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.03 – Roadmap version 2 on Programming model

Due date of deliverable: February 28, 2006
Actual submission date: April 21, 2006

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

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

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

Contents

1 Executive Summary

This document describes the roadmap of the Programming model Institute in the CoreGRID network of excellence. It is, in fact, the second, improved version of the roadmap. The first version was delivered as D.PM.01 in February 2005. This version incorporates a number of distinct changes, with respect to the first roadmap document:

- changes needed to keep track of evolving research in the research topics covered by the Programming model Institute. For example, some of the projects originally listed in Sections 2.2 and 3 have changed or may have ended. To acknowledge this, the citations have been dealt with, either removed or updated.
- changes inserted to take into account comments from reviewers or CoreGRID Scientific and Industrial Advisory Boards. For example, the security section 5 is new.

The second version of the Programming model Institute roadmap, however, incorporates most of the material included in the first version of the document (D.PM.01) together with the new and/or improved sections. This is to ensure that the document is self-contained, thus avoiding the necessity to retain the two versions of the roadmap for completeness.

This document is organized as follows: Section 2 introduces the main Institute goals. A new subsection briefly outlines the Institute's achievements in the first year. Section 2.2 describes the context of the Institute activities. In particular, the projects related to the Institute research topics that involve Institute partners are listed, the problems, challenges, objectives and drivers are outlined and the (sub)tasks planned for organizing the Institute activities are described, and the partners participating in each of the tasks listed. Section 3 outlines the positioning of the Institute activities with respect to other European and Worldwide projects concerning component frameworks. Both Sections 2.2 and 3 have been slightly updated with respect to D.PM.01. Section 4.3 actually contains the technical roadmap of the Programming model Institute. Material has been added to take into account new NGG documents (NGG3, delivered at the beginning of 2006), security issues (this is Section 5) and relationships with industry (as recommended by CoreGRID Scientific and Industrial advisory boards and by the project referees during the first CoreGRID review meeting) and an outline of the research groups established within the Programming model Institute. Section 7, slightly modified with respect to the previous document, describes the relationships and links with the other Network Institutes. Finally, Section 8 contains a table of the participants in the Institute activities and a few (new) subsections setting out the expected contribution of the individual partners to the Institute roadmap.

This is the "last" document describing the roadmap of the Programming model Institute. There will be no other deliverable refining it or modifying it. Despite being somewhat generic in parts, we have tried to define in sufficient detail to indicate how the Programming model Institute will try to achieve the goals stated in the first DoW.

2 Introduction

The 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.

The component programming model should be a *lightweight* model, to be better supported on a variety of architecture and middleware tools. In other words, the model should be specified abstractly with minimal specification being given, so that existing component models can be adapted–extended–integrated to fit the new model. The advantage of this choice is twofold: on the one hand, only those features that are fundamental to grid programming will be studied and included in the new model; on the other hand, giving a minimal specification, covering just those features that are deemed to be the features *essential* to make the model adequate for grid programming, allows the Institute partners, as well as other CoreGRID partners, to expend minimal effort in adapting existing component models to the Institute framework.

The component based programming model's main aim is to address the novel characteristic challenges of Grid computing - viz. heterogeneity and dynamicity - in terms of programmability, interoperability, code reuse and efficiency. Grid programmability, in particular, presents the biggest challenge. Grid programs cannot be constructed using only 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 effort.

The Institute's activities are concerned mainly with the coordination of the partner activities and the exploitation of the results achieved by the Institute partners within the frameworks of the research programmes and projects they are involved in. The Institute does not finance any kind of research activity other than the ones 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 emerge from an assessment of the (many) different model proposals put forward by the Institute partners in the context of the projects they are separately involved in, allowing a commonly agreed component model to be built.

2.1 First year summary

During the first year of work by the CoreGRID Institute on Programming model, a number of activities related to topics discussed in the first version of the Institute's roadmap document have already been addressed. In particular, the following aspects of the work can be assessed as a result:

- all of the partners of the Institute agreed on the roadmap as described in the D.PM.01 deliverable of February 2005
- during the Barcelona Institute plenary meeting, in June 2005, all of the partners presented their roadmap related activities, thus facilitating effective integration on the roadmap topics
- at the same meeting, all of the partners agreed on three important points:
  - they decided to adopt Fractal, the component model developed at INRIA and France Telecom, as the reference model of the Institute. It will be

used as a basis for discussing new features to be introduced in the grid component model developed by the Institute. This decision does not necessarily imply that the grid component model developed during the activities coordinated by the Institute will be an extension of Fractal.

  - they decided to focus the Institute's activities on a set of *research topics* with a finer grain with respect to the original Institute organization in three tasks described in the DoW
  - they agreed to include in the more focused research topics a specific *security* research topic, that was not explicitly included in the DoW.

- during the summer of 2005, the partners agreed on the contents of the JPA relative to M12 to M30. The JPA2 was prepared in accordance with the JPA
- over the same period, the partners discussed the features to be included in the grid component model, which is the subject of the research activities of the Institute. Indeed, a deliverable was discussed and produced by the end of October 2005 (which is logically a second year activity) that hosts a first rough idea of the main features that will be included in the grid component model.

Overall, the partners of the Institute exploited most of the mechanisms of CoreGRID to support the coordination of their research activities relating to the topics and goals stated in the first roadmap document. Beside participating in the general meeting events (in January, in Pisa; in June, in Barcelona; and in October, in Sophia Antipolis) the partners initiated an intense and promising "short visit" program. In addition, one of the positions of the fellowship program involved two of the Institute partners and directly addressed the Institute topics, while two other fellowships addressed topics involving a group outside the Institute and another CoreGRID Institute.

2.2 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 that have constructed an ambitious joint programme of activities, structured around six complementary research areas selected on the basis of their strategic importance, their research challenges and recognized European expertise to underpin the development of the next generation of Grid middleware. The Institute on programming model will address one of these six research areas, namely, an investigation leading to 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 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 high abstraction and that 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 Institute work enjoys a number of links and close relationships with the activities of the other Institutes in the NoE. These will be described in detail later in Section 6. In particular, the Programming model Institute will provide inputs to other Institutes, mainly relating to component based programming models and techniques, whereas other Institutes will provide to the

Programming model Institute those requirements that are essential for the definition of a component model which really addresses all of the issues associated with grid programming.

The partners involved in the programming model Institute activities are also involved in a number of distinct research projects that are related to the activities of the Institute and 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 Institute, are:

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 also involved in this project (2002–2007)

Expected synergies between this project and the Programming model 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 a number of Grid related topics. Many of the partners involved in the Institute are also involved in one or more projects in these two programs.

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

GridCoord SSA to support coordination in Grid research within EU (2004 - 2006). UNIPI is leading the project; QUB is involved in Collaboration amongst the individual researchers; the creation of European excellence and competence centers. 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 organization of workshops. In particular, a workshop (October 2005) entitled "The use of open source middleware for Grids" was devoted to identifying convergence with other middleware communities. Another workshop, entitled "Really large-scale Grid architecture" (September 2005) is also relevant in the context of this CoreGRID WP's activities.

The experience acquired by partners participating in the GridCoord SSA will be exploited within the 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 can exploit the results of the Programming model Institute's activities; on the other hand, experience in the GridCoord SSA will be exploited at the Institute management level.

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 this project (2004 - 2007)

The experience acquired by partners in the NextGrid project will be used within the Programming model Institute to better understand and take into account the architectural constraints and features that the Institute should address when discussing and designing the grid component model. The grid component model developed in the context of the Programming model Institute activities will probably be suitable for exploitation in the context of the NextGrid activities.

GridCOMP Grid programming with components: an advanced component platform for an effective invisible grid (2006-2008). Actually, this is a STREP that represents a spin-off of CoreGRID, in that CoreGRID Programming model Institute partners designed and presented the project and are involved in the project (INRIA, UNIPI, UOW, ISTI/CNR), and the overall project goal is the design of a programming framework suitable for grid programming based on the component model developed in the Programming model Institute.

XtreemeOS Building and Promoting a Linux-based Operating System to Support Virtual Organizations for Next Generation Grids (2006-2008). Several partners of the Programming model Institute are involved in the project: INRIA, ISTI/CNR, VUA, aimed at designing and prototyping a network centric operating system suitable to support next generation grids on top of Linux.

Partners of the Institute are also involved, or have been recently involved, in the activities of other large projects, related to the activities of the Institute but not directly arising 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 approximately 5000 CPUs. The infrastructure has been made available during 2005. Grid5000 offers an opportunity to experiment with an innovative programming model for large-scale Grid applications.

INRIA/IRISA is also involved in **HydroGrid**, a 3 year ACI GRID project. This is a multidisciplinary project aiming to model and simulate fluid and solute transport in a subsurface medium using a multiphysic approach. The INRIA/IRISA contribution is to provide PaCO++/GridCCM as a programming model to couple parallel codes (MPI & OpenMP).

In the context of Programming model Institute, we intend 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 to be developed by this 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 has been responsible for the Programming environment work package (development of a prototype, component based, high performance Grid programming environment), 2003–2005.

From the GRID.it project we expect to inherit significant experience in the development of component-based grid programming models and tools, the development of a component based programming environment being one of the major results of the project.

In addition, partners of the 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 and in the 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 year partnership with France-Telecom R&D, specifically aimed at further extending the Fractal model so that it is more suitable for Grid component programming (2004–2006)

COFFEE is a project involving WWU Münster, funded by the German Research 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 system has been developed that integrates predefined algorithmic components with efficient means of communication, realized on top of RMI.

CompSpread is a project at UNIPASSAU funded by the German Research Foundation. It commenced in October 2005, and its duration is two years with a possible renewal for two further years. Its charter is to adapt sophisticated methods of loop parallelization for supercomputing, based on the polytope model, to grids. As opposed to a dedicated supercomputer, a grid has a slow network and its processors are heterogeneous and dynamically changing. The slowness of the net has already been

addressed by the team at UNIPASSAU and others with work on tiling, a technique of coarsening the granularity of parallelism. CompSpread will address the problem of heterogeneity. The challenge is to decide which data to send at what time to which grid node.

COA is an INRIA ARC project starting in February 2005. The goal of this project is to propose an experimental software platform for both dynamic adaptation and steering of components. It also investigates how aspect weaving can be used to achieve these 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), aimed at developing a Grid-based inter-operability platform capable of supporting next generation Supply Chain Management applications specifically addressing the needs of SMEs located in industrial districts and dynamic supply networks (2005-2007)

Layered Components UOW is working on a project entitled: A Hierarchical and Reconfigurable Layered Component Model for a Generic Grid Platform.
The objective is to produce a layered structure of the components system by differentiating system and application components. Another goal of this work is to identify the properties that should be satisfied 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 starting work on this project, aimed 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 a 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 in Grid applications that require 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 visualization resources within their future research activities. Existing terascale computing environments can 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)

ISS: Intelligent Grid Scheduling System EIA-FR in collaboration with EPFL are involved in the Swiss project ISS. The aim of ISS is to establish a test bed facility of Swiss national scope for implementation of wide area computing and data handling. In order to set up such a test bed, existing Grid middleware tools need to be extended and modified to suit distributed applications. The Grid programming tool POP-C++, developed by EIA-FR, will be enhanced within ISS by improving its resource management part and by integrating POP-C++ with existing Grid middleware such as UNICORE.

Exchange programs several partners of the Institute are involved in bilateral exchange programs associated with research topics that are related to the Institute activities: DAAD/ARC is an exchange program involving UNI-PASSAU Paul Kelly's group, Imperial College. The main topic is dynamic program optimization (7/2004-6/2006), DAAD/PROCOPE is an exchange grant involving UNI-PASSAU and Albert Cohen's group, INRIA FUTURS, Paris. The main topic is metaprogramming with Meta-OCaml (1/2004-12/2005). BFHZ/CCUFB is an exchange grant involving UNI-PASSAU and Paul Feautrier's group, ENS Lyon (1/2004-12/2004), the main topic is concerned with issues of loop parallelization. VIGONI is an exchange program involving WWU MUENSTER and UNIPI. The main topic was skeleton based Grid programming environments and components (2004-2005). These last two exchange programmes have finished but the work overlapped with early CoreGRID activities.

Most of these research programmes have direct connections and synergies with the topics covered by the Programming model Institute. Involvement of the Institute partners in these (and possibly, in other) projects facilitates the research framework which furthers the ongoing integration activity of CoreGRID, and, as a consequence, the Programming model Institute activities.

2.2.1 Problems and challenges

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

New programming models are required that exploit a layered approach to GRID programming and which 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 number of sub-challenges 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 organized 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 constructed by suitably wrapping existing code. Component composition must guarantee scalability, i.e. the implementation of component composition and the composition model itself must be "bottleneck free"
- Semantics must be defined, precisely modeling 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

- Security issues must be investigated and proper implementation techniques must be adopted to allow safe and secure component based grid computing.

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, scaleable, and suitable for use in pervasive and ubiquitous contexts.

2.2.2 Objectives

The research program of this 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 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 create new (sequential or parallel) components

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

All of these tracks must address problems related to semantics, security and performance prediction, i.e. they must involve activities aimed at defining suitable semantic tools, as well as performance models to reason about component based programs, at all levels (viz. the level of the programming model of the single component, the component framework and at the level of advanced programming models built on top of the base component model). In addition, all of the tracks must consider the security issues related to component code and (parameter) data located on the distributed grid resources, as well as all the security problems related to user authentication and grid resource access management.

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 sub 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 application parallelism exploitation patterns, and to supplying users with mechanisms 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 Institute will help place the European Grid community at the forefront of the field.

The organization of Institute activities using these three related tracks has been agreed by the Institute partners as being the most effective way to address the problems listed at the beginning of Section 2.2.1. In particular, by dividing the work into activities related to the basic programming model, to component and component compositions and to advanced programming models, the problems such as programming language/platform independence, 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.

2.2.3 Tasks

All of the activities of the Institute are organized within three separate tasks based on the three objectives described in Section 2.2.2, namely:

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

ISTI/CNR, IC, INRIA, WWU Muenster, UCHILE, UNIPASSAU, UNIPI, EIA-FR and VUA are involved in the activities of Task 3.1.

The main goal of this task is to coordinate on-going research activities inside the Institute related to programming models for the implementation of single components of a GCM (Grid Component Model, the component model currently investigated in the Programming Model Institute) application. Single components can be either sequential or parallel, they can include legacy codes and can be very demanding with regard to optimal use of the underlying hardware. Finally a component can interact with non-GCM based applications. Based on on-going research activities in the Institute, task 3.1 aims to produce the definition of a precise range of programming models suitable for use in single components. In addition, this task aims at designing suitable techniques to optimize the communications (point to point as well as collective) occurring between and within components, i.e. inter- and intra-component and with non-GCM based applications. As a consequence, activities of task 3.1 comprise two sub-tasks: 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 as well as interaction with non-GCM based application. In addition, task 3.1 is concerned, in association with tasks 3.2 and 3.3, with the followings topics:

- Component run time support: concerned with the features/mechanisms needed to run components on Grids as well as the (optimized) run time systems supporting component execution
- Interoperability: that is, all the problems related to the possibility of run component programs interacting constructively with non-component software built according to existing grid standards as well as with component software developed according to different component models

Sub-task 3.1.1 programming model Beside the numerous already available programming models for programming parallel and distributed single components (such as MPI, OpenMP, HPF, RMI,...), several CoreGRID partners are working on original approaches for parallel programming models as, for example, PaCO++ [25], Pro-Active [6], Ibis [32], Superscalar, ASSIST [2] or POP-C++ [23]. One of the main activities of sub-task 3.1.1 will be to survey existing models and those under development, to compare them and to identify suitable primitives and mechanisms for programming single parallel components. Chosen models should support programming parallel components based on SPMD and MPMD programming styles. In the next months, this activity will mainly be undertaken trough the deliverable D.PM.06: "Programming models for the single GCM component: a survey".

Sub-Task 3.1.2 communication mechanisms Communications are used to transfer data between tasks and to synchronize them. Two types of com-

munications are identified: point-to-point communications and global communications. The way in which a programmer is exposed to communications (communication model) depends on the programming model he is using. It can be explicit messages transfers, as in MPI, or implicit communications as in Pro-Active or POP-C++. However, in both cases, communication mechanisms may greatly influence the performances of both the single component and the interaction between components. Thus better co-design of the communication model and the communication mechanism may significantly improve the efficiency of the computation. In this sub-task we will make explicit (and compare) the communication mechanisms used by the various CoreGRID teams implementing parallel and distributed programming models (see above) in order to identify the most suitable for implementation of the programming models identified in the sub-task 3.1.

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.

ISTI/CNR, IC, INRIA, UCHILE, UNIPASSAU, UNIPI, EIA-FR and UOW are involved in the activities of Task 3.2.

A proposal for the GCM has been delivered (in D.PM.02). It highlights the main features to be included in a component model suited to Grid programming. This proposal needs to be further refined and assessed in order to produce a generalization of the component model.

With this aim, the activities of task 3.2 will focus on the following aspects:

- Refinement of the abstract definition of the GCM. The current abstract definition needs agreement on the following items:
  - basic programming models as defined by task 3.1,
  - communication modes among components and parallel communications: by default, GCM components should rely on asynchronous method invocations, but the component model must accept other modes, such as streaming, events, file transfer. All of these modes must coexist, and allow numerous variations (synchronous/asynchronous, one way/two ways, 1-to-1/1-to-N/N-to-M, etc.)
  - parallel components: it is important to be able to define which components rely on several identical processes (how to define them, how and where to launch them, how to synchronize them, how to enable them to communicate, etc.),
  - introduction of controllers to realize Grid specific features, such as adaptivity, autonomicity,
  - introduction of dynamic reconfiguration of controllers to provide adaptivity at the level of the components themselves.
- Concrete definition of the component model: This definition relies on the definition of an API, an ADL (Architecture Description Language), etc. The objectives of the Institute include defining such a concrete view:
  1. Component Specification as an XML schema or DTD (Document Type Definition) (inspired by existing ADLs)
  2. Run-Time API defined in several languages (inspired by and extending the current Fractal API spec)
  3. Packaging described as an XML schema

- Definition of several conformance levels. These conformance levels should allow any CoreGRID partner to relate existing component models to the GCM, by determining the conformance levels that are reached. Conformance levels will be defined according to several distinct features: the basic conformance levels will encompass basic characteristics upon which more sophisticated ones will be built.

  The following characteristics of the component model are to be considered. We list the characteristics below , beginning with the most fundamental. The first conformance level will probably include most of the first items below:

  - Hierarchical features,
  - Distribution and definition of distributed components (required deployment/QoS attributes),
  - Introspection, at the component level and at its interface level,
  - Dynamic (functional) reconfiguration,
  - Type definition for components (inspired by existing ADLs),
  - Supported communication modes,
  - Interoperability between components: this should rely on standard Web Services interfaces, but should not prevent ad-hoc interactions, like Babel for CCA components,
  - GCM ADL (Architecture Description Language), in order to instantiate components thanks to factories,
  - Multicast/Gather interfaces, to optimize and parameterize 1-to-N and N-to-M component interactions,
  - Component controllers as first class entities, allowing the definition of controllers using components, thus permitting the reconfiguration of non-functional properties dynamically,
  - Autonomicity level, through the definition of standard autonomicity controllers,
  - Association of a behavioral definition with each component, to verify that their composition is semantically correct.

Those definitions must be addressed in task 3.2. This activity will be conducted through 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.

INRIA, QUB, WWU Muenster, UNIPASSAU, UNIPI, UOW, UPC and VUA are involved in the activities of Task 3.3.

The activities of task 3.3 are all related to the coordination of the research activities of the Programming model Institute partners on advanced programming models that can be built on top of, or integrated with, or usefully exploited in conjunction with the component based programming model which will eventually be defined within the Institute.

In particular, within this task, several research tracks will be coordinated, relating to:

- "theoretical" computation models that can be used to support the grid component model, that is abstract programming models raising the level of abstraction provided to grid programmers,
- performance models developed to cover major grid computing parallel patterns,
- program transformation techniques, aimed at improving the performance features of the user supplied code via automatic program rewriting techniques
- autonomic component and parallel application control, taking care of all the details related to the management of component interaction with the underlying grid middleware,
- advanced security issues related, for example, to secure component code deployment, secure data transmission, authenticated component access, etc.
- "invisible grid" concept, as defined in the NGG expert group documents [31, 19] and provided to the application programmer level, allowing efficient grid application development without requiring any kind of knowledge in specific, grid-related programming techniques.

The expected outcome of this Task is related both to the grid component model whose associated research activities are coordinated in Task 3.2 and to the research activities of other CoreGRID Institutes. In particular, the activities of Task 3.3, being related to "advanced" programming models, are close to those engaged in by the Institute on Problem solving environments, tools and GRID systems. This is because both the "advanced programming environments" simply build on top of the grid component model of Task 3.2. In addition, the integrated programming environments that are investigated in the Institute on Problem solving environments, tools and GRID systems are directed towards the same goal: *raising the level of abstraction provided to the grid application programmer.*

The activities associated with Task 3.3 build on the experiences of Institute partners that either predate CoreGRID activities or have been (and possibly are currently being) developed within "private" (that is non-CoreGRID) partner research projects. Such experiences include:

- the ASSIST structured, high performance programming environment, supporting component programming and providing users with components, including autonomic managers taking care of ensuring (best effort) user supplied performance contracts (aka QoS requirements) and with components modeling common parallelism exploitation patterns,
- the HOC programming environment, providing users with grid programming components, which are pre-packaged with parallel implementations and middleware support and which are accessible via Web services, abstracting over the middleware platform.
- experience of using semantically well founded and elegant abstract programming models, such as the ORC model of Misra [22].

The activities of Task 3.3 are related to the integration of the existing research activities at the partner institutions as well as to the development of new research activities financed either through new projects or extended projects. Partner of the Institute will exploit short visit and researcher exchange programs.

UNIPI has responsibility for Task 3.3 (Marco Danelutto).

2.2.4 Drivers

The programming of sequential computers has benefited 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 view of 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):

- Institute 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),
- HES-SO/EIA-FR: Haute Ecole Spécialisée de Suisse Occidentale (EIA-FR)

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

- 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 the agreed component model. To give a complete picture of the approaches, we have selected the following requirements to distinguish approaches:

1. Interoperability
2. Platform independence
3. Programming language independence
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 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 Institute. 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
POP-C++		yes		yes	yes			ongoing	yes

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

- *INRIA*, OASIS team at Sophia-Antipolis have developed *ProActive* [6], 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 hierarchical component model implemented is Fractal. An important characteristic of this implementation, which uses ProActive, is that components are distributed, possibly running on several JVMs simultaneously. A Fractal-ProActive component is thus a powerful abstraction of distributed Grid activities which could thus facilitate their programming, their configuration, and their deployment.

- INRIA, SARDES team at Grenoble, jointly with France Telecom R&D has designed an abstract component model, Fractal [7]. Fractal proposes a hierarchical, dynamic and extensible component-oriented programming approach which could be 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 developing a framework for designing parallel adaptable components (Dynaco [13] and AFPAC frameworks). The objective of this framework is to help developers of applications that are suitable for execution on Grid to take into account the changes that may occur in such environments during application runs [8].

  The PARIS team has also developed a parallel extension to the CORBA component model, GRIDCCM [26]. GridCCM enables a simple and efficient embedding of SPMD code in a parallel CORBA component. Thus, parallel communication flows and data redistribution are possible during an operation invocation.

- *IC* has developed *ICENI* [16], 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, behavior 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 [30], a portable reflective system for Java that has evolved to a versatile kernel for aspect-oriented programming [29] (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 [14].

  – 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 ([17], [18]).

- *WWU* has developed and implemented *Higher-Order Components (HOCs) [12].* HOCs are program components 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 mobility in a WSRF (Web Service Resource Framework) 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 management systems for reuse of data and code
  – Potential for semantics-preserving transformations on programs built from 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 [5] 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 using a graphical workflow description language based on colored 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 [21], 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 [11] 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 which evolved in the FIRB research project GRID.it. GRID.it involves a number of research institutions in Italy. ASSIST is a 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, programmers simply structure their parallel applications,

then the ASSIST compiler tools and run time system execute the application on the Grid architecture and adapting the application to the particular Grid configuration. In particular, the ASSIST tools take care of adapting the application execution to target Grid architecture features, according to an abstract performance contract provided by the user/programmer. The adaptation process is integrated both in the single parallel modules compiled from ASSIST `parmod` generic parallel skeleton and in some 'super-components" modeling common grid parallelism exploitation patterns. Within super-components, user provides a sort of performance contract to be respected and the super-component autonomic managers take care of ensuring, in a "best effort" way, such contract. In addition, ASSIST provides full interoperability with existing CCM and Web Services frameworks, in that ASSIST components can be wrapped as CCM or WS (Web Services) 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 [32], a Java programming environment for parallel applications, providing efficient communication mechanisms, including an efficient RMI, group communication (GMI) as well as replicated objects (RepMI), message passing (MPJ), and a system for Grid-aware divide-and-conquer parallelism (Satin). The Ibis runtime system supports efficient communication in Grid environments by using parallel TCP streams and communication through firewalls. Support for encryption and compression is under development. The VUA group has also implemented a Java version of GridLab's Grid Application Toolkit [4] (GAT), enabling dynamic access to Grid resources via a variety of Grid middleware platforms (Globus 2, 3, 4, Unicore, ssh, gLite, etc.). Ibis can use the Java GAT for integration with Grid middleware platforms.

- EIA-FR is developing POP-C++ [23] a comprehensive object-oriented system for developing HPC applications on the Grid. POP-C++ environment consists of a programming suite (language, compiler) and a run-time system for running POP-C++ applications. POP-C++ language is an extension of C++ that implements the parallel object model with the integration of resource requirements into distributed objects. POP-C++ has been expressly kept as close as possible to C++ so that programmers can easily learn it and that existing C++ libraries can be parallelized using POP-C++ without too much effort. POP-C++ V.1.1 is available at www.eif.ch/gridgroup/popc and on SourceForge.net

3.2 Extended context

Currently, the starting point for many research projects and experimental implementations of component architectures are two widely known concepts:
the *Common Component Architecture (CCA)* [15] and the *CORBA Component Model (CCM)* [24].

- *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 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.

On the basis of the features available in the current implementations of CCM, CCA and other component models for the grid, most current projects exchange 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

A vision which tries to foresee what can be expected in the future for the Grid Programming is sketched below.

This vision stems from the conviction that the great majority of complex software applications will, in the mid-long term, be dynamically built by composing services which will be available in an open market of services and resources. In this sense, the Grid might be conceived as a "world-wide cyber-utility" populated by self-* (self-configurating, self-healing, self-optimizing, self-protecting) cooperating services which interact as in a complex and gigantic software ecosystem. Those services could be set up by a variety of agents (e.g., software companies, open source developers, but also states and institutions, as a public service for their citizens), and will be available to developers or proxies of them (e.g. software agents), who will choose to use software solutions from different vendors, sold at different prices, with different performance and QoS. This will lead to a new style of application development based on software services, which will in turn need the expansion of existing practice regarding service location, standardization, semantics of services, certification, and others.

In this style programmers do not start from scratch but build instead new applications by reusing existing off-the-shelf services available in a galaxy of services offered within the Grid. The ecosystem is the set of connections among different services, competing for visibility and adoption (which would help them survive when confronted with other, competing services), and which, notwithstanding, will cooperate when put together in large applications. When a complex service uses smaller services, an association is created between all of them. This can be easily compared to the well-known concept of the Web. In this view, mainly for cost reasons, and also in the quest for diversification which would make it possible to survive in a rapidly changing world, it is foreseeable that no single company or organization would be interested in creating by itself very complex and diverse software applications. Future business and scientific applications will be built as a complex network

of services offered by different providers, on heterogeneous resources, constrained by administrative problems when crossing the borders of different organizations [20, 27, 9].

NGG3 and NESSI In the last quarter of 2005 the Next Generation Grids expert group was reconvened to identify the gaps between the leading edge of Grid technologies and the end-user. The previous NGG work had identified the need for two aspects: semantically rich facilities and a service-oriented approach. The convergence of the evolving NGG vision with the service-oriented vision of significant European industry stakeholders in NESSI (Networked European Software and Services Initiative), an industry-led Technology Platform that aims to provide a unified view for European research in Services Architectures and Software Infrastructures [NESSI]) naturally led the NGG3 group to define the scientific and technological requirements necessary to evolve Grids towards the wider and more ambitious vision of Service Oriented Knowledge Utilities (SOKU) [19]. The SOKU vision identifies a flexible, powerful and cost-efficient way of building, operating and evolving IT intensive solutions for use by businesses, science and society. It builds on existing industry practices, trends and emerging technologies and gives the rules and methods for combining them into an ecosystem that promotes collaboration and self-organization. The benefits are increased agility, lower overhead costs and broader availability of useful services for everybody, shifting the balance of power from traditional ICT (Information Communications Technology) players towards intermediaries and end-consumers of ICT. The need for developing the SOKU vision stems from the necessity of effectively bringing knowledge and processing capabilities to everybody, thus underpinning the emergence of a competitive knowledge-based economy.

Cooperation between CoreGRID and NESSI The CoreGRID NoE and the NESSI ETP organized a joint workshop in Brussels on January 27, 2006. At the conclusion of this event the convened participants agreed on the need to foster efficient collaboration and co-operation between CoreGRid and NESSI. In particular, it was stressed the idea that CoreGRID might provide an effective contribution to the European ICT industry's wealth creation, demonstrating the value of long term focused research to NESSI and to NESSI enterprises. From the other side, NESSI might express long term scientific and technological needs and challenges, providing innovative industry-driven Grid usage scenarios. In this framework the NESSI people manifested particular interest in order to start a scientific discussion on the design methodology for a Generic Reconfigurable Component-based Grid Platform with two CoreGRID Institutes: (i) Programming Models and (ii) Grid Systems, Tools and Environments" [10].

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

The Institute on Programming Model aims at coordinating and enforcing 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 Institute activities are Grid application programmers and, in particular, those programmers that presently 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[28, 31, 19], the Institute activities will be focused on providing these programmers with suitable mechanisms and programming environments to implement the invisible Grid concept, while supporting all of 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 as end users for Institute activities. For example, programmers developing Grid PSEs will clearly benefit from the use of high-level component programming model that represents the Institute's main goal. The Institute on Programming Model plans to build on existing Grid and programming model technology. In particular, the GCM (Grid Component Model) will be developed from established results in the component programming models community, possibly addressing and solving those problems that are still considered open, always remembering that the goal is to provide a component model suitable for efficient Grid programming. Thus, all existing Grid technology features and results will be evaluated and exploited as needed. In particular, the results achieved during the development of the CORBA Component Model [24] will be considered, as will those available from CCA research programs [15] and those from the Institute partners' component related activities, such as those cited in the State of the art Section. Overall, the Institute envisages a scenario where Grid programmers can exploit the GCM to build efficient, scalable and performant Grid applications by simply exploiting their application domain specific knowledge, rather than domain specific *and* Grid middleware/tool knowledge, and by exploiting the compositionality, modularity and interoperability features which characterize component models.

4.2 Strategy

Overall, the members of the Programming Model Institute agreed that the goal of defining a joint *Grid Component Model* (GCM) could be best realized by undertaking the three sub-tasks originally proposed and outlined in the Section 2.2.2, 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 is organized as follows.

First, we worked on the development of an *abstract view* of the Grid Component Model. Such a view allows all the partners to define a joint view of what should be in a component model for the Grid. This abstract view consists mainly in defining what a primitive component is, what a hierarchical composition is, and what the various kinds of ports and interfaces are, but also what are the different features that could or should be defined by a component platform. This first phase resulted in an initial proposal for the GCM, constituting the first GCM deliverable (D.PM.02).

This deliverable defines the main features to be included in the GCM, as currently assessed within the Programming Model Institute. The GCM is intended to be a precise specification of an effective Grid Component Model. The GCM proposal defines Grid related features taking Fractal as the reference model. The main features proposed by the GCM are: virtual nodes, collective interfaces, dynamic controllers, and autonomic components.

Second, starting from the first deliverable, we intend to provide a second deliverable in a form that can be used as an abstract model.

Another important part of this second phase will be the creation of the technical specifications of the GCM. Based on the specifications of Fractal (ADL), but also being mindful of the technical specifications of other component models, such as CCM, the long term objective of this work is to specify the ADL, the API, and all the

necessary technical details of the GCM. This specification should be precise enough to allow interoperability of GCM components, but general enough to allow almost all existing component programming methodologies to be implemented as GCM components. The 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.

It is also crucial that, in this phase, the single component and the advanced programming model tasks (tasks 3.1 and 3.3) develop the programming models and features that will support and fit with the GCM.

This second phase also necessitates interactions with other initiatives, and the other CoreGRID Work Packages. Indeed, since the objective of the GCM is the development of a well-defined and effective component model for the Grid, the Programming Model Institute needs feedback on and assessment of the process being followed, and the decisions being taken in the design of the GCM.

The third phase will focus on assessment of the GCM.

Each Programming Model Institute partner will exploit the GCM within its own research activities, which will constitute the first assessment of the GCM. More generally, the objective of this third phase is the assessment of the GCM, and its adaptation to the particular needs of those within the Programming Model Institute, in other CoreGRID Work Packages, or in other EU funded projects. In addition, we aim to implement GCM reference implementations, experiment with GCM features and so on, in new projects.

At all stages, both practical and more theoretical aspects will be addressed. For example, the first task of section 2.2.2 will deal both 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 intended to culminate in the definition of GCM *and* the definition of all the related theoretical aspects, such as for instance the semantics of the component and component composition. This will be done, of course, in the perspective of the lightweight idea of a component model discussed in Section 2.

4.3 Roadmap

4.3.1 Phases of the roadmap

In the roadmap, we discern the following three phases:

Initiation During the initiation phase, the Institute activities are devoted to collecting ideas currently being pursued by the partners, possibly arising in the context of the different "local" projects in which the partners are involved. In this phase information and requirements will be gathered from the other Institutes in the NoE. This phase ends when we finalize the second roadmap deliverable. The only thing not completed is the collection of the requirements coming from the other Institutes, as this is deemed to be a "continuous" process.

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

elucidated in the component model definition. This phase is already well advanced. The partners have agreed on the general features to be included in the component model. These features are currently being refined and we expect the "complete" model to be ready by the end of the second year. Subsequently, the component model documents will be passed to the other CoreGRID Institute leaders, so that collective knowledge can be assessed. Results will be presented at the second CoreGRID Integration Workshop.

Assessment All of the partners will agree on the component model, start using it in their own projects, and promote it outside the Institute and throughout Europe. During the assessment phase, partners of the Institute are expected to actively participate in standardization bodies which support the component model. They are also expected to be involved in the preparation of new project proposals (in different contexts) related to the grid component model topics.

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

Institute activities in the context of the roadmap discussed in this document will be checked against a set of milestones. Some of the milestones were included in the original DoW. Others have been added through the new JPA documents prepared during the first year. At the moment, we assume the following list of milestones for the Institute:

1. M.PM.01: Coordinated research framework infrastructure established and work package web site up &running (M3) (completed)
2. M.PM.02: Agreement on the main features of the grid component model. (M14) (completed)
3. M.PM.03: Agreement on the complete GCM (about M24)
4. M.PM.04: GCM assessed, all partners either adopt it in their research projects, or they demonstrate why they use something different. In the latter case, new features to be included in the GCM, may be proposed. (about M36)
5. M.PM.05: Partners agree on continuing Institute activities beyond CoreGRID (M48)

4.3.2 Mechanisms

The chief mechanisms for ensuring close and profitable cooperation among the partners will be workshops, meetings, short visits and researcher exchange programs, 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 one or two months prior to its due date to discuss its content, and some E-meeting activity will take place during the last month prior to the due date to finalize content. In general, we will try to have at least two general meetings per year, possibly one at the beginning of the year and one just before summer. These meetings will be used to improve partner cooperation through presentations of the work carried out at each site.

E-meetings E-meetings may either be realized as teleconferences, or as email-based coordination and integration activity. The former is more effective where a relatively small number of participants are involved. The second is more effective when a larger number of participants is involved and is also

attractive because of their natural asynchrony. We plan to have e-meeting activity both to prepare the meetings and to discuss all the major deliverables or events of the Institute.

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

Documents Working documents will be shared via the collaborative tool BSCW or, if needed, via specific CVS sites set up for this purpose. Completed documents (and other documents relevant to the work package activity) will be available on the Work package web site.

Software Software (e.g. reference implementation components) will be shared via a tool coordinated by Work package 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 involve a direct contribution to the activities of the Work package.

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

In addition to these mechanisms, during the first year of activity we have decided upon another mechanism which will help to ensure continuing cooperation among the Institute partners and facilitate exploitation of common research results: the *research group*. In particular, during the Barcelona meeting, in June 2005, the partners agreed to concentrate coordinated partner activities on a small set of topics. These are more focused with respect to the original list of tasks given in the DoW and are evident already in the Institute organization. This also arose in response to suggestions of the SAB following the first meeting held in Barcelona in June 2005.

Partners of the Institute agreed on a list of topics, some new, and some being taken directly from those listed in the original DoW. In particular, the partners agree to have explicit research groups on Security issues, since both SAB and the first year reviewers pointed out that this was a highly important topic, which had not been explicitly mentioned in the first DoW.

The list of the currently active research groups, along with a short description, is reported below. An up-to-date list of the current research groups is currently listed under the Institute home page at the CoreGRID web site. However, we stress that these research groups are dynamic in nature, in that when their goals have been attained, they can be dissolved and new groups initiated. In addition, we expect that the main contribution of research groups will be in technical papers summarizing the results achieved in particular research areas, and that these results can be used by both the Institute partners and by other CoreGRID partners to better focus their research activities related to programming models.

The currently active research groups are the following:

Component communication responsible Duennweber, WWU Muenster, devoted to investigation of all the aspects related to efficient inter and intra component communications (WWU Muenster, INRIA, ISTI/CNR, EIA-FR)

Programming model for the single component responsible Pierre Kuonen, EIA-FR, devoted to investigation of the models suitable to be used to program a single (possibly parallel) component (EIA-FR, INRIA, UNIPI, UNIPASSAU)

Component definition responsible Ludovic Henrio, INRIA, devoted to investigation of all the aspects related to the correct way of defining and describing components (INRIA , UNIPI, VUA, IC)

Advanced programming models responsible Marco Danelutto, UNIPI, devoted to investigation of the more abstract, efficient and user friendly programming models that can be built on top of the basic grid component model (UNIPI, INRIA, WWU Muenster, UNIPASSAU, VUA, QUB, UPC)

Performance models, adaptivity responsible Marco Aldinucci, UNIPI, devoted to investigation of those aspects that can be exploited to provide components with autonomic adaptive control (UNIPI, ISTI/CNR, INRIA, WWU Muenster, UNIPASSAU, VUA)

Component run time support responsible Thilo Kielmann, VUA, devoted to investigation of the aspects related to the run time support of components on grids (VUA, INRIA, UNIPI)

Security responsible José Piquer, UCHILE, devoted to investigation of all the aspects specifically related to the secure usage of components on the grid (All partners)

Standards responsible Thilo Kielmann, VUA, devoted to propagation of the GCM model through national and international standard committees and bodies (partners involved in National initiatives and decision making committees)

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 the GCM should act as an impetus for joint applications for support under international bilateral funding arrangements.

4.3.4 Institute roadmap in the NGG perspective

Within the three approaches to describing the viewpoints for NGGs [28, 31] (end-user, architectural, software) the activities of the Programming model Institute are more directly concerned with the *Software viewpoint*, in which the grid is considered as a programmable system, even though the work may also have an impact on the *Architectural* and *End-user* viewpoints.

The Programming model Institute roadmap, focusing on component orientation, clearly contributes to *Programming Grids through abstraction*, thanks to a virtualization of grid resources. Indeed, component orientation provides a strong structuring mechanism by which resource and other sorts of requirements can be expressed as a declarative, external property of a component (i.e. as distinct from its functional part). The way in which 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 Programming model Institute 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, by building on top of Web Services technologies) dovetails with the service-oriented utility vision. More precisely, even if Web Services support is considered as a means of realizing interoperability, this does not mean that Web Services per se are a suitable programming model 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 naturally enable *Interoperability as a basic means for problem solving*. In this strategy, it is critical to define a standard for component 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 if the adopted component model offers a well-defined **hierarchical composition** model rather than a simple flat assembly. For instance, self-reconfiguration of a hierarchical component may imply changing instances of inner components, to modify bindings, etc. This can be achieved without clients being aware of it if the component is a hierarchical one.

4.3.5 Interaction with Industry

As part of the activities undertaken during the phases of the Institute roadmap, interactions with European industry will be given increasing priority. Our main channel for initiating such contacts are being established via the CoreGRID Industry Advisory Board but many good contacts with industry exist via individual partners or research groups of the Institute. Another important avenue to be explored in the near future is the proposed collaboration with the NESSI initiative following the recent CoreGRID–NESSI Workshop, 27 January 2006 in Brussels. Some more specific forms of interaction with industry include information exchange activities such as invited speakers from industry, seminar talks for industrial partners, discussion panels, etc. A stronger level of interactions has been initiated as part of the "Component Model Definition" phase of Roadmap-2, where the Institute expects to receive feedback from industrial partners about the GCM specification. In parallel, we are expecting the initiation of focused collaborations involving higher levels of commitment. During the assessment phase, in addition to the already mentioned types of activities, the Institute is aiming to have established spin-off projects in collaboration with partners from industry with joint goals and deliverables in the topic areas emerging from the work of the Institute.

5 Security

Security is a major concern in grid programming. So important is it that every new version of grid middleware provides more and better mechanisms to handle security issues. For example, the Globus toolkit 4 includes security mechanisms for both pre-WS and WS worlds. In particular, in the WS world, GT4 provides mechanisms to guarantee message level security, transport level security and a full authorization framework supporting several authorization schemas (www.globus.org/toolkit/docs/ 4.0/security). These mechanisms can be used explicitly to take care of grid application security and, in particular, to avoid data capture, third party intrusions or even misuse (unauthorized usage) of grid resources. However the use of these mechanisms and tools is still completely under the control of the grid application programmers. In spite of the fact that the use of a secure transport layer can guarantee most of the protection needed to implement a secure grid application, the full exploitation of secure mechanisms requires a huge

programmer effort.

As the Programming model Institute concentrates its activities on the Grid Component Model, it is natural that security issues be considered *in the light of the GCM*. This entails the analysis of all the ramifications associated with secure mechanism use, as well as carefully taking into account the extent of security considerations needed to implement a GCM suitable for the development of *secure* and efficient grid applications. Following the approach taken within the overall Network of excellence, we have identified security as an "horizontal task" in our Institute. Rather than setting up a specific task concerned with security, we have activated a research group, embracing all the teams participating in the Institute's activities. This research group will investigate security related topics concerned with all aspects of the design of the GCM: security of the single components used to build a grid application, of the mechanisms used to make components interact each other and of the mechanisms used to deploy and run GCM applications on grid nodes.

More specifically, the Programming model Institute activities related to security issues will seek solutions for different problems associated with the implementation of grid applications with GCM:

Secure deployment of components Components should be deployed on grid nodes in a secure way, so that they cannot be read/caught by unauthorized people nor be substituted during the deployment.

Secure access to components When components have been deployed or, in general, in the case of access to remote components, authorization of accesses has to be guaranteed in such a way nobody can benefit from component services but the users legally authorized to interact with those components.

Secure data transfer To protect data, users should be enabled to demand the securing of all (or part of) the messages flowing through the grid during component program execution. In particular, the user should be allowed to ask for partial message securing (only to a sensible level, in order to minimize performance overhead paid) as well as to specify the kind of securing mechanism used.

Secure access to grid component middleware To avoid the possibility that unauthorized components be able arbitrarily to access machine resources (CPUs, disks, network facilities, etc.) some regulated way of accessing GCM grid middleware should be introduced, taking care to avoid the introduction of unacceptable overheads while ensuring that only certified components can use machine resources.

All of these problems will be addressed while keeping in mind that:

- the more security is introduced, the better the features offered to the grid application users,
- secure mechanisms usually have an associated cost in terms of the overhead introduced: they should be introduced carefully and only when they provide benefit.

For example, consider a GCM application running on a grid platform. If the user annotates the application explicitly requesting that only certain component interactions must be secure because they involve sensitive data, it is extravagant to use a secure transport layer across the whole application. In addition, it is well known that the secure transport layer introduces a significant overhead with respect to a non-secure transport layer. The indiscriminate introduction of such a layer will likely degrade the performance of the grid application unacceptably. Also, if higher-level

programming models are used on top of the GCM, performance related considerations could be used to motivate the introduction of secure mechanisms, as outlined in [3].

Overall, security related activities in the Programming model Institute can be classified as those involving only middleware security, rather than direct grid application security. Grid application security is achieved by exploiting the secure features of the GCM and of the GCM run time environment and support tools.

Institute partners are committed to participation in any cross-Institute security related initiatives, such as those related to the production of deliverables relating to security. In addition, Institute partners are committed to including a specific security session in the annual plenary Institute meetings.

6 Link with other CoreGRID scientific work packages

Each track of this Institute requires and will implement mechanisms to ensure cooperation with those research activities planned within the other Institutes of the Network. The component model, as a whole, will be used in the Institute on Systems, Tools, and Environments (WP7). Close liaison will be maintained with those working in WP7 activities, oriented towards using the grid component model (GCM) for building an integrated whole of both application-level and system-level components. This integration task will especially evaluate the set of requirements on GCM in systems-level software. The Institute on System Architecture (WP4) will contribute to aspects of dependability, fault tolerance and robustness of grid component frameworks. The Institute on Knowledge & Data Management (WP2) will provide suitable mechanisms for handling all the knowledge-based 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 Institute, advantage will be taken of the products created within the other Institutes of the Joint Plan of Activities. Results from the Knowledge & Data Management Institute (WP2) will be exploited when addressing the problems associated with the initialization and maintenance of the component framework on the Grid. Results from the System Architecture Institute (WP4) will be exploited during implementation of the component framework. Results from the Institutes on Information, Resource and Workflow Monitoring Services (WP5) as well as Resource Management and Scheduling (WP6) 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 Institute.

Throughout the duration of CoreGRID, this Institute will provide results (essentially, the common component model) that can be used effectively in other Institutes of this JPA in the development of component based applications, run time systems, and support tools on the Grid. The two-way exchange of results between the Programming model Institute and the other 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 the CoreGRID framework, where the partners merely pursue their local goals without specific recourse to the important results generated in the rich diversity of related European projects.

7 Contribution for the European GRID roadmap

The main "scientific contribution" aspects of the Institute have been already outlined in Section 4.3.4. Here we point out further expected contributes of the Programming model Institute activity to the European GRID roadmap.

7.1 Contribution of the project

Within the three approaches to describing the viewpoints for NGGs [28, 31] (end-user, architectural, software) the activities of the Programming model Institute are more directly concerned with the *Software viewpoint*, in which the grid is considered as a programmable system, even though the work may also have an impact on the *Architectural* and *End-user* viewpoints.

The Programming model Institute roadmap, focusing on component orientation, clearly contributes to *Programming Grids through abstraction*, thanks to a virtualization of grid resources. Indeed, component orientation provides a strong structuring mechanism by which resource and other sorts of requirements can be expressed as a declarative, external property of a component (i.e. as distinct from its functional part). The way in which 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 Programming model Institute 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, by building on top of Web Services technologies) dovetails with the service-oriented utility vision. More precisely, even if Web Services support is considered as a means of realizing interoperability, this does not mean that Web Services per se are a suitable programming model 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 naturally enable *Interoperability as a basic means for problem solving.* In this strategy, it is critical to define a standard for component 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, flexibility* are much more manageable if the adopted component model offers a well-defined **hierarchical composition** model rather than a simple flat assembly. For instance, self-reconfiguration of a hierarchical component may imply changing instances of inner components, to modify bindings, etc. This can be achieved without clients being aware of it if the component is a hierarchical one.

7.2 Improvements in communication

The aim of reaching agreement on a common Grid Component Model at the European level will foster communication among European research teams and facilitate a coordinated effort towards the realization of a model for Grid computing. A questionnaire completed by the various partners, indicates what component model and component model features they recommend. Establishing a standard for representing component metadata using the XML notation will create synergies yielding the identification of an agreed core set of features that the GCM should have (though this set will be open to further extension).

7.3 Concentration of resources

The definition of a common GCM will be the concrete outcome of bringing together knowledge and concepts relating to Grid programming that are today scattered across Europe.

An implementation of the proposed component model will be definitely needed, however. In particular, the implementation of an effective component framework to permit the integration of the diverse components currently used in different institutes to effectively let already existing components interoperate (as a means) for problem solving on the Grid is needed. The role of such a framework would be that of a 'super broker' in the sense that it would not demand the replacement of existing component model frameworks but, rather, allow them to negotiate and collaborate. In this respect, the concentration of resources would still be within current developments and work in progress. In the first version of this document, we envisaged that specific efforts to build a reference implementation of a common Grid component framework will require additional and targeted funding from the EU. Actually, in the meanwhile partners of the Programming model Institute presented a STREP proposal that was accepted in the FP6 Call 5. The STREP, named "GridCOMP", is aimed at providing a first prototype of the GCM proposed within the Institute on top of existing open source middleware frameworks. This project has to be intended as a first step devoted to make available to the CoreGRID partners a prototype to play with and to conduct experiences in the different research areas involved in the GCM design. We expect that partners of the Programming model Institute participate in the preparation of other project proposal while the GCM model is being developed, assessed and refined, in such a way that eventually the contribute of the Institute to the European GRID roadmap is a full implementation of the final GCM model, perfectly comparable to other component frameworks but fully supporting the development of efficient, high performance component based grid applications on the grid architectures.

7.4 Reduction of complexity

The proposed GCM is primarily intended to offer a common programming model for Grid applications. However, it is envisaged that this model will provide the basis for programming in the Grid environment itself. Indeed, a component-oriented Grid environment would benefit greatly from possessing all of the advanced features that components can offer, e.g. through the dynamic configuration capabilities. Overall, at the foundation level, components would be configured according to managed resources or hosting devices; at the service middleware level, compositions would result from on-demand required services; at the application level, compositions would result from the engagement of applicative-level services.

Concentration on a single component model at all levels of the Grid architecture would help define a clear European-wide Next Generation Grid vision.

7.5 Collaboration

The Institute partners are presently discussing possible collaborations within the European grid research framework. The most promising possibility at the moment is to participate in or, better, to organize a consortium for the now incoming call, aimed at defining a project whose goal is to experiment and implement the component model discussed in this Institute. In addition, the partners are currently considering making more explicit all links with other EU-funded projects for which partners of this Institute are also partners of various work packages in other projects. This should enhance the quality of the activities performed within the Institute, as

well as facilitating the dissemination of the concepts and ideas arising from the Institute's activities in the European grid research framework.

8 Participants

The table 1 presents a summary of the degree and range of involvement of participants in the Programming Model Institute. Columns headed Task 3.1, Task 3.2 and Task 3.3 indicate their declared involvement in the tasks described in Section 2.2.3.

Partner	*No.*	*Nation*	*Responsible*	*T 3.1*	*T 3.2*	*T 3.3*
ISTI CNR	4	I	Laforenza	yes	yes	
IC	12	UK	Panagiotidi	yes	yes	
INRIA	14	F	Caromel, Perez	yes	yes	yes
QUB	21	IRL	Kilpatrick			yes
WWU Münster	22	D	Duennweber	yes		yes
UCHILE	26	CHI	Piquer	yes	yes	
UNIPASSAU	34	D	Lengauer	yes	yes	yes
UNIPI	35	I	Danelutto	yes	yes	yes
EIA-FR	36	CH	Kuonen	yes	yes	
UOW	37	UK	Getov		yes	yes
UPC	38	E	Gabarro			yes
VUA	39	NL	Kielmann	yes		yes

Table 1: Partners involved in the Institute activities.

8.1 Expected contribution of Institute partners to the roadmap

8.1.1 ISTI/CNR

Isti-Cnr expects to contribute to the roadmap by providing its expertise in the field of service discovery.

It will contribute in the definition of the component programming model, in order to make its interface easily searchable and browsable by automatic tools. In particular, it will contribute to the definition of hierarchically composed components: the way the structure of these components is exposed to programmers can strongly affect the possibility of searching and finding them.

Also, it will contribute to the definition of the programming API so that automatic tools can connect independent components after a search activity.

8.1.2 IC

IC aims to contribute to the roadmap through the following independent activities:

- The development of the middleware ICENI and its refactoring to ICENI II in LeSC. This offers a great opportunity to get involved in the component definition task and explore hierarchical component models as part of the final targeted middleware.
- Research into, and collaboration with partners concerning, Cross Component Optimisations. In more detail, an area of great interest is the challenge which occurs in trying to create such a re-usable and optimal (in terms of genericity, size and optimizer awareness) set of metadata for all the components of a

workflow, taking into account that the optimization itself relies on metadata that characterize a component's performance.

- Gridification and deployment of legacy code applications, such as GENIE (Grid ENabled Integrated Earth system model). These tasks require an in-depth exploration of the (non application specific) meta-data needed when gridifying legacy simulation environments. This environment is also a proof-of-concept of the need for hierarchical component architectures. IC expects to use it as an exemplar application to identify component structure features and component composition characteristics, through collaboration with partners.

8.1.3 QUB

QUB will contribute to the goals of Programming model Institute, and, in particular, to Task 3.3 by devising a methodology in which dynamic/adaptive aspects of Grid applications can be modeled in an abstract notation which will, in turn, allow reasoning about and prototyping of such applications.

Objective: To determine if ORC [22] is a suitable notation for specifying dynamic/adaptive behavior of Grid applications and, if not, to propose appropriate extensions.
The objective will be pursued as follows:

- Elicit suitable case studies from the Belfast e-Science Centre and/or Programming model Institute partners. We require case studies which possess rich dynamic and/or adaptive behavior.
- Describe dynamic/adaptive aspects of case studies in terms of Fractal/GCM controllers. This will allow assessment of the adequacy of the GCM for describing dynamic/adaptive behavior.
- Specify the behavior identified in (2) using ORC. This may require the extension of ORC.
- Simulate key features of the case studies using the ORC specification of (4) and the ORC system (or modified ORC system).
- Assess the suitability of ORC by
  - comparing simulated behavior with actual behavior of case studies; and
  - reasoning about properties of ORC models.

8.1.4 WWU Muenster-22

The WWU Muenster plans to contribute to **Roadmap Task 3.1** and **Roadmap Task 3.3**

- In **Task 3.1**, The WWU will work on communication among distributed components.
  In particular, collective operations, which involve the exchange of an operator, such as `Scan`-operations or `Reduce`-operations, are analyzed.
  - The WWU conducts experiments with HOCs for running `Scan`-operations and `Reduce`-operations on the Grid. In the HOC implementation, operators are represented as mobile code parameters, which can be uploaded to a Web-enabled service.

By comparing the communication mechanisms implemented in HOCs with other existing implementations of collective operations (e. g. , the RMI-based ones in the ProActive library from INRIA), the WWU studies optimizations and data distribution patterns for implementing the most commonly used group communication operations for Grid components as efficiently as possible.
The WWU aims at implementing flexible communication routines, where part of the necessary data transfer between the components participating in a collective operation is handled in parallel, allowing the data to be exchanged more efficiently than via a straight linear distribution (i. e. , the typical broadcast behavior).
The WWU communication experiments have the goal of identifying best practice for implementing cross-component communication. The WWU works on this topic with other partners: Currently INRIA, EIA-FR and ISTI/CNR have declared their interest in participation in the *CoreGRID Research Group on Component Communication*, established by the WWU for this purpose. The results achieved in this group are planned to be gathered in form of a white paper, finished by the end of 2006.

- In **Task 3.3**, the WWU works on advanced infrastructures for running distributed components. In particular, *Monitoring*, *Scheduling* and *Discovery* mechanisms for components are analyzed.
  - The WWU works on the integration of standard tools, such as *Globus* MDS, DUROC, WS-GRAM etc., with HOCs to enable the execution of new, component-based software into existing large-scale infrastructures, e. g. , the DAS-2 Grid test bed in the Netherlands. There is an ongoing collaboration on this topic between the WWU, the TU-Delft and the *Virtual Lab for e-Science* (VL-e).
- Another research topic, where the WWU does research related to Roadmap **Task 3.3**, is *component adaptations*
  - Components should be adaptable, not only to requirements of the environment, e. g. , a hardware platform with special features or limitations, but they should also be adaptable to requirements of certain applications, e. g. , given data dependencies, which prevent the components from processing their input data in parallel using their inherent implementations. The WWU experiments with components that allow for restructuring (indexing) and re-ordering of their input. When components require a complex configuration (e.g., WSRF-compliant service interfaces with resource properties, RSL-format descriptions of subtasks for *Globus* GRAM-based scheduling etc.), adaptations allow reuse of the available configuration files, even if the procedures executed by the components are altered. The WWU is planning to contribute to a general description of the mechanisms necessary for Grid component adaptations, which should also be incorporated with the CoreGRID GCM specification.

8.1.5 UCHILE

UCHILE partner will contribute to the roadmap by participating in the activities related to Tasks 3.2 and 3.1.

In particular, developing applications to use Grid resources efficiently, working on resource allocation (distributed memory garbage collection) and considering Aspect Oriented Programming models to approach Grid programming. Also, a PhD

student is shared with INRIA working on algorithms for CPU allocation. We are also looking into the security aspects of the programming model.

8.1.6 EIA-FR

HES-SO/EIA-FR partner will contribute to the roadmap through the following activities:

- HES-SO/EIA-FR is in charge of Task 3.1 and will contribute to the activities related to the programming model by making available to the CoreGRID NoE the experience gained on parallel/distributed programming model and on communication mechanisms, as well as all tools developed in the context of the POP-C++ environment.
- participation in the activities related to Task 3.2, mainly by participating in the definition of a component model suitable for Grid programming in order to ensure that programming model defining in task 3.1 is suitable for the component model defined in Task 3.2.

8.1.7 UOW

The UoW partner expects to contribute to the Roadmap-2 in the following activities.

Participation in activities related to Task 3.2 and Task 3.3. The partner's aim is to establish the theoretical foundations for dynamic configuration and reconfiguration of components in a distributed system in an automated way based on the mathematically justified GCM and evaluated experimentally. This will include:

- formal definition of configuration and reconfiguration and adaptation of branching-time specification techniques to GCM;
- application of formal verification techniques to obtained specifications;
- development of theoretical foundations of a reasoning engine for multi-layer self-organizing systems, which will automatically manage reconfiguration of components in a system in a safe and optimal way.

8.1.8 INRIA

The INRIA partner expects to contribute to the roadmap through the following activities:

- participation in the activities related to the Task 3.1 – in particular, contributing to the communication between parallel components. The INRIA implementations of ProActive - Fractal distributed components and of GridCCM should also contribute to the programming model for the primitive component definition and implementation.
- participation in the activities related to Task 3.2 – in particular, relating to the hierarchical component model, capitalizing on the experience achieved with the Fractal and CCM component models. INRIA intends to contribute to primitive and hierarchical component definitions.
- participation in the activities related to the advanced programming environment, Task 3.3. Capitalizing on the experience gained with Dynaco and AFPAC frameworks, INRIA intends to contribute to dynamic self-adaptability of parallel component-based applications. INRIA also intends to study the implementation of autonomic controllers for GCM components.

8.1.9 UNIPASSAU

The WP3 work of UNIPASSAU is centered around three topics:

1. The research on the dynamic restructuring of Grid components has been aligned with the agenda of WP3 to make Fractal its reference model. In the second phase, experiments will be made with the Fractal implementation ProActive, which will lead to recommendation for the extension of the object model and its implementation.

2. The research on loop parallelization techniques in the polytope model has been awarded a 2-year renewable grant by the DFG. While this work is going on, a component interface is to be developed that brings it into CoreGRID.

3. A proposal for a national research action on metadata and metaprogramming for the Grid is pending with the DFG.

8.1.10 UNIPI

UNIPI expects to contribute to the roadmap through the following activities:

- participation in the activities related to the advanced programming environment Task 3.3, capitalizing on the experience gained in the development of the ASSIST high performance programming environment. In particular, UNIPI intends to contribute to the integration of structured programming techniques in the component framework and to the introduction of autonomic control capabilities

- participation in the activities related to Task 3.2, in particular, concerning the non-functional subsystems of the components

- participation in the activities of Task 3.1, and, in particular, contributing to the topics concerned with parallel components, programmed according to structured parallel programming models

- explicitly contributing to the security topic, in particular, contributing to the activities related to the specialization of security techniques to structured parallel/distributed, component-based programming environments.

8.1.11 UPC

UPC will contribute to the roadmap, in particular, to Task 3.3 developing semantic aspects of orchestration languages, specially those aspects related to the GRID Component programming models. More specifically, we will consider the aptness of certain approaches:

- development of a collection of GRID component examples based on ORC. Components will include probabilistic information about their behavior.

- a general approach to handling ORC expressions will be studied.

- the probabilistic aspects of ORC-based component models will be studied with a view to integration with other frameworks such as model logics.

- prototype implementations of GRID applications will be developed.

8.1.12 VUA

Partner VUA expects to contribute to the roadmap with the following activities:

- participation in the activities of the Task 3.1, in particular contributing to communication within parallel components, capitalizing on the experiences from the Ibis programming environment.
- participation in the activities of Task 3.3 on advanced programming models. In particular, VUA expects to contribute to adaptation, malleability, and fault-tolerance aspects of parallel, component-based applications.

9 Appendix A

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

- Table 1 lists the names of the CoreGRID partners involved in the institute activities along with the acronyms used to name the partners in the CoreGRID official documentation and in this roadmap document
- Table 2 presents some non-technical acronyms used throughout the document, along with their meanings.

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 Münster	University of Münster	Germany
UCAM	University of Cambridge	UK
UCHILE	University of Chile	Chile
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

Figure 1: Partners of the Programming Model Institute: 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 2: 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] M. Aldinucci and M. Danelutto. The cost of security in skeletal systems. Technical Report TR-06-03, Dept. Compute Science, University of Pisa, February 2006.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

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

[9] M. Carro, M. Hermenegildo, and D. Laforenza. Beyond Europe: Towards Trans-Continental Grid Exploitation, 2005. contribution to WP5 in the EU GridCoord Project, working in progress.

[10] CoreGRID-NESSI Joint Workshop, January 2006. Brussels.

[11] 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.

[12] 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.

[13] Dynaco framework. http://dynaco.irisa.fr/.

[14] 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.

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

[16] 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.

[17] 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.

[18] 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.

[19] K. Jeffery and D. De Roure *et al.* Future for European Grids: GRIDs and Service Oriented Knowledge Utilities - Vision and Research Directions 2010 and Beyond, January 2006. ftp://ftp.cordis.lu/ pub/ist/ docs/grids/ ngg3_eg_final.pdf.

[20] D. Laforenza. Grid Programming: Some Indications Where We Are Headed. *Parallel Computing*, 28(12):1701–1720, Dec. 2002.

[21] 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.

[22] J. Misra and W. R. Cook. Computation Orchestration: A Basis for Wide-Area Computing, 2006. To appear in Journal of Software and Systems Modeling.

[23] T. Nguyen and P. Kuonen. An Object-Oriented Framework for Efficient Data Access in Data Intensive Computing. In *in the proceeding of IPDPS 2003/Workshop on Advances in Parallel and Distributed Computational Models*, April 2003. Nice, France.

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

[25] C. Pèrez, T. Priol, and Andrè Ribes. PaCO++: A parallel object model for high performance distributed systems. In *Proceedings of the Distributed Object and Component-based Software Systems Minitrack in the Software Technology, Track of the 37th Hawaii International Conference on System Sciences (HICSS-37)*. IEEE Computer Society Press, January 2004. Big Island, Hawaii, USA.

[26] 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.

[27] D. Puppin, F. Silvestri, D. Laforenza, and S. Orlando. Toward GRIDLE: A Way to Build Grid Application Searching Through an Ecosystem of Components. In J. C. Cunha and O. F. Rana, editors, *Grid Computing: Software Environments and Tools*. Springer Verlag, 2006. ISBN: 1-85233-998-5.

[28] 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.

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

[30] 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.

[31] 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.

[32] 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.