Universitatea "Politechnica" din Bucureşti

Facultatea de Inginerie Electrica –CIEAC-LMN

Marie Curie Doctoral School in EE and CS – EST3 project

Spl. Independenţei 313, RO-060042, Bucureşti ROMANIA
http://www.lmn.pub.ro/ Tel/Fax: (40 21) 311 8004, (40 21) 316 95 71,
e-mail: lmn@lmn.pub.ro

Trends in Distributed Applications and Problem-Solving Environments

Author: Todor Nikolov

Date: July 2009

Supervisor: Prof. Daniel IOAN

Contents

ABSTRACT 3

1. Introduction 4
   1.1 Trends in Distributed Applications 6
   1.2 The Need for New Approaches 7
   1.3 Distributed, Parallel and Grid Computing 7

2. Problem-Solving Environments 10
   2.1 Problem-Solving Environments. 10
   2.2 Application Characteristics. 10

3. Influences of Grid Computing on Application Development 12
   3.1 Grid Computing Environments. 12
   3.2 Grid Applications and User Profiles. 13
   3.3 Dynamic Characteristics. 15

4. Challenges of Software Environments for Distributed Applications 16
   Component-based Models and Software Architectures. 16

5. Summary 19

Acknowledgements 20

Bibliography 21

ABSTRACT

As Grid computing technologies and infrastructures are being developed, suitable abstractions, methods, and tools will become necessary to enable application development, and software development of the components of Grid Computing Environments. Grid Computing will enable distributed applications with large numbers of involved components with dynamic interactions. This requires new approaches to understand and manage structure and behavior, and the diversity of interactions among system components.

This paper examines emerging trends in distributed applications on large scale and dynamic Grid computing infrastructures. These trends allow identifying the need to develop suitable software models, methods and tools for Grid Computing Environments, in order to help specify, compose, and develop dynamic distributed large scale applications.

Key words:

Distributed applications,
Grid software environments,
Problem Solving Environments

1. Introduction

In the past two decades, advances in parallel and distributed computing have enabled the development of advanced applications [35, 56]. Recent efforts have been centered around Grid Computing initiatives all over the world [1, 11, 15, 16, 26–28, 33, 39].

In this paper I look at the recent past of distributed and parallel computing, in order to try to understand trends of application development on integrated, heterogeneous large scale distributed environments.

However, it should be noted that such an intensive research effort and current interest on the Grid does not mean this is the only way to make applications reliable and robust to the heterogeneity and the unpredictable occurrences in distributed computing environments. Other approaches and technologies are bringing important contributions to address these challenges.

Distributed computing research has been addressing large scale distributed resource utilization and management issues for the past decade. However, there has been a gap between academy research and commercial and industrial practice, as far as distributed systems technology is concerned. For example, the success of the Web is partly due to its impact upon user driven interactions and its effective, even if limited, client-server organization.

On the other hand, fundamental research on distributed computing has been trying to find answers to challenging theoretical and technical problems, with important contributions, although with less immediate impact upon real applications, but this is changing with the emergence of new technologies.

The Grid is having a positive contribution in raising the concern for such distributed computing issues, such as providing clear and stronger guarantees for performance reliability, security, scalability, and consistency, which have not been considered in most commercial software, except for a small niche of critical application domains.

Furthermore, as the Grid initiatives emerge in relation to scientific applications, in order to enable complex problem solving, this may well contribute to promote user adoption of these technologies.

On one hand, it is expected that technologies, for example, based on the evolution of Web services, P2P computing, and Grid computing, will play complementary roles, concerning the proposal of new concepts and paradigms, and the development of new support environments and infrastructures.
On the other hand, it is not clear which technologies will exist or be dominant in the future. In the past, commercial and marketing issues, and technology transfer aspects, as well as the degree of user adoption, have been critical barriers to adopt new technologies and solutions.

The main motivation of this paper is that by looking at Grid Computing from a transparency-oriented perspective, we may better understand the main issues involved in the development of large scale distributed applications.

Transparency in this context refers to the capability to hide levels of complexity at particular points in the infrastructure hierarchy. A programmer, for instance, should not need to know precise details of computational hardware in order to use it. Here is maintained the "traditional" use of the term, although, some of the researchers pointed out that, if "Transparency" means to hide complexity of the underlying layers, then it should be called "Opaqueness" instead.

Transparency is a significant requirement for enabling existing Grid systems to work effectively. Being able to run application over distributed resources (both computational and data) remains a significant challenge for Grid computing systems. It is therefore necessary to couple this capability with environments which enable users to express their application demands, and manage and monitor the execution of the application subsequently.

Figure 1 provides a layered view of how Problem Solving Environments may interact with Grid Computing systems. From this perspective, Grid computing provides a way to aggregate access to a diverse range of different resources. Such systems must provide interfaces (perhaps as APIs), that enable access to different kinds of resource managers (such as Grid Engine, LSF, etc), and data managers (such as access to file systems, or structured data bases). Problem Solving Environments therefore act as mediation between user access Portals (generally problem specific), and resource management systems that enable the execution of tasks generated by the user via the Portal. Focus in this paper is to discuss the importance of this mediation, and the need for abstractions that can facilitate this.

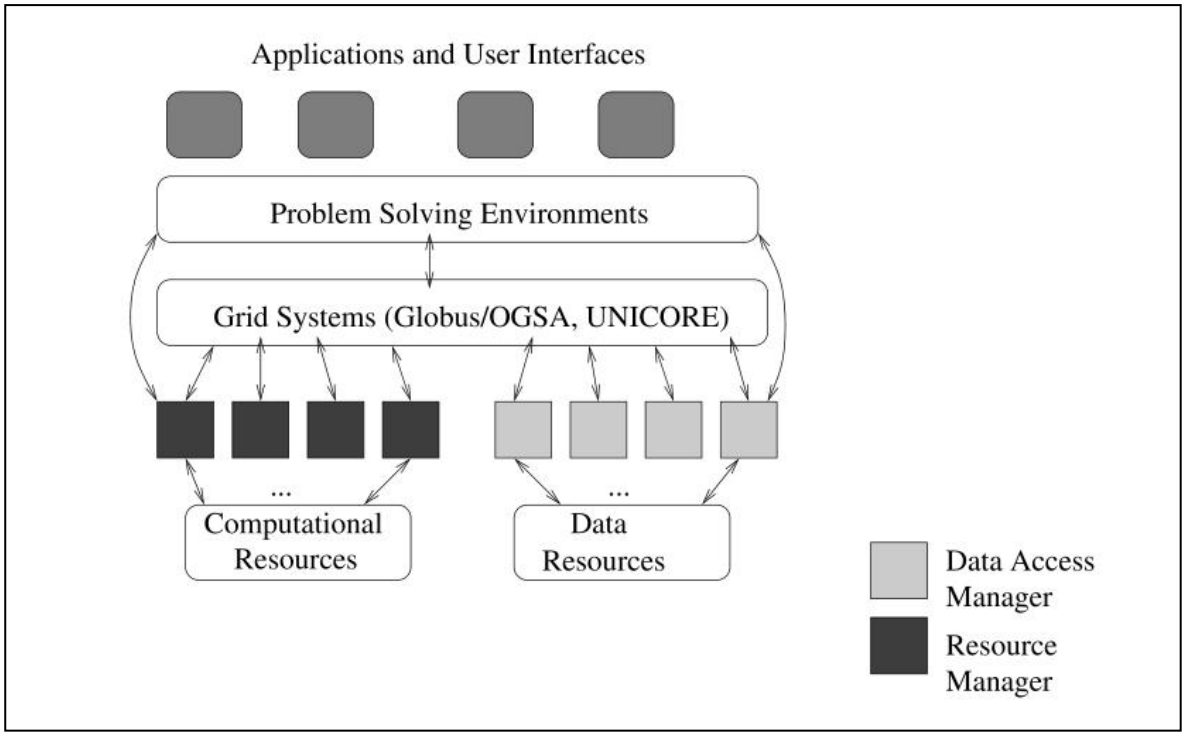


Fig. 1. Interaction between Problem Solving Environments and Grid Computing Systems

In the remaining of this section a summary of trends in distributed applications is presented in order to motivate the need for new approaches. Then a global view of distributed, parallel and grid computing is presented.

In the following sections, the main characteristics of parallel and distributed PSEs are surveyed in Section 2. In Section 3, Grid Computing Environments are discussed and their influences on application development. The challenges in the development of distributed applications for large scale dynamic computing environments are discussed in Section 4 and then some conclusions are presented.

1.1 Trends in Distributed Applications

In the past decades, there have been advances in architectures, systems, tools and environments, and programming models, languages and methods for parallel and distributed computing [35]. Such advances have increased the pressure to develop advanced applications in a wide range of domains, for instance, complex simulations in science and engineering, massive data archiving and searching in Bioinformatics or natural language processing, information retrieval and indexing mechanisms for search engines, to the dynamism in mobile multimedia and distributed agent systems. Such applications pose requirements beyond 'traditional' high-performance parallel computing systems:

- Higher degrees of user interaction, requiring increased flexibility in observation, control, or modification of application components.
- Intelligent advisory and assistance tools for the development and execution of different phases of an application life-cycle.
- Multidisciplinary applications, requiring interactions between distinct sub-models, and distributed user collaboration.
- The need to undertake computations that are not "regular", and therefore difficult to map easily to traditional parallel computing models.
- Dynamic nature of the applications and the environments, as new application components or system resources are dynamically generated or made unavailable, or due to mobility.
- Spatial distribution of application components and system resources, at small, medium or large scales.
- Increasing importance of connecting to distributed data resources (both structured and unstructured).

The above requirements should be met by development and execution support environments so that software developers and experts may have the desired degrees of flexibility in the design and implementation of advanced applications.

Recently, the above aspects have been increasingly influencing the academic and industrial communities, in their aims to exploit parallelism and distribution as a way to meet the functionalities and the performance requirements of many application classes [2, 5, 12–14, 21, 28, 37, 56, 72].

1.2 The Need for New Approaches

Grid computing brings several new dimensions, due to its intrinsic large scale capability of providing distributed, heterogeneous, and dynamic resources, spanning the boundaries of human organizations [26, 41]. However, a significant aspect of Grid computing is the intention to provide a unifying abstraction for the end-level user. This requires a major re-thinking of existing computational models, and their supporting tools and environments, and poses challenges that go beyond what has been previously addressed by parallel and distributed computing. Such challenges are not simply about being able to support interoperability between different resource management systems (although, this in itself is a commendable activity) – but also about providing suitable abstractions that could enable programming languages and tools to make more effective use of Grid systems [59].

New models and strategies are required for dynamic or adaptive resource scheduling [32], and their impact must be studied, with the additional difficulty that Grid applications are, themselves, being developed at the same time. New forms of systems organization and problem-solving strategies must be considered. Due to their openness, new approaches are needed to understand structure (topology, types of components) and behavior, and the interactions among their participants (security and trust). Due to the heterogeneity and time varying characteristics of their infrastructures, new performance issues have to be considered.

In the following sections, are examined in more detail why the above issues are relevant and the main challenges that they pose to the software development process. Outlined are, several research directions that I think may contribute to ease the above process and influence the design of future support environments, tools and infrastructures.

1.3 Distributed, Parallel and Grid Computing

Parallel and distributed computing in the 1980s and 90s had great influence upon application development. The improvements in computation and communication capabilities have enabled the creation of demanding applications in critical domains such as the Environment, Health, Aerospace, and other areas of Science and Technology. Similarly, new classes of applications are enabled by the new dimensions of heterogeneous large scale distributed systems which are becoming available nowadays.
In Figure 2 the joint influences of distributed, parallel and grid computing are sketched.

Parallel Computing Systems exploit a large diversity of computer architectures, from supercomputers, shared-memory or distributed-memory multiprocessors, to local networks and clusters of personal computers. Such diversity has contributed, to increasing the difficulties of parallel application development, in order to meet the correctness and performance specifications. Many of these difficulties still remain, although there were great improvements due to a long term research effort on models, support tools and environments [35].

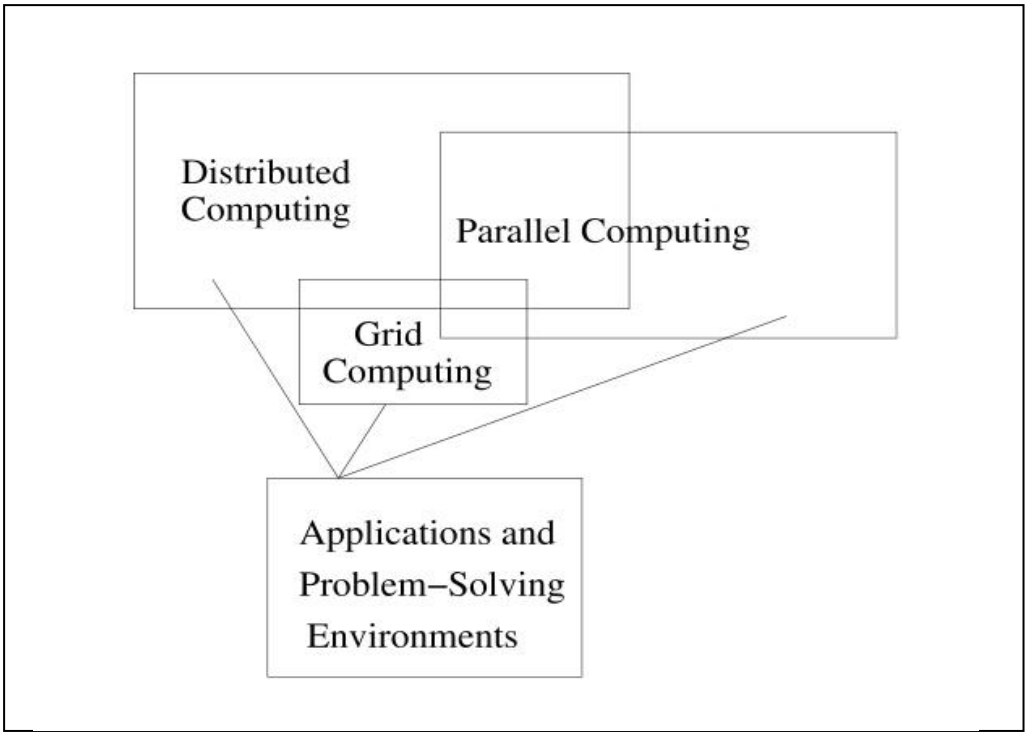


Fig. 2. Parallel and Distributed Computing

The emergence of large scale distributed and grid computing systems has opened new possibilities for the exploitation of large scale parallelism. This has enabled more ambitious applications, but it also brought new dimensions and difficulties to the process of application development as modern parallel and distributed platforms become more complex heterogeneous systems. These platforms include a diversity of subsystems, with varying performance behavior in their computation, storage and communication elements, with distinct computational models and with varying scales of distribution.

Distributed and Grid Computing Systems exploit several degrees of decentralization of data and control, in order to adapt to the application requirements. For instance, this adaptation can involve geographical distribution, availability and reliability, and abstractions for computation, communication, and storage.

The above requirements must be met with appropriate degrees of transparency, by hiding low level concerns and trying to provide only meaningful abstractions to the application layer. At each level of the layered architecture of a distributed computing system, there are design choices concerning the appropriate transparency degrees that must be provided to upper layers.

By looking at the recent past, one can observe that many design choices related to the offered transparency degrees, are typically revised as time passes and technology evolves. For example, support for failure transparency was greatly influenced by developments in hardware technologies, such as redundant disk arrays or replicated processing units.

Communication abstractions have also evolved, depending on the underlying support technologies. In distributed computing systems, communication models have evolved with increasing transparency degrees, from message-passing communication, to higher level RPC-based models, and to Distributed Shared Memory. The latter were made feasible due to

advances in communication infrastructures and to the evolution in the consistency management algorithms. Support for consistency in distributed process groups is also dependent on the available communication infrastructures and requires trade-offs between the supported transparency and the awareness to failure situations and to the system scale. The above abstractions need to be adjusted in order to fit the characteristics of applications and issues arising from the availability of Grid environments.

Transparency and awareness concepts evolve as new functionalities or supporting technologies become available. For example, mobile computing requires awareness to disconnected operation modes, as well as to location-sensitive information.
For example, consistency algorithms for managing distributed process groups need to be adapted for mobile computing environments, in order to consider location awareness issues. The semantics of the programming abstractions that are provided to the programmer must also be adjusted, in order to reflect the new transparency / awareness trade-offs.

Current research on Grid Computing reflects a revision of transparency concepts, aiming at more ambitious application and user profiles. Such revision was made possible because of the recent advances in communication and computation technologies, enabling the deployment of high speed networks, giving access to a large diversity of computational and data resources [38].

Also, by revising the degrees of transparency provided to the application layer, some of the inner layers of the underlying distributed system architecture must also be revised, in order to meet the application requirements. For example, Quality of Service (QoS) issues become of great importance in Grid environments due to the dynamic and uncertainty characteristics of such environments. In the context of Grid systems, QoS can also be defined at different levels – the Application QoS (A-QoS) relates to the user perception of a running program. The A-QoS is therefore particularly influenced by the type of Portal being used, and the interface between the Portal and the Grid System. To support real time interactions, for instance, it is necessary that A-QoS be maintained within tolerable bounds.
The next level of QoS relates to the Grid system itself, which I refer to as Middleware QoS (M-QoS), and related to the Grid system itself. Hence, the time required to spawn new tasks and manage their subsequent execution is governed by the M-QoS. This level of QoS must be measured between the Grid system and the one or more resource managers that are being used at any given instance.
The third level of QoS is the network or resource QoS (R-QoS) – and extensive work exists in the literature on measuring, monitoring, and in some cases, enforcing this. Network QoS (with parameters such as jitter, bandwidth, latency, delay) has been extensively investigated. Similarly, one can consider the existence of computational resource QoS, where parameters such as the number of CPUs, or minimum memory become significant.
However, to enable effective deployment of applications, it becomes necessary to aggregate QoS over all three of these levels. Intermediate software layers are required in order to provide the appropriate application semantics, for example, by enabling dynamic configuration and adaptive behavior capabilities in order to support such QoS.

2. Problem-Solving Environments

PSEs represent a significant approach to help manage the complexities of problem-solving in specific domains, as well as to provide integrated support and easy access to parallel and distributed resources. In this section, the main functionalities of a PSE are briefly surveyed.

2.1 Problem-Solving Environments.

Problem-Solving Environments (PSE) [45, 46, 52, 53, 73] are integrated environments for solving a class of related problems in an application domain. Typically they encapsulate state-of-the-art algorithms and problem solving strategies, in order to provide transparent and easy to use interfaces to an engineer or a scientist.

In the past decades, PSEs had a significant impact in many areas, ranging from fully developed environments in industrial applications, such as Automotive and Aerospace, to many applications in Science and Engineering. In such applications, PSEs have found major use, in the support of the design process, in the development of rapid prototypes, in studies on application behavior, and in decision support and control systems [85].

Recently, PSEs have been developed in other areas such as Education, Environment, Health, Finance, etc. This trend is creating a new profile of end-user, beyond the typical scientist and engineer. This is a significant development, as requirements for users in other application domains are now being incorporated within PSEs – thereby enabling the uptake of new development environments (such as Web Services and .NET), which were not traditionally part of the scientific computing genre.

2.2 Application Characteristics.

During the 1990s, applications in Science and Engineering became more complex and typically exhibit the following main characteristics [35]:

- They are based on complex models in a given problem domain, requiring computation intensive simulations.
- They must handle large volumes of input and generated data, involving difficult interpretation and classification.
- They require a high degree of user interaction, with offline or online processing modes, and scientific visualization, with distinct user interfaces, and often relying upon computational steering functionalities.
- They are multidisciplinary, combining multiple heterogeneous models, and possibly requiring the online collaboration of multiple users.

The above characteristics have influenced the design of applications and problem solving environments based on heterogeneous components, in order to allow:

- Access to a diversity of sequential, parallel or distributed problem solvers, supporting distinct types of simulators, mathematical packages, possibly based on legacy codes.
- Use of existing (third party) tools for data processing, interpretation and visualization.
- Online access to repositories of scientific data sets and databases.
- Interactive computational steering.

Such more complex requirements triggered a more complex cycle of user activities for application development, deployment and execution, when compared to older stand alone and monolithic PSEs, which were typically supported by a single process with a built-in user interface, running on a single workstation:

1) Problem specification.
2) Configuration of the environment (component selection for simulation, control, visualization).
3) Component activation and mapping.
4) Initial setup of simulation parameters.
5) Start of execution, possibly with monitoring, visualization and steering.
6) Analysis of intermediate and final results.

These requirements motivated the development of PSEs to exploit heterogeneous parallel and distributed resources, and trying to automate the above tasks as much as possible [29, 52].

The functionalities provided by a PSE can be divided in three main classes [85]:

- Support for problem specification.
- Support for resource management.
- Support for execution monitoring and control services.

Such functionalities are supported by Component frameworks (such as CORBA and Java Beans) for the integration of heterogeneous components into unified environments [30, 60, 61, 84]. Such frameworks also provide interfaces for the transparent access to distributed resources, and facilities for collaborative design and simulation. Often such frameworks are limited in their capability to support parallelism, and often have performance penalties not acceptable for scientific users.

However, in the late 1990s such Parallel and Distributed PSEs were typically mapped onto a parallel and distributed platform, e.g. based on PVM [17] or MPI [10], running on multiprocessors or local computer networks. Web-based interfaces and portals opened the way to support remote access to PSE and problem-solving in specific disciplines.
Overall, during the 90s, PSEs have played a significant role in supporting advances in Computational Science and Engineering [46].

When moving to Grid Computing Environments, it is expected that PSEs will play an even more significant role, due to the increased complexity of Grid applications and Grid Computing Environments.

3. Influences of Grid Computing on Application Development

In this section, I will examine Grid Computing Environments, and how they are enabling the development of advanced distributed applications, from a *user-oriented perspective*. I will describe the main characteristics of grid applications, and identify the main classes of applications and user profiles.

3.1 Grid Computing Environments.

The architecture of a Grid Computing System is composed of multiple layers, following the traditional architecture of a distributed computing system:

- User interfaces, applications and PSEs
- Development tools, programming models and environments
- Grid middleware: Resource management and scheduling; information registration and discovery; authentication and security; storage access, computation, and communication services
- Heterogeneous resources and network infrastructures

The main distinctive characteristics of the Grid is the goal to provide single unifying abstractions to the end-user, in order to allow transparent execution of highly demanding computations, user interaction in large scale virtual communities, and access to a diversity of computational and data resources and information repositories. Grids will enable 'heavy-weight' applications in Science and Engineering, based upon complex large scale distributed simulations with visualization and steering, access and analysis of large distributed datasets, and access to remote data sources and scientific instruments [6, 36, 39, 40, 50].

However, Grids Computing Environments are much more complex than the parallel and distributed computing environments of the 1990s. Such increased complexity is not only due to the diversity and heterogeneity of the physical support infrastructures, but also due to the critical dimensions of large scale distribution, the crossing of multiple administrative domains, increased security concerns, and the need to manage heterogeneous and dynamic resources. All of these aspects must be managed by Grid Computing Environments [8, 43, 44], in order to allow multidisciplinary and online collaboration among geographically distributed users, for example via Web interfaces and portals [19, 22, 24, 65–67, 79].

Many ongoing projects are developing sets of tools and software layers to provide easier access to Grid resources. A diversity of technologies and languages is being used to implement such tools [28, 49]. For example, Java, Python, Perl, CORBA, and their interfacing to the more basic Grid functionalities, as provided by the Globus toolkit. Web Services and XML [71] based technologies are also used to exploit integration or interoperability.

Significant efforts are also under way concerning the development of Web interfaces and portals, of a more generic applicability, possibly supported by Grid Portals development

middleware [66], or more application-specific and more related to the PSE concepts [57, 75, 79].

Such efforts are relevant in order to experiment with existing technologies and their possible extensions for the Grid. However, currently there is a large diversity and heterogeneity in those models and tools, and their implementations are not completely stable, standard or fully developed. In fact, even the lower levels, such as the Globus layer, are also evolving. This makes the development or experimentation with higher level abstractions, supporting the application layer, more difficult.

This concerns, for example, approaches for the development, organization and coordination of the application components, and for supporting application-specific problem solving strategies, resource discovery and scheduling, adaptive behavior, and access control policies.

How can we enable the experimentation and development of higher level application abstractions, for example, supporting specific concepts and tools for a given class of applications or problem domain, as well as more generic sets of high-level functionalities? This can be achieved by PSE layers and/or Intermediate Frameworks which provide more stable APIs and hide lower level concerns, but still allow extensions and incremental development [21, 43]:

- Higher level Application Abstractions
- PSE Layer
- Intermediate Frameworks
- Basic Grid Services

Several significant projects and working groups within the Global Grid Forum [15] are developing such kinds of intermediate layers [7, 9, 15, 25, 51, 81, 82]. For instance, OGSA (Open Grid Services Architecture), several 'Commodity Grid Kits', and research projects such as GrADs and GridLab. PSEs contribute to this goal [3, 18, 20, 36, 42, 54, 64, 74, 76].

Currently there is no uniform Grid computing model, although there are several well-identified needs for certain typical user tasks [43], for instance, running jobs, performing data management, composing workflows, online interactions for collaboration, etc.
In the following, are described several types of application and user profiles, which may help in identifying the requirements posed to the layers supporting high-level application abstractions.

3.2 Grid Applications and User Profiles.

Current efforts in the development and deployment of large scale distributed applications are not just academic exercises. There is a real pressure from research and industry to develop applications that require computation and information resources for bigger, longer experiments supporting more accurate models in specific areas. Easier and transparent access to remote resources is also required to increase the application and users' ability to analyze

and react in real-time. Increased levels of user interaction with the applications and among collaborative users is also required for increased productivity [34, 69, 83].

By looking at ongoing experiments on large scale applications, the following main profiles can be identified [21, 28, 77]:

1) **Computational Grids.** A single access point to a large-scale high-performance computing service, with transparent access to the underlying parallel and distributed servers.
2) **Scientific Data Grids.** Access to large scientific datasets, with optimized data transfers and interactions for data processing and manipulation.
3) **Virtual Organizations and Collaborative Virtual Spaces.** Access to virtual environments for resource sharing, online interaction and collaboration.
4) **Information, Knowledge and Semantic Grids.** Large and geographically distributed information repositories, made available for searching and data mining, and for intelligent knowledge management and decision support.

The above profiles are not mutually exclusive. In fact, they share several more generic goals. Access is globally unified through virtual layers:

- To solve new or larger problems by aggregating available resources.
- To access a large diversity of computation, data and information services.
- To enable coordinated resource sharing and collaboration across virtual environments.

There is a need to rely upon more uniform, standard and/or agreed upon interfaces for large scale computing environments, in order to allow the integration and cooperation of distinct services.

There is a need to support the concept of virtual resources. This is required to support the execution of experiments involving distributed computation and data, as well as virtual collaboration spaces. Access and management of persistent resources such as databases, catalogues, and archives is also required.

For example, in a range of distributed applications involving distinct functionalities, for example, Data Mining, Distributed Simulation, Visualization and Steering, Collaborative Mobile Multimedia, and/or Distributed Intelligent Agent Systems, one can find several common issues:

(i) large volumes of data (text or images), and efficient search, requiring parallel processing and parallel input/output;
(ii) dynamic, distributed, and mobile application entities, requiring appropriate structuring, interaction, and coordination abstractions;
(iii) integration of distributed heterogeneous components in a highly interactive environment, supporting dynamic reconfiguration of components, and execution at small or large scales;
(iv) organization, management, and coordination in a distributed agent system, requiring a dynamic system organization and intelligence within each agent. (further discussed in section 3.3).

It is a challenge to design and develop large scales integrated (development and execution) environments offering support for the above common functionalities. This is further motivated by current trends towards multidisciplinary applications and the convergence of several technologies, such as Web Services, P2P Computing, Grid Computing, Mobile and Ubiquitous Computing [48].

3.3 Dynamic Characteristics.

In existing parallel and distributed systems, changes in the configuration and availability of resources, variations of their characteristics and behavior, have already motivated multiple approaches to address dynamic system reconfiguration, for example, in systems that support fault tolerance, dynamic task spawning and load balancing.

Such dynamic changes are intrinsic characteristics of large scale distributed computing systems.

On application initiation, contract negotiation schemes will guide intermediate agents, acting as planners/brokers/schedulers on behalf of the application resource requirements. During execution, there is a need to provide new problem-solving strategies with adaptive behavior. Intermediate layers of the Grid Computing Environment must be aware to Quality of Service factors, and be able to detect changes in the corresponding system behavior. Dynamic revision of the initially formulated agent plans must be supported, and appropriate interfaces must be devised to support the interaction with the monitoring and runtime reconfiguration functionalities.

Dynamic characteristics may also be required by the applications or problem-solving environments. For example, collaboration models may be supported by group oriented abstractions, where dynamic changes may correspond to users entering or leaving a group. Such abstractions also require awareness to factors related to current group members availability or connectivity, for example, in a mobile computing environment.

Online experiments or simulations may require a change to a distinct operation mode, e.g. offline or online, as well as usage of a distinct tool. An ongoing experiment may need to be re-configured on-the-fly, by including new components or tools (e.g. for visualization). Also, multiple users or agents may concurrently join ongoing experiments, possibly assuming distinct roles (observation, steering), or requiring distinct views.

In general, in future distributed computing environments, mobility of users, agents, and devices, pose a clear requirement to provide support for dynamism, in several layers of the computing system hierarchy. This requirement encompasses the application and PSE level, the intermediate frameworks supporting tools and environments, down to the middleware and execution support layers.

4. Challenges of Software Environments for Distributed Applications

In this section, are examined the main challenges to the development of distributed applications, from a *software development perspective*. Examined are issues and approaches to face those challenges.

The development of infrastructures to support Grid Computing Environments and their basic functionalities is a significant goal that is being pursued by many national and international projects. However, many issues of providing support for application development over such infrastructures, as well as facilities to help software developers in building and integrating software components are still left open.
This is a significant challenge to enable effective application and software development for Grid Computing Environments [8, 15, 58].

Component-based Models and Software Architectures.

Component-based models [78] provide effective ways to develop Grid applications and PSEs, as they allow varying degrees of complexity and granularity, and facilitate the access to wrapped software packages [4, 23, 30, 60, 61, 84]. Component models allow a clear separation between computation and interaction. This is a significant aspect to model the heterogeneity of Grid application components, their complex interactions, and the need for their dynamic modification and reconfiguration [55, 75, 80].

New abstractions and models are required to build and manage large scale dynamic organizations of components, for example, allowing manipulation of individual components or groups of components. Facilities for component integration, reuse and dynamic composition, and coordination of component interactions must also be revised to address Grid application characteristics.

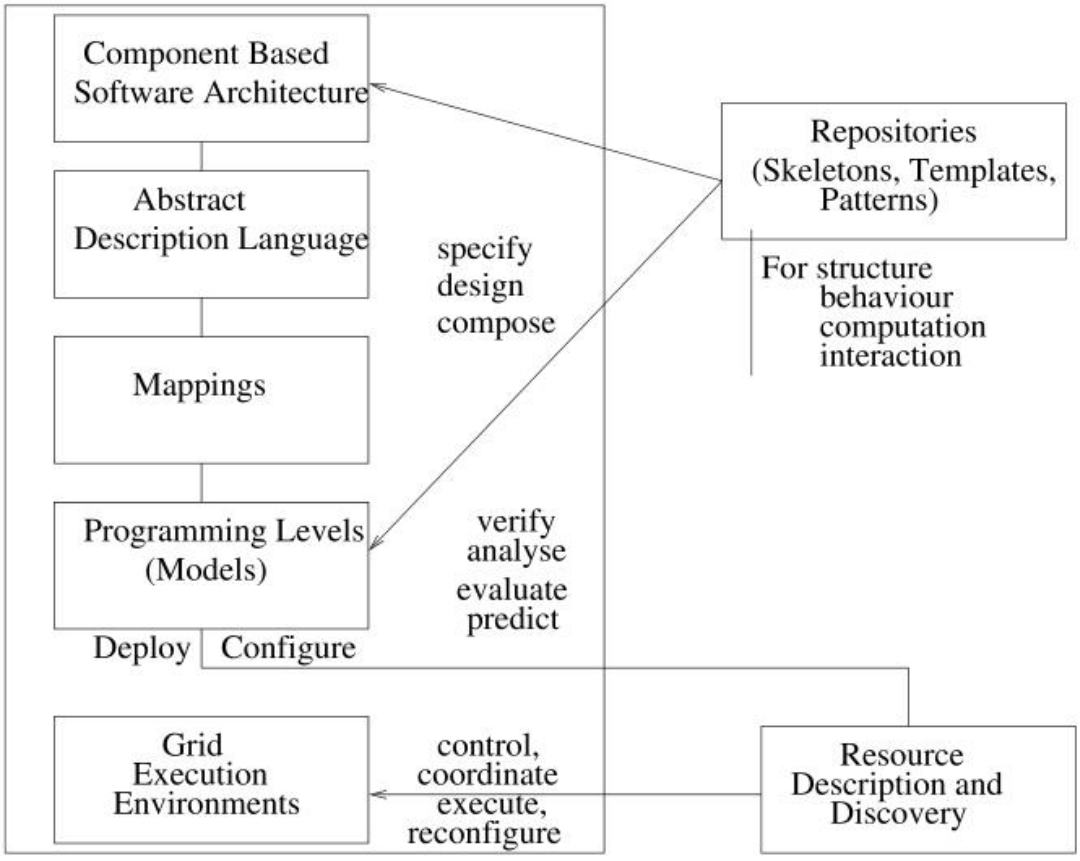
Support for modeling and reasoning on system structure and behavior is required in order to evaluate and predict global properties of an applications, concerning its functional and non-functional aspects.

There are already several approaches to model composition and workflow abstractions using components [18, 62, 63, 84]. Their integration into visual programming and development environments facilitate the graphical design of a Grid application.
Such environments can be complemented by other development support tools, for example, for testing and debugging, and for visualization and steering, simulation and evaluation tools.

Appropriate levels of flexibility must be provided at all stages of the software lifecycle, from application specification and design, including program transformation and refinement, simulation, evaluation and prediction of behavior, code generation, configuration and deployment. The following issues should be considered when designing the software development and execution support environments:

- A clear separation of computation and interaction (coordination) issues, and structural and behavioral properties of an application.
- The specification of an application in terms of multiple components, still enabling alternative mappings onto the underlying layers, i.e. with varying degrees of automated processing, and possibly targeted at distinct programming languages and models.
- Ways to bridge the gap between the higher layers (such as programming models and languages) and the underlying cluster/Grid platforms, so that application configuration and deployment can be adjusted according to the dynamic changes in the available resources.
- Support for coordination of distributed entities, with adaptability and dynamic reconfiguration.
- Support for infrastructure services – such as event management, naming, transaction management, that traverse different Grid systems. Providing a common way to name objects generated from a program that must be distributed across different Grid systems remains a difficult challenge.

Abstract view of the main elements involved in the software lifecycle is sketched in Figure 3. This figure does not aim to propose a new method to model the software lifecycle activities. It only aims to illustrate the relationships among the above mentioned issues.



Fig

Ter nd
cor ly
use es
and by
ena nd
inte ith
rele m
des of
aut

The ng
pla n.
Thi ry
ser

Above concerns should drive further research on several layers of the software hierarchy [29]:

- Software architecture models and specification languages for composition and component interaction, and coordination models.
- Organization models for managing large scale dynamic applications.
- Resource management, discovery, and scheduling; execution monitoring and control; runtime support for dynamic reconfiguration on heterogeneous hardware/software platforms.

For each layer, there is a need to design appropriate abstractions, models and tools, in order to enable the development of future generations of applications.

The presentation of the above issues aims at giving a global view of what is involved in the development of future generations distributed applications. Of course, for each case and user profile, there will be multiple alternative ways to meet the application needs and requirements.

5. Summary

As Grid computing technologies and infrastructures are being developed, the availability of suitable abstractions, methods, and tools will become necessary to enable application development, and software development of the components of Grid Computing Environments.

There is a challenge to develop software engineering techniques for Grid Computing Environments, in order to help specify, compose, and develop dynamic distributed large scale applications.

Future distributed applications on Grid Computing Environments will tend to exhibit higher degrees of user interaction, with increased flexibility in observation and control of application components. Multidisciplinary and online collaboration involving distributed users will be more frequent, as well as dynamic applications and environments, with new application components or system resources being dynamically generated, made unavailable, or mobile.

Awareness to the spatial distribution of application components and system resources, and their organizations at small, medium or large scales, will be an increasingly significant concern.

Distributed Problem-Solving Environments are expected to play a significant role in bridging the gap from the application level concepts to the complexity of the emerging Grid computing platforms.

New models, support tools and environments are required for component integration, for flexible mappings between software levels, for predicting and evaluating global application and system properties, for coordination of dynamic distributed applications, and for adaptive application behavior.

Current initiatives in Grid computing technologies and infrastructures are opening the way to enable the research and development of the software models, methods and tools that will integrate future generations of software environments.

Acknowledgements:

I’d like to thank for the scientific support I’ve received from **Prof. Daniel IOAN**, www.lmn.pub.ro/~daniel, my research supervisor at LMN

The work was financed by **Marie Curie EST3 project** http://est3.lmn.pub.ro/

Bibliography

[1] ASCI DoE Advanced Simulation and Computing Program. http://www.lanl.gov/asci.
[2] Biomedical informatics research network BIRN grid. http://www.nbirn.net.
[3] Cactus Grid Computational Toolkit. http://www.globus.org.
[4] Common Component Architecture. http://www.cca-forum.org.
[5] European grid application toolkit and test bed. http://www.gridlab.org.
[6] Globus Grid Project. http://www.globus.org.
[7] GrADS grid application development software project. http://nhse2.cs.rice.edu/grads.
[8] Grid Computing Environments Working Group. http://www.computingportals.org.
[9] The GridLab project. http://www.gridlab.org.
[10] Message Passing Interface. http://www.mpi-forum.org.
[11] NASA information power grid. http://ipg.nasa.gov.
[12] National collaboratories: Applying information technology for scientific research. Washington, D.C.: National Academy Press, 1993, http://www.nap.edu/books.
[13] Particle physics data grid. http://www.ppdg.net.
[14] SETI@Home Internet Computing. http://setiathome.ssl.berkeley.edu.
[15] The Global Grid Forum. http://www.gridforum.org.
[16] UK e-Science. http://www.escience-grid.org.uk.
[17] Parallel Virtual Machine. http://www.csm.ornl.gov/pvm/pvm.home.html, 2003.
[18] Triana. http://www.triana.co.uk, 2003.
[19] Sudesh Agrawal, Jack Dongarra, Keith Seymour, and Sathish Vadhiyar. NetSolve:Past, present and future: a look at a grid enabled solver. In Grid Computing: Makingthe Global Infrastructure a Reality. John Wiley and Sons, 2003.
[20] G. Allen, T. Dramlitsch, I. Foster, N.T. Karonis, M. Ripanu, E. Seidel, and B. Toonen. Supporting efficient execution in heterogeneous distributed computing environment with Cactus and Globus. In Proceedings Super Computing 2001, Denver, USA,November 2001.
[21] G. Allen, T. Goodale, M. Russell, E. Seidel, and J. Shalf. Classifying and enabling Grid applications. In Grid Computing: Making the Global Infrastructure a Reality.Wiley, 2003.
[22] A. Aloisio and M. Cafaro. Web-based access to the Grid using the Grid Resource Broker. Concurrency and Computation: Practice and Experience, 14(13-15):1145–1160, 2002.
[23] R. Armstrong, D. Gannon, A. Geist, K. Keahey, and L. Mcinnes S. Kohn, andS. Parker. Toward a common component architecture for high performance scientificcomputing. In Proceedings of the 8th IEEE Symp. on High Performance DistributedComputing, 1999.
[24] D. Arnold, H. Casanova, and J. Dongarra. Innovations of the NetSolve GridComputing System. Concurrency and Computation: Practice and Experience, 14(13-15):1457–1479, 2002.
[25] M. Atkinson. Rationale for choosing the Open Grid Services Architecture. In Grid Computing: Making the Global Infrastructure a Reality. Wiley, 2003.
[26] M. Baker, R. Buyya, and D. Laforenza. Grids and Grid technologies for wide–area distributed computing. Software Practice and Experience, 32(15):1437–1466,December 2002.
[27] F. Berman, G. Fox, and T. Hey. The Grid: Past, Present and Future. In Grid Computing: Making the Global Infrastructure a Reality. Wiley, 2003.
[28] F. Berman, G.C. Fox, and A.J.G. Hey, editors. Grid Computing: Making the Global Infrastructure a Reality. Wiley, 2003.
[29] R.F. Boisvert and P.T.P. Tang, editors. The Architecture of Scientific Software. Kluwer Academic Publishers, 2001.
[30] R. Bramley, D. Gannon, T. Stuckey, J. Villacis, J. Balasubramanian, E. Akman,F. Breg, S. Diwan, and M. Govindaraju. Component architectures for distributed scientific problem solving. IEEE Computational Science and Engineering, 5(2):50–63, 1998.
[31] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. Pattern-Oriented Software Architecture: A System of Patterns. Wiley, 1998.
[32] R. Buyya. Economic-based Distributed Resource Management and Scheduling for Grid Computing. PhD thesis, School of Computer Science and Software Engineering, Monash University, Melbourne, Australia, 2002.
[33] M. Chetty and R. Buyya. Weaving computational grids: How analogous are they with electrical grids. IEEE Computing in Science and Engineering, pages 61–71, jul-aug 2002.

[34] J.A. Clarke and R.R. Namburu. A distributed computing environment for interdisciplinary applications. Concurrency and Computation: Practice and Experience, 14(13-15):1161–1174, 2002.
[35] J. Dongarra, I. Foster, G. Fox, W. Gropp.and K. Kennedy, L. Torczon, and A. White, editors. Sourcebook of Parallel Computing. Morgan Kaufmann, 2003.
[36] Dietmar W. Erwin. UNICORE- a grid computing environment. Concurrency and Computation: Practice and Experience, 14(13-15):1395–1410, 2002.
[37] S.C. Farantos, S. Stamatis, N. Nellari, and D. Maric. A joint scientific and technology activity and study on grid enabling technology. Technical report, ENACTS, December 2002.
[38] I. Foster. The physiology of the grid: An open grid services architecture for distributed systems integration. Technical report, Argonne National Laboratory, IL, 2002. [39] I. Foster and C. Kesselman, editors. The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, 1999.
[40] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the Grid: Enabling scalable virtual organizations. International Journal of Supercomputing Applications, 15(3), 2001.
[41] Ian Foster. The grid: A new infrastructure for the 21st century. In Grid Computing: Making the Global Infrastructure a Reality. Wiley, 2003.
[42] G. Fox, W. Furmanski, T. Haupt, and S. Klasky. Web flow: A visual problem solving environment for wide-area, 1996. NPAC Proposal to the NSF New Technologies Program.
[43] G. Fox, D. Gannon, and M. Thomas. A summary of grid computing environments. Concurrency and Computation: Practice and Experience, 14(13-15):1035–1044, nov- dec 2002.
[44] Geoffrey Fox, Dennis Gannon, and Mary Thomas. Overview of grid computing environments. Technical report, Global Forum International, 2003. http://www.gridforum.org/.
[45] E. Gallopoulos, E.N. Houstis, and J.R. Rice. Computer as thinker/doer: Problem- solving environments for computational science. IEEE Computational Science and Engineering, 1(2):11–23, 1994.
[46] E. Gallopoulos, E.N. Houstis, and J.R. Rice. Workshop on problem-solving environments: Findings and recommendations. ACM Computing Surveys, 27(2):277– 279, 1995.
[47] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1994.
[48] D. Gannon, R. Bramley, G. Fox, S. Smallen, A. Rossi, R. Ananthakrishnan, F. Bertrand, K. Chiu, M. Farrellee, M. Govindaraju, S. Krishnan, L. Ramakrishnan, Y. Simmhan, A. Slominski, Y. Ma, C. Olariu, and N. Rey-Cenvaz. Programming the grid: Distributed software components, P2P, and Grid Web Services for scientific applications. In Proceedings of Grid 2001, 2001.
[49] Steve Graham, Simeon Simeonov, Toufic Boubez, Glen Daniels, Doug Davis, Yuichi Nakamura, and Ryo Neyama. Building Web Services with Java: Making Sense of XML, SOAP, WSDL, and UDDI. Sams Publishing, Indianapolis, IN, 2001.
[50] A.S. Grimshaw, A. Natrajan, M.A. Humphrey, M.A. Lewis, A. Nguyen-Tuong, J.F. Karpovich, M.M. Morgan, and A.F. Ferrari. From Legion to Avaki: The persistence of vision. In Grid Computing: Making the Global Infrastructure a Reality. Wiley, 2003.
[51] A. Hoheisel. Fraunhofer resource grid – grid application definition language. Technical report, Global Grid Forum, 2002.
[52] E. Houstis, J.R. Rice, E. Gallopoulos, and R. Bramley, editors. Enabling Technologies for Computational Science: Frameworks, Middleware, and Environment. Kluwer Academic Publishers, 2000.
[53] E.N. Houstis, A.C. Catlin, N. Dhanjani, J.R. Rice, N. Ramakrishnan, and V. Verykios. MyPITHIA:a recommendation portal for scientific software and services. Concurrency and Computation: Practice and Experience, 14(13-15):1481–1505, 2002.
[54] K.A. Iskra, R.G. Belleman, G.D. vanAlbada, J. Santoso, P.M.A. Sloot, H.E. Bal, H. J.W. Spoelder, and M. Bubak. The Polder Computing Environment: a system for interactive distributed simulation. Concurrency and Computation: Practice and Experience, 14(13-15):1313–1335, 2002.
[55] C. Johnson, S. Parker, and D. Weinstein. Large-scale computational science applications using the SCIrun problem-solving environment. In Proceedings of Supercomputing 2000, 2000.
[56] A. E. Koniges, editor. Industrial Strength Parallel Computing. Morgan Kaufmann, 2000.
[57] S. Krishnan, R. Bramley, M. Govindaraju, R. Indurkar, A. Slomski, D. Gannon, J. Alameda, and D. Alkaire. The XCAT science portal. In Proceedings SuperComputing 2001, Denver, USA, November 2001.
[58] C. Lee, S. Matsuoka, D. Talia, A. Sussman, N. Karonis, G. Allen, and M.Thomas. A grid programming primer. In Programming Models Working Group, Global Grid Forum 1, Amsterdam, March 2001.
[59] C. Lee and D. Talia. Grid programming models: current tools, issues and directions. In Grid Computing: Making the Global Infrastructure a Reality. Wiley, 2003.
[60] Maozhen Li, Omer F.Rana, and David Walker. Wrapping mpi-based legacy codes as Java/CORBA components. Future Generation Computer Systems, 18(2):213–223, October 2001.

[61] Maozhen Li, Omer F. Rana, David Walker, Matthew Shields, and Yan Huang. Component-based problem-solving environments for computational science. Technical report, Department of Computer Science, University of Cardiff, UK.
[62] Dan A. Marinescu. A grid workflow management architecture. Technical report, Global Grid Forum Working Document (submitted). School of Electrical and Computer Engineering, University of Central Florida, Orlando, USA, 2002.
[63] Dan A. Marinescu. Internet Based Workflow Management: Towards a Semantic Web. Wiley, 2002.
[64] J. Michopoulos, P. Tsompanopoulou, E. Houstis, J. Rice, C. Farhat, M. Lesoinne, and F. Lechenault. DDEMA: A data-driven environment for multiphysics applications. In Proceedings of ICCS 2003 International Conference on Computational Science 2003, Lecture Notes in Computer Science. Springer Verlag, 2003.
[65] A. Natrajan, A. Nguyen-Tuong, M.A. Humphrey, M. Herrick, B.P. Clarke, and A.S. Grimshaw. The Legion Grid Portal. Concurrency and Computation: Practice and Experience, 14(13-15):1365–1394, 2002.
[66] J. Novotny. The Grid Portal Development Kit. Concurrency and Computation: Practice and Experience, 14(13-15):1129–1144, 2002.
[67] M.E. Pierce, C. Youn, and G.C. Fox. The Gateway computational Web portal. Concurrency and Computation: Practice and Experience, 14(13-15):1411–1426, 2002.
[68] F.A. Rabhi and S. Gorlatch, editors. Patterns and Skeletons for Parallel and Distributed Computing. Springer Verlag, 2003.
[69] N. Ramakrishnan, L.T. Watson, D.G. Kafura, C.J. Ribbens, and C.A. Shaffer. Programming environments for multidisciplinary Grid communities. Concurrency and Computation: Practice and Experience, 14(13-15):1241–1173, 2002.
[70] O.F. Rana and D.W. Walker. Service Design Patterns for Computational Grids. In Patterns and Skeletons for Parallel and Distributed Computing. Springer-Verlag, 2003.
[71] Omer F. Rana, David Walker, Maozhen Li, and Matthew Shields. An XML based component model for generating scientific applications and performing large scale simulations in a meta-computing environment. In Proceedings of Generative Component Based Software Engineering, Erfert, Germany, 1999.
[72] R. Rheinheimer, J.I. Beiriger, H.P. Bivens, and S.L. Humphreys. The ASCI Computational Grid: Initial deployment. Concurrency and Computation: Practice and Experience, 14(13-15):1351–1363, 2002.
[73] J.R. Rice and R.F. Boisvert. From scientific software libraries to problem-solving environments. IEEE Computational Science and Engineering, 3(3):44–53, 1996.
[74] A. Schreiber. The integrated simulation environment TENT. Concurrency and Computation: Practice and Experience, 14(13-15):1553–1568, 2002.
[75] K. Schuchardt, B. Didier, and G. Ecce. Ecce - a problem solving environment's evolution towards Grid services and a Web architecture. Concurrency and Computation: Practice and Experience, 14(13-15):1221–1139, 2002.
[76] Matthew S. Shields, Omer Rana, David W. Walker, Maozhen Li, and David Golby. A Java/CORBA-based visual program composition environment for PSEs. Concurrency - Practice and Experience, 12(8):687–704, 2000.
[77] D.B. Skillicorn. Motivating computational grids. Technical report, Department of Computing and Information Science, Queen's University, Ontario, Canada, November 2001.
[78] C. Szyperski, editor. Component Software: Beyond Object-oriented Programming. Addison-Wesley, 1999.
[79] M. Thomas, M. Dahan, K. Mueller, S. Mocka, C. Mills, and R. Regno. Application portals: Practice and experience. Concurrency and Computation: Practice and Experience, 14(13-15):1427–1433, 2002.
[80] J. Villacis, M. Govindaraju, D. Stern, A. Whitaker, F. Berg, P. Deuskar, B. Temko, D. Gannon, and R. Bramley. CAT: A high performance, distributed component architecture toolkit for the grid. In Proceedings of the 8th IEEE Symp. on High Performance Distributed Computing, 1999.
[81] G. von Laszewski, J. Gawor, S. Krishnan, and K. Jackson. Grid kits – middleware for building Grid environments. In Grid Computing: Making the Global Infrastructure a Reality. Wiley, 2003.
[82] G. von Laszewski, J. Gawor, P. Lane, N. Rehn, and M. Russel. Features of the Java Commodity Grid Kit. Concurrency and Computation: Practice and Experience, 14(13-15):1045–1055, 2002.
[83] G. von Laszewski, M. Russell, I. Foster, J. Shalf, G. Allen, G. Daues, J. Novotny, and E. Seidel. Community software development with the Astrophysics Simulation Collaboratory. Concurrency and Computation: Practice and Experience, 14(13-15):1289–1301, 2002.
[84] D. Walker, M. Li, O.F. Rana, M.S. Shields, and Y. Huang. The software architecture of a distributed problem-solving environment. Concurrency: Practice and Experience, 12(15):1455–1480, December 2000.
[85] David W. Walker. Emerging distributed computing technologies. Technical report, Department of Computer Science, Cardiff University, UK, 2001.
http://www.cs.cf.ac.uk/User/David.W.Walker/IGDS/GridCourse.htm.