Transferring Large Eddy Simulation tools from laboratories experts to industry users: a challenge for the INCA community

A. Dauptain [a], G Frichet [a], F. Duchaine [a], E. Riber [a], G. Dejean [a], T. Poinsot [b],

[a] *CERFACS, 42 avenue G. Coriolis, 31 057 Toulouse Cedex 01, France*

[b] *IMFT, 1 allée du professeur Camille Soula, 31 400 Toulouse, France*

Received *****; accepted after revision +++++

Presented by A. Dauptain

Abstract

This paper describes a methodology to build automatically industrial-proof versions of research softwares developed in the academic community (typically Large Eddy Simulation tools for combustion). The present methodology is built upon feature modeling, an application of graph theory to computer science. The necessity of such tools within INCA is first demonstrated and examples highlighting their advantages are discussed on the basis of a project called C3S developed by CERFACS and SAFRAN in the last four years. In particular, such an approach is compulsory to master the complexity of multi-models simulations, and proved to be beneficial for both academic and industrial users. This point is evidenced in the present paper with an application to conjugate heat transfer in a gas turbine solved by code coupling.

To cite this article: A. Dauptain, G. Frichet, C. R. Mecanique 333 (2005).

Résumé

Transfert des outils de Simulation des Grandes Echelles des experts scientifiques vers les bureaux d'études industriels : un défi pour la communauté INCA. Cet article décrit une méthodologie pour construire automatiquement des versions de logiciels scientifiques, en particulier des outils de simulation des grandes échelles pour la combustion, prêtes à être utilisées par des bureaux d'études industriels. Cette approche est basé sur le "feature modeling", une application de la théorie des graphes à la science du développement informatique. Le besoin de tels outils au sein d'INCA est tout d'abord évalué, et des exemples montrant leur avantages sont discutés d'après l'expérience d'un projet en cours depuis 2006 appelé C3S entre le groupe SAFRAN et le CERFACS. En particulier, une telle approche est obligatoire pour maîtriser la complexité de simulations multi-modèles, et s'est avérée avantageuse pour la communauté scientifique comme pour la communauté industrielle. Ce point est mis en évidence par une application au transfert de chaleur dans une turbine à gaz résolue par un couplage de codes.

Pour citer cet article : A. Dauptain, G. Frichet, C. R. Mecanique 333 (2005).

Key words: Feature Modelling ; Graph Theory ; Multiphysics

Mots-clés : Modélisation par fonctions ; Théorie des graphes ; Multiphysique

1. Introduction

Large Eddy Simulation (LES) and multiphysics computations are the two methods which are presently changing research in combustion simulation but also design of combustors in industry. The present paper discusses an issue which has never been considered before within the INCA community but is now becoming a first-order priority, according to many present actors working in this field.

LES and multiphysics are extremely sophisticated methods, developed by INCA laboratories for their own research activities. Because of their success, time has come to transfer these methods towards industrial applications. Running these simulations in laboratories *for* industry is not sufficient anymore: these computations must now be performed *by* industry. At the same time, they must also be transmitted to new generation of researchers (PhDs and research scientists).

Have we thought enough of what this transmission of knowledge means and how it must be organized? Even the apparently simple tasks of the necessary improvements for industrial use and the exponential amount of degrees of freedom for an industrial case are very complex issues which have never been really addressed within INCA. CERFACS experience was that typically 50 to 80 percent of CPU time invested in most LES was wasted because users had misjudged the importance of some submodels, or simply entered wrong parameters in the multiple input files made available to them. More importantly, these CPU time losses were accompanied by endless discussions and fruitless interactions between academic experts and industrial users. This was not surprising: LES is a revolutionary method, introduced for combustion only since 2000, still under development in many laboratories and far from the scientific maturity found in many other fields. Multiphysics computations are even more complicated. SAFRAN and CERFACS agreed

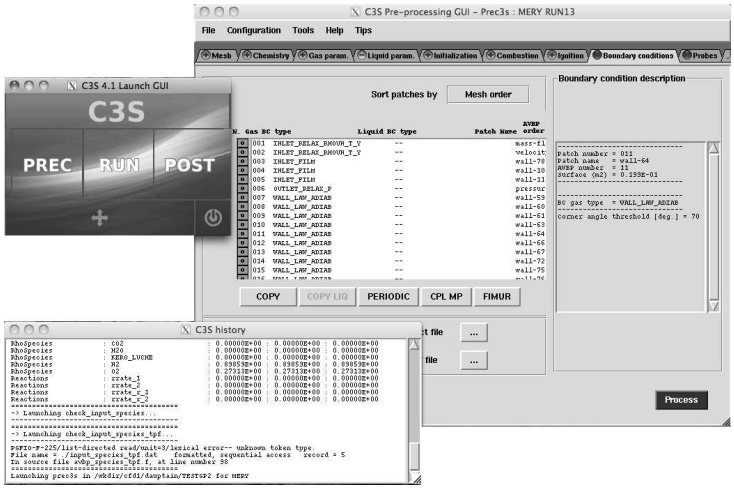

Figure 1. Snapshot of the C3S 4.1 graphical user interface, which is able to setup AVBP 6.2β simulations, submit and retrieve the jobs on distant computers (CEA's Titane, Platine, CINES's Jade), store and post-process (Ensight, Paraview, Tecplot) the dataset produced in a traceable and secured project.

that this issue was the most difficult one in the process of making LES or multiphysics a real design tool for industry and that a key question to tackle these difficulties was the maturation of the AVBP code [7] co-developed by CERFACS and IFP into an industrial-proof version. In the last four years, CERFACS and

Email addresses: Antoine.Dauptain@cerfacs.fr (A. Dauptain), Guillaume.Frichet@cerfacs.fr (G Frichet), Florent.Duchaine@cerfacs.fr (F. Duchaine), Eleonore.Riber@cerfacs.fr (E. Riber), Gerard.Dejean@cerfacs.fr (G. Dejean), Thierry.Poinsot@imft.fr (T. Poinsot).

SAFRAN have been involved in a first large-scale effort to build a Graphical User Interface (GUI) allowing SAFRAN engineers but also new PhDs to work with AVBP. A first version of such an interface for AVBP called C3S (Chaîne de Calcul Combustion Safran) was built and is now used as the standard tool in most SAFRAN centers but also at CERFACS and other laboratories in Europe (Fig 1).

CERFACS is now involved in a more ambitious project C3S**M** aiming at the construction of a unified industrialization platform for INCA which will be able to incorporate other codes. YALES2 [11], and AVSP [3]. will be the first codes in addition to AVBP included in this tool. The main objectives of C3S**M** are:

- Allow non-expert users to choose the level of modelisation adapted to their problems.
- Allow academic experts to give a fast and efficient support to non-expert users.
- Allow academic experts of INCA to access all C3S**M** tools.
- Allow academic developers of all codes of interest for INCA to build their own industrialized version (and GUI) in a simple way, whitout specific formation, nor the need to interact with the C3S**M** development team.
- Allow systematic verification and bug detection before run time by checking the validity of the parameters set by the users.
- Provide a total traceability of simulations performed by all users.

The present paper starts with an overview of the C3S project in Section 2. This software is now used by 150 people in the INCA community and is a good test case for industrialization technologies as well as an excellent working basis for C3S**M**. Based on this practical experience, a general model for computation setup is proposed (Section 3). An implementation is then suggested and main issues are discussed: how can industrialization work be split, how to build a common language to describe codes, how to build a unified GUI engine for all applications (Section 4). The proposed methodology is based on recent developments from science of computer programming (Feature Model diagrams, Model Driven Architecture concept) which give an answer to all bottlenecks currently identified through the C3S experience. An illustration of this implementation is commented with the application of the concept to code coupling in Section 5.

2. Industrialization of the LES code AVBP: the C3S project

The C3S project is an industrialization effort started in 2007 and funded by the SAFRAN group. The objective was to build a GUI to provide to the aeronautic engineers the ability to use the combustion LES code AVBP. This project is still running successfully: the GUI meets the needs of industrial users and in 2011, LES of non-reacting/reacting mono-phase/two-phase flows can be set up by the engineers in Villaroche, Bordes, or Vernon. Several unexpected aspects of the work detailed in this section are of particular interest.

2.1. *The industrialization benefits on CPU/human time waste*

Evaluating the human/CPU time wasted along the study of an industrial configuration is subjective. To give an estimation, a small survey was done on four distinct usual configurations (compressible, multi-species, reactive) under study at CERFACS between April and July 2011.

The minimum number of parameters to be set in the input files for such simulations exceeds in all cases 150, most of them corresponding to the specification of boundary conditions. Of course, some of these parameters can be introduced by cut-and-paste, but this very same operation can introduce errors by itself. Therefore a quality of 9 successful setups over 10 implies a typing quality of less than 1 error out of 1500 inputs in the most ideal case.

In parallel, the computer support group of CERFACS was investigating the simulation projects stored on the in-house supercomputers hard-drives : in an AVBP project folder, the ratio of files accessed (i.e. files produced then copied/compressed/downloaded) versus total files was roughly ranging from 1 to 20%; in an AVBP project folder managed by C3S, the ratio was always more than 50 and often 100%. This ratio can be seen as a zero-order approximation of CPU power wasted. Industrialization had a strong positive impact on this problem by rising the quality of inputs (for example, the values of input files parameters entered by users are systematically checked and values such as $P = -1\emptyset13\emptyset1\emptyset1Pa$ or $T = 3\emptyset OK$ are refused) and by introducing a strong traceability between the datasets produced and their inputs. As a result, non useful LES (runs performed with wrong input parameters or wrong choice of models) have decreased by a factor larger than two, leading to a significant CPU time reduction. In terms of data storage, a significantly larger amount of the datasets produced is duly stored and reduced to the pertinent files.

2.2. *The hidden activity of refactoring and gluing*

Industrializing a code is often associated to the creation of a GUI. As a matter of fact, a significant part of the work was performed in AVBP itself to increase the robustness of the tools, to ensure the portability to the various flavors of operating systems, to ensure compatibility of existing models, and above all to make it *GUI-compliant*: a GUI controls a software through a dialog, by standardized input/output files and sound error handling. Ensuring these basic expectations has consumed 80% of the human resources in the C3S project. As an unexpected benefit, this activity induced an in-depth code refactoring [9] for the solver, substantially increasing the quality of the code for all users.

2.3. *The hidden activity of hotline and support*

The hotline task is necessary to support users. From the C3S experience, some characteristic figures can be outlined from phone records between June 2010 and June 2011. The traffic observed in average was two calls per day and 1.45 hours of communication per week, with maximal peaks around 5 hours a week after each release. This demonstrates that the hotline activity took a significant but acceptable amount of time from the two persons in charge of C3S at CERFACS. As an unexpected benefit, this activity induced regular contacts and confidence between the partners. This situation eventually created various new opportunities of (funded) academic/industrial collaborations *provided the work was done using the official GUI.*

In traffic engineering, the carried traffic is the average of concurrent calls to a hotline on a period of time. It is a non dimensional number expressed in erlangs. In the present case, 1.45 hours of traffic on a weekly basis of 40 hours yields 0.036 Erlang. The *escalation* process is the decision to re-direct an incoming call to a specialist. For a smooth operation of the call center, non-escalated calls are kept short, less than five minutes, and must represent a large majority. In the present case, more than a half of the calls exceeds five minutes and would probably require an escalation. These two last figures prevent the hotline outsourcing to a call center: calls too scarse, and half of traffic would need a redirection to researchers.

2.4. *The need of a new method to industrialize solvers*

The C3S project is a success in the sense that the initial 2007 strategy -a small group of permanent researchers gluing an homemade GUI to a LES code- reached the initial 2007 aim -making industrials able to use by themselves, from their offices, the LES code-. Today, objectives have changed: industrials want more solvers, more quality in service, more multiphysics available, while academics seek a way to reduce

the weight of industrialization, and redirect its manpower and funds to research topics. Consequently, a new generation of industrialization is needed.

3. The strategy

The previous section highlighted two major constraints: increase the number of industrialized solvers while reducing the weight of industrialization for research groups. In the C3S project, there is a strong overlap between the solver team and the industrialization team: new models introduced by Phd's must be tested, understood, and generalized by both teams before being compliant to GUI. The overlap is illustrated in Fig. 2a for the C3S project. The same overlapped organization for multiple codes in Fig. 2b shows a linear increase of the weight of industrialization, particularly on the industrialization team. Note that the added weight of code coupling is not taken into account here.

The pivot of the problem is therefore "Is it possible to separate conveniently the work of solver teams and industrialization team?"as sketched in Fig. 2c: solver teams reach by themselves the status of a GUI-compliant code, while industrialization team focuses on the releases of a unified product. The path towards such an Industrialization Task Separation (ITS) is discussed hereafter.

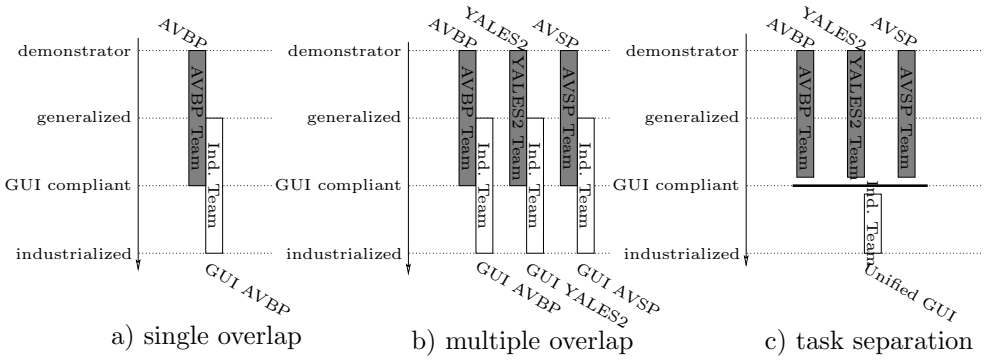

Figure 2. Strategies of extension of the C3S organization to multiple codes.

Tiwana *et al.* [12] explain the knowledge overlap by the constant need of both teams to match a foreign concept: industrialization requirements for solver team, physical model peculiarities for the industrialization team. This two-way learning is done through an intense team-to-team communication. The creation of a common language i.e. a standardized description of solvers, is the logical shortcut.

A similar problem is encountered in software engineering with Software Product Lines (SPL) when it comes to create a collection of similar software systems from a shared set of software assets using a common means of production. In the SPL context, the common language is called Feature Modelling. Feature models were first introduced in the Feature-Oriented Domain Analysis (FODA) method by Kang in 1990 [10]. A "feature" is defined as a "prominent or distinctive user-visible aspect, quality, or characteristic of a software system or system" [10]. It is possible to prove that the high complexity of a research software can be treated as a SPL, and to use the FM concept for its description. To keep the generality of the discussion, a formal model is needed to represent an arbitrary solver setup, like Kang *et al.* did for the 1990's software systems [10].

3.1. *The graph formulation*

From the user point of view, the input parameters of a solver (AVBP, YALES2, AVSP) are clustered in groups and subgroups, like in Table 1. However, the interdependencies between all parameters do not necessarily respect this hierarchical segregation. Both hierarchical vision and verification of dependencies are compulsory for an industrialized software: the user will use the first one to navigate through parameters, and the second one to know the implications of his actions. The best approach to describe hierarchical vision and verification of dependencies is to use graph theories.

First, a *parameter* is a variable to specify to the solver. A parameter can have many different natures: integer, real, boolean, choice, filename, coordinates. The ensemble of parameters, or *parametrization* defines a unique instance of the solver.

By definition [6], graphs describe the connectedness of systems and can help to create a formal model of parameter setups. A *general graph* is a set $\mathcal{V}$ of vertices with a set $\mathcal{E}$ of 2-subsets of $\mathcal{V}$ called edges. If the edges have an orientation so that they go from one vertex to another, they are *directed edges*, and the graph is a *directed graph*. In the present context, the vertices are linked to the parameters, and the directed edges to their dependencies. More precisely, each vertex includes a boolean information about the validity of the parameter. If the validity of parameter B needs to be tested when parameter A changes, the associated graph is $A \to B$. To ease the discussion, in the relation $A \to B$, A is the *child* of B and B is the *father* of A. Furthermore, validity is recessive, i.e. if $A_1 \to B$ and $A_2 \to B$ then B can be true only if both A_1 and A_2 are true. In other words, a parameter can be valid only if all its children parameters are valid.

The parametrization of an arbitrary CFD solver is shown as a directed graph in Fig. 3, taking into account the hierarchical dependencies only. In the hierarchical graph of Fig. 3, one can show that the n vertices are connected by exactly $n-1$ edges by construction, since all parameters are grafted either to the root vertex (CFD solver) or to a pre-existing vertex. By theorem [6] this graph is a tree, i.e. a connected graph without circuits. This particular graph allows a very efficient data storage [2] used by all filesystem browsers.

A second graph sketched in Fig. 4 includes the cross-dependencies between parameters of different kinds. By construction, the n vertices are connected by more than $n-1$ edges, excluding this graph from the tree family [6]. The descriptions of coupled parameters, like the choice between a tetrahedron-based numerical scheme and a hexahedron-based one are represented with a circuit (cf. example mesh file $\rightleftharpoons$ scheme of Fig. 4). [1] . These circuits rise the complexity of a graph.

The graph theory yields two conclusions:

(i) The hierarchical part of the setup, or "the way the user comprehend the setup", can be implicitly modeled by the structure of a directed tree.

(ii) The cross-dependencies can link any parameters, and cannot be implicitly modeled by the structure of a directed tree. If the tree structure is used, these dependencies must be explicitly declared.

Note that the implicit modeling of hierarchy makes native the "error tracking": according to the graph of Fig. 3, setting the integer parameter "Dimensions" to 1.3 makes the vertex "false". The path:

Dimensions→Domain→CFD solver

is recursively set to false. The user can quickly track down the parameter blocking the whole setup using this highlighted path. This process is illustrated in Fig.5.

The tree model considered until now is static, in the sense that no part of the graph can appear or vanish. The actual setup of a solver is more dynamic: the number of boundaries is not known in advance,

1. Coupled parameters are common in scientific solvers because they gather the different aspects of the same approach. For example, the wall modeling and the sub-grid scale modeling is a reccurent couple in Large Eddy Simulation solvers.

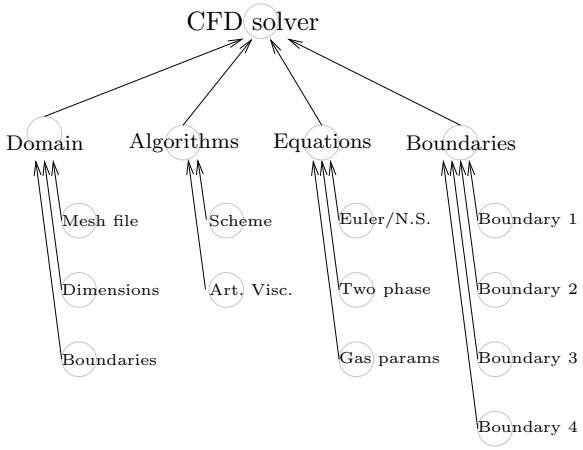

Figure 3. CFD solver information shown as a directed graph, showing only hierarchical dependencies of Table 1. 17 vertices for 16 edges.

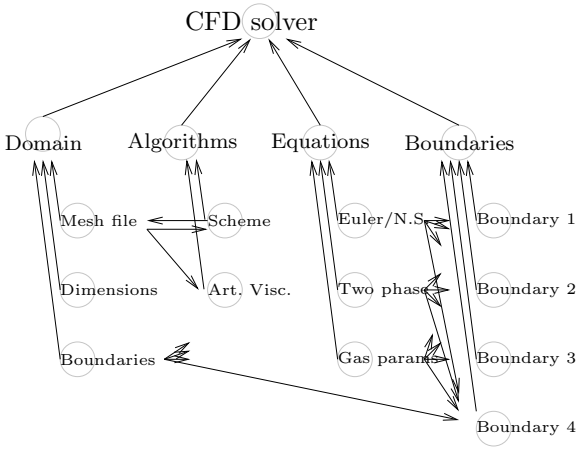

Figure 4. CFD solver information stored in as a directed graph including the cross-dependencies of Table 1. 17 vertices for 35 edges.

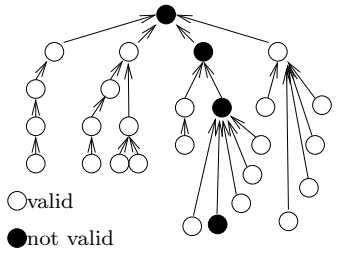

Figure 5. Path toward a invalid parameter using the directed tree structure. A non-validity is propagated to the ancestors. The search for the non-valid parameter among 27 possibilities is highlighted in 4 steps from the root.

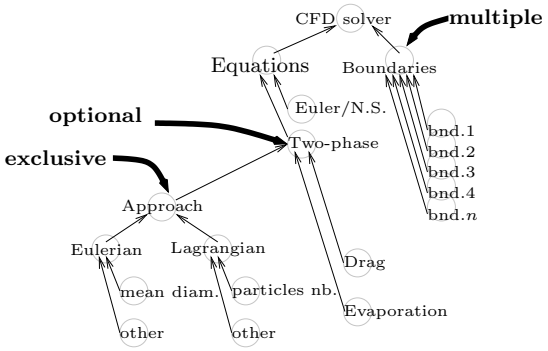

Figure 6. Three dynamic vertices: a multiple vertex for the boundaries, an optional vertex for single/two phase computations, and an exclusive vertex whose children are mutually exclusive.

some equations are optional, and some are mutually exclusive. Consequently, a supplementary property must be added to some vertices in order to allow the variety of setups, illustrated in Fig. 6. The exact property to add is discussed in the next section.

3.2. *The Feature Modeling*

The requirements or research softwares being known by the graph theory, are they compatible with the FM approach? Can a research software be regarded as Software Product Line (SPL) , i.e. a family of related programs. The basic Feature Model notation includes relationships between a parent feature and its child features:

– *Mandatory* – child feature is required. Can be extended to multiple.

– *Optional* – child feature is optional.
– *Or* – one or more sub-features must be selected.
– *Alternative (xor)* – only one of the sub-features must be selected

In addition to the parental relationships between features, cross-tree constraints are allowed. The most common are:

– A *requires* B – The selection of A in a product implies the selection of B.
– A *excludes* B – A and B cannot be part of the same product.

This basic model can be extended [8] by describing multiplicities of some mandatory (resp. optional) features: mandatory multiple means A must have 1 or more B children (resp. optional multiple means A can have 0, 1 or more B children). These six notations on a directed tree are sufficient to describe the parametrization of a research software. The parametrization of the LES code AVBP is showed thought a Feature Modeling diagram in Fig. 7. For the sake of clarity, only five out of the sixty boundary conditions available in AVBP 6.2β and only major parameters are shown.

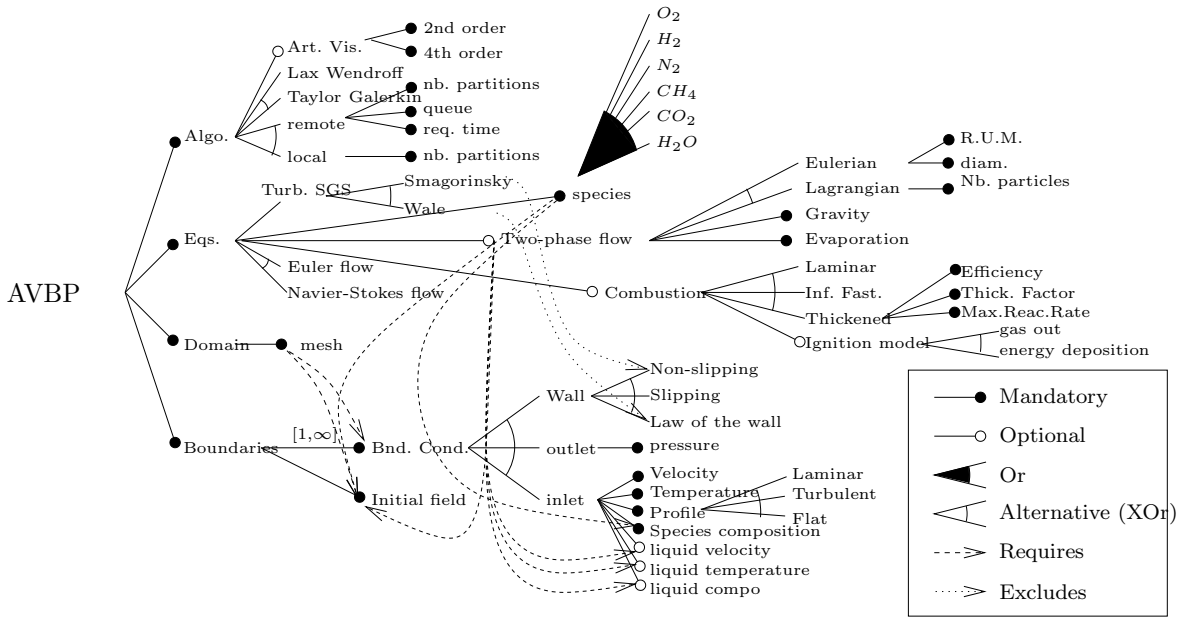

Figure 7. Feature Diagram of the LES code AVBP 6.2β using the notation of Kang [10]. The diagram is simplified: more than sixty boundary conditions are available, and only the major model parameters are shown.

4. Practical implementation

On the way to simplify the communication between solvers teams and industrialization teams, Boucher *et al.* [4] suggests a text-based approach to describe the softwares and shows an application of the concept to a family of printer drivers. This reduces the knowledge overlap to an exchange of text files. As the language needed to model research software will be used only reluctantly by solver teams, there is a strong constraint: this language must be "research-oriented" i.e. as explicit as possible, handled by the classical academic tools (editing with a *vi/emacs/xedit* console, grafting/pruning with a console *cp/mv/rm* or a

browser, management with a CVS/SVN-like file manager). The perception of this language by researchers is the cardinal point which will condition the ease of solver teams to reach the GUI-compliant state.

A Domain Specific Language is therefore the best option, with a tree-shaped data structure, and the six notations of FM to enrich the nodes. The present section explains why an eXtensive Markup Language (XML) is a good candidate to this purpose, what vocabulary is necessary for markups, and why a scattered cloud of XML files is more suited to the present context. Afterwards, a quick overview on the GUI engine completes the picture of the methodology.

4.1. *XML files with explicit Feature Model notations*

XML is a set of rules [5] to write data structures. Each file is composed of structured elements. An *element* begins with a start-tag and end with an end-tag. Attributes can be attached to start-tags to describe the element and content is what is between the start-tag and the end-tag. If no content is written, the element can just be an only tag named empty-element tag. Elements are nested into each other so that an element has always one father (except the first one which is usually the document) and can have children. The directed-tree structure is therefore implicitly described by any XML file, which is illustrated in Fig. 8.

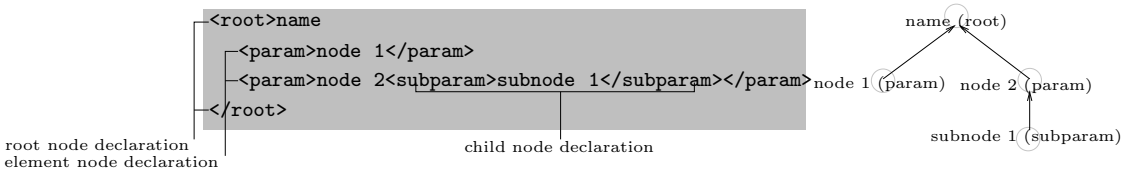

Figure 8. Simple example of XML-like file and its associated directed tree

The introduction of FM notations in the XML structure is a matter of vocabulary. A possible example is shown in Fig. 9. The six notations (mandatory /optional, or/Xor, require/exclude) are explicitly shown. In this example, both CFD and boundaries conditions are mandatory, but several nodes "boundary conditions" can exists. A verbose mode supported only in Euler resolution is secured by the required parameter. One can note that any choice (or/Xor) implies the creation of an intermediate node, which helps to store the choice result.

```
<model name="MySolver"  >
        <param name="CFL" type="mandatory" \>
        <param name="verbose" type ="optional" require="equations Euler" \>
        <param name="B.C." type="mandatorymultiple" \>
        <param name="passive scalar" type="optionalmultiple" exclude="species empty"\>
        <param name="species" type="or" >
                <choice name="hydrogen"\>
                <choice name="oxygen"\>
                <choice name="water vapor"\>
                <choice name="nitrogen"\>
        </param>
        <param name="equations" type="Xor" >
                <choice name="Euler"\>
                <choice name="Navier-Stokes"\>
        </param>
<\model>
```

Figure 9. Example of an XML file with the Feature Modeling (FM) notations

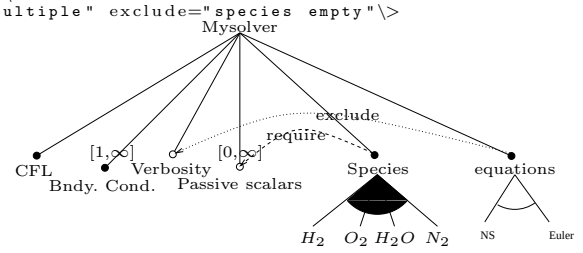

Figure 10. FM diagram associated to the XML-FM file of Fig. 9

4.2. *Scattering files*

The description of a full solver in its entire complexity in one file is possible but heavy. A alternative is scattering the solver description into smaller files. As files are stored in a computer within an arborescence, the tree structure of the arborescence can be used for in the description of the global tree using the "grafting" operation. The advantage of storing the description of each model into a specific file is twofold:

(i) Like any source code subroutine in academia, these small files can be edited with lightweight editors (e.g. vi), managed by release managers (e.g. SVN/ CVS), and installed/uninstalled from a console or a file browser. Adding simply one file to the arborescence gives more autonomy in a trial-and-error attempt. In all aspects, researchers can interact with these files like they do with source code.

(ii) Contractually, each of these files becomes the deliverable entities associated to the industrialization of the associated model. Each file has its own traceability, and its own confidentiality properties. The deliverable is paid to the solver team, redirecting the industrialization funding towards the scientific team.

4.3. *GUI engine using a Model Driven Architecture*

The remaining question is wether it is possible to construct a generic GUI engine based upon an arborescence of XML-FM files, which is a classical problem in software engineering addressed by the Model-Driven Architecture (MDA) approach, launched by the Object Management Group (OMG) in 2001. First the MDA approach defines system functionality using a platform-independent model (PIM) with an appropriate domain-specific language (DSL)-in the present case the cloud of XML-FM files- . Then, given a platform definition model the PIM is translated to one or more platform-specific models that computers can run.

Following this concept, the GUI engine is straightforward. Its structure is given in Fig. 11. First a *parser* checks for correct syntax and builds a oriented-tree data structure with the six FM components. A *widget generator* walks through the tree and creates a GUI widget adapted to each feature: an entry for a real, some radiobuttons for a choice, etc... Separately an *event generator* create the reactions of the software to the user sollicitations: making a certain choice requires the update of a certain amount of widgets. The combination of widgets and events is a GUI.

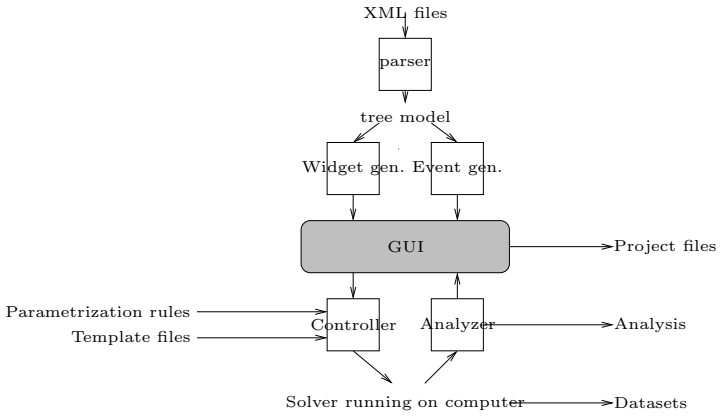

Figure 11. XML-based GUI engine

Once the parametrization is done, one action in the GUI asks to the *controller* to prepare the input files of the selected solver thanks to some *parametrization rules* (grammar of input files) and/or *template files* (pre-existing files with replaceable keywords) and submit the job to a computer. An other action in the GUI asks to the *analyzer* the monitoring of jobs, the retrieval of datasets, or more in-depth post-processing.

5. Toward multi-physics applications

A multi-physics application can be addressed either by a multi-physics solver (monolithic approach) or by several dedicated solvers that exchange boundary conditions (coupled approach). Industrialization of a monolithic approach is straightforward with the present methodology, but a coupled approach rises new issues. Conjugate Heat Transfer (CHT) problems treated by coupled legacy codes are a good illustration of these issues. This solution has the advantage of using existing state-of-the-art codes to solve fluid and solid equations and of being able to exchange one solver with another easily. The main drawback of this coupling methodology is that an adapted CHT framework is requested for the simulations especially on parallel machines. The performances of such a coupling framework are linked to (1) the strategy to couple the solvers in an accurate and stable fashion and to (2) the exchange of information between the solvers in an efficient and scalable fashion when using a large number of processors.

Point (1) imposes to be able to extract and to impose information in the legacy codes during the computation at given times. This work is done by collection of empty routines, or *User Defined Functions*, i.e. called at strategic places. The success of point (2) relies on a coupling library able to:

- efficiently connect coupled geometric interfaces (meshes or sub part of meshes) of parallel solvers distributed on a large number of processors,
- produce high quality interpolations of exchanged data.

The OpenPALM coupler [1] co-developed by CERFACS and ONERA tackles these issues. It is used to control AVBP for the resolution of the fluid part and AVTP [2] for the resolution of the conduction in solids. In addition to the AVBP and AVTP parameters, a set of *coupling parameters* has to be specified : the frequency of meeting points for data exchange, the number of meeting points, the location where information need to be exchanged, the type of information to exchange and some parameters for the interpolation.

The present industrialization methodology can be extended to the coupling of legacy code by *grafting* the solvers tree to an application-specific coupling tree, as illustrated in Fig. 12. Note that some exclusion/requirements will be necessary: impose temperature from fluid to solid and also temperature from solid to fluid for example will lead to exchange always the same quantity and ... no convergence.

6. conclusion

After a case study about a five year experience on a LES solver industrialization, a new framework for the industrialization of scientific softwares is developed. Elements of graph theory show how a massive part of dependencies between parameters can be implicitly imbedded in the data structure. A cloud of XML file is the basis of a domain specific language directly accessible to researchers. The general algorithm of GUI engine taking advantage of this DSL is depicted. The application of the concept of a LES solver/Thermal solver coupling is illustrated.

2. Parallel thermal solver developed at CERFACS.

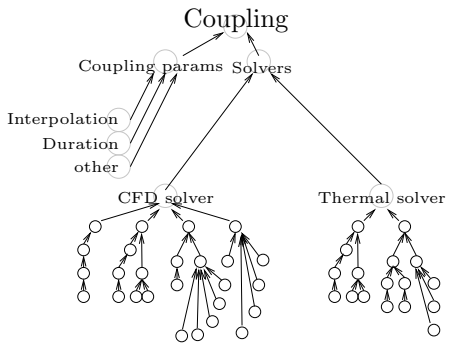

Figure 12. Grafting operation for a CHT coupling setup: a CFD solver coupled with a thermal solver

References

[1] A. Thévenin A. Piacentini, T. Morel and F. Duchaine. O-palm : An open source dynamic parallel coupler. In *IV International Conference on Computational Methods for Coupled Problems in Science and Engineering - Coupled Problems 2011*, Kos Island, Greece, June 2011.

[2] A.V. Aho, J.E. Hopcroft, and J. Ullman. *Data structures and algorithms*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1983.

[3] L. Benoit. *Prédictions des instabilités thermoacoustiques dans les turbines à gaz - TH/CFD/05/41*. PhD thesis, Université Montpellier II - DOCTORALE ISS: Spécialité Mathématiques et Modélisation, 2005.

[4] Q. Boucher, A. Classen, P. Faber, and P. Heymans. Introducing tvl, a text-based feature modelling language. In *Proceedings of the Fourth International Workshop on Variability Modelling of Software-intensive Systems (VaMoS'10), Linz, Austria, January*, pages 27–29.

[5] T. Bray, J. Paoli, and CM Sperberg-McQueen. Extensible markup language (xml) 1.0. 1999.

[6] P.J. Cameron. *Combinatorics: topics, techniques, algorithms*. Cambridge Univ Pr, 1994.

[7] O. Colin and M. Rudgyard. Development of high-order taylor-galerkin schemes for les. *Journal of Computational Physics*, 162(2):338–371, 2000.

[8] K. Czarnecki, S. Helsen, and U. Eisenecker. Staged configuration using feature models. *Software Product Lines*, pages 162–164, 2004.

[9] M. Fowler and K. Beck. *Refactoring: improving the design of existing code*. Addison-Wesley Professional, 1999.

[10] K.C. Kang, S.G. Cohen, J.A. Hess, W.E. Novak, and A.S. Peterson. Feature-oriented domain analysis feasibility study. *Software Engineering Institute, Pittsburgh CMU/SEI-90-TR-21*, 1990.

[11] V. Moureau, P. Domingo, and L. Vervisch. From large-eddy simulation to direct numerical simulation of a lean premixed swirl flame: Filtered laminar flame-pdf modeling. *Combustion and Flame*, 2010.

[12] A. Tiwana. Beyond the black box: knowledge overlaps in software outsourcing. *Software, IEEE*, 21(5):51–58, 2004.

Description of ...	**Physical problem**	**Solver information**	**Cross Dependencies**
Domain	Spatial location	Mesh	Algorithm
Equations	Physical phenomenons	Models parameters	none
Boundary	Frontiers definition	Boundary parameters	Domain, Equations
Algorithm	-	Numerical parameters	Domain, Equations

Table 1
Repartition of parameters in four arbitrary groups, from the user point of view