Checking Consistency in role oriented Dependence Networks

G. Boella [a] V. Genovese [a] L. van der Torre [b] S. Villata [a]

[a] *University of Turin, Italy, {guido, genovese, villata}@di.unito.it*
[b] *University of Luxembourg, Luxembourg, leendert@vandertorre.com*

Abstract

In this paper we first formalize dependence networks that can be automaticaly build to model goal-based relationships among agents. Then, we propose three algorithms to build and check the consistency of a dependence network. We start presenting the elements composing our ontology such as agents, goals, skills, dependencies with the addition of the institutional notions of roles, institutional goals, institutional skills. We investigate the reasons behind the possible inconsistencies in building the combined dependence network and we propose an algorithm to check them.

1 Introduction

The definition of appropriate mechanisms of communication and coordination in open Multiagent Systems (MAS), motivates the development of models and methodologies with the aim to support the MAS designer for the whole development process of the software, e.g., the TROPOS methodology [5], developed for agent-oriented design of software systems. The TROPOS methodology [5] is based on the multiagent paradigm consisting of a set of agents and their features but it does not consider the addition of an institutional perspective to this paradigm. Recently, institutions have emerged as a new mechanism in the design of artificial social systems, which are used in conceptual modeling of multiagent organizations in agent oriented software engineering [19, 2]. In the MAS design phase, there may exist two separate views specifying agents interaction: the agent view and the institutional view. The first one models the relationship among agents represented by dependencies to achieve a particular goal, for instance agent A may rely on agent B for a goal G.

In the institutional view, relationships are among roles, which are abstractions that model the expected behaviour that an agent has to fulfill while playing a specific role. In this case, the dependence of role R on role $R^{'}$ for the institutional goal IG represents the fact that an agent which enters the role R has to adopt the goal IG that, in order to be achieved, needs the cooperation of the agent who plays role $R^{'}$. E. g., in a Grid-based virtual organization, node a enters role adm, the VO administrator role, and it can authorize the other nodes to access to a resource.

In this paper we address the following research questions: How to automatically build a dependence network from the agent view and from the institutional view describing the multiagent system and how to check automatically the inconsistency causes during the building of the combined view of the multiagent system.

First, starting from [2, 15], we present the elements composing the ontology of our model such as agents, goals, facts, skills, dependencies with the addition of the institutional notions of roles, institutional goals, institutional facts, institutional skills. Our model is a directed labeled graph whose nodes are instances of the metaclasses of the metamodel, e.g., agents, goals, and whose arcs are instances of the metaclasses representing relationships between them such as dependency. Second, we present two algorithms to build the dependence network from the agent view and to build the institutional dependence network from the institutional view. Third, we investigate the reasons behind the possible inconsistencies in building the combined dependence network and we present an algorithm to check the presence of these inconsistencies.

The remainder of the paper is as follows. Section 2 describes a Grid computing scenario as case study for the design of virtual organizations. In Section 3, we present the notions composing the ontology. In Section 4, we define the algorithms to check the consistency of the dependence networks. Related work and conclusions end the paper.

2 The Grid Scenario

The Grid Computing paradigm provides the technological infrastructure to facilitate e-Science and e-Research. Grid technologies can support a wide range of research including amongst others: seamless access to a range of computational resources and linkage of a wide range of data resources. It is often the case that research domains and resource providers require more information than simply the identity of the individual in order to grant access to use their resources. The same individual can be in multiple collaborative projects, each of which is based upon a common shared infrastructure. This information is typically established through the concept of a virtual organization (VO) [10]. A virtual organization allows the users, their roles and the resources they can access in a collaborative project to be defined. In the context of virtual organizations, there are numerous technologies and standards that have been put forward for defining and enforcing authorization policies for access to and usage of virtual organizations resources. Role based access control (RBAC) is one of the more well established models for describing such policies. In the RBAC model, virtual organization specific roles are assigned to individuals as part of their membership of a particular virtual organization. The general idea of the RBAC model is that, permissions are associated with functional roles in organizations, and members of the roles acquire all permissions associated with the roles. Allocation of permissions to users is achieved by assigning roles to users.

3 Institutional MAS: agents, roles and assignments

We divide our ontology in three submodels: the agent model, the institutional model, and the role assignment model, as shown in Figure 1. Such a decomposition is common in organizational theory, because the organization can be designed without having to take into account the agents that will play a role in it. Also, if another agent starts to play a role, for example if a node with the role of simple user becomes a VO administrator, then this remains transparent for the organizational model. Likewise, agents can be developed without knowing in advance in which institution they will play a role.

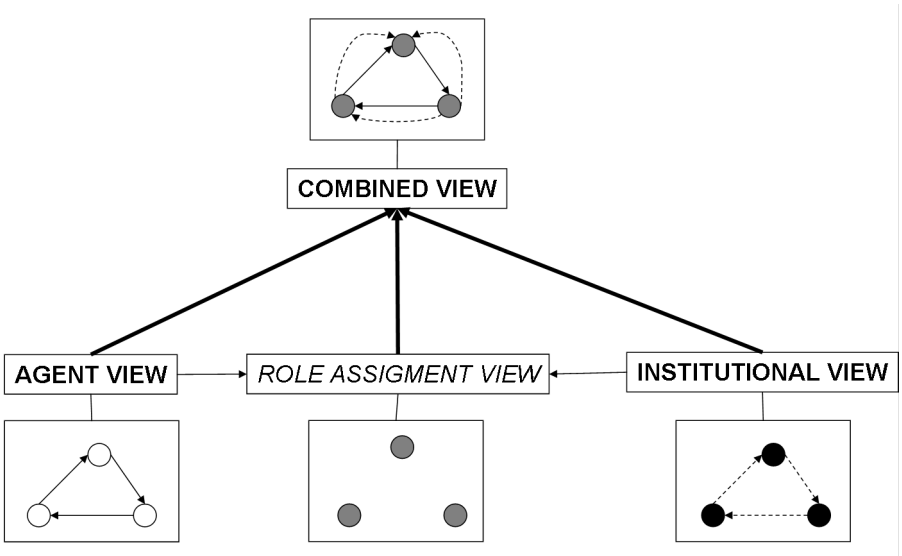

Figure 1: The conceptual metamodel.

The notion of agent and all its features as goals, capabilities, are used in the conceptual modeling as in TROPOS [5]. In our model, we add to these notions those related to the institution such as the notion of role and its institutional goals, capabilities and facts. Both these notions, merged in the combined view,

are used in the conceptual modeling and to each agent it is possible to assign different roles depending on the organization in which the agent is playing. Adding the institution, to each agent are associated both a number of physical features and a role with all its institutional features. An agent can be defined as an entity characterized by a number of features as his capabilities, called skills, his world description, called facts, and his goals, such as the tasks he wants to achieve. The definition of the agent view is as follows:

Definition 1 (Agent view) $\langle A, F, G, X, goals : A \rightarrow 2^G, skills : X \rightarrow 2^A, R : G \rightarrow 2^X \rangle$ *consists of a set of agents A, a set of facts F, a set of goals G, a set of actions X, a function $goals$ that relates with each agent the set of goals it is interested in, a function $skills$ that describes the actions each agent can perform, and a set of rules, represented by the function R that, given a goal, provides the set of actions that must be done in order to achieve it.*

Example 1 *Considering a virtual organization on a Grid with a role based access control policy, the agent view is used to describe the set of legitimate users of the system, represented inside the Grid as nodes. Each user is provided by a set of actions he can do, represented by the set X, e.g., to save a file on his file system or to start a computation on his personal computer, and by a set of goals he would fulfill, represented as the set G, e.g., he wants to reserve half of his available memory for his data or he has to obtain the result of a computation in two hours. These actions X can be compared to the operations that are recognized by the system. Functions $skills$ and $goals$ link each agent with the actions he can perform and with the goals he would obtain. Function R is a sort of action-consequence function, relating sets of actions with the goals they allow to fulfill, e.g., to obtain the results of a computation in two hours, the user has to start the computation on his personal computer.*

A social structure is modeled as a collection of agents, playing roles regulated by norms where "interactions are clearly identified and localized in the definition of the role itself" [19]. The notion of role is notable in many fields of Artificial Intelligence and, particularly, in multiagent systems where the role is viewed as an instance to be adjoined to the entities which play the role. The institutional view is defined as follows:

Definition 2 (Institutional view) $\langle RL, IF, IG, X, igoals : RL \rightarrow 2^{IG}, iskills : RL \rightarrow 2^X, IR : G \rightarrow 2^X \rangle$ *consists of a set of role instances RL, a set of institutional facts IF, a set of public goals attributed to roles IG, a set of actions X, a function $igoals$ that relates with each role the set of public goals it is committed to, a function $iskills$ that describes the actions each role can perform, and a set of institutional rules IR that relates a set of actions and the set of institutional facts they see to.*

Example 2 *The institutional view represents in the Grid scenario a model for the role based access control policy. For example, consider a Grid system with the two basic roles of VO administrator and VO member where the VO administrator has the possibility to assign to the VO members the privileges they need to enable the access to its resource. Our approach gives the opportunity to define not only the capabilities of a particular role but it allows also the institutional goals associated to roles. For example, a user asks for saving a file on the file system of another node. This user is associated to a role, since he belongs to a virtual organization regulated by a RBAC policy. The request can be processed either by the local VO administrator or by the user that has received the request. If the user requesting the service has a role that can perform this action, the request is accepted and the file is saved.*

In our model, we introduce the third submodel, the role assignment view, which links the agent and the institutional view to each other, by relating agents to roles.

Definition 3 (Assignment view) $\langle A, RL, roles : RL \rightarrow A \rangle$ *consists of a set of agents A, a set of role instances RL, and a function $roles$ assigning a role to its player in A.*

Finally, the combined view unifies the agent view and the institutional view, thanks to the assignment view, providing a unified conceptual metamodel:

Definition 4 (Combined view) *Let $\langle A, RL, roles : RL \rightarrow A \rangle$ be a role assignment view for the agents and role instances defined in the agent view $\langle A, F, G, X, goals : A \rightarrow 2^G, skills : X \rightarrow 2^A, R : G \rightarrow 2^X \rangle$ and institutional view $\langle RL, IF, IG, X, igoals : RL \rightarrow 2^{IG}, iskills : X \rightarrow 2^{RL}, IR : IG \rightarrow 2^X \rangle$. The role playing agents are $RPA = \{\langle a, r \rangle \in A \times RL \mid r \in roles(a))\}$. The combined view associates with the role playing agents the elements of the agent and institutional view.*

Our model is a directed labeled graph whose nodes are instances of the metaclasses of the metamodel, e.g., agents, goals, facts, and whose arcs are instances of the metaclasses representing relationships between them, dependencies. Our modeling is based on the theory of the social power and dependence pioneered by Castelfranchi [7] as starting point and then developed by Conte and Sichman [13]. In a multiagent system, since an agent is put into a system that involves also other agents, he can be supported by the others to achieve his own goals if he is not able to do them alone. This leads to the concept of power representing the capability of a group of agents (possibly composed only by one agent) to achieve some goals (theirs or of other agents) performing some actions without the possibility to be obstructed.

The notion of power brings to the definition of a structure with the aim to show the dependencies among agents, called dependence network. In order to define these relations in terms of goals and powers, we adopt, as said, the methodology of dependence networks as developed by [13] without distinguishing AND and OR dependencies. Note that Definition 5 of dependence networks, differently from Boella et al. [6] in which the function *dep* is $2^A \times 2^A \rightarrow 2^{2^G}$, defines *dep* in the view of the algorithmic approach where we are interested in obtaining the relation of one agent, at each iteration cycle, with the other agents for a particular goal. A dependence network is defined as follows:

Definition 5 (Dependence Networks (DN)) *A dependence network is a tuple $\langle A, G, dep \rangle$ where:*

- *A is a set of agents and G is a set of goals;*
- *dep : $A \times 2^A \times G$ is a relation which maps each pair of an agent and a set of agents with the goal on which the first depends on the second.*

In early requirements analysis, we model the dependencies among the agents and the roles associated to the agents of the organization representing, in this way, the domain stakeholders. Figure 2 represents a dependence networks where plain arrows represent material dependencies while the dotted ones represent institutional dependencies.

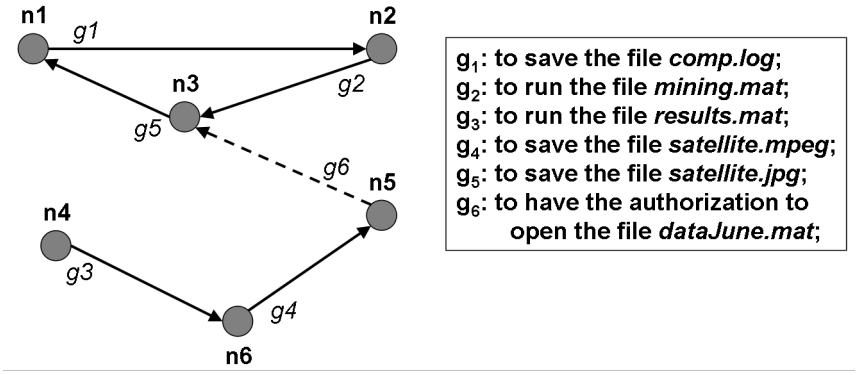

Figure 2: An example of dependence network.

4 Algorithms

In this section, we present three algorithms to automatize the building of the two dependence networks from the agent view and the institutional one, and to check the consistency of the combined view which arises from the previous two views. Moreover, we propose an evaluation of the combined view in the context of coalition formation.

Algorithm 1 CREATE_DEPNETS takes as input the sets of agents A, goals G, actions X and the functions $goals$, $skills$ and R composing the agent view and returns the dependencies which can be formed due to the starting view. The algorithm works as follows: for each agent a in the set A, it checks what are its goals g thanks to the function $goals$ and it finds what are the actions ac which can lead to an achievement of the goal a thanks to the function R. At this point, the algorithm finds the agents which have the power to perform the selected actions thanks to the function $skills$ putting them in the set I. Finally, it takes the union of these agents θ on which a depends on and it builds the dependency $dep(a, \theta, g)$. Trivially, the algorithm works in $O(|Ag| \cdot |G| \cdot |X|)$.

Algorithm 2 CREATE_IDEPNETS starts with the sets of roles Rl, institutional goals IG, actions X and the functions $igoals$, $iskills$ and IR composing the institutional view and returns the institutional dependencies which can be formed due to the starting view. The algorithm works exactly as the previous one, that is why we omit its detailed description. The presented algorithms leads to the automatization of the building process of the two dependence networks representing them. These dependence networks are independent to each other at this point of the design process and they are unified only when the necessity to build the combined view arises. The building of the dependence network representing the combined view is much more complex than the previous one since the unification of two independent networks could lead to some inconsistencies. In particular, the causes of inconsistency are: given an agent of the agent view playing a role, if there is one of the roles to which it depends on in the institutional view which is not played by anyone, and if there is a conflict between a goal and an institutional goal of the same agent.

Algorithm 3 CHECK_DEPNETS takes as input the function $Roles$, previously defined in the assignment view, the set of dependencies composing the dependence network DEP, the set of dependencies composing the institutional dependence network $IDEP$ and returns a consistent combined dependence networks. The algorithm works as follows: for each agent a in A, it takes all the roles on which it depends on, given its role thanks to the function Roles(a), and the insitutional goals Ig on which the dependency is based. Then it checks the incompatibility causes: if there is a conflict between the goals of a and the institutional goals of the role r of a then it returns NON_CONSISTENT otherwise it checks, for all the roles $r \in \theta^{'}$ on which a depends on, if they have a player thanks to the function Players(r), returning an inconsistency if Players$(r) = \emptyset$. If no inconsistencies are found, the algorithm takes all the agents A_i which play a role on which a depends on. At the end, it takes the union of these agents Γ and builds the conditional dependence network composed by dependencies Cdep(a, Γ, Ig). The algorithm works in $O(|Ag| \cdot |2^X| \cdot |G|)$.

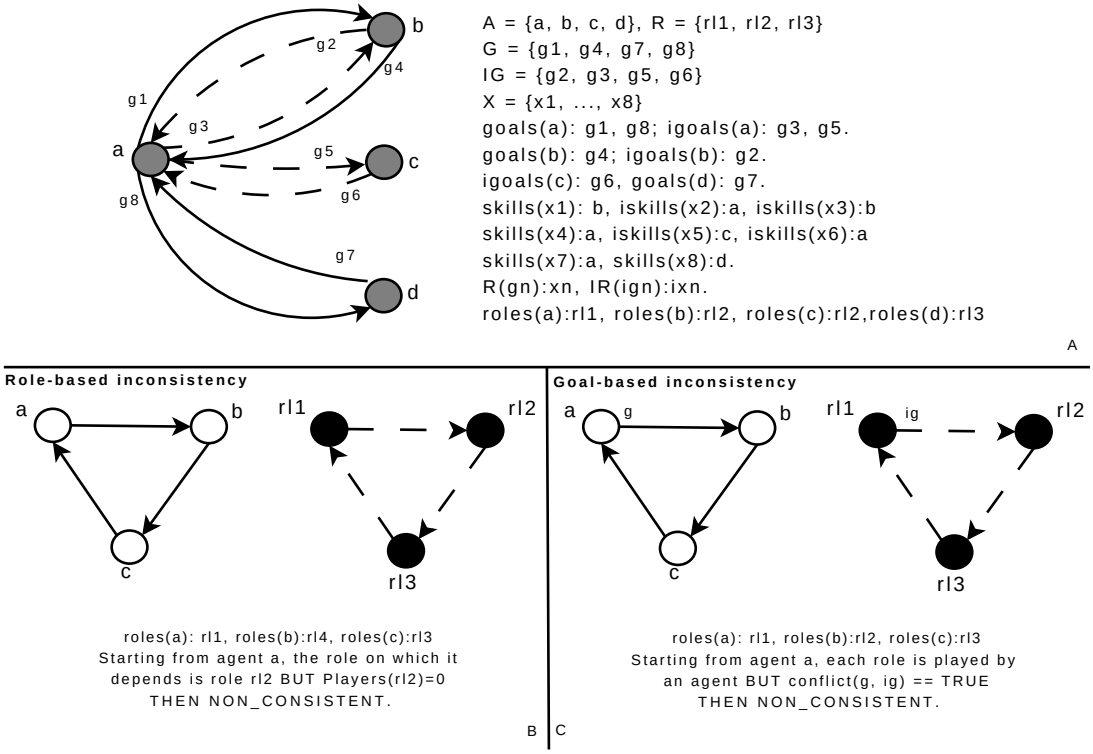

Figure 3: An example of combined dependence network and the possible conflicts.

An example of the application of the algorithm is provided in Figure 3.A while in Figure 3.B-C are provided two examples of the two kinds of conflicts which are detected by the automatic building of the combined dependence networks.

Input: $\langle A; G; X; R : G \rightarrow 2^X; goals : A \rightarrow 2^G; skills : X \rightarrow 2^A \rangle$
Output: dep : $A \times 2^A \times G$

```
forall a ∈ A do
    forall g ∈ goals(a) do
        forall ac ∈ R(g) do
            I_i = skills(ac);
            i++;
        end
    end
    θ = ∪_j I_j;
    dep(a, θ, g);
end
```

Algorithm 1: CREATE_DEPNETS

Input: $\langle Rl; IG; X; IR : IG \rightarrow 2^X; igoals : Rl \rightarrow 2^{IG}; iskills : X \rightarrow 2^{Rl} \rangle$
Output: Idep : $Ag \times 2^{Ag} \times G$

```
forall r ∈ Rl do
    forall ig ∈ igoalsr do
        forall iac ∈ IR(ig) do
            I_i = iskills(iac);
            i++;
        end
    end
    θ' = ∪_j I_j;
    Idep(a, θ', ig);
end
```

Algorithm 2: CREATE_IDEPNETS

Input: Roles : $Ag \rightarrow RT$; Players : $RT \rightarrow Ag; DEP; IDEP; goals(a) : x \in 2^G$
Output: $Consistecy$

```
forall a ∈ Ag do
    forall θ' : Idep(Roles(a), θ', Ig) do
        forall g ∈ goals(a) do
            if conflict(g, Ig) then
                NON_CONSISTENT;
            end
        end
        forall r ∈ θ' do
            if Players(r) = ∅ then
                NON_CONSISTENT;
            end
            else
                A_i = Players(r)
                i++;
            end
        end
    end
    Γ = ∪_j A_j
    Cdep(a, Γ, Ig)
end
CONSISTENT;
```

Algorithm 3: CHECK_DEPNETS

5 Related work

The idea of focusing the activities that precede the specification of software requirements, in order to understand how the intended system will meet organizational goals, has been first proposed in requirements engineering by Eric Yu his i* model [18]. The rationale of the i* model is that by doing an earlier analysis, one can capture not only the what or the how, but also the why a piece of software is developed. As stated throughout the paper, the most important inspiration source for our model is the TROPOS methodology [5] that spans the overall software development process, from early requirements to implementation. Other approaches to software engineering are those of KAOS [8], GAIA [17], AAII [12] and MaSE [11] and AUML [1]. The comparison of these works is summarized in Figure 4. The main difference between these approaches and our one consists in the use of the notion of institution.

Different approaches on the application of the notion of institution and role within open multiagent systems are defined in Sierra et al. [14], Bogdanovych et al. [4] and Vazquez-Salceda et al. [16] and Dastani et al. [9].

	Early requirements	Late requirements	Architectural design	Detailed design
i*	X	X		
Kaos		X		
GAIA		X	X	
AAII and MaSE			X	X
AUML				X
TROPOS	X	X	X	X

Figure 4: Comparison among different software engineering methodologies.

6 Conclusions

In this paper we formalize an extension of dependence networks by adding roles, presented in the institutional and combined views. We present three algorithms which permit to build and check the consistency among agent view and institutional view. Using dependence networks as methodology to model a multiagent systems advantage us from different points of view. First, they are abstract, so they can be used for conceptual modeling, simulation, design and formal analysis. Second, they are used in high level design languages, like TROPOS [5], so they can be used also in software implementation.

A first evaluation of the usefulness of the algorithm CHECK_DEPNETS can be found in the context of coalition formation. An improvement of the TROPOS approach [5] consists in the introduction of a notion of coalition for dependence networks. We propose the usage of a reciprocity-based coalition formation theory in which each agent belonging to the coalition has to contribute something and to get something out of it. A definition of reciprocity-based coalitions is provided in Boella et al. [3]. Considering our three views, three kind of reciprocity based coalitions can be highlighted: coalitions based only on institutional dependencies called institutional coalitions, coalitions based only on material dependencies called material dependencies and coalitions based on both the two kinds of dependencies called full coalitions. An example of these kinds of coalitions is presented in Figure 3.A where coalition $\{a, b\}$ is a full coalition, coalition $\{a, c\}$ is an institutional coalition and coalition $\{a, d\}$ is a material coalition. These three classes of coalitions may influence agents' decisions about goals' execution, establishing preferences on what coalition you want to belong to. A deeper analysis of this issue is left for future research.

References

[1] B. Bauer, J. P. Müller, and J. Odell. Agent UML: A formalism for specifying multiagent software systems. *Software Engineering and Knowledge Engineering*, 11(3):207–230, 2001.

[2] Guido Boella, Leendert van der Torre, and Serena Villata. Changing institutional goals and beliefs of autonomous agents. In *PRIMA*, pages 78–85. Springer, LNCS, 2008.

[3] Guido Boella, Leendert van der Torre, and Serena Villata. Social viewpoints for arguing about coalitions. In *PRIMA*, pages 66–77. Springer, LNCS, 2008.

[4] Anton Bogdanovych, Marc Esteva, Simeon J. Simoff, Carles Sierra, and Helmut Berger. A methodology for developing multiagent systems as 3d electronic institutions. In *AOSE*, volume 4951 of *Lecture Notes in Computer Science*, pages 103–117. Springer, 2007.

[5] Paolo Bresciani, Anna Perini, Paolo Giorgini, Fausto Giunchiglia, and John Mylopoulos. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, 2004.

[6] Patrice Caire, Serena Villata, Guido Boella, and Leendert van der Torre. Conviviality masks in multiagent systems. In Lin Padgham, David C. Parkes, Jörg Müller, and Simon Parsons, editors, *AAMAS (3)*, pages 1265–1268. IFAAMAS, 2008.

[7] C. Castelfranchi. The micro-macro constitution of power. *Protosociology*, 18:208–269, 2003.

[8] A. Dardenne, A. van Lamsweerde, and S. Fickas. Goal-directed requirements acquisition. *Sci. Comput. Program.*, 20(1-2):3–50, 1993.

[9] Mehdi Dastani, Birna van Riemsdijk, Joris Hulstijn, Frank Dignum, and John-Jules Ch. Meyer. Enacting and deacting roles in agent programming. In James Odell, Paolo Giorgini, and Jörg P. Müller, editors, *AOSE*, volume 3382 of *Lecture Notes in Computer Science*, pages 189–204. Springer, 2004.

[10] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *Int. J. of Supercomputer Applications*, 15, 2001.

[11] Juan C. García-Ojeda, Scott A. DeLoach, Robby, Walamitien H. Oyenan, and Jorge Valenzuela. O-mase: A customizable approach to developing multiagent development processes. In *Agent-Oriented Software Engineering VIII, 8th International Workshop*, volume 4951 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2007.

[12] David Kinny, Michael P. Georgeff, and Anand S. Rao. A methodology and modelling technique for systems of bdi agents. In Walter Van de Velde and John W. Perram, editors, *MAAMAW*, volume 1038 of *Lecture Notes in Computer Science*, pages 56–71. Springer, 1996.

[13] Jaime Simão Sichman and Rosaria Conte. Multi-agent dependence by dependence graphs. In *AAMAS*, pages 483–490. ACM, 2002.

[14] Carles Sierra, John Thangarajah, Lin Padgham, and Michael Winikoff. Designing institutional multiagent systems. In Lin Padgham and Franco Zambonelli, editors, *AOSE*, volume 4405 of *Lecture Notes in Computer Science*, pages 84–103. Springer, 2006.

[15] Serena Villata. Institutional social dynamic dependence networks. In Guido Boella, Gabriella Pigozzi, Munindar P. Singh, and Harko Verhagen, editors, *NORMAS*, pages 201–215, 2008.

[16] J. Vzquez-Salceda, V. Dignum, and F. Dignum. Organizing multiagent systems. *Autonomous Agents and Multi-Agent Systems*, 11(3):307–360, 2005.

[17] M. Wooldridge, N. R. Jennings, and D. Kinny. The GAIA methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems*, 3(3):285–312, 2000.

[18] E. Yu. Modeling organizations for information systems requirements engineering. In *First IEEE International Symposium on Requirements Engineering*, pages 34–41, 1993.

[19] F. Zambonelli, N. Jennings, and M. Wooldridge. Developing multiagent systems: The gaia methodology. *IEEE Transactions of Software Engineering and Methodology*, 12:317370, 2003.