FSE Grid Middleware: Collaborative Grid Environment for Distributed Computing

● Roger Henri ● Pierre Lagier ● Didier Plaindoux

(Manuscript received June 2, 2004)

While traditional distributed computing technologies provide critical building blocks and frameworks for Grid application development, distributed computing is not concerned with and does not address the following: large-scale and dynamic resource sharing, frequently stringent performance requirements, large resource needs, and the multidisciplinary nature of Grid applications. The goal of the FSE Grid middleware project is to develop the knowledge and technology base required to support application execution in a collaborative environment for distributed computing along with application development strategies to make FSE Grid middleware accessible to ordinary scientists, engineers, and software developers for problem solving.

1. Introduction

Despite the tremendous potential associated with distributed and Grid computing paradigms, the dynamic and complex nature of these environments as well as the sophistication of a new generation of applications leads to many daunting challenges.[1),2)] Issues such as global resource management, distributed job scheduling, data locality, security, multiple site policies, and a variety of software environments are but a few of the complexities involved when distributed computer systems are linked together as a common resource. Few software tools exist today[3)] to meet the challenges of these working environments, and the suitability of the middleware, which is available for a broad class of applications, remains largely unknown.

Quite simply, the goal of the FSE Grid middleware[note)] project is to build a software environment that overcomes these challenges and makes the use of distributed or "Grid-like" systems usable on a daily basis. To achieve this, FSE Grid middleware has introduced an innovative way of utilizing the global information infrastructure as a platform for computation, data sharing, and collaboration to make distributed computing usable, comfortable, and effective.

FSE Grid middleware is a collaborative infrastructure built on top of a software agent society[4)] to provide users with a scalable service environment. It introduces the concept of a "computing community" between users, applications, and computers. FSE Grid middleware includes software development facilities, seamless access to networked computers, process integration through complex task definition, execution with a workflow[5)] engine, and an open job process management architecture that can adapt itself to any available back-end system.

2. User-centric approach

FSE Grid middleware software is built for end-users. It has a user-centric nature that hides the hardware and network complexities most

note) This is not product name.
FSE (Fujitsu Systems Europe Limited) will fix product name of "FSE Grid middleware" soon.

commonly attributed to today's computing environments from the user.

Users can run jobs on a multitude of hardware platforms without having to worry about which machine or architecture is needed for each execution step or where and how data is moved from its current location to the systems involved in the execution sequence. Even the complexities regarding how to interface with batch subsystems and other system services are made transparent to the user.

FSE Grid middleware can be applied to a variety of environments, including the linking of local and remote computing resources. With distributed production environments in which systems are scattered over geographically separated locations, FSE Grid middleware transparently manages the resources of the participating systems. It allows a user to submit jobs on a network of hardware systems and have their results (and output data if necessary) returned to the local machine on which they are working. This is achieved through a single user interface that publicises the available services in the FSE Grid middleware network. The user simply provides the location of the data required by the service and then requests that the service be started through the user GUI. FSE Grid middleware transparently moves data between computers as required (**Figure 1**).

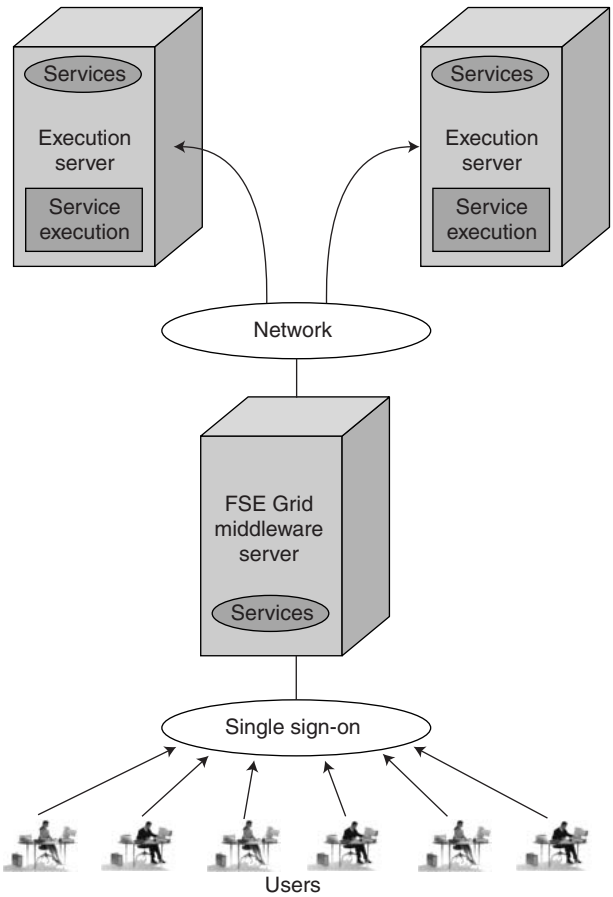


Figure 1
FSE Grid middleware usage.

3. FSE Grid middleware concept

FSE Grid middleware design started from the definition of the key concepts it relies on (**Table 1**).

The entire FSE Grid middleware framework design is based on object-oriented technology, and each defined concept is related to a class of object to manipulate through networks. As a natural consequence, the Java programming language was chosen for the implementation.

Within FSE Grid middleware, applications

Table 1
FSE Grid middleware concept hierarchy.

Concept	Description
Service	A collection of actions that apply to a common entity. The entity can be a user application or system command that can be used through FSE Grid middleware.
Action	An activity that is part of a service. For example, a service may provide actions like: • Run an application. • List the files on an execution machine. • Monitor an application.
Task	An instantiation of a service/action, usually the action "run" of a service. When a service action is added to a workflow, we designate this as a workflow task or simply a task.
Workflow	A framework for managing a sequence of tasks.
Workspace	The area where users store their workflow and related data.
Project	A logical container for holding items related to a specific workflow or group of workflows (e.g., workflow definitions and input/output files).

are utilized through what is known as a service definition. Each service definition describes how an application can be used on a particular system and what actions can take place for it. Services and actions related to services are published on the FSE Grid middleware community, and FSE Grid middleware clients see them automatically. The FSE Grid middleware administrator can update or remove services and actions for a user or from the network dynamically without the user's involvement.

FSE Grid middleware can also be used to build a workflow of multiple applications that are described as a set or a graph of tasks. Such a graph can have cycles, and each single task is defined with "join" and "split" conditions as well as "fetch in" and "fetch out" operations for files. The workflow module allows applications to be coupled through their files. Input and output files are transparently transferred across platforms according to constraints in order to run several applications one after the other (**Figure 2**).

The user is provided with a specific file explorer that gives a single view of all available file storage areas. These areas can be on the local user machine, on the execution machines, or on the server side. This logical view defines a workspace that belongs to a user and is eventually shared with other users. The user can freely access, add, move, or delete files within a logical workspace, regardless of the physical location of the aggregated storage areas.

The workspace is organized in logical containers called projects, each of which being a logical view of a subset of files, including a specific workflow or a group of workflows and all related files. The workspace is normally organized in a tree structure with sub-directories.

Finally, FSE Grid middleware can be applied to the provision of application services. It allows a company to provide their users and clients with seamless access to high-performance computing.

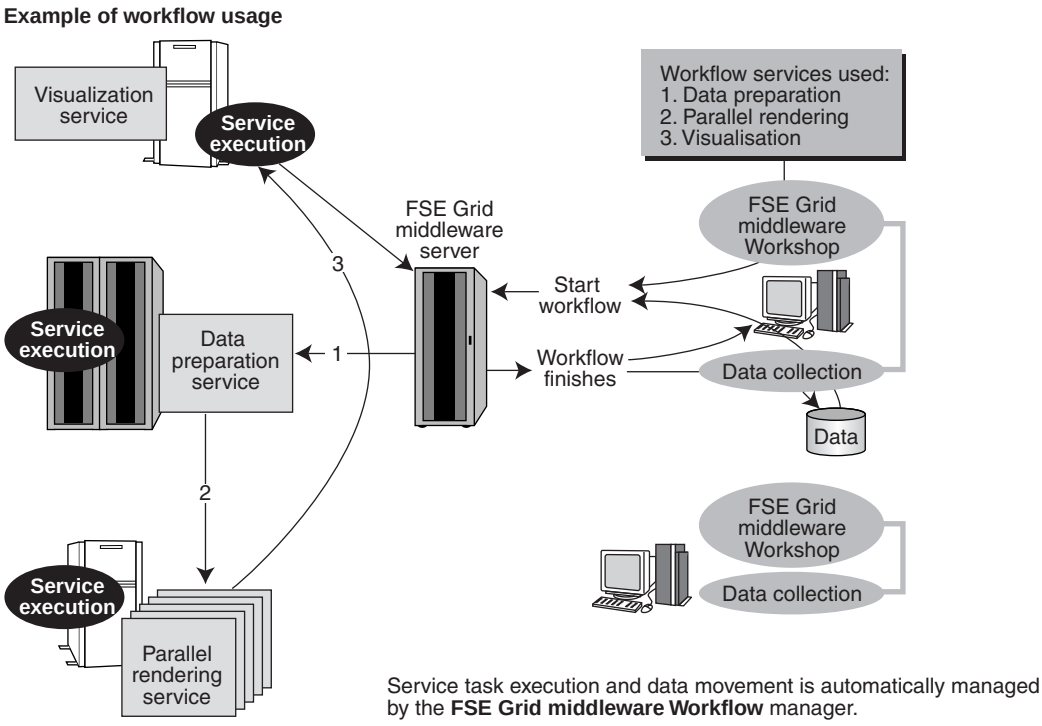


Figure 2
Workflow example.

4. Architecture

The effective usage and management of heterogeneous networked computer resources presents some very difficult challenges; for example:

1) The environment is highly dynamic and changes occur often and rapidly.
2) The execution environment is complex, especially when executing complex tasks.
3) The entire system is stable by itself, but independent parts such as the network, individual machines, and available services are volatile.

The most efficient way to design a framework for such an environment is to build an adaptive, self-configuring collection of distributed heterogeneous autonomous agents that interact with humans and each other over networks in real-time in a dynamic environment.

Such agents will be able to do the following:

1) Find each other through discovery in open environments in which network connections, information sources, and agents can unpredictably appear and disappear.
2) Access, filter, aggregate, and distribute information.
3) Integrate information management and decision support.
4) Interleave planning, information management, execution, and monitoring.
5) Anticipate and perform human information processing tasks and problem solving.
6) Notify users and other agents about significant changes in the environment.
7) Discover and interoperate semantically.
8) Adapt to the user, task, and situation.

Building FSE Grid middleware as a society of multiple agents has the following advantages over a single agent, client/server, or centralized approach:

1) Computational resources and capabilities are distributed across a network of interconnected agents. A centralized system may be plagued by resource limitations, performance bottlenecks, or critical failures, while the FSE Grid middleware agent society is decentralized and thus does not suffer from the "single point of failure" problem of centralized systems.
2) Problems are modelled in terms of autonomous, interacting component-agents, which is proving to be a more natural way of representing task allocation, team planning, user preferences, open environments, etc.
3) Information can be efficiently retrieved, filtered, and globally coordinated from sources that are spatially distributed. It allows the interconnection and interoperation of multiple legacy systems. By building an agent wrapper around such systems, they can be incorporated into the agent society.

5. FSE Grid middleware agents society

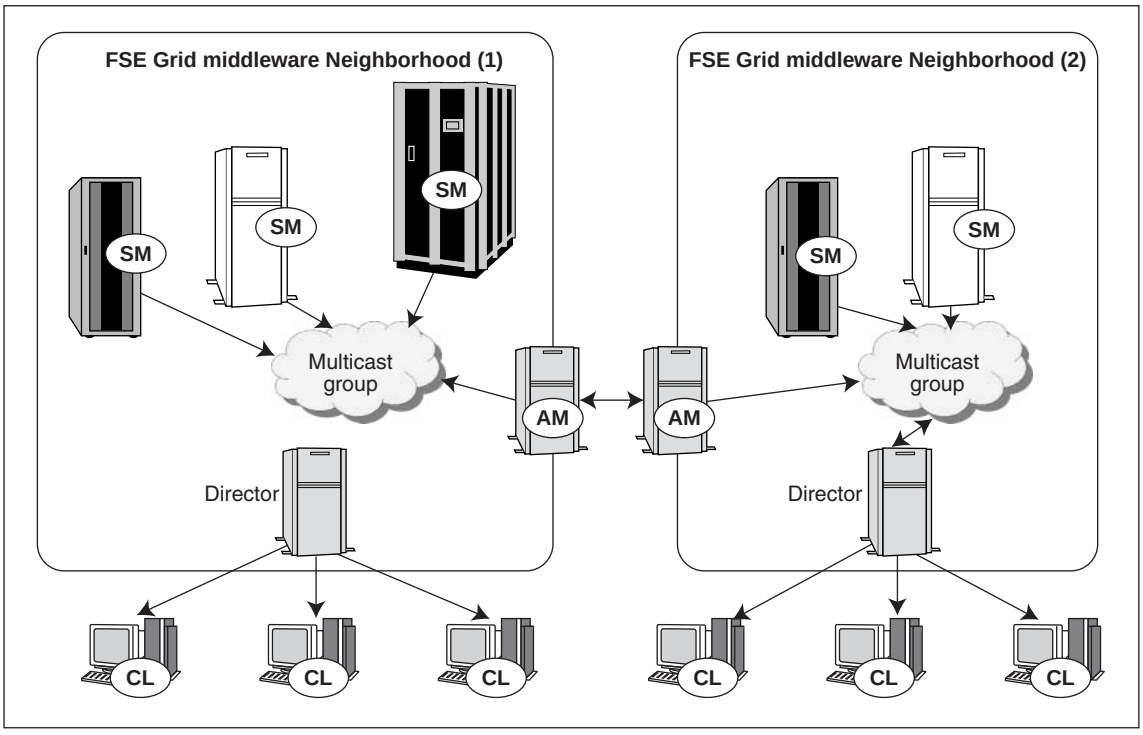
A FSE Grid middleware agents society consists of a variety of software agents that communicate together to form the backbone of the FSE Grid middleware network. Software Agent is an autonomous, collaborative, adaptive, and intelligent computational entity. Here, "intelligence" means the ability to infer and execute needed actions and seek and incorporate relevant information to meet given goals.[7)]

All these agents are bundled into components that collectively define a Grid environment called an FSE Grid middleware community (**Table 2**). Within this community, all applications and resources can be accessed via an integrated GUI desktop known as the FSE Grid middleware Client.

Figure 3 shows an example of an FSE Grid middleware community made up of two neighborhoods. Services from Neighborhood (2) are discovered by the Director in Neighborhood (1) and vice versa. FSE Grid middleware also makes it possible to restrict resource discovery to a single direction if that is desirable. Therefore, in this example, the Director in Neighborhood (1) could be allowed to discover the services being published

Table 2
Grid environment components of FSE Grid middleware.

Component	Description/function
FSE Grid middleware Community	A group of one or more neighborhoods that contain at least one Director and one or more Clients. Neighborhoods are linked together via the Acquaintance Manager (AM) to create a community with multiple neighborhoods.
FSE Grid middleware Neighborhood	A group of FSE Grid middleware components that normally includes at least one Service Manager (SM). If users connect directly to this neighborhood, it must also contain a Director.
Director	This component allows users to log into an FSE Grid middleware community to access their workspaces and run their workflows. It "directs" other components in their activities on behalf of the end-user. The Director discovers services and resources available within the FSE Grid middleware community and publishes them to the users.
Client	The Client component runs on the end-user's workstation. It enables connection to the Director and displays the FSE Grid middleware graphical user interface.
Service Manager	The Service Manager publishes services and runs services on behalf of users. Services are the way FSE Grid middleware allows users to access computing resources. Therefore, one SM is required for each system or cluster that will participate in the FSE Grid middleware community.
Acquaintance Manager	The Acquaintance Manager component gives access to Service Managers residing in another neighborhood, thus extending the scope of services and resources available to users. There is no limit on the number of neighborhoods that can be linked together.



CL: Client **AM:** Acquaintance Manager **SM:** Service Manager

Figure 3
FSE Grid middleware community example.

Table 3
Director's agents.

Name	Function
User Management Agent	This agent allows users to log into the FSE Grid middleware community. It authenticates users and passwords. It can be used with LDAP (Lightweight Directory Access Protocol) servers if they are available.
Workflow Agent	The Workflow Agent manages the tasks that are run as part of the workflow. It supervises the linkage of tasks and the conditions under which branching and looping within the workflow take place.
Job Manager Agent	The Job Manager Agent controls the workflow from an execution perspective. It provides a job style management of the workflow.
Scheduler Agent	This agent contains the heuristics and methods used to determine which systems will be used to run the tasks of a workflow.
Global Environment Agent	This agent creates the global environment for a workflow to operate under. This environment is kept as part of the entire "job" that is running.
Global Services Agent	This agent manages the overall view of services and how they can be utilized by the community. It therefore provides a global view (community view) of services that exist in localized FSE Grid middleware communities.

Table 4
Service Manager agent.

Name	Function
Environment Agent	This agent manages the local environment for running a particular task of a workflow. It contains information about how input is received, where output data is sent, and what directories and workspace resources are used for the task.
Services Agent	This agent manages how the requested service of a workflow task is to be executed. It knows exactly how the service is invoked on the given system or cluster. It therefore needs to know information such as how to interface with batch subsystems if they are used to run the service.

Table 5
Acquaintance Manager agents.

Name	Function
Multicast/Unicast (MU) Agent	This agent receives multicast packets from one neighborhood and propagates them to other neighborhoods it knows about.[8]

in Neighborhood (2) but the Director in Neighborhood (2) could be disallowed to see any services from Neighborhood (1). This is particularly interesting if you are using an ASP (Application Service Provider) style service arrangement whereby the ASP must not see any resources at the customer site but the customer should be able to see resources at the ASP center. This may also be applicable to certain situations within the same organization.

A variety of agents of the FSE Grid middleware Director, the Service Manager, and the Acquaintance Manager are shown in **Table 3**, **Table 4**, and **Table 5**, respectively.

Note that agent functionalities can be combined in one component. For instance, a running component could function both as a Director and as a Service Manager. Indeed, in its simplest usable form, an FSE Grid middleware neighborhood consists of the following:

1) An FSE Grid middleware Director, which also acts as a Service Manager
2) An FSE Grid middleware Client

Even the Client and Director could co-exist on the same system. However, in reality FSE Grid middleware will be used to manage a far more complex system consisting of many Clients, one or more Directors, and numerous Service Manag-

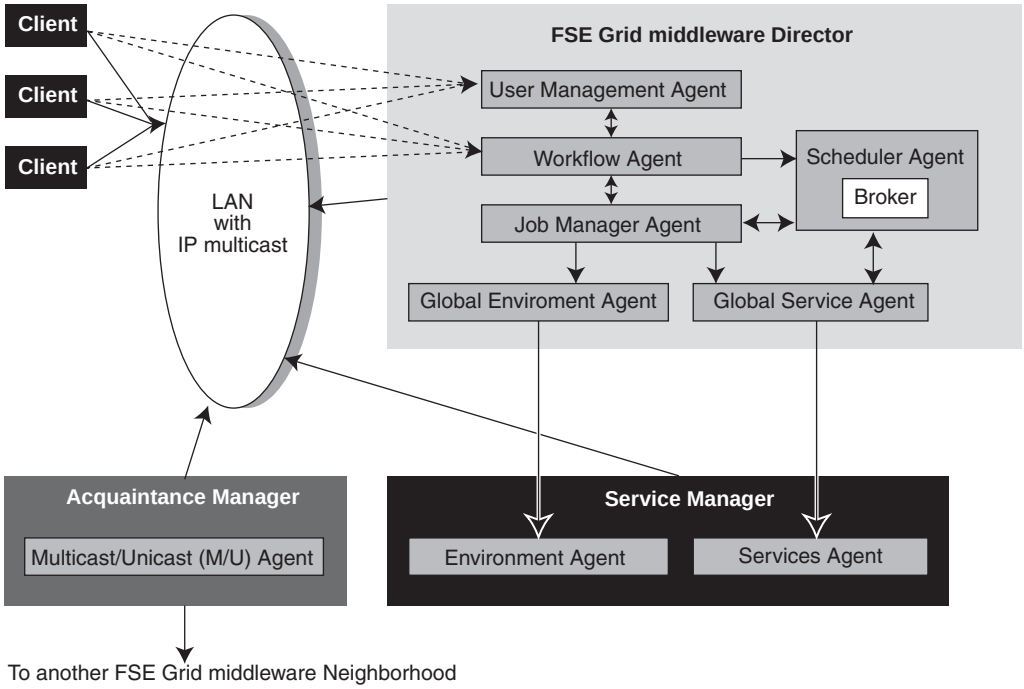


Figure 4
FSE Grid middleware agents.

ers scattered over various systems.

Figure 4 shows the main agents used in FSE Grid middleware and their locations within the Director, Service Manager, and Acquaintance Manager components.

6. Service oriented approach

One of the key factors that make FSE Grid middleware a powerful tool for all users is the fact that it is geared towards service management. This means that any unit of work that a user needs to exercise on a regular basis can be defined as a service and published on the FSE Grid middleware network.

Regular or complex tasks can be packaged into services for easy execution. Using these services, a user can easily run services regardless of how complex they may be. These services provide virtualisation of computing resources that exist somewhere in the FSE Grid middleware community (**Figure 5**). The benefits of virtualization are that users no longer need to be aware of the location of each resource they want to use. They do not need to have a direct contact with the systems that provide resources, and they do not need to know exactly how the resources must be used from a system's perspective. For example, traditionally if users wanted to run a complex application on a server, they needed to know the exact hostname or IP address of the server. They also needed to transfer any data required for the job to the system before it started execution, and then they needed to create and submit a batch job script. In FSE Grid middleware, the application is defined as a service and will be accessible via the Client GUI. In this GUI, you do not need to know the server's hostname or how to run a batch job. You will need to specify the source of the input data for the job, but you do not need to transfer data to the selected execution host because FSE Grid middleware will automatically do this before the job starts to run.

Services can be dynamically added, removed, and modified in real time with updates being instantly available to all FSE Grid middleware Client users.

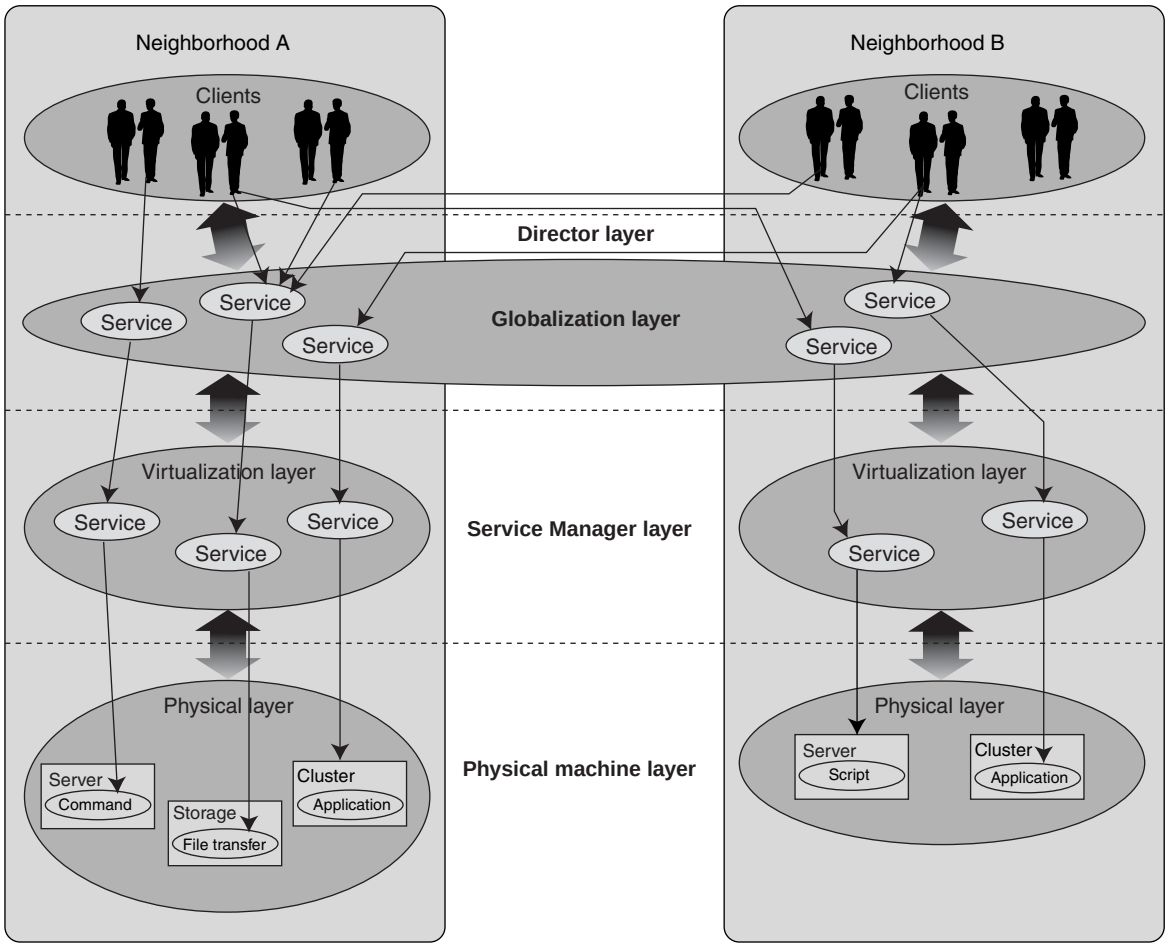


Figure 5
FSE Grid middleware virtualization of resources.

One example of a service that is integrated in FSE Grid middleware is service accounting. This service is highly configurable. The administrator can easily set up accounting per service, per user, per project, per site, or per another parameter. Project Managers can then track the usage of resources.

Services are developed on the basis of existing user or system applications. The basic actions for an application are running the application, listing intermediate and final result files in the run directory, viewing the contents of result files, and monitoring the status of the application run. The User Interface (UI) of the service is defined as part of the service itself. This UI allows users to specify parameters for running an application.

7. Workflow management

The decision to adopt an existing standard to represent a workflow is quite delicate—firstly because of the large variety of usage that currently exists and secondly because most of these standards are mainly related to business process management and are therefore Web-transaction oriented.

The emerging BPM (Business Process Management) industry has been considering multiple alternative paths for the modelling of executable business processes. Microsoft pioneered the adoption of the Pi-Calculus model with XLANG, IBM rejuvenated the use of Petri Nets with WSFL (Web Service Flow Language), and BPMI.org unified the two approaches with BPML (Business Process Management Language) 1.0:

1) XLANG - XLANG is from Microsoft. XLANG uses WSDL (Web Service Description Language) to describe the service interfaces of each participant. The behavior is specified with a control flow that controls the WSDL operations. Currently, there is no means for specifying data flow between operations.
2) WSFL - WSFL is IBM's proposal. WSFL is primarily focusing on describing Web Service compositions and uses WSDL to describe the service interfaces.
3) BPML - BPML is a specification from the www.bmpi.org organization (Business Process Management Initiative). BPML aims to provide a comprehensive means of globally specifying the processes of an enterprise.

Alongside these parallel efforts, other organizations have advocated radically different approaches for business process modelling, for example, ebXML BPSS (Business Process Specification Schema) developed by OASIS (Organization for the Advancement of Structured Information Standards). With the new release of BPEL4WS (Business Process Execution Language for Web Services), BEA, IBM, and Microsoft have adopted a model that is significantly similar to the one promoted by BPMI.org. BPML is a strict superset of BPEL4WS. BPML and BPEL4WS share an identical set of idioms and similar syntaxes as the basis for convergence.

Unfortunately, all these efforts toward a common workflow definition standard seem to induce more battles than treaties. Finally, it has been decided to follow the original recommendation from the workflow management coalition, which, even though it is the oldest, appeared to be the most realistic.[6] The FSE Grid middleware workflow conceptual model is compliant with the original one from the WfMC (Workflow Management Coalition) (**Figure 6**).

7.1 Workflow editing

A workflow is developed on the basis of previously published services. For each phase of a workflow, called a "task," a service is chosen and the total number of input and output files is specified. The coherency of the workflow can be checked, and the workflow itself can be simulated to ensure its coherency.

A graphical editor is used to build the graph defining a workflow. As an agent of the FSE Grid middleware society, the workflow editor has access to all available resources, giving the capability to check in real time that the application attached to a task is available to the current user (**Figure 7**).

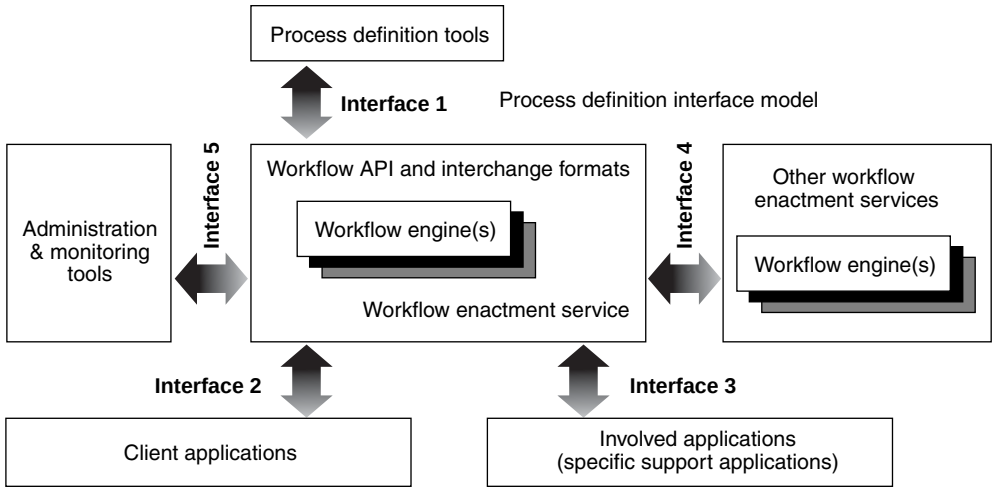


Figure 6
WfMC workflow model.

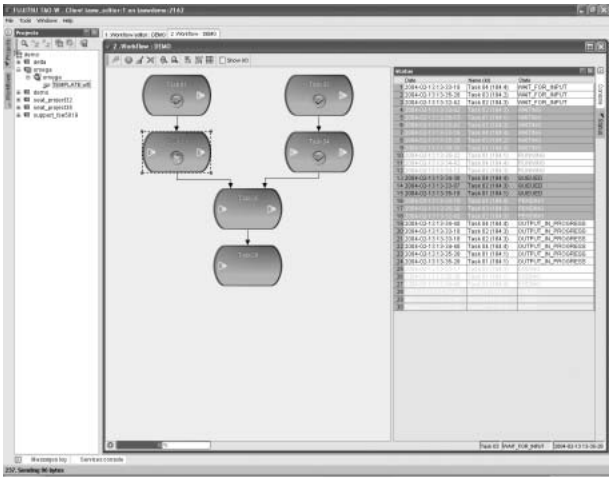


Figure 7
Workflow display window.

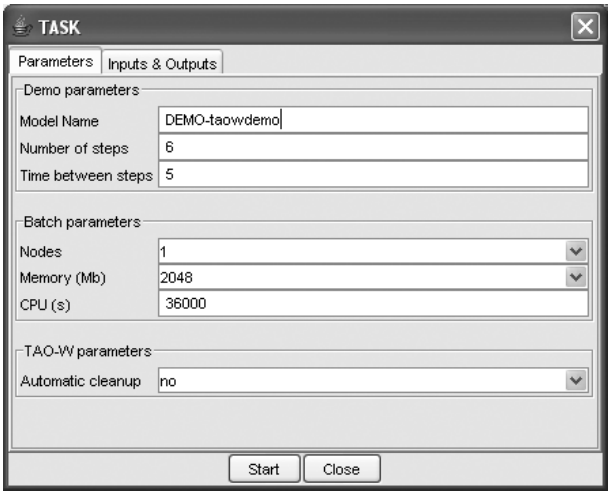


Figure 8
User input form.

7.2 Workflow execution

End-users can subsequently launch existing workflows at any time just by selecting them within the workspace/projects browser. Application parameters are eventually specified using the dynamically defined service UI. The specific input and output files are specified in this step.

The end-user does not need to be concerned about staying connected to enter the information for a given task before it is started. The workflow engine firstly manages task queues, including the "hold" queue, which keeps tasks that are waiting for user input away from the main scheduling queues. Secondly, all task definitions and UIs are instantly available, so the user can prepare and specify all the data needed to run a workflow and then just disconnect from the framework after launching it (**Figure 8**).

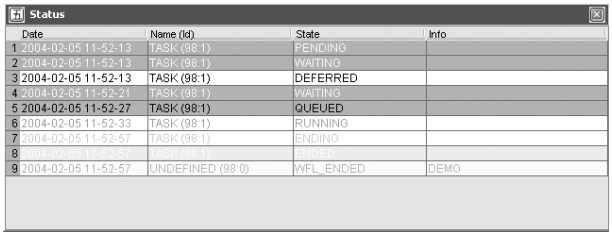


Figure 9
Workflow monitoring.

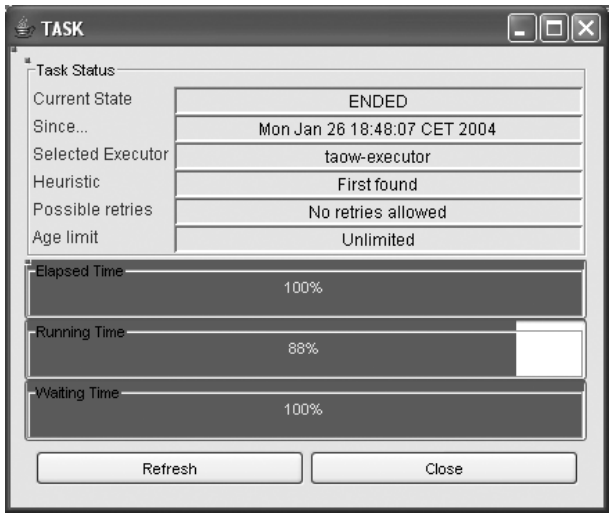


Figure 10
Task status panel.

7.3 Workflow monitoring

Once a running workflow is launched, it can be monitored at any time using various methods. The first method is to use a global status display that gives the entire history of all events that occurred since the workflow was launched (**Figure 9**).

Another facility is also provided to the end-user to check the status of a given task (**Figure 10**).

8. Usage topologies

The FSE Grid middleware software has been built to be as flexible and user friendly as possible. It can be deployed over a local area network (LAN) as well as over a wide area network (WAN) like the Internet to enable transparent and location-independent access to all available resources by the end-users.

In particular, the FSE Grid middleware software has been developed so as to adapt to environments in which both local and remote computing systems are used. We will now briefly discuss these two scenarios:

1) Local system usage: When used in the local system environment, users can utilize resources on their intranet. FSE Grid middleware systems and services can be accessed via the FSE Grid middleware Client agent running on each end-user's system. Local system usage normally uses one FSE Grid middleware agent society.
2) Remote system usage: If the FSE Grid middleware network consists of remote service execution agent systems as well as local execution systems, the user does not need to take any additional or special actions to access authorized services in the FSE Grid middleware Neighborhood. Remote system usage utilizes multiple FSE Grid middleware agent societies.

However in the current implementation, the end-user system must be able to make a direct connection with a remotely located FSE Grid middleware server to be able to start services and exchange data with the remote location. This restriction will be removed in the next release.

9. ASP usage

In this environment, users of FSE Grid middleware connect to a third-party computer center, which offers FSE Grid middleware based services. The principle of ASP is to provide seamless access to scalable and available high-performance computing. A provider hosts and manages a range of technical computing servers. On these servers are installed computing-intensive applications that typically require many hours of computation even on large multi-processor servers.

These centers can be used for special projects or for managing resource overloads at the end-user's in-house computing centers. It also may provide access to specific equipment that is not commonly available at the end-user's organization.

In such an environment, the strengths of FSE Grid middleware are as follows:

1) Seamless

Transparent network access combined with asynchronous data transfer.

2) Security

Security of the connection, security of execution on the ASP computing servers, and data integrity.

3) Confidentiality

Secure and confidential user access and data transfer, and protection against unauthorized human access to the data within the ASP facility.

10. Future development

The FSE Grid middleware was released in 2003 and was followed by new version in 2004. Nevertheless, the FSE Grid middleware project is not yet completed. The ultimate goal is to provide a unique framework for communication, intermediate brokering functions, and global resource management. The target of this framework is to construct a closed-loop control system called the application execution optimizer. This system uses various dynamic performance information sources to guide an application to completion despite performance variations in the underlying resource base via a process of adaptive control of both application behavior and resource demands.

All this framework will do is to secure a performance contract according to the rules of the resource economy that is in place. On top of this substrate, the key issue is to develop techniques for deciding how and when a performance contract has been violated (e.g., managing temporal vari-

Table 6
User feedback about FSE Grid middleware and Globus.

	FSE Grid middleware	Globus Toolkit 3.x
Installation	2 days (installation + configuration). No major changes in OS configuration; only a single TCP/IP port is opened.	1 month. Root user access mandatory to change and configure the OS. Many TCP/IP ports needed.
Functionalities	Shared resource management, multi-site security management, services publication, workflow and meta scheduler, transparent secured asynchronous file transfers even between remote sites.	Shared resource management, multi-site security management, services publication.
System usage	About 33 Mbytes of memory for a single multi-threaded server process.	At least 200 Mbytes and more than 25 processes running concurrently.
Orientation	User centric.	Service centric.
Utilization	Simple and easy to use, friendly GUI.	Relatively complex, no usable GUI.
Security	OpenSSL, embedded encryption, connection to LDAP servers, trusted users between different FSE Grid middleware communities.	GSI: X-509 certificate OpenSSL
Firewall	Very simple, with MU-agent facility provided for each FSE Grid middleware. Successfully demonstrated through multiple LANs and firewalls with NATing.	Very complex, not yet tested because Globus is not fully compatible with firewalls with NATing.

ation and distributed contract testing) and how to respond to violations in order to maximize application performance. To meet these requirements, FSE Grid middleware will define new strategies that let the compiler, scheduler, runtime system, resource brokers, and other components cooperate to non-intrusively extract pertinent information from the running application as well as from the execution environment.

11. Conclusion

Today, FSE Grid middleware is a powerful and yet easy-to-use environment that empowers end-users with the ability to use deployed applications as services regardless of their location within the enterprise network.

FSE Grid middleware is already used by many end-users within European companies from various domains, including automotive and aeronautic manufacturers, universities and research laboratories, and independent software vendors. According to end-user feedback, when FSE Grid middleware is compared to Globus Toolkit, the proven efficiency of FSE Grid middleware is clearly emerging (**Table 6**).

Fujitsu Systems Europe is also providing access to its ASP facilities through FSE Grid middleware. The Fujitsu Technical Computing Facility (FTCF) located in Paris is providing on-demand application services using FSE Grid middleware as the Client GUI. Access to FTCF is granted to qualified users through the FSE Grid middleware Client agent. This solution offers the maximum level of security for access and data transfer as well as a strict control of confidentiality between different users.

References

1) I. Foster et al.: The Anatomy of the GRID - Enabling Scalable Virtual Organizations. *http://www.globus.org/research/papers/anatomy.pdf*
2) I. Foster et al.: The Physiology of the GRID - An Open Grid Services Architecture for Distributed Systems Integration. *http://www.globus.org/research/papers/ogsa.pdf*
3) GRID survey - Global Grid Forum. *http://www.gridforum.org/*
4) Katia P. Sycara: Multi-agent Infrastructure for Agent Interoperability in Open Computational Environments. School of Computer Science Carnegie Mellon University.
5) R. Allen: Workflow: An introduction. WfMC External Relations Committee.
6) Workflow Management Coalition: Workflow

Standard - Interoperability. Wf-XML Binding Document Number WFMC-TC-1023.
7) H. Chi and K. Sycara: A Taxonomy of Middle Agents for the Internet. Robotics Institute, Carnegie Mellon University.
8) D. Plaindoux: The "MU Agent." FSE Grid middleware technical papers. *http://www.fr.fse.fujitsu.com/papers*

Dr. Roger Henri received the Ph.D. in Computer Science in 1978 from Toulouse University. He started his professional carrier at Burroughs Inc., first as an operating system designer and later as a compiler developer. Next, he worked at IGL as a software engineering expert for several years before entering Thomson-Syseca, where he became Director for Innovation in 1994. He also played a major role in France as a software engineering consultant for large companies. He joined Fujitsu Ltd. in 1999 to help design Parallelnavi/Workbench. He is now the FSE Grid middleware project leader at Fujitsu Systems Europe.

Dr. Pierre Lagier received the Ph.D. in Computer Science in 1986 from Toulouse University focusing on distributed operating systems. He started working as a system designer at the Software Engineering Institute in Paris and then joined the French National Space Agency (CNES) to design a new-generation spacecraft control center. In 1989, he left this position and moved to Alliant Computer Inc. in Boston to work as a system architect. He joined Fujitsu Ltd. in 1991, where he has been responsible for the design of a message-passing library (MPI2) for all Fujitsu hardware platforms; the design of performance analysis techniques for very large parallel applications; and various tools, for example, Parallelnavi/Workbench, for the VPP supercomputer series and the PRIMEPOWER computer environment. He is now Technical Director of Fujitsu Systems Europe and an FSE Grid middleware project designer.

Dr. Didier Plaindoux received the Ph.D. in Computer Science in 1997 from Toulouse University. His background is functional languages and strong-typed logical object-oriented languages. He started working with IT Service Company, SSTI, where he was in charge of the study of a specific language dedicated to SGML transformation for the aerospace industry and was also project leader for the CyberBox product. He joined Fujitsu Ltd. in 2000 to help design Parallelnavi/Workbench. He is now a senior researcher at Fujitsu Systems Europe and an FSE Grid middleware project designer.