Workflow for Simulators Based on Finite Element Method

Felix C.G. Santos, Mardoqueu Vieira, and Maria Lencastre

Federal University of Pernambuco - Department of Mechanical Engineering
Rua Acadêmico Hélio Ramos, s/n - Recife - PE 50740-530 - Brazil
fcgs@demec.ufpe.br, {msv,mlpm}@cin.ufpe.br

Abstract. Current workflow systems usually do not provide adequate support for workflow modeling. Real life work processes can be much richer in variations and more dynamic than a typical workflow model is capable of expressing; this means that the users need to be able to adjust workloads and modify workflow models on the fly [10]. Plexus, a system for the development of simulators [5,6], is a typical case where such a difficulty arises. Simulators provide an economical means of understanding and evaluating the performance of abstract and real-world systems; their design and implementation is almost as complex as the systems being simulated, to be efficient they must be adaptable to an ever-increasing system complexity. The use of workflow technology helps the development of more flexible and versatile strategies. This paper proposes a workflow management framework, called GIG, for controlling the simulator workflows in the Plexus context.

1 Introduction

Due to tremendous ongoing activity in the fields of application of the Finite Element Method (FEM), there is a need for tools, which could help the development of simulators with a high reusability degree in both the academic and industrial worlds. Nowadays simulation systems supporting coupled multi-physics phenomena can be important predictive tools in many industrial activities. However, the need for suitable numerical tools, which could more appropriately simulate a large amount of coupled phenomena, and the need for computational environments, which could help the building of those tools, are still a reality. The Finite Element Method is a way of implementing a mathematical theory of physical behavior. Simulations using FEM can become very complex, particularly when the designer wants to guarantee high levels of abstraction and reuse of the developed solutions. Those requirements comprise the main strategies in saving the production costs of high quality simulation software.

This work was done as a part of the activities of Plexus, a project for the development of a computational environment that helps the design and implementation of simulation software for coupled phenomena, through flexible and friendly tools, based on the FEM [3,4,5,6]. By simulator we mean a computational system aimed at obtaining approximate solutions to systems of coupled

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2658, pp. 555–564, 2003.

partial differential equations, together with a set of restrictions (differential-algebraic relationships involving one or more vector fields). In this paper we analyse the importance of workflows in Plexus.

This work was devised from the experience obtained during the implementation of several simulators in the FEM context. Researchers of the Mechanical Engineering Department - UFPE wanted to organize their code in a way that was easier to adapt to new strategies and also to allow process reuse. So they designed and implemented the GIG, a generic interface graph, which provides a process to achieve the mentioned advantages. This workflow has been used in the development of FEM-systems and a variety of other numerical methods. Due to their good performance in different case studies we decided to standardize the proposed solution, extending it with some other workflow concepts, and also evaluating the final result.

The paper motivation can be summarize by the need for: (i) easiness of translating from the natural language representation of the processes into a computer representation; (ii) simplicity of use (iii) versatility in the implementation of solution processes in the FEM Domain (iv) reduction of the possibility of errors in the coupling of processes (v) support of adaptability at run time.

The paper is organized in the following way: section 2 presents some background about workflows; section 3 details Plexus System; section 4 describes the proposed workflow. Section 5 presents some conclusions.

2 Workflow Technology

Workflow technology and process support lies at the centre of modern information systems architectures [1]. Workflow can be defined as representing the operational aspect of a work procedure: (i) the structure of tasks and the application and humans that perform them; (ii) the order of task invocation; (iii) task synchronization and the information flow to support the tasks and; (iv) the tracking and reporting mechanisms that measure and control the tasks.

One of the chief tasks of workflow management system is to separate process logic from task logic, which is embedded in individual user applications. This separation allows workflow users to modify one without affecting the other, that is, the two can be independently modified and the same logic can be reused in different processes. This promotes software reuse and the integration of heterogeneous software applications [1].

The workflow management system is a system that completely defines, manages and executes "workflows" through the execution of software whose order of execution is driven by a computer representation of the workflow logic. These management systems automate the process logic, while humans and software applications perform workflow tasks, implementing the task logic. The majority of workflow systems share a small set of common features. Some common features of workflow systems are: (i) flow independence, (ii) domain independence, (iii) monitoring and history and (iv) manual interventions.

Considering the levels defined in [12], a Workflow Model system may be characterized as providing support for: (i) Built functions, concerned with defining, and possibly modeling, the workflow process and its constituent activities; (ii) Run-time control functions, concerned with managing the workflow processes in an operational environment and sequencing the various activities to be handled in each process; (iii) Runtime interactions with human users and IT application tools for processing the various activity steps.

3 Plexus System

Plexus is a system whose objective is to reduce the complexity and cost involved in the development and implementation of a simulation system, providing a more flexible environment with efficient techniques for coupled phenomena simulation. Despite of being a specific simulation environment, which uses FEM in coupled phenomena solution, the Plexus system can be used in several types of applications, allowing at first hand the modelling of different classes of simulators, considering a set of pre-defined features, for example, a solution scheme involving time stepping and adaptation.

The Plexus system is divided into 4 subsystems, representing the main processes, Fig. 1: (i) Administration/ System Loading, which supports the system management and the loading of general system data and metadata; (ii) Preprocessing, where the user inputs problem data, and where dynamic structures for a simulation are built; (iii) Simulation Processing, where data are processed to obtain the solution and where the verification occurs; (iv) 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 components, 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.

The Plexus simulation implementation is represented with the use of algorithm skeletons and also with other predefined object-oriented structures, like computational phenomena, exploring the FEM polymorphism. A computational phenomenon (for example a computational representation of heat transfer) is described by its vector field and weak forms defined in its geometric entity together with boundary conditions information, which is also implemented as fictitious phenomena defined on the respective geometric entity of the boundary of its domain. It has also Math Methods that implement: Mesh generation, Integration Rules, Shape Functions, etc [3,4,5,6].

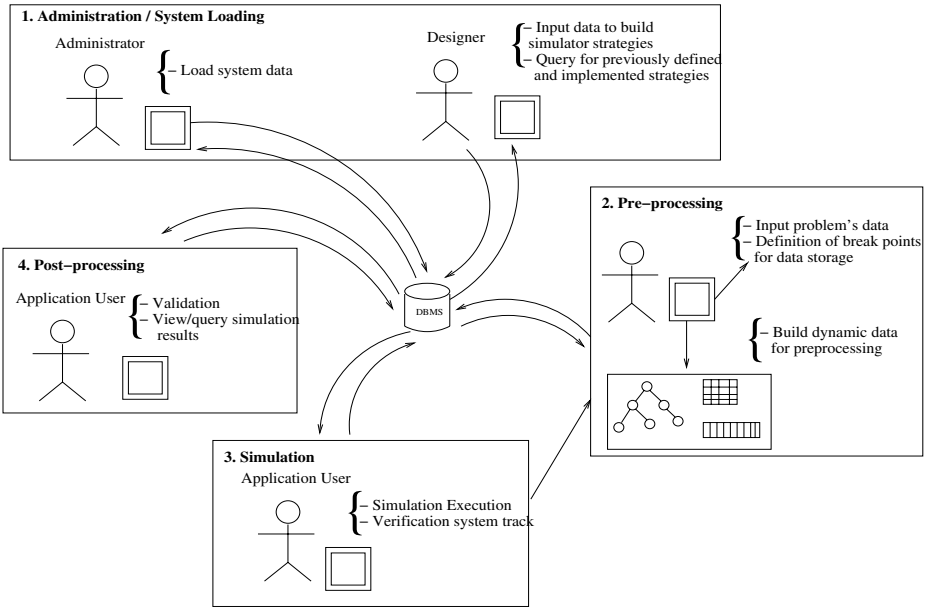

Fig. 1. Plexus Overview

The FEM-Simulator Skeleton pattern [3] suggests a FEM-Simulator algorithm organization within 4 levels of computational demands: Global Skeleton, Block Skeletons, Group Skeleton and the Phenomenon level. These levels were defined due to the high number of repeated (similar) structures and the degree of reusability of the involved algorithms, see subsection 4.1.

4 FEM Workflow

This section shows that the implementation complexity for coupled phenomena simulation can be greatly reduced by the use of predefined object oriented structures, due to FEM polymorphism, and also by the use of workflow and dataflow management. It proposes a generic framework for Plexus Workflow called GIG. By frameworks we mean, reusable semi-complete applications that can be specialized to produce custom applications [9].

GIG already attends, or can be easily adapted to, satisfy the following Plexus 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. GIG framework follows the object-oriented style (modeling and programming). For purposes of simplicity of use and easy correctness verification, the GIG is restricted to be a direct acyclic graph (DAG).

We assume that the simulator building and assembling will be based on a variable designer data model, which describes: the initial scenery, algorithm skeletons and auxiliary numerical methods, phenomena, geometry and so on. The initial scenery defines the class of problems that the simulator will be able to tackle in a broad sense. The simulator model is able of considering the use of many procedures (for instance: Time Loop; Adaptation Iteration; Time Step Estimation; Solution of Algebraic Systems; Error Estimation; etc), which may be either present or not, depending on the configuration of the initial scenery. The implementation of those procedures is done through the algorithm skeletons and auxiliary numerical methods, which define a procedure within the 4 levels of computational demand. The other data model abstractions (like computational phenomena, geometry and so on) are used to describe the problem domain. Some processes used during the simulation are encapsulated in those data models; in this work we will not consider such level of detail.

In what follows we will describe the FEM process and the GIG solution.

4.1 FEM Simulator Process

The FEM simulator process can be basically defined by 4 levels of computation demands (skeletons and methods). From the first level down to the last one, the procedures get more and more specific, making it possible to separate the most reusable components. Thus, in each level, the so-called skeletons represent the activity flows. The skeletons are those parts of that solution process which can be replaced, making it possible to build different solution strategies. Each skeleton is able of articulating skeletons in the immediate lower level. When they are assembled together, they represent the entire solution process.

We detail, in what follows, each level:

- Global Skeleton is the first level of computation and represents the global algorithm skeleton (the core of the simulator). It is unique for each simulator, but may be replaceable, producing another simulator. The global algorithm skeleton articulates the procedures involving all blocks.
- Block Skeletons 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.
- Group Skeletons 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 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.

– 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 parameterised way. The phenomena classes will be composed of phenomenon data and a group of numerical methods (MathMethods), which are replaceable.

4.2 GIG Solution

The GIG solution propouses starting from an algorithm in natural language. The procedure is first divided into different algorithm nodes and is organized in the form of a graph. For the whole algorithm there is a data repository, which contains the existing data decomposed into different algorithm data classes.

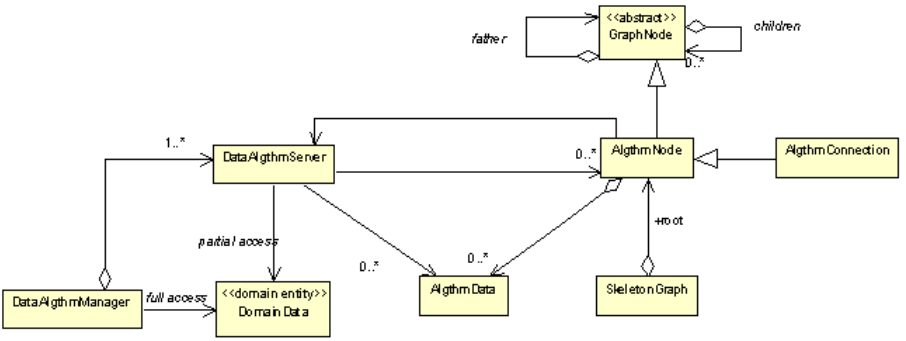

Fig. 2. GIG UML Class Diagram

The GIG framework structure is show in the Fig. 2. As we can realize, the GIG is composed of the following participants:

DataAlgthmManager its function is to control the whole workflow, managing its lifecycle. This involves build, reprogramming the application workflow at runtime.

DomainData represents the whole set of objects, which represent the problem domain.

GraphNode represents the relationship of the workflow nodes. It is an abstract class that implements low level operations to visit the graph.

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.

AlgthmNode represents the procedure (algorithm) of each workflow node. It is used as a base class for all algorithm classes of the application. We can say that it is a white box [9] of the GIG framework.

AlgthmData represents a data type to be used by an AlgthmNode during workflow execution. It is used as a base class for all algorithms data classes of the application.

DataAlgthmServer it provides a service that relates AlgthmNode with AlgthmData to be used in the building and modification of a workflow graph.

AlgthmConnection represents an algorithm that was not expanded 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.

4.3 Workflow Building and Execution

The workflow building and execution process is decribed in the UML sequence diagram of the Fig. 3. Initially the application send a request to *DataAlgthmManager* for building a workflow starting with an identification of the desired driver component. *DataAlgthmManager* forwards this request to *DataAlgthmServer*, which is the provider of all data and algorithm components. Then, *DataAlgthmServer* creates an object of the class *SkeletonGraph* and the *AlgthmNode* correspondent to the driver component. Next, *DataAlgthmServer* asks the *SkeletonGraph* to build the graph which in turn asks the Root (the driver) to recursively build the entire graph. The driver, then, asks *DataAlgthmServer* for its *AlgthmData* and its children *AlgthmNodes* objects. After that it asks each one of its children *AlgthmNodes* to build the graph recursively. This process goes on until all nodes of the workflow are created and assembled.

When it is the time for the application to execute the workflow, it asks SkeletonGraph to execute it. Then the SkeletonGraph asks its Root (driver) to execute its algorithm. Next, the root starts executing its procedure, which involves the execution of its children, and the execution of the entire workflow unfolds.

4.4 Considerations

Plexus deals with a fixed domain, where high levels of FEM abstraction were modeled [3,4,5,6,]. For the whole solution, hierarchical levels of processes were defined, each one with several possibilities of algorithms. Plexus suggests an architecture of component-based systems, which are significantly more powerful than that of traditional monolithic integrated solutions because they are easier to understand, adapt, reuse, customize and extend. Plexus clearly identifies levels of its architecture where the workflows are to be dynamically defined, built and controlled.

A Workflow Model system may be characterized as providing support for: built-functions, run-time control functions and runtime interactions (see section 2). We have separated the run-time control functions in the GIG solution; the other workflow model functions were just left to Plexus.

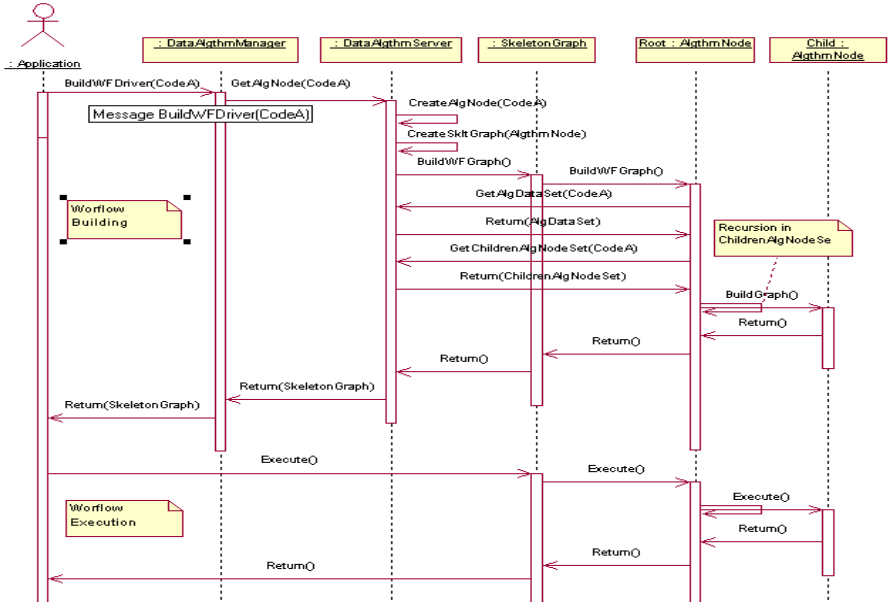

Fig. 3. Workflow building and execution

We can identify many advantages that GIG framework brings to FEM simulators development, like:

- Simplicity and easier production of algorithms from natural language: Plexus main expected users are scientists and engineers who develop or use FEM simulators, and already deal with development of FEM codes in some level. This users usually program in procedural style. The requirements of simplicity, easier production of algorithms from natural language into a computer (executable) representation and easier integration with object-oriented application, are very relevant. The GIG solution tries to attend this requirement. It allows an easier organization in a graph level, allowing the distribution of code in a very flexible way, not compelling a rigid division of code, like in [1]. The skeletons levels of computation (global skeleton, blocks and groups skeleton algorithms) are easily implemented in the GIG proposal.
- Restriction of Flow Granularity: An important consequence of the scale difference between micro workflow and macro workflow concerns users and activities [1]. Macro workflow targets process designers who typically are non-technical users; workflow activities involve applications and humans. In contrast, micro workflow targets people who build applications. At a smaller scale, micro workflow involves the objects that make up object-oriented applications. Plexus needs a mixture of small-scale and large-scale workflow, once its designers are also the programmers. The scales go from largest down to smallest, when the levels of computation go from the highest down to the

lowest. GIG allows a flexible representation for a mixture of scales, since it does not restrict the levels of programming into which the code is defined.
- Solution independence, easiness to change: Due to the frequently changing of numerical methods (for achieving the more suitable ones), FEM simulators usually suffer adaptations, forcing heavy reprogramming. The solution independence is important in order to allow the designer to specify the system features and strategies, supporting high level of control. The support for system reconfiguration and versatility is a desired feature. Flow independence guarantees more flexibility and reuse. In GIG, it's easy to change parts of the GIG graph, maintaining the desired ones intact. Also the levels of computation allow the tracking of the right process substitution.
- Providing Dynamic Procedural Programming and OO abstractions: Plexus maintains two paradigms: object oriented abstractions and the procedural programming (so common in scientific algorithms). The abstractions found in the FEM domain yield powerful and reusable systems through the established concepts of simulator, computational method, algorithm skeletons, and the defined levels of computation. On the other hand, the workflow perspective solves problems related to dynamic programming, which is very important in the production of FEM simulators by Plexus. So workflows following an object-orientation style can be of great help.

5 Conclusion

In this work we analysed the GIG solution, identifying some aspects of the solution domain that lead to more simplified workflow systems considering also some object-oriented aspects. As any workflow, GIG separates process logic (simulator definition and configuration) from task logic (problem strategies), which is embedded in individual user applications, allowing the two to be independently modified and the same logic to be reused in different cases.The best of GIG is that it is an object-oriented framework, which provides a powerful workflow control with simplicity of specification, programming and use. However, GIG, is not yet implementing monitoring, history and manual intervention and explicit parallelism control.

The Plexus system clearly identifies levels of its architecture where the workflows are to be dynamically defined, built and controlled. It is worthwhile observing that the proposed solution makes it easier and flexible the simulator definition and implementation, with a smaller overhead than other workflow management systems.

The Plexus system, focus on the development of simulators for chemo-thermo-mechanical interactions, which occur inside a given system and between such a system and its surrounding environment. However, the proposed GIG framework can be applied to the development of general-purpose systems, despite of being adequate for the underlining Plexus context.

References

1. Dragos-Anton Manolescu, *Micro-Workflow: A Workflow Architecture Supporting Compositional Object-Oriented Software Development*, Ph.D, Department of Computer Science University of Illinois at Urbana-Champaign, 2001.
2. Fiorini S. Leite J.P. and Lucena C.J., *Process Reuse Architecture*, CaiSE 2001, LNCS pp 284–298.
3. Lencastre M., Santos F. Rodrigues I., *FEM Simulator based on Skeletons for Coupled Phenomena (FEM – Simulator Skeleton)*, The Second Latin American Conference on Pattern Languages of Programming SugarloafPLoP'2002 Conference, Itaipava, Rio Janeiro, Brasil.
4. Lencastre M., Santos F., Rodrigues I. *Data and Process management in a FEM Simulation Environment for Coupled Multi-Physics Phenomena*, Fifth International Symposium On Computer Methods In Biomechanics And Biomedical Engeneering; 2001 Rome- Italy.
5. Lencastre M., Santos F., *FEM Simulation Environment for Coupled Multi-physics Phenomena*. Simulation and Planning In High Autonomy Systems – AIS2002; Theme: Towards Component-Based Modeling and Simulation. Lisboa, Portugal, 2002.
6. Santos F., Lencastre M., Araújo J., *A Process Model for FEM Simulation Support Development*, SCSC 2002 – Summer Computer Simulation Conference, US Grant Hotel- San Diego, California, 2002.
7. Yoder J., Balaguer F. and Johnson R., *Architecture and Design of Adaptative Object-Models*, 2000.
8. Roberts D., Johnson R., *Evolving Frameworks: A Pattern Language for Developing Object Oriented Frameworks*, University of Illions.
9. Fayad M., Douglas S., Johnson R., *Building Application Frameworks: Object- Oriented Foundations of Framework Design*, Wiley Computer Publishing, 1999.
10. Kwan M.M., Balasubramanian P.R., Dynamic Workflow Management: A Framework for Modeling Workflows,System Sciences, 1997, Proceedings of the Thirtieth Hwaii International Conference on , Volume: 4 , 7–10 Jan 1997 Page(s): 367–376 vol.4.
11. Workflow Management Coalition: The Workflow Reference Model, Workflow Management Coalition Specification, – Winchester, Hampshire – UK, 95.