A new actor-based structure for distributed systems

Antonina Dattolo
Dipartimento di Matematica ed Applicazioni "R Cacciopoli"
Università degli Studi di Napoli Federico II, Italy
E-mail: dattolo@unina.it

Flaminia L. Luccio
Dipartimento di Informatica
Università Ca' Foscari Venezia, Italy
E-mail: luccio@unive.it

Abstract - In this paper we propose a formal analytic graph-based description of zz-structures, an unusual, graphic-centric way of linking and organizing information. Starting from this formalized model, we make active and operative all the entities of the system, obtaining an actor-based model, called AZ. The actors are organized in different hierarchical levels and cooperate in order to achieve common goals and solve problems. In particular, the use of actor-based technology is helpful in enhancing distributed computing capabilities, and interoperability in grid systems. The combination of zz-structures and of cooperation activities of different actor classes allows us to define dynamic virtual organizations and to analyze some of issues related to system topological evolutions.

1 Introduction

The massive use of the Web's functionality and the need to gather enough computational resources for running different applications at a single location are some of the aspects that have led to the birth of the grid infrastructure. Distributed and heterogeneous institutions and communities may share data, programs, and computing resources to implement different decentralized services for science, government and business. The *grid* can thus been seen as an extended Web, where users share information, computer resources and services. Although machines and resources are heterogeneous, i.e., are based on different platforms, hardware and software architectures or computer languages, the way to access them is via open standards. Moreover, differently from classical distributed computing, grid computing on grid infrastructures offers innovative applications and in some cases also high-performance computation. Examples of grid applications are: visualization of large distributed data sets, sharing scientific instruments in remote computers, distributed processing of demanding data analysis and intensive simulations on remote supercomputers.

More formally, in [7], the authors define the grid problem as "coordinated resource sharing and problem solving in dynamic, multi-institutional, virtual organizations".

Certainly, distributed computer networks may be considered the heart of the computational grids, but additional mechanisms must be added to the networks in order to model a grid in the stronger sense. The technical demands of grid concepts at this level require increasing amounts of "intelligence", collaborative ability, adaptability, component mobility, etc.; also, gridness seems to imply that the system is "aware of itself" to a certain extent, and has the ability to carry out its tasks "itself", without a great deal of manual intervention; in other words, characteristics frequently associated with agents. The use of agent technology may be helpful in simplifying and enhancing distributed computing capabilities, and in particular enhancing intelligent interoperability in grid systems [2]. Representing the computational and communication components of a computational grid as agents allows these components to be both uniformly represented within the architecture, and managed in a straightforward way by higher level components.

In this work we concentrate on AZ, a grid model based on an extension **A**ctor-based of **Z**z-structures.

Actors are a class of computational agents, firstly defined by Hewitt [9], and successively intensively studied by Agha [1].

Zz-structures are particular hypermedia structures, used for linking and organizing information; they were firstly introduced by Ted Nelson [13], and then have been studied from many different perspectives (an example is provided in [4]).

The choice of modeling with zz-structures some fundamental parts of a grid, such as the organizations, responds to key aspects of grids, that are, e.g., composition of resources, closure and fractal properties [10]; also, zz-structures are minimalist and may be defined in a recursive way, by composition, generating local and global grids, or a hierarchy of grids, and larger grids can be constructed by composing smaller (perhaps local) grids. It may be possible for local grids to operate independently from larger grids, of which they may temporarily be a part. Local grids can provide heterogeneous enclaves where different standards, policies, or systemic properties such as adaptability, scalability, security, reliability, etc., hold.

In Section 2 we first propose a formal analytic graph-based description of zz-structures. Note that, a first and preliminary step towards the formalization of the notion of zz-structures in terms of graph theory can be found in [8], where the authors intuitively present this new model. Then, in Section 3, starting from our formalized model, we make active and operative all the entities of the system: the result is a distributed, actor-based model capable of representing both hypermedia distribution and collaborative schemes among different and heterogeneous entities. These entities and the data they store, may be viewed as a virtual organi-

zation VO, and are part of a particular grid infrastructure, the *data grid*. This VO is organized in different hierarchical levels and entities cooperate in order to achieve common goals and solve problems. As an example, in Section 4, we show how these entities may collaborate in order to support dynamically joining of new resources to a VO. We conclude in Section 5.

2 Virtual Organizations

According to [15], a virtual organization VO is defined as a set of concrete organizations O, that collaborate as peer, by: a the set of resources and services; the interface for accessing them; the set of policies for the different operations; and the set of protocols for implementing the policies. We can recursively define a concrete organization as follows.

Definition 1 *An* organization O *is a tuple* (S, SE, P, PR) *where* S *is a structured set of resources, equipped with an interface for accessing them. Each resource offers a subset of services* SE *and supports a subset of policies* P *for the operation of* S *and a subset of protocols* PR *for the implementation of* P.

The interface of an organization O is the set of externally visible operations through which the resources and services of the O are accessed; the policy P is a set of rules specifying the admissible patterns of use for some types of resources and/or services; finally, a protocol PR defines the sequence of interactions among the resources and services of an O, used to implement a policy P.
In this section we first recall some basic graph theory notation and then we propose a formal model for S, the first component of an organization O.

2.1 Basic graph theory definitions

A *graph* G is a pair $G = (V, E)$, where V is a finite non-empty set of elements called *vertices* and E is a finite set of distinct unordered pairs $\{u, v\}$ of distinct elements of V called *edges*. A *multigraph* is a triple $MG = (V, E, f)$ where V is a finite non-empty set of vertices, E is the set of edges, and $f : E \rightarrow \{\{u, v\} \mid u, v \in V, u \neq v\}$ is a surjective function.
An *edge-colored multigraph* is a triple $ECMG = (MG, C, c)$ where: $MG = (V, E, f)$ is a multigraph, C is a set of colors, $c : E \rightarrow C$ is an assignment of colors to edges of the multigraph. In a multigraph $MG = (V, E, f)$, edges $e_1, e_2 \in E$ are called *multiple* or *parallel* iff $f(e_1) = f(e_2)$. Thus, a graph as a particular multigraph $G = (V, E, f)$ without parallel edges. Given an edge $e = \{u, v\} \in E$, we say that e is *incident* to u and v; moreover u and v are *neighboring* vertices. Given a vertex $x \in V$, we denote with $deg(x)$ its degree, i.e., the number of edges incident to x, and with d_{max} the maximum degree of the graph, i.e., $d_{max} = \max_{z \in V}\{deg(z)\}$. In a edge-colored (multi)graph ECMG, where $c_k \in C$, we define $deg_k(x)$ the number of edges of color c_k incident to vertex x. A vertex of degree 0 is called *isolated*, a vertex of degree 1 is called *pendant*.
A *path* $P = \{v_1, v_2, \ldots, v_s\}$ is a sequence of neighboring vertices of G, i.e., $\{v_i, v_{i+1}\} \in E$, $1 \leq i \leq s-1$. Finally, a graph $G = (V, E)$ is *connected* if: $\forall x, y \in V$, $\exists$ a path $P = \{x = v_1, v_2, \ldots, v_s = y\}$, with $\{v_k, v_{k+1}\} \in E$, $1 \leq k \leq s-1$.

2.2 Resources and interface of an organization O

The set of resources S in a concrete organization O are arranged using zz-structures, a new, graph-centric system of conventions for data and computing. Zz-structures are intrinsically non-hierarchical (hierarchy is optional), and there are intrinsic visualizations for everything, viewable in rows and columns, to considerable benefit. Some examples of interfaces for accessing them are proposed in [13].
A zz-structure can be thought of as a hypermedia space filled with cells: each cell (resource) may have a content: in this case, it is *atomic* if it contains only one unit of data of one type [11], or it is *referential* if it represents a package of different cells. There are also special cells that do not have a content and thus are *positional*, i.e., have a positional or topographical function.
Cells are connected together into linear sequences, each of which is composed of links of the same color: the glue of a linear series is its *dimension*. A single series of cells connected in the same dimension is called *rank*, i.e., a rank is in a particular dimension. Moreover, a dimension may contain many different ranks. Each rank has a starting and an ending cell, called *headcell* and *tailcell*, respectively, and the direction from the starting (ending) to the ending (starting) cell is called *posward* (respectively, *negward*). For any dimension, a cell can only have one connection in the posward direction, and one in the negward direction. This ensures that all paths are non-branching, and thus embodies the simplest possible mechanism for traversing links.

Zz-structures. A zz-structure can be viewed as a multigraph where edges are colored, with the restriction that every vertex has at most two incident edges of the same color. Differently from [8], we consider undirected graphs, i.e., edges may be traversed in both directions.

Definition 2 *A* zz-structure *is an edge-colored multigraph* $S = (MG, C, c)$*, where* $MG = (V, E, f)$*, and* $\forall x \in V$, $\forall k = 1, 2, ..., |C|$, $deg_k(x) = 0, 1, 2$. *Each vertex of a zz-structure is called* zz-cell *and each edge a* zz-link.

An example of a zz-structure is given in Figure 1. Observe that we do not admit the property of having multiple edges of the same color as it is not interesting in the model we are considering.

Dimensions. An alternative way of viewing a zz-structure is a union of subgraphs, each of which contains edges of a unique color.

Proposition 1 *Consider a set of colors* $C = \{c_1, c_2, ..., c_{|C|}\}$ *and a family of indirect edge-colored graphs* $\{D^1, D^2, ..., D^{|C|}\}$*, where* $D^k = (V, E^k, f, \{c_k\}, c)$*, with* $k = 1, ..., |C|$*, is a graph such that:* $E^k \neq \emptyset$*, and* $\forall x \in V$, $deg_k(x) = 0, 1, 2$. *Then,* $S = \bigcup_{k=1}^{|C|} D^k$ *is a zz-structure.*

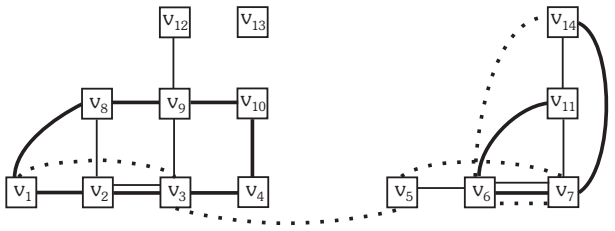

Figure 1. A zz-structure where thick, normal and dotted lines represent three different colors.

Definition 3 *Given a zz-structure $S = \bigcup_{k=1}^{|C|} D^k$, then each graph D^k, $k = 1, \ldots, |C|$, is a distinct* dimension *of S.*

From Proposition 1 we derive that, all vertices exist in all dimensions, and that the maximum degree $d_{max} \leq 2|C|$. Finally, $\forall e_i \in E^k,\ c(e_i) = c_k$.
From Figure 1 we can extrapolate three dimensions, D^{thick}, D^{normal}, and D^{dotted}, as shown in Figure 2. Moreover, a dimension $D^k = (V, E^k, f, \{c_k\}, c)$ is composed of:

- A set of *isolated vertices*, i.e, $\{x \in V \ : deg_k(x) = 0\}$, eventually empty. E.g., in dimension D^{thick}, v_5, v_{12}, v_{13} are isolated vertices.
- A set of connected components identified by a sequence of vertices $\{x_1, x_2, \ldots, x_s\}$. These vertices may create *distinct paths*, i.e., $deg_k(x_i) = 2,\ \forall\ i : 2 \leq i \leq s-1$, while $deg_k(x_1) = deg_k(x_s) = 1$. For example, path $\{v_{11}, v_6, v_7, v_{14}\}$ in dimension D^{thick}. Vertices may also create *distinct cycles*, i.e., paths where $x_1 = x_s$, $s > 2$, and $deg_k(x_i) = 2\ \forall\ i : 1 \leq i \leq s$. E.g., the cycle $\{v_1, v_2, v_3, v_4, v_{10}, v_9, v_8, v_1\}$ in dimension D^{thick}.

Ranks. Each series of connected cells, identified in the three different dimensions of Figure 2, is defined as a *rank*.

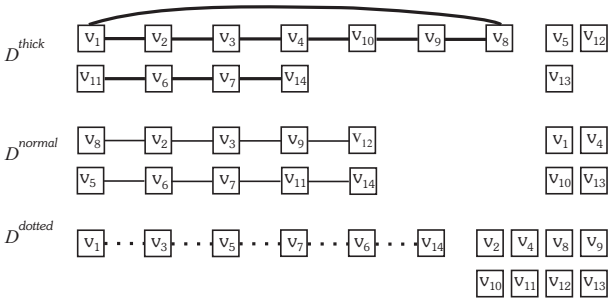

Figure 2. The dimensions of the zz-structure of Figure 1.

Definition 4 *Consider a dimension $D^k = (V, E^k, f, \{c_k\}, c)$, $k = 1, \ldots, |C|$ of a zz-structure $S = \bigcup_{k=1}^{|C|} D^k$. Then, each of the l_k connected components of D^k is called a* rank.

Thus, each rank $R_i^k = (V_i, E_i^k, f, \{c_k\}, c)$, $i = 1, \ldots, l_k$, is an indirect, connected, edge-colored graph that owns the following properties: $V_i \subseteq V$, $E_i^k \subseteq E^k$, and $\forall x \in V_i$, $1 \leq deg_k(x) \leq 2$.

A rank is in a particular dimension and it must be a *connected* component; consequently, a dimension can contain one (if $l_k = 1$) or more ranks. Moreover, the number l_k of ranks differs in each dimension D^k. E.g., in Figure 2, in dimension D^{thick}, we have two ranks defined by vertices $\{v_1, v_2, v_3, v_4, v_{10}, v_9, v_8, v_1\}$ and $\{v_{11}, v_6, v_7, v_{14}\}$, and in dimension D^{dotted}, we have only rank $\{v_1, v_3, v_5, v_7, v_6, v_{14}\}$.

Definition 5 *A* ringrank *is a rank $R_i^k = (V_i, E_i^k, f, \{c_k\}, c)$, $i = 1, \ldots, l_k$, where $\forall x \in V_i,\ deg_k(x) = 2$.*

E.g., a ringrank is the cycle defined by vertices $\{v_1, v_2, v_3, v_4, v_{10}, v_9, v_8, v_1\}$ in dimension D^{thick} of Figure 2.
Many ranks can be in the same dimension [13]. In this case, they are defined as parallel ranks.

Definition 6 *Given a zz-structure $S = \bigcup_{k=1}^{|C|} D^k$, m ranks $R_j^k = (V_j, E_j^k, f, \{c_k\}, c)$, $(j = 1, 2, \ldots, m,\ 2 \leq m \leq l_k)$ are* parallel ranks *on the same dimension D^k, $k \in \{1, \ldots, |C|\}$ iff $V_j \subseteq V$, $E_j^k \subseteq E^k$, $\forall j = 1, 2, \ldots, m$, and $\bigcap_{j=1}^{m} V_j = \emptyset$.*

Thus, the ranks are on the same dimension D^k, and cannot have cells in common. In Figure 2, the two ranks in the dimension D^{thick} are parallel, as well as the two ranks in the dimension D^{normal}.

Definition 7 *Given a zz-structure $S = \bigcup_{k=1}^{|C|} D^k$, a rank $R_i^a = (V_i, E_i^a, f, \{c_a\}, c)$ in dimension D^a and a rank $R_j^b = (V_j, E_j^b, f, \{c_b\}, c)$ in dimension D^b, $a \neq b$, we say that R_i^a and R_j^b are* intersecting ranks *on different dimensions iff $V_i, V_j \subseteq V$, $E_i^a \subseteq E^a$, $E_j^b \subseteq E^b$, and $V_i \bigcap V_j \neq \emptyset$.*

Thus the ranks are on two different dimensions, and have one or more cells in common, given from the intersection of V_i and V_j. In Figure 1 and 2, e.g., the rank $\{v_1, v_2, v_3, v_4, v_{10}, v_9, v_8, v_1\}$ of dimension D^{thick} intersects, e.g., the rank $\{v_8, v_2, v_3, v_9, v_{12}\}$ of dimension D^{normal} in vertices v_2, v_3, v_9, v_8.

Local and global orientation.

Definition 8 *Consider a rank $R_i^k = (V_i, E_i^k, f, \{c_k\}, c)$ of a zz-structure $S = \bigcup_{k=1}^{|C|} D^k$. Then, $\exists$ a function $g_x^i : E_i^k \rightarrow \{-1, 1\}$, such that, $\forall x \in V_i$, if $\exists y, z \in V_i : \{x, y\}, \{x, z\} \in E_i^k$, then $g_x^i(\{x, y\}) \neq g_x^i(\{x, z\})$. Thus, we say that each vertex $x \in V_i$ has a* local orientation *in R_i^k. Given an edge $\{a, b\} \in E_i^k$, we say that $\{a, b\}$ is in a* posward *direction from a in R_i^k iff $g_a^i(\{a, b\}) = 1$, else is in a* negward *direction.*

Thus, local orientation ([6]) is a property related to each vertex of a rank. The vertices of the zz-structure have also a global orientation, i.e., we can extend the previous property to all the ranks and dimensions. Moreover, all the local choices of orientation are consistent.

Definition 9 *Assume a zz-structure* $S = \bigcup_{k=1}^{|C|} D^k$ *has* $l = \sum_{k=1}^{|C|} l_k$ *ranks* $R_i^k = (V_i, E_i^k, f, \{c_k\}, c)$, $i = 1, \ldots, l_k$, *and* $k = 1, \ldots, |C|$. *Then,* S *has* global orientation *iff,* $\forall \{x, y\} \in E_i^k$, $\forall i = 1, \ldots, l_k$, *and* $\forall k = 1, \ldots, |C|$, *we have* $g_x^i(\{x, y\}) \neq g_y^i(\{x, y\})$.

3 Actor-based zz-structures

In this section we present AZ, a grid model based on an extension **A**ctor-based of **Z**z-structures. AZ is a new model for describing autonomous organizations, where actor classes are organized based on a hierarchy of interacting levels, as depicted in Figure 3. Organizations are at the top of the hierarchy (zz-a-structure), viewed in terms of complex actors that know and directly manipulate dimensions and isolated cells; the dimensions are uniquely identified by their colors and know and manipulate the ranks. The ranks know and coordinate cells and the links that connect them; finally, cells and links are primary entities and exist independently from the structures at the above levels.

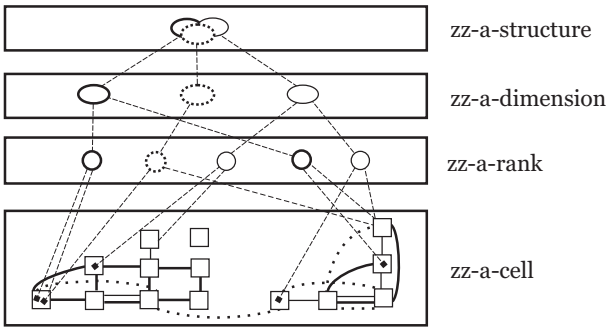

Figure 3. Actor-based hierarchy.

We assume that services SE, policies P and protocols PR of an organization are distributed among actor classes as shared agreements or capabilities, and are not centralized in some framework. These three components are managed by appropriate scripts, interaction schema and by the cooperation activities among actors. Now, we first recall a brief description of the actor model and we then propose an actor-based model for the components SE, P and PR of an organization O.

3.1 Brief description of the actor model

In distributed systems the actor model [1] is used in order to model concurrent computations.

Each actor is defined by a *passive part*, that is, a set of local variables, termed *acquaintances* in [1], that constitute its internal state; an *active part*, that reacts to the external environments by executing some predefined procedural skills, called *scripts* defining the behavior of the actor; finally, its *mail queue*, that buffers incoming communication (i.e., messages).

Each actor has a unique name (the uniqueness property) and a behavior, and communicates with other actors via asynchronous messages.

Actors are reactive in nature, i.e., they execute only in response to messages received, and may perform three basic actions: create a finite number of actors with universally fresh names; send a finite number of messages; assume a new behavior.

An actor's behavior is deterministic i.e., its response to a message is uniquely determined by the message contents and its internal state.

An actor is thus described by specifying two elements: its data part and its script part, that is a set of scripts which can be executed by the specific actor.

$$(DefActor\, ActorName \\ \langle acquaintances\text{-}list \rangle\ \{scripts\text{-}list\})$$

However the "pure" actor model suffers from rigid point-to-point communication protocol, that limits the design of efficient collaboration strategies; an extension of communication strategies is proposed in [5] and used in this work. So, the communication between a sender and one or more receivers may use multicasting message passing protocols and is accomplished by the $send$ command types: $send$ allows an actor to send a point-to-point message; $send\text{-}m$ allows an actor to send multicasting messages on the network; $send\text{-}now$ and $send\text{-}now\text{-}m$ are similar to the previous send, but they require an acknowledge message from the receiver actors in order to continue the computation.

A general form of the send construct is the following:

$$send\text{-}type\ (script\text{-}name\ argument\text{-}list; \ldots; \\ script\text{-}name\ \ argument\text{-}list)\ to\ destination\text{-}list$$

where: $send\text{-}type\ \ldots\ to$ is one of the send commands; $script\text{-}name\ argument\text{-}list$ determines the script (with its arguments, if any) that the destination actors trigger once they have received the message; $destination\text{-}list$, introduced by the keyword to , identifies the actor(s) to which the message is addressed.

3.2 Services, protocols and policies

The first level of the hierarchy is composed by a unique special actor $zz\text{-}a\text{-}structure$, that coordinates the whole organization; it is a referential actor that collects and manages the family of dimensions and of isolated cells, it is composed of.

$$(DefActor\ zz\text{-}a\text{-}structure \\ \langle dimensions\ isolated\text{-}cells \ldots \rangle \\ \{return\text{-}colors\ return\text{-}ranks\ return\text{-}cells \\ return\text{-}links\ check\text{-}global\text{-}orientation \\ delete(cell_1, \ldots, cells_j) \ldots\})$$

Dimensions and isolated cells are respectively addressed by the acquaintances $dimensions$ and $isolated\text{-}cells$. Other information on the zz-structure can be obtained by activating specific scripts: for example, the first four scripts presented in this actor class enable it to derive colors, ranks, cells and links that constitute the zz-structure. These operations are performed by sending querying messages to $dimensions$ and $isolated\text{-}cells$; ranks and used colors are obtained directly from dimensions actors, while cell

and link references are requested from dimensions to ranks actors. Other scripts, such as *check-global-orientation* that checks whether local orientation of neighboring cells are consistent, and $delete(cell_1, \ldots, cells_j)$, that deletes a chosen set of cells, are used in dynamic operations illustrated in Section 4.
The dimension is at the second level of the hierarchy.

$(DefActor\ zz\text{-}a\text{-}dimension$
$\langle color\ ranks\ structure\ isolated\text{-}cells\rangle$
$\{check\text{-}dir(x,y)\ merge\text{-}ranks\ update\ldots\})$

Any dimension knows its *color*, the set of $ranks$ it is composed of and the zz-$structure$ it belongs to. The script $check\text{-}dir(x,y)$ checks if the addition of a new edge $\{x,y\}$ directed from x to y is admittable, and $merge\text{-}ranks$ merges two distinct ranks into a new single one.
The third level of the hierarchy is composed by the ranks.

$(DefActor\ zz\text{-}a\text{-}rank$
$\langle(cell_1, cell_2, \ldots, cell_n)$
$/((cell_1, cell_2), \ldots, (cell_{n-1}, cell_n))$
$color\ dimension\ ringrank\rangle$
$\{check\text{-}local\text{-}orientation\ get\text{-}color\ return\text{-}ring$
$set\text{-}ringrank\ set\text{-}head\ set\text{-}tail\ delete\ldots\})$

The $(cell_1, cell_2, \ldots, cell_n)$ acquaintances represent the ordered list of zz-cells present in the rank. Derived only-read attributes are the colored links $((cell_1, cell_2), \ldots, (cell_{n-1}, cell_n))$. $ringrank$ is a boolean attribute, that assumes $ringrank = true$ if the rank is a ringrank (i.e., if $cell_1$ = cell$_n$), $false$ otherwise. The script $return\text{-}ring$ checks if the rank is a ringrank, while the script $set\text{-}ringrank$ transforms a rank into a ringrank.
The last layer of our hierarchy is constituted by the zz-cells.

$(DefActor\ zz\text{-}a\text{-}cell$
$\langle resource\ zz\text{-}structure/zz\text{-}rank(s)\ freedir\ldots\rangle$
$\{check\text{-}degree(color)$
$return\text{-}freedir(color)\ (delete, x, y, c_j)\ldots\})$

A zz-cell is an atomic or composite $resource$ in the organization, and it has minimal information about the external environment: it addresses the $zz\text{-}structure$, if it is an isolated cell, or alternatively the set of ranks it is contained in. The script $check\text{-}degree(color)$ checks the degree of two vertices in a specific dimension; the script $return\text{-}freedir(color)$, given a color k, returns the non-used direction; thus, the acquaintance $freedir$ assumes value $both$ if the cell x is an isolated vertex, $posward$ (or $negward$) when the posward (or negward) part is free, and finally $none$ if $deg_k(x) = 2$.

4 Evolution of an organization O

An interesting issue that we want to address is the dynamical join of a new resource to the organization O. This can be handled at the actor level with the addition of a new link. In this section we will show how a session actor may coordinate the addition of an edge between two cells of an organization O. We will show the execution code of a dimension actor D^k, and we will describe the different actions executed by ranks, zz-cells and the zz-structure. The interaction between actors is defined using the diagrammatic language AUML (Agent Unified Modeling Language) [14], extension of UML (Unified Modeling Language) for agents.
Assume A is the user-author that interacts in order to add the link, O is the name of the zz-structure, D^k is the dimension with color c_k, $R^k_{(x)}$ is the name of the rank x belongs to. If x is an isolated vertex in dimension D^k, that is, $deg_k(x) = 0$, then we assume that $R^k_{(x)} = \emptyset$. Let us now consider a session actor SA of a O. Whenever SA receives from an author A the message $(add\ (x,y)\ c_k)$, regarding the addition of the new edge $\{x,y\}$ of a particular color c_k and directed from x to y, it sends the following multicast messages to vertices x and y:

$send\text{-}now\text{-}m(checkdegree(c_k))$ to x y

The control returns back to the session actor SA that will continue the addition operation only if both x and y have at least a not-occupied direction (i.e., $deg_k(x) \neq 2$ and $deg_k(y) \neq 2$). If this condition is not satisfied, then SA communicates to A that is not possible to add the edge $\{x,y\}$ in dimension D^k, and thus A will decide whether to send a new request or not. On the other hand, if the addition is admissible then SA sends to dimension D^k all the information necessary to add the edge i.e., (x,y), the edge in form of an ordered list to identify posward and negward directions, the address $R^k_{(x)}$ of the rank x belongs to in dimension D^k, with the relative degree $deg_k(x)$, and the acquaintance $freedir_x$ that identifies the direction(s) on which x has no edge. In the same message, analogous information are sent also for y, i.e. $R^k_{(y)}$, $deg_k(y)$ and $freedir_y$. The behavior of D^k is formally described in Table 1.

Addition$((x,y), R^k_{(x)}, deg_k(x), freedir_x, R^k_{(y)}, deg_k(y), freedir_y)$
1. $check\text{-}dir(x,y)$;
2. **if** $dir = incoherent$
3. **then** send $(incoherentdir\ (x,y)\ c_k)$ to SA
4. **else Case**: $R^k_{(x)} = R^k_{(y)} \neq \emptyset$: send $(ringrank)$ to $R^k_{(x)}$;
5. $R^k_{(x)} = \emptyset$ and $R^k_{(y)} \neq \emptyset$: send $(addHead\ x)$ to $R^k_{(y)}$;
6. $R^k_{(x)} \neq \emptyset$ and $R^k_{(y)} = \emptyset$: send $(addTail\ y)$ to $R^k_{(x)}$;
7. $R^k_{(x)} = R^k_{(y)} = \emptyset$: $createNewRank\ ((x,y))$;
8. $R^k_{(x)} \neq R^k_{(y)}$: $mergeRanks\ (R^k_{(x)}, R^k_{(y)})$.

Table 1. Addition of an edge. Code of dimension D^k.

At the receipt of this message, D^k checks (line 1) if the introduction of the edge (x,y) maintains a coherent direction of the edges (i.e., maintains global orientation). If this is not possible (line 2), then D^k communicates to SA (and SA respectively to A) that is not possible to add the edge $\{x,y\}$ in dimension D^k (line 3). Else, if the edge maintains global orientation, the dimension D^k will assume four different behaviors depending on the values of $R^k_{(x)}$ and $R^k_{(y)}$:
Case $R^k_{(x)} = R^k_{(y)} \neq \emptyset$ (line 4 of Table 1). If $R^k_{(x)} = R^k_{(y)} \neq \emptyset$, then x is the tailcell and y the headcell of the same rank

$R^k_{(xy)} = R^k_{(x)} = R^k_{(y)}$. In this case, $R^k_{(xy)}$ is transformed into a ringrank under D^k's request. This operation is performed by script *set-ringrank*, local at $R^k_{(xy)}$ (see Figure 4), and will end only after x e y have updated the variable $freedir$. Then $R^k_{(xy)}$ communicates to D^k the end of the operation and this information is forwarded to SA.

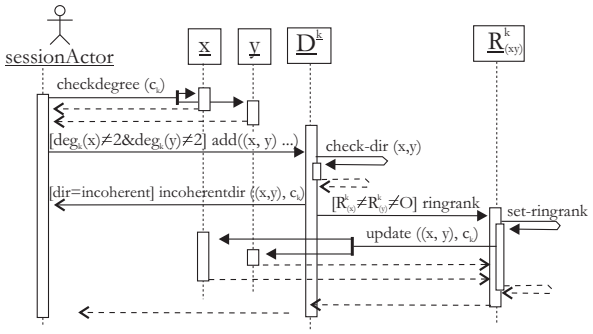

Figure 4. Case $R^k_{(x)} = R^k_{(y)} \neq \emptyset$.

Case $R^k_{(x)} = \emptyset$ **and** $R^k_{(y)} \neq \emptyset$ (line 5 of Table 1). If $R^k_{(x)} = \emptyset$, then x is an isolated vertex; for this reason, $R^k_{(y)}$ inserts vertex x as its new head under D^k's request . In this case, beside the update of x, y, $R^k_{(y)}$ and D^k, the zz-structure O has to delete x from the set of isolated cells. As in the previous case the operation ends with a message that is forwarded up to the session actor SA. The case $R^k_{(x)} \neq \emptyset$ and $R^k_{(y)} = \emptyset$ (line 6 of Table 1) is analogous.

Case $R^k_{(x)} = R^k_{(y)} = \emptyset$ (line 7 of Table 1). If both x and y are isolated cells, D^k will create a new rank $R^k_{(xy)}$. This operation requires an update of the cells x and y, of the zz-structure O and of dimension D^k itself . Also in this case the operation ends with a message that is forwarded up to the session actor SA.

Case $R^k_{(x)} \neq R^k_{(y)}$; $R^k_{(x)}, R^k_{(y)} \neq \emptyset$ (line 8 of Table 1). In this case, x is the tailcell of its rank $R^k_{(x)}$ and y is the headcell of its rank $R^k_{(y)}$. The merge operation is carried out by D^k that sends a request of inglobing $R^k_{(y)}$ into $R^k_{(x)}$. When $R^k_{(x)}$ acknowledges this operation, then D^k destroys $R^k_{(y)}$, x and y update their information and the operation ends with a message from D^k to the session actor SA.

The proof of correctness of this algorithm is omitted for lack of space.

5 Conclusion

In the first part of this paper, we have provided a formal graph-based description of virtual organizations, in terms of zz-structures and of computational agents. We have shown how the use of actor-based technology may be helpful in simplifying and enhancing distributed computing capabilities, and in particular enhancing interoperability in grid systems.

References

[1] G. Agha. *Actors: A Model of Concurrent Computation in Distributed Systems*. MIT Press, 1986.

[2] J. Bradshaw (ed.). *Software Agents*. American Association for Artificial Intelligence, MIT Press, 1997.

[3] H. Casanova. Distributed Computing Research Issue in Grid Computing, *ACM SIGCAT News*, 33(3), pp. 50-70, 2002.

[4] S. Canazza and A. Dattolo. Open, dynamic electronic editions of multidimensional documents. *IASTED Proc. of EuroIMSA*, Chamonix (France), March 14-16, pp. 230-235, 2007.

[5] A. Dattolo and V. Loia. Distributed Information and Control in a Concurrent Hypermedia-oriented Architecture. *Int. Journal of Software Engineering and Knowledge Engineering*, 10(6), pp. 345-369, 2000.

[6] P. Flocchini, B. Mans, N. Santoro. Sense of Direction: Definitions, Properties and Classes, *Networks*, 32(3), pp. 165-180, 1998.

[7] I. Foster, C. Kesselman, S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of High Performance Computing Applications*, 15(3), pp. 200-222, 2001.

[8] M.J. McGuffin. A Graph-Theoretic Introduction to Ted Nelson's Zzstructures, January 2004, http://www.dgp.toronto.edu/ ∼mjmcguff/research/zigzag/.

[9] C. Hewitt. Viewing control structures as patterns of message passing, *Artificial Intelligence*, 8(3), p. 323-364, 1977.

[10] F. Manola and C. Thompson. *Characterizing Computer-Related Grid Concepts*. Object Services and Consulting, Inc., 1999.

[11] A. Moore, J. Goulding, T. Brailsford and H. Ashman. Practical Applitudes: Case Studies of Applications, *Proc. of 15th ACM Conf. on Hypertext and Hypermedia - HT'04*, August 9-13, Santa Cruz, California, USA, pp. 143- 152, 2004.

[12] T.H. Nelson. ZigZag (Tech briefing): Deeper Cosmology, deeper documents. *Proc. of the 12th ACM Conf. on Hypertext and Hypermedia - HT'01*, pp. 261-262, University of Aarhus, Denmark, August 14-18, 2001.

[13] T.H. Nelson. A Cosmology for a different computer universe: data model mechanism, virtual machine and visualization infrastructure. *Journal of Digital Information*, 5(1), article No. 298, 2004.

[14] J. J. Odell, H. V. D. Parunak and B. Bauer. *Representing Agent Interaction Protocols in UML*. Agent-Oriented Software Engineering, P. Ciancarini and M. Wooldridge eds., Springer, Berlin, pp. 121-140, 2001.

[15] M. Parashar and J. C. Browne. Conceptual and Implementation Models for the Grid. *Proc. of the IEEE*, 93(3), pp. 653-658, 2005.