Author(s): Faulkner, Stéphane., Dehousse, Stéphane., Kolp, Manuel., Mouratidis, Haralambos., Giorgini, Paolo.
Article Title: Delegation Mechanisms for Agent Architectural Design
Year of publication: 2005
Citation: Faulkner, S. et al. (2005) 'Delegation Mechanisms for Agent Architectural Design' Proceedings IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT), Compiegne - France, pp. 503 – 507.
Link to published version: http://dx.doi.org/10.1109/IAT.2005.65
DOI: 10.1109/IAT.2005.65

Delegation Mechanisms for Agent Architectural Design

Stéphane Faulkner[1], Stéphane Dehousse[1], Manuel Kolp[2], Haralambos Mouratidis[3] and Paolo Giorgini[4]

[1]*Information Management Research Unit*
University of Namur,
{stephane.faulkner, stephane.dehousse}@fundp.ac.be

[2]*Information System Unit, University of Louvain,*
kolp@isys.ucl.ac.be

[3]*School of Computing and Technology, University of East London*
haris@uel.ac.uk

[4]*Department of Information and Communication Technology*
University of Trento - Italy
giorgini@dit.unitn.it

Abstract

Multi-agent Systems (MAS) are now being considered a promising architectural approach for designing collaborative information systems. In such a perspective, the concept of delegation has often been considered as a key concept for modeling cooperative behavior in MAS. However, despite considerable work on delegation mechanisms in MAS, few research efforts have aimed at truly defining a delegation model for designing MAS. This paper deals with this issue in defining the foundations for a delegation model aimed to help developers during the phase of designing collaborative MAS.

1. Introduction

Collaborative information systems have been growing and gaining substance in technological infrastructure (e.g., middleware and Web technologies) and application areas (e.g., Business Process Management, e-Commerce, e-Government, and virtual enterprises). They involve large networks of information systems that manage large amounts of information and computing services and cooperate to fulfill their mission.

In the last few years, one promising source of ideas for designing collaborative information systems is the field of multi-agent systems (MAS) architectures. They appear to be more flexible, modular, and robust than traditional including object-oriented ones. Research in this area has notably emphasized that MAS is conceived as a society of autonomous, collaborative, and goal-driven software components (agents).

In such a distributed and cooperative perspective, the concept of delegation has often been considered as a key concept for modeling collaborative behavior in MAS [3, 11, 14]. Delegation allows an agent to assign authority and/or responsibility for the execution of an action or the fulfillment of a goal. However, even if the concept of delegation has received increasing attention, the majority of researches have been focusing either on requirement analysis [7, 8] or on the definition of delegation models through communication acts [14, 17] or socio-cognitive theories [4].

Few research efforts have aimed at truly defining delegation models for designing MAS architectures. This paper deals with this issue in defining a "core" set of concepts, including relationships and constraints that are fundamental to propose a delegation model. This model is aimed to help the developers during the design phase of collaborative MAS.

The paper is structured as follows. Section 2 provides an overview of the delegation model and details some concepts using the Z specification language. Section 3 applies the delegation model on a case study. Section 4 discusses some of the related work and Section 5 summarizes the contributions of the paper and proposes some possible extensions.

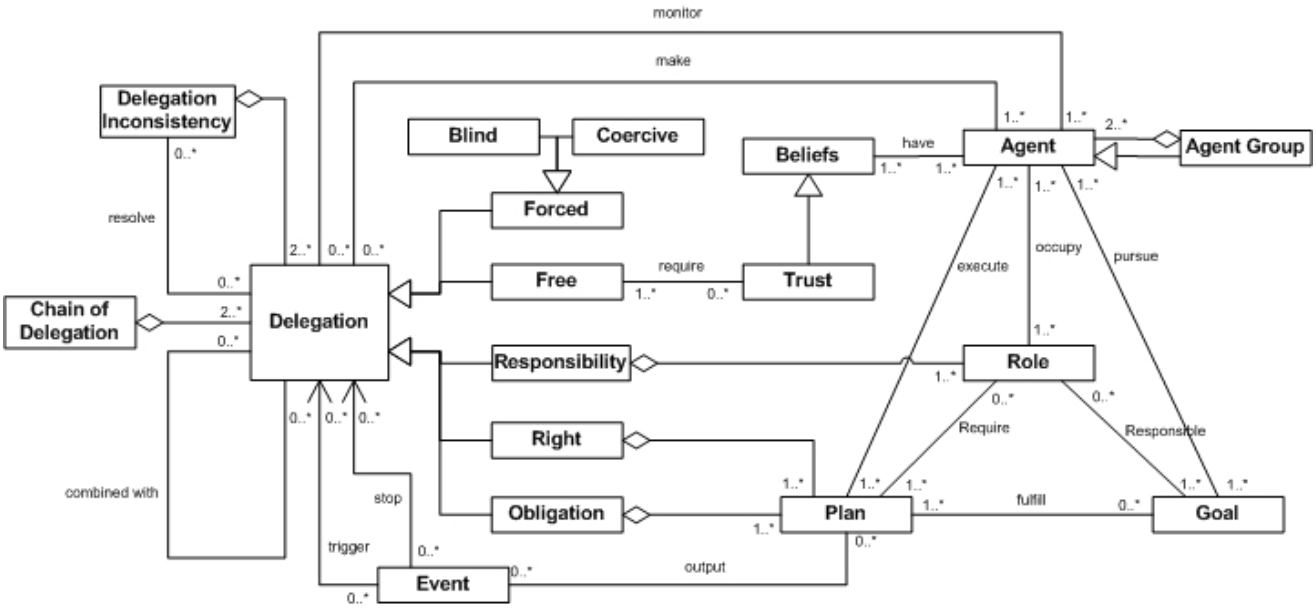


Figure 1. The delegation Model

2. The Delegation Model

Figure 1 depicts the Delegation Model using a UML type class diagram. An Agent is an intentional entity, which has some Beliefs and Goals that guide its action. A Belief defines current states about the MAS, while a Goal defines desired states that the Agent wants to bring about. Each Agent occupies one or more Roles that are a characterization of the expected behavior of an Agent in the MAS. A Role requires a set of Plans to fulfill the Goals for which it is responsible. A Plan defines a sequence of actions.

In order to easier or better achieve its Goals, one or more Agents can delegate to another Agent, called the delegatee, the execution of a Plan or the responsibility of a Role (i.e., Responsibility). The Delegation of a plan can be classified as a Right or an Obligation. A Delegation of Right defines one or more Plans that the delegatee is allowed (or not) to execute, while a Delegation of Obligation defines Plans that the delegatee must execute. The Delegation can also be self-determined (i.e., Free), or imposed (i.e., Forced) by the context of the environment (i.e., Blind) or by other Agents (i.e., Coercive). In the case of Free Delegation, Agent requires Trust in the delegatee in order to effectively make the Delegation. Trust is a Belief an Agent has on the ability and the dependability of another agent to execute a Plan or to achieve a Goal. An agent can also be allowed to re-delegate a given delegation to another agent. This kind of situation leads to a Chain of Delegation.

A Delegation is trigged or stopped by an Event. An Event is an instantaneous state of the system. It is either output of a Plan, or exogenous to the system.

Because Agents can collaborate autonomously, Delegation Inconsistencies may exist. A Delegation Inconsistency concerns two or more Delegations that cannot be assigned to the same Agent in the system. The resolution to a Delegation Inconsistency takes the form of new Delegations.

Figure 1 shows only concepts and relationships. In the next sections, we specify attributes and constraints of some concepts using the Z state-based specification language [19]. Following UML to Z translation rules from Shroff and France [18], attributes are specified as Z state variables and constraints as Z predicates. However, due to a lack of space we only specify in details the concept of Delegation and its specializations (i.e., Responsibility, Right, Obligation, Free and Forced).

2.1 Delegation and Chain of Delegation

Figure 2 shows the Z formal specification of the Delegation concept. The first part of the specification represents the definition of types. A given type defines a finite set of items. The Delegation specification first defines the type Name (which represents the Name attribute) by writing [Name]. Such a declaration introduces the set of all names, without making assumptions about the type (i.e., whether the name is a string of characters and numbers, or only characters, etc.).

More complex and structured types are defined with schemas. A schema groups a collection of related

declarations and predicates into a separate namespace or scope. The schema in Figure 2 is entitled Delegation and is partitioned by a horizontal line into two sections: the declaration section above and the predicate section below the line. The declaration section introduces a set of variable declarations, while the predicate section provides predicates that constrain values of the variables.

[Name]
[Actor] := Agent | Agent Group
[Value]

Delegation

delegation_name : *Name*
delegator : *Actor*
delegatee : *Actor*
trigger: *Event*
expiration : ℙ *Event*
depth : *Value*
linked_ delegation : *Boolean*
monitor : *Actor*
combined_with : *Delegation* ↔ *Delegation*

name ≠ ∅ ∧ delegator ≠ ∅ ∧ delegatee ≠ ∅ ∧
trigger ≠ ∅
dom depth = ℕ
disjoint⟨Responsibility, Right, Obligation⟩
disjoint⟨Free, Forced⟩
∀ d1, d2: Delegation ● d1.combined_with = d2
⇒ d2.delegator = d1.delegator ∧
d2.delegatee = d1.delegatee
∀ d1, d2: Delegation ● d1.combined_with = d2 ∧
d1.expiration
⇒ d2.expiration

Figure 2. Delegation Concept

A delegation defines an action by which an agent assigns to another agent the responsibility of a role, or the right or the obligation to execute a plan. It is specified with the following variables:

- *delegator* and *delegatee*: the delegator identifies the actor initiating the delegation, while the delegatee identifies the actor to which the delegation has been assigned. An actor is either an agent or a group of agent. The concept of agent group allows modeling multiple delegations (i.e., delegation from a group of agent and/or delegation to a group of agent). An agent group is specified with two variables: leader and joint. The leader variable defines one or more agents that cannot leave the group without causing the extinction of the group, and consequently the revocation of the delegation. The joint variable determines whether all or only one agent of the group has to contribute to the fulfillment of the delegation
- *trigger* and *expiration*: the trigger defines the reason why the delegation starts and the expiration why the delegation stops.
- *depth*: defines restrictions on re-delegations. A depth set to 0 implies that no re-delegation is allowed, while a depth set to a value greater than 0 allows a chain of delegation composed of a number of re-delegations equivalent to the depth's value. The depth attribute is particularly interesting in a chain of delegation [7, 11, 12, 15] to control delegation propagation and avoid erratic re-delegation that could destabilize the system.
- *linked_delegation*: specifies if the delegation stops its effects when the delegator leaves the system. This variable is useful to design open MAS architectures [9] in which agents can constantly integrate or leave the system.
- *monitor*: identifies one or more agents that monitor the delegation. Monitoring is used in order to check and to evaluate the fulfillment of a delegation assigned to a trusted or distrusted agent. The act of monitoring can be done by the delegator himself or by other agents. Depending on the kind of delegation, a monitor is required or not.
- *combined_with*: specifies if the delegation is combined with another delegation. By combining delegation of responsibility and delegation of obligation, an agent can delegate a role and force the execution of one or more plans in order to fulfill a goal for which this role is responsible. Such a combination is useful to constraint some agent's behaviors in cooperative MAS. For instance, an agent may delegate to another agent a role with the goal get personal data and a delegation of obligation on the plan, encrypt the data. For the delegator, this ensures that whatever the plans used to get the personal data, they will be encrypted.

A Delegation Chain is a non-empty set of delegations. The predicate section of the Chain schema specifies that all delegations which belong to the same chain have an identical value for their respective linked_delegation and combined_with variables, and that for each re-delegation the depth variable is decreased by 1.

Delegation Chain

chain: ℙ *Delegation*

delegations ≥ 2
∀ d_1 , d_2 : Delegation, c : Chain
$d_1 \in c \wedge d_2 \in c$
⇒ d_1.linked_delegation = d_2.linked_delegation
∧ d_1.combined_with = d_2.combined_with
∀ d_1 , d_2 : Delegation, c : Chain, x ∈ ℕ
$d_1 \in c \wedge d_2 \in c \wedge d_1$.delegatee = d_2.delegator ∧
d_1.depth = x
⇒ d_2.depth = x - 1

Figure 3. Chain of Delegation Concept

2.2 Delegation of responsibility

A delegation of responsibility defines a set of roles that the delegatee has to occupy. Roles provide the building blocks for agent social systems and the requirements by which agents interact [6]. The concept of role is important to abstractly model the agents in multi-agent systems and helpful to manage its complexity without considering the concrete details of agents (e.g., implementation architectures and technologies) [18]. They enable separation between different functionalities of software agents (e.g., mobility from collaboration), or between different phases of the development process (e.g., functions in the design from methods in the implementation).

A delegation of responsibility involves that the delegatee who has to occupy the roles is constrained to fulfill the goals for which the roles are responsible. However, the delegation of responsibility is the least restrictive kind of delegation. Indeed, it only constrains the delegatee to fulfill goals without constraining the selection or the execution of plans which will make possible to fulfill them.

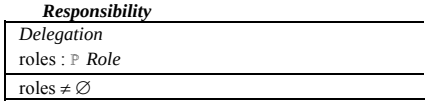


Figure 4. Responsibility Delegation Concept

2.3 Delegation of right

A delegation of right defines a set of plans that the delegatee is allowed to execute. It is specified in Figure 5. This definition allows the representation of the concept of authorization or permission [4, 7, 10, 12] that are often described in the literature. An authorization is defined as the right to grant to an agent an access to a resource in the system. In our case, this kind of authorization can be modeled by a delegation of right on the plan with which the resource is associated. For instance, the authorization on a personal data resource can be modeled as a delegation of right on the plan which accesses to the personal data.

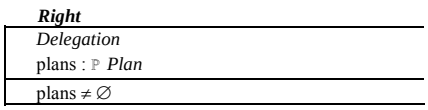


Figure 5. Right Delegation Concept

2.4. Delegation of obligation

A delegation of obligation defines a set of plans that the delegatee must (or not) execute. It is specified in Figure 6. The polarity variable describes whether the obligation is positive or negative. A negative delegation of obligation enables an agent to forbid another agent to execute a plan. It corresponds to the notion of prohibition [7].

[Obligation_polarity] := Positive | Negative

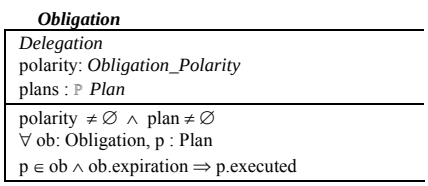


Figure 6. Obligation Delegation concept

2.5. Forced Delegation

Delegation is generally strictly based on trust. However, [4, 8] has mentioned rarer cases of delegation where trust is not required. This kind of delegation is called a forced delegation. It occurs when an agent is in a situation of blind or coercive delegation [4].

A blind delegation occurs when the delegator does not have sufficient information to form a trust opinion on the delegatee. Compare to other forms of delegation, a blind delegation requires a monitor in order to compensate the lack of trust in the delegatee.

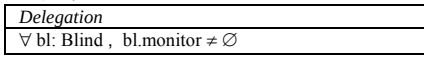


Figure 7. Blind Delegation Concept

A coercive delegation implies one or more agents, called the requesters, that force the delegator to delegate the responsibility of a role or the execution of a plan. The requesters can force the delegation towards a given delegatee (i.e., full coercive delegation), or can only force the delegation without mentioning the delegatee (i.e., partial coercive delegation).

[Status] := Full | Partial

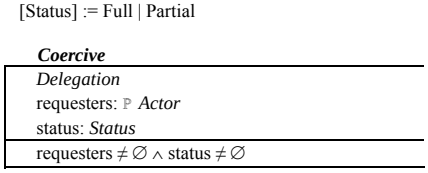


Figure 8. Coercive Delegation Concept

2.6 Free Delegation

Contrary to the forced delegation, a free delegation requires trust in the delegatee. Indeed, a delegation implies that the delegator exposes himself through the behavior of a delegatee. A free delegation is defined as an intentional delegation (i.e., the delegator can freely decide to delegate or not). Consequently, the delegator decides to delegate only if it trusts the delegatee.

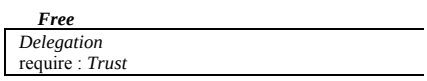


Figure 9. Free Delegation Concept

Trust is a belief an agent has on the ability and the dependability of another agent to execute a plan or to achieve a goal. The concept of trust is essentially human mental state and therefore difficult to transpose in the agent paradigm. Authors have, many times [3, 4, 13, 16], tried to formalize this concept through, for example, computation of different components without being able to achieve the definition of a universally admitted model. However, due to a lack of space, we do not address in this work the issues related to trust models.

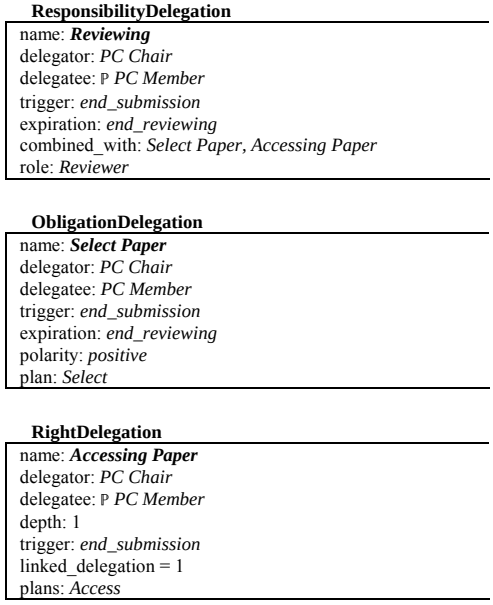
3. Using the Delegation Model in a Case Study

The following examples are part of a substantial case study on the development of an open system that supports the management of paper submission and the reviewing process for a conference. We focus on the phase of reviewing which involves three categories of actors: *PC Chair (PCC)*, *PC Member (PCM)* and *Scientific Expert (SE)*.

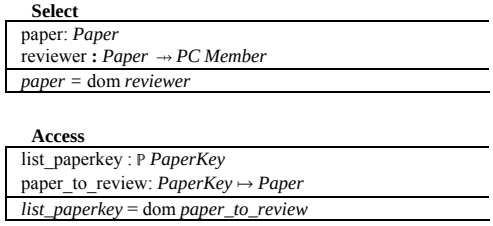
Example 1: *The PCC distributes submitted papers for reviewing to the PCM. Each PCM have to select papers they agree to review. For selecting a paper, a PCM depends on PCC to get access to the submitted papers*.

The example suggests the use of a delegation of responsibility (*Reviewing*) from the PCC to the PCM about the role of "reviewer". This delegation of responsibility is combined with the delegation of obligation to select a paper (*Select Paper*) and the delegation of right to access a paper (*Accessing Paper*).

The rest of this section describes the mentioned delegations using the Z specification language.

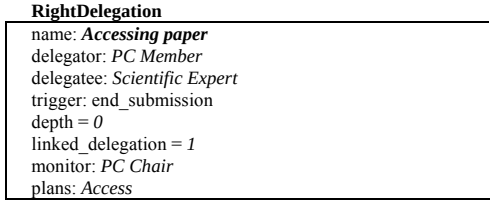


We specify the plans Select Paper and Accessing Paper that are used in both delegations as follows:



Example 2: *A PCM has to review a paper for which he has no enough expertise. The PCM chooses to delegate the review to a Scientific Expert with which he is not familiar*.

This example illustrates the situation of a chain of delegation. The chain of delegation concerns the right to execute the plan Access. This plan was delegated firstly by the *PCC* to the *PCM* and then by the *PCM* to the *SE*. This re-delegation is only possible if the first delegation have a depth attribute with a value greater than zero. As this re-delegation also corresponds to a blind delegation, we define the *PCC* like monitor.



Example 3: *The PCM cannot achieve the review of the selected paper because the author is a colleague. The PCM has to ask to a SE, designated by the PCC, to do the review of the paper. To make the review, the SE depends on the PCM to get access to the submitted papers.*

While the previous example illustrated a blind delegation, this one refers to a coercive delegation. Indeed, the *PCC* requests that the *PCM* delegates the reviewing of the paper to a specific *SE*. Therefore, this form of delegation corresponds to a full coercive delegation. In case of a partial coercive delegation, the *PCM* could have chosen himself the delegatee.

RightDelegation

name: **Accessing paper**
delegator: PC Member
delegatee: *Scientific Expert*
trigger: *end_submission*
depth = *0*
linked_delegation = *1*
requester: *PC Chair*
status: *full*
plans: *Access*

4. Related work

Giorgini et al. [8] in their attempt to model security requirements emphasize the distinction between delegation of execution, i.e. at-least delegation, and delegation of permission, i.e. at-most delegation. The model proposed in this work takes into account this distinction respectively through the concepts of delegation of obligation and delegation of right. Moreover, from the idea of Norman and Reed [15] that have argued for a theory of delegation on an action to be done and on a goal to be achieved, our model proposes the delegation of responsibility. Such a delegation constrains the delegatee which has to occupy the roles to fulfill the goals for which the roles are responsible.

The concept of delegation chain has been largely discussed in the literature [7, 12, 15]. In addition, [8] mentions the combination of several delegations in order to deal with complex delegation behaviors. Our model allows specifying a chain, as well as a combination of delegations. Moreover, we add the concept of agent group in order to define multiple delegations (or sponsoring) [15].

Finally, Castelfranchi and Falcone mention [4] that in exceptional cases the delegator is not free to delegate. From this statement, we propose to distinct two classes of forced delegation: coercive and blind. The importance to handle forced delegation has been confirmed by [8] which recognize, after a large study, that for pragmatic reasons agents may be forced to delegate to agent they do not trust.

5. Conclusion and Future Work

MAS constitute a highly promising software architectural approach for collaborative application domains such as peer-to-peer, information retrieval, semantic web services or e-business. The literature has often considered the concept of delegation as a key concept for modeling cooperative behavior in MAS [3, 11, 14].

Unfortunately, despite considerable work in software design and architecture during the last decade, few research efforts have aimed at defining a delegation model for designing collaborative mechanisms when MAS architectures are developed.

This paper has attempted to gather the key points of different perspectives on delegation, as discussed in Section 4. It has defined a conceptual model to design delegation in MAS architectures. The main contribution of this work is that our approach aims at modeling delegation to use it at the design level, while others approaches focus on requirement analysis or on the definition of delegation models through communication acts or socio-cognitive theories.

The research reported here calls for further work. We are currently working on:

- the extension of the delegation model with trust management concepts;
- the specification of the delegation model according to a set of rules in order to perform consistency analysis to be included in verification tools such as PVS;
- the identification of a suitable set of delegation abstractions, inspired by organizational metaphors, to be used during the detailed design phase of the MAS architecture.

6. References

[1] K. S. Barber and J. Kim, “Soft Security: Isolating Unreliable Agents”, Proceedings of the AAMAS 2002 Workshop on Deception, Fraud and Trust in Agent Societies, Bologna, Italy, July 2002.

[2] M. Blaze, J. Feigenbaum, J.Ioannidis, and K. Keromytis, ”The Role of Trust Management in Distributed System Security”, Secure Internet Programming, J. Vitec, and C. Jensen (Eds)., 1999.

[3] J. Carter, E. Bitting and A. A. Ghorbani, "Reputation Formalization Within Information Sharing Multiagent Architectures", Computational Intelligence, Vol 18, No. 4, pp. 45-64, 2002.

[4] Castelfranchi C. and Falcone R., "Principles of trust for MAS: cognitive anatomy, social importance, and quantification". In Proceedings of the International

Conference of Multi-Agent Systems (ICMAS'98), pp. 72–79, 1998.
[5] S. Faulkner, T. T. Do, T. Hoang and M. Kolp, Architectural Styles and Patterns for Multi-Agent Systems, in N. Ichalakaranje, R Khasla and L.C. Jain (Eds), Design of Intelligent Multi-Agent Systems: Human-Centredness, Architecture, Learning and Adaptation, vol. 162, pp 67-98, Springer, 2004.
[6] J. Ferber, "Multi-agent Systems". Addison-Wesley, Reading, MA, 1999.
[7] P. Giorgini, F. Masscci, J. Mylopoulos, and N. Zannone, "Filling the Gap Between Requirements Engineering and Public Key/Trust Management Infrastructures", Proc. of the 2nd Int. Conf. on Trust Management iTrust 2004.
[8] P. Giorgini, F. Massacci, J. Mylopoulos and N. Zannone, "Modelling Social and Individual Trust in Requirements Engineering Methodologies", 3rd international conference on trust management (iTrust 2005), Rocquencourt, France, 23-26 May 2005.
[9] D. Huynh, N. R. Jennings, and N. R. Shadbolt, "FIRE: An integrated trust and reputation model for open multi-agent systems". In Proceedings of the 16th European Conference on Artificial Intelligence (ECAI), 2004.
[10] L. Kagal, T. Finin, and Y. Peng, "A Framework for Distributed Trust Management". In Proceedings of IJCAI-01, Workshop on Autonomy, Delegation and Control, 2001.
[11] Lalana Kagal et al., "Developing Secure Agent Systems Using Delegation Based Trust Management", InProceedings, Security of Mobile Multi-Agent Systems Workshop, Autonomous Agents and Multiagent Systems (AAMAS 2002) , July 2002.
[12] Ninghui Li, Joan Feigenbaum , Benjamin N. Grosof, "A Logic-based Knowledge Representation for Authorization with Delegation", Proceedings of the 1999 IEEE Computer Security Foundations Workshop, p.162, June 28-30, 1999
[13] L. Mui, M. Mohtashemi, and A. Halberstadt, "A Computational Model of Trust and Reputation", Proceedings of the 35th Hawaii International Conference on System Sciences, Big Island, Hawaii, January 2002.
[14] T. J. Norman, and C. A. Reed, "Delegation and responsibility", In Proceedings of the Seventh International Workshop on Agent Theories, Architectures, and Languages. Edited by C. Castelfranchi and Y. Lesp'erance. Springer-Verlag, Berlin, pp. 136-149, 2001.
[15] T. J. Norman and C. A. Reed, "Group Delegation and Responsibility", In Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems, pages 491-498, 2002.
[16] S. D. Ramchurn, D. Huynh, and N. R. Jennings, "Trust in multi-agent systems". The Knowledge Engineering Review, 2004.
[17] J. Sabater, "Trust and Reputation for Agent Societies". Phd thesis, Universitat Autnoma de Barcelona, 2003.
[18] M. Shroff and R. B. France, "Towards a formalization of UML class structures in Z", Proceedings of the 21st International Computer Software and Applications Conference, IEEE Computer Society, 1997.
[19] J. M. Spivey, "The Z notation: a reference manual", Prentice Hall International (UK) Ltd., Hertfordshire, UK, 1992.
[20] H.C. Wong, and K. Sycara, "Adding Security and Trust to Multi-Agent Systems", Proceedings of Autonomous Agents'99 (Workshop on Deception, Fraud and Trust in Agent Societies), 1999.
[21] Qi Yan, Xin-Jun Mao and Zhi-Chang Qi, "Modeling role-based organization of agent system", UKMAS'02, 2002.