A Framework for an Adaptive Grid Scheduling: An Organizational Perspective

Inès Thabet[1,2], Chihab Hanachi[1], and Khaled Ghédira[2]

[1] Institut de Recherche en Informatique de Toulouse IRIT, UMR 5505, Université Toulouse 1 Capitole, 2 rue du Doyen-Gabriel-Marty, 31042 Toulouse Cedex 9
{Ines.Thabet,Chihab.Hanachi}@irit.fr
[2] Stratégie d'Optimisation et Informatique IntelligentE, ISG Tunis, 41 Rue de la Liberté, Cité Bouchoucha 2000 Le Bardo, Tunis-Tunisie
Khaled.Ghedira@isg.rnu.tn

Abstract. Grid systems are complex computational organizations made of several interacting components evolving in an unpredictable and dynamic environment. In such context, scheduling is a key component and should be adaptive to face the numerous disturbances of the grid while guaranteeing its robustness and efficiency. In this context, much work remains at low-level focusing on the scheduling component taken individually. However, thinking the scheduling adaptiveness at a macro level with an organizational view, through its interactions with the other components, is also important. Following this view, in this paper we model a grid system as an agent-based organization and scheduling as a cooperative activity. Indeed, agent technology provides high level organizational concepts (groups, roles, commitments, interaction protocols) to structure, coordinate and ease the adaptation of distributed systems efficiently. More precisely, we make the following contributions. We provide a grid conceptual model that identifies the concepts and entities involved in the cooperative scheduling activity. This model is then used to define a typology of adaptation including perturbing events and actions to undertake in order to adapt. Then, we provide an organizational model, based on the Agent Group Role (AGR) meta-model of Freber, to support an adaptive scheduling at the organizational level. Finally, a simulator and an experimental evaluation have been realized to demonstrate the feasibility of our approach.

Keywords: Grid Scheduling, Multi-Agent System, Adaptation.

1 Introduction

1.1 Context

Grid computing [1] provides computing capacities to high performance and data intensive applications by taking advantage of the power of widely distributed and heterogeneous resources.

Grids may be viewed as complex computational organizations. Indeed, grids are made of a large number of *heterogeneous, distributed* and *autonomous* components (administrative domains, network links, grid schedulers, local scheduler, resources, etc.) that have to interact frequently and cooperate to perform efficiently user applications. Without regulation, these interactions can become numerous, costly and unpredictable and therefore lead to a chaotic collective behaviour.

Regulation and efficiency are mainly dependent on the *grid scheduling* (GS) system and its interactions with the other components. The grid scheduling system allocates application's tasks to the most efficient resources at the appropriate time. Its interactions are also of paramount importance for finding available resources and allocating them tasks.

Adaptiveness is a required quality of the grid scheduling system since it has to face numerous disturbances due to the grid dynamicity. This dynamicity is due to the open character of the grid (resources can join or leave at any moment) and to performances variability of the resources: the network links or compute nodes may become overloaded or unavailable because of crashes or because they have been claimed by a prior (higher priority) application. Also, a more suitable resource may become available through time, etc.

1.2 The Problem Being Addressed

Giving this context, as detailed in the related work section, much work has been developed to design and implement adaptive grid scheduling. However, they remain at low-level, focusing on the scheduling component taken individually and modifying its structure and behaviour. This kind of ad-hoc solutions is not adequate since it is difficult to control and act on the internal behaviour of each component (micro-level) which are dynamic, numerous and, above all, unknown a priori.

Here, we argue that thinking the scheduling adaptiveness at a macro level, through the components organizations and interactions, is important and realistic and could better capture the grid complexity. Following this view, this paper adopts an agent-based organization perspective to design and simulate grid scheduling as a cooperative activity. Indeed, agent technology provides high level organizational concepts (groups, roles, commitments, interaction protocols) to structure, coordinate and ease the adaptation of distributed systems efficiently.

Giving these observations, the question addressed is: "how to design and simulate *an adaptive grid scheduling system* at *a macro level* (regulation of the components) according to an agent organizational perspective?"

1.3 An Agent-Based Organizational Approach

This paper deals with the modelling and simulation of *an adaptive grid scheduling* system able to detect the environment disturbances and to respond to dynamic changes in an efficient way. In fact, grid scheduling adaptation can be considered according to different views and requires to identify *i) the perturbing events*, *ii) the objects/components* that could be subject to adaptation, and *iii) the actions* to undertake in order to adapt.

From the conventional grid scheduling system [2][3], we have identified the *four following perspectives* to be considered in order to perform the adaptation:

- *The environment.* The grid environment is mainly composed of grid resources and applications. The environment adaptation is considered according to two aspects. The first aspect concerns the grid resources that could be aware of perturbations events and be able to perform adaptation actions such as rejecting a submitted task when it is overloaded, preempting a prior task, sending a failure notification, etc. The second aspect is about the submitted applications that could adapt to perturbations when the scheduling components is integrated in the application itself [4][5][6]. In this case, each application has its own scheduler that determines a performance-efficient schedule, monitors resources performances at the execution time and undertakes adequate adaptation action in case of disturbances.
- *The software components.* Grid scheduling components have to be endowed with the capacity to monitor the execution of tasks and to adapt in the case of disturbances such as rescheduling a failed task, scheduling policy adaptation, etc.
- *The grid organization.* Grid resources are physically organized in clusters, sites, domains, etc. One or more grid resources can also form a virtual organization [7] to offer new or better quality of service. In this context, the grid could be considered as a flexible computational organization made of interacting components: any change in the environment can be easily translated as reorganization, role modification, use of different interaction protocols, etc.
- *The interactions between grid components.* Interactions are here crucial for adaptation since they allow the grid components to cooperate efficiently to find an adequate partner to execute optimally application's tasks even if a perturbation occurs (use of contract net, auction protocols, etc.).

Giving these views, we define a typology of the adaptation in the grid (see section 3) covering the four previous views. Then, we focus on the organizational view following an agent-based approach.

The agent-based organizational approach is a conceptual framework providing the advantages of abstracting a system at a high level with macro and social concepts (roles, interaction protocols, groups) and allows the easy design and implementation of open and dynamic systems such as grids. Moreover, organizations are flexible entities able to adapt their design to any change in their environment and techniques to make them adaptive are today available [8] and could be used with benefit for the adaptation of the grid scheduling. *Multi-agent systems (MAS),* widely recognized in the grid literature [9], also provide abstractions to build autonomous, reactive and proactive software components able to communicate with sophisticated languages and protocols. Not only these concepts should help us to better structure and regulate the components behaviour, their interactions and their execution but also it should improve the efficiency of the grid scheduling by structuring the high number of communications and by supporting the grid organization.

1.4 Our Contributions

The contribution of this paper is the definition of a *framework for designing and simulating a multi-agent organizational model for an adaptive grid scheduling*. Our contributions consist of four components:

- *A conceptual model of the whole grid system (section 2) and its related typology of adaptation (section 3).* Our grid modelling includes both a conceptual model and a multi-agent model of the grid. The first model represents an abstraction of the grid system in which the concepts, their properties, and their relations are identified. This representation is computation independent and is used to identify i) a number of situations requiring an adaptation (according to the objects of the conceptual models) such as resource breakdown, departure, overload, etc. ii) a number of actions to undertake to adapt to disturbances (application migration, rescheduling, change of the scheduling policy, etc.). The second model describes the representation of the first one in terms of agents. In fact, our grid components are fully rethought and modelled as agents in order to implement an *effective distributed and coordinated adaptive scheduling*. Our multi-agents model is represented according to the "vowel" approach [10] that describes the MAS as a set of four main perspectives: the environment, the software components (agents) involved in the system, their interactions and their organization. The proposed multi-agents model is platform independent since it is not tied to any particular implementation and is used to describe a general multi-agents architecture for an adaptive grid scheduling.
- *An adaptation model for grid scheduling.* This model follows an agent-based organizational perspective. It describes the actors involved in the system, the links that exist between them and the protocols that rule their interactions. In this model, the adaptation is supported by reorganization and by the flexibility of the interaction protocols. This model follows the Agent Group Role (AGR) meta-model of Ferber [11] and is platform dependent.
- *A simulator of the adaptation model.* This simulator is compliant with our organizational model and it simulates any organization represented by our model. The implementation is based on the Madkit platform [12] that integrates the AGR meta-model.
- *Performance criteria definition and an evaluation of our adaptation model (section 6).* The performance criteria are related to the organization qualities. They are adapted from Grossi [13] and Kaddoum [14] and include structural and statistical aspects. The *structural aspect* is concerned with the network topology and is based on graph theory's measures. The *statistical aspect* is concerned with the communication load and with the execution time.

1.5 Organization of the Paper

The remainder of this paper is organized as follows. In section 2, we describe our grid conceptual model and our multi-agent grid model according to the environment, agent, organization and interaction views. We also specify accurately in section 2 the components behaviour and the protocols ruling their cooperation using Petri Nets [15]. In section 3, we introduce a typology of disturbing events and the type of adaptation that could be undertaken. An overview of the proposed organizational model for grid scheduling (AGR model, protocols, groups, roles, etc.) is presented in section 4. In section 5, we describe the developed simulator used to evaluate our model experimentally. We propose performance criteria and the evaluation of our organizational model for grid scheduling in section 6. In section 7, we review the related work. Finally, we summarize and lay out the future work.

2 Conceptual Modelling of the Grid Scheduling

This section presents the definition of the concepts and the different entities involved in the grid scheduling system. To this end two models are introduced: a *conceptual model* and *a multi-agent model* of our grid scheduling system. The *first* one represents a high level abstraction of our domain that is independent of any implementation technology. This model has been designed in order to give a description of our universe of discourse and is mainly used in order to define an adaptation typology (see section 3). The *second* one represents the agent approach adopted by our work. To this end, the conceptual model of the grid scheduling is modelled as a multi-agent system that specifies how the functionalities of our grid are realized using agents. This model is platform independent. Finally, a multi-agent organizational model that represents our developed framework is introduced in section 4.

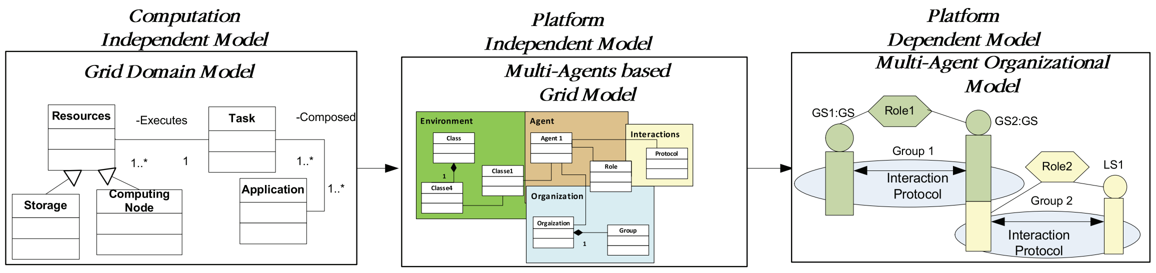


Fig. 1. Conceptual modelling of the grid scheduling

2.1 A Conceptual Model of the Grid Scheduling

In this section, we describe our domain which is a representation of the concepts implied in the grid scheduling and their relations. They are represented using UML (Unified Modelling Language) notation.

The concepts identified and the properties mentioned were selected following a deep investigation of the domain [2][3][16]. The instances of these models define the objects that can be subject to adaptation and the actions to undertake in order to adapt in case of disturbances.

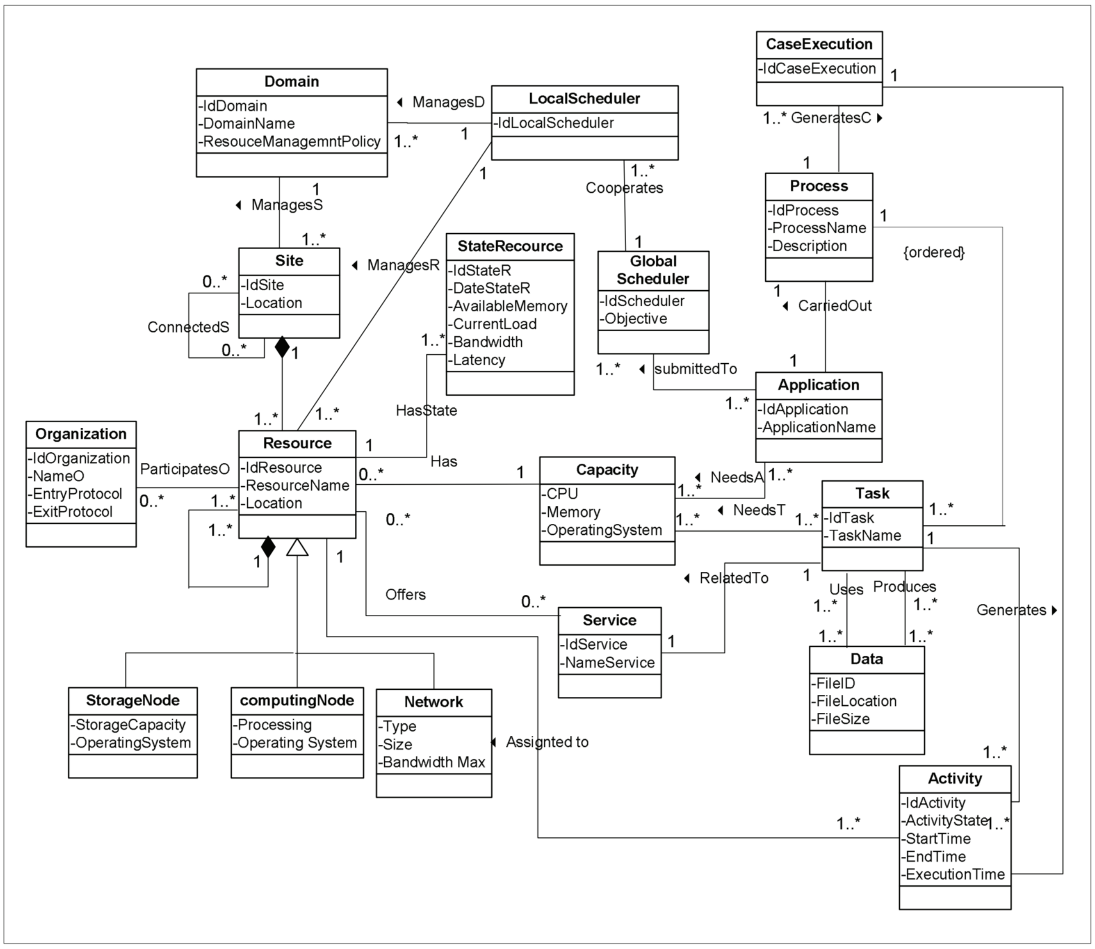


Fig. 2. Conceptual model of the grid

The final model is depicted in figure 2. The most important concepts of this model are:

- *Site*. It is an autonomous entity composed of one or several resources and managed by an administrative domain in which the local resource management policy is well specified.
- *Resource*. It is the central class of our model. According to [16], a resource is a basic device where jobs are scheduled/processed/assigned. Resources are required to execute users' applications. The resource can be a processor responsible for tasks processing, a data storage in which the data needed for tasks execution is stored, or a network link used for data transfer. Each resource can be connected to one or more other resources and has both static and dynamic characteristics. Static characteristics are about the resource name, the memory size, the operating system, etc. Dynamic characteristics concern information that can evolve over time such as the as the resource processing speed or load capacity.
- *Service*. Resources can be endowed with some capabilities (skills), called "Service" that are needed to execute a task. The quality of the offered service is dependent on the resource capacity (memory, CPU speed, etc.).

- *Organization.* One or more grid resources can form a virtual organization to offer new or better quality of service. Each organization has its own protocol of resource entrance and departure.
- *StateR.* This entity stores dynamic resource information such as the measurement of bandwidth and latency, the load of the processor, the memory available, etc. This information is used by the global scheduler in order to make appropriate scheduling decision.
- *Global Scheduler (GS).* It receives users' requests for applications execution. It selects feasible resources for these applications according to acquired information about the grid resources, the applications needs and it finally generates an application-to-resource mapping based on a certain objective function (maximizing resource use, minimizing time execution, etc). Several GS can be deployed on the grid and cooperate in order to execute optimally users applications. These GS could be organized in a decentralized or a hierarchical way.
- *Local Scheduler.* It is mainly responsible for two jobs: the local scheduling inside a domain according to the local management policy and reporting resource information to the Global Scheduler.
- *Request.* It concerns an external request of a user application execution (medical, biological, weather, etc).
- *Application.* It is a set of tasks coordinated according to a process that will be executed on a set of resources. An application has specific requirements for its execution such as the amounts and type of needed resources, the required time intervals on these resources, the termination time, etc. This information is stored in the entity capacity.
- *Capacity.* It can define task or application requirements as well as the capacity of a resource. The Global Scheduler has to generate an application-to-resource mapping, based on certain objective function, resource information (its capacity and its state when the application will begin its execution on), applications and tasks information.
- *Process.* It describes the order of the application's tasks execution. From a process, several alternatives execution plans can be generated.
- *Task.* It is an atomic unit to execute on a resource. Each task has specific resources requirements for its execution (CPU, memory, beginning/finish time, etc.) stored in the entity capacity.
- *CaseExecution.* The execution of an application according to a process generates a process instance that we call "CaseExcecution".
- *Activity.* The execution of a task on a grid resource generates a task instance. This entity stores all information necessary for the adaptation, such as the task instance state (affected, waiting, in execution, stopped or finished) and the results of the execution on a given resource. These results are used by the Global Scheduler to predict the performance of task execution on a given resource.

In addition to the defined concepts, our model has an integrity constraint that we specify textually. It relates the application time execution and the sum of its tasks durations. These durations are defined through applications and tasks requirements and are stored in the class capacity. The constraint is as follows: "the application time execution must be higher or equal to the sum of its tasks execution times". This is due to the time necessary for tasks synchronization and particularly for data transfer.

2.2 A Multi-Agent Model of the Grid Scheduling

In this section we describe our multi-agent model of the grid scheduling. In fact, as we previously mentioned, all grid components are modelled as agents. Moreover, we have followed the separation of concerns principles and modelled our multi-agent system as several interacting models: the *environment*, the *agent*, the *organization* and the *interactions models*. Each model is coping with a well-defined function and is used to describe the structure and the functioning of our multi-agent system. This representation is compliant with the “Vowel” approach [10] of Demazeau that eases the construction of the multi-agent system and facilitates the reusability and the maintainability of each model. According to Demazeau:

The ***environment view*** describes the environment in which evolves the agent which may be the physical world, a user via a graphical user interface, a collection of other agents, a combination of these objects. In our case, our environment defines the physical resources with their connections, their states and capacities, the users’ applications with their tasks and the agents managing the grid.

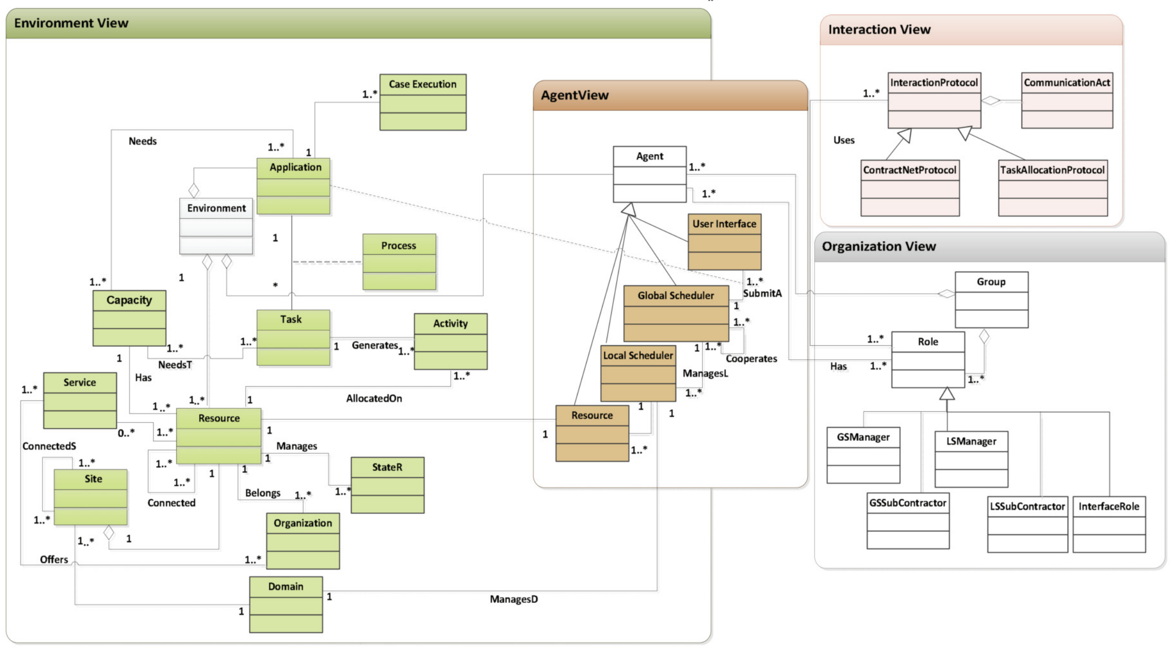


Fig. 3. Multi-agents model of the grid scheduling

The ***Agent* view** describes the agents, their internal architecture and their functioning. In our case, all grid components are modelled as agents. Let us introduce the agents’ architecture and their functioning.

— The ***user interface (UI)*** is able to interact both with the user and with the middle layer. It submits the user query to the most adapted Global Scheduler. A user query corresponds to an application made of several coordinated tasks.

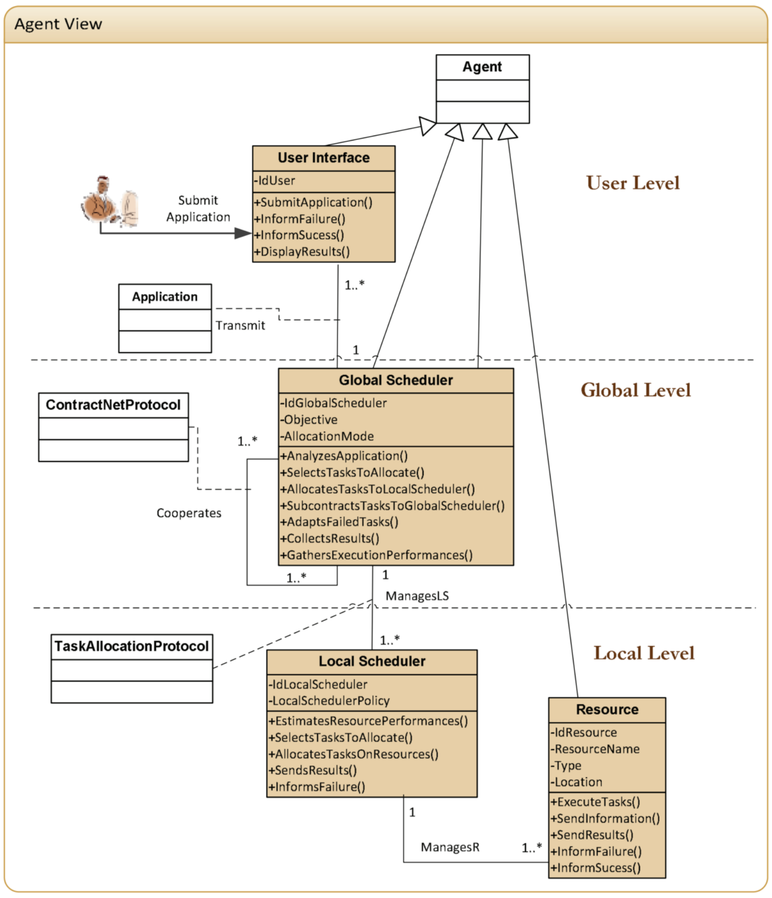


Fig. 4. The agent view

— The ***Global Scheduler Agent*** *(GS)* is responsible for managing one or several domains and scheduling users' applications on a set of adequate resources. To do that, the GS behaves as a workflow engine to allocate applications tasks to resources. The GS orchestrates the user' application modelled as a process to identify the set of tasks ready for execution (without preceding constraints) and tasks waiting for their preceding task execution completion, then searches for the best pool of resources so an objective function is optimized (minimizing makespan, execution time, etc.). Resources selection is based on application tasks requirement, on resources information (capacities, CPU, memory available, etc.) and on resource performance estimation provided by Locals Schedulers in order to choose the best pool of resources for application tasks execution. Tasks could therefore be allocated locally by the GS using a tasks allocation protocols or sub-contracted to GS' acquaintances (Other Global Schedulers Agents) using the Contract Net Protocol [17] if there are no adequate resources in the GS' local domains. Selected resources form a virtual organization made by multiple distributed resources for application tasks execution. The virtual organization functioning is detailed in section 4. Once tasks are submitted, the GS is also responsible for monitoring, if necessary adapting the grid functioning in case of disturbances, collecting the execution results and reporting them to the user interface. The resource monitoring concerns gathered information related to the resource executing the submitted tasks and includes resource availabilities, resource load and resource estimated completion time, etc. The adaptation mechanism is detailed in the next section and are basically based on the use of organizational structure adaptation, on rescheduling failed tasks and on interaction protocols ruling the cooperation between the grid components and allowing the applications execution completion even when perturbations occur.

- The ***Local Scheduler Agent*** manages one or several resources belonging to the same domain. Local Scheduler Agents receive tasks from their Global Scheduler and they are responsible for their scheduling according to a local policy. It consists in determining the order in which tasks are executed in their local domain (FIFO algorithm, greedy algorithm, etc.). The Local Scheduler Agent is also in charge of detecting failure and for reporting information to its Global Scheduler Agent to undertake adequate adaptation actions.
- Resources provide high computing capabilities to enable task execution. Each resource is managed by a ***Resource Agent*** in charge of reporting information (resource state and availabilities, tasks execution progress, execution results) to the Local Scheduler.

The ***organization view*** provides a representation of the logic agents' organization in groups in order to execute the user applications. In our case, one or multiple specialized resources (dedicated to a type of task) can form a virtual organization to offer new or better quality of service. The organizations view is modelled using the AGR (Agent Group Role) Meta-model of Ferber [11]. This model organizes the agents architecture described above using different roles and groups and will be detailed in section 4.

The ***Interaction View*** describes the relationship between agents through protocols or interaction language. In our architecture, the Global Scheduler Agent performs two types of protocols: a *task allocation protocol* with its Local Schedulers and the *Contract Net protocol* with other GSs. Each protocol transmits, receives and interprets communication acts. We use here FIPA-ACL [18] that supports high level communication between the different grid components. FIPA-ACL has several advantages. Each communication act follows a <performative (<message>)> form. Figure 5 shows an example of FIPA-ACL message for task execution call for proposal. Performatives transmit the intention of the communication acts (inform, query, request, call for proposal, etc.), while the message is a complex data structure where the domain and the content of the messages may refer respectively to a variable ontology and language. This feature eases interoperability between grid components and improves communication between them. Let us detail each protocol inspired from multi-agent protocols and adapted to our scheduling context. We use Agent Unified Modelling Language (AUML) to describe at a high level the interaction protocols.

```
(cfp
:sender(agent-identifier: name GS1
:receiver(set(agent-identifier: all GS
:language XML
:ontology gridModelOntology (compliant with the conceptual model of the grid see section 2.1)
:content
(<?mxl version="1.0"
<taskExecutionRequest>
<type>matrix-multiplication</type>
<size>15000</size>
<numberProcessorsMin>5</numberProcessorsMin>
<Cpufrequency>2Ghz</CpuFrequency>
<MemoryMin>512</MemoryMin>
<StartTime>t+10</StartTime>
<EndTime>t+50</EndTime>
</taskExecutionRequest>
)
```

Fig. 5. Example of FIPA-ACL message for the task execution call for proposal

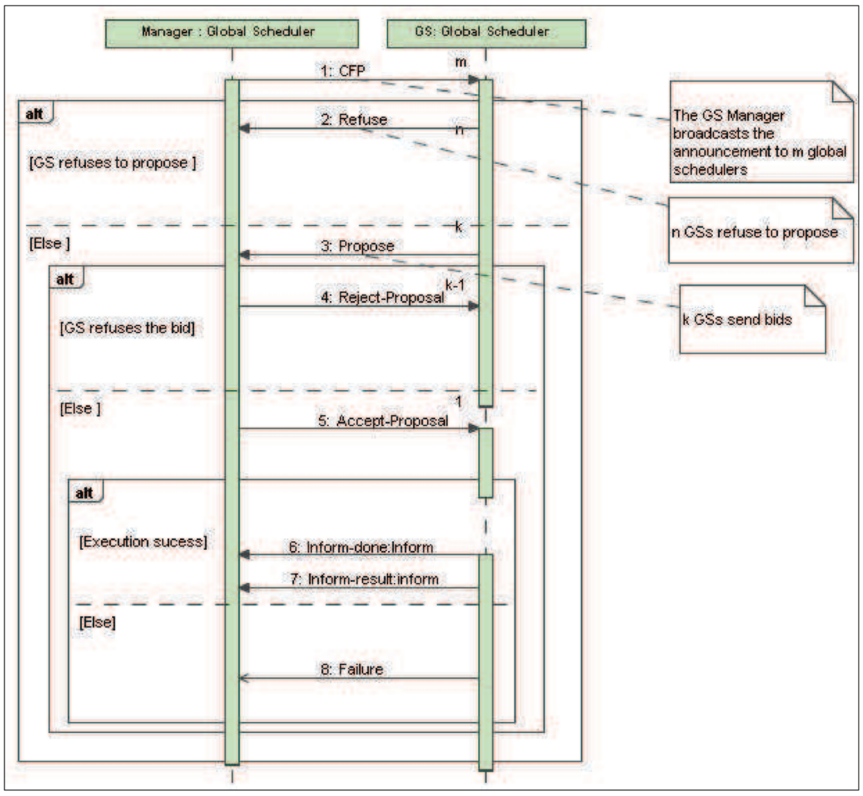


Fig. 6. The Contract Net protocol inspired from [19] for task execution sub-contracting

The *Contract Net protocol* allows an initiator Global Scheduler to sub-contract tasks to other GSs. The GS requests proposals by advertising a call for proposal using the performative <*cfp*> for task execution to its acquaintances. The initiator GS then acts as the manager of the task. The call for proposal specifies the task to be executed and the condition the manager places upon the execution. Conditions include requirements such as time or performance constraints, needed service, etc. Performance constraints describe metrics such as CPU load needed, storage capacity, bandwidth needed, etc. Time constraint describes a desirable period to complete the task execution. The constraint values may be fixed or bounded such as lower and upper bound. See example of FIPA-ACL call for proposal (Figure 5). The Global Schedulers receiving the call for proposal can either propose to execute the task under certain conditions (the price, the time, etc), or refuse to propose. The Manager receives proposals from Global Schedulers, evaluates them and chooses the Global Scheduler that maximizes its objective (to minimize time, to minimize cost, etc). An acceptance message will be sent (<*accept-proposal*>) to the chosen GS and a rejection message (<*reject-proposal*>) will be sent to the other GSs. Once the Manager sends an acceptance message, the GS has to perform task and to send back a completion message (<*inform-done*) when it performs task successfully or a message (<*failure*>) when it fails to execute the submitted task. Figure 6 shows an instantiation of the Contract net protocol [19] for task execution subcontracting. The use of the Contract Net is justified since it is one of the most flexible and efficient negotiation mechanisms and it eases tasks distribution.

The Task Allocation Protocol allows the Global Scheduler to allocate tasks to its Local Schedulers. The task allocation protocol depends on the Global Scheduler objective (minimizing the time execution, load balancing between domains, etc.). It selects the best Local Scheduler according to that objective. The LS has the possibility to accept or reject the transmitted task according to its policy, and to the state of the resources it manages. Once it has accepted a task, if some problem occurs, the LS may have to perform some adaptation. This protocol is shown in Figure 7.

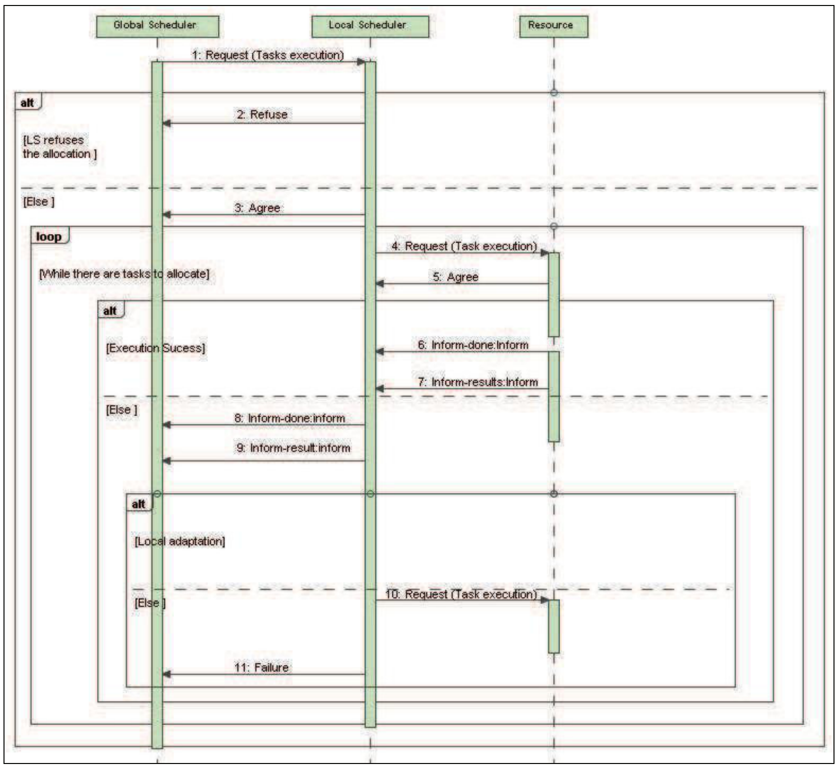


Fig. 7. The task allocation protocol (AUML sequence diagram)

2.3 Specification of Agents Behaviour and Their Interaction Protocols

The Petri Net of figure 8 provides a detailed view of interaction protocols between the Grid schedulers and their local behaviours in order to perform an application. This Petri Net shows how a Global Scheduler (called Manager) interacts with other Local and Global Schedulers. This Petri Net has been specified and simulated with the renew platform. The Petri nets [15] are known to have the adequate expressive power to describe behaviour (control structure, internal actions and communication) of distributed and parallel systems. Moreover, they provide several advantages such as a well-founded semantics, an executable specification, and a mean to enable simulations and proofs of behavioural properties.

This whole behaviour is made of four interconnected sub-nets:

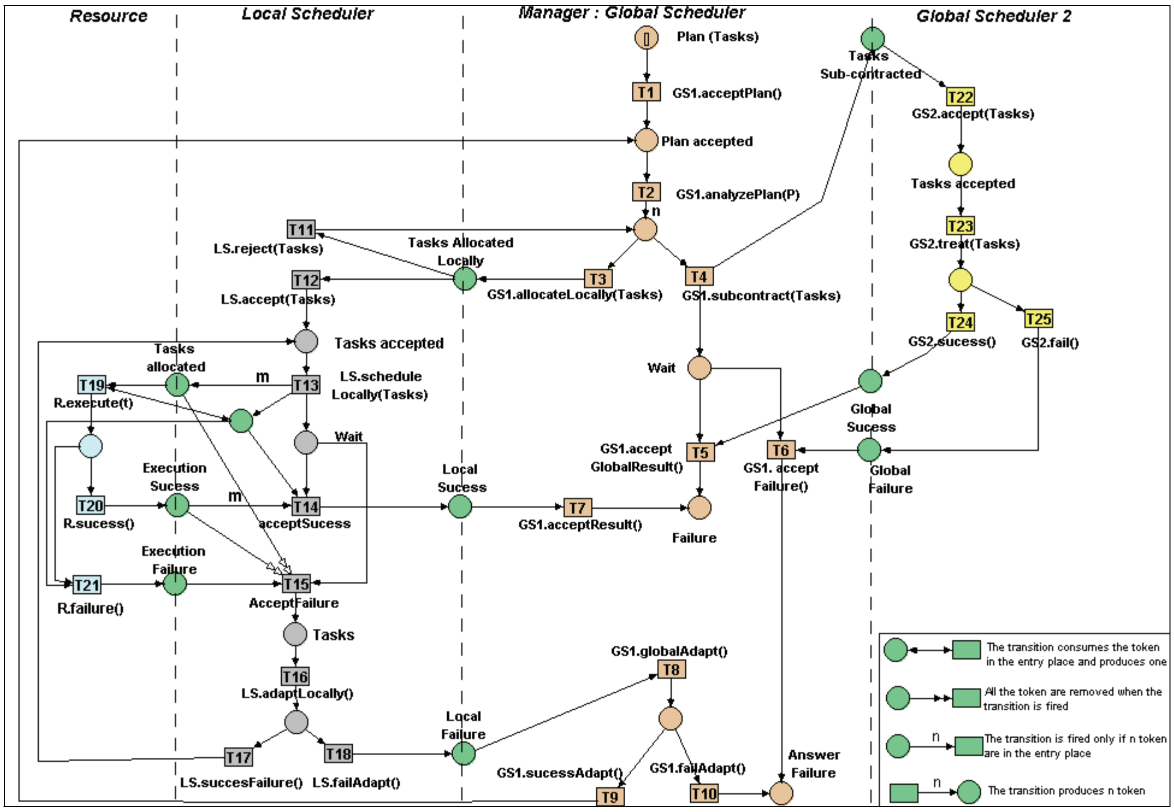


Fig. 8. Petri Net global scheduling functioning

- The Manager is a Global Scheduler in charge of a given user application made of several coordinated tasks (Plan of tasks). The Manager's part of the net is composed of transitions T1 to T10. Once the Manager accepts a plan (T1), it decomposes it into sub-plans (T2) and allocates them to a Local Scheduler (T3) and/or sub-contracts them to another Global Scheduler (T4). If the sub-plan is allocated to the Local Scheduler, the result could be a success stored by the transition T7 in the "Answer Success" place. On the contrary, if the Local Scheduler fails, a global adaptation is performed (T8). This adaptation could fail, and the final result is put in the "Answer failure" place, or could succeed and then a new loop is started by putting the sub-plan in the "Plan Accepted" place in order to schedule it (T1) again and so on.
- The Local Scheduler and resource's part of the net, made up of the transitions T11 to T21, describes the behaviour of a Local Scheduler (LS) and a resource in a local domain. The LS could accept (T12) or reject (T11) an allocated sub-plan. If the sub-plan is rejected, the Manager will try to allocate it again to another LS

(T3) or to sub-contract it to one of its acquaintances (T4). If the LS accepts to treat the sub-plan (T12), the LS schedules it in the local domain (T13). It consists in allocating a resource to each task of the sub-plan and then the execution by the resources could take place. The execution on resources could succeed. If the execution of all the tasks succeeds, the result is put in the "Local Success" place. In any other case, the tasks execution is interrupted and a local adaptation is performed (T16). If the local adaptation succeeds, a new loop is repeated.
- The right part of the net, made up of the transitions T22 to T24, is a view on the behaviour of a Global Scheduler acquaintance (called GS2) playing the role of a sub-contractor for a GS. After the reception of a sub-plan ("sub-plan sub-contracted") place, GS2 will treat it (T22). T22 is an abstract transition which could be expanded to be compliant with any Global Scheduler behaviour (Manager's net). The treatment could succeed and be put in the "Global success place" or could fail and be put in the "Global Failure" place.

It is important to remark that the transitions T3 and T4 are also abstracted transitions corresponding respectively to the tasks allocation and to the contract net protocols as described in section 2.2. The Global Plan (T2) represents the internal behaviour of the Global Scheduler.

3 Typology of Adaptive Grid Scheduling

The Grid environment can be disturbed by various events: resources arrival/departure, resource crashes, priority task arrival, overloads or tasks competition for a resource, etc. The adaptation consists on a set of actions to undertake to maintain or improve a nominal functioning in the case of perturbations. These perturbations are mainly related to resources while the adaptation can relate to various Grid entities described in the conceptual model previously presented (figure 2).

In this section, we define a typology of disturbance events and a typology of adaptations.

3.1 Typology of Disturbance Events

- *Resources entrance/departure.* The Grid environment is an open environment that authorizes the resources to join or leave the Grid freely which can disturb the tasks scheduling and execution. The arrival of new resources disturbs the Grid organization at first but it can be useful to optimize and improve grid performances.
- *Prior tasks arrival.* A task submission (higher priority, more profitable, local, etc.) could modify the order of tasks execution and disturbs tasks allocation.
- *Resource crashes.* It corresponds to the situation where a resource is temporarily unavailable. This problem can occur on a storage device necessary for input data retrieval, on a compute node on which the task is running or on the networks connection (network congestion, overload, etc.). Resource crash can deteriorate

the offered quality of service and has many consequences that we detect using our model (task state change, resource state change, organization restructuring, etc).
- *Resource overload*. It is mainly due to applications competition for resources at the execution time. In fact, tasks could be carried out in parallel on the same resource or waiting for resource availability. However a task could act in an unpredicted way by consuming more resource or time and thus hinders the execution of other competing tasks.

3.2 Typology of the Adaptations

We introduced in the previous section the disturbing events. Here we identify the possible types of actions. The types of adaptation are different depending on several criteria: *temporal* (reactive, pro-active), *dynamics* (static or emergent), the *adaptation objects* (plan, process, resources organization, scheduling policy, etc.).

3.2.1 Reactive/Pro-active Adaptation

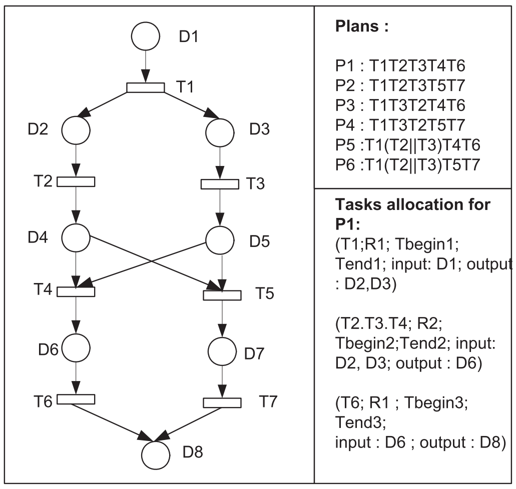
- *Reactive adaptation*. It assumes a set of corrective actions that are executed as soon as the scheduler detects a disturbing event.
- *Pro-active adaptation*. It assumes a permanent monitoring of the environment which allows to the scheduler to act in an opportunist way. For example, if a resource were requested a long time, it would be judicious to reduce its load. If two resources function in an optimal way when they are associated, it could be beneficial to put them in an organization, etc.

3.2.2 Static/Emergent Adaptation

- *Static adaptation*. It corresponds to an adaptation whose rules are predetermined. For example, we quote the following rule: If "disturbing Event = resource breakdown "Then" compute new tasks scheduling plan".
- *Emergent adaptation*. It assumes that the scheduler has the capacity to produce new adaptation rules.

3.2.3 Adaptation Objects

- *Application process adaptation*. In order to illustrate our remarks, we can formally represent an application process with a Petri net graph [15] where tasks correspond to the transitions, the data necessary for tasks execution to the input places and the data produced to the output places of the corresponding transition. The network structure shows the tasks coordination and can describe various control structures: sequence, alternation and parallelism. From these control structures, we can deduce several execution plans. The adaptation of the application process consists in giving a higher priority to a plan compared to another, removing certain plans that are not realizable in the current context, etc. These adaptations result in the modification of the corresponding network structure of the application process.



Plans :

P1 : T1T2T3T4T6
P2 : T1T2T3T5T7
P3 : T1T3T2T4T6
P4 : T1T3T2T5T7
P5 :T1(T2||T3)T4T6
P6 :T1(T2||T3)T5T7

Tasks allocation for P1:
(T1;R1; Tbegin1; Tend1; input: D1; output : D2,D3)

(T2.T3.T4; R2; Tbegin2;Tend2; input: D2, D3; output : D6)

(T6; R1 ; Tbegin3; Tend3; input : D6 ; output : D8)

Fig. 9. Application process modelling by a Petri Net

Figure 9 shows an application execution modelled by a Petri net. This application is made up of seven tasks, T1 to T7. The network structure expresses the precedence constraints between tasks. For example, the tasks T2 and T3 need respectively the input data D2 and D3 and can be carried out in parallel. The tasks T4 and T5 need the input data D4 and D5 and are mutually excluded. T5 and T6 are carried out sequentially. Using the following process, a set of all possible application executions plans (P1, P2, P3, P4, P5 and P6) is derived. Scheduling will consist in choosing one of these plans and allocating resources to these tasks. In our example, the plan P1 is selected, and its tasks are assigned to the resources R2 and R1 by specifying the start time, the end time, the input and the output data. We notice that tasks T2, T3 and T4 are associated to the resource R2 to be executed on.

– *Rescheduling*. In the case of disturbances, the tasks to resources mapping must be revised. As a result, a new scheduling plan is computed and carried out by the GS. This type of adaptation requires that the Grid infrastructure offers mechanisms for tasks migration, checkpointing, etc. Most of the adaptive scheduling systems apply this type of adaptation [4], [20], [21].

– *Scheduling policy adaptation*. The Global Scheduler uses scheduling policies which can be fixed or variable and dependent on the resources and their states. The adaptation consists here in modifying this policy when a disturbance occurs. For example, we can quote the change of the use of static scheduling algorithm that uses performance prediction information, to a dynamic algorithm in order to balance the results of the static one and take into consideration the current resources states. Another example is when following the prior tasks arrival, the GS decides to broadcast a call for bid to find the best offer (related to the quality of service, cost, execution time, etc.) instead of using a traditional push or pull policy.

– *Quality of service adaptation*. It assumes at first defining the criteria of quality of service (economic profit, resource utilisation, execution time, application cost execution) which can be quantitative (100 Mb/s or 10 ms) or qualitative (high, average, weak), a range of values for each criteria or for a combinations of these criteria and a certain threshold for these values. If the monitored value exceeds the defined threshold, the system must adapt to maintain it or negotiate to release this threshold [22].

– *Organisational structure Adaptation.* Some resources are not isolated but belong to an organization in which they hold a role (function or type of service). Kreaseck [23] uses a scheduling algorithm that aims to search for a group of candidate machines in order to find an application to a group of resources mapping. The resources in the same site or the same administrative domain are pooled in the same organization (communication time in each subset is lower than between the subsets). However, grid resources can have sufficient intelligence to form coalitions in order to execute an application collectively that the resource could not carry out alone. In this context, the adaptation consists in reorganizing these organisational structures (resources adding or removal, roles modification, etc.). Within the same organization, groups of resources can be formed on the basis of the observation of their collective performance. For example, one can notice that the resources R and S joined together carry out in a more optimal way the application A than the resources S and U, that can lead the GS to give a preference to the first group of resources when the application A will be executed. In the case of coalitions, the resource takes the initiative of modifying its dealings according to the last co-operations and the results obtained with each one of its dealings. This can be managed by reputations mechanisms as the mechanisms used in project CONISE-G [24].

4 An Organizational Model for Grid Scheduling

We have introduced in the previous section a typology of perturbation events and actions that could be undertaken in order to adapt. In our work, we choose to focus on adaptation based on an organizational approach. For that purpose, we have modelled the grid scheduling system with an organizational view. In fact, this approach offers several advantages for modelling the grid system in general and the grid scheduling in particular. In fact, the design and implementation of a grid scheduling system is a difficult task due to the grid environment constraints (resources distribution, autonomy, performance variation, etc.). The organizational perspective constitutes a design support since it makes it possible to apprehend and to structure a multi-agents system (MAS) through various roles and their interactions. Moreover, the grid scheduling system operates through the co-operation of many interacting subsystems. The organizational perspective structures the MAS execution since it defines, through the attribution of roles to the agents, behavioural and interaction rules to which the agents must conform. Finally, the grid environment is an open and dynamic environment. The organizational perspective allows the design of open systems with heterogeneous components where agent internal architectures are not specified.

The system can thus be adapted to an increased problem size by adding new roles and groups, and this does not affect the functionality of the other agents.

The remainder of this section first presents the Agent Group Role (AGR) meta-model of Ferber [11]. In fact, our organizational model for the grid scheduling is based on the AGR meta-model which is appropriate to the grid context. Then, we present how we apply this meta-model in order to structure and organize our grid scheduling system.

4.1 The AGR Meta Model

In this section, we describe the AGR Meta Model [11] which core concepts are Agent, Group and Role.

- An ***agent*** is defined as an active communicating entity that can participate in several communities (groups) in parallel. It can play one or several roles corresponding to its activities or interactions in each group. No constraint is placed on the internal structure of the agent.
- A ***group*** is defined as a set of agents. Agents can belong to different groups. The communication between agents is possible only if they belong to the same group space. Communication between two groups is made by agents that belong to both.
- A ***role*** is a representation of an agent function. Agent may play one or several role within a group.

AGR has several advantages useful in our context: it eases the *modularity* through the organization of tasks carried out by agents and their interactions in logical entities: working group, organizational unit or private space of conversations. The organization of the tasks carried out by the agents in logical entities enhances the *security* of applications. In fact, in the AGR model, security mechanisms such as entrance protocol, authorizations and permissions can be integrated in order to define how an agent can enter, leave, and behave inside the organization. As a result, agents not belonging to a group cannot listen or join freely in a conversation in this group. This allows secure interactions inside and between groups. Moreover, at the role level, describing norms such as obligations, permissions, interdictions, etc. allows to prevent unauthorized actions to be executed by agents. Also, AGR allows the design of *open* systems such as grid. The system can thus be adapted to an increased problem size by adding new roles and groups and this does not affect the functionality of the other agents. The system can also be designed with *heterogeneous* components where agent internal architectures are not specified. Finally, the *reusability* is facilitated since the organizational approach identifies, expresses and makes available recurring interaction patterns which become reusable.

4.2 An AGR Organizational Model for Grid Scheduling

In this section, we describe our organizational model which identifies the actors implied in our system, the links that exist between them, and the protocols that rule their interactions.

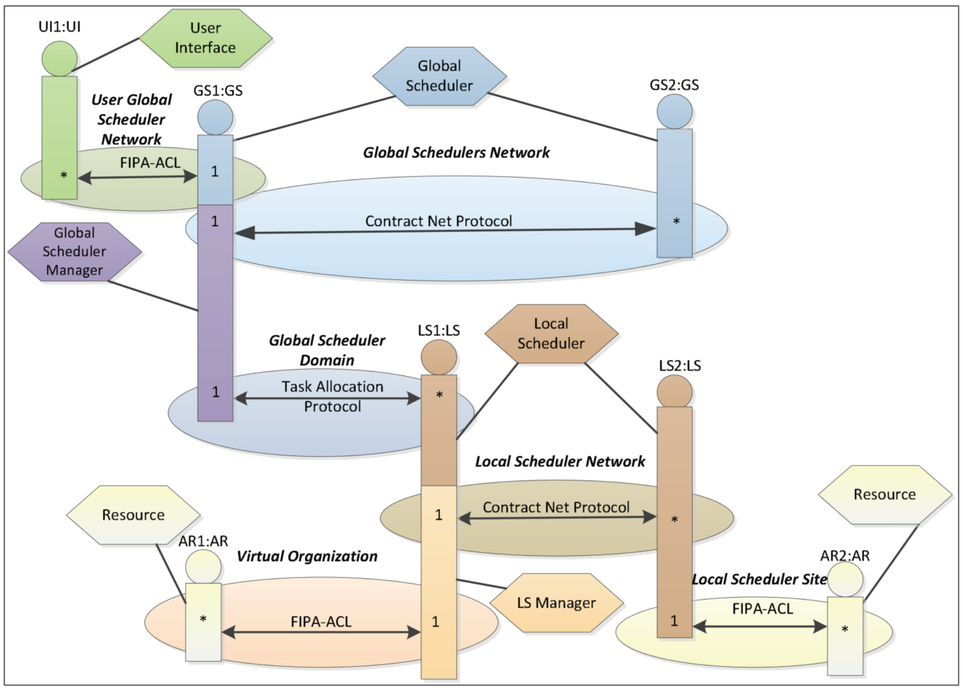


Fig. 10. Organizational multi-agents based grid architecture

Our organization model (see figure 10) organizes the architecture' agents introduced in section 2.2 around the following components:

- Four types of agents represented by a candle that are: the User Interface Agent, the *Global Scheduler Agent*, the *Local Scheduler Agent* and the *Resource Agent*. The multiplicity of the agents participating to a group is represented by a star inside the candle.
- Six types of groups represented by an ellipse that are: the Users *Global Scheduler Network,* the *Global Schedulers Network*, the *Global Schedulers Domain*, the *Local Scheduler Site*, the *Local Schedulers Network* and the *Virtual Organization*.
- Six types of roles since each agent can plan different roles in different groups. The defined roles are: the *User Interface Role*, the *Global Scheduler Manager Role* that is a GS playing the role of tasks manager, the *Global Scheduler Contractor Role* that is a GS acquaintance responsible of treating subcontracted tasks, the *Local Scheduler Role,* the *Resource Role* and the *Local Scheduler Manager Role*. The Local Scheduler Manager is responsible for managing a virtual organization. In fact, as we previously mentioned the GS allocates tasks on selected resources that will form a virtual organization made by multiple distributed resources for application tasks execution. When a virtual organization is formed, an agent playing the role Local Scheduler Manager is launched by the

Global Scheduler in order to orchestrate the different tasks in the virtual organization and to adapt the organization functioning in case of disturbances. The adaptation includes recruiting a new resource to participate to the organization, excluding resources that slowdown the execution performance, rescheduling tasks in the organization, etc. Role is represented as a hexagon and a line links this hexagon to agents.

The communication between agents in different groups follows the FIPA-ACL language that supports high level communication between the different grid components (see the section 2.2 for more details).

Let us detail how does each group operates.

1. The *User Global Scheduler (GS) Network* Group allows user interface agents to ask for application execution, to submit user applications and to collect the execution results.
2. The *Global Scheduler (GS) Network* Group. This group allows the GS to cooperate in order to allocate application's tasks efficiently when resources under a Global Scheduler control are not available or in the case of disturbance. In this context, Global Scheduler (GS) agents cooperate in order to sub-contract the execution of application's tasks using the *Contract Net Protocol* [17] (see the section 2.2 for more details).
3. The *Global Scheduler Domain* Group is composed by the Global Scheduler and the Local Scheduler agents at its disposal. This group allows the Global Scheduler to allocate the application's tasks to the adequate Local Scheduler to be executed on its local resources. The allocation is based on the Global Scheduler objective, resources states in the local domains and tasks requirements. Moreover, this group allows the Local scheduler to send execution results, resources states and to inform its Global Scheduler if some problem occurs on resources to allocate the failed tasks on another resource.
4. The *Local Scheduler Site* Group is composed by the Local Scheduler (LS) and the resource at its disposal (in the local site). This group allows to the Local Scheduler to allocate the submitted tasks on the local resources, to collect the resources information (availabilities, breakdown, task execution progress, etc.), the execution results, etc.
5. The *Virtual Organization* Group is composed by selected resources for the application execution and by Local Scheduler agents. In fact, in each formed virtual organization, a Local Scheduler Manager coordinates the functioning of resources inside the organization, monitors their execution and undertakes adequate action to adapt in case of perturbation (resource breakdown, resource overload, etc.). In case of perturbation, adaptation consists in reorganizing the virtual organization by adding a new better resource using the contract net protocol, removing an overloaded resource, replacing resource by another, etc. The adaptation results in the modification of the structure of the virtual organization.

6. The *Local Scheduler Network* Group is made by local scheduler agents in the same virtual organization. This group allows to the Local Scheduler Manager to cooperate efficiently with other Local Scheduler in order to add new resources for tasks execution in case of perturbation and when resources at the LS Manager disposal are not available, busy, broken, etc. The LS cooperate using the Contract Net Protocol described above.

In the introduced groups, adaptation is made thanks to the use of high-level protocols, like contract net protocol that eases task allocation even if the grid environment evolves. The use of interaction protocols allows replacing one resource by another, sending tasks to be executed by acquaintance, etc. Here, the structure of the organization remains the same.

5 The Organizational Grid Scheduling Simulator "OGSSim" with the AGR Meta Model

We have implemented an organizational grid scheduling simulator based on the AGR meta-model using Java and the Madkit [12] platform that (or which) integrates the AGR meta-model.

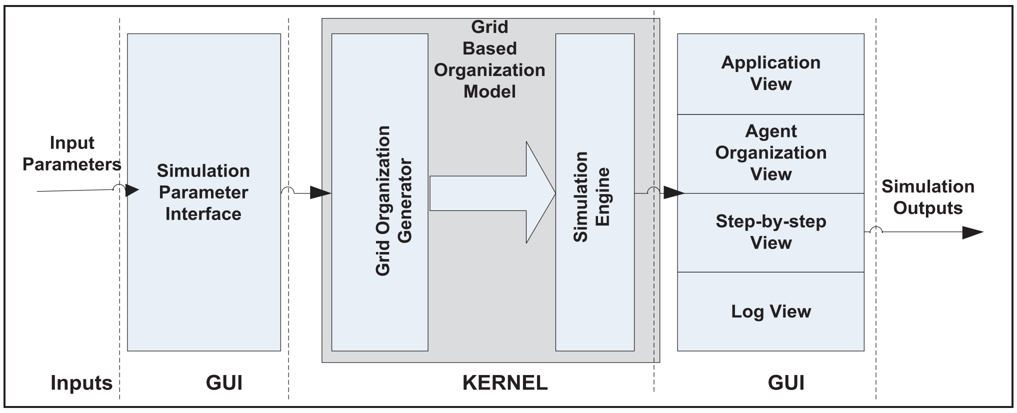


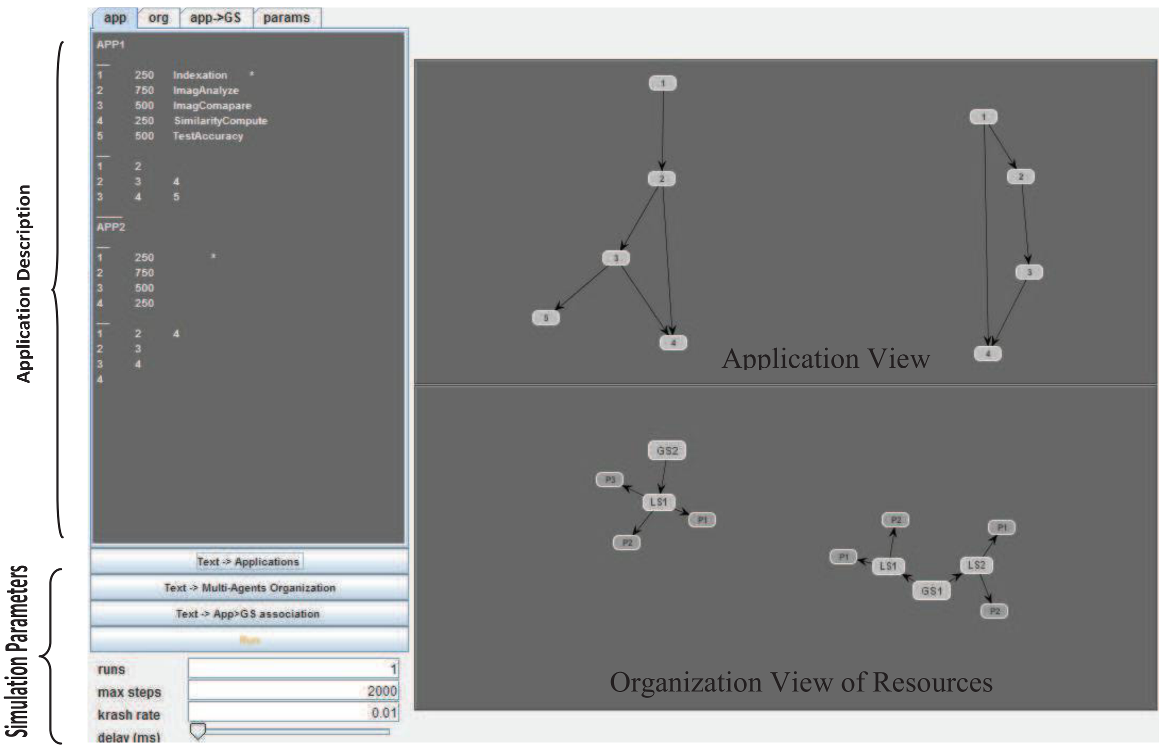
Fig. 11. Multi-agents based grid simulator

Our simulator offers the following services:

- It allows users to enter parameters for the simulations,
- It simulates the applications' tasks scheduling and their execution step by step,
- It finally outputs the statistical data of the performance metrics (execution time, number of messages, etc.) on a log file.

The structure of our simulator is illustrated in figure 11. It includes an *input GUI,* a *kernel* and an *output GUI.*

Figure 12 shows some screenshots of our GUI. We distinguish between two types of GUIs: the *input GUI* (left side Fig.12) and the *output GUI.*



```
APP1
__
1       250     Indexation      *
2       750     ImagAnalyze
3       500     ImagComapare
4       250     SimilarityCompute
5       500     TestAccuracy
__
1       2
2       3       4
3       4       5
____
APP2
__
1       250             *
2       750
3       500
4       250
__
1       2       4
2       3
3       4
4
```

```
runs          1
max steps     2000
krash rate    0.01
```

Fig. 12. The simulator's GUI

- The *input GUI* allows the user 1) to define the *grid specification*, the *application description*, the *simulation parameters* and 2) to display this data in a graphical form.

1. The first service concern defining the simulated environment and more precisely:

- The *grid specification* described by:
  - The *resources* (number, identifier) with their computing capacities rated as MIPS (Million Instructions Per sec) and theirs skills (types of tasks they are able to perform).
  - The *physical resource organization* (in a domain, site, etc.) and thus the number of Local Scheduler and Global Scheduler in the simulated system. It is described thanks to the second slot called "org" in our input GUI (Fig. 12).

- The applications are described by their tasks, their coordination and their submission time. For each task, we provide the number of instructions, the type and tasks' coordination process (preceding constraints). The top left side of Figure 12 shows an example of two applications submitted to the grid. The first one is composed of five tasks and three preceding constraints (the constraints line 1 means that tasks 1 must be performed before task 2 and 4). Each task has a certain number of instructions and a type. The second application is composed of four coordinated tasks, etc.

- The three simulation parameters are described by the grid resource failure's probability (which is defined as the expected number of failures per step which is a discrete unit of time), the number of runs and the maximum execution step (bottom left Fig 12).

2. The second service concerns *input GUI* allows also displaying an application view (top right figure 12) of the applications entered by the user and an organization view (bottom right Fig.12) of the resources. For example, the organization view shows two directed graphs in which vertex represent the Global Scheduler, local scheduler and resource an edge the relation between them.

- The *KERNEL* has two roles. It generates the agent based organization model from a user's inputs and it launches the execution of the simulations. More precisely, the grid organization generator (figure 11) transforms grid information (Grid Resources, Global scheduler, Local Scheduler) into an agent-organization made of Agent-Group-Role model. For example, the grid components (resource and organization) are converted into Resource, Local Scheduler and Global Scheduler agents, each agent belong to one or several groups (Global Scheduler Network, local Scheduler Site, etc.). Then, the simulation engine launches the execution of the agents. Since applications are modelled as processes, each Global Scheduler agent behaves as a workflow engine (or orchestrator). Finally, when the simulation ends, results are displayed to the user on the *GUI output*.
- The *output GUI* displays the simulations results on a log file. The structure of this file is a tuple of the following form < Run Number, ApplicationID, Application Start time, application End Time, TaskID, Task Start Time, Task End Time, Number of exchanged Messages>.

6 Evaluation of Our Model

6.1 Performance Metrics

In this section we introduce the performance criteria used in our approach and the evaluation of our organizational model for grid scheduling based on these criteria.

Our study covers two aspects: the *structural* and the *statistical* aspect:

- The *structural level* concerns the communication topology of the organization and is based on the Grossi's structural measures [13] that are described below. These measures allow to proof organizational qualities such as the flexibility (the capacity of the organization to adapt in a flexible way to changing circumstances), the robustness (how stable the organization is in the case of anticipated risks) and the efficiently (refers to the amount of resources used by the organization to perform its tasks) by using relations holding between roles. In our case, we will evaluate this qualities and proof that our model is sufficiently robust and the flexible. These two qualities are the most important in our grid scheduling context targeted to execute users' applications efficiently while taking into account the dynamicity and complex interactions among the different components.

- The *statistical level* concerns the evaluation of our organization model performances at run time. The used metrics are intended to observe the functioning of our organizational based grid in the case of disturbances to proof its robustness. To do that two metrics are observed related to the grid and to multi-agent system (MAS). The robustness of a system is defined as the ability of the system to maintain its functioning in highly dynamic environments such as the grid.

  — *The grid metric* corresponds to the application execution time in a scenario without perturbation and in a scenario with perturbations.
  — *The MAS metric* corresponds to the number of exchanged messages that is commonly utilized for evaluating multi-agents system in a scenario without perturbation and in a scenario with perturbations [14].

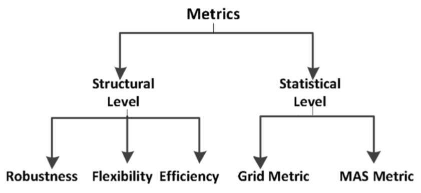


Fig. 13. Performance metrics

Let' us details the ***Structural level*** based on Grossi's [13] measures. Evaluating the organizational structure property involves three steps:

3. The first step consists of building a role graph of the organization based on the possible relations between two roles. To do that, Grossi introduces three dimensions characterizing relations:

- The *power structure* defines the task delegation patterns (existence of power link between agent *a* and *b* means that every delegation of tasks from agent *a* to agent *b* ends in creating an obligation to agent *b*),
- The *coordination structure* concerns the flow of knowledge within the organization.
- The *control structure* deals with the task recovery functions, that means that an agent "*a"* that controls one another "*b*" has to monitor its activity and possibly take over the tasks which agent *b* has not accomplished.

4. Secondly, for *each dimension* (power, coordination, control), a set of concepts and equations from the graph theory are applied in order to measure specific property of organizational structures (OS). Grossi defines the following concepts:

- The *connectedness* of an OS shows the connection degree between roles. The more this degree is high, the more the structure can be divided into fragments.
- The *economy* of an OS expresses how to keep the structure connected while minimizing the number of links between roles (for example redundant links must be avoided).
- The *univocity* allows us to have an idea on the degree of ambiguity in the organization structure. For example stating that an agent "a" controls an agent "b"

and in the same way, agent “b” controls agent “a” generates some ambiguity. It is a ratio between the number of roles which have exactly one link in the same structural dimension and the total number of roles.

5. Finally, these previous concepts are compared with optimum values defined by the author (See [13] for more details) in order to measure the qualities (robustness, the flexibility and the efficiency) of the organizational structure. In our context, we will calculate these measures and compare them to the optimum values to evaluate the qualities of our model.

6.2 Evaluation of Our Organizational Model

In this section we describe the structural and statistical evaluation (experimentation results) of our organizational model.

6.2.1 Structural Evaluation

According to the dimensions described above (power, coordination, control), we generate the role graph (figure 14) corresponding to our organizational model. Our graph includes the roles described in section 4.2.

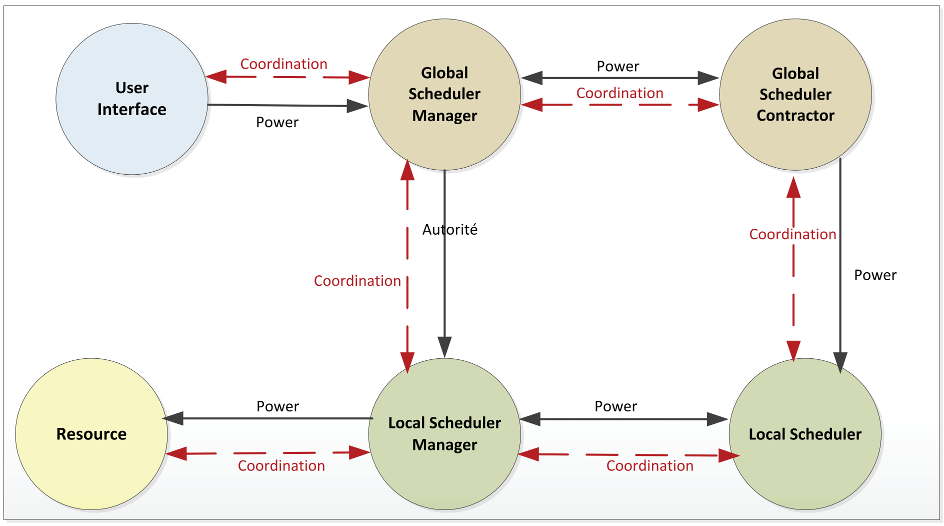


Fig. 14. The roles graph

Our graph includes only the power and coordination dimensions represented as edges:

- *Power* expresses the relation which can occur between an agent “delegator” and the one who receives the delegation “delegatee”. The delegation is a mechanism allowing the delegator to assign tasks to the delegatee. In our context, the user interface delegates tasks to the Global Scheduler Manager. The Global Schedulers (Manager and contractor) can delegate tasks to the Local Schedulers under their control and to their GS acquaintances. The Local Schedulers (including the Manager) has a power relation with the Resources under their control and with their Local Scheduler acquaintances in the same virtual organization.

- *Coordination*: In our graph, the coordination is a symmetric relation. It consists in information exchange or protocols-based interaction between roles. It concerns the same couple of roles as the power relation.

The *Control* relation is not present in our organization, since we do not model the evaluation of agent's behaviour by their acquaintances. In future work, we intend to refine our model to include this dimension in order to improve the adaptation process. This assumes to model the continuous monitoring of the agent work to check whether agents are doing their tasks as agreed.

Following this role graph and the set of equations introduced in [13], we compute the three following organizational qualities: the robustness, the flexibility and the efficiency as shown in the following tables:

Table 1. Structural measures of robustness

Optimum Values	**Economy $_{\text{Coord}}$**	**0**	**Univocity $_{\text{Power}}$**	**0**	**Connect$_{\text{Coord}}$**	**1**
Obtained Values		**0,72**		**0.2**		**1**

Table 2. Structural measures of flexibility

Optimum Values	**Connect$_{\text{Power}}$**	**0**	**Connect$_{\text{Coord}}$**	**1**	**Economy$_{\text{Coord}}$**	**0**
Obtained Values		**0,64**		**1**		**0,72**

Table 3: Structural measures of efficiency

Optimum Values	**Economy $_{\text{Power}}$**	**1**	**Economy $_{\text{Control}}$**	**1**	**Economy $_{\text{Coord}}$**	**1**
Obtained Values		**0,96**		**1**		**0,72**

Comparing our obtained results to the optimum ones, we can state that our system is sufficiently *robust* and *efficient* but not enough *flexible*. Lets' us explain these results:

- *Robustness* of an organization requires a coordination structure highly connected and weakly economic and a power structure weakly univocal (see Table1). In our organization, roles are well connected (Connectedness$_{\text{Coordination}}$ value equal to the optimum) while avoiding ambiguities (Univocity$_{\text{Power}}$ value approaches the optimum). An optimal robustness would require a complete connectivity between all nodes. Due to the privacy and local management policy, this complete connectivity is not possible in a grid context. For example each resource in a specific domain has its own management policy and doesn't authorize the Global Scheduler to interact directly with it.
- *Flexibility* of an organization is related to the ability to easily adapt. So, in order to enhance the flexibility, a low degree of connectedness for the power dimension and a high degree for the coordination dimension are needed. Considering the connectivity aspect, while the power relation restrains the flexibility (the delegation pattern imposes constraints on the interactions), the coordination one eases the interactions within the organization: we reach a

trade-off (see Table 2). Considering the economy aspect, we have an average value which corresponds to a moderate flexibility. To obtain a lower value of the $\text{Economy}_{\text{Coordination}}$, for more flexibility, we should increase the redundancy of the coordination relation which is as previously noticed incompatible with the privacy constraint of the grid context.

- *Efficiency* of an organization can be obtained if it is economic (value equal to 1) in all the dimensions. In our case, the roles are well connected while avoiding redundant ones and we approach the optimal efficiency. The value of the $\text{Economy}_{\text{Control}}$ is equal to the optimum one since we don't take into consideration this dimension. The $\text{Economy}_{\text{power}}$ is quite optimal while the $\text{Economy}_{\text{coordination}}$ has an average value.

It is important to notice that the framework of Grossi [13], used for analysing organizational structures, does not take into consideration the number of roles' instances and the links among them. However, in our context, introducing the roles' cardinalities could be more significant and realistic. For example, a broken resource doesn't imply that the resource role dysfunctions, but this specific instance is out of work and could be replaced by one another.

Also, obviously, designing an organization that maximizes simultaneously the three qualities cannot be reached. For example, as we explained above, the more the organization' nodes are well and directly connected, the more the organization is robust and flexible and the less it is efficient.

Since our organization is devoted to the grid scheduling system, we can claim that the qualities obtained (sufficiently robust and efficient) of our organization are adequate for this context. Decreasing the number of power relations can improve the flexibility but it will decrease the robustness.

As our context is highly dynamic and characterized by a frequent resources performances fluctuations and crashes, we are also interested in the capacity of our organization to be adaptive. The adaptation is the ability of the system to react against the environment changes. The adaptability cannot be measured using the previous framework. In the next section, we measure the quality of our model experimentally by simulations and more precisely its adaptability. For this purpose, we compare the nominal functioning of our system and its functioning with disturbances.

6.2.2 Statistical Evaluation

In this section, we describe our experiments and the obtained results.

Our experimental system is configured with twelve agents, illustrated by the figure 15. The Global Schedulers are represented with a square (GS1, GS2), the Local Schedulers are represented with an ellipse and resource with a rounded rectangle (including the CPU speed). The following grid specification is simulated and organized according to our proposed AGR model described in section 4.2.

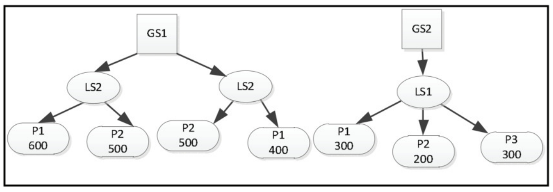


Fig. 15. Case study: agent's organization

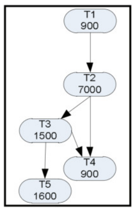


Fig. 16. Case study: application description

The goal of these simulations is to demonstrate the feasibility (figure 17) and the efficiency (figures 18, 19, 20, 21 and 22) of our model and more precisely of its adaptation.

To demonstrate *the feasibility of our proposition*, we computed the number of successful executions of applications in the case of adaptation comparing to the case with no adaptation. To do that, we considered four applications (figure 16 shows an example of an application description) and vary the resources' failure probability. We measured the percentage of successful runs while varying this failure probability. The total number of run is fixed to 100.

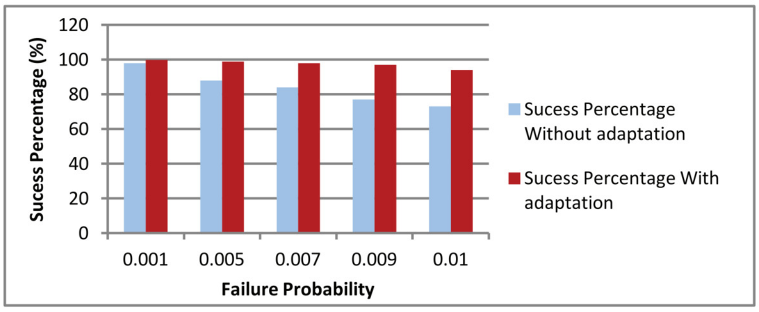
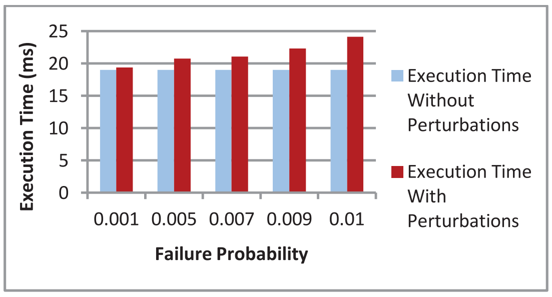


Fig. 17. Execution success percentage

The results show the efficiency of our adaptation model since we obtained better percentage of successful executions. We can however notice that we do not reach 100% of success since even with adaptation we can meet situations where some tasks could not be executed by lack of specific resources.

To demonstrate the *efficiency of the adaptation* of our model, we made three experiments. In the first one, we considered only one application and vary the resource's failure probability. Figure 18 compares the application execution time with and without disturbances. The results show that the adaptation phase is not costly (at the maximum 27.05 % of the application execution time). Figure 19 shows how the number of communications increases as the failure probability raises. This evolution is due to the fact that adaptation is mainly based on interactions.



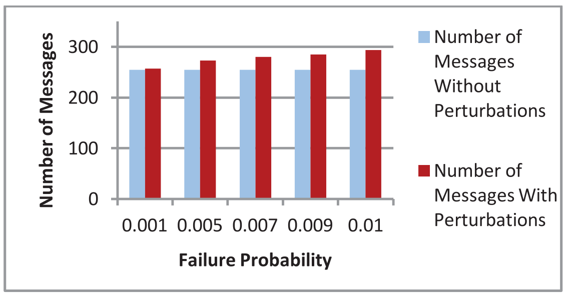
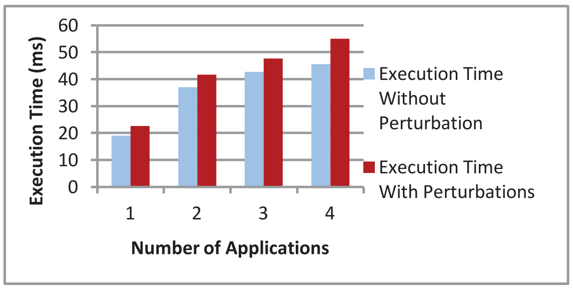


Fig. 18. Comparison of application execution time

Fig. 19. Comparison of number of messages

The second experiment varied the number of concurrent applications executed by the system with a fixed resource's failure probability (0.01). Figure 20 shows again that the adaptation time remains low even if the number of the number of applications increases (20.8% of the application execution time). Figure 21 shows that the number of messages increases with the number of applications in both cases: with or without perturbations. In case of perturbations, we noticed a higher but reasonable number of communications due to the adaptation phase.



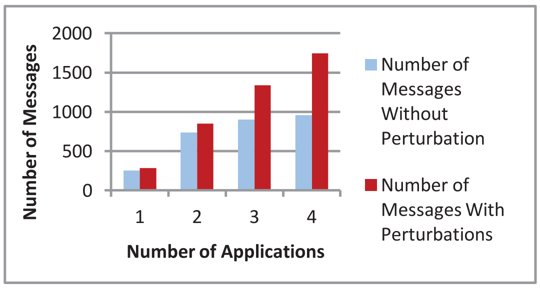


Fig. 20. Comparison of application execution time

Fig. 21. Comparison of number of messages

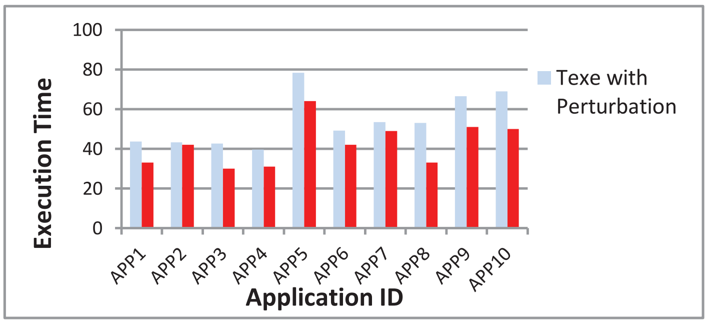


Fig. 22. Comparison of application execution time

In the third experiment, we increased the number of processors to fifteen, the number of Local Scheduler to forty and the number of Global Scheduler to ten. The aim is to check the system performance scalability. We considered ten applications. Each application arrived to the system every unit of time. We fixed the resource's failure probability to 0.01. Each result is the average value that is derived from 100 simulation experiments. Figure 22 compares the execution time for each application in the case of perturbations and the case where there is no perturbation. Figure 22 shows again that the adaptation time remains low even if the number of applications increases (the average is 20.86% of the application execution time).

7 Related Work

The problem of adaptation in grid computing has been investigated in many works in the literature. As a result, adaptive scheduling systems have already been proposed [20][23][25][26], but these systems show some restrictions.

The condor system [27] offers rescheduling and checkpointing mechanisms allowing the application to be restarted from checkpoints. Rescheduling consists in migrating the impacted application (or portion of application) to another resource when a perturbation occurs. Buyya et al. propose the Nimod/G [28] and Grace systems [29] that implement an economics model for grid resource management and scheduling. Marketing concepts such as commodity market, posted price and bargaining modelling are proposed for resource trading and scheduling. Moreover, the Nimrod/G and Grace systems incorporate schedule adaptation in order to meet several user QoS (quality of service) requirements (deadline or budget), and also to adapt to resource availability and performances. It is interesting to note that our proposition can be used to build a system like Nimrod/G since we use the contract net protocol for resource trading. However, in our system the scheduling component decides which tasks are to be executed at which site based on a certain cost function (in the current implementation, the global scheduler objective is to minimize the application makespan). Unlike our system, the Nimrod/G system supports user-defined deadline and budget constraints for application scheduling and manages the supply and demand of resources based on resources prices. Moreover, the introduced systems (condor and Nimrod/G) implement a centralized strategy. The scheduling decision is made by one centralized component. This approach has the advantage of easier implementation, but suffers from the lack of scalability, the possibility of generating a bottleneck and fault tolerance since the failure of the scheduling component will result in the entire system failure.

The system DIET [30] and Legion [31] implement hierarchical scheduling combined with rescheduling techniques. In the hierarchical scheduling [32], there is a central scheduler that distributes applications tasks to multiple lower-level sub schedulers. Unfortunately, even if the hierarchical strategy offers a better scalability than the centralized one, the failure of the central scheduler will also result in the entire system failure.

Unlike the hierarchical strategy, in the decentralized architecture there are multiple schedulers without a central one. Each scheduler has the responsibility to carry out its own portion of scheduling and can communicate with other schedulers to allocate tasks to another one with lower load. This naturally led us to choose a decentralized strategy in order to implement an adaptive grid scheduling more scalable than the centralized and hierarchical scheduling and offering naturally fault tolerance against failures.

In the decentralized scheduling, the scheduling components can work independently or cooperatively. In the independent scheduling, each scheduler acts as an autonomous entity and takes scheduling decision regardless of the effects of the decision on the rest of the system. The systems GraDs [4][20][33] and AppLeS [5] are two well-known adaptive grid scheduling systems that integrate rescheduling techniques. Both implement a non-cooperative strategy. In fact, the scheduler is tightly integrated in the application and optimizes its private individual objectives. As a result, these systems are not easily maintainable and are not easily applied to other types of applications. Here, a significant advantage of our system compared to the AppLeS and GraDs systems is the reusability of the scheduling architecture for various applications since the scheduling component is decoupled from the application to execute on the grid.

Our framework implements a decentralized and cooperative scheduling. In the cooperative scheduling, each scheduler makes scheduling decisions considering the other schedulers in order to achieve a global goal. Multi-agent systems (MAS) have been used with success for managing the grid environment in a decentralized and cooperative way. In fact, the benefits of MAS are widely recognized in the literature [9]. This paradigm is one of the promising approaches due to the characteristics of the agent like autonomy, proactivity, mobility and adaptability allowing to take naturally into consideration the complexity of the grid context. Moreover, MAS allows the design of software components that exhibit social capacities that could be used with benefits for managing the interaction among the grid components. As a consequence, a great number of systems applying the agent paradigm have been proposed such as the AgentScape [34], the AGEGC [35], the CONOISE [24] projects, the AGRD_NFRP [36] system, etc. Unfortunately, most of these systems neglect the adaptation process. However, we must mention the system ARMS [37] that implements an adaptive scheduling. The system ARMS [37] is an agent based resource management that uses a hierarchy of homogenous agents for scheduling grid applications efficiently. The system ARMS focuses on the dynamicity of the resource performances and the scalability of the grid. However, this work suffers from a lack of flexibility due to its hierarchical structure, which makes it unable to perform well in the rapidly changing environments of the grid. Finally, in [38], a MAS is proposed in which a set of event-condition-action rules is applied to guide the adaptation process. The adaptation is made either by reassigning the failed task to another agent for completion or by asking the user to relax its constraints. The advantage of this approach is its extensibility since adaptation rules can be added easily to the rule base. The limit is that evaluation and rescheduling are repeated iteratively at runtime which can incur a certain overhead and decrease the system performance. Moreover, all these systems impose an internal structure to the agents and do not consider their organization and regulation to reach the system objective. On the other hand, our system implements an organizational perspective allowing to structure and regulate the scheduling system and to improve its efficiency.

A last limitation that we can address to the systems presented above (GraDs, AppeLs, Nimor/G, etc.) is that the adaptation is limited to rescheduling. When a perturbation occurs on a resource, the adaptation consists in migrating the failed tasks to another resource to adjust to changes in resource availabilities. Obviously, as described in section 3.2 and in our previous work [38], other types of adaptation can be investigated such as application process adaptation, grid resource reorganization, quality of service adaptation, etc. In our work, adaptation is based on rescheduling, on resource reorganizations and on the use of high level interactions protocols. In fact, rescheduling is carried out by the local and global scheduler agents (see section 4.2). Reorganization is made by the virtual organization scheduler, in case of disturbances. Such restructuring includes integrating a new more suitable resource, excluding a resource slowing the execution, etc. Adaptation is made thanks to the use of the contract net protocol that eases task allocation even if the grid environment evolves (call for proposal to replace one resource by another, to send tasks to be executed by acquaintance, etc.). As a result the offered mechanisms ease the adaptation and make it more efficient.

8 Conclusion

Grid computing is known to be a heterogeneous, distributed and dynamic environment. In order to take fully advantages of computational grid power, grid scheduling must take into consideration the grid constraints (resource availability variation, prior task submission, breakout, etc.) and be adaptive. In this work, we proposed a conceptualization of the grid environment used to define a typology of the adaptation. The defined typology introduces the perturbing events and the possible adaptation actions. Adaptation can be related to the environment and to the software components with their interactions and organizations. In our work, we choose to focus on the adaptation according to an organizational view. More precisely, we introduce a framework for designing an evaluating a multi-agent organizational model for an adaptive grid scheduling. The framework includes an Agent Group Role (AGR) organizational model to describe and implement an adaptive grid scheduling system, an organizational grid scheduling simulator “OGSSim” and performance criteria to evaluate the proposed model. The proposed model has been validated: it has been implemented and its design and performance has been discussed. Following the Grossi’s framework we have evaluated its conceptual aspect and shown that it is sufficiently robust and efficient. According to our implementation, we have shown the feasibility and efficiency of our approach and more precisely measured the adaptability of our organizational model. In addition, the proposed system presents the following qualitative advantages:

Easy Design and Implementation of the Grid Scheduling System. The design of such system is a difficult task due to the constraints of the environment (complex interactions, autonomous and heterogeneous components, disturbed, etc.). The organizational perspective constitutes a design support and makes it possible to structure the overall functioning through the attribution of roles and interaction rules to which the agents must conform.

Openness. The grid is an open environment allowing the resources to enter and leave at any moment. The use of an organizational perspective allows the design of open systems such as grid since organizations are open structures allowing 1) to agents playing specific roles to enter groups without limits, 2) to add new roles and groups and this does not affect the functionality of the other agents.

Dynamicity. The grid is a dynamic environment were resources exhibit variable performances due to the openness of the grid, possible breakdowns, tasks executions concurrence, etc. The organizational model is well-adapted for such context since organizations are active entities that can be able to reorganize dynamically and to adapt to the environment changes.

Security. Security is a key requirement to ensure secure application executions on the grid and involves protection of data exchanges between tasks, controlling access to computers, data and other resources, etc. The organizational perspective allows the design and implementation of secure grid scheduling architecture. In fact, our proposed model organizes the grid components in groups where security policies such as protocols entrance (authorization and authentication procedures) can be integrated to restrict access to only authorized entities and keep the undesirable ones out of a group. Communications and data exchanges inside and between groups are also protected against malicious attacks since our agents are organized in private spaces and agents that do not belong to a group cannot listen or freely integrate in a conversation in this group, etc.

Reusable Framework. Our work is reusable and could be considered as a first step towards a framework for designing and testing grid organizations. Indeed, the designers could use our system to model their own organizations, measure their qualities (robustness, flexibility, efficiency), tune their organizations in order to fit with their requirements, validate these qualities and finally simulate their functioning.

This work opens a number of issues for future research.

The first issues we are currently investigating are related to the *structural evaluation* of our organization:

Firstly, how to add the *control dimension* in our model since we have only implemented the power and coordination dimensions and what are the effects on the qualities of our organization?

Secondly, we do believe that a refinement of the theoretical evaluation is needed. Indeed Grossi framework doesn't take into account the occurrences of roles involved in the modelled organizations. However, taking into account the roles cardinalities and the relations occurrences will certainly influence the evaluation of our model.

Regarding future work, since we have only proven the feasibility of our model, we intend to compare our model with existing scheduler such as the system ARMS that implements adaptive scheduling based on hierarchical multi-agents system. More precisely, we need to compare the adaptiveness and the efficiency of these systems, in particular, interaction overhead, robustness and scalability in case of disturbances.

Finally, we need to consider more applications and possibly in a real grid environment.

Acknowledgements. We thank the anonymous referees for critically reading the manuscript and making several useful remarks. We also wish to thank our colleague Rui Tramontin from the Federal University of Santa Catarina, Brazil, for kindly reading our paper and offering valuable suggestions.

References

1. Foster, I., Kesselman, C.: The grid: blueprint for a new computing infrastructure. Morgan Kaufmann (2004)
2. Dong, F., Akl, S.G.: Scheduling algorithms for grid computing: State of the art and open problems. School of Computing, Queen's University, Kingston, Ontario (2006)
3. Schopf, J.M.: Ten actions when grid scheduling. International Series in Operations Research and Management Science, 15–24 (2003)
4. Wrzesinska, G., Maassen, J., Bal, H.E.: Self-adaptive applications on the grid. In: Proceedings of the 12th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, New York, NY, USA, pp. 121–129 (2007)
5. Berman, F., et al.: Adaptive computing on the grid using AppLeS. IEEE Transactions on Parallel and Distributed Systems 14(4), 369–382 (2003)
6. Berman, F., et al.: New Grid Scheduling and Rescheduling Methods in the GrADS Project. International Journal of Parallel Programming 33(2-3), 209–229 (2005)
7. Foster, I., Kesselman, C., Tuecke, S.: The anatomy of the grid: Enabling scalable virtual organizations. International Journal of High Performance Computing Applications 15(3), 200 (2001)
8. Dignum, V.: The Role of Organization in Agent Systems. Multi-agent Systems: Semantics and Dynamics of Organizational Models. IGI (2009)
9. Foster, I., Kesselman, C., Jennings, N.: Brain Meets Brawn: Why Grid and Agents Need Each Other. In: Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems, pp. 8–15. IEEE Computer Society (2004)
10. Demazeau, Y.: Invited lecture, 1st Ibero-American Workshop on Distributed AI and Multi-Agent Systems (IWDAIMAS 1996), Mexico (1996)
11. Ferber, J., Gutknecht, O., Michel, F.: From agents to organizations: an organizational view of multi-agent systems. Agent-Oriented Software Engineering IV, 443–459 (2003)
12. The MADKIT Agent Platform Architecture, `http://www.madkit.org`
13. Grossi, D., Dignum, F., Dignum, V., Dastani, M., Royakkers, L.: Structural evaluation of agent organizations. In: Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems, New York, NY, USA, pp. 1110–1112 (2006)
14. Kaddoum, E., Gleizes, M.P., Georgé, J.P., Picard, G.: Characterizing and evaluating problem solving self-* systems. In: Computation World: Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns, pp. 137–145 (2009)
15. Petri, C.A.: Fundamentals of a Theory of Asynchronous Information Flow, Amsterdam. Presented at the IFIP Congress 62, pp. 386–390 (1962)
16. Fibich, P., Matyska, L., Rudová, H.: Model of grid scheduling problem. In: Exploring Planning and Scheduling for Web Services, Grid and Autonomic Computing, pp. 17–24 (2005)
17. Smith, R.G.: The contract net protocol: High-level communication and control in a distributed problem solver. IEEE Transactions on Computers 100(12), 1104–1113 (2006)
18. Foundation for Intelligent Physicals Agents, `http://www.fipa.org`

19. Foundation for Intelligent Physicals Agents: FIPA Contract Net Interaction Protocol Specification, `http://www.fipa.org/specs/fipa00029/SC00029H.pdf`
20. Vadhiyar, S.S., Dongarra, J.J.: Self adaptivity in grid computing. Concurrency and Computation: Practice and Experience 17(2-4), 235–257 (2005)
21. Reed, D.A., Mendes, C.L.: Intelligent Monitoring for Adaptation in Grid Applications. Proceedings of the IEEE 93(2), 426–435 (2005)
22. Iosup, A., et al.: On grid performance evaluation using synthetic workloads. In: Frachtenberg, E., Schwiegelshohn, U. (eds.) JSSPP 2006. LNCS, vol. 4376, pp. 232–255. Springer, Heidelberg (2007)
23. Kreaseck, B., Carter, L., Casanova, H., Ferrante, J.: Autonomous protocols for bandwidth-centric scheduling of independent-task applications. Presented at the 17th International Parallel and Distributed Processing Symposium, IPDPS 2003 (2003)
24. Patel, J., et al.: CONOISE-G: agent-based virtual organisations. In: Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems, New York, NY, USA, pp. 1459–1460 (2006)
25. Buisson, J., André, F., Pazat, J.-L.: Dynamic Adaptation for Grid Computing. In: Sloot, P.M.A., Hoekstra, A.G., Priol, T., Reinefeld, A., Bubak, M. (eds.) EGC 2005. LNCS, vol. 3470, pp. 538–547. Springer, Heidelberg (2005)
26. Therasa, A.L.S., Sumathi, G., Dalya, A.S.: Dynamic Adaptation of Checkpoints and Rescheduling in Grid Computing. International Journal of Computer Applications 2(3), 95–99 (2010)
27. Condor Project Homepage, `http://research.cs.wisc.edu/condor/2006`
28. Abramson, D., Buyya, R., Giddy, J.: A Computational Economy for Grid Computing and its Implementation in the Nimrod-G Resource Broker. Future Generation Computer Systems (FGCS) Journal 18(8), 1061–1074 (2002)
29. Buyya, R., Abrasmson, D., Venugopal, S.: The Grid Economy. Proceedings of the IEEE 93(3), 698–714 (2005)
30. Caron, E., Desprez, F.: Diet: A scalable toolbox to build network enabled servers on the grid. International Journal of High Performance Computing Applications 20(3), 335 (2006)
31. Chapin, S., Katramatos, D., Karpovich, J., Grimshaw, A.: The legion resource management system. In: Job Scheduling Strategies for Parallel Processing, pp. 162–178 (1999)
32. Cao, J., Spooner, D.P., Jarvis, S.A., Nudd, G.R.: Grid load balancing using intelligent agents. Future Generation Computer Systems 21(1), 135–149 (2005)
33. Dail, H., et al.: Scheduling in the grid application development software project. International Series In Operations Research and Management Science, pp. 73–98 (2003)
34. Wijngaards, N.J.E., Overeinder, B.J., van Steen, M., Brazier, F.M.T.: Supporting internet-scale multi-agent systems. Data & Knowledge Engineering 41, 229–245 (2002)
35. Shi, Z., Huang, H., Luo, J., Lin, F., Zhang, H.: Agent-based grid computing. Applied Mathematical Modelling 30(7), 629–640 (2006)
36. Muthuchelvi, P., Anandha Mala, G.S.: Agent Based Grid Resource Discovery with Negotiated Alternate Solution and Non-Functional Requirement Preferences. Journal of Computer Science (2009), `http://www.scipub.org/fulltext/jcs/jcs53191-198.pdf`
37. Cao, J., Jarvis, S.A., Saini, S., Kerbyson, D.J., Nudd, G.R.: ARMS: an Agent-based Resource Management System for Grid Computing. Scientific Programming 10(2), 135–148 (2002)

38. Wang, M., Ramamohanarao, K., Chen, J.: Robust Scheduling and Runtime Adaptation of Multi-agent Plan Execution. In: Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, WI-IAT 2008, vol. 2, pp. 366–372. IEEE Computer Society, Washington, DC (2008), http://dx.doi.org/10.1109/WIIAT.2008.136
39. Thabet, I., Hanachi, C., Ghédira, K.: Vers une Architecture de Type Agent BDI pour un Ordonnanceur de Grille Adaptatif. In: Conférence sur les Architecture Logicielles, Montréal, Canada. Revue des Nouvelles Technologies de l'Information RNTI-L-2, pp. 19–33. Cépaduès-Éditions (2008)
40. Foster, I.: Globus toolkit version 4: Software for service-oriented systems. Journal of Computer Science and Technology 21(4), 513–520 (2006)
41. Erwin, D., Snelling, D.: UNICORE: A Grid computing environment. In: Euro-Par 2001 Parallel Processing, pp. 825–834 (2001)