An integration platform for metacomputing applications

Toan Nguyen and Christine Plumejeaud

INRIA Rhône-Alpes
655, Avenue de l'Europe
Montbonnot, F-38334 Saint Ismier Cedex
Toan.Nguyen@inrialpes.fr

Abstract. Simulation and optimisation applications involve a large variety of codes that result in high CPU loads on existing computer systems. Advances in both hardware and software, including massively parallel computers, PC-clusters and parallel programming languages somewhat alleviate current performance penalties. It is the claim of this paper that formal specification techniques, together with distributed integration platforms, provide a sound and efficient support for high performance distributed computing in metacomputing environments. Formal specifications provide rigourous and provable approaches for complex applications definition and configuration, while distributed integration platforms provide standardised deployment and execution environments for coupling heterogeneous codes in problem-solving environments supporting multi-discipline applications.

1. Introduction

Simulation and optimisation applications, e.g., as used for digital mockups in aerospace design, and for numerical propulsion systems simulation [5], raise important challenges to the computer science community [9, 12]. They require powerful and sophisticated computing environments, e.g., parallel computers, PC-clusters. They also require efficient communication software, e.g., message passing protocols. Although these problems are not new, they have given rise to specific solutions that have become de-facto standard, e.g., MPI and PVM.
Simultaneously, computer science has provided interesting solutions to the problems of:

- formal specification of distributed and communicating systems
- standardized exchange and development environments for distributed applications

This paper explores the design and implementation of an integration platform which is a computerised environment dedicated to the *specification, configuration, deployment and execution* of collaborative multi-discipline applications. Such applications may

invoke a variety of software components from various disciplines that are connected together to collaborate on common projects.
It details the development of an integration platform called *CAST* (an acronym for "*Collaborative Applications Specification Tool*") allowing the execution of test-cases for aerodynamic design in the aerospace industry. The platform is based on an high-level graphic user interface (Figure 1) allowing the formal specification of optimisation applications, based on the work of Milner's process communication algebra SCCS [8]. It is deployed on a network of workstations (NOW) and PC-clusters connected by a high-speed network called VTHD [13] and communicating through a CORBA object request broker [10]. This was developed in part for a european Esprit HPCN project, called "DECISION": *Integrated Optimisation Strategies for Increased Engineering Design Complexity* [3].

The paper is organised as follows. Section 2 is an overview distributed integration platforms. It details the *CAST* integration platform dedicated to distributed multi-discipline applications, including example test-cases. Section 3 is a conclusion.

2. Distributed integration platforms

Research in operating systems has emphasised the advantages of software components for distributed computing, e.g., EJB (Enterprise Java Beans), CCM (Corba Components Model) [1]. They allow the definition, deployment and execution of distributed applications formed by various heterogenous software modules which were not necessarily designed to cooperate. Further, they can reside on heterogeneous platforms, provided that they communicate through a common hardware or software medium.

Hence, the idiosyncracies of the data and module distribution is left to the software developers. This is in constrast with dedicated standards that are used for parallel programming, e.g., MPI. Here, the developers must code inside the modules the grouping and spawning of the execution processes. The actual application code is therefore mixed with distribution and parallelisation statements. This is not the case with environments where modularity and transparent distribution is supported, e.g., CORBA. Here, the users' code can be used as is, without modifications.

A particular test-case will illustrate the following sections. It consists in the shape optimisation for an airfoil in stationary aerodynamic conditions [6]. The goal is to reduce the shock-wave induced drag (Figure 2). Two algorithms are involved. One, which computes the airflow around the airfoil, the other which optimises the airfoil's shape. The entry data are the airfoil shape (RAE2612), its angle of attack (2 degrees), the airflow speed (Mach 0.84) and the number of optmisation steps (100). This problem requires several CPU hours on an entry level SUN Sparc workstation running the Solaris 2.6 operating system.

2.1 Principles

CAST is an integration platform designed to fulfill the requirements of distributed simulation and optimisation applications in complex engineering design projects.
Example applications are industrial projects in the electronics and aerospace industry, in particular concerning multi-discipline optimisation, e.g., coupled aerodynamics, structure, acoustics and electromagnetics optimisation.
The target user population includes project managers and engineers that are experts in their particular domain, i.e., electromagnetics, aerodynamics. They are well aware of computer technology, but not necessarily experts in formal process specifications, theoretical computational mathematics, nor object-oriented, CORBA and components programming.
This constrains the integration platform to provide a high-level user interface, with no exotic idiosyncracies related to process algebras and distributed computing.
The fundamentals of the platform are therefore:
- to rely on a sound theoretical background for process specifications
- transparent and widely accepted standards for distributed computing
- transparent use of a large variety of software
- transparent use of distributed and heterogeneous computer platforms
- hign-level user interface for ease of use by end-users, that are experts in non computer science areas of interest.

The platform supports the definition, configuration, deployment and execution of multi-discipline applications distributed over NOW, LAN, WAN and PC-clusters.
They are formed by collaborating modules which may run on heterogeneous computers. The modules may be implemented using various programming languages and they dynamically exchange data.
They are synchronised by user-specified plans which include sequences, embedded and interleaved loops, and fork operators. These synchronisation operators are components of a process algebra which is primarily based on Milner's SCCS algebra [8].

For instance, the formal specification of the HBCGA (Hybrid Binary Coded Genetic Algorithm) wing shape optimisation example detailed in Section 2 produces the following SCCS formula:

```
InitBCGA:InitHybrid:BGGA:(TRUE:(END)+FALSE:(FUN:(TRUE:(
HYBRID:(TRUE:(=>InitHybrid)+FALSE:(=>FUN)))+FALSE:(=>BG
GA))))
```

where BCGA, FUN, HYBRID and the various INIT modules are connected by the loop operator, depicted by '=>', the sequence operator ' :' and the choice operator '+'. Other SCCS operators are available, including the synchronisation of tasks '&' and the parallelisation operator '/'. The icons on the left-hand menu are used to build the application on the screen by the use: they represent the SCCS operators. The

tasks are defined by the users on menu driven screens that are not depicted here. Note that this example involves two embedded loops and two interleaved loops.

The icons on the top menu line are tools to invoke various operations: opening of a file, undo/redo, save to file, etc. Predefined applications may be loaded and modified on-line or included in new applications, i.e., existing applications may be saved and reuused later to form new ones.
The SCCS formula is produced automatically by *CAST 2.0* from the application's graphic specification. It is used internally to generate the adequate distributed execution structure.

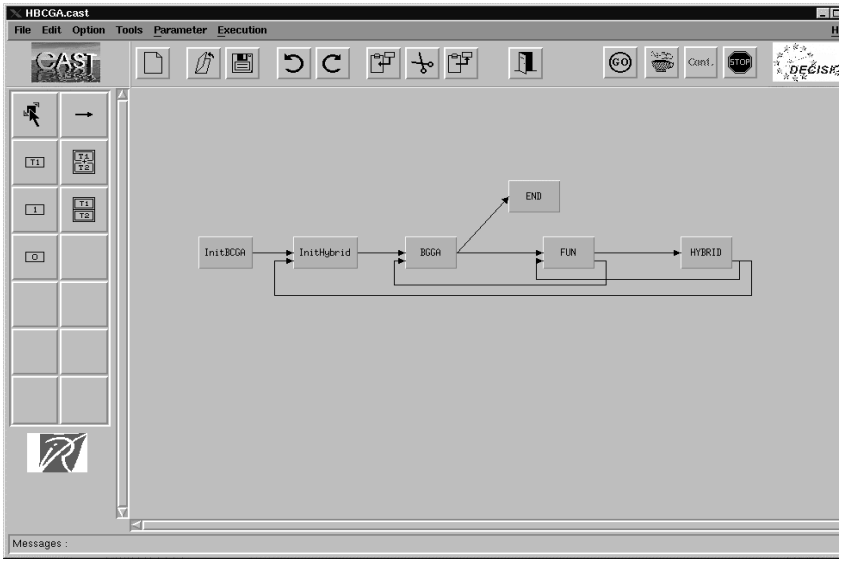

Fig. 1. *CAST: the user inerface*

The integration platform provides a high-level icon-based interface to the users which can invoke simulation, verification, configuration, deployment and execution tools. The result of the above HBCGA example is depicted by Figure 2.

Hardware configurations are uniformly supported, i.e., workstations, PC-clusters and computers connected through NOW, LAN, WAN are transparently supported through the definition of execution locations for the user modules. This is done by specifiying the CORBA name servers for the various modules. Currently, no dynamic migration of modules is supported by CAST. The end-user has no choice concerning the execution location of the modules. Execution of the modules are invoked by dynamic request to the ORB: the name server includes the name of the methods implementing the required codes in the wrapper object definitions for any particular module.

2.2 Specification of complex applications

The benefit gained from using process algebras to define the user applications is twofold:

- it constrains the application developers to specify rigorously the modules interactions and scheduling, without programming first the modules inputs/outputs, thus leaving implementations considerations to a later phase in the application design
- it produces formal specifications which can be processed by automatic verification and reduction algorithms, thus ensuring further computational properties, e.g., fairness, deadlocks freeness, accessibility of the modules, etc.

Although the last point is not yet implemented, it is foreseen that the introduction of verification and reduction algorithms will permit a significant gain in the integration of complex distributed applications involving a large number of modules. This provides a sound and theoretical background on which to rely in order to eventually formally specify, prove and verify the application processes. For example, electronic chips co-simulation involves dozens of modules which cooperate to simulate the various aspects of electronics: electrical, thermal, clocks, etc. Managing and controlling the interactions of these modules may require intricate relationships. Proving that that these interactions are properly implemented and that the simplest simulator has indeed been designed can be a cumbersome task that may benefit from model validation and verification tools inherited from formal verification techniques. This is common use in electronic chip design, and can be straightforwardly implemented in CAST.

2.3 Implementation

A demonstrator was developed during a first six months phase (April-October 1998).This preliminary Corba-compliant integration platform was demonstrated during the second DECISION project review in Sophia-Antipolis, October 1998. It relied on the *ILU* object request broker (ORB) from Xerox [4] and the *OLAN* distributed application configuration environment [1].
ILU has interesting features, including interfaces with a variety of programming languages, e.g., C, C++, Lisp, Java, Python. But inherent limitations (e.g., no interface repositories, no multi-threading) and performance penalties lead however to the replacement of these software.

The replacement by the *Orbacus* ORB [10] from Object-Oriented Concepts, Inc., a fully Corba-compliant ORB, started in November 1998. *Orbacus* permits a seamless integration with the C++ and Java languages. This made possible the fast implementation of the Corba version of *CAST* on Unix workstations.

The prototype was demonstrated in April 1999. A contributing factor was the C++ language chosen for the implementation of *CAST*. Based on these grounds, the upgrade to an extensive Corba-compliant integration platform, named *CAST 2.0*, was therefore straightforward.

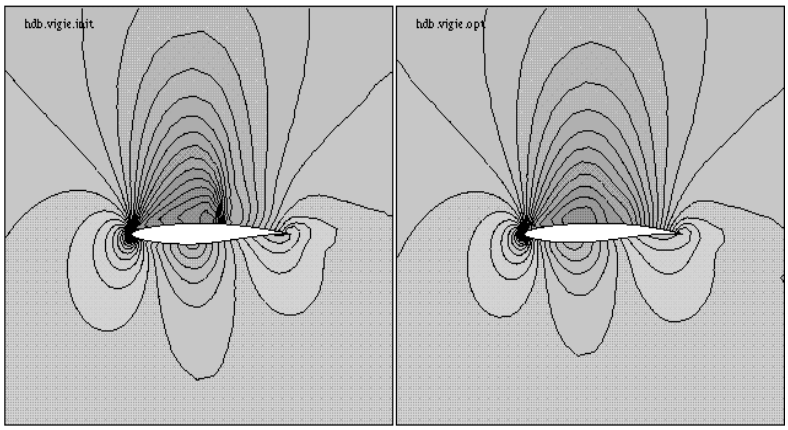

Fig. 2. *Wing shape optimisation: induced drag reduction in high-speed cruise configuration*

Extensions to the platform have since been implemented to support the dynamic plug-in of solvers and optimisers to the platform. This uses the full dynamic creation and invocation capabilties of the CORBA object-request broker.
It is clear that the configuration, deployment and execution of the optimisation applications remain unchanged for the end-users and the project managers in both cases.

2.4 Integration of parallel codes

A later development phase started in January 2000. Its goal was to allow the integration of parallel codes and execution on remote PC-clusters over wide-area networks. By the end of year 2000, this was operational concerning the execution of parallel codes on PC-clusters.
Therefore users can now define distributed simulation and optimisation applications that involve sequential and parallel codes that cooperate. The codes run on parallel computers, NOW and PC-clusters that are distributed over wide area networks.
Interfacing existing codes that involve MPI statements with CORBA was a challenge. Various work has already been carried out concerning the integration of parallel codes in distributed computing environments and was of invaluable help for this matter. Our implementation uses the notion of " parallel CORBA objects " developed in PACO by the *PARIS* project at IRISA [11]. Basically, a parallel CORBA object is a collection of similar objects running in parallel on a PC-cluster for example. They are managed

by a particular server object which interfaces with the remote clients. The parallelism is therefore transparent to the end user. This provides for the seamless integration of parallel codes, written in Fortran using MPI for example, up to the interface of the code with the parallel CORBA objects. PACO is itself built on the MICO object-request broker [7]. MICO has therefore become the new ORB layer used by *CAST* since September 2000.
The 2D wing shape optimisation example (Figure 2) presented in Section 2 was run on a PC-cluster using a parallel version of the genetic algorithm written in Fortran, using MPI statements. It was later made compliant with CORBA using PACO. It was run as a set of servers and a *CAST* client on the network. In the last implementation, there are several servers, one which is dedicated to the genetic optimisation algorithm, and the others which are dedicated to the CFD and mesh calculations. Performance is of course in par with the parallel execution of the code on multiple processors (Table 1).

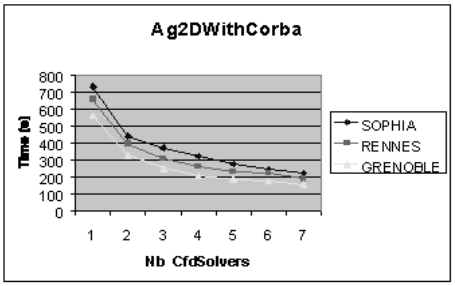

Table 1. Performance data for 2D wing shape optimisation on PC-clusters.

2.5 Metacomputing

The next step, which started in January 2001, involved the execution of simulation applications concerning aerodynamic optimisation of airfoils in a metacomputing environment, including workstations and PC-clusters distributed over several locations at INRIA Rennes, INRIA Sophia-Antipolis near Nice and INRIA Rhône-Alpes in Grenoble all connected by a high-speed gigabits/sec network (Figure 5).
Based on performance tests conducted on various deployment configurations (Table 2 and Table 3), the *CAST* software runs on the PC-cluster in Grenoble, the parallel genetic optimisation algorithm runs on the PC-cluster in Sophia-Antipolis, and several instances of the CFD solver run on the PC-cluster in Rennes. CAST is a CORBA client for the genetic algorithm server, which acts in turn as a client for the CFD solvers (Figure 3).
The PC-cluster in Grenoble includes 225 Bi-Pentium III 733 MHz processors connected by a 100 Mbits/sec FastEthernet network, running Linux (Mandrake 7.0). The PC-cluster in Rennes includes 20 bi-Pentium III, 500 MHz, and bi-Pentium II, 450 MHz., connected by a FastEthernet 100 Mbits/sec network, running Linux

(Debian 2.2). The PC-cluster in Sophia-Antipolis includes 33 bi-Pentium III processors (19 running at 933 MHz and 14 at 500 MHz), running Linux 2.2.

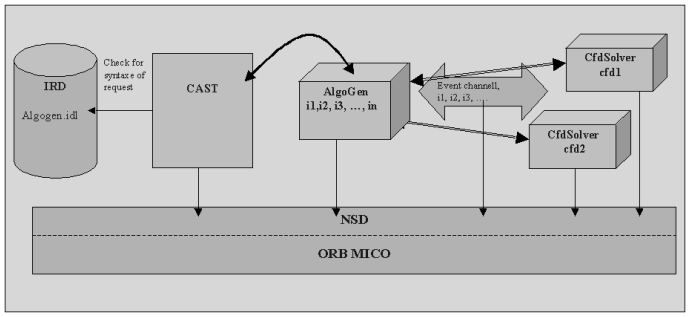

Fig. 3. Clients and servers for the optimisation application.

Several instances of the CFD solver run in parallel to account for the candidate solutions generated by the genetic algorithm. Each CFD solver is allocated several

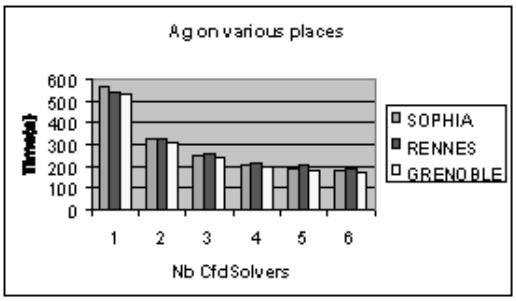

Table 2. Performance data for the execution of the airfoil example.

processors to account for the domain decomposition of the area around the airfoil, which is statically divided into four sub-domains (Figure 4). One processor is allocated to each of the sub-domains. The airflow around the wing shape is computed each time the airfoil is modified by the optimisation algorithm, i.e., for each optimisation loop. Each optimisation loop generates several candidate solutions, which are then evaluated in parallel, i.e., the mesh and the flow is computed for each optimisation loop and for each solution generated by the optimisation algorithm.

In this example, the CFD solvers are the most time-consuming tasks. It is therefore of most importance to duplicate and parallelize the mesh generation and computation of the airflow around the wing shape. This is why there are several instances of the solver executing in parallel for each candidate solution, in fact four of them for one wing shape instance. There are in turn several candidate solutions that are produced by the genetic optimisation algorithm that are evaluated in parallel. The processor allocation is done statically: a requested number of processors is queried on each PC-cluster

prior to the execution of the application. It is then allocated to the specific tasks to execute. Various performance tests have been conducted for this particular application. Results are given in Table 2 for various execution locations of the genetic algorithm (Ag), when the CAST software and the CFD solvers are located in Grenoble, depending on the number of CFD solvers running. Table 3 shows the results when the latter are running in Sophia-Antipolis. The PC-clusters are connected by the high-speed gigabits/sec VTHD network [13] (Figure 5).

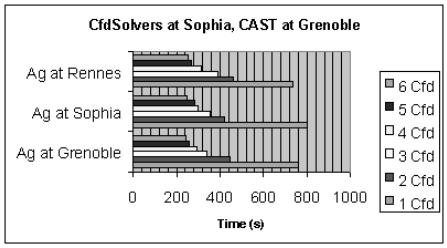

Table 3. Performance data for the execution of the airfoil example.

3. Conclusion

This paper presents a software integration platform called *CAST* that combines both techniques of formal specifications and distributed object-oriented development, for the specification, configuration, deployment and execution of complex engineering applications in a unified and operational way.

The platform was developed on Unix workstations and communicate through a commercial CORBA object request broker. It was successfully tested, showing blazing performance, against complex optimisation applications in a distributed environment involving several remotely located PC-clusters at INRIA, connected to a gigabits/sec high-speed network called VTHD.

The target user population includes project managers and engineers that are expert in their particular domain, e.g., aerodynamics, electromagnetics.

The application tasks and their interactions are transparently mapped to distributed software components. They are implemented using object-oriented programming techniques. The actual user modules may include simultaneously non object-oriented code, e.g., Fortran subroutines, as well fully object-oriented software, e.g., C++. In this case, *CAST* takes automatically into account the diversity of the interfaces and parameters for the various modules.

The *CAST* integration platform also offers a high-level graphics interface for the specification of complex multi-discipline applications, thus making the formal algebraic and theoretical background on which it is based, as well as the technicalities of distributed computing, totally transparent to the end-users.

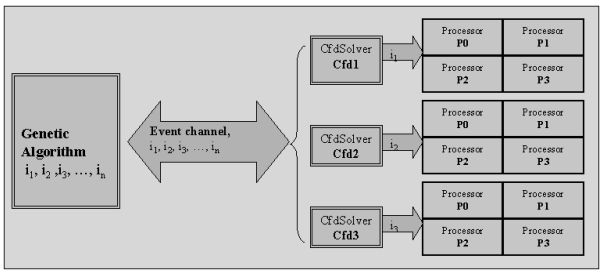

Fig. 4. *Interaction between the genetic algorithm and the CFD solvers.*

Also, an interesting feature in *CAST* is that it supports the coupling of CORBA and non-CORBA modules simultaneously. This provides for a smooth transition from existing application software and mathematical libraries to state-of-the-art integration platforms and metacomputing environments. It is also expected that this will permit the connection to third party environments that support widely accepted communication standards and exchange protocols, e.g., CORBA.

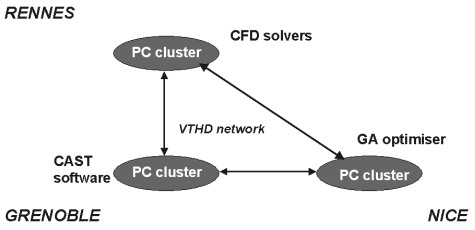

Fig. 5. The optimisation application in the metacomputing environment.

Although metacomputing has focused much attention and efforts in the last few years, large scale applications, including CPU and graphics intensive applications that require large amounts of computing ressources are still demanding easy to use high-performance problem-solving environments [2]. Examples of such applications include the co-simulation of electronic chips in the *CADNET* project, numerical propulsion systems simulation [5] and multidiscipline aircraft simulation, as for the Digital Dynamic Aircraft approach in the *AMAZE* project. Indeed, the design environments required now by aircraft manufacturers will include CFD, CSM, acoustics and electromagnetics simulation and optimisation, ranging from the initial CAD outlines up to the dynamic behavior of the airframe and systems under operating constraints. This will put an unprecedented load on hardware and software resources in problem-solving environments.

Still lacking specific features, e.g., load balancing, guaranteed QoS and security, the *CAST* integration platform provides the functionalities that make it a good candidate for high-performance metacomputing environments dedicated to multi-discipline simulation applications.

The software documentation, including the software specifications and the *CAST* user's manual, is available on-line at: http://www.inrialpes.fr/sinus/cast. Further information on the project is available at: http://www.inrialpes.fr/sinus.

Acknowledgments

The authors wish to thank Alain Dervieux and Jean-Antoine Desideri from the *SINUS* project at INRIA Sophia-Antipolis, as well as Jacques Periaux from Dassault-Aviation (Pôle Scientifique), for their strong support and helpful advice.
Part of this project is funded by INRIA through ARC " Couplage " and by the French RNRT program, through the *VTHD* project. It was also partly supported by the EEC *Esprit* Program, through the HPCN " *DECISION* " project: " *Integrated Optimisation Strategies for Increased Engineering Design Complexity* ".

References

[1] Balter, R., et al. Architecturing and configuring distributed applications with OLAN. Proc. IFIP Int'l. Conf. Distributed Systems platforms. Middleware '98. Lake District. September 1998.
[2] Barnard, S., et al. Large-scale distributed computational fluid dynamics on the Information Power Grid using Globus. Proc. NASA HPCC/CAS Workshop. NASA Ames Research Center. February 2000.
[3] J. Blachon, T. Nguyen " DECISION prototype integration platform: CAST CORBA prototype ". Soft-IT Workshop. Invited lecture. " Next generation of interoperable simulation environments based on CORBA ", INRIA-ONERA-Simulog. June 1999.
[4] XEROX ILU Reference manual. 1998. ftp://ftp.parc.xerox.com/pub/ilu/2.0a14/manual_html
[5] Lopez, I.,et al. NPSS on NASA's IPG: using Corba and Globus to coordinate multidisciplinary aeroscience applications. Proc. NASA HPCC/CAS Workshop. NASA Ames Research Center. February 2000.
[6] Marco N., Lanteri S. A two-level parallelization strategy for genetic algorithms applied to shape optimum design. Research Report no. 3463. INRIA. July 1998.
[7] http://www.mico.org
[8] Milner R. Calculi for synchrony and asynchrony. Theoretical computer science. Vol. 25, no. 3. July 1983.
[9] Nguyen G.T., Plumejeaud C. Integration of multidiscipline applications in metacomputing environments. Systèmes, Réseaux et Calculateurs Parallèles. Hermès Ed. To appear 2002.
[10] Object-Oriented Concepts, Inc. ORBacus for C++ and Java. Version 3.1.1. Object-Oriented Concepts, Inc. 1998.
[11] Th. Priol. Projet PARIS " Programmation des systèmes parallèles et distribués pour le simulation numérique distribuée ". INRIA. May 1999.
[12] Sang, J., Kim, C., Lopez, I. "Developing Corba-based distributed scientific applications from legacy Fortran programs" Proc. NASA HPCC/CAS Workshop. NASA Ames Research Center. February 2000.
[13] http://www.telecom.gouv.fr/rnrt/projets/pres_d74_ap99.htm