Project no. FP6-004265

CoreGRID

European Research Network on Foundations, Software Infrastructures and Applications for large-scale distributed GRID and Peer-to-Peer Technologies

Network of Excellence

GRID-based Systems for solving complex problems

D.PM.01 – Roadmap version 1 on Programming model

Due date of deliverable: February 28, 2005
Actual submission date: 15 April 2005

Start date of project: 1 September 2004 Duration: 48 months

Organisation name of lead contractor for this deliverable: UNIPI (35)

Revision: draft

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	PU

Keyword List: programming model, components, grid, high performance, scalability

Table of content

1 Summary

This document is the first version of the roadmap of the CoreGRID Virtual Institute on programming models (Workpackage 3, WP3). This version of the roadmap is being delivered as D.PM.01 in February 2005. The roadmap sketched here will be expanded and more precisely defined in the second version D.PM.03 which is due at M18, that is in February 2006. In particular, the phases of the roadmap covering the second part of the project are only sketched in this document: these will be fleshed out and formalized in D.PM.03.

The document is structured as follows: Section 2 outlines the context of the Programming Model Virtual Insitute within CoreGRID, along with its main goals, driving forces and its major work framework organization. Section 3 positions the activities of the Virtual Institute in the context of the current projects being undertaken by the partners and introduces an outline state-of-the-art survey relating to the main topics covered by the Virtual Institute activities. Section 4.3 presents the main features of the Virtual Institute roadmap. Section 5 summarizes the core concerns shared with the other Virtual Institutes in the CoreGRID NoE, that is, it describes the expected results, relationships, exchange of information and experiences to be achieved in this Virtual Institute in conjunction with the other Virtual Institutes of the NoE. In section 6 the groups in the partner institutions participating in the Virtual Institute activities are briefly described.

2 Introduction

The Virtual Institute on programming models aims to deliver a definition of a component programming model that can be usefully exploited to design, implement and run high performance, efficient Grid applications. The same component model can also be exploited in the design of tools supporting Grid programming, such as in the development of PSEs or in the development of tools supporting resource management or system architecture related activities.

Possibly, the component programming model should be a *lightweight* model, to be better supported on a variety of architectures and middleware tools. In other words, the model should be specified in abstract and minimal specification should be given, in such a way that existing component models can be adapted–extended–integrated to fit this new model. The advantage of such choice is twofold: on the one hand, only those features that are really fundamental to grid programming will be studied and included in the new model. On the other hand, giving a minimal specification, covering anyway the features that are assessed as the features *essential* to make the model suitable to address grid programming, allows the Virtual institute partners, and the other CoreGRID partners as well, to make the minimal effort to adapt his existing component models to the Virtual institute framework.

It is assumed that the component based programming model's main aim is to address the new characteristic challenges of Grid computing - viz. heterogeneity and dynamicity - in terms of programmability, interoperability, code reuse and efficiency. Grid programmability, in particular, represents the biggest challenge. Grid programs cannot be constructed using traditional programming models and tools (such as those based on explicit message passing or on remote procedure call/web service abstraction, for instance), unless the programmer is prepared to pay a high price in terms of programming, program debugging and program tuning efforts.

The Virtual Institute activities are mainly concerned with the coordination of the partner activities and the exploitation of the results achieved by the Virtual Institute partners in the framework of the research programmes and projects they are involved

in. The Virtual Institute does not finance any kind of research activity other than the activities aimed at exchanging knowledge, setting up research projects and exploiting common results among partners. From this perspective, the "distillation" of a lightweight, component programming model such as the one sketched above will result from an assessment of the (many) different model proposals put forward by the Virtual Institute partners in the context of the projects they are separately involved in, and agreeing on the use of these results to build a commonly agreed component model.

2.1 Context

The CoreGRID Network of Excellence (NoE) aims at strengthening and advancing scientific and technological excellence in the area of Grid and Peer-to-Peer technologies. To achieve this objective, the Network brings together forty-two institutions who have constructed an ambitious joint programme of activities, structured around six complementary research areas that have been selected on the basis of their strategic importance, their research challenges and the recognised European expertise to underpin the development of the next generation of Grid middleware. The Virtual Institute on programming model will address one of these six research areas, namely an investigation of the definition of a programming model for the Grid with the aim of reducing the complexity of Grid programming.

In particular, a software component model for Grids will be investigated within the Virtual Institute at a higher level of abstraction than is currently practiced in models based on traditional message-passing primitives. New Grid programming models are required that rely on a higher level of abstraction and are based on component technology. The main objective is to define a novel common European component model, suitable for future large-scale Grid and P2P computing.

The Programming model Virtual Institute activities sport a number of inks and close relationships with the activities of the other Virtual Institutes in the NoE. These links and relationships will be described in detail in Section 5 later on in this document. In particular, the Programming model Virtual institute will provide inputs to other Virtual institutes, mainly concerning component based programming models and techniques, whereas other Virtual institutes will provide to Programming model Virtual institute those inputs necessary to come to the definition of a component model really addressing all the issues typical of grid programming.

The partners involved in the programming model Virtual Institute activities are also involved in a number of distinct research projects that are related to the activities of the Virtual Institute, as well as on more general Grid-related research topics that are of interest to the whole CoreGRID NoE. Among the National/European initiatives involving partners of the Virtual Institute, we mention:

UK e-Science Programme Basic research on Grid technology: middleware, security, tools and applications for a range of topics including bioinformatics, media, finance and data mining. QUB is involved in Design and implementation of architectures and models. IC is involved in this project as well (2002–2007)

Expected synergies among this project and Programming model Virtual institute include both application requirement definition and component based programming environment design.

Grid Ireland Grid interoperability across domains. QUB is involved in Application construction and performance measurement. (2004 - 2006)

EC FP5/FP6 Basic and applied research on different Grid related topics. Many of the partners involved in the Virtual Institute are also involved in one or more projects of these two programs.

In the framework of EC FP5/FP6 initiatives, partners of the Virtual Institute are also involved in several different projects, including:

GridCoord SSA to support coordination in Grid research within EU (2004 - 2006). UNIPI is actually leading the project, QUB is involved in Collaboration amongst the individual researchers; the creation of European excellence and competence centres; A visionary research agenda. INRIA is involved in Education material on Grid, i.e. gathering and development support. More relevant for CoreGRID, INRIA is also involved in strengthening activities for Grid researchers through the organisation of several workshops. In particular, a workshop (October 2005) entitled "The use of open source middleware for Grids" has the objective to identify convergence with other middleware communities. An other one, entitled "Really large-scale Grid architecture" (September 2005) is also very relevant for this CoreGRID WP activities.

The experience acquired by partners participating in the GridCoord SSA will be exploited within the Virtual institute activities in two different ways: on the one hand, it will be used to identify the synergies with the other EC funded projects that are worth to be exploited in the Programming model Virtual institure activities. On the other hand, experience in the GridCoord SSA will be exploited at the Virtual institute management level, of course.

NextGrid Secure and economically viable business models for Grid computing. QUB is involved in Making the Grid more scalable and usable. ISTI/CNR is also involved in the same project (2004 - 2007)

The experience developed by partners in the NextGrid project will be used within the Programming model Virtual institute to better understand and take into account the architectural constrains and perspectives that the Virtual institute should take into account while discussing and designing the grid component model. On the other hand, the grid component model developed in the context of the Programming model Virtual institute activities, will probably be suitable to be exploited in the context of hte NextGrid activities.

GridLab is an EU project involving VUA. The goal of the project related to grid programming models is the development of simple APIs for grid resources and services, implemented within the Grid Application Toolkit (GAT), providing language bindings to various languages like C, C++, Java, Python, and Perl. (2002-2005)

GridLab expertise will be used within the Virtual institute to consider features of assessed grid architectures as the testbed targeted by the Virtual institute grid component model. Also, Virtual institute results will possibily be used in the context of GridLab, to drive the development of the GridLab tools, hopefully.

Partners of the Virtual Institute are also involved in the activities of other big projects, related to the activities of the Virtual Institute but not directly spawn from the initiatives and projects mentioned above:

French ACI-GRID INRIA is involved in the **GRID5000** national project. The aim of Grid5000 is to set up an experimental Grid infrastructure, with approximatively 5000 CPUs. The infrastructure is still at the stage of deployment and configuration, but should be fully available in the first quarter of 2005. Grid5000 is an opportunity to experiment novative programming model for large-scale Grid applications.
INRIA/IRISA is also involved in **HydroGrid** is a 3 year ACI GRID project. It is a multidisciplinary project that aims at modeling and simulating fluid and solute transport in a subsurface media using a multiphysic approach. The INRIA/IRISA contribution was to provide PaCO++/GridCCM as a programming model to couple parallel codes (MPI & OpenMP).

In the context of Programming model Virtual institute, we assume to exploit the knowledge derived during the development of HydroGrid software. In particular, experience gained while developing such complex applications will be used to enhance the features of the grid component model representing the target of this Virtual institute.

GRID.it an Italian project involving major Italian research centers and universities and covering several aspects related to Grid deployment, programming and utilization for complex, multidisciplinary applications. Within GRID.it, UNIPI is responsible for

the Programming environment workpackge (development of a prototype, component based, high performance Grid programming environment). (2003–2005)

From the GRID.it project we expect to derive consistent experience in the development of component based grid programming models and tools, being the development of a component based programming environment one of the major goals of this project.

Eventually, partners of the Virtual Institute are also involved in several different, possibly smaller projects, including:

Co-algorithms and GRID Computing (QUB/European Social Fund studentship) The development of a formal framework for a restricted class of GRID applications: component communication mechanisms; high performance requirements. QUB is involved in Design and analysis of a Grab-and-Go communication harness; evaluation of its effectiveness in component-based GRID numerical software. (2002 - 2005)

Fractal The OASIS joint team between CNRS I3S, INRIA Sophia-Antipolis and University of Nice-Sophia Antipolis is involved in a 2 years partnership with France-Telecom R&D, specifically aimed to further extend the Fractal model so as it better suits to Grid component programming (2004–2006)

COFFEE is a project involving WWU Muenster, funded by the German Reserach Foundation. The focus of the project is on developing semantically sound, efficient framework for organizing collective communication in parallel and distributed systems, in particular in Grids. A Java-based Grid programming systems was developed that integrates predefined algorithmic components with efficient means of communication relized on top of RMI.

COA is an INRIA ARC project starting in February 2005. The goal of this project is to conceive an experimental software platform for both dynamic adaptation and steering of components. It also investigates how aspect weaving can be used to achieve those two features.

VL-e (virtual laboratories for e-science) Involving VUA. The goals of this project include the development of Grid programming environments with suitable programming models in the context of the Java-based Ibis environment. (2004–2008)

SFIDA UNIPI is involved in this 30 month national project (co-funded by the Italian Government), aiming at developing a Grid-based inter-operability platform able to support next generation Supply Chain Management applications specifically addressing the needs of SMEs belonging to industrial districts and dynamic supply networks (2005-2007)

Layered Components UOW is working on a project called: A Hierarchical and Reconfigurable Layered Component Model for a Generic Grid Platform

The objective is to have a layered structure of the components system by differentiating system and application components. This work also aims at identifying the properties that should be verified by the non-functional aspects. In addition, it is intended to provide a theoretical foundation on which a generic Grid platform can rely.

AUTOGRID Temporal Modeling of Intelligent Grids

UOW is also starting this project, aiming at developing the theoretical foundations for a multi-layer self-organizing generic Grid architecture, which will automatically manage reconfiguration of components in Grid systems in a safe and optimal way.

GridSAM GridSAM is an open-source job submission and monitoring web service funded by the Open Middleware Infrastructure Institute and involving IC. This project is funded by the UK Open Middleware Infrastructure Institute (OMII) managed programme. The aim of GridSAM is to provide a Web Service for submitting and monitoring jobs managed by a variety of Distributed Resource Managers (DRM). The modular design allows third-party to provide submission and file-transfer plug-ins to GridSAM. Moreover the job management API used by the GridSAM web service can be embedded into Grid application that requires job submission and monitoring capabilities. (2004-2005)

RealityGrid The RealityGrid is an EPSRC funded pilot project to examine how the condensed matter, materials and biological sciences communities can make more effective use of distributed scientific computing and visualisation resources within their future research activities. Existing terascale computing environments are able to generate data at a rate several orders of magnitude beyond our ability to extract the knowledge produced during the simulation. Simulations take several days to run but the data analysis still takes months! The goal of the RealityGrid project is to generate a new computational science analysis pipeline that "moves the bottleneck out of the hardware and back into the human mind." IC is involved in this project (2002 – 2005)

Exchange programs several partners of the Virtual Institute are involved in bilateral exchange programs concerning research topics that have relationship with the Virtual Institute actiivites: DAAD/ARC is an exchange program involving UNI-PASSAU and the group of Paul Kelly, Imperial College. Main topic is dynamic program optimization (7/2004-6/2006, renewable by one year), DAAD/PROCOPE is an exchange grant involving UNI-PASSAU and the group of Albert Cohen, INRIA FUTURS, Paris. Main topic is metaprogramming with Meta-OCaml (1/2004-12/2005, renewable by one year), BFHZ/CCUFB is an exchange grant involving UNI-PASSAU and the group of Paul Feautrier, ENS Lyon (1/2004-12/2004), the main topicwere issues of loop parallelization: This is actually finished, but the program overlapped with initial CoreGRID activities, VIGONI, is an echange program involving WWU MUENSTER and UNIPI. Main topic was skeleton based Grid programming environments and components (2004-2005) This is actually finished, but the program overlapped with initial CoreGRID activities

Most of these research programmes have direct connections and synergies with the topics covered by the Programming model Virtual institute. Involvement of the Virtual institute partners in these (and possibly, in other) projects guarantees the expected research framework substaining the durable integration activity proper of CoreGRID, and, as a consequence, of the Programming model Virtual institute activities.

2.2 Problems and challenges

The GRID poses new challenges in terms of programmability, interoperability, code reuse and efficiency. These challenges mainly arise from the features that are peculiar to GRID, namely heterogeneity and dynamicity. GRID programmability, in particular, represents a big challenge. GRID programs cannot be written using normal programming models and tools, unless the programmer is prepared to pay a high price in terms of programming, program debugging and program tuning efforts.

New programming models are required that exploit a layered approach to GRID programming which will offer user friendly ways of developing efficient, high performance applications. This is particularly true in cases where the applications are complex and multidisciplinary. Within CoreGRID, the challenge is to design a component based programming model that overcomes the major problems arising when programming GRIDs.

The challenge, per se, requires that a full set of sub challenges will be addressed:

- A suitable programming model (that is user friendly and efficient) to program individual components is needed
- Component definition, usage and composition must be organised according to standards that allow interoperability to be achieved.
- Component composition must be defined precisely in such a way that complex, multidisciplinary applications can be constructed by the composition

of building block components, possibly obtained by suitably wrapping existing code. Component composition must support and, in addition, guarantee scalability.

- Semantics must be defined, precisely modelling both the single component semantics and the semantics of composition, in such a way that provably correct transformation/improvement techniques can be developed.

- Performance/cost models must be defined, to allow the development of tools for reasoning about components and component composition programs

All of these sub-challenges must be dealt with taking into account that improvements in hardware and software technology require new GRID systems to be transparent, easy to use and to program, person centric rather that middleware, software or system-centric, easy to configure and manage, scalable, and suitable to be used in pervasive and ubiquitous contexts.

2.3 Objectives

The research program of this Virtual Institute is organized in three related tracks:

1. Basic programming models aiming at defining programming models and tools suitable for programming the single components that eventually will constitute the component based Grid applications

2. Components and hierarchical composition aimed at defining a component model suitable for Grid programming and allowing components to be composed to get new components, either sequential or parallel

3. Advanced programming models aiming at defining higher level models that allow programmers to use components and component compositions in more efficient and user friendly ways.

All these tracks must address problems related to semantics and performance prediction, i.e they must devise activities aiming at defining the suitable semantic tools, as well as performance models to reason about component based programs, at all levels (at the level of the programming model of the single component, of the component framework as well as at the level of advanced programming models built on top of the base component model).

Overall, outcomes of the three related tracks will be combined to achieve the principal goal of this Institute: to agree on a common component model suitable for use in the area of Grid programming. A key subgoal of this goal is the definition of a hierarchical composition model that allows new components to be derived from existing ones. Provision of such a facility is fundamental both to the development of pre-defined components modeling common Grid aplication parallelism exploitation patterns and to supply users with mechanisms incrementally to abstract the level of control specified in application code. This is a crucial consideration. Currently available and agreed component models do not support structural component composition. Consequently, considering hierarchical component composition as a key mechanism in the design/derivation of new components from given components will ensure that the component model proposed and evaluated in this Virtual Institute will help place the European Grid community at the forefront of the field.

The organization of Virtual institute activities using these three related tracks has been assessed among the Virtual institute partners as the most effective way to address the problems enumerated at the beginning of Section 3.1. In particular, by dividing the activities in activities related to the basic programming model, to

component and component compositions and to advanced programming models, the problems such as programming language/platform indipendence, support of native code, support of advanced programming techniques (skeleton, aspect oriented), interoperability, etc. can all be dealt with at the more appropriate abstraction level.

2.4 Tasks

All the activities of the Virtual Institute are organized within three separate tasks based upon the three objectives described in Section 2.3, namely:

Task 3.1 *Basic Programming Models* models and tools to support the programming of single components that will constitute component based Grid applications.

The activities of task 3.1 comprise two further strands: 3.1.1 the programming model identifying the primitives and their properties required to implement a single component; and 3.1.2 identifying suitable communication mechanisms to support both intra- and inter-component interactions.

EIA-FR has responsibility for Task 3.1 (Pierre Kuonen).

Task 3.2 *Components and Hierarchical Composition* aimed at defining a component model suitable for Grid programming, and allowing components to be composed, either sequentially or in parallel, to construct new components

The activities of task 3.2 comprise two further tracks: 3.2.1 primitive component definition, defining the basic features of a component, in particular those related to the Grid; and 3.2.2 hierarchical composition for defining how components can be hierarchically composed.

INRIA has responsibility for Task 3.2 (Denis Caromel).

Task 3.3 *Advanced programming models* aimed at defining higher level models that permit programmers to use components and component compositions in more efficient and user friendly ways.

The activities of task 3.3 are not divided further. They are aimed at investigating advanced programming models to be built on top of the component programming model defined by task 3.2 thereby providing higher level programming abstractions for programmers and/or more comprehensive optimization and transformation techniques.

UNIPI has responsibility for Task 3.3 (Marco Danelutto).

2.5 Drivers

The programming of sequential computers has benefitted from the start (in the 1940s) from the existence of a commonly accepted programming model, the von Neumann model. Computer architectures have deviated substantially from this model in recent decades, but language implementations have succeeded to this day in maintaining the von Neumann view for the programmer.

There has never been a commonly accepted programming model for parallel and distributed computing – although there are approaches aspiring to it, like MPI for message passing parallelism, OpenMP for shared-memory parallelism, and remote method invocation (RMI) for distributed object-oriented programs.

It would help the evolution of Grid software substantially, if there were a commonly held abstract view of Grid programming. And, since the Grid is a restricted

form of distributed architecture, there is the hope that such a model might be developed.

A component model will be a central aspect of a model for Grid programming. The driving influences on such a model are:

- *Application domains for the Grid:* Partners will represent or identify different application domains and contribute their views of the Grid to the common model.
- *Models already in existence:* Some partners have already developed and are using programming models at different levels of abstraction. In a negotiation process, features of these models can enter into the common Grid model.
- *Need of Flexibility:* The common model must represent a view of the Grid which is shared by all applications and which can be specialized according to the needs of each individual application.
- *Theoretical foundations:* The common model should have a clean structure and sound properties, which support a modular and robust method of programming diverse Grid applications.

3 Positioning

Many CoreGRID participants have defined their own particular approaches to component models for the Grid [1] including concrete implementations.

The approaches described below originate from the members of CoreGRID listed below (abbreviations are used in the table later):

- Institut National de Recherche en Informatique et en Automatique (INRIA),
- Imperial College (IC),
- Queen's University Belfast (QUB),
- Westfälische Wilhelms-Universität Münster (WWU),
- University of Chile (UCHILE),
- University of Passau (UNIPASSAU),
- University of Pisa (UNIPI),
- Technical University of Catalunya(UPC) and
- Vrije Universiteit Amsterdam (VUA).

3.1 State of the art (existing approaches)

The current component approaches focus on different requirements that should be met by an agreed uniform solution. To obtain a complete picture of the approaches, we have selected the following requirements to distinguish approaches:

1. Interoperability
2. Platform independence
3. Programming language independence

[1] some of which are based on the CCA- or CCM-model as outlined below

4. Support for native code
5. Support for aspect-oriented techniques
6. Support for skeletal programming and mobile code
7. Application specific high-performance libraries or utilities
8. Elements constructed upon web services/SOAP
9. Integration with standard middleware (UNICORE or Globus)

These requirements have been discussed and agreed among the partners of the Virtual Institute. Most of them can be found in many grid reference works and reports. Some of the requirements are more specific of the component perspective adopted in the Programming model Virtual Institute, indeed. The capabilities of the various approaches taken by the CoreGRID members in relation to these requirements are summarized in the following table:

feature	1	2	3	4	5	6	7	8	9
ProActive		yes		ongoing	yes	ongoing	yes	yes	yes
FRACTAL	yes	yes	yes	yes	yes	yes	yes	yes	yes
ICENI		yes		yes					yes
REFLEX			yes		yes				
QUB		yes	yes	yes		yes			
HOC-SA	yes	yes	yes	yes		yes		yes	yes
Polytope		yes	yes	yes		yes	yes		yes
ASSIST	yes	yes	yes	yes		yes	yes	yes	yes
MALLBA				yes		yes	yes		
GAT	yes	yes	yes	yes				yes	yes
GridCCM		yes	yes	yes			yes		yes
Ibis		yes		yes			yes		via GAT

In summary, the ongoing projects' most characteristic features are as follows:

- *INRIA*, OASIS team at Sophia-Antipolis have developed *ProActive* [5], a Java implementation of an environment that features a hierarchical architecture for components that extends simple port definitions (like in CCA) by detailed specifications of component nesting and compositions. The hierachical component model that is implemented is Fractal. An important characteristic of this implementation, using ProActive, is that components are distributed, possibly running on several JVMs simultaneously. A Fractal-ProActive component is thus a powerful abstraction of distributed activities on the Grid, which could thus simplify their programming, their configuration, and their deployment.

- INRIA, SARDES team at Grenoble, conjointly with France Telecom R&D designs an abstract component model, named Fractal [6]. Fractal proposes a hierarchical, dynamic and extensible component oriented programming approach which could prove well suited to Grid programming. As ProActive, Fractal is a project hosted by ObjectWeb, the consortium for Open Source Middleware.

  INRIA, PARIS team in Rennes, is currently developping a framework for designing parallel adaptable components (AFPAC framework). The objective

of this framework is to help developpers of applications that should run on Grid to take into account the changes that may occur in such environments during application runs [7].

The same PARIS team has also developed a parallel extension to the CORBA component model, named GRIDCCM [17]. GridCCM enables a simple and efficient embedding of SPMD code into a parallel CORBA component. Thus, parallel communication flows and data redistribution are possible during an operation invocation on such a parallel CORBA component.

- *IC* has developed *ICENI* [12], a Java Grid Middleware system, based upon a Service-Oriented Architecture and a Component Programming Model. The ICENI component framework captures information relating to the application, its structure and inter-component data and control flow. The model provides a clear separation between meaning, behaviour and implementation of the component, which allows for both communication and implementation selection at run-time, while providing the user with a flow-based programming model.

- UCHILE is developing REFLEX [20], a portable reflective system for Java that evolved to a versatile kernel for aspect-oriented programming [19] (AOP). The objective is to leverage the interest of AOP, possibly using multiple domain-specific aspect languages, to distributed components on the Grid. Aspect languages can be used to address specific concerns, such as multithreading, communication modes, etc."

- Current work at QUB relating to the development of a component model has two strands which address heterogeneity and dynamicity in a Grid environment.

  – Transformations of component specifications
  Automatic transformation of component specifications, expressed in a functional style, to efficient architecture-specific implementations. This approach can contribute to implementation of components on heterogeneous Grid resources by supporting the generation, from a common source, of multiple implementations [10].

  – Composition of Components
  Experience of combining components of highly restricted type using a Grab and Go communication model. This approach is suitable for creating certain types of Grid applications software and addresses some aspects of dynamicity ([13], [14]).

- *WWU* has developed and implemented *Higher-Order Components (HOCs) [9].* HOCs are program components that are offered to the application programmer as partially implemented services, which can be customized for a particular application by plugging in mobile code units. This approach combines the advantages of service-orientation with those of customizable components such as skeletons, paradigms, etc.
  The most significant features of the HOC-approach include

  – The definition of a mechanism for handling code mobilty in a WSRF compliant manner

  – The Globus Toolkit-based reference implementation of a runtime environment, the HOC-Service Architecture (HOC-SA),

  – Adherence to analytical performance models and performance prediction in the course of application development

  - Integration with multiple database managment systems for reuse of data and code
  - Potential for semantics-preserving transformations on programs built of HOCs
  - A user friendly HTTP-based portal application
  - A novel mechanism for component customization, so-called *behavior customization*, which allows the specification of how a HOC can handle the data-dependencies in particular applications.

  The WWU group has also developed a Java-based Grid programming system [4] providing application programmers with predefined algorithmic skeletons, which can be viewed as components encapsulating typical algorithmic patterns of parallelism. Skeletons are implemented on high-performance Grid servers in an architecture-specific manner, thus providing potential for portable performance across different, heterogeneous platforms. Several skeletons of the same kind running on different servers can be combined to form a single, distributed skeleton implementation, with the client monitoring execution on different servers and implementing appropriate load-balancing strategies. When a complex application is designed, several different skeletons can be combined together using a graphical workflow description language based on coloured Petri-Nets.

- The component work carried out at *UNIPASSAU* focuses on performance, especially on the run-time optimization of component composition by restructuring and exploiting domain-specific, high-level algebraic properties of the components. One source for these high-level algebraic properties is the *Polytope* model [15], which has been studied intensively at UNIPASSAU, and which is currently being extended to cope with the dynamicity and heterogeneity of the Grid.

- Research at *UNIPI* includes the development of the programming environments P3L, SkIE, Lithium [8] and *ASSIST* [2] all of which are based on the algorithmic skeleton concept and focus on structured parallel programming. The latest of these is ASSIST that evolved in the FIRB research project GRID.it, which involves a number of research institutions in Italy. ASSIST is an high performance, component based, structured parallel programming environment targeting workstation clusters/networks and Grids. ASSIST provides programmers with high level programming patterns. Using these patterns, the programmers simply structure their parallel application, then the ASSIST compiler tools and run time system provide to execute the application on the Grid architecture and to adapt the application to the Grid configuration at hand. In particular, the ASSIST tools take care of adapting the application execution to target Grid architecture features, according to some kind of abstract performance contract provided by the user/programmer. In addition, ASSIST provides full interoperability with existing CCM and Web Services frameworks, in that ASSIST components can be wrapped as CCM or WS and CCM components and WS services can be invoked from within the ASSIST components.

- *UPC* has developed *MALLBA* [1], a C++ library for solving combinatorial optimization problems in parallel using components spread across wide-area networks or local clusters. Theoretical research includes the study of computing with approximate data, global synchronization, correctness and program transformation.

- The VUA has developed Ibis [22], a Java programming environment for parallel applications, that provides efficient communication mechanisms, like an efficient RMI, group communication (GMI) as well as replicated objects (RepMI) and a system for Grid-aware divide-and-conquer parallelism (Satin). Further programming models, like message passing (MPJ), are currently under development. The Ibis runtime system supports efficient communication in Grid environments by using parallel TCP streams and communication through firewalls. Support for encryption and compression are in development. The VUA group has also implemented a Java version of GridLab's Grid Application Toolkit [3] (GAT), that enables dynamic access to Grid resources via a variety of Grid middleware platforms (Globus 2, 3, 4, Unicore, ssh, etc.). Ibis can use the Java GAT for integration with Grid middleware platforms.

Currently, the starting point of many research projects and experimental implementations of component architectures are two commonly known concepts:
the *Common Component Architecture (CCA)* [11] and the *CORBA Component Model (CCM)* [16].

- *CCA* has been defined by a group of researchers from laboratories and academic institutions committed to defining standard component architectures for high performance computing. The basic definition of a component in CCA states that a component "is a software object, meant to interact with other components, encapsulating certain functionality or a set of functionalities. A component has a clearly defined interface and conforms to a prescribed behavior common to all components within an architecture. Multiple components may be composed to build other components." Currently the CCA Forum maintains a web-site gathering documents, projects and other CCA-related work (www.cca-forum.org) including the definition of a CCA-specific format of component interfaces (Babel/SIDL) and framework implementations (Ccaffeine)

- *CCM* is a component model defined by the Object Management Group (OMG), an open membership for-profit consortium that produces and maintains computer industry specifications (e.g. CORBA, UML, XMI, ...). The CCM specifications include a Component Implementation Definition Language (CIDL); the semantics of the CORBA Components Model (CCM); a Component Implementation Framework (CIF), which defines the programming model for constructing component implementations and a container programming model.

Concluding from the features available in the current implementations of CCM, CCA and other component models for the grid, most current projects exchanging data, executable code or both across network boundaries using a portable format. Thus, the underlying technologies XML, Java and Web Services should be supported by a future grid component model.

4 Vision, Strategy, Roadmap

4.1 Vision and Scenarios (end-users, technologies, computer science)

The Virtual Institute on Programming Model aims at coordinating and enforcing Virtual Institute partner activities on the topics related to the development of suitable, high performance Grid programming environments based on the component concept.These topics include the component model itself, the programming

methodologies and techniques used/exploited within a single component, as well as any advanced programming model developed on top of the component model and providing higher level programming abstractions. The target end users for the Virtual Institute activities are the Grid application programmers. In particular, those programmers that nowadays must know in detail all the features of common Grid middleware in order to be able to develop effective parallel Grid applications. In the perspective suggested by the NGG Expert group documents[18, 21], the Virtual Institute activities will be focused on providing these programmers suitable mechanisms and programming environments that implement the invisible Grid concept, while supporting all the good features that are typical of component based programming models. However, system programmers, that is programmers developing high-level mechanisms and policies needed to implement efficient Grid tools, are also considered target end users for Virtual Institute activities. As an example, programmers developing Grid PSEs will obviously benefit of the high level component programming model that represents the Virtual Institute's main goal.Virtual Institute on Programming Model plans to build on the existing Grid and programming model technology. In particular, the GCM (Grid Component Model) will be developed starting from known results in the component programming models community, possibly addressing and solving those problems that are still considered open problems and taking always into account that the goal is to provide a component model suitable for efficient Grid programming. Therefore, all the existing Grid technology features and results will be taken into account and exploited as needed. In particular, the results achieved during the development of the CORBA Component Model [16] will be considered, as well as the results available from CCA research programs [11] and all the results coming from the Virtual Institute partners component related activities, such as those projects cited in the State of the art Section. Overall, the Virtual Institute envisions a scenario were Grid programmers can exploit the GCM to build efficient, scalable and performant Grid applications just exploiting their application domain specific knowledge, rather than domain specific *and* Grid middleware/tool knowledge, and exploiting the compositionality, modularity and interoperability features proper of component models.

4.2 Strategy

Overall, the members of the Programming Model Virtual Institute agreed that the goal of defining a joint *Grid Component Model* (GCM) could be best realised by undertaking the three sub-tasks originally proposed and outlined in the Section 2.3, that is: investigating component model design, single component programming techniques and higher level programming models built on top of the component model.

Our general approach/strategy will be organized as follows. First, we will work on an *abstract view* of the Grid Component Model. Such a high-level view should allow all the partners to define a joint view of what should be in a component model for the Grid. This abstract view will most likely be at the level of defining what a primitive component is, what a hierarchical composition is, and what the various kinds of ports and interfaces are.

Second, the Virtual Institute members recognized the need for defining a *concrete view* of the Grid Component Model, i.e. a technical specification of the GCM. Such specification is essential as it is the required standard agreed upon that will permit the composition of Grid codes coming from various groups, the interoperability of those components, and the sharing of common component tools.

Eventually, each Virtual Institute partner will exploit the GCM within its own research activities, possibily including activities in other EU funded projects. Also, we will consider the chance to use the GCM to start new projects, aimed at imple-

menting GCM reference implementations, at experimenting GCM features and so on.

Along all the path, both practical and more theoretical aspects will be taken into account. As an example, the first task of section 2.3 will both deal with the practical aspects related to the programming models needed to program the single component and, at the same time, with the theoretical foundations needed to support those practical aspects. In the same way, the task related to the GCM specification is supposed to come to the definition of GCM *and* to the definition of all the related theoretical aspects, such as the semantics of the component and component composition, for instance. This will be done, of course, in the perspective of the lightweight idea of component model discussed in Section 2.

More technically, the following architecture has been proposed for concretely defining the GCM:

1. XML Component Specification as a Schema
2. Run-Time API defined in several languages
3. Packaging XML as a schema

The first part, using a schema to precisely define a component description language, a kind of ADL (Architecture Description Language) in XML, is the basic element to be able to define inter-operable component descriptions. The second aspect, run-time API allows to manipulate components at execution in a uniform manner. Finally, the packaging schema authorizes the development of common deployment tools. Details on the elements comprising each part of the specification are given below.

1. XML Component Specification as a Schema:

The first element in a component specification is the notion of *primitive components*. One must be able to define, from a given piece of code, the attributes of the component being constructed. A piece of standard code, or a module, is promoted to the status of a component. Then, provided we are targeting a hierarchical model to tackle the complexity and large scale nature of Grids, one must be able to compose the primitives to build *hierarchical* entities.

Of course, the specification of components, both primitive and composite, calls for defining interfaces (various kind of ports), and the binding between those ports.

With respect to dealing with language interoperability, a key aspect of the proposed specification is to rely on external references (Java Interface, C++ .h, Corba IDL, WSDL) to specify the nature of ports. As such, one can define components specific to a given platform, one can also specify a component that is exported in several interfaces (e.g. Java and C#), or even exported with portable ports such as a WSDL definitions.

Finally, the Grid aspects are covered with the specification of specific elements such as distribution, virtual Nodes, QoS, etc.

Figure 1 summarizes the structure and the key aspects of the component specification. An important feature of such a specification is the extensible nature of an XML Schema.

2. Run-Time API defined in several languages:

We propose to define a common run-time API for manipulating components at execution. The basic functions of the API will be related to:

- Life cycle managment,

- Definition of Primitive Components
- Definition of Composite Components (composition)
- Definition of Interfaces (ports)
  - Server, Client, event, stream, etc.
- Including external references to various specifications:
  - Java Interface, C++ .h, Corba IDL, WSDL, ?
- Specification of Grid aspects:
  - Parallel, Distribution, Virtual Nodes,
  - Performance Needs, QoS, etc.

Figure 1: XML Component Specification

- Introspection
- Basic Control (Monitoring, Reconfiguration, ...)
- Optimization
- etc.

The languages and formats in which we would like to define related APIs for manipulation of Grid components at runtime include Java, C++, C#, Fortran, etc.

Upon executing components, the API will facilitate the portability and interoperability within a given language. Standard implementations of component infrastructures and containers will be possible. We believe it would not be realistic to attempt a direct multi-language infrastructures, with inter-language interoperability. However, one can even imagine WSDL specification of part of the run-time API. It would allow implementing and deploying Web Services that provide portable run-time manipulation of components (management of life cycle, etc.). In any case, for the sake of efficiency, and expressive power, language specific implementations are needed.

3. Packaging XML as a schema

Together with the component specification defined in chapter 1, there is also a need to provide more practical information about Grid components. For instance, one must specify the dependencies between codes, where to find the appropriate bundles, for what hardware platform, etc. Here is an incomplete list of such information:

- Requirements on the hardware platform
- Location of the code needed to instantiate the component on a platform
- Dependencies between versions
- etc.

With such information, the components may be deployed in various contexts. Defined as an XML schema, this extensible specification will provide for generic tools to help solve a complex problem at hand: large scale deployment on the Grid.

4.3 Roadmap

4.3.1 Phases of the roadmap

In the roadmap, we can logically recognize three phases:

Initiation During the initiation phase, the Virtual Institute activities are devoted to collecting the current ideas being persued by the partners, possibly arising from the different "local" projects in which the partners are involved. In this phase information and requirements will be gathered from the other Virtual Institutes in the NoE.

Component model definition During the assessment phase all the partners will agree on the common features and functionalities of the component based Grid programming model. Some partners may already be engaged in projects which will result in preliminary reference implementations. All the partners will interact with the other NoE Virtual Institutes to propagate the concepts assessed in the component model definition.

Assessment All of the partners will agree on the component model, start using it in their own projects, and promote it outside the Virtual Institute and throughout Europe.

The initiation phase will last roughly until the end of the first year of the project. The Component model definition phase will last for the following two years and the Assessment phase will start at year three and continue far beyond the NoE project end.

During all the three phases, the mechanisms described below will be used to coordinate the Virtual Institute activities.

Virtual institute activities in the context of the roadmap discussed in this document will be checked against a set of milistones. Some of the milestones were included in the original DoW. In paritcular, in that document two milestones were assumed:

1. M.PM.01: Coordinated research framework infrastructure established and workpackage web site up &running (M3)
2. M.PM.02: Guidelines for the definition of a lightweight, composable, component based programming model for GRIDs. (M12)

Such milestones were the ones relative to the first 18 months JPA. During the preparation of the JPA for the next period, other milestones will be precisely defined that can be used to check whether the Virtual institute activities adhering or not to the roadmap. At the moment, further milestones can be assumed, including:

- M.PM.03: Definition of the GCM (about M24-M36)
- M.PM.04: Guidelines for the definition of advanced programming models on top of the GCM (about M24-M36)

4.3.2 Mechanisms

The chief mechanisms for ensuring close and profitable cooperation among the partners will be workshops, meetings and facilities for the exchange of research papers, project reports, software, etc. In particular these will be organized as follows:

Meetings It is envisaged that, for each major deliverable, a workshop will be held two months prior to its due date to discuss its content, and an E-meeting one month prior to the due date to finalise content.

Workshops
1. Workshop at M12 to consider proposals for a Grid Component Model (GCM).
2. Workshop at M22 to consider White Paper on GCM
3. Workshop at M34 to consider full proposal for GCM
4. Workshop at M46 to consider assessment of GCM

E-meetings
1. Consensus meeting at M14 to finalise proposals for a GCM.
2. Consensus meeting at M23 to finalise White Paper on GCM
3. Consensus meeting at M35 to finalise full proposal for GCM
4. Consensus meeting at M47 to finalise assessment of GCM

Further workshops and E-meetings may be scheduled during the Virtual Institute activities to allow partners to discuss particular topics or simply to report about the activities performed within the Virtual Institute.

Documents Working documents will be shared via the collaborative tool BSCW. Completed documents (and other documents relevant to the Workpackage activity) will be available on the Workpackage web site.

Software Software (e.g. reference implementation components) will be shared via a tool coordinated by Workpackage 9.

New projects As the component model evolves partners will seek opportunities, individually and in collaboration, to obtain support from funding bodies (national and international) for projects which include a direct contribution to the activities of the Workpackage.

Within current and new projects every opportunity will be taken to address implementation issues in the light of the emergent GCM.

4.3.3 Future steps

On completion of the project, work on the further development and use of the GCM will be facilitated by the following mechanisms:

- Agreement of the Workpackage partners to use the GCM as a basis for the future development of Grid related software. For example, to build a reference implementation and evaluate its effectiveness in diverse application areas.
- An annual workshop where progress on the continuing use of the model is reported.
- Continued commitment to use of the GCM should act as an impetus for joint applications for support under international bilateral funding arrangements such as UK/Italy (British Council).

4.3.4 Virtual Institute roadmap in the NGG perspective

Among the three perspectives presenting the visions for NGGs [18, 21] (end-user, architectural, software) we are more directly concerned with the *Software vision* one, aiming at considering the grid as being a programmable system, even if our work may also have impacts on the *Architectural* and *End-user* visions.

WP3's roadmap focusing on component orientation clearly contributes to *Programming Grids through abstraction* thanks to a virtualization of grid resources. Indeed, component orientation is a strong structuring mean through which resource and other sorts of requirements can be expressed as a declarative, external property of a component (i.e. apart from its functional part). The way such requirements

are implemented can be expressed through component controllers (using meta-level programming techniques) or as non-functional properties offered by component containers. Overall, the various tasks of the WP3 will collectively contribute to the establishment of a *programming model combining parallel and distributed programming practices in a coherent way.*

NGG's most recent vision is that of an *"invisible Grid", offering key features for a service-oriented knowledge utility, a new paradigm for software and service.* The fact that component instances and the functional services they offer may be published, discovered, etc, (for instance, building on top of Web Services technologies) dovetails with this vision of the NGG as a service-oriented utility. More precisely, even if Web Services support is considered as a means to allow interoperability, this does not mean that Web Services per se are a suitable programming model and language for grid applications and environments. Indeed, Web Services are a mechanism, and so not sufficiently abstract to adopt as a model for grid programming.

Instead, a common grid component model (GCM) is proposed. GCM will directly contribute to enable *Interoperability as a basic means for problem solving.* In this strategy, it is critical to define a standard for components metadata representation. Additionally, properties of special interest in the context of grid environments (including foundations and middleware), such as *self-management, self-adaptation, self-healing, self-reconfiguring, flexibilility* are much more manageable and tractable if the adopted component model offers a well-defined **hierarchical composition** model rather than a flat assembly. For instance, self-reconfiguration of a hierarchical component may imply changing instances of inner components, to modify bindings, etc., but this can be achieved without clients being aware of it if the component is a hierarchical one.

5 Link with other CoreGRID scientific workpackages

Each track of this Virtual Institute requires and will put in place mechanisms to ensure cooperation with other research activities planned within the other Virtual Institutes of the Network. The component model, as a whole, will be used in the Problem solving Environments, tools and Grid Systems Virtual Institute (WP7). There will be close liaison with those working in WP7 activities with a view first to acquiring a reasonable set of requirements for the component model and then to delivering a component model suitable for exploitation in the design and implementation of PSEs. The System Architecture Virtual Institute (WP4) will give guidance on the kind of lower layers that can be used to implement the component framework. The Knowledge & Data Management Virtual Institute (WP2) will provide suitable mechanisms for handling all the knowledge base features needed to underpin the component framework. In addition, WP2 might use the proposed component model in the design and implementation of the higher-level support programs and applications related to data and knowledge management.

To achieve the goals of the three tasks of this Virtual Institute, full advantage will be taken of the products created within the other Virtual Institutes of this Joint Plan of Activities. Results from the Knowledge & Data Management Virtual Institute (WP2) will be exploited when addressing the problems associated with the initialization and maintenance of the component framework on the Grid. The System Architecture Virtual Institute (WP4) results will be exploited during implementation of the component framework. Results from the Information and Monitoring (WP5) and Resources Management and Scheduling (WP6) Virtual Institutes will be exploited when addressing problems related to the design and implementation of

component based, advanced programming models for the Grid, i.e. in Task 3.3 of this Virtual Institute. Throughout the duration of CoreGRID, this Virtual Institute will provide results (essentially, the common component model) that can be used effectively in other Virtual Institutes of this JPA in the development of component based applications, run times, support tools on the Grid. The two way exchange of results between this Virtual Institute and the other Virtual Institutes of the JPA is a direct result of the integration facilitated by the NoE and substantially exceeds the degree of cooperation that could be expected without CoreGRID, where the partners merely pursue their local goals without specific recourse to the important results generated by the rich diversity of related European projects.

6 Participants

The table 1 presents a summary of the degree and range of involvement of participants in the Programming Model Virtual Institute. Columns headed Task 3.1, Task 3.2 and Task 3.3 indicate their declared involvement in the tasks described in Section 2.4.

Partner	No.	Nation	Task 3.1	Task 3.2	Task 3.3
ISTI CNR	4	I	yes	yes	
FHG	8	D		yes	
IC	12	UK	yes	yes	
INRIA	14	F	yes	yes	yes
QUB	21	IRL			yes
WWU Muenster	22	D	yes		yes
UCAM	24	UK		yes	
UCHILE	26	CHI		yes	yes
UNCL	33	UK	yes		
UNIPASSAU	34	D	yes	yes	yes
UNIPI	35	I	yes	yes	yes
EIA-FR	36	CH	yes	yes	
UOW	37	UK	yes	yes	
UPC	38	E			yes
VUA	39	NL	yes		yes
VTT	40	FIN	yes		

Table 1: Partners involved in the Virtual Institute activities.

7 Appendix A

In this appendix the support material to the roadmap document is collected, namely:

- Table 2 collects the names of the CoreGRID partners involved in the Virtual institute activities along with the acronyms used to name such partners in the CoreGRID official documentation and in this roadmap document as well
- Table 3 hosts a few non technical acronyms used across the document, along with their full names.

Acronym	**Full Name**	**Nation**
ISTI/CNR	Istituto di Scienze e Tecnologia dell'Informazione, Consiglio Nazionale delle Ricerche	Italy
IC	Imperial College	UK
INRIA	Institut National de Recherce en Informatique et en Automatique	France
QUB	The Queen's University of Belfast	UK
WWU Muenster	University of Muenster	Germany
UCAM	University of Cambridge	UK
UCHILE	University of Chile	Chili
UNCL	University of Newcastle Upon Tyne	UK
UNIPASSAU	University of Passau	Germany
UNIPI	University of Pisa	Italy
EIA-FR	Ecole d'ingenieurs et d'architectes de Fribourg	Switzerland
UOW	University of Westminster	UK
UPC	Technical University of Catalonia	Spain
VUA	Vrije Universiteit	Netherlands
VTT	Technical Research Centre of Finland	Finland

Figure 2: Partners of the WP3: Acronym, full name

Acronym	**Full name**
BSCW	Basic Support for Cooperative Work, the cooperative web site tool used in the network
DoW	Description of Work, the technical Annex I, presented to EC
FPx	Framework Programme X
EC	European Community
GCM	Grid Component Model
NoE	Network of Excellence
SSA	Special Support Action
WP	work package
NGG	Next Generation Grid (expert group)

Figure 3: Acronyms used in the document (excluding the technical ones)

References

[1] E. Alba, F. Almeida, M. Blesa, M. Diaz C. Cotta, I. Dorta, J. abarro, J. Gonzalez, C. Leon, L. Moreno, J. Petit, J. Roda, A. Rojas, and F. Xhafa. Mallba: A library of skeletons for combinatorial optimisation. In *Proceedings of the Euro-Par, Paderborn (GE), LNCS 2400*. Springer-Verlag, 2002.

[2] M. Aldinucci, M. Coppola, M. Danelutto, M. Vanneschi, and C. Zoccolo. Assist as a research framework for high-performance grid programming environments. In Jose C. Cunha and Omer F. Rana, editors, *Grid Computing: Software environments and Tools*. Springer-Verlag, 2004.

[3] Gabrielle Allen, Kelly Davis, Tom Goodale, Andrei Hutanu, Hartmut Kaiser, Thilo Kielmann, Andre Merzky, Rob van Nieuwpoort, Alexander Reinefeld, Florian Schintke, Thorsten Schütt, Ed Seidel, and Brygg Ullmer. The grid application toolkit: Towards generic and easy application programming interfaces for the grid. *Proceedings of the IEEE*, 93(3):534–550, 2005.

[4] Martin Alt and Sergei Gorlatch. Using Skeletons in a Java-based Grid system. In Harald Kosch, László Böszörményi, and Hermann Hellwagner, editors, *Euro-Par 2003*, volume 2790 of *Lecture Notes in Computer Science*, pages 682–693. Springer-Verlag, August 2003.

[5] Francoise Baude, Denis Caromel, and Matthieu Morel. From distributed objects to hierarchical grid components. In *International Symposium on Distributed Objects and Applications (DOA), Catania, Sicily, Italy, 3-7 November*, Springer Verlag, 2003. Lecture Notes in Computer Science, LNCS.

[6] E. Bruneton, T. Coupaye, and J. Stefani. Recursive and dynamic software composition with sharing. Proceedings of the 7th ECOOP International Workshop on Component-Oriented Programming (WCOP'02), June 2002.

[7] J. Buisson, F. André, and J.-L. Pazat. Dynamic adaptation for grid computing. In *European Grid Conference 2005*, Amsterdam, February 2005.

[8] Marco Danelutto and Paolo Teti. Lithium: A structured parallel programming enviroment in java. In *Proceedings of Computational Science - ICCS 2002, LNCS No. 2330*, pages pp. 844–853. Springer-Verlag, 2002.

[9] Jan Dünnweber and Sergei Gorlatch. HOC-SA: A Grid Service architecture for Higher-Order Components. In *IEEE International Conference on Services Computing, Shanghai, China*. IEEE Computer Society Press, September 2004.

[10] Stephen Fitzpatrick, Maurice Clint, Terence J. Harmer, and Peter Kilpatrick. The tailoring of abstract functional specifications of numerical algorithms for sparse data structures through automated program derivation and transformation. *Comput. J.*, 39(2):145–168, 1996.

[11] CCA forum. The Common Component Architecture (CCA) Forum home page, 2005. http://www.cca-forum.org/.

[12] N. Furmento, W. Lee, A. Mayer, S. Newhouse, and J. Darlington. A parallel corba component model for numerical code coupling. In *International Journal of High Performance Computing Applications, Vol. 17, No. 4, 417-429*. SAGE Publications, 2003.

[13] Joaquim Gabarró, Alan Stewart, and Maurice Clint. Grab and Go system: a CPO approach to concurrent web and grid-based computation. *Electronic Notes in Theoretical Computer Science*, 66(3), 2002.

[14] Joaquim Gabarró, Alan Stewart, Maurice Clint, Eamonn Boyle, and Isabel Vallejo. Computational models for Web- and Grid-based computation. In Harald Kosch, László Böszörményi, and Hermann Hellwagner, editors, *Euro-Par 2003*, volume 2790 of *Lecture Notes in Computer Science*, pages 640–650. Springer-Verlag, August 2003.

[15] Christian Lengauer. Loop parallelization in the polytope model. In *CONCUR '93: Proceedings of the 4th International Conference on Concurrency Theory*, pages 398–416. Springer-Verlag, 1993.

[16] omg.org team. CORBA Component Model, V3.0. http://www.omg.org/technology/documents/formal/components.htm, 2005.

[17] Christian Perez, Thierry Priol, and Andre Ribes. A parallel corba component model for numerical code coupling. In *International Journal of High Performance Computing Applications, Vol. 17, No. 4, 417-429*. SAGE Publications, 2003.

[18] D. Snelling and T. Priol *et al.* Next Generation Grid(s) European Grid Research 2005 - 2010 Expert Group Report, June 2003. http://hpc.isti.cnr.it/ lafo/domenico/talks/siena2003/ngg_eg_final.pdf.

[19] Éric Tanter and Jacques Noyé. Versatile kernels for aspect-oriented programming. Research Report RR-5275, INRIA, July 2004.

[20] Eric Tanter, Jacques Noye, Denis Caromel, and Pierre Cointe. Partial behavioral reflection: spatial and temporal selection of reification. In *OOPSLA '03: Proceedings of the 18th annual ACM SIGPLAN conference on Object-oriented programing, systems, languages, and applications*, pages 27–46. ACM Press, 2003.

[21] D. Snelling *et al.* Next Generation Grids 2 Requirements and Options for European Grids Research 2005-2010 and Beyond, 2004. Available at http://www.semanticgrid.org/docs/ngg2_eg_final.pdf.

[22] Rob V. van Nieuwpoort, Jason Maassen, Gosia Wrzesinska, Rutger Hofman, Ceriel Jacobs, Thilo Kielmann, and Henri E. Bal. Ibis: a flexible and efficient java-based grid programming environment. *Concurrency & Computation: Practice & Experience*, 17(7–8):1079–1107, 2005.