A Multi-Agent Framework for Execution of Complex Applications

Alexandru Cicortas, Victoria Iordan

Computer Science Department
Faculty of Mathematics and Computer Science
University of the West Timisoara
e-mail: cico@info.uvt.ro, iordan@info.uvt.ro

Abstract: Complex applications execution needs a lot of conditions starting with hardware and software resources until the task sequencing and verifying the results of the execution. The needed resources can be found locally or on the Web and Grid. For an efficient usage the needed resources application and the effective resources found must be described in an appropriate and way and we propose the XML. The matching between the resources of the application and the Web/Grid resources is done in our proposal by the agents. As a new approach in the last years is the Service Oriented Architecture (SOA) that is developed as an efficient solution. The services are furnished by the providers and these are used by the clients in some specific conditions. The realization of the SOA technology presents a real opportunity to improve effectiveness. On the Web and Grid we dispose for services and these can be used in a given contest. SOA provides the application of well-founded concepts that exploits the ability of modern system resources to collaborate, independent of location across heterogeneous technologies.

Keywords: multi-agent systems, grid, resource management, complex systems design, service oriented architecture.

1 Introduction

Real world systems grow in complexity. The execution of complex applications implies adequate tools and also support for execution. The Web and Grid offer an efficient way for executing the requirements of companies that must interact in a very complex manner.

The grid model, with its use of resources tends to be heterogeneous and distributed across multiple management domains. For example, in a grid environment shared resources must remain accessible, not but that the hardware configuration is dynamic, the resources can be available in a time instant, but in an another time instant these can disappear. In the grid key infrastructure services must be available, and virtual organizations must be maintained. It must also be

possible to detect report and deal with faults that may occur in any of the member domains.

Effective system management is only possible if resources are manageable, and if tools are available to manage them. However, these tools tend to operate independently and to use proprietary interfaces and protocols to manage a limited set of resources, making it difficult for an organization to build an efficient, well-integrated management system. In order to allow the development of manageability standards that will enable conforming management tools to manage conforming resources in a uniform manner, and to interoperate with each other. In turn this will enable system administrators to choose their management tools and suppliers in the knowledge that, regardless of their origin, the tools can work cooperatively in an integrated management environment.

As stated in [15] the resource management includes:

- reservation, brokering and scheduling;
- installation, deployment and provisioning;
- metering;
- aggregation (service groups, WSDM collections, etc.);
- VO management;
- security management;
- monitoring (performance, availability, etc.);
- control (start, stop, etc.);
- problem determination and fault management.

Some of other major problems are to use and discovery the resources.

Beside resources, the grid offers also services. These services can be simple services or complex ones and use resources. Like resources the services must be represented in an adequate manner in order to be found and used. In many grid organizations the registers are used in order to furnish the resources and services that can be used or invoked.

The problem of execution of complex applications that must be solved in an efficient manner using the grid requires a lot of complex tools. Some of aspects of the problem concern:

- formulating the problem that is proposed to be solved;
- expressing an adequate decomposition of the application in the tasks;
- for every task to express the hardware and software resources that are needed;

- giving an appropriate expression of the task sequencing and conditioning;
- defining the services that will be invoked;
- searching and discovering on the grid for the appropriate resources and services;
- if there is the case to allow the migration of the software agents (SA) on the grid and also to install the needed (adequate) software;
- defining the data that must be partially hosted for the future tasks (temporary storage of partial data or needed data for recovering in the case of failure);
- defining the recoveries when some failure occurs.

The execution of such grid application can be expressed in terms of workflow. This subject will be detailed in another paper. Concerning the workflow of the complex application on the grid, there are many works like [4], [16] that describe how the workflow must be designed.

There exist a lot of works that threat partially or totally such a kind of approach in the grid context. The GLOBUS [11] UNICORE [18] and the works done for the Fraunhofer Resource Grid [10] solve in some partially sense the above requirements.

Being in many cases a complex problem the jobs execution needs to overcome some of the exception situations as rollbacks and resuming in the incidents. In a framework it implies some additional activities like:

- localization of the failed task;
- analysis of the task execution;
- the resuming of the task from an intermediary point if there exists a such point;
- resuming the data bases implied in the same node or other nodes.

There exist many frameworks that allow to the application designers to dispose for:

- tools, standards and languages to define express and use the information concerning the evolution of the application during its execution;
- an adequate Grid framework that allows to:
    - explore and find the grid resources;
    - establish in the grid context, the conditions for the task execution and in the failure cases the conditions for resuming the task (like the temporary storage of data);
    - launching and controlling the task execution;

- in order to obtain the desired results the framework must allow the rollbacks.

We propose also a framework.

Multi-agent systems are used in many domains and their strength allows developing complex systems. The agent abilities i.e., self adaptation, learning from own experience or from collectivity experience, the communication and collaboration are very expressive tools in order to reach the system goals.

The following concepts [20] are closely related:

- *Choreography* describes required patterns of interaction among services and templates for sequences (or more structures) of interactions;
- *Orchestration* describes the ways in which business processes are constructed from Web services and other business processes, and how these processes interact;
- *Workflow* is a pattern of business process interaction, not necessarily corresponding to a fixed set of business processes. All such interactions may be between services residing within a single data center or across a range of different platforms and implementations anywhere.

The XML is used for expressing the appropriate information that can be processed using various platforms in most recent initiatives concerning the usage of adequate standards that are applied in many domains. In many of real projects the XML [22] is an adequate mean in order to allow expressing the needs and it in platform independent. Based on the previous experience in the following the XML will be used.

The paper proposes:

- a framework that uses the XML that allows to express and furnish the required information to the designer. The information concerns the grid its resources, its services, its location and also the details of the application concerning its decomposition in tasks, the sequencing of the tasks and consequently the resources and services needed for tasks execution;
- a multi-agent system as a solution for solving such kind of problems;
- a workflow model for the execution of the applications in the grid context. It will be the subject of another work and papers due the particularities of the workflow.

The paper is structured as follows. The information used is shortly revised in the next Section. The third Section presents the concepts and tools used for modeling. The fourth Section, propose the multi-agent system as a possible solution. In Section 5 the framework is sketched and some details are given. The last Section contains some conclusions and future works.

2 The Information for the Grid Resources

Some of the resource Grid components usually are not described in the necessary detailed form in order that its description can be efficient used, i.e., all its attributes that specify if the resource or service can be used by certain service or software. For overtaking it we must define a function that allows to the user to receive the desired and detailed hardware resource attributes.

The resources are basically hardware resources and software resources. The hardware resource is characterized by:

- the resource type;
- the resource name, in some cases can be used an ID(entifier);
- the resource location;
- a set of its attributes;
- a set of its components; we can define in turn a structured resource.

The software resource is characterized by:

- the resource type;
- the resource name, in some cases can be used an ID(entifier);
- the needed support that is composed by a set of hardware resources every being quoted with appropriate attributes.

The information concerning the authorization, access and security is not detailed here in order to simplify the presentation.

The information that describes the hardware resource, software resource and the services, given as XML files, furnished by the grid will be processed in an appropriate way by the specific tools in order to fulfill the system requirements. In order to find the node where a task can be executed, a matching between the resource node in formation and the resource needed for the software resource (of the task) will be done.

In [7] is given a detailed resource Schema as a XML file where the authors describe a resource that can contain a structured resource one, its components, and every component having its own attributes and being also a structured one. The resource Schema will be used for describing the resource on the Grid or on the Web and also the needed resource for a software resource or for a service offered by the Web or by the Grid. The main feature of the proposed schema is that it can describe a complex resource. Concerning its usage depending on the level of detail that is given the user that will be some specific agent must be able to parse it and match with the given information offered by the requester.

3 The Concepts and Tools used in Modeling

A complex application is composed from a set of tasks that generally must interact and that must be processed (executed) in a specified sequencing order. For its execution every task needs a set of specified resources. The resources are specific and every resource has particular properties. In order to define a complete specification for a resource we will use some conventions from [12] that will be extended.

In many other grid projects are used various means for describing the dynamic behavior of complex grid applications. These means consist from coupled software components and appropriate data. The data is used for:

- mapping the grid application onto the underlying grid middleware;
- controlling the workflow and data flow of the grid application.

Based on the previous, we need tools that allow describing:

- resources of the grid;
- basic resources that are required to define the grid job;
- the model of the grid job workflow using various concepts like Directed Acyclic Graphs (DAG) [18] or Petri nets derivatives [1], [17].

Directed Acyclic Graphs (Figure 1) are widely spread due their structure but they have some disadvantages like: being directed it is impossible to define bidirectional coupling schemes, it is also impossible to define loops. The DAGs describe only the behavior but not the state of the system. The Petri nets can be used to control the workflow of complex applications [1]. Details concerning Petri nets and their properties can be found in [17]. The models for workflow that use Petri nets can be found in papers of van der Aalst [1]. In Figure 2 is given an example of Petri net that is equivalent to the DAG from Figure 1.

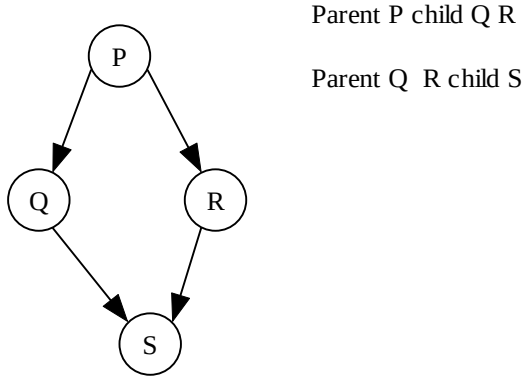

Figure 1
Example of a Directed Acyclic Graph (DAG)

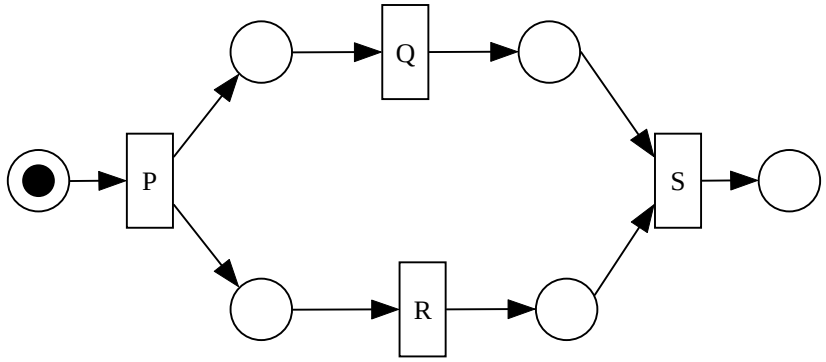

Figure 2
Example of a Petri Net equivalent to the DAG in Figure 1

The Petri nets can be used for modeling the workflow and also to control the workflow of complex applications. In many cases the workflow within grid applications can be equivalent to the data flow. That means that the decision when a software component can be launched in execution depends upon the availability of the input data. As a consequence the tokens in the Petri net represent real data that is exchanged between the software components in the grid. So, the Petri net can be used to model the interaction between software components in the grid represented by software transitions and data resources represented by data places.

In order to control the workflow due to the fact that in some cases the workflow is independent from the data flow is necessary to introduce control transitions and control places. In this case the tokens (that represent the process state) need some additional information that is the color of the token so the colored Petri nets [13]. The Petri nets used for modeling the workflow must have some particular properties that can be seen in [1]. These properties are:

- the net has one input location and one output location;
- the initial marking is that contains only one token in the input location;
- the final marking contains only one token in the output location;
- from any marking (state) between the initial marking and final marking there is a path to the final marking;
- by placing one transition that has an input arc form the output location and an output arc to the input location the Petri net is safe.

A formal verification of the modeled system can be used and it is based of the Petri net properties. This verification can be easy done and expressed in a mathematical form. Basic features that can be expressed like: conflict confusion, contact, trap, deadlock are well-defined properties of Petri nets that can be very useful during of the process of optimization of the workflow of a complex grid

application. We do not enter into the details concerning the features that can be given for express in Petri nets terms of the execution in parallel, sequential execution, definitions of the conditions or loops. There exist many approaches to describe Petri nets with XML-based language, e.g., the Petri Net Markup Language (PNML) [14].

Recently, UML activity diagrams [21] have been used for workflow modeling. There exist some approaches [9] that use the UML's activity diagrams those was added some especially semantic in order to improve a better utility in workflow modeling. The ebXML [8] is used to model e-business services. In the ebXML the event features of activity diagrams are used quite extensively: events being the standard means of communication between different business partners.

Business Process Modeling [23] offers also a set of standards and languages that allow expressing the workflow. Based on WfMC [23] documents was developed Business Process Modeling Notation (BPMN) [6] as a language that allows to describe a process that in our case can be a workflow.

4 The Service Oriented Architecture

The Service Oriented Architecture (SOA) can be understood as the necessary structure that supports communications between services. The service can be defined as a unit of work to be performed on behalf on an entity that can be human user or another program. Service interaction is loosely coupled and self-contained, every interaction being independent one another. The implementation of SOA can use the Simple Object Access Protocol (SOA). The protocol independence of SOA allow to the different consumers to communicate with the service in different ways. A management layer between the providers and consumers ensures a real flexibility regarding implementation protocols.

For an efficient usage of SOA, the Service Oriented Integration (SOI) and Service Oriented Management (SOM) were developed. SOI is used instead of the traditional Enterprise Application Integration (EAI). SOI has as significant characteristics:

- well defined standardized interfaces-consumers are provided with well understood and consistent access to the underlying service;
- opaqueness- the technology and location of application providing the functionality is hidden behind the service interface;
- flexibility- the service is constant both the provider of the service and the consumer of the service can change.

SOM constitutes the operational management of service delivery within a SOA. One of the major features of the SOM is to provide a differentiated service delivery capability during operation using specific objectives for driving the system behavior. A SOM solution supervises and controls the delivery of a service from a service provider to a service requester. A SOM solution can also be viewed as supervising and controlling the consumptions of services by a requester from a number of providers. A SOM solution should by able to manage any service from any technology without requiring code changes, special deployment, or special development environments.

Using a SOA solution we obtain a set of advantages: lower costs, flexibility, the web services that are point to point and it can not solve all the problems, a higher security.

5 The Multi-Agent Systems

Due of the complexity of problems around the development of grid applications an adequate and efficient solution can be developed using the agents that act in a multi-agent system. The agents must be designed in order to fulfill their own goals and also the system goals. The agents must cooperate and concur in order to obtain the needed resources. In many approaches is used the negotiation.

Negotiation between intelligent agents is one of the basic issues in Distributed Artificial Intelligence and Multi-Agent Systems [3]. A negotiation process tends to modify the own plan of each agent in order to achieve agreement among a subset of agents in the system. One of the main works in this area Contract Net Protocol (CNP) [19] serves as a basis for a lot of variants that are already frequently used. The CNP was initially developed for decentralized task allocation and is a distributed negotiation model based on the notion of call for bids on markets. The relation between clients (managers, customers, buyers) and suppliers (bidders, contractors, sellers) is created in a call for bids and evaluation of the proposals submitted by the bidders to the managers. This CNP has several limits:

As a multi-agent system is distributed, several managers can concurrently call for bids, so an agent may have to manage several negotiation processes in parallel in order to reduce the length of its negotiation processes. Some of the applications of the CNP force the contractors to sequence their negotiation processes, i.e., to answer with a single bid at a time. Sequencing the processing of contractor answers to the calls emitted by the various managers may make the contractors miss some contracts. When the multi-agent system is equipped with delay failure detectors, it may also force the managers to consider these contractor agents as failures.

CNP-based applications enable the agent to break its commitments when the agent receives an offer for a better task in comparison with those for which the agent is committed. However breaking the commitments is not always a good solution because it makes managers call again for bids to find anew contractors for their tasks.

6 The Proposed Framework Based on Multi-Agent and SOA

SOA is an architectural manner that has as objective to achieve loose coupling among interacting software agents. A service being a unit of work done by a service provider done by achieves desired results for a given consumer, both provider and consumer are roles played by software agents on behalf of their owners. The results of a service are usually the change of state for the consumer, or for the provider or for both. SOA achieve loose coupling among interacting software agents using as architectural constraints:

- a small set of interface to all participant software agents. Only generic semantics are encoded at the interfaces. The interfaces should be universal available for providers and consumers;
- descriptive messages constrained by an extensive schema delivered through the interfaces. An extensible schema allows new versions of services to be introduced without breaking existing services.

Interfacing in SOA and in distributed applications is important and also expensive and error-prone. The interface prescribes the system behavior and this is very difficult to implement correctly across different platforms and languages. In some cases is recommended to reuse generic interfaces, but this implies that the designer expresses application-specific semantics in messages. Due to the fact that the architecture is service oriented the messages sent must comply with some specific rules:

- the messages must be descriptive, rather that instructive, due to the fact that the service provider is responsible for solving the problem;
- the vocabulary used must be clear for all (provider and consumer) and the structure of the messages must be also well defined for an efficient communication;
- the extensibility allows to evolve the system and it is very important;
- the SOA must give to the consumer a mechanism that enables the consumer to discover a service provider under the context of a service sought by the consumer. This mechanism can be flexible and it (if this is

possible) does not have to be a centralized registry (in the Grid the registry is frequently used especially for nodes).

In order to improve its scalability, performance and reliability of the SOA the following constraints are needed:

- stateless service - the message that a consumer sends to a provider must contain all necessary information for the provider to process it. This makes a service more scalable because the provider does not have to store state information between requests. Each request can be treated as generic. In a way is improved visibility because the monitoring software can inspect one single request and figure out its intention. Due to the fact that there are no intermediate states to worry about the recovery from partial failure is relatively easy and the service becomes more reliable.
- stateful service – between the particular situations that must be carefully designed are: the existence of a session between a consumer and a provider due to the reasons of efficiency; providing of a customized service. This kind of services requires both the consumer and the provider to share the same consumer-specific context that is either included in or referenced by messages exchanged between the provider and the consumer. The drawback of this constraint is that it may reduce overall scalability of the service provider because it may need to remember the shared context for each consumer. It also increases the coupling between a service provider and a consumer.
- idempotent request – (it can be treated carefully due to the fact that means) duplicate requests received by a software agent have the same effect as a unique request. This allows providers and consumers to improve the overall service reliability by simply repeating the request if faults are encountered.

Concerning the service it can be considered in SOA as a piece of functionality whose properties are:

- the interface contract to the service is platform independent; the service can be consumed by a client from anywhere, on any OS and in any language;
- the service can be dynamically located and invoked; it hints that a discovery service is available. The directory service enable a look-up mechanism where consumers can go to find a service based on some criteria;
- the service is self contained, that is the service maintains its own state.

In Figure 3 is illustrated the usage of a directory service.

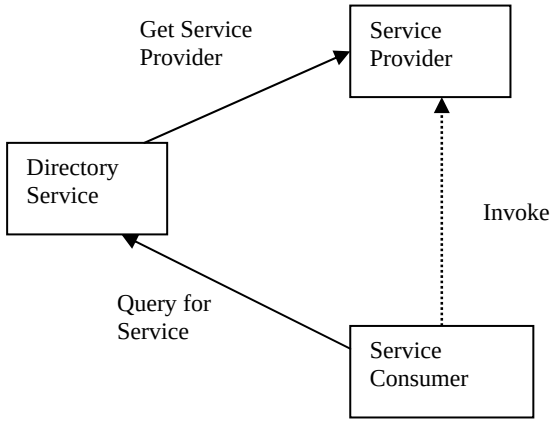

Figure 3
The usage of a directory service

In [12] is proposed a very complex model that uses a Grid Application Definition Language (GADL). It contains a set of XML-based description languages in order to define and assemble complex grid applications for mapping these applications onto the available hardware and software resources and to control the workflow during the execution. The GADL is composed from description languages for resources, interfaces data and jobs. Based on these the user interfaces with by Task mapping, grid job builder and Grid job handler.

The complex application executed on the grid (and not only) imposes a strong analysis concerning:

- its decomposition in tasks with the specification of the resources that are needed during the task execution;
- the tasks sequencing and where the case is the tasks correlation. It is a major challenge the tasks parallelization during their execution;
- the failures and erroneous task execution that imposes: the rescheduling or reminded tasks for execution in the new contest; the temporary data storage imposed by the failure and the supplementary services that are required for replying the rest of the tasks execution.

Concerning the execution, a task in characterized by:

- its identifier;
- the preconditioning of the task execution;
- the post conditioning (that is unleashed by its execution);
- the needed resources and their amount.

All these are given in some forms, usually as XML files.

The sequencing can be done in some ways, one of these being based on Colored Petri Nets. In this paper it is done in a simplified way using Place/Transition Petri Nets. Using such a representation we are able expressing the: concurrency (the parallelism) and synchronization between tasks (and tasks sequences).

In the following figures, we present some of the possibilities that illustrate the modeling of the application execution. In the first one is represented the data flow. Tokens in the places are the data and the transitions are the tasks. The application execution in time is given by tokens flow in the net. Also a tool that manages the Petri net that manages the application execution is needed. The Petri nets used are timed Petri nets. In Figure 4, the data flow and the tokens in the places represent data and the transitions are the tasks execution. Here the sequence of tasks t_2, t_4 and the sequence t_3, t_5 can be executed concurrent (in parallel). The task t_6 is synchronization (instant transition). The initial marking of Petri net is $(1,0,0,0,0,0,0,0,0,0)$ i.e., the start of the application. This marking enables [17] the transition (the task t_1) that fires and based on the Petri net rules the transitions t_2 and t_3 are enabled and also these transitions can fire.

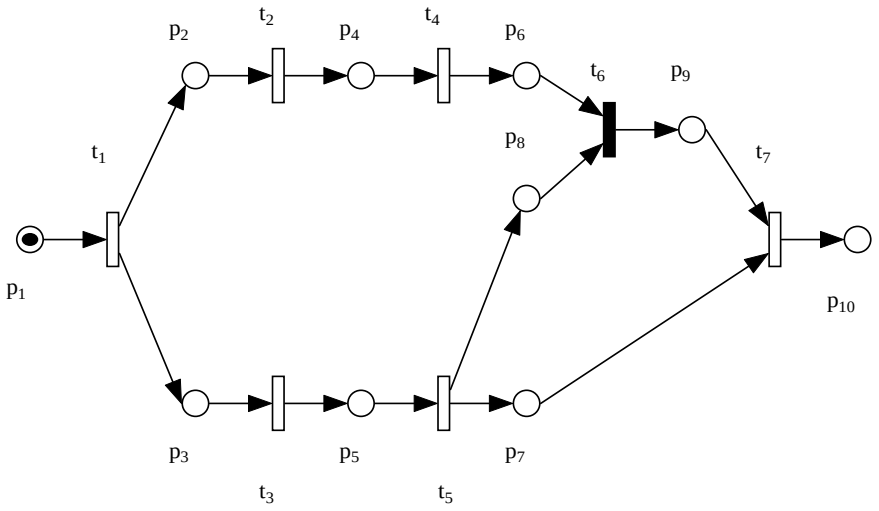

Figure 4
The Petri Net for an application

A major lack of workflow models is that the case instantiation is treated in isolation. In real life that is not true. The major problem concerns to the resources their contention and sharing (these are not treated by workflow models). Based on an analysis done we propose that the following features to be used:

- the time must be taken into consideration, i.e., all the activities evolve in time (the workflow also);
- as an intuitive representation we can imagine that every workflow of an application is represented in a plane and the application are represented in parallel planes;
- the resources that are available in the grid and are used in the application execution are situated between these planes. As an explanation we propose the following. Somewhere a resource is used by a task of an application. This resource is unavailable to other tasks because the token that represents the resource in its appropriate place does not exist, being used by the task. When the task execution is finished then the token that represents the resource is placed back into its appropriate place and can be used by another task.

The other one possibility, where the resources are represented, can be done in two ways. The first one, given in Figure 5 the resource that is in the place p is in the total amount #r and the task t_{1i} needs an amount a_{1i} for its execution that is acquired when it is available in the location p and when the transition t_{1i} fires (the appropriate task is executed) after that this amount is returned back to the location p. The same is the execution of the task represented by the transition t_{2j} from the Application 2.

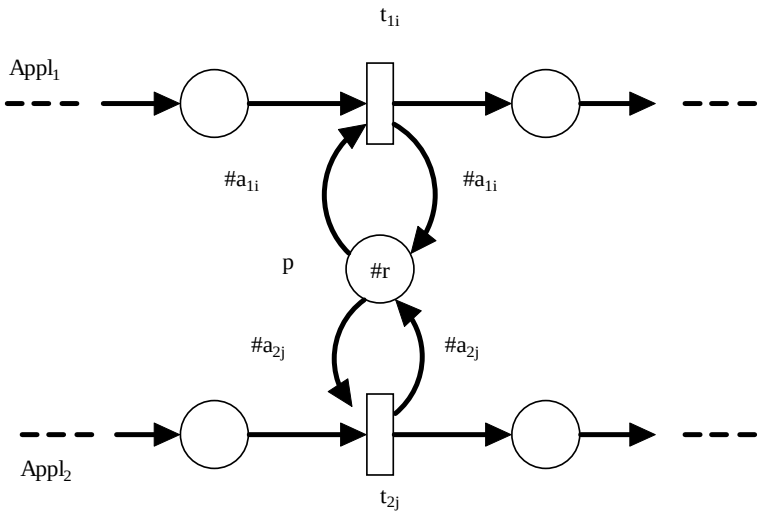

Figure 5
The first way of the representation of task execution

Another way is given in Figure 6, where the task execution is represented in a more complicated manner. Here the transition t_s represents the start of the task execution and the transition t_e represents its end. The time spent by the token into the place between the two transitions t_s and t_e. So, the transitions t_s start the task execution when the resource is available in the desired amount (as in the previous figure) and at the end of the execution, the amount used during the task execution is released and returned to the appropriate place of the resource (denoted by #r).

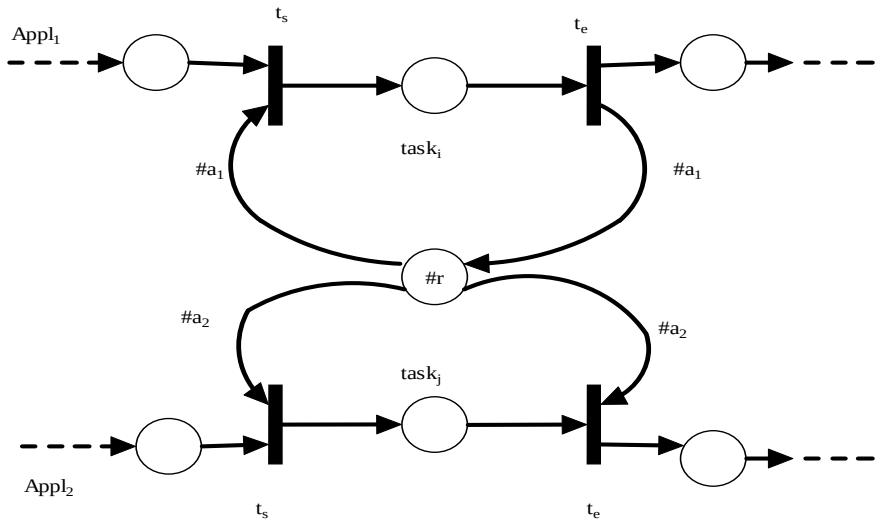

Figure 6
The second way of the representation of task execution

The proposed framework is basically a multi-agent system (MAS), where the agents execute appropriate actions in order to improve their goals and also the general objective: the execution of the application (job) on the grid. In the following, for specific actions are proposed agents that can play different roles. These requirements will be allocated to the roles of the agents. The agents that will be interacting in the system are: Descriptive Agent (DA), Explorer Agent (EA), Broker Agent (BA) and Supervisor Agent (SA).

The complexity domain constitutes a challenge in design due to the multiple factors that influence it. In the following we will consider that an application can be decomposed in tasks, a task being an atomic piece of the application that uses resources during its execution. First there are a set mo major problems concerning the requirement of complex applications like: a set of tools that must be used during the design and also during the execution. These tools are influenced by platforms that support these tools and also by the standards used.

Second, during the execution of a task it can fail. The failure imposes an analysis that must evaluate the effect on the whole execution of the application and the impact of the failure. based on this analysis there is need to take a decision concerning the fact that the execution of the application can be resumed or the execution of the application must be restarted (and in this case when).

Based on the application and failed task there are some aspects that will influence the reply of the execution:

- the conditions for the reply of failed task;
- data that must be partially stored as a consequence of the execution until the failed task and used in the reply of failed task;
- the node or location of the reply of the execution and in this case the time instant in that the reply will be started.

In these conditions we dispose for a set of agents that collaborate one to other and also that will help the user in order that in the proposed framework allows to the users to:

- describe and decompose the application in the tasks, their sequencing during the execution;
- describe the services that are needed for the execution of the application (including the resources needs);
- dispose for the capabilities that allow to reply the execution after a failure of a given task;
- exploring the Internet and/or the Grid in order to discovery the needed resources and services that are necessary for the execution of the application;
- to starting the execution of the application;
- dispose for an efficient tool for overseeing task execution in order to allow to analyses and reply in the case of the failure of a task;
- optimize in some sense the usage of the resources concerning the owners (of the resources) and also optimize the execution after the user criteria.

The Descriptive Agent (DA) allows to the user describes the application. As the result of the description of the application are:

- the tasks and their sequencing for the execution of the application;
- for every task the needed resources and where is the case their amount;
- the reply conditions in the case of the failure of some (every) task.

This description is given as XML files.

The Explorer Agent explores the Internet or the Grid in order to discovery the needed resources and services. It explores the grid and where exist registers it uses their content.

The result of its discoveries is expressed also as XML files.

The Supervisor Agent (SA) as main functions:

a) to receive the requirement for an execution of an application in the form of appropriate XML files that describes the application as we stated above;

b) send to a Broker Agent the requirements concerning the execution of the application;

c) oversees the application execution and in the case of a failure analyses and in the case of reply modifies (updates) the reminded execution adding the needed and supplementary activities (services);

d) interact with the user during the execution (especially in the case of failure that requires the user decision).

The Broker Agent (BA) using the application description and the results of the discoveries made by the EA parses these and negotiate with the Grid/Internet owners in order to allow that the execution can be done and when it can be scheduled at the location (for execution).

During the execution of the application based on the evolution of the execution, SA decides (if it is the case the user is consulted) when a failure occurs and updates the reminded files of the application, contacting a BA for the replies. In the case of a Failure the BA must send some notifications to the Grid/Internet owners concerning the fact that the rest of execution will be canceled and in accordance with the future evolution of the execution the new requirements.

In Figure 7 is given a generic schema of the framework. The user requires an application description (flow 1) that is done by DA. Using it the user requires (flow 2) to the SA to initiate a request for an execution. Based on it the BA requires to EA exploring the grid or the web and furnishes the appropriate information. BA negotiates (flow 3) with the owners of the grid or web the conditions for the execution informing the SA. After the confirming done by the SA (with the user accept eventually), the execution is initiated (flow 4) or the negotiation is repeated until the SA requirements will be satisfied. In the case of a failure for a task (flow 5), the new conditions are stated by the SA and the BA and EA evaluate these conditions and the flows 3, 4 (and eventually 5) are replayed. By numbers are presented main flows:

1-Requirement for the description of the application;

2-Requirement for the execution;

3-Negotiation;

4-Execution;

5-Failure.

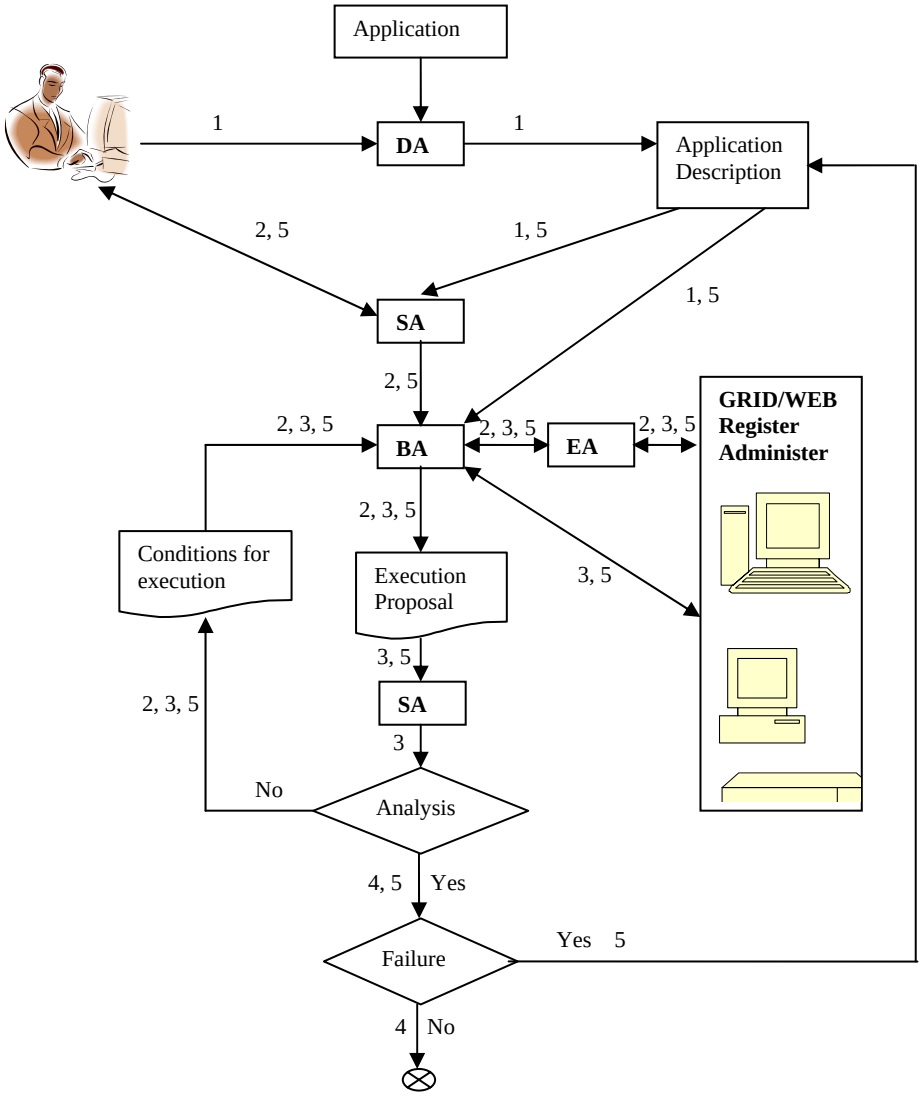

Figure 7
Generic framework schema

7 Applications

The first application is designed in Java. It is based on a client server protocol the client send a request and the server returns a XML file with requested information. If we intend to make a remote software installation the client will be able to interact with the obtained process as the result of the execution of the program that was transmitted for being installed. The client will interact with the standard input and standard output for possible errors of the process. The client will be able to examine only one sub-interval of IPs (or all the IP that are in the same sub network with him). The clients do not depend on the server and they have a Graphical Interface.

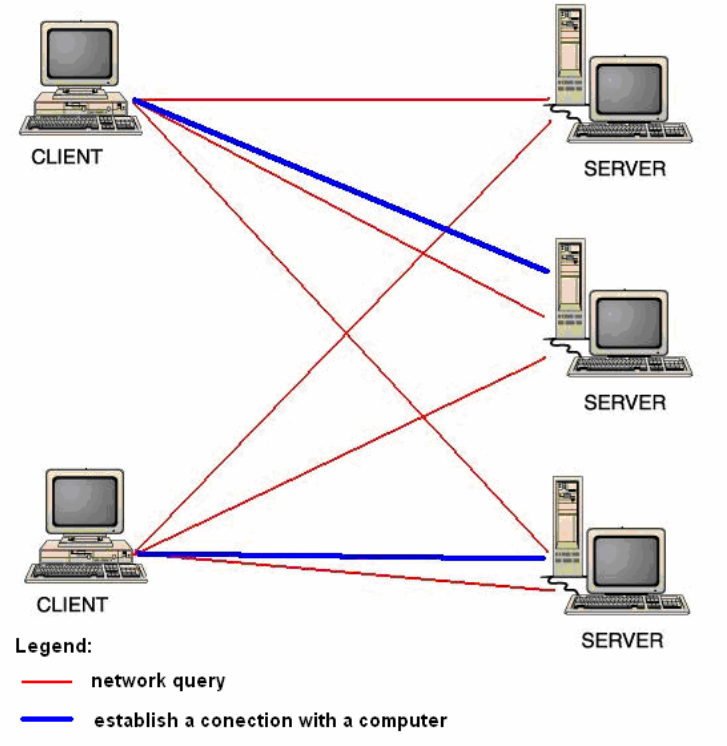

Figure 8
The possible requests of a client

Figure 8 illustrates the possible actions in the first applications. The client can made as requests connections with a computer and send queries.

The second application is designed in Microsoft platform (Visual C++) and allow to finding solutions for mapping of software resources over the available hardware resources. The user defines and can update for every hardware resource its characteristics and also for every software its requirements.

Having the hardware resources and the software with its requirements concerning the hardware that is necessary the user can verify if in the available hardware there the required resources are concerning the hardware for specified software. The user can save the project in an XML file. As a small example of such project is the following.

```
<root>

<resource id="software1" type="software">
        <resourceRef type="cpu-mhz" value="1000" />
        <resourceRef type="ram-mb" value="64" />
        <resourceRef type="os" value="linux" />
        <resourceRef type="kernel-version" value="2.6" />
</resource>
<resource id="software2" type="software">
        <resourceRef type="cpu-mhz" value="1200" />
        <resourceRef type="ram-mb" value="128" />
        <resourceRef type="os" value="linux" />
</resource>
<resource id="software3" type="software">
        <resourceRef type="cpu-mhz" value="2200" />
        <resourceRef type="ram-mb" value="512" />
        <resourceRef type="dot-net-framework" value="1" />
</resource>
<resource id="grid-node-01" type="hardware">
        <resourceRef type="cpu-mhz" value="1700" />
        <resourceRef type="ram-mb" value="512" />
        <resourceRef type="os" value="linux" />
        <resourceRef type="kernel-version" value="2.6" />
</resource>
<resource id="grid-node-02" type="hardware">
        <resourceRef type="cpu-mhz" value="2700" />
        <resourceRef type="ram-mb" value="1024" />
        <resourceRef type="os" value="windows" />
        <resourceRef type="dot-net-framework" value="1" />
</resource>

</root>
```

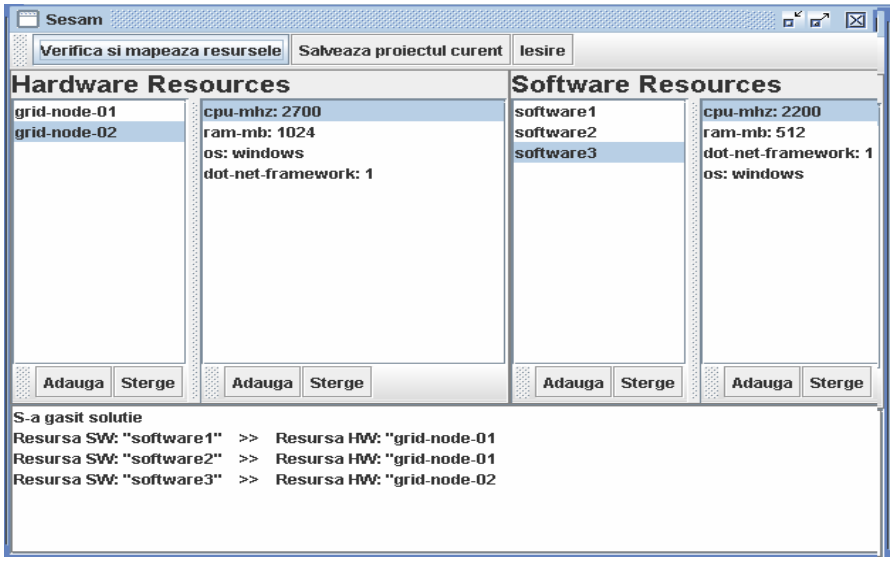

Figure 9

The Graphical interface for controlling the existence of the required software in the available hardware resources

In Figure 9 is given the graphical interface that allows to the user to update and verify that the needed hardware resources for specified software exist in the real available context of hardware.

Conclusions

The grid and its applications is one of the most dynamic and challenging domains. Most of companies and academic organizations are involved in grid and its applications. It is very difficult to develop a complex and complete framework for the execution of applications on the grid due to complexity of problems. The paper intended to propose a framework that is based on a multi-agent system and uses the XML as basis for information that is used in to the framework.

The design in the nearest future will be greatly influenced by the SOA. The services are most used in complex applications and in the application that uses the Grid.

References

[1] van der Aalst, W. M. P.: The Applications of Petri Nets to Workflow Management, The Journal of Circuits, Systems and Computers 8, pp. 21-66, 1998

[2] Activex-Grid: http://www.clientserverhelp.com/activex/activex_grid.html

[3] Akine, s., Pinson, S., Shakun, M.: An Extended Multi-Agent Negotiation Protocol, Autonomous Agents and Multi-Agent Systems, Vol. 8(1), pp. 5-45, 2004

[4] Brown, J.L, Ferner, C.S., Shipman, W.J: GridNexus and JXPL: A Grid Services Workflow System, http://www.globusworld.org/program/, 2006

[5] Business Process Management Initiative, http://www.bpmi.org

[6] Business Process Modeling Notation http://www.bpmn.org/Documents/BPMN%20V1-0%20May%203%202004.pdf, 2004

[7] Cicortas, A., Iordan,V.: Multi-Agent Systems for Resource Allocation, SACI 2005 2nd Romanian-Hungarian Joint Symposium on Applied Computational Intelligence, Timisoara, Romania, pp. 221-232, 12-14 May 2005

[8] ebXML http://www.ebxml.org, 2003

[9] Ershuis, R., Wieringa R.: Comparing Petri Net and Activity Diagram Variants for Workflow modeling- A Quest for Reactive Petri Nets, In H. Ehrig, W. Reisig, and G. Rozenberg, editors, Petri Net , Technologies for Communication Based Systems, LNCS, Springer-Verlag, Berlin, pp. 2325-354, 2002

[10] Fraunhofer Resource Grid http://www.fhrg.fhg.de, 2003

[11] The Globus Project: The Globus Resource Specification Language (RSL v.1.0), http://www.globus.org.gram/rsl_spec1.html, 2000

[12] Hoheisel, A., Der, U,: AN XML-Based Framework for Loosely Coupled Applications on Grid Environment, P.M.A. Sloot et al. (Eds.),ICCS 2003, LNCS 2657, pp. 245-254, 2003

[13] Jensen, K.: Colored Petri Nets, Vol1-3 Springer, 1997

[14] Jungel, M., Kindler, E., Weber, E.: The etri Net Markup Language, Philippi, S., (ed) Workshop Algorithmen und Werkzeuge fur Petri Netze. Univ. Koblenz-Landau, pp. 47-52, 2000

[15] Maciel, et.all: Resource Management in OGSA, http://www.ggf.org/documents/GFD.45.pdf, pp. 1-24, 2005

[16] Kacsuk, P., Dosza, G., Kovacs, J., Lovas, R., Podhorszki, N., Balaton, Z., Gombas, G: P-GRADE: A Grid Programming Environment, J. Grid Comput. Vol. 1, No. 2, pp. 171-197, 2003

[17] Petri, C.A.: Kommunication mit Automaten, Ph.D. Dissertation, Insitut fur Instrumentale Mathematik, Bonn, 1962

[18] Romberg, M.: The UNICORE Architecture: Seamless Access to Distributed Resources, Proc. of the 8th Int'l Symposium on High

Performance Distributed Computing (HPDC-8), Los Alamitos, pp. 287-293, 1999

[19] Smith, G. R: The Contract Net Protocol: High-level Communication and Control in a Distributed Problem Solver, IEEE Transactions on Computers, Vol. 12, 1980

[20] Treadwell, J.: Open Grid Services Architecture Glossary of Terms, http://www.ggf.org/documents/GFD.44.pdf, pp. 1-15, 2005

[21] Introduction to OMG's UML, http://www.omg.org, 2005

[22] Walsh, N.: A Technical Introduction to XML, http://www.xml.com/pub/a/98/10/guide0.html, 1998

[23] Workflow Management Coalition, http://www.wfmc.org