# SCOPE a Framework of Objects to Develop Structural Analysis Programs in C++

Klaus Reimann[1], Lluís Gil[2], Michael Jentsch[2] and Montserrat Sánchez[2]

[1]Institute for Statics and Dynamics of Aerospace structures ISD
University of Stuttgart
Pfaffenwaldring 27, D-70569 Stuttgart. Germany.
reimann@isd.uni-stuttgart.de

[2]Department of Strength of Materials and Structures in Engineering.
Polytechnical University of Catalonia
Edifici C1, Campus Nord UPC, c/ Gran Capità s/n, 08034 Barcelona. Spain.
lluis@cimne.upc.es

**Abstract.** SCOPE stands for Structural Computations using an Object-oriented Programming and Engineering. The Finite Elements Method FEM has become the most popular technique for solving a wide range of complex engineering problems. In this paper are pointed out the advantages and drawbacks of object-oriented programming when this technique is used in the implementation of FEM programs. Moreover, it is described a framework for developing such kind of applications. The framework, called SCOPE, distributes in a web-based organisation all the objects and code, written in C++, that are necessary to build personal sized programs.

## 1. Introduction

Looking at history we can see that the world has always grown towards higher complexity. This becomes especially true in the field of computational mechanics and structural analysis. Modern numerical methods for computers can provide answers for problems with non-linear behaviour, optimisation, sensitivity analysis, etc. At the same time, numerical facilities are supported by graphical facilities that makes more friendly and comfortable the analysis and interpretation of results. Nowadays, engineers can design and test a lot of numerical prototypes saving time and money.

However, the task of programming the Finite Element Method FEM and his graphical interfaces have been increasing in difficulty along the years. The most the engineers want to compute the most it is difficult to implement. The reasons are several, from the language and programming strategies to the mathematical boundary problems. In this paper the authors present an overview of the difficulties for dealing with FEM programming and offer a solution for future developments in a web-based framework of objects.

## 2. The FEM and programming strategies

The FEM is, by now, the most common methodology for solving a wide range of engineering problems. It is the most used in structural problems, for discrete structures or for continuous ones, for linear and non-linear materials, etc. But the scope of the method goes far beyond and it is also used in thermal mechanics, fluids, electromagnetism and acoustics.

The majority of the engineering problems deal with large complex structures without having to apply error-producing simplifications. For that scenario the analytical solutions became unable to provide useful answers and, on the contrary, the FEM has become a successful computing technology and it has been able to achieve amazing results. Among the other numerical methods like finite differences, finite volumes or boundary elements, the FEM has become the most successful and has been capable of working with all the problems that can be stated as a differential or integral equilibrium equation with some boundary conditions.

### 2.1 basics of FEM

In a short description it can be said that FEM is a numerical procedure for solving boundary problems with an equation state (equilibrium of the system) written in a differential or in an integral form. In particular, in structural mechanics the differential equilibrium equation comes from equilibrium of forces in a differential piecewise, meanwhile the integral form comes from an energetic principle. However, mathematically, it is possible to shift from one expression to other because both represent physically the same problem. The most common expression of equilibrium in a structural system can be stated from the known virtual principle of work:

$$\int_V \sigma^t \delta\varepsilon dV = \int_V b^t \delta u dV + \int_s q^t \delta u dS \qquad (1)$$

Where $\sigma$ means stresses, **b** volume forces, **q** surface forces, $\delta\varepsilon$ virtual strains and $\delta$**u** virtual displacements.

The FEM meshes the domain in small pieces called finite elements. In the nodes of these elements it is forced equilibrium and compatibility equations. Nodes are shared by different elements so it is expected the contribution of all of them to the solution of the problem. Finally, in the interior of the elements the solution is interpolated from the values of the nodes using the polynomial interpolation called shape functions. In this case it is possible to express the displacements in a point of the domain as:

$$u(x,y,z) = \sum_{\text{nodes}} N(x,y,z)\, a \qquad (2)$$

Where **N** is the matrix of shape functions and **a** the displacements in the nodes. Depending on the type of elements this **N** matrix and **a** vector change of size and polynomial expressions.

Setting this expression (2) in the virtual work (1) it is possible to obtain a discrete equilibrium equation that drives directly to an equation system as follows:

$$\mathrm{Ka} = \mathrm{f} \tag{3}$$

Where:

$$\mathrm{K} = \sum_{\mathrm{elems}} \int_{\mathrm{Velem}} \mathrm{B}^{\mathrm{t}}\mathrm{DB}dV \tag{4}$$

and

$$\mathrm{f} = \sum_{\mathrm{elems}} \int_{\mathrm{Velem}} \mathrm{N}^{\mathrm{t}}\mathrm{b}dV + \sum_{\mathrm{elems}} \int_{\mathrm{Selem}} \mathrm{N}^{\mathrm{t}}\mathrm{q}dS \tag{5}$$

It can be seen that the mathematical needs of FEM are integration in the element, algebra of matrices and vectors, solution of equation system and interpolation.

### 2.2 The programming strategies

The FEM was born in 60's and it grows quite fast during 70's and 80's. In the pioneering years of FEM *the language* in science was FORTRAN and it was choose because there was no other possibility. However, it must be said that it was not an inconvenience, because FEM algorithm is a sequence of procedures that works perfectly with FORTRAN instruction. Moreover, the first algorithms mainly deal with linear analysis and elastic constitutive equations thus the complexity of the problems did not show the limitations of the language. Afterwards, the existence of mathematical powerful libraries, like ISML, available in FORTRAN helped the development of codes in such language.

When language C appears noting changes and the FEM codes still were developed in FORTRAN. The reason were mainly three: firstly the researchers would had rather manipulated the written FORTRAN codes rather had started new ones in C. Secondly, the advantage of dynamic memory allocation and abstract data were not seen enough positives to be able to change the programming paradigm. And finally, the strategy of C programming still were a sequence of procedures like FORTRAN. So, in

conclusion, C was not strong enough to break FORTRAN language in FEM programming.

## 3. The FEM programming drawbacks in 90's

The scenario briefly outlined previously has changed in deep during the 90's. The reasons why things change are related to the drawbacks and limitations in programming with FORTRAN and C.

First the complexity of the engineering problems increases very fast. Engineers want to solve for non-linear constitutive materials, coupled problems, dynamics, optimisation, etc. This means that it was not possible to write and maintain a code that solves everything. The dream of having '*the FEM program*' vanishes as people requested more and more. This problem of maintain and make the code to grow is especially dramatic when someone has to work in research. The research programmers always tend to forget that others will in future continue their job. The rule for them should be 'write programming code thinking in long terms because others will use it as a starting point'. But it is difficult, in most cases researchers learn while program and are foreign people that come back home after the degree. Their codes are seen as a tool for the research purposes not a result itself so these codes contain bugs, bad documentation and bad programming style.

Secondly, the existence of new languages, new syntax and new programming paradigms makes possible to shift the strategy of FEM programming from the main goal of organise procedures to the proper data organisation. What is more, the possibility of writing codes using the syntax of the natural language of the problem increases the programming skills for new constitutive equations.

Thirdly, it has been seen that FEM works well with abstract entities. Elements, matrices, nodes, etc. are mathematical concepts that exist, are real stuff. In FORTRAN the coordinates of the nodes always were stored in a vector and the stiffness matrix too. Notice that for two completely different entities in concept it was used the same programming implementation. This is a clear drawback because FORTRAN makes the natural language of the problem far from the implementation.

Finally, the graphical needs for meshing and post-processing of results are difficult to achieve with libraries written in FORTRAN. On the other hand as C as C++ have very good graphical libraries.

For all these reasons the tendency to write FEM programs shifts to the object-oriented syntax's because the codes seems to be easier to maintain. However, object-oriented is not the definitive solution to the FEM programming because a bad organisation of objects can drive to a data and procedure complexity which difficult the achievement of new programming challenges. The key for a successful FEM programming activity is to achieve a proper combination between the language skills and programming

abilities: write thinking in long terms, build programs according your needs and document codes as a starting point.

## 4. FEM from object-oriented point of view

The problem of programming FEM with object-oriented has been treated a lot in literature [1]-[7]. However, the natural approach to the problem can be seen from different points of view. In our opinion and based in previous experiences, [8] and [9], programming FEM following object-oriented syntax's can be a difficult task. In general, the most complicate is the engineering problem, the most is difficult to organise data structure and relationships between objects. For example, the trial of using inheritance can complicate too much the implementation instead of simplify it. Moreover, the use of non-linear constitutive equations can be considered from different points of view. On one hand non-linear behaves as linear until certain equilibrium point then, should a linear element (elasticity) inherit from a non-linear one? But from which one: plasticity, hyper-elasticity, damage models? Or it is better to consider the non-linear like a completely new one? In all cases the implementation is not easy and depends on personal views of the problem.

In procedural languages people used to program thinking that 'I will write a program that will do everything'. It is a common wrong approach to program in object-oriented with the same idea, 'I will write few objects that will do everything'. This philosophy drives directly to complicate codes; objects are too long and do too many things. Moreover, if ever the programming generality (high abstraction, templates, lists, etc.) can help to continue the development the program will run very slowly. On the contrary programming specific objects and procedures produces fast programs but difficult of enlarge. Finally, the object-oriented approach takes care of data abstraction and the programmer tends to forget that FEM is a procedure. Procedural languages 'forgets' about data and object-oriented 'forgets' procedures. Like many things in life the good one is the middle term, to reach equilibrium between the concept (objects organisation) and the procedure (problem solving).

Another problem appears when you try to solve large structures with FEM. It has been seen that the use of inheritance and templates tend to make the solution process very slow. Thus, if you program for a specific problem then the executable will run fast but the code will be difficult of sharing. On the other side, if you program generally the codes tend to be too slow. Now we must found again equilibrium between both things.

The latest problem is the difficult of growing object-oriented programs. One basic thing that can help is the facility that provides a good developing framework. For us, a developing framework is an amount of objects properly organised and documented. Thus the programmer can get the needed objects and after modify and compile them and reaches the wished program.

In the next pages it is described the solution to an object-oriented FEM code and a framework of development.

## 5. SCOPE the framework and the FEM object-oriented code

### 5.1 The general concept

The basic concept of SCOPE is its modular structure with clearly defined interfaces. This renders a step by step development and extension of the software on the one hand and the confirmed use of the program for different problems on the other hand. So it is possible to reach a high efficiency and flexibility also for programming as for the use of the program.

The basic structure is represented by an onion model that separates the framework in three layers.

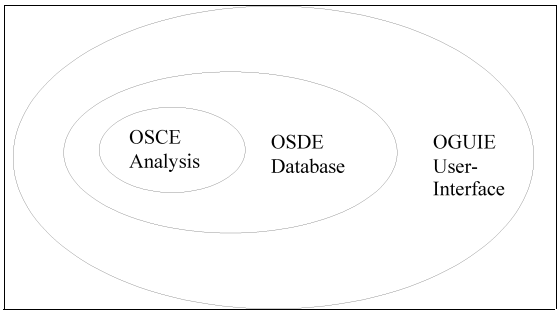


**Fig. 1.** The onion model for SCOPE

- The kernel OSCE (Objects for Structural Computations in Engineering) performs all necessary data structures and algorithms for the solution of finite element problems.
- The OSDE (Objects for Structural Databases in Engineering) is the second layer of the onion and consists of elements, materials and document libraries etc.
- The third layer called OGUIE (Objects for Graphic User Interface in Engineering) is an interface between OSCE, OSDE on one side and the environment of the user on the other side. For pre- and post-processing can be used some other existing programs as for example GID (http://gid.cimne.upc.es).

### 5.2 The kernel of SCOPE

The kernel of the onion model (OSCE), in the following called Solver, has to be independent of the nature of what has to be solved. So the steps of the solving process

has to be the same for static or dynamic, or even stability or thermodynamic, elastic or plastic, linear or non-linear or for continuos elements or discrete bars, etc.

The structure of SCOPE and the kernel can be found in the figure 2.

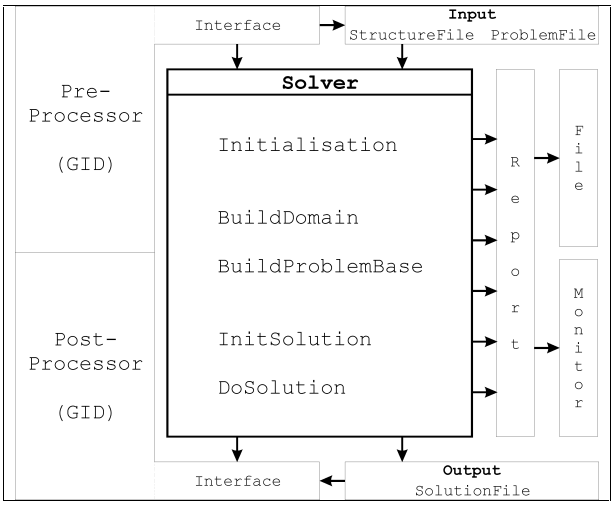


**Fig. 2.** Flowchart of SCOPE

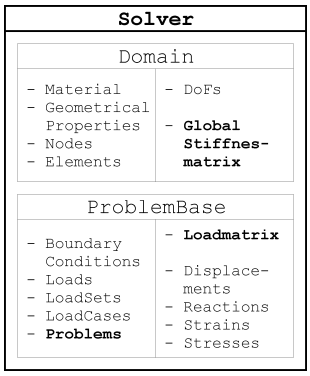


**Fig. 3.** Kernel objects

The solver consists of objects, which will be initialised during the solving process. The **domain** contains the domain objects built on the structure data. These data will build the material, the geometrical properties, the nodes and the elements. Later, depending on the problem, there will be built the degrees of freedom and the needed matrices for the solving process (e.g.: the global stiffness matrix).

The **problembase** contains the problem objects built on the problem data. These are the boundary conditions, the loads, loadsets and loadcases (which build the loadmatrix for each problem). Furthermore there is the possibility to solve various problems with the same domain in one solving process. The results are stored in Displacements, Reactions, etc.

During the development process or for bug-finding there is the possibility of getting a report file of the calculation process.

The organisation of the figure 3 is the basis for all future developments.

### 5.3 The project organisation

The realisation of all the claimed terms charges a strict organisation and separation of the needed objects. Therefore the SCOPE project will be separated in the following projects (figure 4).

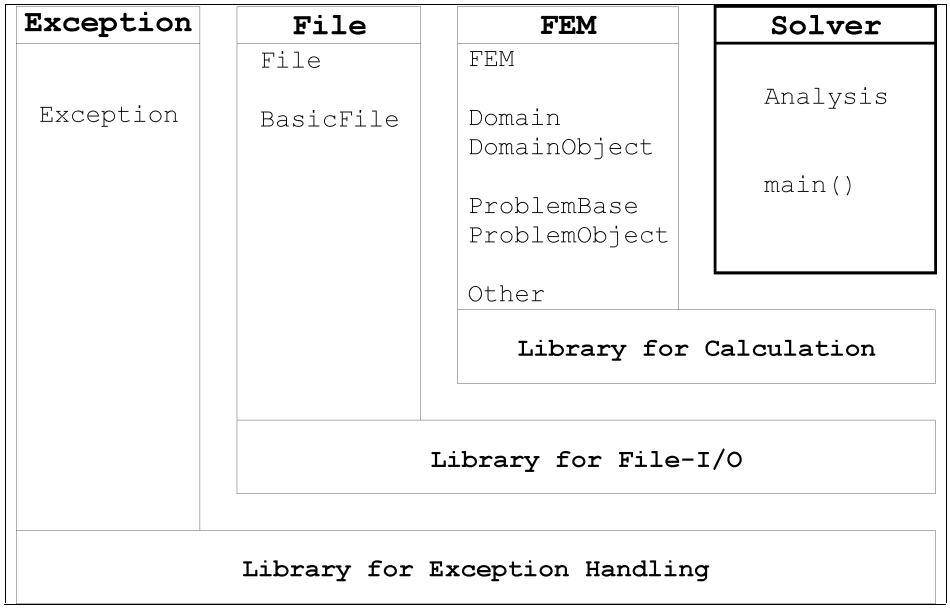


**Fig. 4.** Organisation of the SCOPE projects

The **Solver** project consists of the object Solver, which is an instance of the class Analysis, and of the main-function which will be compiled to an executable program. The Solver project depends on the FEM project. The flow of a FEM solution of a problem appears clearly in the next:

```
Analysis  Solver;

Solver.BuildDomain();       // Create Domain

Solver.BuildProblemBase();  // Create ProblemBase

Solver.InitSolution();

Solver.DoSolution();
```

The **FEM** project consists of all objects that are needed for the solving process. Its internal structure hides the problem-depending objects by the generally used objects. So the Solver is completely independent of the problem. The FEM project depends on the File project.

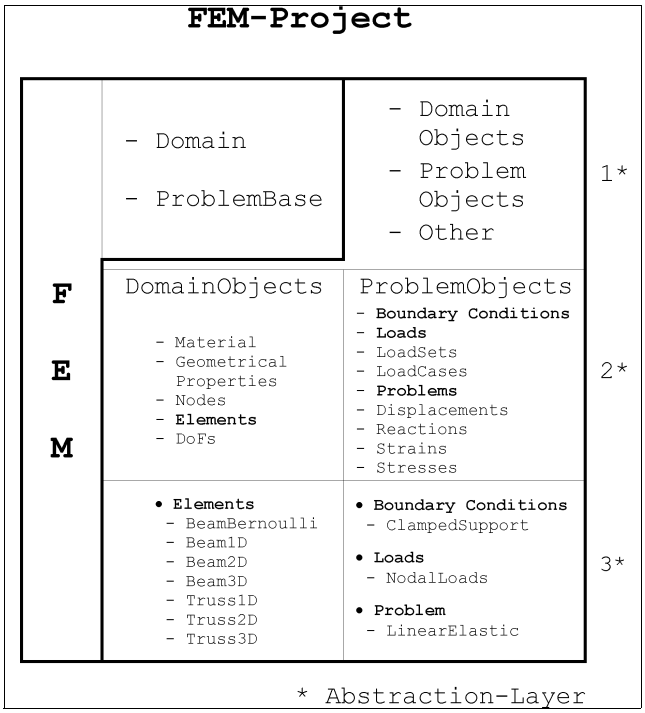


**Fig. 5.** FEM project and objects

The **File** project handles the input and output of data. It depends on the **Exception** project, which contains the classes for the Exception Handling.

All projects the Solver depends on can be manipulated or changed independent of each other.

### 5.4 The web-based developing framework

A framework must provide easy and general solutions. What you want is not a few lines of code that do everything, what you want are small pieces of code, easy to understand and easy to modify and maintain, join all together, compile and reach the executable file that solves such specific problem in different operative systems.

What you need is a place where all the objects are organised and documented, a place where you can navigate trough and download what you want. In this case what you want is what you get. The simplest strategy is to take advantage of the use of web-based technologies and build a framework as a web place. Mainly there are two strategies for building applications in such environment:

- Search objects to solve some type of problem.
- Search for a specific type of object.

In the first case we think in something like ‘I want to solve a linear analysis of a plane pin-jointed structure’. In the second case we talk about 'I need a plane element'. The web-based framework allows you to download by the type of problem or by the type of object.

## 6. Conclusions

The complexity of the engineering problems that FEM has to solve makes impossible the idea of developing a program for solving all cases. Object-oriented strategy has the positive approach because it makes data closest to the natural language of the problem. However, on the contrary, FEM is clearly a procedure. The best programming strategy for FEM is to write thinking in long terms and build objects for specific problems. There must be equilibrium between generality (slow executables) and specificity (difficult to grow).

A framework is a useful environment for developing FEM applications. The use of web-based technology can help the navigation and download among the needed objects and documentation. Moreover, users can download their request by the type of the problem or by the type of the object.

## References


1. Zimmermann, T., Dubois-Pèlerin, Y. y Bomme, P. “Object-Oriented finite element programming: I Governing principles”. *Computer Methods in Applied Mechanics and Engineering*. 98, 291-303, 1992.
2. Zimmermann, T. y Eyheramendy, D. “Object-Oriented finite element. I Principles of symbolic derivations and automatic programming”. *Computer Methods in Applied Mechanics and Engineering.* 132, 259-276, 1996.
3. Dubois-Pèlerin, Y. y Zimmermann, T. “Object-Oriented finite element programming: III An efficient implementation in C++”. *Computer Methods in Applied Mechanics and Engineering*. 108, 165-183, 1993.
4. Eyheramendy, D. y Zimmermann, T. “Object-Oriented finite element. II A symbolic environment fo automatic programming”. *Computer Methods in Applied Mechanics and Engineering*. 132, 277-304, 1996.
5. Forde, B. W. R., Foschi, R. O. y Stiemer, S. F. “Object-Oriented Finite Element Analysis”. *Computers and Structures*, 34, 355-374, 1990.
6. Kong, X. A. y Chen, D. P. “An Object-Oriented Design of FEM Programs”. *Computers and Structures*. 57, 157-166, 1995.
7. Miller, G. R. “An Object-Oriented Approach to Structural Analysis and Design” *Computers and Structures*, 40, 75-82, 1991.
8. Galindo, M. “*FemLab versión 1.0.”* Technical Report núm IT-114. Published by CIMNE. Barcelona, 1994.
9. K. Reimann, L. Gil, E. Blanco, J. López, E. Oñate y B.H. Kröplin. *ED-TRIDIM Lehrprogramm zur Berechung Dreidimensionaler Balken-und Strabtragwerke*. ISBN: 84-87867-02-X . Ed.Cimne, Barcelona, 1997.