A Component Architecture for the Two-Phase Flows Simulation System NEPTUNE

Céline Béchaud [1], Marc Boucker [1], Alexandre Douce [1], Marc Grandotto [2], Marc Tajchman [3]
[1] EDF-R&D/MFTT, Chatou, France
[2] CEA-DEN/DTP/STH, Cadarache, France
[3] CEA-DEN/DM2S/SFME, Saclay, France

Abstract

Electricité de France and Commissariat à l'Energie Atomique have planed a large project to build a new set of softwares in nuclear reactors analysis.

One of the main idea is to allow coupled calculations in which several scientific domains are involved.

This paper presents the software architecture of the two-phase flows simulation Neptune project.

Neptune should allow computations of two-phase flows in 3D under normal operating conditions as well as safety conditions. Three scales are identified: the local scale where there is only homogeneization between the two phases, an intermediate scale where solid internal structures are homogeneized with the fluid and the system scale where some parts of the geometry under study are considered point-wise or subject to one dimensional simplifications.

The main properties of this architecture are as follow:

- coupling with scientific domains, and between different scales,
- re-using of quite all or parts of existing validated codes,
- components usable by the different scales,
- easy introducing of new physical modelings as well as new numerical methods,
- local, distributed and parallel computing.

The Neptune architecture is based on the component concept with stable and well suited interface. In the case of a distributed application the components are managed through a Corba bus.

The building of the components is organized in shell: a programming shell (Fortran or C++ routines), a managing shell (C++ language), an interpreted shell (Python language), a Corba shell and a global driving shell (C++ or Python).

Neptune will use the facilities offered by the Salomé project: pre and post processors and controls.

A data model has been built to have a common access to the informations exchanged between the components (meshes, fields, physical and technical informations).

This architecture has first been setup and tested on some simple but significant cases and is now currently in use to build the Neptune software.

Introduction

NEPTUNE is the two phase flow thermalhydraulic part of a new and large scientific software project started in 2001 by Electricité de France and Commissariat à l'Energie Atomique. This project includes other scientific domains such as neutronic and nuclear fuel studies. One of the main ideas is to have the ability to drive as realistic as possible numerical simulations of the behavior of a nuclear reactor power plant. At least this implies to run coupled calculations where different phenomena interact themselves : neutronic, fluid mechanic and structural mechanic ; we can think that chemistry phenomena can also play a role in these simulations. To make this simulations as easily as possible and to run them under the best computer efficiency it has been decided to build the new softwares with a new architecture.

This paper presents the choices that have been done for NEPTUNE .

NEPTUNE summary

The NEPTUNE project is the development of a new platform for two phase flow thermalhydraulic numerical simulations. It is based on a tool box of components and intends to allow industrial calculations as well as research studies.

The NEPTUNE project is not limited to its software part ; it includes also basic researchs on physical modelings and numerical methods and the realisation of experimentations for validation purposes.

Three scales have been identified for the numerical simulations: the local scale where there is only homogeneization between the two phases, an intermediate scale where solid internal structures are homogeneized with the fluid (this scale is typically used for core flows and heat exchanger flows like steam generators or condensors) and the system scale where some parts of the geometry under study are considered point-wise or subject to one dimensional simplifications but is able to compute full nuclear reactor loops.

Main purposes of the architecture

- interoperability: we want to set up as easily as possible studies where computations at different scales or from different scientific areas have dynamic exchange of informations ; in other words these are coupling processes,
- mutualization: we want to pool resources in software developing and to identify common basic programming elements.

  Exemple: in NEPTUNE there are elliptic as well as hyperbolic numerical solving methods, there are local scale as well as homogeneized scale approximations ; all these methods and scales will share some functions as for instance discretized data (mesh and field), material properties, physical closure laws or algebraic solver.

Constraints

- computations in the new architecture should be at least as good as the already available computations (non regression item),
- we have to be able to re-use parts of existing computer codes,
- as the thermalhydraulic calculations are very time and memory consuming and use very large data volumes, the computational efficiency should be of first importance. The parallel computation is one of the best way to do this and should be taken in account in the developments.

These constraints explain why we have to keep Fortran routines in the new architecture. There is indeed a very large basis of Fortran softwares in which we want to keep the best efficient numerical algorithms (solvers, eigenvalues computation) or very nicely physically validated functions (closure laws, thermodynamic properties).

Global description of the architecture

The figure 1 presents the basic organisation of the NEPTUNE architecture. The gray areas are the wrapping interfaces.

There are five levels:

- **C1**: components managing through the CORBA bus ; one can use a textual interface (Python) or a graphical interface ; this is the application interface level of the Salomé system,
- **C2**: components wrapping to make them usable in the Salomé system,
- **C3**: components managing out of the CORBA bus ; a textual interface (Python) is used but a graphical interface can also be set up ; one can also directly manage the components with a C++ main program,
- **C4**: components setting up and objects structuration ; the exchange data interfaces are defined at this level,
- **C5**: actual programming of the computational functions and methods.

The components

We call components elementary softwares that offer services in a specific domain. One access the components through their interfaces. They should be used when building a global application. The components exchange objects.

We give here a non exhaustive list of some NEPTUNE components.

- EquationsOfState: this component gives the thermodynamic properties of the fluids and more generally the physical properties of the materials.
- Interpolation: this component contains methods to interpolate fields between different discretizations.

- ClosureLaws: this component gives the physical closure terms of the partial differential equations that govern the two phase flows ; these are for instance the interfacial transfert terms.
- Solver: this is the algebraic solver component.
- BoundaryCondition: this component manages the set up of the boundary conditions.
- FiniteVolumeFlux: this component computes the different kind of finite volume fluxes (convective and diffusive fluxes in both the elliptic or the hyperbolic formulation).
- TimeScheme: this component gives the time schemes parameters.
- Problem: this components gives all the general informations that define the problem under study (the two phase flow model).

The libraries

The libraries gather basic routines of common interest and are used in the programming of the components. They also have their interfaces but the data exchanges are not always structured in objects. This can be as an example the basic routines that describe the thermodynamic properties of some fluids.

The interfaces

The interfaces are the most important point in the building of the new architecture. They should be designed with a very sharp attention. One can say that the interface design is more important than the inside programming of the components as updating an interface is more costly than updating an inner programmation.

The data exchange model

A common data exchange model implies that all the components have the same way to access the informations they exchange. Nevertheless the components may have their own inner data structuration. But the efficiency constraint would suggest to avoid internal structural conversions, in particular when the data volume is very large.

The data exchange model for the discretized data is set up by the MedMemory component. But there will be also necessary to have a model for the non discretized data (geometry, material description, global physical or numerical parameters)

The encapsulation

This is an intermediate step in the NEPTUNE project. It is done to introduce existing codes of interest in the new architecture without modifying them. They are taken as they are with just the addition of a

minimal interface that just contains the necessary input/output functions. Nevertheless it has been decided that encapsulated codes should access the data exchange model in order to keep the interoperability property.

The portability

The NEPTUNE platform should be portable on a large panel of computer architectures (operating systems). It is known that a good portability at the present time is a guaranty of the portabilty on new computer architectures that can appear in the future. We recall that the use of norms and standards, as well as the use of open source tools improve the portability and is worth to maintain in time the software investments.

The documentation

All the software parts should be correctly documented. This include:

- theoritical informations on the numerical methods and the physical modeling,
- the information concerning how to use each part of the software. It is mainly a full and clever description of the interfaces,
- the information about the inner programming of the software parts,
- several user examples,

Some specific problems

- Working at the C1 level, i.e. through a CORBA bus implies duplications of data ; as it is subject to be in a distributed architecture, copy of data between components is needed as they are running on processors that do not share their memory.
- The Python language has an efficient garbage collector ; it is important to make this garbage collector to manage the C++ memory areas ; it is generally obtained in declaring constructors and destructors in the C++/Python interface (SWIG).
- At the C1 level one can not use the Python garbage collector. It does not exist at this time a good solution to this problem: how to run a garbage collector through a CORBA bus.
- Common's in the Fortran language are not well suited in the new architecture. This is mainly a problem of data localisation. When one can not remove them it still exists a solution through a mapping of the Common's with C/C++ structures [3][5].

NEPTUNE implementation

The architecture previously described has been tested with success on the existing Fortran code of the 3D two-phase flow solver (local scale). In a first step, we choose to preserve the independance of the Fortran code from the C++ and Python parts. This is done by modifying the Fortran subroutines as little as possible when the code is decomposed into components and wrapped as objects.

At the present time, they are two components : the first one is devoted to the mesh reading and geometrical data calculation. This new geometrical component is based upon MedMemory, a library of C++ programs.

The second one consists in solving the set of two-phase flow equations. These components communicate with each other by the use of a Python script, which is in fact the main file of this new NEPTUNE code.

Firstly, this Python script instantiates a geometrical data object from the first component and then starts the time loop by creating a "3D local scale" solving object from the second component. This last component has several methods, which allow respectively to initialize variables, to initialize the time loop, to set boundary conditions, to calculate a single time step, to finish the time loop and to control the code output. All these methods are called by the Python script. It is also important to note that all Fortran COMMON are especially wrapped in order to access the contains of these COMMON from the Python script. This wrapping allows to control the simulation from the legacy Fortran routines as well as from the Python script.

NEPTUNE applications

- NEPTUNE **use cases**
  In order to assess the ability of the NEPTUNE software architecture to facilitate the coupling between the codes of different scales (*i.e.* "local", "intermediate" and "system", as mentioned above) and even different physical domain (*i.e.* thermal-hydraulics, neutronics, structural mechanics), a set of three use cases has been defined. These use cases correspond to actual PWR reactor issues.

  - SLB (Steam Line Break) case : this case represents a scenario of accident caused by a SLB, involving a system scale calculation (primary and secondary loops), an intermediate scale calculation (steam generator), and a neutronics calculation (which is out of the NEPTUNE scope), the three being coupled together ;
  - PTS (Pressurized Thermal Shock) case : this case represents a scenario of accident caused by a small break LOCA which have to be analyzed regarding the thermal shock undergone by the core vessel, due to the cold water injected by the emergency core cooling devices. This involves a system calculation (primary and secondary loop) coupled to a local scale calculation in the cold legs and the downcomer ;
  - SG (Steam Generator) case : in this case, we seek a better understanding of the flow in the upper plenum of a steam generator by coupling an intermediate scale calculation in the main part of the exchanger (where the tube bundles are located) with a local scale calculation in the upper plenum, where there is no obstacle ;

  These three tests will be carried out by the end of 2003.

- **Examples of 3D local scale applications**
  The current 3D local scale solver, based upon the general two-fluid model, has been recently developped following a finite volume multi-element numerical method [1]. This code is now integrated in the NEPTUNE software architecture, according to the "NEPTUNE implementation" above section. We show here two examples of test applications done with the current version of the code, in order to briefly illustrate its potential. The first one is a fast pressure blowdown case in a straight pipe, involving a strong phase change phenomenon, the SUPER CANON experiment [2]. In this 1D case representing a break in a primary loop of a PWR reactor, the pressure drops very fastly from 150 bars to 1 bar in a short time, causing the flashing of water. In figure 2 is shown the time evolution of the computed and measured pressure values near the open end of the pipe which is simulating the break. We can see that the results are in very good agreement with the reference result of the CATHARE [4] safety system code, and compare fairly well with the experiment data.
  The second example illustrates the ability to treat stratified two-phase flow. This axisymmetric 2D case consists in the collapse of a "cylinder" of water initially placed in the center of a cylindincal container [2]. In figure 3 and 4 are shown the liquid volume fraction at two different times : just after the initial state and at the moment when a first peak occurs at the center of the container. The results are in good agreement with the measured data, consisting of the height values of the two first center peaks (not reported here).

Glossary

bus:	communication area
container:	memory dynamic loading for a component
CORBA:	components communication norm
C++:	object oriented programming language
C++ to F77:	C++ - Fortran77 interface
DATA:	Salomé data component
F77:	Fortran programming language
Garbage collector:	automatic removing process for unreacheable memory areas
GEOM:	Salomé geometry (CAO) component
GUI:	Graphical User Interface
MED:	Data exchange model
MedMemory:	managing component for meshes and fields
mapping:	programming the same data structure with different languages
MESH:	Salomé mesh component
omniORB:	the CORBA bus used in SALOME
PTS:	Pressurized Thermal Shock
PWR:	Pressurized Water Reactor
Python:	interpreted command language
QT:	toolkit for GUI building
SALOME:	components managing service, based on CORBA and providing some general processes
SG:	Steam Generator
SLB:	Steam Line Break
SUPERV:	Salomé supervision component
SWIG:	Standard Wrapper Interface Generator, wrap C functions or C++ methods with a command language
TUI:	Textual User Interface
VISU:	Salomé visualization component

References

[1] N. Méchitoua, M. Boucker, S. Mimouni, S. Pigny, and G. Serre. Numerical simulation of multiphase flow with an elliptic oriented fractional step method. In *Third International Symposium on Finite Volume for Complex Applications*, Porquerolles, France, June 2002.

[2] S. Pigny and P. Coste. Two-phase flow averaged codes : Criteria for numerical methods. In *Two-Phase Flow Workshop*, Cargèse, France, September 2001.

[3] J. Sang, G. Follen, C. Kim, and I. Lopez. Development of corba-based engineering applications from legacy fortran programs. *Information and Software Technology*, 44:175–184, 2002.

[4] G. Serre and D. Bestion. Benchmarks calculated with the ICE numerical method using CATHARE and TRIO_U codes. ECUME project report, CEA/EDF, 2001.

[5] T. Slawig. Coupling distribute fortran applications using c++ wrappers and the corba sequence type. *Computer Standards & Interfaces*, 23:5–17, 2001.

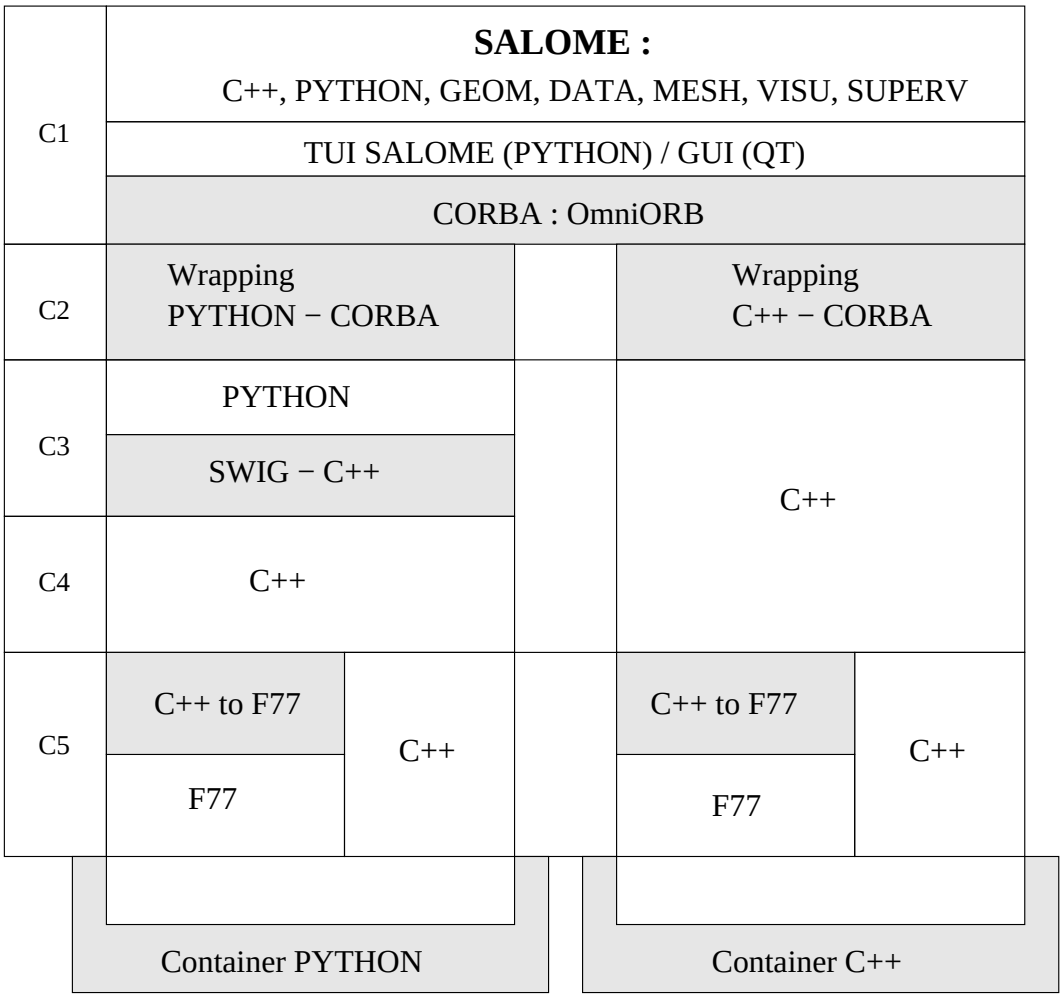

Figure 1: NEPTUNE Architecture.

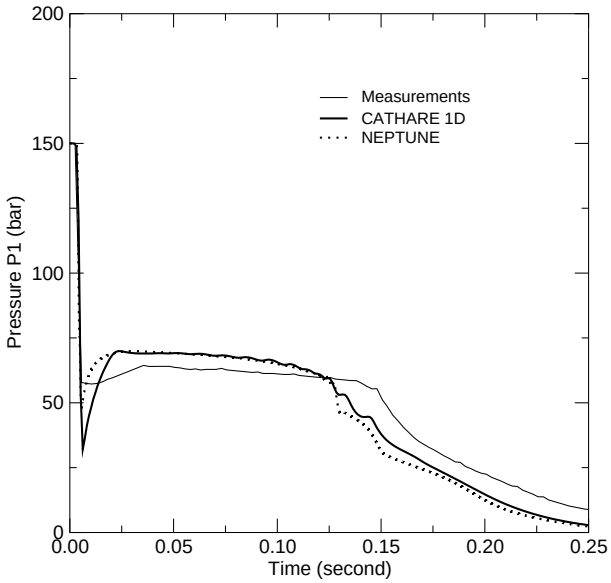

Figure 2: Pressure blowdown test case (SUPER CANON) : pressure time evolution at 50 *cm* from the break.

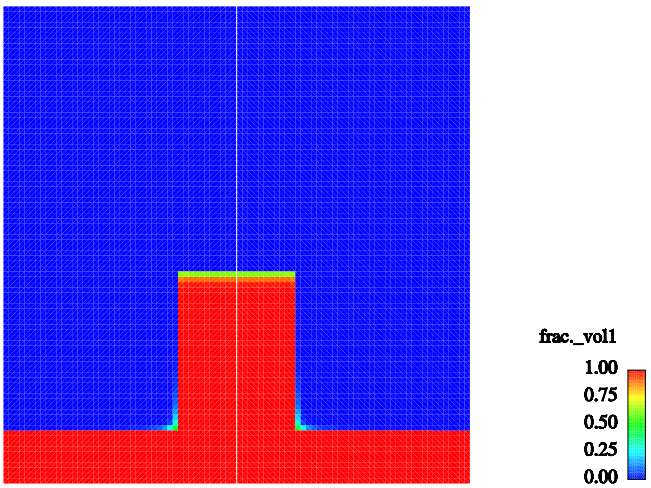

Figure 3: Sloshing test case (MASCHEK experiment) : liquid volume fraction at $t = 0.02\ s$ (close to the initial state).

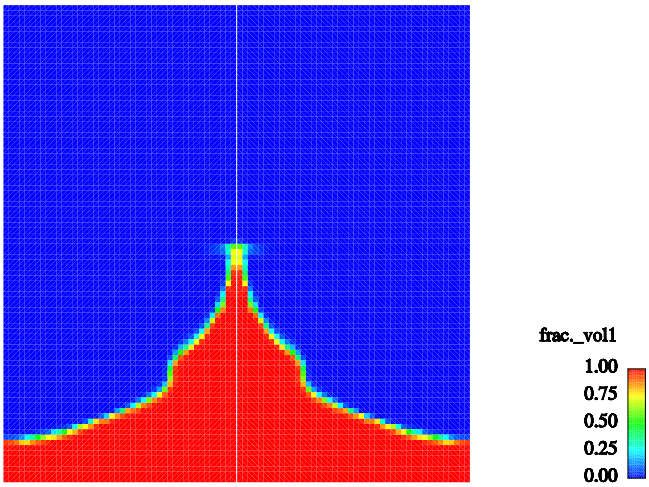

Figure 4: Sloshing test case (MASCHEK experiment) : liquid volume fraction at $t = 0.58\ s$ (first peak).