An approach for FEM simulator development

Felix C.G. Santos*, Maria Lencastre

Rua Acadêmico Hélio Ramos, s/n - Recife, PE 50740-530, Brazil

Received 30 April 2003

Abstract

In this paper we present a computational environment called Plexus, whose main goal is to help the project and implementation of simulation software for coupled phenomena through flexible and friendly abstractions, based on the finite element method. This paper shows the relevance of that system, identifying a set of advantages, which can be granted with its use, exploring its understanding through examples.

© 2005 Published by Elsevier B.V.

MSC: 65M60; 65Y10; 68U20

Keywords: Coupled phenomena; Modeling process; Finite element method simulation; Abstractions; Simulation environment

1. Introduction

Simulators provide an economical means of understanding and evaluating the performance of both abstract and real-world systems [1,2]. Unfortunately, the design and implementation of simulators is almost as complex as the systems being simulated. To be efficient, therefore, simulators must be able to adapt to ever-increasing system complexity. However, except for a few numerical analysis computations, scientists and engineers write most of their technical software. It is not likely that someone from another field is able of writing a program from what they find in numerical analysis journals, because those papers are very mathematical [2,4,15]. It could be speculated that numerical analysis specialists would be dispersed among the sciences; so maybe those people are doing the programming. If so, then numerical analysis software is even more of a concern, because most software engineer techniques do not emphasize

* Corresponding author.
E-mail addresses: fcgs@demec.ufpe.br (F.C.G. Santos), mlpm@cin.ufpe.br (M. Lencastre).

0377-0427/$ - see front matter © 2005 Published by Elsevier B.V.
doi:10.1016/j.cam.2005.03.014
software methodologies specific to attend this kind of developers, requiring them to go deeper in the computer world, spending time developing proper abstractions, and having to deal with problems not of their specific area and interest [15].

The main contribution of this work is providing higher levels of abstraction to allow for a more flexible use of the finite element method (FEM) in multiphysics applications. This means reducing the development complexity and increasing reusability of software components, that is, dramatically decreasing the time spent by scientists and engineers in the design and production of simulators. The FEM is a way of obtaining a numerical approximation of a mathematical theory, which describes physical behaviour. This method is considered a powerful computational technique for the solution of differential and integral equations that arise in various fields of engineering and applied science. By its very definition FEM is deeply polymorphic, which means that FEM based-simulators with a high degree of versatility may be produced. Such generality does not exist in classical analytical methods [3,8,16]. As a conclusion, this work presents domain specific techniques for the FEM representation and implementation, decreasing the difficulty, time spent and costs in the development on FEM simulators. A simple example is described throughout the article showing the applicability in the development of a simulator for the dynamics of a rigid body attached to an elastic beam (with temperature dependent constitutive relation), where both are also submitted to thermal loads. After another simple example shows that most of the previous defined system can be reused for another purpose.

The paper is organized as follows. Section 2 introduces Plexus System. Section 3 shows Plexus Architecture. Section 4 describes an example that will be used throughout this article. Section 5 presents computational representation for Geometry. Section 6 describes the computational Phenomenon. Section 7 describes the Generic Interface Graph, which deals with the control of processes. Section 8 details the Simulator proposed structure. Section 9 shows the second example. Section 10 discusses some related work. Finally, Section 11 draws some conclusions.

2. Plexus system

Plexus is a simulation environment whose aim is to support the simulation of coupled phenomena through the FEM, guaranteeing high level of abstraction, system reusability, adaptability and maintainability. Plexus implements flexible tools for: decomposing the geometry into the relevant components, where the phenomena are defined, copying and selectively distributing geometry data between phenomena; defining different solution strategies for different phenomena groups; defining algorithms as data in several levels of the simulation; integrate pre-built software components to the environment and so on.

Figs. 1 and 2, show an overview of the system, which is divided into 4 components, representing the main processes: administration/system loading, which supports the system management and the loading of general system data and basic type tables; pre-processing, where the user inputs the problem data, and where dynamic structures for simulation are built; simulation processing, where data are processed to obtain the solution and where the verification occurs; post-processing, where the solution is processed in order to obtain the quantities of interest for the user and for the needed visualization. This component also deals with system validation.

The system manages great volumes of data, previously built components, phenomena, phenomena coupling, algorithms, performance analysis, definition of persistent data and simulation knowledge reuse. To give support to the high level of abstraction, flexibility, reusability, and data security, available in the
Plexus environment, there is a database management system (DBMS), which maintains the general abstract data related to the context, the algorithms that take part in different simulation strategies, the simulation problem’s data and also the simulation’s intermediary data and results.

3. Plexus architecture

The Plexus system architecture can be described through its packages specifications [12]. Fig. 3 shows an overview of the main ones, which are divided into:

(i) Controller packages composed of: domain registration, preprocessor, simulation, and pos-processor.

(ii) Managers build the ControlStructure package. Each manager has a very specific function. The Phenomena Domain Manager deals with all structures that are responsible for the phenomena data manipulation; the GeomDomain Manager deals with all geometric data from the supplied geometry and meshes up to the slave geometry; the Graph Manager is a generic package that supports the system with structures and tools for dealing with graph structures to be used in the representation of phenomena.
and geometry. There is also the Object Manager, which helps building complex objects that are going to be used during the process. Finally, there is the Algorithm Manager that is responsible for building the control strategy of the algorithms and methods to be executed for the specified simulation.

(iii) Repository Manager, which function is to support the whole system with higher level of abstraction to access data methods for the system repository.

4. Example 1

Consider the geometry defined in Fig. 4. It is composed of two sub-domains \( \Omega_1 \) and \( \Omega_2 \). The physical phenomena defined therein are (transient state): linear elasticity with temperature dependent constitutive equations in \( \Omega_1 \); rigid body motion of \( \Omega_2 \) (this body has a certain distributed mass density \( p_M \); heat transfer in \( \Omega_1 \) and \( \Omega_2 \)). Let a point \( p_M \in \Omega_2 \) be a reference point for the rigid body. It is very convenient if such a point could be the centre of mass of \( \Omega_2 \), because the weight and inertia terms would not generate moments around \( p_M \).
Fig. 4. Example 1.

The exact mathematical models are as follows:

(i) **Phenomena in** $\Omega_1$: $w_1$ is the displacement of the points in $\Omega_1$ and $T_1$ is the temperature in $\Omega_1$.

(i.1) **Elasticity:**

\[
\frac{\partial^2 w_1}{\partial t^2} - \nabla \cdot (\sigma(w_1, T_1)) = 0,
\]

\[w_1 = 0 \quad \text{on} \quad \Gamma_2,\]

\[w_1 = \beta \cdot w_2 \quad \text{on} \quad \Gamma_7,
\]

where

\[
\beta = \begin{bmatrix} 1 & 0 & -\delta_2 \\ 0 & 1 & \delta_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & \mathbf{e}_1 \cdot (\mathbf{k} \times \delta) \\ 0 & 1 & \mathbf{e}_2 \cdot (\mathbf{k} \times \delta) \end{bmatrix},
\]

with

\[
w_2 = \begin{bmatrix} \mu_{M_1} \\ \mu_{M_2} \\ \phi_M \end{bmatrix}
\]

and $w_2$ being the vector with the displacement and rotation of $\Omega_2$ with respect to the reference point $p_M$.

\[
\delta = \begin{bmatrix} \delta x \\ \delta y \end{bmatrix} = \mathbf{x} - p_M \quad \text{for} \quad \mathbf{x} \in \Gamma_7,
\]

\[
\sigma_1(w_1, T_1) \cdot \mathbf{n}_j = 0, \quad \text{on} \quad \Gamma_j, \quad j = 1, \ldots, 3,
\]

where $\mathbf{n}_j$ is the normal to $\Gamma_j$ pointing outwards $\Omega_1$.

(i.2) **Heat transfer:**

\[
\rho_1 c_1 \frac{\partial T_1}{\partial t} - \nabla \cdot (K_1 \cdot \nabla T_1) = 0 \quad \text{on} \quad \Omega_1,
\]

\[
\mathbf{n}_j^T \cdot (K_1 \cdot \nabla T_1) = h_j (T_j^\infty - T_1) \quad \text{on} \quad \Gamma_j, \quad j = 1, \ldots, 3,
\]
where $h_j$ and $T^\infty_j$ are the coefficient of heat transfer by convection and the environment reference temperature on the boundary $\Gamma_j$, $j = 1, \ldots, 3$. Also,

$$n_7^T \cdot (K_1 \cdot \nabla T_1) = q = n_7^T \cdot (K_2 \cdot \nabla T_2),$$

where $n_7$ is the normal to $\Gamma_7$ pointing outwards $\Omega_1$.

(ii) **Phenomena in $\Omega_2$:** $s_2$ is the displacement of the reference point $p_m$ and $T_2$ is the temperature on $\Omega_2$.

(ii.1) **Rigid body motion:**

$$I_2 \cdot \frac{\partial^2 \mathbf{w}_2}{\partial t^2} = \mathbf{F}_2 + \int_{\Omega_2} [e_3 \cdot (\delta \times \mathbf{g}_2)] \, d\Omega - F^{bd}_2,$$

where $I_2$ is the matrix of moments of inertia

$$I_2 = \int_{\Omega_2} \rho_2 \beta^T \cdot \beta \, d\Omega$$

$\mathbf{g}_2$ is the body load defined on $\Omega_2$.

$$\mathbf{F}_2 = \begin{bmatrix} F_{M_1} \\ F_{M_2} \\ M_M \end{bmatrix}$$

is the force vector acting directly on the reference point $p_M$ and, finally,

$$F^{bd}_2 = \int_{\Gamma_7} \beta^T \cdot (\sigma_1 \cdot n_7) \, d\Gamma$$

is the force that domain $\Omega_1$ exerts over $\Omega_2$ through $\Gamma_7$.

(ii.2) **Heat transfer:**

$$\rho_2 c_2 \frac{\partial T_2}{\partial t} - \nabla \cdot (K_2 \cdot \nabla T_2) = R_2,$$

$$n_7^T \cdot (K_2 \cdot \nabla T_2) = h_j(T^\infty_j - T_2) \quad \text{on } \Gamma_j, \quad j = 4, 5, 6,$$

$$T_2 = T_1 \quad \text{on } \Gamma_7.$$

**Simulation regions:** are the regions where the phenomena are defined. The regions, where restrictions involving one or more phenomena are defined, are also included as simulation regions. In the above example we have:

- $\Omega_1$, where elasticity and heat transfer are defined,
- $\Omega_2$, where rigid body motion and heat transfer are defined,
- $\Gamma_7$, where one restriction involving $T_1$ and $T_2$ and another involving $w_1$ and $w_2$ are defined,
- $\Gamma_2$, where a restriction involving $w_2$ is defined.
5. Geometry

Plexus uses a computational representation for the geometry based on the boundary representation method (Brep). This is extremely useful due to a number of reasons. For instance, when a phenomenon is defined on a region, its boundary conditions will be defined on all parts of its boundary. Thus, the relation between a phenomenon and its boundary conditions resembles that of the boundary representation (for example in a direct acyclic graph implementation).

5.1. Geometry diagram

The geometry diagram is considered as direct acyclic graph pattern (DAG). Fig. 5 shows such a pattern.

5.2. Geometry example

Geometry components: as it can be seen, from the example in Fig. 4, there are 6 points, 7 curves and 2 plane regions defined for the problem geometry. The boundary of the plane region $\Omega_1$ is composed of curves $\Gamma_1$, $\Gamma_2$, $\Gamma_3$ and $\Gamma_7$, which in turn have boundaries defined by the points 1, 2, 3, 4. In a similar way the plane region $\Omega_2$ is also defined.

6. Phenomenon

This section suggests a physical phenomenon abstraction, called here a computational phenomenon [11]. The main objective with such an abstraction is to make it intuitive and easier the representation of data sharing and dependence between different phenomena.
6.1. Phenomenon diagram

A computational phenomenon, Fig. 6, is defined by its vector field and weak forms, which in turn are defined in their respective geometric entities together with boundary conditions information, which is also implemented as fictitious phenomena defined on the respective geometric entity of the boundary of the simulation region. It has also MathMethods that implement: Mesh generation, Integration Rules, Shape Functions, etc. A computational phenomenon is considered as a set of processes, which produces matrices and vectors related to a specific weak form (as a whole or only parts of it) defined on a specific geometric entity. Each one of those matrices and vectors may be dependent on vector fields from other phenomena. The reference adopted in the definition of a computational phenomenon is based on the simulation region in which it is defined. Phenomenon is an abstraction and may also be used to represent restrictions (involving one or more phenomena), boundary conditions and other types of relationships and processes.

An original phenomenon may generate several computational phenomena by the time the modelling is finished. The generated phenomena are related to boundary conditions, which are defined for the original one and implemented as fictitious phenomena. Fictitious phenomena defined in order to implement boundary conditions inherit many pieces of information from the original phenomenon, such as vector field, geometric and phenomenon mesh, etc. We assume that the phenomenon class is organized in a suitable form, so there is a preprocessor, that makes the necessary mapping from the input modeling information related to the simulation strategy onto its data [12].

6.2. Phenomenon example

6.2.1. Problem specification

(a.0) Define phenomena: in the defined example problem, see Fig. 4, we can identify the following simulation phenomena:

- Heat transfer, phenomenon represented by vector field $T_1(\Omega_1)$.
- Heat transfer phenomenon represented by vector field $T_2(\Omega_2)$.
- Phenomenon represented by vector field $\mu_q$ (Lagrange multiplier in $\Gamma_7$, due to restrictions between $T_1$ and $T_2$).
- Phenomena represented by vector fields $w_1$ and $w_2$.
- Phenomena represented by vector fields $\mu$ and $\mu_f$ (Lagrange multipliers in $\Gamma_2$ and $\Gamma_7$, respectively due to restrictions in $w_1$, and between $w_1$ and $w_2$).

(a.1) Phenomenon specification: each one of the described phenomena has its own discrete vector field, geometric domain, coupling with other phenomena, discrete weak form, and other relevant data. Furthermore all of the defined phenomena will be provided the following:

- Integration rule: a tool for providing integration points and respective weights with respect to the reference finite element.
- Shape functions (test function and trial function): a tool for providing the values of the shape functions and their derivatives in any point of the reference finite element.
- Phenomenon sub-algorithms
Fig. 6. Phenomenon classes.

- Initial state.
- Compute initial time step.
- Compute the contributions to matrices and vectors used in the computation of the initial auxiliary data.
- Initialize group solution iteration state.
- Compute contributions to matrices and vectors depending on a set of parameters.
- Compute next time-step.
- Restrictions (Dirichlet or other type)
  - Lagrange multipliers due to restrictions involving \( w_1 \) and \( w_2 \) on \( \Gamma_7 \) (\( w_1 - \beta \cdot w_2 \)).
  - Restriction involving \( w_1 \) on \( \Gamma_2 \) (\( w_1 = 0 \)).
  - Restriction involving \( T_1 \) and \( T_2 \) on \( \Gamma_7 \) (\( T_1 = T_2 \)).
- Phenomena couplings
  - Between \( w_1 \) and \( T_1 \) on \( \Omega_1 \).
  - Between the \( w_1 \) and the Lagrange multiplier \( \mu_\ell \) on \( \Gamma_7 \).
  - Between \( w_2 \) and the Lagrange multiplier \( \mu_\ell \) on \( \Gamma_7 \).
  - Between \( w_1 \) and the Lagrange multiplier \( \mu \) on \( \Gamma_2 \).
  - Between \( T_1 \) and the Lagrange multiplier \( \mu_q \) on \( \Gamma_7 \).
  - Between \( T_2 \) and the Lagrange multiplier \( \mu_q \) on \( \Gamma_7 \).

(a.2) Phenomena boundary conditions:
- Boundary conditions for \( w_1 \):
  - Zero Neumann boundary conditions on \( \Gamma_1 \) and \( \Gamma_3 \). Two phenomena should be considered for loads (Neumann with zero load). They conceptually exist but do nothing.
• Dirichlet restriction on $\Gamma_7$.
• **Boundary conditions for $w_2$:**

  • Zero Neumann boundary conditions on $\Gamma_i$, $i = 4, 5, 6$. Thus three fictitious phenomena will be defined. They will perform no action.
  • Dirichlet restriction on $\Gamma_7$.

• **Boundary conditions for $T_1$:**

  • Zero Neumann boundary condition on $\Gamma_2$.
  • Mixed boundary conditions on $\Gamma_1$ and $\Gamma_3$. Therefore two fictitious phenomena will be defined there.
  • Dirichlet restriction on $\Gamma_7$.

• **Boundary conditions for $T_2$:**

  • Mixed boundary conditions on $\Gamma_i$, $i = 4, 5, 6$. Three fictitious phenomena will be defined there.
  • Dirichlet restriction on $\Gamma_7$.

7. **Generic interface graph—GIG**

    GIG is a generic interface graph, which deals with definition and control of processes, taking into account some specific requirements of simplicity, easiness of definition from algorithmic natural language and flexibility in the granularity of defined processes. The pattern is intended to help the design and reuse of software components [13].

7.1. **GIG diagram**

The GIG structure is presented in the UML diagram in (Fig. 7).

The GIG framework is composed of the following participants:

• **DataAlgorithmManager:** Its function is to control the whole workflow, managing its lifecycle. This involves building and re-programming the application workflow at runtime. It also relates each workflow with a data server. It may be extended in order to represent more specific features of the problem domain.
• **DomainData:** Represents the whole set of objects, which contains the problem domain data.
• **GraphNode:** It is an abstract class that implements low-level operations related to the interoperability between workflow nodes.
• **SkeletonGraph:** It has the root of a workflow graph (skeleton). It manages the building and the modification of its skeleton. It is responsible to execute the workflow when requested to.
• **Algorithm:** Represents the procedure (algorithm) of each workflow node. It is used as a base class for all algorithm classes of the application.
• **AlgorithmData:** Represents a data type to be used by an AlgorithmNode during workflow execution. It is used as a base class for all algorithm data classes of the application. It offers identification services.
• **DataAlgthmServer**: It provides a service that relates AlgthmNode with AlgthmData to be used in the building and modification of a workflow graph. It may be extended in order to represent more specific features of the problem domain.

• **AlgthmConnection**: Represents an algorithm component that was not connected yet, that is, it references an algorithm that was not incorporated yet to the workflow. So when it is executed it fetches the algorithm and replaces itself with the fetched algorithm.

We can summarize the building of a workflow: it starts with an application sending a request to a DataAlgthmManager instance for the building of a workflow starting with an identification of the desired driver component. DataAlgthmManager forwards this request to a DataAlgthmServer instance, associated with that workflow, which will be its provider for all data and algorithm components. Then, the DataAlgthmServer instance creates an object of the class SkeletonGraph and the AlgthmNode correspondent to the driver component. Next, DataAlgthmServer asks the driver to recursively build the entire graph: the driver asks DataAlgthmServer for its AlgthmData and its children AlgthmNode objects; after that it asks each one of its children to build its sub-graph recursively. This process goes on until all nodes of the workflow are created and assembled.

### 7.2. GIG example

In our case study we consider a FEM simulator specification capable of solving problems involving transient phenomena. Usually, it is observed that an algorithm defined for the solution of a problem by the FEM method has repeated (similar) structures. Thus in the pursuing of a high degree of reusability, four hierarchical levels of processes were used, each one with several possibilities of algorithms (Section 8 presents more details). The levels are Global Skeleton, Block Skeleton, Group Skeleton, and Phenomena procedures [11]. Only two blocks are needed in the present case. In the \(i\)th-Block Skeleton \(Ng(i)\) is its
Fig. 8. Global and Block Algorithm skeleton graphs.

number of groups. Fig. 8 present GIG direct acyclic graph to implement Global and Block Algorithm skeletons.

Global Algorithm Skeleton:
I. From Blocks $i=1$ until 2
I.0) Retrieve initial state for Block $i$
I.I) Compute initial time step $\Delta t_i$ for Block $i$
I.II) Compute initial auxiliary data for Block $i$
II. Compute initial $t_0 = \min_{1 \leq i \leq 2} \{\Delta t_i\}$, set time instant $t_1 = 0$
III. While $t_1 \leq T_{\text{max}}$ do:
III.0) Set $t_0 = t_1$ and $t_1 = t_0 + \Delta t$
III.I) For Block $i = 1$ until $Nb$
III.I.0) Solve for Block $i$
III.I.I) Compute next time step $\Delta t_i$ for Block $i$
III.I.II) Compute next time step $\Delta t = \min_{1 \leq i \leq 2} \{\Delta t_i\}$
III.I.III) Continue with time iteration
IV. End of the simulation

Examples of Block Algorithm skeletons:
Is-Bi) Retrieve Initial State for Block $i$ (see(I.0)):
Is-Bi.0) For $r=1$ until $N_i$
Is-Bi.0.0) Group $r$, compute phenomena initial state
It-Bi) Compute initial time step $\Delta t_i$ for Block $i$ (see(I.I)):
It-Bi.0) For $r=1$ until $N_i$
It-Bi.0.0) Group $r$, compute Initial time step $\Delta t_i$
It-Bi.1) Set $\Delta t_i = \min_{1 \leq r \leq N_i} \{\Delta t_i\}$

In GIG we start from an algorithm in natural language. The procedure is first divided into different algorithm components (algorithm nodes) and then it is organized in the form of a graph [10]. In our example we use the conversion made in [14]. The Global Algorithm Skeleton Graph is basically built using a SkeletonGraph object, which has a reference to the driver, and AlgthmNode objects as the algorithm components. Each AlgthmNode object has a set of references to AlgthmData objects. In our example, there should be AlgthmData objects, which includes pieces of information needed in the identification of
the Block Skeletons that will be plugged in the Algorithm Skeleton Graph as AlgthmNode objects (from another graph, integrating in this way the graphs presented in Fig. 8), substituting the AlgthmConnection objects.

7.3. Considerations

Simulations using FEM can become very complex, particularly, when the designer wants to guarantee high level of abstraction and reuse of the developed solutions. In the studied case, one clearly identifies levels of its architecture where the workflows are to be dynamically defined, built and controlled. Those requirements comprise the main strategies in saving the production costs of high quality simulation software. In this paper we identify that GIG solution satisfies the application requirements: easiness of translating from the natural language representation of the processes into a computer (executable) representation; simplicity of use; versatility and flexibility in the implementation of solution processes; reduction of the possibility of errors in the coupling of processes; need for support of adaptability at run time, due to dynamic change of business rules, and need of dynamic data creation.

8. Simulator

The skeleton Simulator is a basic structure of a simulator for coupled phenomena based on what is here called algorithm skeletons. By simulators we call a computational system aimed at obtaining approximate solutions to systems of coupled partial differential equations, together with a set of restrictions (possibly integral-differential-algebraic relationships involving one or more vector fields). The main advantage in its application is its high level of abstraction, reusability and modularity, whose importance, when implementing a complex simulation, can never be overestimated.

This subsection describes a general set of classes, and their interactions in the guidance of the development of simulator models based on the Finite Element Method. The problem is first divided into different pre-defined levels of abstraction (global solution, blocks of groups, groups of phenomena, phenomena), where the simulator designer must supply algorithm skeletons for each one of those levels. The main advantage of the pattern is its high level of abstraction leading to reusability and modularity in the design of simulators for coupled multi-physics phenomena.

The proposed approach supports the abstraction for the coupling of different phenomena simulation in a single strategy, identifying which parts can be reused, proposing a modular solution. The idea is to make it easy the representation of the shared data between different phenomena and also their data dependence (Fig. 9).
The solution suggests a simulator structure organization considering a global algorithm class (global skeleton) for the global problem, based on blocks, groups of phenomena and phenomena objects. The Global Algorithm Skeleton articulates the procedures involving all blocks/groups. In the definition of the Global Algorithm matrices and vectors were used, and then each phenomena will contribute in the building of them. So, the skeleton algorithm is responsible for the control of the main process flow. The implementation of a skeleton depends on a series of decisions about the kind of problem that the strategy will attend. For instance, the type of phenomena (transient, pseudo-transient, steady-state), estimation error consideration, kind of adaptation if it exists, etc.

The solution considers a level of computation in which blocks of groups of phenomena are articulated. This produces more clean and reusable global skeletons. Each block has its own set of skeletons (Block Skeletons) and each group has its own and more simple (but less reusable) set of Group skeletons. Observe that those Blocks skeletons articulate the groups and the Group skeletons in turn articulate the phenomena in their computations. What is left to be defined are the contributions of each phenomena in each one of the phenomena skeletons. Many of the phenomena (mainly the fictitious ones) will simply do nothing when asked to perform some actions, because they do not contribute to them. So, each group will have its own solution skeletons (Group Solution Skeletons) and methods. The phenomena classes will be composed of phenomena data and a group of numerical methods, which are changeable (can be modified by the users through input data).

8.1. Decomposition of the solution algorithm

The 4 levels of computation demands (skeletons and methods) are detailed in what follows [11].

(a) **Global Skeleton** is the first level of computation and represents the global algorithm skeleton (the core of the simulator). The Global Algorithm Skeleton articulates the procedures involving all blocks. The procedures here deals with a higher level of the simulation execution, like time loops, adaptive iterations, and so on. It also includes general requirements such as asking the blocks to obtain the block solution or to perform an adaptation procedure. There is no need for matrices and vectors manipulations in this level. The building of a Global Skeleton depends on a series of decisions about the whole classification of the simulation. A Global Algorithm Skeleton is unique for each simulator, but may be replaceable, producing another simulator. Global Algorithm Skeleton is the procedural structure representing the algorithm to be performed with demands defined still in a higher level. It does not make any requirements directly neither to a Group of phenomena nor to any phenomenon.

(b) **Block Skeletons** are made in order to articulate the Groups of Phenomena in the execution of tasks demanded by the Global Skeleton. Each block has a set of skeletons (Block Skeletons), which satisfies the demands from the Global Skeleton by decoding them into demands for the groups in a previously defined order. A simulator may have a Block Skeleton changed without needing to change its Global Skeleton. Nevertheless, a well-designed Block Algorithm Skeleton is also very reusable and it is not supposed to be substituted even in the case of very severe changes in the solution algorithm in the level of the Group of phenomena. Block Skeleton defines solution procedures such as iterations in the case of operator splitting solution strategies (which involves all Groups), iterations in the case non-linear solvers (involving one or more Groups) and so on. It also transfers directly to its Groups some of the demands coming from the Global Skeleton (time step estimation, error estimation, etc.) and possibly post-processing the output from the Groups.
(c) **Group Skeletons** are made in order to articulate the Phenomena in the execution of tasks demanded by the Block Skeletons. A Group is provided with a set of Group Skeletons, which represent very specific procedures and may not be very reusable. Its purpose is to segment (encapsulate) the parts from the solution scheme, which are specific of the particular solution method being used for a group of phenomena. Usually, the more reusable parts of the solution scheme are best located either in a Block Skeleton or in the Global Skeleton. In the Group Skeletons the quantities produced by the Phenomena Skeletons are manipulated in the way required by the solution method, which characterizes the Group. Thus the Group becomes specialized in the solution of any subset of a set of possible phenomena and so, all vectors and matrices used in the solution are located in the Groups. The Group also needs to have knowledge of the couplings of its Phenomena whenever building coupled terms. This is so because the coupled terms are built using a possibly already computed discrete vector field (possibly related to other group), which should be appropriately defined. Frequently, Group Skeletons make use of MathMethods, whenever there is a task, which can be encapsulated representing either a reusable or a replaceable procedure (solution of an algebraic system of equations, for instance).

(d) **Phenomenon Procedures** represent the lowest level of all procedures in the simulation and are specific of all possible contributions its Phenomenon can provide to any solution scheme. Starting from the computation of the Global Skeleton and going through the two other levels of articulation, what remains to be defined are the contributions of each phenomenon to its Group solution scheme in a uniform parameterized way. The phenomena classes will be composed of phenomenon data and a group of numerical methods (MathMethods), which are replaceable (can be modified by the users through input data, like integration rules, for instance).

### 8.2. Simulator diagram

The main structure of the solution for representing a general FEM-Simulator is composed of Simulator, Block of Groups, Group of Phenomena, Phenomenon and Algorithm Skeletons, see Fig. 10. The FEM-Simulator Skeleton pattern suggests a FEM-Simulator algorithms organization with 4 levels of
computational demands: Global Skeleton, Block Skeletons, Group Skeleton and Phenomenon Skeletons. These levels were defined due to the high number of repeated (similar) structures and the degree of reusability of the involved algorithms (see example in Section 7). Each Block has its own set of Block Skeletons and each Group has its own and more specific (thus problem dependent, less reusable) set of Group skeletons.

The FEM-Simulator is composed of the following participants [11]:

- **Simulator** represents a class of possible simulations and it is responsible for the control of the main process flow; thus it maintains the core of simulation through the Global Skeleton.
- **Algorithm skeleton** is an algorithm described by the simulator designer, corresponding to one of the levels of computation (Global, Block, Group), using the pattern-defined abstractions.
- **MathMethod** is a tool with a very specific purpose and is used by either algorithm skeletons or encapsulated procedures inside a phenomenon. For instance, MathMethods are defined for numerical integration, mesh adaptation, error estimation and other tasks.
- **Global Skeleton** is the highest level of the solution scheme and it articulates the action of all Blocks. It is supposed to be strongly reusable.
- **Block** is a set of Groups of phenomena. Each Block has a set of skeletons called Block Skeletons. More than one block is justified, for instance, in the case where a problem can be partitioned into either independent or one-side dependent sets of groups of phenomena.
- **Block Skeletons**, where the Groups are required to perform a certain number of categories of procedures (for instance, partitioned (staggered) solution procedures involving groups of phenomena). When a Group is asked to execute a category of procedures (for instance, to compute a solution for its group of phenomena), it executes a very specific algorithm, which is a member of that category. Block Skeletons are supposed to be strongly reusable.
- **A Group** is a set of phenomena, which are going to be solved monolithically. A Group is provided with a set of Group Skeletons.
- **Group Skeletons** represent very specific procedures. Due to its problem- and method-specific definition and organization, the Group Skeletons are the less reusable among all skeletons. Nevertheless, it may be implemented in such a way that it becomes able of considering a varying number of phenomena, depending on the requirements from the simulation design.
- **A Phenomenon** represents a complex system composed of data and tools. Its primary responsibility is to provide the contributions of each phenomenon to a Group System to be solved in each instant of the solution process. This level is the place where the couplings and other processes of data sharing and dependence are considered in the formation of the needed vectors and matrices. It is the lowest level of the procedures in the solution schemes and thus it represents a tremendous effort in terms of programming, testing and validation. Therefore, the reusability of the tools located in the classes, which compose what we call a Phenomenon is fundamental in the saving of time and cost whenever one is programming new simulations.

We can summarize the structure participants major interaction in the following way: the global skeleton articulates the procedures involving all blocks. It does not make any requirements directly neither to a Group of phenomena nor to any Phenomenon. The block skeletons then define the activities of the groups. The Group skeletons in turn articulate the phenomena in their computations. This produces more clean and reusable Global and Blocks algorithm skeletons leaving to the Groups Algorithm Skeletons the
responsibility of defining the specific problem dependent (non-reusable) procedures of the whole solution algorithm.

8.3. Simulator example

**Global scenery:** Defines the general specification for the simulator, which includes:

- Skeleton Specification, which describes Transient Phenomena, with no Estimation Error and no Adaptation.
- Phenomena Context, composed of: elasticity, rigid body motion and heat transfer.

**Group scenery:**

- With no Front tracking.
- With Assembly.
- The inner procedure for each solver group is linear and iterative with pre-conditions and no Domain Decomposition.

**Group specification:**

- Group 1: phenomena represented by vector fields $T_1$ and $T_2$ (heat transfer in $\Omega_1$ and $\Omega_2$).
- Group 2: phenomena represented by vector field $\mu_q$ (Lagrange multiplier in $\Gamma_7$, due to restriction involving $T_1$ and $T_2$).
- Group 3: phenomena represented by their vector fields $w_1$ and $w_2$ (elasticity in $\Omega_1$ and rigid body motion in $\Omega_2$).
- Group 4: composed of the phenomena represented by their vector fields $\mu$ and $\mu_f$ (Lagrange multipliers in $\Gamma_2$ and $\Gamma_7$, respectively, due to restrictions in $w_1$).

**Define blocks**

The blocks are a level of computation in which block of groups of phenomena are articulated. This produces global skeletons that are more clean and reusable. Each block has its own set of skeletons and each group has its own more simple set of group skeletons. For the example we have: Block 1, composed of Group 1 and Group 2; and Block 2, composed of Groups 3 and 4.

8.4. Considerations

It is worthwhile observing that a simulator does not depend on the actual implementation of the Blocks, Groups and Phenomena. Those sets of data are provided after a simulator pattern structure is built, because they are related to the specific simulation being carried out. Thus a simulator should be able of solving completely different simulations, defined on completely different geometries and considering completely different sets of phenomena. Nevertheless, there is no universal simulator, meaning that a simulator is able of simulating only a class of simulations, which should be defined in a somehow clear way. For instance, considering only its Global Skeleton, the simulator built in the example (Section 7.3) is able of solving simulations in the class of dynamic problems with neither adaptation nor error estimation. Considering the Block Skeletons we built it is able of solving only linear (or very mild non-linear) problems. The
Group Skeletons are very specific to the solution scheme used and even slight modifications may cause a need to change them. As it was said just before this paragraph, the **couplings** and other process of **data sharing** and **dependence** are considered in the phenomenon level leaving the Global Skeleton and the Block Skeletons free of having to consider them. Since the Group Skeletons are less reusable, it may (and frequently is) deal with the requesting from the phenomenon and assembling in the right way the coupled quantities.

9. Example 2

In this section we show that the simulator built for the example previously defined is actually a simulator for a class of problems. In order to do that we describe another problem composed of three rigid bodies inserted into an elastic bounded region, with no heat transfer.

9.1. Example description

The geometry of the problem is a square plane elastic region with three rigid bodies inserted in it as shown in **Fig. 11**. Two of the rigid bodies are only partially inserted in the elastic region, while the third one is in its interior. The rigid body $\Omega_2$ will receive a sudden discharge of energy on $\Gamma_5$, modelled by an initial nonzero velocity, while all the system has zero initial displacement. The other bodies have zero initial velocity, with the exception of the elastic interface with $\Omega_2$. The free surfaces $\Gamma_j$, $j = 4, 6, 7, 8$, has zero Neumann conditions. Each rigid body has its own point of reference and other geometric properties (see the definition of the first example).

The linear elasticity phenomena in $\Omega_1$ will be denoted by its vector field $w_1$, and the rigid body motion of body $\Omega_j$ will be denoted by its vector field $w_j$, $j = 2, 3, 4$.

9.2. Application of the proposal

All vector fields will be governed by the equation of motion (elastic and rigid bodies) as described in Section 4 (with the exception of the temperature dependence, for which will be given a reference temperature). $w_1$ will be subjected to Dirichlet restrictions on the surfaces $\Gamma_j$, $j = 1, 2, 3, 9, 10, 11, 12, 13, 14, 15$.

![Fig. 11. Example 2.](image-url)
which will generate 10 (ten) Lagrange multipliers phenomena denoted by the vector fields \( \mu_j, j = 1, 2, 3, 9, 10, 11, 12, 13, 14, 15 \), defined on the respective surfaces. There will be no phenomena in the Groups 1 and 2, because there is no heat transfer problems defined in this case (those groups are ruled out). There will be 4 (four) phenomena in Group 3 (one elasticity phenomenon and three rigid body phenomena). There will be 10 (ten) phenomena in Group 4 due to the Lagrange multipliers. The number of Blocks is reduced to only one (Block 2) and the whole solution algorithms remains the same in all levels of computation. The couplings between phenomena occur on the interfaces where Dirichlet restrictions are prescribed.

9.3. Considerations

Note that the exactly same simulator is used in this problem which is different from the other geometrically, but not in its essence: the phenomena set and solution scheme remain in the same expected scenery. Whenever solution strategies are changed, the algorithms either in the Block level or in the Group level would be changed, but the phenomena definition and the whole interoperability between phenomena end between all levels of computation would be kept unchanged.

10. Related work

Due to the great relevance of simulation in different application areas, the community of simulation researchers is very active and we can name several existing works available, which are related somehow to the Plexus Project. It is not the objective of this work to make a full list of them, but we would like to cite some of them, due to their importance and achievements:

- SCIRUN, a scientific programming environment that allows the interactive construction, debugging and steering of large-scale scientific computations [7]. SCIRun enables scientists to design and modify models and automatically change parameters and boundary conditions as well as the mesh discretization level needed for an accurate numerical solution.

- DIFFPACK [5], addresses an object-oriented strategy for the development of software solving system of partial differential equations (PDEs). The proposed strategy encourages reuse of modules capable of solving the involved sub-problems. It extends the basic ideas of an object oriented numerical library to a higher level where the objects reflect partial differential equations. It also opens the possibilities of building repositories of solvers for single PDEs that can be combined with each other in a flexible way. In [6] a preprocessor is used to generate geometric input data required by finite element methods. It also defines an abstraction for simulating coupled problems by operator splitting techniques.

- [9] presents an algorithm framework for flexible Finite Element-based Modeling. Its goal is to explore the development of applications accommodating the addition of new and adaptation of existing modeling capabilities by developers, in such a way as to be adequately accessed and applied by the users. The algorithm flexibility is provided by offering families of algorithms that can be easily accessed and changed dynamically. Distinct algorithms can be used in different parts of the model.

These works support good practices and fundamentals useful for defining worthy simulation systems solutions and capabilities. However, despite of the richness of their contribution, the authors of this paper
did not find an integral environment that attends the demanded features for coupled phenomena environments described in earlier sections. There are still important questions related to abstractions of numerical algorithms in order to allow for easy interchange between numerical methods and strategies; repository organization and management and its relationship with simulation instances; satisfactory abstractions of couplings, which could be defined independently of the actual implementation of the participating phenomena; abstractions of groups of phenomena, which are to be solved together, making it possible to organize operator splitting strategies in different forms; higher abstractions of the phenomena-geometry relationship, in order for each phenomenon to decide if it wants to share mesh or not with other phenomena defined in the same geometry; and so on.

11. Conclusion

We have shown different aspects which contribute for automatically building a simulator based on FEM. This work proposed high level of abstraction, which allowed for the computational representation of phenomena and their interactions, restrictions and relationships. Also, abstractions of algorithms in several levels of computation allowed for a high reusability of software components and easiness of software development. The four levels of computation have different characteristics. It can be seen that the Global Skeleton and Phenomena Procedures are highly reusable and are critical to the simulator definition. The Block Skeleton is also very reusable, but, since it articulates the action of groups, if that changes the former will also be changed. Furthermore, since the simulator definition and problem specification are based on software components, a software architecture was defined, an a data base system (repository) plugged in the Plexus system, increasing its performance in the sense of reusability. For future works we consider the evaluation of distribution techniques, that can contribute to a cooperative environment, as well as the development of more studies.

References

[4] J. Grčar, Personal communication, Lawrence Berkeley National Laboratory, Mail Stop 50A-1148, One Cyclotron Road, Berkley, CA 94720, jfgrcar@lbl.gov


