Dynamic Protocols for Open Agent Systems

Alexander Artikis[1 2]
[1]Institute of Informatics & Telecommunications, NCSR "Demokritos", Athens, 15310, Greece
[2]Electrical & Electronic Engineering Department, Imperial College London, SW7 2BT, UK
a.artikis@acm.org

ABSTRACT

Multi-agent systems where the members are developed by parties with competing interests, and where there is no access to a member's internal state, are often classified as 'open'. The specification of protocols for open agent systems of this sort is largely seen as a design-time activity. Moreover, there is no support for run-time specification modification. Due to environmental, social, or other conditions, however, it is often required to revise the specification during the protocol execution. To address this requirement, we present an infrastructure for 'dynamic' protocol specifications, that is, specifications that may be modified at run-time by agents. The infrastructure consists of well-defined procedures for proposing a modification of the 'rules of the game' as well as decision-making over and enactment of proposed modifications. We evaluate proposals for rule modification by modelling dynamic specifications as metric spaces. Furthermore, we constrain the enactment of proposals that do not meet the evaluation criteria. We illustrate our infrastructure by presenting a dynamic specification of a resource-sharing protocol, and an execution of this protocol in which the participating agents modify the protocol specification.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence

General Terms

Design

Keywords

organised adaptation, norm, action language

1. INTRODUCTION

A particular kind of Multi-Agent System (MAS) is one where the member agents are developed by different parties, and where there is no access to an agent's internal state. In this kind of MAS it cannot be assumed that all agents will behave according to the system specification because the agents act on behalf of parties with competing interests, and thus may inadvertently fail to, or even deliberately choose not to, conform to the system specification in order to achieve their individual goals. Two examples of this type of MAS are Virtual Organisations and electronic marketplaces. MAS of this type are often classified as 'open'.

Open MAS can be viewed as instances of *normative systems* [11]. A feature of this type of system is that actuality, what is the case, and ideality, what ought to be the case, do not necessarily coincide. Therefore, it is essential to specify what is permitted, prohibited, and obligatory, and perhaps other more complex normative relations that may exist between the agents. Among these relations, considerable emphasis has been placed on the representation of *institutional power* [12] — a standard feature of any normative system whereby designated agents, when acting in specified roles, are empowered by an institution to create specific relations or states of affairs (such as when an agent is empowered by an institution to award a contract and thereby create a bundle of normative relations between the contracting parties).

Several approaches have been proposed in the literature for the specification of protocols for open MAS. The majority of these approaches offer 'static' specifications, that is, there is no support for run-time specification modification. In some open MAS, however, environmental, social or other conditions may favour, or even require, specifications modifiable during the protocol execution. Consider, for instance, the case of a malfunction of a large number of sensors in a sensor network, or the case of manipulation of a voting procedure due to strategic voting, or when an organisation conducts its business in an inefficient manner. Therefore, we present in this paper an infrastructure for 'dynamic' protocol specifications, that is, specifications that are developed at design-time but may be modified at run-time by agents. The presented infrastructure is an extension of the framework for static specifications of [3], and is motivated by 'dynamic argument systems' [5] — argument systems in which, at any point in the disputation, agents may start a meta level debate, that is, the rules of order become the current point of discussion, with the intention of altering these rules.

Our infrastructure for dynamic specifications allows agents to alter the rules of a protocol P during the protocol execution. P is considered an 'object' protocol; at any point in time during the execution of the object protocol the participants may start a 'meta' protocol in order to decide whether the object protocol rules should be modified. Moreover, the participants of the meta protocol may initiate a meta-meta protocol to decide whether to modify the rules of the meta protocol, or they may initiate a meta-meta-meta protocol to modify the rules of the meta-meta protocol, and so on.

We employ a resource-sharing protocol to illustrate our

Cite as: Dynamic Protocols for Open Agent Systems, Alexander Artikis, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. XXX-XXX.

Table 1: Main Predicates of the Event Calculus.

Predicate	Meaning
happens(Act, T)	Action Act occurs at time T
initially($F = V$)	The value of fluent F is V at time 0
holdsAt($F = V$, T)	The value of fluent F is V at time T
initiates(Act, $F = V$, T)	The occurrence of action Act at time T initiates a period of time for which the value of fluent F is V
terminates(Act, $F = V$, T)	The occurrence of action Act at time T terminates a period of time for which the value of fluent F is V

infrastructure for dynamic specifications: the object protocol concerns resource-sharing while the meta protocols are voting protocols. In other words, at any time during a resource-sharing procedure the agents may vote to change the rules that govern the management of resources. The resource-sharing protocol was chosen for the sake of providing a concrete example. In general, the object protocol may be any protocol for open MAS, such as a protocol for coordination or e-commerce; similarly a meta protocol can be any procedure for decision-making over rule modification (argumentation, negotiation, and so on).

The remainder of this paper is organised as follows. First, we briefly review the Event Calculus, the action language that we employ to formalise protocol specifications. Second, we review a static specification of a resource-sharing protocol. Third, we present a dynamic specification of the resource-sharing protocol and an infrastructure for modifying the protocol specification at run-time. Fourth, we present an execution of the protocol, demonstrating how the agents may alter the protocol specification. Finally, we compare our work to research on dynamic specifications, and discuss further research.

2. THE EVENT CALCULUS

The Event Calculus (EC), introduced by Kowalski and Sergot [14], is a formalism for representing and reasoning about actions or events and their effects in a logic programming framework. In this section we briefly describe the version of the EC that we employ. EC is based on a many-sorted first-order predicate calculus. For the version used here, the underlying time model is linear and it may include real numbers or integers. Where F is a *fluent* (a property that is allowed to have different values at different points in time), the term $F = V$ denotes that fluent F has value V. Boolean fluents are a special case in which the possible values are true and false. Informally, $F = V$ holds at a particular time-point if $F = V$ has been *initiated* by an action at some earlier time-point, and not *terminated* by another action in the meantime.

An *action description* in EC includes axioms that define, among other things, the action occurrences (with the use of happens predicates), the effects of actions (with the use of initiates and terminates predicates), and the values of the fluents (with the use of initially and holdsAt predicates). Table 1 summarises the main EC predicates. Variables (starting with an upper-case letter) are assumed to be universally

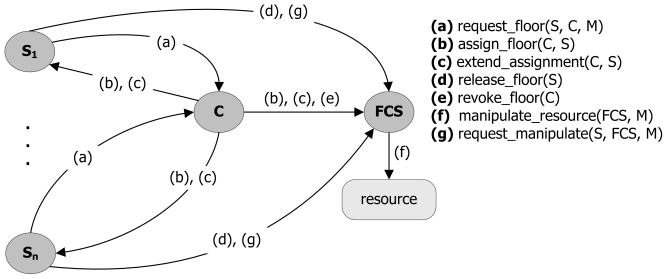

Figure 1: A chaired floor control protocol.

quantified unless otherwise indicated. Predicates, function symbols and constants start with a lower-case letter.

The following sections present a logic programming implementation of an EC action description expressing an infrastructure for a dynamic resource-sharing protocol.

3. A RESOURCE-SHARING PROTOCOL

We present a specification of a resource-sharing or *floor control* protocol in the style of [3]. In the field of Computer-Supported Co-operative Work the term *floor control* denotes a service guaranteeing that at any given moment only a designated set of users (subjects) may simultaneously work on the same objects (shared resources), thus, creating a temporary exclusivity for access on such resources. We present a 'chair-designated' Floor Control Protocol (cFCP), that is, a distinguished participant is the arbiter over the usage of a specific resource. For simplicity we assume a single resource.

The roles of cFCP are summarised below:

- *Floor Control Server* (FCS), the role of the only participant physically manipulating the shared resource.
- *Subject* (S), the role of designated participants requesting the floor from the chair, releasing the floor, and requesting from the FCS to manipulate the resource.
- *Chair* (C), the role of the participant assigning the floor for a particular time period to a subject, extending the time allocated for the floor, and revoking the floor from the subject holding it.

Figure 1 provides an informal description of the possible interactions between the agents occupying the roles of cFCP. More details about these interactions will be given presently.

It has been argued [12] that the specifications of protocols for open MAS should explicitly represent the concept of institutional power, that is, the characteristic feature of organisations/institutions — legal, formal, or informal — whereby designated agents, often when acting in specific roles, are empowered, by the institution, to create or modify facts of special significance in that institution — *institutional facts* — usually by performing a specified kind of act. Searle [17], for example, has distinguished between *brute facts* and institutional facts. Being in physical possession of an object is an example of a brute fact (it can be observed); being the owner of that object is an institutional fact. The cFCP specification explicitly represents the concept of institutional power; moreover, there is a distinction between institutional power, physical capability, permission and obligation. Table 2 presents the conditions in which a cFCP participant has the physical capability, institutional power, permission and obligation to perform an action (to save space Table 2 displays only three protocol actions).

According to the cFCP specification all protocol actions

Table 2: cFCP Specification.

Action	Capability	Power	Permission	Obligation
$request\_floor(S, C, M)$	$\top$	$f\_requested(S) = \mathsf{false}$	$\top$	$\bot$
$assign\_floor(C, S)$	$\top$	$status = free$, $best\_candidate = S$	$status = free$, $best\_candidate = S$	$status = free$, $best\_candidate = S$
$revoke\_floor(C)$	$\top$	$status = granted(S, Tg)$ $CurrentTime \geq Tg$, $best\_candidate \neq S$	$status = granted(S, Tg)$, $CurrentTime \geq Tg$, $best\_candidate \neq S$	$status = granted(S, Tg)$, $CurrentTime \geq Tg$, $best\_candidate = S'$, $S \neq S'$

are physically possible at any time. In other examples the specification of physical capability could have been different.

In this example, a subject S is empowered to request the floor from the chair C when S has no pending requests:

$$\begin{array}{l}\mathsf{holdsAt}(pow(S, request\_floor(S, C, M)) = \mathsf{true},\ T) \leftarrow \\ \quad \mathsf{holdsAt}(role\_of(S) = subject,\ T), \\ \quad \mathsf{holdsAt}(role\_of(C) = chair,\ T), \\ \quad \mathsf{holdsAt}(f\_requested(S) = \mathsf{false},\ T)\end{array} \quad (1)$$

The variable M in *request_floor* represents the requested type of resource manipulation. The *pow* fluent expresses institutional power, the $role\_of(Ag)$ fluent expresses the roles an agent Ag occupies, while the $f\_requested(Ag)$ fluent is true if Ag has pending requests for the floor. To avoid clutter, in Table 2 we do not display the *role_of* fluents and assume that S denotes an agent occupying the role of subject and C denotes an agent occupying the role of chair.

Having specified the institutional power to request the floor, it is now possible to define the effects of this action: a request for the floor is eligible to be serviced if and only if it is issued by an agent with the institutional power to request the floor. Requests for the floor issued by agents without the necessary institutional power are ignored. Due to space limitations, we do not present here the EC axioms expressing the effects of protocol actions (we show only one such axiom below).

We chose to specify that a subject is always permitted to exercise its power to request the floor. Moreover, a subject S is permitted to request the floor even if S is not empowered to do so (see Table 2). In the latter case a request for the floor will be ignored by the chair (since S was not empowered to request the floor) but S will not be *sanctioned* since it was not forbidden to issue the request. In general, an agent is sanctioned when performing a forbidden action or not complying with an obligation. To save space, we do not present a specification of sanctions for cFCP. Finally, a subject is never obliged to request the floor.

The chair's power to assign the floor is defined as follows:

$$\begin{array}{l}\mathsf{holdsAt}(pow(C, assign\_floor(C, S)) = \mathsf{true},\ T) \leftarrow \\ \quad \mathsf{holdsAt}(role\_of(C) = chair,\ T), \\ \quad \mathsf{holdsAt}(status = free,\ T), \\ \quad \mathsf{holdsAt}(best\_candidate = S,\ T)\end{array} \quad (2)$$

The chair C is empowered to assign the floor to S if the floor is free, and S is the best candidate for the floor. The *status* fluent expresses the status of the floor; it can be either *free* or $granted(S, Tg)$, that is, granted to some subject S until time Tg. The *best_candidate* fluent denotes the best candidate for the floor. The definition of this fluent is application-specific. For instance, the best candidate could be the one with the earliest request, that with the most 'urgent' request (however 'urgent' may be defined), and so on. There is no difficulty in expressing such definitions in the formalism employed here. Indeed, the availability of the full power of logic programming is one of the main attractions of employing EC as the temporal formalism.

The effects of assigning the floor are expressed as follows:

$$\begin{array}{l}\mathsf{initiates}(Act,\ status = granted(S, T{+}60),\ T) \leftarrow \\ \quad Act = assign\_floor(C, S), \\ \quad \mathsf{holdsAt}(pow(C, Act) = \mathsf{true},\ T)\end{array} \quad (3)$$

The result of exercising the power to assign the floor to S at time T is that the floor becomes granted to S until T+60. In other versions of the specification (as we shall see later) the floor could be allocated for a longer/shorter period.

The conditions in which the chair is permitted and obliged to assign the floor are the same as the conditions in which the chair is empowered to assign the floor (see Table 2).

Similarly we specify the power, permission and obligation to perform the remaining protocol actions, and the effects of these actions. For instance, the chair's power to revoke the floor is defined as follows:

$$\begin{array}{l}\mathsf{holdsAt}(pow(C, revoke\_floor(C)) = \mathsf{true},\ T) \leftarrow \\ \quad \mathsf{holdsAt}(role\_of(C) = chair,\ T), \\ \quad \mathsf{holdsAt}(status = granted(S, Tg),\ T),\ T \geq Tg, \\ \quad \mathsf{not}\ \mathsf{holdsAt}(best\_candidate = S,\ T)\end{array} \quad (4)$$

The chair C is empowered to revoke the floor if the floor is granted to a subject S until time Tg, the current time is greater or equal to Tg, and S is not the best candidate for the floor. 'not' represents negation by failure.

The specification of the power, permission or obligation to revoke the floor, assign the floor, or perform some other protocol action, should include a deadline stating the time by which the action of revoking the floor, assigning the floor, etc, should be performed. There is no particular difficulty in including deadlines in the formalisation but it lengthens the presentation and is omitted here for simplicity. Example formalisations of deadlines may be found in [3].

4. A DYNAMIC RESOURCE-SHARING PROTOCOL

Being motivated by Brewka [5], we present an infrastructure that allows agents to modify (a subset of) the rules of a protocol at run-time. Regarding our running example, we consider the resource-sharing protocol as an 'object' protocol; at any point in time during the execution of the object protocol the participants may start a 'meta' protocol in order to potentially modify the object protocol rules — for instance, replace an existing rule-set with a new one. The

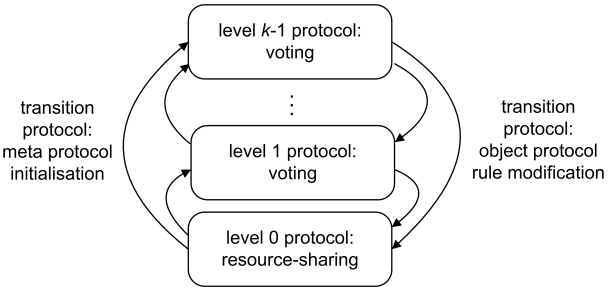

Figure 2: A k-level Infrastructure for Dynamic Specifications.

meta protocol may be any protocol for decision-making over rule modification. For the sake of presenting a concrete example, we chose a voting procedure as a meta protocol, that is, the meta protocol participants take a vote on a proposed modification of the object protocol rules. The participants of the meta protocol may initiate a meta-meta protocol to modify the rules of the meta protocol, or they may initiate a meta-meta-meta protocol to modify the rules of the meta-meta protocol, and so on. For simplicity, in this example *all* meta protocols are voting procedures (in other systems each meta protocol may be a different decision-making procedure). In general, in a k-level infrastructure, level 0 corresponds to the main (resource-sharing, in this example) protocol while a protocol of level n, $0<n\leq k-1$ (voting, in this example), is created, by the protocol participants of a level m, $0\leq m<n$, in order to decide whether to modify the protocol rules of level $n-1$. The infrastructure for dynamic (resource-sharing) specifications is displayed in Figure 2.

Apart from object and meta protocols, the infrastructure for dynamic specifications includes 'transition' protocols — see Figure 2 — that is, procedures that express, among other things, the conditions in which an agent may successfully initiate a meta protocol (for instance, only the members of the board of directors may successfully initiate a meta protocol in some organisations), the roles that each meta protocol participant will occupy, and the ways in which an object protocol is modified as a result of the meta protocol interactions. The components of the infrastructure for dynamic specifications, level 0 protocol, level n protocol ($n>0$), and transition protocol, are discussed in the following sections.

4.1 Level 0

For illustration purposes we chose a resource-sharing protocol — the specification of which was presented in Section 3 — as a level 0 protocol. A protocol specification consists of the core rules that are always part of the specification, and the *Degrees of Freedom* (*DoF*), that is, the specification components that may be modified at run-time. Consider, for instance, the definition of the best candidate for the floor as a DoF: the participants of the resource-sharing protocol may decide, at run-time, to change the way the best candidate is computed. To provide a simple example, consider the following rules:

$$\begin{aligned}&\mathsf{holdsAt}(best\_candidate = S,\ T) \leftarrow\\ &\quad \mathsf{holdsAt}(active(bc) = fcfs,\ T),\\ &\quad \mathsf{holdsAt}(requests = List,\ T),\\ &\quad first(List,\ (S, Tr, M))\end{aligned} \tag{5}$$

$$\begin{aligned}&\mathsf{holdsAt}(best\_candidate = S,\ T) \leftarrow\\ &\quad \mathsf{holdsAt}(active(bc) = random,\ T),\\ &\quad \mathsf{holdsAt}(requests = List,\ T)\\ &\quad random(List,\ (S, Tr, M))\end{aligned} \tag{6}$$

The above two rules provide simple, alternative specifications of what constitutes the best candidate for the floor, that is, they are two possible 'values' of the best candidate DoF. According to rule (5) the best candidate is the subject with the earliest pending request for the floor, while according to rule (6) the best candidate is chosen randomly from the list of subjects that have pending requests. The *requests* fluent provides a list of the pending requests for the floor, each request represented as a triple ($subject, request-time, manipulation-type$), while *first* and *random* are suitably chosen atemporal predicates returning, respectively, the first and a random element of a list of requests (this list is sorted by request time). The $active(DoF)$ fluent denotes the current or 'active' value of a DoF. For instance, $active(bc) = fcfs$ states that the best candidate (bc) is determined on a first-come, first-served ($fcfs$) basis whereas $active(bc) = random$ states that the best candidate is chosen randomly. Rules (5) and (6) are replaceable in the sense that the participants of the resource-sharing protocol may activate/deactive one of them at run-time (that is, change the value of the corresponding *active* fluent) by means of a meta protocol.

The resource-sharing protocol may have other DoF, such as the specification of the permission and the obligation to perform an action. The classification of a specification component as a DoF is application-specific.

4.2 Level n

To provide a concrete example we chose a three-level infrastructure for dynamic specifications. Moreover, both levels 1 and 2 are voting procedures, such as that presented in [16]. Due to space limitations we do not present here a specification of a voting procedure. Briefly, we assume a simple procedure including a set of voters casting their votes, 'for' or 'against' a particular motion, which would be in this example a proposed modification of the rules of level $n-1$, and a chair counting the votes and declaring the motion carried or not carried, based on the *standing rules* of the voting procedure — simple majority, two-thirds majority, etc.

Our infrastructure allows for the modification of the rules of all protocol levels apart from the top one. Consequently, we define DoF for all protocol levels apart from the top one. For the protocol of level 1 — voting — we chose to set as a DoF the standing rules of the voting procedure. In other words, a level 2 protocol may be initiated in order to decide whether level 1 voting should become, say, simple majority instead of two-thirds majority. Note that the voting procedures of levels 1 and 2 may not always have the same set of rules. For example, at a particular time-point level 2 voting may require a simple majority whereas level 1 voting may require a two-thirds majority (as mentioned above, the standing rules of level 1 constitute a DoF and thus the specification of this part of the protocol may change at run-time).

Each protocol level, including level 0, has its own state; for instance, at a particular time-point agent Ag_1 may have the institutional power to vote in level 1 but have no powers in levels 0 and 2, Ag_2 may occupy the role of subject in level 0 and the role of voter in level 1 (role-assignment in a meta protocol will be discussed presently), etc. In order to distinguish between the states of different protocol levels, we

add a parameter in the representation of actions and fluents of all protocol specifications, expressing the protocol level PL, as follows: $role\_of(Ag, PL)$ expresses the set of roles Ag occupies in level PL, $active(DoF, PL)$ expresses the value of DoF in level PL, etc.

4.3 Transition Protocol

In order to modify the protocol rules of level m, $m \geq 0$ (for example, to change the value of the best candidate DoF from *fcfs* to *random*), that is, in order to start a protocol of level $m+1$, the participants of level m need to follow a 'transition' protocol — see Figure 2. The infrastructure for dynamic specifications presented in this paper requires two types of transition protocol: one leading from the resource-sharing protocol to a voting one (level 0 to level 1 or 2), and one leading from one voting protocol to the other (level 1 to level 2). We will only present the first type of transition protocol; the latter type of protocol may be specified in a similar manner. An example transition protocol leading from resource-sharing to voting is the following: a subject of the resource-sharing protocol proposes a modification of the rules of this protocol (or of the rules of level 1). If the subject is empowered to make such a proposal, then the modification is directly accepted, without the execution of a meta protocol, provided that another subject seconds the proposal, and no other subject objects to that proposal. If the proposal is not seconded then it is ignored. If the proposal is seconded and there is an objection then an argumentation procedure commences, the topic of which is the proposed rule modification, the proponent of the topic is the subject that made the proposal, and the opponent is the subject that objected to the proposal. The argumentation procedure is followed by a meta protocol (level 1 or 2) in which a vote is taken on the proposed rule modification.

In this example transition protocol we have specified the power to propose a rule-set replacement as follows:

$$\begin{aligned} &\mathsf{holdsAt}(pow(Ag, propose(Ag, P)) = \mathsf{true},\ T) \leftarrow \\ &\quad P = replace(DoF,\ OldVal,\ NewVal,\ PL), \\ &\quad \mathsf{holdsAt}(role\_of(Ag, 0) = subject,\ T), \\ &\quad \mathsf{holdsAt}(protocol(PL{+}1) = idle,\ T), \\ &\quad \mathsf{holdsAt}(active(DoF, PL) = OldVal,\ T) \end{aligned} \tag{7}$$

An agent Ag is empowered to propose to change the value of DoF in protocol level PL from $OldVal$ to $NewVal$ if:

- Ag occupies the role of subject in level 0. In this example, the chair is not empowered to propose a modification of the protocol rules.
- There is no protocol taking place in level $PL{+}1$. A protocol for modifying the rules of level PL, that is, a protocol of level $PL{+}1$, may commence only if there is no other protocol of level $PL{+}1$ taking place.
- The value of DoF in level PL is $OldVal$.

A proposal for rule modification needs to be seconded by a subject having the institutional power to second the proposal, in order to be directly enacted, or initiate an argumentation procedure, that is followed by a voting procedure. We chose to specify the power to second a proposal for rule modification as follows:

$$\begin{aligned} &\mathsf{holdsAt}(pow(Ag, second(Ag, P)) = \mathsf{true},\ T) \leftarrow \\ &\quad \mathsf{holdsAt}(role\_of(Ag, 0) = subject,\ T), \\ &\quad \mathsf{holdsAt}(proposal(Ag', P) = \mathsf{true},\ T),\ Ag \neq Ag' \end{aligned} \tag{8}$$

Ag is empowered to second any proposal for rule modification P made by some other subject. The *proposal* fluent records proposals made by empowered agents.

In other examples the specification of the power to propose a rule modification, or second such a proposal, could have different, or additional conditions further constraining how a protocol specification may change at run-time. For instance, in some systems an agent would not have the power to propose a rule modification, or second a proposal, that would create some type of protocol inconsistency. In Section 4.4 we present a way of constraining the process of run-time specification modification. Other ways of achieving that, such as the one described above, could have been formalised (see Section 6).

We have specified that any subject is empowered to object to a proposal for rule modification. Exercising the power to object to a proposal for rule modification initiates an argumentation procedure, the topic of which is the proposed modification. To save space we do present here a specification of an argumentation procedure; see [2] for an example formalisation of such a procedure.

The completion of the argumentation taking place in the context of a transition protocol initiates a meta protocol. The latter protocol is a voting procedure concerning a proposed rule modification. The agents participating in this procedure and the roles they occupy are determined by the transition protocol that results in the voting procedure. We chose to specify that the chair of the resource-sharing protocol becomes the chair of the voting procedure. Furthermore, the agents occupying the role of voter, thus having the power to vote, are the subjects that have not been sanctioned for exhibiting 'anti-social' behaviour, that is, performing forbidden actions or not complying with obligations:

$$\begin{aligned} &\mathsf{holdsAt}(role\_of(Ag, PL) = voter,\ T) \leftarrow \\ &\quad \mathsf{holdsAt}(role\_of(Ag, 0) = subject,\ T), \\ &\quad \mathsf{holdsAt}(protocol(PL) = executing,\ T),\ PL > 0, \\ &\quad \mathsf{holdsAt}(sanctions(Ag) = \mathsf{false},\ T) \end{aligned} \tag{9}$$

The value of $protocol(PL)$ becomes *executing*, in the case where $PL > 0$, at the end of the argumentation of the transition protocol that leads to level PL. We chose not to relativise the fluent recording sanctions to a protocol level. Therefore, the *sanctions* fluent records 'anti-social' behaviour exhibited at any protocol level. Depending on the employed treatment of sanctions, 'anti-social' behaviour may be permanently recorded, thus permanently depriving a subject of participating in a meta level, or it may be temporarily recorded, thus enabling subjects to participate in a meta level after a specified period has elapsed from the performance of forbidden actions or non-compliance with obligations.

The fact that an agent may successfully start a protocol of level $m{+}1$ by proposing a modification of the protocol rules of level m, does not necessarily imply that the rules of level m will be modified. It is only if the motion of level $m{+}1$, that is, the proposed rule modification in level m, is carried that the rules of level m will be modified. Consider the following rule expressing the outcome of a voting procedure of level $m{+}1$ that takes place in order to change the value of DoF in level m from $OldVal$ to $NewVal$:

$$\begin{aligned} &\mathsf{initiates}(Act,\ active(DoF, PL) = NewVal,\ T) \leftarrow \\ &\quad Act = declare(C,\ Motion,\ carried,\ PL{+}1), \\ &\quad Motion = replace(DoF,\ OldVal,\ NewVal,\ PL), \\ &\quad \mathsf{holdsAt}(pow(C, Act) = \mathsf{true},\ T) \end{aligned} \tag{10}$$

Exercising the power to declare the motion carried in level $PL{+}1$ results in setting the value of DoF in level PL to $NewVal$, if the motion was the replacement of $OldVal$ with $NewVal$. If the chair of the voting procedure did not declare the motion carried, or was not empowered to make the declaration, then the value of DoF would not have changed in level PL. To save space we do not present here the specification of the power to make a declaration.

4.4 Constraining the Process of Run-time Specification Modification

As already mentioned, all protocol levels apart from the top one have DoF and, therefore, their specification may be modified at run-time. In this section we present a way of evaluating a proposal for protocol modification. Moreover, we present a way of constraining the enactment of proposals that do not meet the evaluation criteria.

A protocol specification with l DoF creates an l-dimensional specification space where each dimension corresponds to a DoF. A point in the l-dimensional specification space, or *specification point*, represents a complete protocol specification, and is denoted by an l-tuple where each element of the tuple expresses a value of a DoF. Consider, for example, the resource sharing protocol with three DoF: the specification of the best candidate, the power to request the floor, and the floor allocation time. The specification of these three protocol features may change at run-time — for instance, the best candidate may be determined randomly, on a first-come, first-served basis, priority may be given to subjects requesting a particular manipulation type, or to subjects that have not been sanctioned. The specification of the power to request the floor may dictate that a subject is empowered to request the floor if it has no pending requests, or it may be that a subject is always empowered to request the floor. Finally, in this example, the floor may be allocated for 60, 120 or 180 time-points. In the resource-sharing example with these three DoF, a specification point s is, for instance:

$$s = (fcfs, anytime, 120)$$

According to the above specification point, the best candidate is determined on first-come, first-served basis ($fcfs$), a subject is empowered to request the floor at any time ($anytime$), and the floor is allocated for 120 time-points.

We evaluate a proposal for protocol modification by modelling a dynamic protocol specification as a *metric space* [6]. More precisely, given the set of specification points of a protocol, we define a 'desired' specification point, and compute the 'distance' between the desired point and the specification point that would be reached if the proposal for protocol modification was accepted ('resulting' point). We constrain the process of run-time protocol modification by forbidding agents to propose a modification in a way that the resulting specification point is 'far' from the desired point. In what follows we describe when a specification point is considered 'desired', the way we compute the distance between two specification points, and the way we forbid agents to propose protocol modifications.

A desired specification point is a tuple of the desired DoF values. A desired specification point for the resource-sharing protocol could be one where the best candidate is determined on a first-come, first-served basis, the power to request the floor depends on whether a subject has pending requests, and the floor allocation time is 60 time-points.

We follow two steps to compute the distance between two specification points s and s', each represented as an l-tuple, where each element of the tuple expresses a DoF value. First, we transform s and s' to l-tuples of non-negative integers qs and qs' respectively. To achieve that we define an application-specific function v, that 'ranks' each DoF value, that is, associates every DoF value with a non-negative integer. The i-th element of qs, qs_i, has the value of $v(s_i)$, where s_i is the i-th element of s (respectively, the i-th element of qs', qs'_i, has the value of of $v(s'_i)$, where s'_i is the i-th element of s'). Second, we employ a *metric* (or *distance function*), such as the Manhattan metric, to compute the distance between qs and qs' (the choice of a metric is application-specific — see [6] for a list of metrics). Depending on the employed metric, we may add weights on the DoF — for instance, we may require that the computation of the distance between qs and qs' is primarily based on the best candidate DoF rather than the other two DoF. The distance between s and s' is the distance between qs and qs'.

We constrain run-time protocol modification as follows:

$$\begin{aligned}&\mathsf{holdsAt}(per(Ag, propose(Ag, P)) = \mathsf{true},\ T) \leftarrow\\ &\quad \mathsf{holdsAt}(pow(Ag, propose(Ag, P)) = \mathsf{true},\ T),\\ &\quad P = replace(DoF,\ OldVal,\ NewVal,\ PL),\\ &\quad \mathsf{holdsAt}(dofValuesExcept(DoF, PL) = List,\ T),\\ &\quad \mathsf{holdsAt}(distance(List \cup [(DoF, NewVal)], PL) = Dist,\ T),\\ &\quad \mathsf{holdsAt}(threshold(PL) = TDist,\ T),\\ &\quad Dist < TDist\end{aligned} \tag{11}$$

The $per(Ag, Act)$ fluent expresses whether Ag is permitted to perform Act. The $dofValuesExcept(DoF, PL)$ fluent returns a list of the degrees of freedom of level PL, except DoF, and their current values. The *distance* fluent calculates the distance between a given specification point (the first argument of the fluent) and the desired specification point in level PL. *distance* includes a logic programming implementation of a chosen metric. The *threshold* fluent returns the maximum distance that a specification point should have from the desired one in level PL. Different protocol levels may have different *threshold* values, different metrics and different desired specification points. According to axiom (11), an agent is permitted to propose a rule modification if it is empowered to propose a rule modification, and the distance between the resulting specification point and the desired one is less than a specified threshold. In this example, an agent is forbidden to propose a rule modification if these conditions are not satisfied.

Similarly we may express the permission to second a proposal for rule modification or the obligation to object to such a proposal. To further constrain the process of run-time specification modification, when the current specification point is too 'far' from the desired one (say twice the distance given by *threshold*), we could impose the obligation to propose a rule modification in a way that the resulting specification point is 'close' to the desired one. Due to space limitations we do not present here such an obligation.

What constitutes a 'desired' specification point often depends on environmental, social or other conditions. Consequently, we need to allow for the possibility that a desired specification point may change during a protocol execution. Similarly, we need to allow for the possibility that the metric with which the distance between two specification points is computed, and the threshold distance, may change at run-time. Changing the desired specification point, metric and

Table 3: Narrative of Events.

Time	Action
0	*request_floor*(s_1, *c*, *cpu*, *0*)
5	*request_floor*(s_2, *c*, *cpu*, *0*)
6	*request_floor*(s_3, *c*, *cpu*, *0*)
8	*request_floor*(s_5, *c*, *cpu*, *0*)
14	*propose*(s_3, *replace*(*sr*, *3/4m*, *sm*, *1*))
16	*object*(s_1, *replace*(*sr*, *3/4m*, *sm*, *1*))
17	*second*(s_5, *replace*(*sr*, *3/4m*, *sm*, *1*))
	<transition protocol argumentation>
28	*vote*([s_2, s_3, s_4, s_5, s_6], *for*, *2*)
30	*vote*(s_1, *against*, *2*)
31	*declare*(*c*, *carried*, *2*)
35	*propose*(s_5, *replace*(*bc*, *fcfs*, *random*, *0*))
36	*second*(s_3, *replace*(*bc*, *fcfs*, *random*, *0*))
39	*object*(s_2, *replace*(*bc*, *fcfs*, *random*, *0*))
	<transition protocol argumentation>
51	*vote*([s_3, s_4, s_6], *for*, *1*)
53	*vote*([s_1, s_2], *against*, *1*)
54	*declare*(*c*, *carried*, *1*)
58	*assign_floor*(*c*, s_3, *0*)

threshold distance, may be achieved in a manner similar to run-time rule modification.

5. ANIMATION

We illustrate our infrastructure for dynamic specifications by animating an example run of the dynamic resource-sharing protocol. A part of the narrative of events of this run is displayed in Table 3. The events of transition protocols (*propose*, *second*, *object*), level 1 and level 2 protocols (*vote* and *declare*) are indented. The last argument of a level 0, 1 and 2 event indicates the protocol level in which the event took place. Recall that the resource-sharing protocol (level 0) includes three DoF (see Section 4.4). Initially, the best candidate is determined on a first-come, first-served basis (see rule (5)), the power to request the floor depends on whether a subject has pending requests, and the floor allocation time is 60 time-points. Level 1 voting has a single DoF: the standing rules (*sr*). Initially, a 75% majority (*3/4m*) is required. In this example run, the initial specification points of levels 0 and 1 happen to coincide with the initial desired specification points of these levels. Level 2 voting does not have a DoF; moreover, a 75% majority is required. The Euclidean metric is used in levels 0 and 1.

Given that our infrastructure is expressed as a logic program, we may query our implementation to determine the state current at each time and protocol level. We may calculate, for instance, which roles each agent occupies, what powers, permissions, obligations each agent has, etc.

The present example includes 7 agents, a chair *c* and 6 subjects s_1–s_6. In the beginning of the run-time activities s_1, s_2, s_3 and s_5 exercise their power to request the floor, all of them requiring CPU cycles (see third argument of *request_floor*). s_3 and s_5 aim to change the best candidate DoF from *fcfs* to *random* because in this way they may acquire access to the resource faster. Before attempting to change the value of the best candidate DoF, s_3 and s_5 attempt to change the standing rules of level 1 voting from 75% majority to simple majority (*sm*), so that less votes would be required in level 1 when they propose to change the value of best candidate DoF. The proposal for changing the standing rules of level 1 takes place at time-point 14. At that time s_3 is empowered to make the proposal (see rule (7)), and permitted to exercise its power (see rule (11)) because the distance, in level 1, between the resulting specification point and the desired one is less than the specified threshold. Effectively the specified threshold distance allows agents to change level 1 standing rules to simple majority. s_3's proposal is followed by an objection, a secondment, and an argumentation. Then level 2 voting commences; the motion is the proposal for modifying the standing rules of level 1. s_2–s_6 vote for the motion while s_1 votes against it. At time-point 31 the motion of level 2 is declared carried (recall that level 2 requires 75% majority) and thus the standing rules of level 1 change to simple majority (see rule (10)). To avoid clutter in Table 3 we omit the motion in *declare*.

s_5 proposes at time-point 35 to change value of the best candidate DoF to *random*. s_5 is empowered to make the proposal at that time. However, s_5 is not permitted to exercise its power because the distance, in level 0, between the resulting specification point and the desired one is greater than the specified threshold. Consequently, s_5 is sanctioned for performing a forbidden action and thus cannot participate in level 1 to vote (see rule (9)). s_3, s_4 and s_6 vote for the motion of level 1 — the proposal for changing the best candidate DoF of level 0 — while s_1 and s_2 vote against it. At time 54 the motion of level 1 is declared carried (recall that level 1 now requires a simple majority) and thus the best candidate in level 0 is chosen randomly from the list of subjects having pending requests (see rule (6)). At time 58 *c* exercises its power (see rule (2)) to assign the floor to s_3.

6. DISCUSSION

We presented a significant extension of [3, 1]: we developed an infrastructure for dynamic protocol specifications for open MAS, that is, specifications that are developed at design-time but may be modified at run-time by agents. Moreover, we modelled dynamic specifications as metric spaces in order to evaluate proposals for specification modification and constrain the enactment of proposals that do not meet the evaluation criteria. Any protocol for open MAS may be in level 0 of our infrastructure whereas any protocol for decision-making over specification modification may be in level n, n>0. The transition protocols linking the different protocol levels can be as complex/simple as required by the application in question.

The use of metric spaces for modelling dynamic specifications was proposed in [13]. In this line of work a protocol is evaluated by computing the distance between the specification point current at each time and the desired point. Our works differs in two ways from [13]. First, we developed an infrastructure (meta protocols, transition protocols) allowing for run-time specification point change. Second, we constrained the process of run-time specification point change by requiring that the specification point current at each time is 'close' to the desired one.

Bou and colleagues [4] have presented a mechanism for the run-time modification of the norms of an 'electronic institution'. These researchers have proposed a 'normative transition function' that maps a set of norms (and goals) into a new set of norms. The 'institutional agents', representing the institution, are observing the members' interactions in order to learn the normative transition function, so that they (the institutional agents) will directly enact the norms en-

abling the achievement of the 'institutional goals' in a given scenario. Unlike Bou and colleagues, we do not necessarily rely on designated agents to modify norms. We presented an infrastructure with which any agent may (attempt to) adapt a protocol specification. This does not exclude the possibility, however, that, in some applications, specific agents are given the institutional power to directly modify a protocol specification.

Chopra and Singh [7] have presented a way of adapting protocols according to context, or the preferences of agents in a given context. They formalise protocols and 'transformers', that is, additions/enhancements to an existing protocol specification that handle some aspect of context or preference. Depending on the context or preference, a protocol specification is complemented, at *design-time*, by the appropriate transformer thus resulting in a new specification. Unlike Chopra and Singh, we are concerned here with the *run-time* adaptation of a protocol specification and, therefore, we developed an infrastructure to achieve that.

Chopra and Singh, and Bou and colleagues, express protocols in terms of 'commitments' (here the term 'commitment' refers to a form of directed obligation). It is difficult to see how an interaction protocol for open MAS can be specified simply in terms of commitments. At the very least, a specification of institutional power is also required.

The Organisational Model for Adaptive Computational Systems (OMACS) [8] is another approach for dynamic MAS. OMACS concentrates mainly on re-organisation from the perspective of a functional assignment — the assignment of goals and roles to agents — whereas we emphasise, like Bou and colleagues, a different perspective, aimed at achieving a mapping from norms to norms, that is, the rules which regulate, among other things, the process of performing such a functional assignment. In general, we have been concerned with a particular aspect of 'adaptation': the run-time modification of the 'rules of the game' of normative systems. Clearly there are other aspects of adaptive/dynamic systems such as the run-time alteration of the (trading and other) relationships between agents, the members of a system, the assignment of roles to agents (as done in OMACS, for example), and the goals of a system. [15, 10, 19] and the papers in [9] are but a few examples of studies of adaptive systems.

Our approach for evaluating proposals for specification modification, that is, modelling dynamic specifications as metric spaces, is computationally efficient — computing the distance between two specification points may be performed in real-time even when there is a large number of DoF — and flexible — agents may change at run-time the desired specification point or the metric. There are several applications, however, where additional constraints are required on the process of run-time specification modification. For example, it may be required that a protocol specification always satisfies a set of properties ('norm consistency', for instance). In this case agents may not be empowered, or permitted, to propose a modification resulting in a specification that does not satisfy the desirable properties. We are currently investigating formalisms (such as that presented in [18]), supported by software implementations allowing for proving properties and assimilating narratives of events in an efficient manner.

7. REFERENCES

[1] A. Artikis, D. Kaponis, and J. Pitt. Dynamic specifications of norm-governed systems. In V. Dignum, editor, *Multi-Agent Systems: Semantics and Dynamics of Organizational Models*. IGI, 2009.

[2] A. Artikis, M. Sergot, and J. Pitt. An executable specification of a formal argumentation protocol. *Artificial Intelligence*, 171(10–15):776–804, 2007.

[3] A. Artikis, M. Sergot, and J. Pitt. Specifying norm-governed computational societies. *ACM Transactions on Computational Logic*, 10(1), 2009.

[4] E. Bou, M. López-Sánchez, and J. Rodriguez-Aguilar. Using case-based reasoning in autonomic electronic institutions. In *Proceedings of COIN Workshop*, LNCS 4870, pages 125–138. Springer, 2008.

[5] G. Brewka. Dynamic argument systems: a formal model of argumentation processes based on situation calculus. *Journal of Logic and Computation*, 11(2):257–282, 2001.

[6] V. Bryant. *Metric Spaces*. Cambridge University Press, 1985.

[7] A. Chopra and M. Singh. Contextualizing commitment protocols. In *Proceedings of AAMAS*, pages 1345–1352. ACM, 2006.

[8] S. DeLoach, W. Oyenan, and E. Matson. A capabilities-based model for adaptive organizations. *Autonomous Agents and Multi-Agent Systems*, 16(1):13–56, 2008.

[9] V. Dignum, editor. *Multi-Agent Systems: Semantics and Dynamics of Organizational Models*. IGI, 2009.

[10] M. Hoogendoorn, C. Jonker, M. Schut, and J. Treur. Modeling centralized organization of organizational change. *Computational & Mathematical Organization Theory*, 13(2):147–184, 2007.

[11] A. Jones and M. Sergot. On the characterisation of law and computer systems: the normative systems perspective. In *Deontic Logic in Computer Science*, pages 275–307. J. Wiley and Sons, 1993.

[12] A. Jones and M. Sergot. A formal characterisation of institutionalised power. *Journal of the IGPL*, 4(3):429–445, 1996.

[13] D. Kaponis and J. Pitt. Dynamic specifications in norm-governed open computational societies. In *Proceedings of ESAW Workshop*, LNCS 4457, pages 265–283. Springer, 2007.

[14] R. Kowalski and M. Sergot. A logic-based calculus of events. *New Generation Computing*, 4(1):67–96, 1986.

[15] C. Martin and K. S. Barber. Adaptive decision-making frameworks for dynamic multi-agent organizational change. *Autonomous Agents and Multi-Agent Systems*, 13(3):391–428, 2006.

[16] J. Pitt, L. Kamara, M. Sergot, and A. Artikis. Voting in multi-agent systems. *Computer Journal*, 49(2):156–170, 2006.

[17] J. Searle. What is a speech act? In A. Martinich, editor, *Philosophy of Language*, pages 130–140. Oxford University Press, third edition, 1996.

[18] M. Sergot. Action and agency in norm-governed multi-agent systems. In *Proceedings of ESAW VIII*, LNAI 4995, pages 1–54. Springer, 2008.

[19] Y. Shoham and M. Tennenholtz. On the emergence of social conventions: modeling, analysis and simulations. *Artificial Intelligence*, 94(1-2):139–166, 1997.