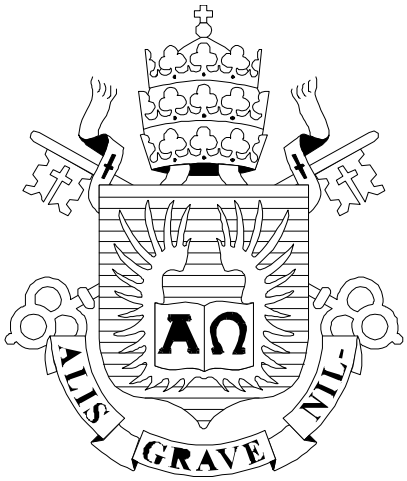

PUC

ISSN 0103-9741

Monografias em Ciência da Computação
n° 06/06

Using Testimonies to Enforce the Behavior of Agents

Viviane Torres da Silva

Fernanda Duran de Moura Augusto

Ricardo Choren

Carlos José Pereira de Lucena

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22453-900

RIO DE JANEIRO - BRASIL

Monografias em Ciência da Computação, No. 06/06 ISSN: 0103-9741
Editor: Prof. Carlos José Pereira de Lucena 02, 2006

Using Testimonies to Enforce the Behavior of Agents

Viviane Torres da Silva, Fernanda Duran de Moura Augusto, Ricardo Choren[1], Carlos José Pereira de Lucena

[1] SE/8 – IME/RJ, Pça General Tibúrcio 80 22290-270, Rio de Janeiro/RJ, Brazil

viviane@inf.puc-rio.br, fduran@inf.puc-rio.br, choren@de9.ime.eb.br, lucena@inf.puc-rio.br

Abstract. Governance copes with the heterogeneity, autonomy and diversity of interests among different agents in a multi-agent system (MAS) by establishing a set of norms. However, a governance enforcement mechanism usually checks norm violations from only one point of view, such as interaction. Besides putting aside other aspects, these mechanisms have an intrusive implementation, for instance they check every message. This paper presents a mechanism to support the implementation of governance in MAS, based on testimonies, i.e. agents can witness to facts that they know may be related to norm violations. This mechanism is composed of three subsystems: reputation, judgment and punishment. We focus on the judgment sub-system, which is responsible for receiving the testimonies and for providing a decision pointing out if an agent really violated a norm. We show the sub-system architecture and a general judgment process. Finally, we illustrate the use of our mechanism through a case study.

Keywords: open systems, multi-agent system, governance, norms and testimonies.

Resumo. Governança trata heterogeneidade, autonomia e diversidade de interesses entre diferentes agentes em um sistema multi-agente (SMA), estabelecendo um conjunto de normas. Entretanto, um mecanismo de aplicação de leis normalmente verifica a violação de normas apenas sob um aspecto, como interação, por exemplo. Além de outras implicações, estes mecanismos são intrusivos, pois eles inspecionam cada mensagem trocada entre os agentes. Este artigo apresenta uma abordagem que implementa um mecanismo de governança em SMA baseado em testemunhos. Agentes podem testemunhar fatos que estão relacionados à violação de normas as quais eles têm conhecimento. Este mecanismo é composto por três sub-sistemas: Reputação, julgamento e punição. Neste artigo, nós focamos o sub-sistema de julgamento, responsável por receber os testemunhos prover decisões apontando se o agente realmente violou uma norma. Mostraremos a arquitetura deste sub-sistema e um processo de julgamento genérico. Finalmente, ilustraremos a utilização do nosso mecanismo através de um caso de estudo.

Palavras-chave: sistemas abertos, sistemas multi-agentes, governança, normas e testemunhos.

In charge for publications:

Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC-Rio Departamento de Informática
Rua Marquês de São Vicente, 225 - Gávea
22453-900 Rio de Janeiro RJ Brasil
Tel. +55 21 3114-1516 Fax: +55 21 3114-1530
E-mail: bib-di@inf.puc-rio.br

Table of Contents

1 Introduction

Open multi-agent systems are societies in which autonomous, heterogeneous and independently designed entities can work towards similar or different ends (Lopez, 2003). In order to cope with the heterogeneity, autonomy and diversity of interests among the different members, governance (or law enforcement) systems have been defined. Governance systems enforce the behavior of agents by establishing a set of norms that describe actions that agents are prohibited, permitted or obligated to do (Boella, 2004), (Singh, 1999).

Several governance systems, such as (Minsky, 2000), (Paes, 2005), have been proposed to regulate the interaction between agents. They use a law-governed interaction (LGI) mechanism (Minsky, 2000) that mediates the interaction between agents in order to make them comply with the set of norms. Every message that an agent wants to send is analyzed by the mechanism. If the message violates an application norm, the message is not sent to the receiver.

Since those mechanisms interfere in every interaction between agents, they influence the systems' privacy and performance. The privacy of an agent is broken since every message sent from an agent to another must be inspected. The performance is a concern in large multi-agent systems, in which a great number of exchanged messages shall be checked. Besides these issues, systems based on LGI mechanisms do not consider every action executed by an agent since they are only concerned about message-driven events. Therefore, norms applied to actions not related to messages, such as reading or updating a resource, cannot be enforced.

Other governance systems, such as TuCSoN (Cremonini, 2000), provide support for the enforce-ment of norms that regulate the access to resources. TuCSoN provides a coordination mechanism to manage the interaction between agents and also an access control mechanism to handle communication events, in other words, to control the access to resources. In TuCSoN agents interact through a multiplicity of independent coordination media, called tuple centres. The access control mechanism controls agent access to resources by making the tuple centres visible or invisible to them. Although in TuCSoN norms can be described to govern the access to resources, the governance is restricted and only applied to resources that are inserted in tuple centre environments.

In this context we propose a governance multi-agent system mechanism (Silva, 2005) that does not invade the agent's privacy and that does not influence the functional performance of the system. The proposed mechanism does not interfere in the messages exchanged between agents or in the agents' access to resources. Besides, it does not impose any specific agent platform or environment.

The proposed governance mechanism is based on testimonies. During the system execution, agents themselves can witness to facts that they know may be related to norm violations. Since every agent knows a set of the application's norms, they can provide testimonies about actions that may be in violation of a norm. The result of an action execution can be perceived as a fact or event and, if this fact or event implies in a norm violation, it can be reported and proper measures can be taken. Thus, the mechanism presents a different approach to multi-agent system governance: it does not actively try to prevent a norm violation. Rather, it lets the system execute normally and, if norm violations are reported, it checks and penalizes the agent that misbehaved. This approach is based on the fact that it may be very difficult and re-straining to prevent

norm violations that can happen after every action execution done by the several agents in an open multi-agent system.

The governance mechanism is composed of three sub-systems: reputation, judgment and punishment. The reputation sub-system is responsible for calculating and tracking the agents' reputation. Upon entering a system, an agent has a default reputation value and it changes according to the occurrence of violations it makes. This sub-system also informs the reputation of an agent to the judgment sub-system and to other agents. The judgment sub-system is responsible for receiving the testimonies and for providing a decision pointing out if an agent really violated a norm. Finally, the punishment sub-system is responsible for applying penalties, specified in the norms, to the agents that are blamed for violating a norm.

In this paper we focus on the judgment sub-system, detailing its architecture and its judgment process. The rest of this paper is organized as follows. Section 2 presents an overall view of the testimony-based governance mechanism. Section 3 de-tails the judgment sub-system. Section 4 describes a case study and Section 5 presents conclusions and some future work.

2 The Testimony-based Governance Mechanism

The governance mechanism presented here is based on testimonies that agents provide attesting facts or events that may be norm violations. Since every agent knows a set of norms, it can report to the governance mechanism their violation. An agent can, for instance, witness to the breaking of an interaction protocol or to a disallowed resource access.

2.1 Governance Mechanism Assumptions

The testimony-based governance mechanism is funded in the following assumptions.

Assumption I: *Every agent knows every norm applied to itself.*

Such as in the real world where everyone should know a code of behavior, we assume that every agent must know all norms that can be applied to their actions independently of the system environment in which it is executing. When an agent enters in a system environment to play a role, it must acquire knowledge about all the norms applied to that specific role. This is important since an agent acting in violation of a norm chooses to do so being aware of that.

Assumption II: *Every agent can give testimonies about norm violations.*

Since an agent knows the norms that can be applied to it and to other related agents, it is able to state that one of these norms is violated. Agents may know about norms that regulate the behavior of agents that influence their execution. Every time an agent perceives the violation of a norm, it can give a testimony to the governance mechanism.

Assumption III: *Some violations might be ignored / not observed.*

The mechanism does not impose that an agent must give its testimony whenever it notices a norm violation. This behavior is application dependent and thus should be motivated by the application. In addition, the mechanism does not guarantee that all

norm violations will be observed by at least one agent. It is also the application that must provide support for the agents to observe the violation of every application norm.

Assumption IV: *Agents can give false testimony.*

In an open system, agents are independently implemented, i.e. the development is done without a centralized control. Thus, the application cannot assume that an agent was properly designed. In this scenario, there is no way to guarantee that all testimonies are related to actual violations. So, the governance mechanism should be able to check and assert the truthfulness of the testimonies.

Assumption V: *The mechanism can have a law-enforcement agent force.*

The mechanism can introduce agents which have the sole purpose of giving testimonies. The testimonies of these agents can always be considered to be truthful and the judgment sub-system can directly state that a norm was violated and a penalty should be assigned.

2.2 The Governance Mechanism Architecture

The governance mechanism architecture defines three sub-systems. The judgment subsystem is responsible for receiving the testimonies and for providing a decision (or verdict) pointing out to the punishment sub-system if an agent really violated a norm. While judging a testimony, the system may use different strategies to judge the violation of the different norms specified by the application. Such strategies might use the agents' reputation afforded by the reputation system to help providing the decision.

The reputation sub-system calculates the reputation of agents and informs the reputations to the judgment sub-system and to other application agents. The reputations are updated based on the decisions provided by the judgment sub-system about a violated norm. Different norms influence the reputation of agents in different ways. Finally, the third sub-system, the punishment sub-system, applies the penalties specified in norms to the agents that are blamed for violating a norm.

The governance mechanism was implemented by using the ASF (Agent Society Framework) framework (Silva, 2004). Such framework provides support for the implementation of agents, organizations and roles. Each one of the three governance subsystems was implemented as a separated organization that interacts with a fourth organization where the application agents are situated. The governance mechanism architecture is illustrated in Figure 1. In this paper we will focus on the judgment subsystem.

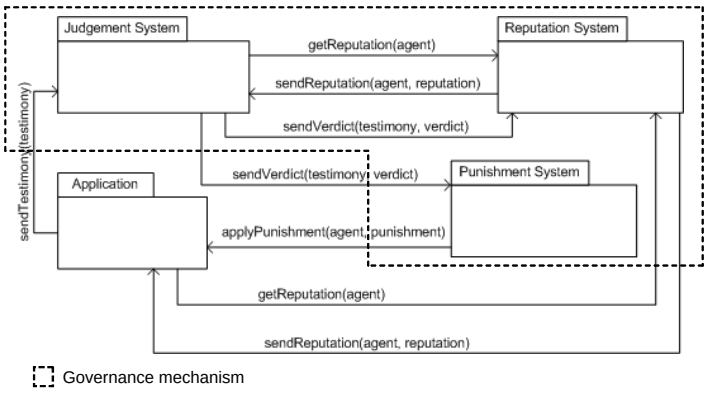

Fig. 1. The architecture of the governance mechanism

3 The Judgment Sub-system

The judgment sub-system has three main responsibilities: to receive testimonies, to judge them and to provide the decision about the violation. Three different agent types were defined to deal with these responsibilities: inspector, judge and mediator agents. The inspector agents are responsible for receiving the testimonies and sending them to judge agents. The judge agents examine the testimonies and provide decisions that are sent to mediator agents. Mediator agents are responsible for inter-acting with the reputation and punishment sub-systems to make the decisions effective.

While judging the testimonies, judge agents may interact with mediators to get information about an agent reputation. Mediator agents get such information from the reputation sub-system. Figure 2 depicts the interactions among the agents that compose the judgment sub-system and among the three sub-systems that orchestrate the governance mechanism.

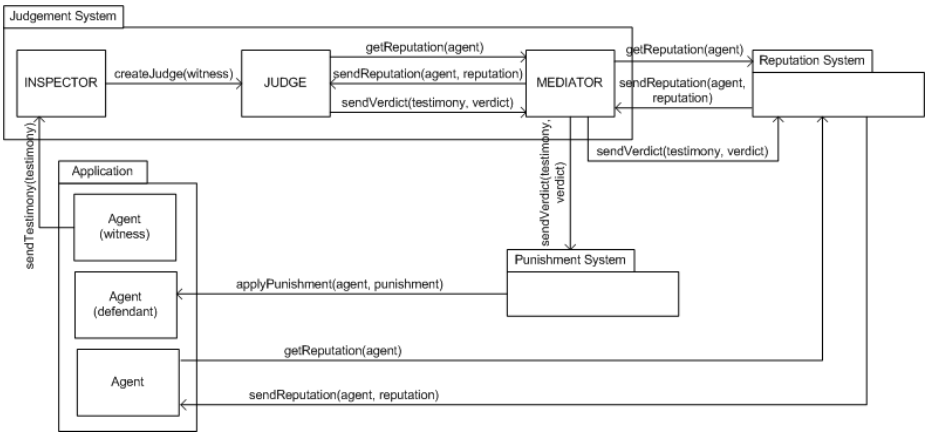

Fig. 2. The interaction among the agents that compose the judgment sub-system

3.1 The Judgment System Architecture

The ASF framework is based on the BDI model and therefore supports the implementation of agents' goals, beliefs and plans. While presenting the judgment sub-system architecture (figure 3), we detail the plans of the agents and the resources they use.

The inspector and mediator agents' implementations are not complex. The inspector agent keeps listening to the testimonies while it executes its only plan called Listening-Testimony. Each testimony provided by application agents states the norm that has been violated, the context that characterizes the violation of the norm, the agent that violated the norm and the agent that is providing the testimony. Note that the context depends on the violated norm. If the norm describes an interaction protocol, the context might be, for instance, a message that was sent in violation of the protocol. If a norm describes access policies to a resource, the context might be a resource update.

Mediator agents execute two plans to accomplish their tasks. Mediators provide the reputation of application agents by executing the plan called ProvideReputation and they also distribute the decisions to the other two sub-systems by executing the plan called ProvideVerdict. Each decision informs the testimony that was judged and the decision stipulated by judge agent.

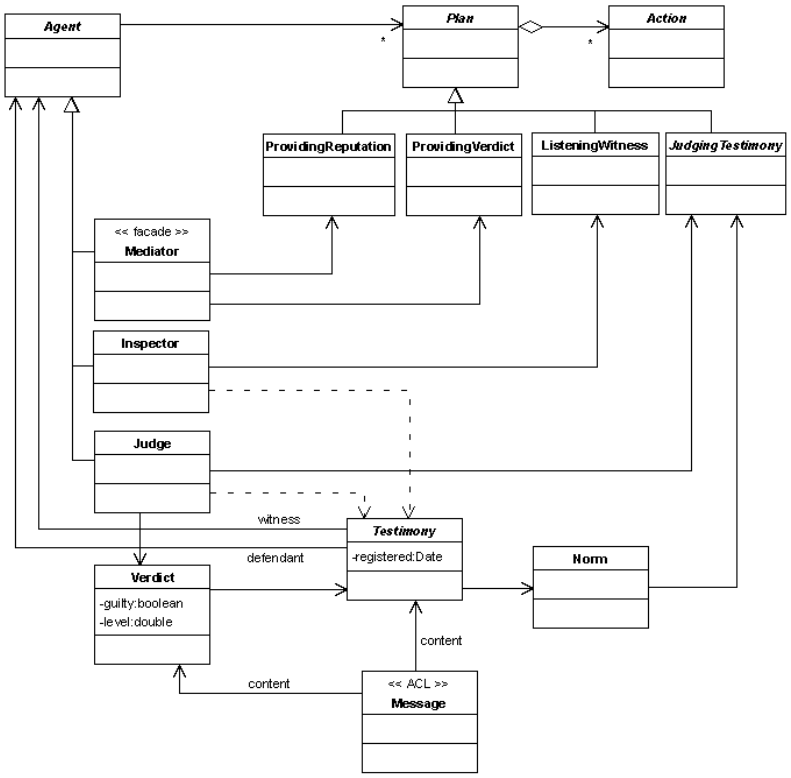

Fig. 3. The judgment sub-system architecture

Judge agents are the most complex agents in the judgment sub-system. The strategies they use to stipulate a decision must deal with truthful and untruthful testimonies. In addition, such strategies must vary according to the norms that have been violated. Depending on the norm, it may be necessary to verify the content of a resource or to ask other agents about another fact that has happened while judging the fact stated in a testimony. Therefore, the judgment sub-system must be flexible enough to be customized for different application norms. It is not possible to define a completely application independently strategy. Application dependent strategies must be implemented by application developers in the plans that extend the *JudgingTestimony* plan. These plans are executed based on the norms that have been violated and are identified in the testimonies being judged. Each application norm must inform the specific plan that must be executed in case it is violated.

Figure 3 shows the judgment sub-system architecture. It pictures the agents, the plans they execute and the content of the messages they send and receive. Note that the judgment sub-system is a framework that must be extended according to the application being governed. This framework defines two extension points: the judgment strategy implemented by the *JudgingTestimony* plan and the *Testimony* itself. As stated before, the *JudgingTestimony* plan must be extended to implement the strategies that are used to judge each application norm and the *Testimony* class must be extended to describe the possible different contexts of norm violations.

3.2 The Judgment Process

The judgment process is composed of five steps where four are application independent ones. Although judgment strategies cannot be completely independent of the application norms, it is possible to define some common steps to be followed by any judgment strategy. The application independent steps, provided by the judgment sys-

tem, are implemented by the JudgingTestimony plan and executed before any specific application strategy. In this section we present the five steps that compose the judgment process and a pseudo-algorithm that describes such process (figure 4).

Step I: *To verify who the witness is.*

According to assumption V, the testimony provided by some specific agents must be considered always truth. Therefore, the first step of the judgment process verifies who the witness is. If it is the case of an always truthful witness, the judgment process is finished and the decision stating that the agent must be penalized is provided.

Step II: *To check if the norm applies to the agent.*

According to assumption IV, agents can lie and end up accusing other agents of doing something they have not. For instance, agents can be accused of violating norms that are not applied to them and agents can be accused of updating a resource that has not been updated. In order to find out if a testimony is true, the first step is to check if the norm applies to the accused agent, i.e., if the norm is one of the norms that must be fulfilled by the agent. If the norm does not apply, the judgment process is finished and the accused agent is absolved.

Step III: *To ask the agent if it is guilty.*

If the norm applies to the agent, the next step is to ask it if it has violated the norm it is accused of. As it happens in the real world, if the agent confesses, the judgment process is finished and the decision is provided. Otherwise, the judgment process continues.

Step IV: *(application dependent step) To judge the testimony according to the norm.*

If the agent did not confess, it is necessary to carefully exam if the agent really violated the norm. Therefore, it is necessary to execute the strategy related to the norm being violated in order to providing a decision. As stated before, these strategies are application dependent ones.

Step V: *To provide a decision.*

After producing the decision, it is necessary to send it to the reputation sub-system so that it can modify the reputation of the accused agent, if it really violated the norm, or the reputation of the testimony agent, if the judgment process did not assign blame. It is also important to inform the decision to the punishment sub-system to punish the agent for violating a norm or to punish the testimony agent for providing an untruth testimony.

```
/* step 1 - To verify who the witness is */
if testimony is a trustful one then
  create a decision stating the accused agent is 100% guilty
  go to step 5

/* step 2 - To check if the norm applies to the agent */
if violated norm is not one of the accused norms
    create a decision stating the defendant is 100% not guilty
    go to step 5

/* step 3 - To ask the agent if it is guilty */
send a message to the accused asking if it is guilty of violating the norm
if it answers yes then
  create a decision stating the defendant is 100% guilty
  go to step 5

/* step 4 - To judge the testimony according to the norm (APPLICATION-DEPENDANT)*/
execute the application dependent strategy

/* step 5 - To provide a decision */
send the decision along with the testimony to mediator agents
```

Fig. 4. The pseudo-algorithm that describes the judgment process

In order to validate our approach we present in the next section a case study. Our purpose is to demonstrate how the judgment system can be used to regulate application norms. In this context, two application norms are described together with the strategies used to verify when agents have violated these two norms.

4 A Case Study: Expert Committee

The Expert Committee is an open multi-agent system that provides support for managing paper submission and revision (deLoach , 2002), (Zambonelli, 2001). The system supports different activities: paper submission, reviewer assignment, review submission, notification of acceptance and rejection and so on.

Since the system being implemented is an open system, it was necessary to define a set of norms to regulate the behavior of the diverse agents (chairs, authors and reviewers) that participate in the system. In this paper we will give examples of two different application norms that were defined and regulated by the proposed mechanism. The chairs of conferences and workshops are responsible for defining the papers submission deadline, the reviewers, the revision deadline, among other things. Once the chair has defined the submission deadline the date must not be anticipated. The anticipation of such deadline would hold up the authors that wanted to submit papers.

Norm I: *Chair cannot anticipate submission deadlines.*

After receiving submitted papers, the chair distributes the papers among the reviewers asking them if they can review the papers. The reviewers respond to the chair the selected papers they want to review.

Norm II: *Reviewers must respond to the chair in two days.*

In sections 4.1 and 4.2 we use the judgment sub-system to judge testimonies describing violations of these two norms. In order to do so, the framework that implements the judgment sub-system was extended. Examples of different testimonies are defined and implementations of strategies used to verify if agents have violated these norms are stated. Figure 5 illustrates the classes that extend the judgment sub-system with the aim of judging these two norms.

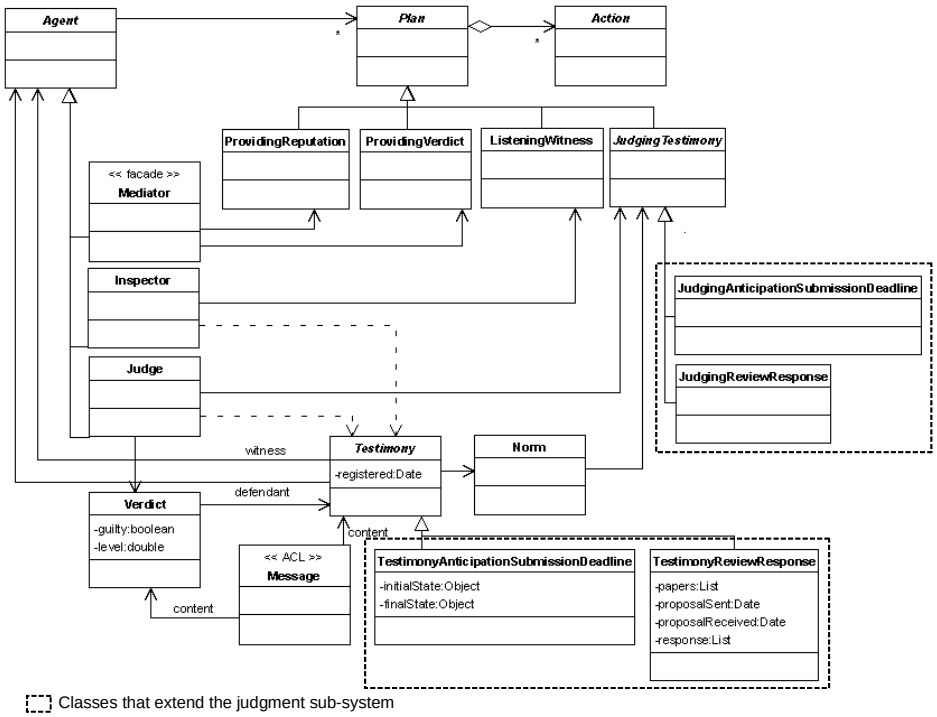

Fig. 5. The judgment sub-system extended to judge two application norms

4.1 Norm I

To bear testimony to the violation of norm I, it is necessary to inform to the judgment sub-system the submission deadline firstly defined by the chair and the actual submission deadline. In order to provide such testimony, the abstract class Testimony defined by the judgment sub-system was extended to include attributes to store these two different deadlines.

The strategy used to judge the violation of norm I is composed of three phases. In phase one, the application resource that stores the conference deadline is analyzed according to the information provided in the testimony. The submission deadline provided by the resource is confronted with the submission deadline provided by the testimony.

If the submission deadline provided by the resource is equal to the first submission deadline available in the testimony, the submission deadline was not changed and the testimony is discarded. If the submission deadline provided by the resource is different to the actual submission deadline provided by the testimony, the testimony is also discarded. The testimony describes a fact that cannot be confirmed. In both cases the witness is providing a false testimony. Nevertheless, if the submission deadline provided by the resource is equal to the actual submission deadline provided by the testimony, the testimony is judged.

In the second phase of the judgment strategy, the information provided by the witness is confronted with the information provided by other agents. Since the application does not have logs to inform when resources are updated, it is necessary to ask other application agents about the fact that may have happened. In this case, three reviewers are asked about the submission deadline. In the third phase, the decision is established based on the information provided in the testimony and the information provided by

the reviewers. The reputation of all agents involved – witness, accused, and reviewers – is also considered to help providing the decision.

Figure 6 presents a pseudo-algorithm of the strategy JudgingAnticipationSubmissionDeadline that extends the plan JudgingTestimony class. Remember that such strategy correspond to the application dependent step (step 4) defined in the judgment process and presented in Section 3.2.

```
/* PHASE I */
/* to check if the testimony's submission deadline is consistent with the event submission deadline
*/
get the resource that stores the conference deadline
if submission deadline provided by the resource == fist submission deadline
provided by the testimony then
  decide that the accused is 100% innocent  /* false testimony */
  go to step 4
if submission deadline provided by the resource != actual submission deadline provided by the testi-
mony then
  decide that the accused is 100% innocent  /* false testimony */
  go to step 4
if submission deadline provided by the resource == actual submission deadline provided by the testi-
mony then
  go to phase II

/* PHASE II */
/* to ask three Reviewers if they know about the event's submission deadline */
select three Reviewers
for each Reviewer
  get Reviewer's statement about the conference event's submission deadline
  get Reviewer's reputation

/* PHASE III */
/* step 1 - to get the chair's and author's reputation */
get chair's reputation
get author's reputation

/* step 2 - to evaluate the statements according to the agents reputation */
/* suppose that three reviewers gave their statements */
```

	Statements	Reputation
Witness	guilty	1
Accused	innocent	5
Reviewer 1	guilty	2
Reviewer 2	guilty	1
Reviewer 3	guilty	3

	∑ Reputation	# Testimony
Innocent	5	1
Guilty	7	4

```
- innocent result: 5 x 1 = 5
- guilty result: 7 x 4 = 28
- final result: (28/33) x 100 = 85% guilt

/* step 3 - to create the decision based on the previous evaluation */
to create the decision

/* step 4 - to return to the judgment process */
return to judgment framework step 5
```

Fig. 6. Strategy I pseudo-algorithm

4.2 Norm II

To bear testimony to the violation of norm II, it is necessary to inform the message sent by the chair to the reviewer with the list of papers to be selected. The abstract class Testimony was extended to provide such information.

The strategy used to judge the violation of norm II is also composed of three phases. We are considering that for each message sent by an agent the receiver must send an acknowledgment to the sender stating that it has received the message. In phase one, the chair must inform if it has received the reviewer acknowledgment stating that it has received the list of papers. If the chair has not such acknowledgment, the testimony is discarded and the reviewer is considered not guilt since it has not received the list of papers.

In phase two, the reviewer must inform if it has the chair acknowledgment stating that it has received the selected papers. The reviewer is asked to present the acknowledgment sent by the chair. If the reviewer has such acknowledge, the testimony is dis-

carded since the chair has informed to the reviewer that it has received the selected papers.

If the chair has the acknowledgment stating that the reviewer has received the list of papers and the reviewer has not the acknowledgement stating that the chair has received the selected papers, the testimony is judge. Two different things may have happened. The reviewer may have sent the selected paper and the chair may have not received or the reviewer may have sent the selected paper and the chair, although it has received it, says that it has not. The decision is provided based on the reputation of the agents. Figure 7 presents a pseudo-algorithm of the strategy above (illustrated in Figure 5 by JudgingReviewerResponse plan).

```
/* PHASE I */
/* to get the acknowledgement stating that the reviewer has received the list of papers */
ask the chair about the acknowledgement
receive message from the chair
if chair has not acknowledgment
  decide the accused is 100% innocent  /* false testimony */
  go to step 4

/* PHASE II */
/* to get the acknowledgment stating that the chair has received the selected papers */
ask the reviewer about the acknowledgement
receive message from the reviewer
if reviewer has acknowledgment
  decide the accused is 100% innocent  /* false testimony */
  go to step 4

/* PHASE III */
/* step 1 - to get chair's and reviewer's reputations */
get chair's reputation
get reviewer's reputation

/* step 2 - to evaluate the statements according to the agents reputation */
```

	Statements	Reputation
Witness	guilty	5
Accused	innocent	3

	∑ Reputation	# Testimony
Innocent	3	1
Guilty	5	1

```
- innocent result: 3 x 1 = 3
- guilty result: 5 x 1 = 5
- final result: (5/8) x 100 = 62,5% guilty

/* step 3 - to create the verdict based on the previous evaluation */
to create the decision

/* step 4 - to return to the judgment process */
return to judgment framework step 5
```

Fig. 7. Strategy II pseudo-algorithm

Note that both strategies presented in section 4.1 and 4.2 are simple examples of plans that can be used to judge the testimonies related to norms I and II. Other more complex and completely different strategies could have been implemented to judge the same testimonies. Our intention while presenting such strategies was to illustrate how the judgment sub-system could be extended to judge two different norms.

5 Conclusion and Future Work

This paper proposes a governance mechanism for open multi-agent systems. The main advantages of the proposed mechanism are: (i) it interferes neither in the agents' privacy nor in the performance of the application; (ii) it can be used to assert that an agent has violated any kind of norm since the violation of both interaction and resource access norms can be witnessed to the governance mechanism; and (iii) it is based on the idea that it may be difficult or impossible to prevent every agent's action in the system.

To properly work, the mechanism assumes that the application agents are aware of the norms they should follow and that they can provide testimonies. This requires some changes in the way these agents are developed or a special agent architecture

that provides these features beforehand. Although the mechanism has these requirements, this action-witness-consequence approach is more adequate than a prevention approach. Norm violation prevention may not be applicable in open systems where several agents are executing.

Whereas we believe that the advantages of our proposed mechanism are really important, it has some potential drawbacks. It may be difficult to find out if a testimony is true or false and, therefore, to provide a good decision. This problem could be overcome using uncertainty, i.e. the judgment sub-system could decide to assign blame or not within an appropriate error margin, as illustrated by the strategies presented in sections 4.1 and 4.2. Also, violations that go without testimonies will not be punished. This could lead to an undesired system state. One way to overcome this issue would be to require a special development effort, which cannot be guaranteed in an open system. Thus it is important to motivate the agents to give their testimonies, using an agent rewards program, for instance.

In other to improve our approach we are in the way of developing a decentralized governance mechanism. The decentralized mechanism is being developed based on the idea of hierarchy norms and organizations presented in (Ao, 2003). In addition, we are also in the process of developing a sub-system to prevent norm violations. Such subsystem will try to foresee actions that could be norm violations based on the norms that agents have already violated.

References

Ao, X., Minsky, N.: Flexible regulation of distributed coalitions. In LNCS 2808: the Proc. of the European Symposium on Research in Computer Security (ESORICS) (2003).

Boella, G.; van der Torre, L: Regulative and constitutive norms in normative multiagent systems. In Proceeding of 9th International Conference on the Prin-ciples of Knowledge Representation and Reasoning. Whistler, CA (2004).

Cremonini, M., Omicini, A., Zambonelli, F.: Coordination and Access Control in Open Distributed Agent Systems: The TuCSoN Approach. In Proceedings of the 4th International Conference on Coordination Languages and Models, LNCS 1906, Springer-Verlag, London, UK (2000) pp 99-114.

López, F.: Social Powers and Norms: Impact on Agent Behaviour. PhD thesis. University of Southampton. UK (2003)

Minsky, N, Ungureanu, V. Law-Governed Interaction: A Coordination & Con-trol Mechanism for Heterogeneous Distributed Systems. In ACM Transactions on Software Engineering and Methodology (TOSEM), July, vol 9, no 3 (2000) 273-305.

Paes, R. Regulating the interaction between agents in open systems – a law ap-proach. Master's thesis, Pontifcia Univeridade Catolica do Rio de Janeiro, PUC-Rio, Rio de Janeiro, BR (2005).

Silva, V, Lucena, C.: Governance in Multi-Agent Systems Based on Witnesses. In: First Workshop on Software Engineering for Agent Oriented Systems, Bra-zilian Symposium on Software Engineering (SBES2005), Uberlândia, BR (2005).

Silva, V., Cortês, M., Lucena, C.: An Object-Oriented Framework for Imple-menting Agent Societies, MCC32/04. Relatório Técnico, Departamento de In-formática, PUC-Rio. Rio de Janeiro, BR (2004).

Singh, M. An Ontology for Commitments in Multiagent Systems: Toward a Unification of Normative Concepts. Artificial Intelligence and Law v. 7 (1) (1999) 97-113.

deLoach, S. A.: Analysis and Design of Multiagent Systems Using Hybrid Coordination Media. In: Proceedings of the 6th World Multi-Conference on Sys-temic, Cybernetics And Informatics (SCI 2002), Florida. (2002)14-18.

Zambonelli, F., Jennings, N., Wooldridge, M.: Organizational abstractions for the analysis and design of multi-agent systems. In: Ciancarini, P.; Wooldridge, M. (Eds.) Agent-Oriented Software Engineering, LNCS 1957, Springer-Verlag, Berlin, GE (2001) 127-141.