education in FEA lies in its interdisciplinary nature including the following basic branches of knowledge:

- The basics of the *numerical finite element method* in application for different types of differential equations describing different physical environments;
- The *solution approaches and strategies* for real engineering problems;
- The *general numerical methods* and algorithms used in FEA;
- *Computer implementation* including pre- and post-processing, results visualization, etc.

Computational grid environments are offering essential profits and opportunities in teaching, research, as well as, performing practical computations and by integrating FE applications into the design and development of real engineering objects and systems. The advance in unification of educational course materials, visualized examples, easy access to up-to-date software systems and libraries can be regarded as several expected goals of the above mentioned prospective efforts.

Educational tools of FEA implemented into computational grid environments should consist of:

- Common education-through-distance techniques (course materials, examples, problems and solutions bases, reference materials to accompanying courses);
- The "numerical laboratory" base enabling to implement the necessary computational schemes and strategies – from simple examples during undergraduate education until complex schemes used in advanced courses and post-graduate research. As generally the implementations from "zero level" are time consuming and require numerous procedures, that can be prepared in advance, the grid should ensure the access to necessary software packages, function libraries, compilers and visualization tools. The desired prospective is to ensure the student or user to have the possibility with minimum routine work to design complex computational schemes using non-linear materials, elements, contact interactions comparable to the ones used in commercial software packages. At the same time the researcher may have much higher level of flexibility in "non-standard" applications.
- The access to commercial FE software and license management.

3. THE STRUCTURE OF THE FEA COMPUTATIONAL GRID ENVIRONMENT

Conceptually the learning-oriented FEA software is not essentially different from the FEA application environment used for solving practical real engineering problems. The characteristic feature of

the present situation is the availability of many libraries and program codes, multi-level solvers, as well as, parallel versions of FEA codes. Among them are commercial software (ANSYS, LSDYNA) having their user interfaces, the mathematically oriented computational environments MATLAB + FEMLAB and many case oriented programs and libraries in C++, FORTRAN, etc.

3.1 Virtual organizations

In education process, a lot of students, post graduates, teachers and researchers come together to create and use the FEA software in order to implement their specific tasks. A number of participants want to share resources in order to perform some task. Furthermore, sharing is more than file or data exchange: it can involve direct access to remote software, computers, data and other resources. This sharing is highly controlled, with resource providers and consumers defining clearly just what is shared, who is allowed to share, and the conditions under which sharing occurs. A set of individuals and/or institutions defined by such sharing rules form what is called a *virtual organization* (VO) [5].

One of possible divisions of users into actual and virtual organizations is presented in Fig.1.

VO1: *FEA performed by using commercial FE software*. The usual practice in using the commercial FEA software is to present a problem in terms of a script written in corresponding input language of a program (e.g., APDL for ANSYS, keyword files for LSDYNA, etc.). The grid services should allow running scripts on remote hosts (may be, on efficient parallel computers or clusters) and the necessary transport of input, as well as, results data;

VO2: *FEA by implementing computational processes in mathematically oriented environments (MATLAB+FEMLAB) and by means of programming in high-level languages (C++, FORTRAN).* This way of problem solution is necessary for "non-standard" applications, where using commercial codes is inconvenient or cumbersome. Furthermore, the approach is highly advised for education in FEA in order to enable the student to implement some steps of the FEA procedure or to design a computational strategy for a given research problem. Two levels can be distinguished in VO2:

- *Basic functions level* (element procedures, structure assembly functions, constraint equations, material (constitutive) models, etc.);
- *Computational strategy development level* (iterative schemes for non-linear problems, semi-analytical approaches, time integration schemes, structural fracture and failure processes, etc.).

VO3: *Usage of numerical and graphical libraries* is closely related with tasks performed in VO2 as the finite element method prescribes only the overall scheme of the solution. A number of accompanying numerical software: sparse solvers, array operations, data processing is still necessary to apply during the solution process. The visualization of results is also one of accompanying actions in any FEA computation.

VO4: *Learning through distance: access to electronic textbooks, tutorials and samples* can be regarded as usual distant learning tools. However, the inherent feature of education in FEA is the participation of each student to a certain extent in VO1, VO2, VO3 for acquiring necessary understanding and experience of practical computations.

There can be much more task groupings, however, here we do not intend to present all possible aspects of FEA and education directions in it.

The real users or organizations can be assumed as consisting of

"*Students*" – persons seeking to get basic or advanced education in FEA. Their primary tasks are to study educational materials and perform the necessary tasks given by the teacher. The students participate basically in VO4 and VO2, or they can participate in VO1 with limited user rights;

"*Post-graduate students*" and "*researchers*" form the user sets interested basically in performing practical analysis problems by using FEA software or in doing programming of complex analysis tasks. The rights of the two real user groups may be different depending upon licensed resources available.

Sharing relationships can vary dynamically over time, in terms of the resources involved, the nature of the access permitted, and the participants to whom access is permitted. The relationships do not necessarily involve an explicitly named set of individuals, but rather may be defined implicitly by the policies that govern access to resources. For example, an organization might enable access by anyone who can demonstrate that they are a "researcher" or a "student".

3.2 Grid architecture

The general layered grid architecture includes the following layers[5]:

The *Fabric* layer (the lowest one) provides the resources to which shared access is mediated by protocols: computational resources, storage systems, catalogs, network, etc. Fabric components implement the local, resource-specific operations that occur on resources (whether physical or logical) as a result of sharing operations at higher levels;

TABLE 1: The sample Grid services used to construct applications of Figure 1

	FEA using licensed commercial software	**FEA by implementing computational processes in mathematically oriented environments (MATLAB+FEMLAB),** **High-level programming languages (C++, FORTRAN)**	**Using numerical and graphical libraries** **(IMSL, OpenGL)**	**Learning through distance:** **access to electronic textbooks,** **tutorials and samples**
Collective **(application-specific)**	Input language service, source data and results transport	Function library organization and retrieval, multi-language compilation, access to visualization tools		-
Collective (generic)	Resource discovery, resource brokering, system monitoring, community authorization, certificate revocation, help and content information			
Resource	Access to computation; access to data; access to information about system's structure, state, performance			
Connectivity	Communication (IP), service discovery (DNS), authentication, authorization, delegation			
Fabric	Storage systems, computers, networks, code repositories, catalogs			

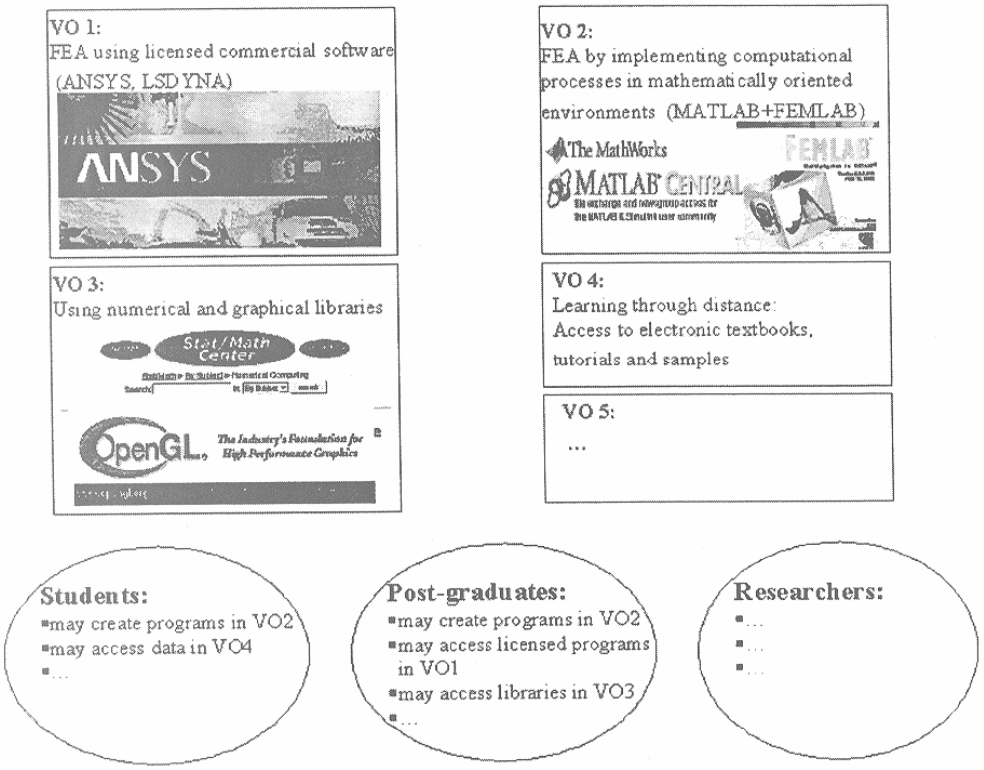

FIGURE 1: An actual organization can participate in one or more VOs by sharing some or all of its resources. We show actual organizations (the ovals), and VOs (the rectangles), which link participants in the finite element programming and analysis field in different kinds of activities.

The *Connectivity* layer defines core communication and authentication protocols required for grid-specific network transactions. Communication protocols enable the exchange of data between Fabric layer resources. Authentication protocols build on communication services to provide cryptographically secure mechanisms for verifying the identity of users and resources. Communication requirements include transport, routing, and naming;

The *Resource* layer builds on Connectivity layer communication and authentication protocols to define protocols, Application Program Interfaces (API) and Software Development Kits (SDK) for the secure negotiation, initiation, monitoring, control, accounting, and payment of sharing operations on individual resources. Resource layer implementations of these protocols call Fabric layer functions to access and control local resources. Resource layer protocols are concerned entirely with individual resources and hence ignore issues of global state and actions across distributed collections; such issues are the concern of the Collective layer discussed next;

The *Collective* layer contains protocols and services, APIs and SDKs that are not associated with any one specific resource but rather are global in nature and capture interactions across collections of resources. It implements a wide variety of sharing behaviors without placing new requirements on the resources being shared.

The final, *Applications* layer comprises the user applications that operate within a VO environment. Applications are constructed by calling upon, services defined at any layer. At each layer, there are well-defined protocols that provide access to some useful service: resource management, data access, resource discovery, etc.

4. CONCLUSION

The paper presents the idea and sample structure of implementation of Finite element education and analysis tools into computational grid environments. The main expected advantages are legal, easy and standardized access to up-to-date software systems and libraries and the integration of educational course materials with tutorials and examples at different levels of programming.

REFERENCES

[1] School of computing, University of Leeds, http://www.comp.leeds.ac.uk/cgi-bin/sis/ext/staff_pub.cgi/sarfraz.html?cmd=displaystaff#pagetop.

[2] L.J.Zhang, J.Y.Chung, Q.Zhou, Developing Grid computing applications, Part 1, Introduction of a Grid architecture and toolkit for building Grid solutions, http://www-106.ibm.com/developerworks/webservices/library/ws-grid1

[3] K.Banaś, J. Płażek, Concepts for implementing adaptive finite element codes for grid computing, Cracow Grid Workshop, 5-6 November, 2001.

[4] International Master's Program at the Technische Universität München, http://www.cse.tum.de/applying/faq.html

[5] I.Foster, C. Kesselman, S. Tuecke. The Anatomy of the Grid. Enabling Scalable Virtual Organizations, http://www-106.ibm.com/developerworks/grid/library/gr-fly.html#IDA5BGQB