Organization Models and Behavioural Requirements Specification for Multi-Agent Systems

Conference Paper · June 2001

CITATIONS
28

READS
85

5 authors, including:

Jacques Ferber
Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LI…
133 PUBLICATIONS **8,186** CITATIONS
SEE PROFILE

Olivier Gutknecht
Apple Inc.
22 PUBLICATIONS **2,126** CITATIONS
SEE PROFILE

Catholijn M. Jonker
Delft University of Technology
543 PUBLICATIONS **6,390** CITATIONS
SEE PROFILE

Jean-Pierre Müller
Cirad - La recherche agronomique pour le développement
92 PUBLICATIONS **925** CITATIONS
SEE PROFILE

Organization Models and Behavioural Requirements Specification for Multi-Agent Systems

Jacques Ferber[1], Olivier Gutknecht[1], Catholijn M. Jonker[2], Jean-Pierre Müller[3], Jan Treur[2]

[1] Universite Montpellier II, Laboratoire d'Informatique, Robotique et Micro-Electronique
161, rue ADA, 34392 Montpellier Cedex 5, France
Email: {ferber, gutkneco}@lirmm.fr
URL: http://www.lirmm.fr/{~ferber,~gutkneco}

[2] Vrije Universiteit Amsterdam, Department of Artificial Intelligence
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
Email: {jonker, treur}@cs.vu.nl
URL: http://www.cs.vu.nl/{~jonker,~treur}

[3] Université de Neuchâtel, Institut d'Informatique
Rue Emile Argand 11, 2007 Neuchâtel, Switzerland
Email: jean-pierre.muller@info.unine.ch
URL: http://iiun.unine.ch/People/jpmuller

Abstract. The main question addressed in this paper is how requirements on the dynamics of a multi-agent systems and individual agents can be related to the dynamics of high level concepts given by an organisation model, such as groups, roles within groups, and role interaction. In terms of these organisational concepts, which abstract from the specific agents assigned to roles, different types of requirement are introduced and formalised, which can be used as templates or patterns. Moreover, logical relationships between these different types of requirements are analysed: proof patterns. It turns out that the organisational concepts and their logical relations, form a very useful level of requirements specification mediating between specification of single agent behaviour and the overall multi-agent system dynamics. Thus the gap between single agent behaviour requirements specification and requirements specifications of the emerging overall dynamics is reduced. A contribution of this work to the area of organisation modelling is that it is shown how the dynamics of an organisation model can be specified in a structured manner using different types of behavioural requirements based on a formal temporal trace language.

1 Introduction

Requirements Engineering is a well-studied field of research. In recent years requirements engineering for distributed and agent systems has been studied, e.g., (Dardenne, Lamsweerde, and Fickas, 1993; Dubois, Du Bois, and Zeippen, 1995; Herlea, Jonker, Treur, and Wijngaards, 1999; Kontonya, and Sommerville, 1998). The approach introduced in this paper is especially relevant for requirements engineering for distributed and agent systems. At the level of the multi-agent system, requirements concern the dynamics of interaction and cooperation patterns. At the level of individual agents, requirements concern agent behaviour. Due to the dynamic

complexity, analysis and specification of such requirements is a difficult process. The importance of using more abstract and intuitive notions in requirements specification, as opposed to more directly formulated behaviour constraints, is emphasised in, e.g., (Dardenne, Lamsweerde, and Fickas, 1993). Because of their intuitive meaning and conciseness, such notions are easier to understand. In this paper organisational concepts are proposed to serve this purpose.

Organization modelling aims at abstracting away from the interactions between the agents of a complex multi-agent systems and their fundamental and recurrent patterns (Ferber and Gutknecht, 1998). For doing so, organization modelling introduces the notions of role, interaction and group structure (or organization for some authors). A role is the abstraction of a recurrent agent behaviour, linked to a status in a society and interacting with other roles. The notion of role becomes independent of any particular agent, an agent playing several roles and a role being played by several agents if needed. The interactions define the relationship linking the roles to each other. Finally a group structure is a set of roles and interactions between these providing a common context and rationale. The notion of group structure can capture goal-oriented organizations, points of view on a multi-agent system or design patterns in a software engineering perspective. The advantage of organization modelling is to deal with complexity, in particular multi-agent systems with heterogeneous global behaviours.

Formulation of requirements using more abstract organizational notions not only helps to reach a common understanding of the intended behaviour of the system, but can also help in the verification of the system being designed. For the verification of the system, the organizational notions should have well-defined logical relationships to each other and to the languages used in the (input and output interfaces of the) agents within the system.

In this paper, in Section 2 our approach to organisation modelling is briefly introduced. An example bank organisation is used to illustrate different groups and roles that can be distinguished. In Section 3 our approach to Requirements Engineering for multi-agent systems is briefly introduced and illustrated for the example multi-agent system without using an organisation model. In Section 4, in relation to an organisation model four different types of requirements are identified: single role behaviour requirements, intragroup role interaction requirements, intragroup communication successfulness requirements, and intergroup role interaction requirements. These requirements abstract from assignments of agents to groups and roles. In Section 5 it is shown how the different types of requirements are logically related to each other. A generic proof structure is introduced that bridges the gap between single agent behaviour requirements and requirements of the overall multi-agent system dynamics, by using single role requirements, intragroup role interaction requirements, intragroup communication successfulness requirements, and intergroup role interaction requirements as mediating requirements. Section 6 points out how the operational semantics based on p-calculus can be related to the requirement specifications. Section 7 is a discussion.

2 Organisation Modelling

In Section 2, it is explained how a organisational model can be used to describe actual multi-agents systems, and what concepts can be defined to support design of organisation during modelling process.

2.1 The descriptive model

For description of actual multi-agent systems from the organisational point of view, we use the "agent/group/role" model, which has been detailed in (Ferber and Gutknecht, 1998). In this definition, an organization is viewed as a framework for activity and interaction through the definition of groups, roles and their relationships. But, by avoiding an agent-oriented viewpoint, an organization is regarded as a structural relationship between a collection of agents. Thus, an organization can be described solely on the basis of its structure, i.e. by the way groups and roles are arranged to form a whole, without being concerned with the way agents actually behave, and multi-agent systems will be analyzed from the outside', as a set of interaction modes. The specific architecture of agents is purposely not addressed in the organizational model. The three primitive definitions are:

- The *agent*. The model places no constraints on the internal architecture of agents. An agent is only specified as an active communicating entity which plays roles within groups. This agent definition is intentionally general to allow agent designers to adopt the most accurate definition of agent-hood relative to their application. The agent designer isresponsible for choosing the most appropriate agent model as internal architecture.
- *Groups* are defined as atomic sets of agent aggregation. Each agent is part of one or more groups. In its most basic form, the group is only a way to tag a set of agents. An agent can be a member of *n* groups at the same time. A major point of these groups is that they can freely overlap.
- The *role* is an abstract representation of an agent function, serviceor identification within a group. Each agent can handle multiple roles, and each role handled by an agent is local to a group.

2.2 Abstract organisation structures

The aim of an organizational modelling process is to specify organisational patterns which will be present in actual multi-agent systems that can be analysed through the previous model. Thus, we need abstractions of the previous definitions. The *organizational structure* is the specification of a specific multi-agent organization. This is a weak specification, in sense that the analysis of requirements is not achieved. Section 3 will describe this process. The organizational structure, defined as a set of group structures.

The *group structure* is the abstract description of a group. It identifies the complete range of roles and interaction that might occur within a group. The group structure is defined by the set of roles, interaction schemes and communication model within the group. For a given group structure, a set of *roles* is identified. It is the enumeration of all possible roles that might be played by an agent within the group a a given time. And finally the interaction definitions are set up between roles in the group, and may involve 1 to *n* roles in a dialog. Moreover, each interaction definition must explicit the role that initiates the interaction. Figure 1 shows an organizational structure.

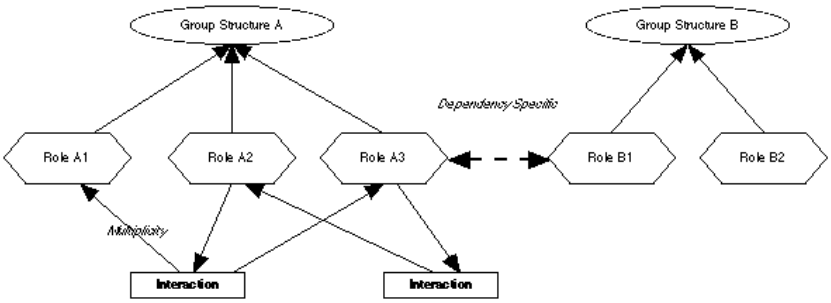

Fig. 1. Organizational Structure

2.3 An example organisation model

Within this paper examples are taken from a case study that has been performed in the context of the Rabobank, one of the largest banks in the Netherlands. The case study addressed design and analysis of an organisation model for a bank service provision problem using a Call Center. An organisation model was defined using the following groups: Client Service, Distribution, and Process Management. The roles within the groups are as follows:

Client Service:	Group Manager, Customer Relator, Client
Distribution:	Group Manager, Distributor, Participants
Process Management:	Group Manager, Process manager, Process executor

Within a Client Service group the service requests of clients are identified in an interaction between Client Relator and Client (this takes place within the Call Center). Within a Distribution group, an identified service request is allocated in mutual interaction between Distributor and Participants. Actually this process takes place in two steps: once in first line Distribution groups (relating a Call Center agent to local bank work manager agents) and once in a second line Distribution groups (work manager and employees within a local bank). The agents with role Participant in a first line Distribution group have the role of Distributor in a second line Distribution group. Also the agents with role Customer Relator in a Client Service group have a role of Distributor in a first line Distribution group.

3 Requirements for Agents and Multi-Agent Systems

In this section, following (Herlea, Jonker, Treur, and Wijngaards, 1999), it is explained how requirements can be specified for agent behaviour and for the dynamics of a multi-agent system as a whole, without assuming an organisation model. In the next sections it is shown how the organisation concepts such as groups and roles can be used to express (dynamic) properties, abstracting from the agents.

Requirements can be expressed in an informal, semi-formal or formal manner. In the context described above, the following is an informally expressed requirement for the dynamics of the multi-agent system as a whole:

R2: Each service request must be followed by an adequate service proposal after a certain time delay.

In a structured, semiformal manner, this requirement can be expressed as follows:

if at some point in time
    an agent A outputs: a service request, to an appropriate other agent B
then at a later point in time
    agent B outputs: a proposal for the request, to agent A
and at a still later point in time
    agent A outputs: proposal is accepted, to agent B

The following temporal formalisation is made:

$$\forall \mathcal{M}, t, A \ \exists B$$
$$[\ state(\mathcal{M}, t, output(A)) \models communication\_from\_to(request(r), A, B) \Rightarrow$$
$$[\ \exists t2 > t1 > t$$
$$state(\mathcal{M}, t1, output(B)) \models communication\_from\_to(proposal\_for(p, r), B, A) \wedge$$
$$state(\mathcal{M}, t2, output(B)) \models communication\_from\_to(accepted\_proposal\_for(p, r), A, B)\ ]\ ]$$

The formal language used is comparable to situation calculus (e.g., compare $\models$ to the holds-predicate), but with explicit variables for traces and time. The expression $state(\mathcal{M}, t, output(A)) \models communication\_from\_to(request(r), A, B)$ means that within trace $\mathcal{M}$ at time point t at the output interface of agent A a communication statement $communication\_from\_to(request(r), A, B)$ is put. Here a trace is a sequence over time of three-valued information states of the system, including input and output information states of all of the agents, and their environment. The time frame can be discrete, or a finite variability assumption can be used. For further details on the use of this predicate logic temporal language, see (Herlea, Jonker, Treur, and Wijngaards, 1999).

Besides requirements on the dynamics of the overall multi-agent system, also requirements can be expressed on the behaviour of single agents. For example, an agent who is expected to adequately handle service requests should satisfy the following behaviour requirements:

A1: If the agent B receives a request for a service from a client A
And the necessary information regarding this client is **not** available
Then agent B issues a request for this information to that client.

$$\forall \mathcal{M}, t, r\ [\ [\ state(\mathcal{M}, t, input(B)) \models communication\_from\_to(request(r), A, B)$$
$$\wedge\ \neg \exists i, t1 < t\ state(\mathcal{M}, t1, input(B)) \models communication\_from\_to(add\_info\_for(i, r), A, B)\ ]$$
$$\Rightarrow \exists t1 > t\ \ state(\mathcal{M}, t1, output(B)) \models communication\_from\_to(add\_info\_query\_for(r), B, A)\ ]$$

A2: If the necessary information regarding a client has become available to agent B
Then the agent B issues a request to that client to wait
And asks another agent for proposals for the request

$$\forall \mathcal{M}, t1, t2, r\ [\ [\ state(\mathcal{M}, t1, input(B)) \models communication\_from\_to(request(r), I{:}C, B)$$
$$\wedge\ \exists i\ \ state(\mathcal{M}, t2, input(B)) \models communication\_from\_to(add\_info\_for(i, r), A, B)\ ]$$
$$\Rightarrow \exists t3 > t1, t2\ \ state(\mathcal{M}, t3, output(B)) \models communication\_from\_to(hold\_on\_for(r), B, A)\ ]$$
$$\wedge\ \exists t4 > t1, t2\ \ state(\mathcal{M}, t, output(B)) \models communication\_from\_to(request(r), B, C)\ ]$$

Requirements on the dynamics of a multi-agent system are at a higher process abstraction level than the behaviour requirements on agents. In (Jonker and Treur, 1998) it is shown how refinement relations between requirements at different process abstraction levels can be exploited to prove the higher level requirements from the lower level ones (compositional verification). Verification of the multi-agent system dynamics from the individual agents requires that adequate proof structures have to be found, which may be difficult without more specific support. In the next subsection the organisation model is used to define mediating requirement types. Section 5 proposes generic proof patterns on the basis of an organisation model.

4 Behavioural Requirements within an Organisation Model

In this section an organisation model is assumed by which the work flows are structured and coordinated. Based on this organisational structure, the following types of requirements are distinguished:

- single role behaviour requirements
- intragroup interaction requirements
- intragroup communication succesfulness requirements
- intergroup interaction requirements

Each of these types is discussed below in more detail.

To be able to specify ongoing interaction between two roles for which multiple appearances exist, the notion of *role instance* is used. This notion still abstracts from the agent realising the role as actor, but enables to distinguish between appearances of roles. For example, the role 'secret agent' has role instances like 'secret agent 001' to 'secret agent 009', and the role instance 'secret agent 007' is sometimes realised by the agent James Bond, sometimes by the agent Jack Jackson (his predecessor), and so on. If R is a role, then a role instance for R is denoted by I:R; so, for example, 007:secret agent denotes the role instance 'secret agent 007'. Throughout this paper the identifiers used to denote instances are considered to be unique (this leads to better readable properties). As for roles, also *group instances* are used. In our example, the Distributor role instance within the Distribution group instance has interaction with the Customer Relator role instance within the Client service group instance.

4.1 Single role behaviour requirements

For a given role within a group, role behaviour requirements specify the dynamics of the role within the group. They are typically expressed in terms of temporal relationships between the *input* and *output* of the role instance, according to the following pattern:

if role instance I:R receives as input ....
[and in the past as input of I:R it received ...
and in the past at the output of I:R it was generated ...]
then some time later role instance I:R generates as output ....

The following are examples of role behaviour requirements (within the Client Service group).

Client role behaviour requirements specification

C1: If the client receives a request for additional information
Then the client either provides this information
Or the client provides a “bye-bye”.

$\forall \mathcal{M}$, t , r, I:C, J:CR
[state($\mathcal{M}$, t, input(I:C)) $\models$ communication_from_to(add_info_query_on(r), J:CR, I:C)

$\Rightarrow$ [$\exists$ t1 > t , i
state($\mathcal{M}$, t1, output(I:C)) $\models$ communication_from_to(add_info_on(i, r), I:C, J:CR)
$\vee$ state($\mathcal{M}$, t1, output(I:C)) $\models$ communication_from_to(bye_bye, I:C, J:CR)]]

C3: If the client receives a proposal for a service requested by him
Then the client either accepts the proposal
Or the client rejects the proposal.

$\forall \mathcal{M}$, t , p, r, I:C, J:CR
[state($\mathcal{M}$, t, input(I:C)) $\models$ communication_from_to(proposal_for(p, r), J:CR, I:C)
$\Rightarrow$ [$\exists$ t1 > t
state($\mathcal{M}$, t1, output(I:C)) $\models$ communication_from_to(accepted_proposal_for(p, r), I:C, J:CR)
$\vee$ state($\mathcal{M}$, t1, output(I:C)) $\models$ communication_from_to(rejected_proposal_for(p, r), I:C, J:CR)
$\vee$ state($\mathcal{M}$, t1, output(I:C)) $\models$ communication_from_to(bye_bye, I:C, J:CR)]]

Client Relator role behaviour requirements specification

CR1: If a customer relator receives a request for a service from a client
And the necessary information regarding this client is **not** available
Then the customer relator issues a request for this information to that client.

$\forall \mathcal{M}$, t , r, I:C, J:CR
[[state($\mathcal{M}$, t, input(J:CR)) $\models$ communication_from_to(request(r), I:C, J:CR) $\wedge$
$\neg$ $\exists$ i, t1 < t
state($\mathcal{M}$, t1, input(J:CR)) $\models$ communication_from_to(add_info_for(i, r), I:C, J:CR)]
$\Rightarrow \exists$ t1 > t
state($\mathcal{M}$, t1, output(J:CR)) $\models$ communication_from_to(add_info_query_for(r), J:CR, I:C)]

CR2: If the necessary information regarding a client has become available
Then the customer relator issues a request to that client to wait.

$\forall \mathcal{M}$, t1, t2 , r, I:C, J:CR
[[state($\mathcal{M}$, t1, input(J:CR)) $\models$ communication_from_to(request(r), I:C, J:CR)
$\wedge$ $\exists$ i state($\mathcal{M}$, t2, input(J:CR)) $\models$ communication_from_to(add_info_for(i, r), I:C, J:CR)]
$\Rightarrow$ $\exists$ t > t1, t2
state($\mathcal{M}$, t, output(J:CR)) $\models$ communication_from_to(hold_on_for(r), J:CR, I:C)]

4.2 Intragroup role interaction requirements

Intragroup role interaction requirements specify the temporal constraints on the dynamics of the interaction protocol between two roles within a group. Intragroup role interaction requirements between two roles instances *I:R1* and *J:R2* in one group instance are typically expressed in terms of the *output* of both role instances, according to the following pattern:

if role instance I:R1 generates as output ....
[and in the past at the output of I:R1 it was generated ...
and in the past at the output of J:R2 it was generated ...]
then some time later role instance J:R2 generates as output ....
or role instance I:R1 generates as output ....

The last conclusion enables, for example, to specify that a request from one role instance to another role instance is withdrawn before the request actually was answered, thus taking away the urge to answer it. Of course it is also possible to express interaction involving more than two role instances. In the simplest situations no references to the past are made, and the pattern takes the form of a direct reactivenes relation:

if role instance I:R1 generates as output ....
then some time later role instance J:R2 generates as output ....

Client service group role interaction requirements

The Client role instances interact with Customer Relator role instances. In the formulae below we use I:C for a Client role instance, and J:CR for a Customer Relator role instance.

CSRI1: If a client provides on his output a service request
Then some time later:
the customer relator provides for that client on his output a proposal for that request
Or that client provides on his output a "bye-bye".

If a bye-bye has been communicated by a role instance I:R1 to a role instance J:R2, it is assumed that from that moment on I:R1 will not listen to communication directed from J:R2 to I:R1, e.g., because the telephone connection has been closed. This has impact on the communication successfulness requirements that will be discussed later. Temporal formalisation of CSRI1:

$\forall \mathfrak{M}$, t, r, I:C, J:CR
[state($\mathfrak{M}$, t, output(I:C)) $\models$ communication_from_to(request(r), I:C, J:CR)
$\Rightarrow$ [$\exists$ t1 > t, p
state($\mathfrak{M}$, t1, output(J:CA)) $\models$ communication_from_to(proposal_for(p, r), J:CR, I:C)
$\vee$ state($\mathfrak{M}$, t1, output(I:C)) $\models$ communication_from_to(bye_bye, I:C, J:CR)]]

CSRI2: If the customer relator provides on his output a request for additional information to a client

Then some time later:
that client provides on his output either the requested additional information
Or that client provides on his output a “bye-bye”.

$\forall \mathfrak{M}$, t , r, I:C, J:CR
[state($\mathfrak{M}$, t, output(J:CR)) $\models$ communication_from_to(add_info_query_on(r), J:CR, I:C)
$\Rightarrow$ [$\exists$ i, t1 > t
state($\mathfrak{M}$, t1, output(I:C)) $\models$ communication_from_to(add_info_on(i, r), I:C, J:CR)
$\vee$ state($\mathfrak{M}$, t1, output(I:C)) $\models$ communication_from_to(bye_bye, I:C, J:CR)]]

4.3 Intragroup communication successfulness requirements

Intragroup role interaction requires communication within the group. Therefore, in order to function properly, requirements are needed that communications are succesful. These requirements have the following pattern:

if role instance I:R1 generates as output a communication directed to J:R2
[and ...]
then some time later role instance J:R2 receives as input this communication

Such a requirement may not hold unconditionally; it may as well be domain and time dependent. In particular, if the other role instance communicated earlier a bye-bye, then we don’t assume it is listening anymore (it may have put the telephone off). We also include the condition that the previous communication of role instance J:R2 was not a request to wait (hold on); it might be the case that after such a request J:R2 will not listen until it resumes communication. A Communication Successfulness requirement then specifies, for example, the following:

CS1: If communication is directed to a role instance
Then it will always receive this communication,
Unless its previous communication was either ‘hold on’ or ‘bye-bye’.

$\forall \mathfrak{M}$, t , r, φ, R1, R2, I:R1, J:R2
[[state($\mathfrak{M}$, t, output(I:R1)) $\models$ communication_from_to(φ, I:R1, J:R2)
$\wedge$ $\neg$ $\exists$ t1 < t [state($\mathfrak{M}$, t, output(J:R2)) $\models$ communication_from_to(bye_bye, J:R2, I:R1)
$\wedge$ $\forall$ t2, φ1 t1 < t2 < t $\Rightarrow$
$\neg$ state($\mathfrak{M}$, t2, output(J:R2)) $\models$ communication_from_to(φ1, J:R2, I:R1)]
$\wedge$ $\neg$ $\exists$ t3 < t
[state($\mathfrak{M}$, t, output(J:R2)) $\models$ communication_from_to(hold_on_for(r), J:R2, I:R1)
$\wedge$ $\forall$ t4, φ2 t3 < t4 < t $\Rightarrow$
$\neg$ state($\mathfrak{M}$, t, output(J:R2)) $\models$ communication_from_to(φ2, J:R2, I:R1)]]
$\Rightarrow$ $\exists$ t2 > t state($\mathfrak{M}$, t2, input(J:R2)) $\models$ communication_from_to(φ, I:R1, J:R2)]

As an example of a more specific communication succesfulness assumption, consider the following informal requirement:

CS2: The client is always allowed to issue a request to the customer relator.

This is formalised by the following temporal statement:

$\forall \mathcal{M}$, t , r, I:C, J:CR

[[state($\mathcal{M}$, t, output(I:C)) $\models$ communication_from_to(request(r), I:C, J:CR)

$\Rightarrow$ $\exists$ t1 > t state($\mathcal{M}$, t1, input(J:CR)) $\models$ communication_from_to(request(r), I:C, J:CR)]

4.4 Intergroup role interaction requirements

Intergroup role interaction requirements specify the temporal constraints on the dynamics of the interaction protocol between two role instances within two different group instances. They are typically expressed in terms of the *input* of one of the role instances and the *output* of the other one, according to the following pattern, in which I:R1 is a role instance within group instance GI1:G1 of group G1, and J:R2 is a role instance within group instance GI2:G2 of group G2:

if role instance I:R1 receives as input ....
[and in the past at the input of I:R1 it was received ...
and in the past at the output of J:R2 it was generated ...
and in the past at the input of J:R2 it was received ...]
then some time later role instance J:R2 generates as output ....

Note that more role instances (of more group instances) might be involved.

Client Service group - Distribution group role interaction requirements

The interaction between Client Service group and Distribution group is realised by one agent to fulfill both roles. However, independent of the assignment of agents to roles, the required intergroup role interaction can be specified. Here I:C is a Client role instance, J:CR is a Customer Relator role instance within the Client Service group instance and K:D a Distributor role instance, and L:P a Participant role instance within the Distribution group instance.

The first intergroup interaction requirement between the Client service and Distribution group instances states that if a Customer Relator role instance J:CR receives a service request as input within the Client service group instance, then this request is put forward within the Distribution group instance by the Distributor role instance K:D to all Participant role instances L:P.

CSDRI1:

$\forall \mathcal{M}$, t , r, I:C, J:CR

[state($\mathcal{M}$, t, input(J:CR)) $\models$ communication_from_to(request(r), I:C, J:CR)

$\Rightarrow$ $\exists$ K:D, t1 > t $\forall$ L:P

state($\mathcal{M}$, t1, output(K:D)) $\models$ communication_from_to(request(r), K:D, L:P)]

The second intergroup interaction requirement between the Client Service and Distribution group instances states that if a Distributor role instance K:D receives as input within the Distributor group instance a service proposal on a request, then this proposal will be put forward within the Client service group instance by the Customer Relator role instance J:CR that received the request earlier, and is directed to the Client role instance I:C from which the request originates.

CSDRI2:

$\forall \mathfrak{M}$, t , r, p, I:C, K:D, L:P

[[state($\mathfrak{M}$, t, input(K:D)) $\models$ communication_from_to(proposal_for(p, r), L:P, K:D)

$\wedge$ $\exists$ J:CR, t1 < t

state($\mathfrak{M}$, t1, input(J:CR)) $\models$ communication_from_to(request(r), I:C, J:CR)]

$\Rightarrow$ $\exists$ t2 > t

state($\mathfrak{M}$, t2, output(J:CR)) $\models$ communication_from_to(proposal_for(p, r), J:CR, I:C)]

It was decided not to add the client name to the requests in the further processing. It is assumed that requests have unique names, so that no request will relate to two clients; the Customer relator knows which client issued what request.

5 Logical Relations between the Different Types of Requirements

In Section 3 single agent behaviour requirements were distinguished from requirements on the dynamics of the whole multi-agent system, without taking into account an organisation model. In Section 4 four types of organisational requirements were distinguished. In this section it is shown how these two and four different types of requirements are logically related (see also Figure 2). The different types of organisational requirements provide a predefined structure (a kind of blueprint) for a proof of the requirements of the multi-agent system as a whole from the single agent behaviours. In Figure 2 an overview of these logical relations is presented; in this picture logical entailment is directed from right to left. Besides the normal logical deduction rules an additional deduction rule is introduced that allows for the substitution of agents by role instances. The agent-role substitution deduction rule (ARS) is defined by the following. Let φ be an arbitrary temporal formula, A an agent, I:R a role instance.

$$\frac{\text{role\_assignment}(A, I : R) \qquad \varphi(A)}{^{2}(I : R)}\ \text{ARS}$$

All logical relationships depicted in Figure 2 will be illustrated by our example.

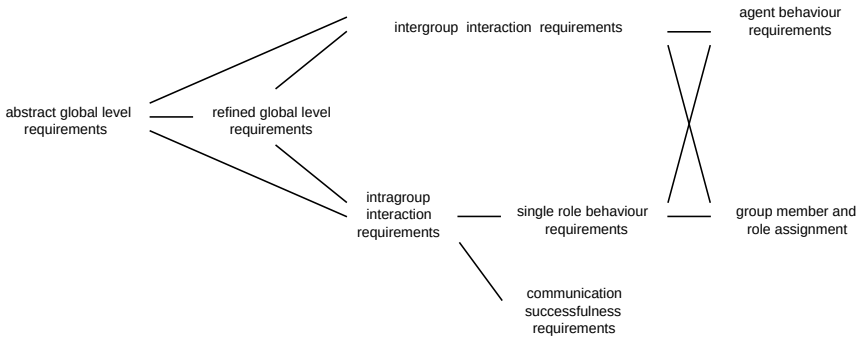

Fig. 2. Logical relationships between the different types of requirements

5.1 Agent Behaviour implies Single Role Behaviour

The agent behaviour requirement pattern relates output of the agent to input; the single role behaviour requirement pattern relates output of the role to input. Given role assignments to agents, the two patterns can be easily related to each other using the substitution rule ARS.

5.2 Single Role Behaviour implies Intragroup Role Interaction

Using the communication successfulness requirements, intragroup role interaction requirements can be proven from single role behaviour requirements. The intragroup role interaction pattern relates output of a role instance I:R1 to output of another role instance J:R2. This can be split up using the communication successfulness requirement pattern (relating output of role instance I:R1 to input of J:R2), and the single role behaviour pattern (relating output of role instance J:R2 to input of J:R2). So, we have the following schematic proof pattern:

CS: *output I:R1* => *input J:R2*
SRB: *input J:R2* => *output J:R2*

IARI: *output I:R1* => *output J:R2*

As an example, within the Client service group, the intragroup role interaction property **CSRI2** is logically implied by the Client role behaviour property **C1** and the Communication Successfulness requirement **CS1** on communication from Customer Relator role instances to Client role instances.

5.3 Agent Behaviour implies Intergroup Role Interaction

The interegroup role interaction requirement pattern relates output of a role instance in one group to input of a role instance in another group. Assuming one agent is assigned to both role instances, and using the substitution rule ARS this can be reduced to the single agent behaviour requirement pattern. In this case a group member and role assignment specification is assumed such that CR and D both have exactly one instance j:CR, resp. k:D, and there is exactly one agent assigned both roles instances:

is_member_of(sonny, csg1:client_service_group)
is_member_of(sonny, dg1:distribution_group)
role_assignment(sonny, k:D)
role_assignment(sonny, j:CR)

Assuming there is exactly one agent assigned to both roles, like sonny for k:D and j:CR, then the agent behaviour requirement specifications for that agent can be used to prove the intergroup role interaction requirement as follows. Consider, e.g., the following agent behaviour requirement specification for sonny:

$$
\begin{aligned}
&\forall\ A, B\ \forall\ \mathcal{M}\ , t\ , r, p \\
&\quad [\ [\ state(\mathcal{M}, t, input(sonny))\ \models\ communication\_from\_to(proposal\_for(p, r), A, sonny) \\
&\wedge\ \ \exists\ t1 < t\ \ state(\mathcal{M}, t1, input(sonny)) \models communication\_from\_to(request(r),\ B, sonny)\ ] \\
&\Rightarrow\ \exists\ t2 > t \\
&\quad state(\mathcal{M}, t2, output(sonny)) \models communication\_from\_to(proposal\_for(p, r), sonny, B)\ ]
\end{aligned}
$$

From this the following has to be proved, using the deduction rule ARS:

$$
\begin{aligned}
&\forall\ K{:}D, \forall\ I{:}C, L{:}P, \forall\ \mathcal{M}\ , t\ , r, p \\
&\quad [\ [\ state(\mathcal{M}, t, input(K{:}D))\ \models\ communication\_from\_to(proposal\_for(p, r), L{:}P, K{:}D) \\
&\wedge\ \exists\ J{:}CR,\ t1 < t \\
&\quad\ state(\mathcal{M}, t1, input(J{:}CR)) \models communication\_from\_to(request(r),\ I{:}C, J{:}CR)\ ] \\
&\Rightarrow\ \exists\ t2 > t \\
&\quad\ state(\mathcal{M}, t2, output(J{:}CR)) \models communication\_from\_to(proposal\_for(p, r), J{:}CR, I{:}C)\ ]
\end{aligned}
$$

One has to show that holds for every role instance of K:D:

$$
\begin{aligned}
&\forall\ I{:}C, L{:}P, \forall\ \mathcal{M}\ , t\ , r, p \\
&\quad [\ [\ state(\mathcal{M}, t, input(K{:}D))\ \models\ communication\_from\_to(proposal\_for(p, r), L{:}P, K{:}D) \\
&\wedge\ \exists\ J{:}CR,\ t1 < t \\
&\quad\ state(\mathcal{M}, t1, input(J{:}CR)) \models communication\_from\_to(request(r),\ I{:}C, J{:}CR)\ ] \\
&\Rightarrow\ \exists\ t2 > t \\
&\quad\ state(\mathcal{M}, t2, output(J{:}CR)) \models communication\_from\_to(proposal\_for(p, r), J{:}CR, I{:}C)\ ]
\end{aligned}
$$

Now, there is only one role instance for D, namely k:D. Within the specification of group membership and role assignments, the agent can be found (i.e., sonny) that is assigned that role. The next task is to prove ∃ J:CR as in the requirement. By checking the role assignment information, j:CR is found to be the only role instance. Since sonny is assigned to both role instances, within the ARS rule the occurences of sonny can be replaced by these role instances. Note that the set of agents having role assignment Participant is a subset of the set of all agents, and similarly the set of agents having role assignment Client is a subset of the set of all agents. Apply ASR four times: substitute k:D for sonny, and j:CR for sonny, and L:P for A, and I:C for B in the agent behaviour requirement specification:

$$
\begin{aligned}
&\forall\ I{:}C, L{:}P, \forall\ \mathcal{M}\ , t\ , r, p \\
&\quad [\ [\ state(\mathcal{M}, t, input(k{:}D))\ \models\ communication\_from\_to(proposal\_for(p, r), L{:}P, k{:}D) \\
&\quad\ \wedge\ \exists\ t1 < t\ \ state(\mathcal{M}, t1, input(j{:}CR)) \models communication\_from\_to(request(r),\ I{:}C, j{:}CR)\ ] \\
&\Rightarrow\ \exists\ t2 > t \\
&\quad\ state(\mathcal{M}, t2, output(j{:}CR)) \models communication\_from\_to(proposal\_for(p, r), j{:}CR, I{:}C)\ ]
\end{aligned}
$$

Then introduce a "∀" quantifier such that K:D appears, and an "∃" quantifier for J:CR. This provides the specified Intergroup Role Interaction requirement:

$$
\begin{aligned}
&\forall\ K{:}D, \forall\ I{:}C, L{:}P, \forall\ \mathcal{M}\ , t\ , r, p \\
&\quad [\ [\ state(\mathcal{M}, t, input(K{:}D))\ \models\ communication\_from\_to(proposal\_for(p, r), L{:}P, K{:}D)\ \ \wedge \\
&\exists\ J{:}CR,\ t1 < t\ \ state(\mathcal{M}, t1, input(J{:}CR)) \models communication\_from\_to(request(r),\ I{:}C, J{:}CR)] \\
&\Rightarrow\ \exists\ t2 > t \\
&\quad\ state(\mathcal{M}, t2, output(J{:}CR)) \models communication\_from\_to(proposal\_for(p, r), J{:}CR, I{:}C)]
\end{aligned}
$$

5.4 Proving an overall requirement of the multi-agent system

For example, the requirement R2 for the multi-agent system as a whole can be proven from a number of organisational intragroup and intergroup requirements, such as the intragroup and intergroup interaction requirements discussed above.

6 Relation between π-calculus semantics and requirements

The link between behavioral requirement and organizational semantics is made at the agent and role instances level. In the p-calculus expressions, we represent the behaviour of an agent by a set of equations representing a process *P* of the form: A_j (*id*, v_{ij}, …, v_{kj}), with *id* the identity of the agent and v*ij* its internal variables and operations. As we represent the operations of an agent as a set of role behaviors, and as the role definition processes are constrained to only refer to the identity process of the agent and the role behavior itself, all roles are opaque to one another. Moreover, the *self behavior* process handles all communications for a given agent, through the *id* symbol that correspond to a single communication channel (see Ferber, Gutknecht 1999 for details). This implies that at any time of the execution of the system, there is no rewriting rule that can directly combine processes defined from different role behaviours (i.e., different role instances) in a single agent. Communication between roles instances are conveyed through the identity symbols of the agents. The global state of the system which appears in the requirements corresponds to the state of the various rewrite contexts (the *solutions*), the execution traces being the sequence of p-calculus reductions of these expressions. Thus, for each role behavior described by a set of processes in the p-calculus, the requirements for intra-group and inter-group interactions expressed in the form described in Section 4 will hold.

7 Discussion

In practice, specification of requirements, in particular for distributed systems with nontrivial dynamics, is a hard problem; e.g., (Dardenne, Lamsweerde, and Fickas, 1993; Dubois, Du Bois, and Zeippen, 1995; Herlea, Jonker, Treur, and Wijngaards, 1999; Kontonya, and Sommerville, 1998). A main question to which this paper contributes is how to formulate requirements on the dynamics of a multi-agent system, and how to relate these requirements to requirements on individual agent behaviour, from which the overall multi-agent system dynamics emerges. The approach proposed in this paper is a to use high level concepts in specification of the dynamics of the system, based on an organisation model. The organisation model structure used includes high level concepts such as groups and roles within groups, and (intragroup and intergroup) role interaction.

Purely in terms of these organisation concepts (not referring to specific agents), different types of requirement patterns have been defined and formalised: single role behaviour requirements, intergroup role interaction requirements, intragroup communication successfulness requirements, and intragroup role interaction requirements. Also the logical relationships between these different types of specifications have been established (proof patterns). In particular, single role behaviour specifications together with intergroup role interaction specifications logically imply intragroup role interaction specifications. These organisational notions also have been formally related to requirements on single agent behaviour, and

requirements for the overall multi-agent system dynamics. In particular, it has been shown that both single role behaviour specifications and intergroup role interaction specifications are logically implied by single agent behaviour specifications plus the assigments of agents to groups and roles. For an overview of these logical relationships, Figure 2 has been included. It turns out that the organisational concepts and their logical relations, form a very useful level of specification mediating between specification of single agent behaviour and the overall multi-agent system dynamics. Thus the gap between single agent behaviour requirements specification and requirements specifications of the emerging overall dynamics is reduced, by making their relationships more transparent in terms of additional high level organisational concepts.

In comparison, the KAOS approach to Requirements Engineering of composite systems (Dardenne, Lamsweerde, and Fickas, 1993; Darimont, and Lamsweerde, 1996; Lamsweerde, Darimont, and Letier, 1998), has some similarity in argument. In their approach a requirement for the overall system is called a *goal*. What they call a *requisite* is a requirement on part of the dynamics controllable by a single agent or (given) environment component. This could be compared to our notion of role behaviour requirements specification. Goal refinement is used to decompose goals into requisites via AND/OR graphs; this can be compared to our logical relationships between the requirements specification of the overall multi-agent system dynamics and the (intragroup and intergroup) role interaction requirements specifications. In their approach assignments of agents to roles are done using *responsibility links*. What they call requirement is in our terms a single agent behaviour requirement specification. Also the term *assumption* is used to indicate requisites on (given) environmental components. They have no concept for group as in our approach. Moreover, they do not distinguish role interaction requirement as a separate concept, neither do they distinguish between intragroup role interaction and intergroup role interaction. Overall, in contrast our approach has a detailed organisation model as a basis, in which groups, roles, and their interaction are distinguished. Furthermore, the organisation model is reflected in the generic proof structures, and therefore provides more support for requirements engineering and verification.

In (Yu, Du Bois, Dubois, and Mylopoulos, 1997) another analysis of the relationship between Requirements Engineering and organisation modelling is made. Their study uses a richer organisation model than the KAOS approach, based on concepts such as goals, tasks, resources and dependencies. Besides that the organisation model differs, they did not establish generic types of requirements and logical relationships between them as is achieved in this paper.

A contribution of this work to the area of organisation modelling is that it is shown how the dynamics of an organisation model can be specified in a structured manner using different types of behavioural requirements based on a temporal formal language.

References

Dardenne, A., Lamsweerde, A. van, and Fickas, S. (1993). Goal-directed Requirements Acquisition. Science in Computer Programming, vol. 20, pp. 3-50.

Darimont, R., and Lamsweerde, A. van (1996). Formal Refinement Patterns for Goal-Driven Requirements Elaboration. Proc. of the Fourth ACM Symposium on the Foundation of Software Engineering (FSE4), pp. 179-190.

Dubois, E., Du Bois, P., and Zeippen, J.M. (1995). A Formal Requirements Engineering Method for Real-Time, Concurrent, and Distributed Systems. In: Proceedings of the Real-Time Systems Conference, RTS'95.

Ferber, J. and Gutknecht, O. (1998). A meta-model for the analysis and design of organizations in multi-agent systems. *Third International Conference on Multi-Agent Systems (ICMAS '98) Proceedings.* IEEE Computer Society, 1998

Ferber, J. and Gutknecht, O. (1999). Operational Semantics of a role-based agent architecture. *Proceedings of the 6th Int. Workshop on Agent Theories, Architectures and Languages*. Lecture Notes in AI, Springer-Verlag. In press.

Herlea, D.E., Jonker, C.M., Treur, J., and Wijngaards, N.J.E. (1999). Specification of Behavioural Requirements within Compositional Multi-Agent System Design. In: F.J. Garijo, M. Boman (eds.), *Multi-Agent System Engineering, Proc. of the 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW'99*. Lecture Notes in AI, vol. 1647, Springer Verlag, 1999, pp. 8-27.

Jonker, C.M., and Treur, J., Compositional Verification of Multi-Agent Systems: a Formal Analysis of Pro-activeness and Reactiveness. In: W.P. de Roever, H. Langmaack, A. Pnueli (eds.), Proceedings of the International Workshop on Compositionality, COMPOS'97. Lecture Notes in Computer Science, vol. 1536, Springer Verlag, 1998, pp. 350-380

Kontonya, G., and Sommerville, I. (1998). Requirements Engineering: Processes and Techniques. John Wiley and Sons, New York.

Lamsweerde, A. van, Darimont, R., and Letier, E. (1998). Managing Conflicts in Goal-Driven Requirements Engineering. IEEE Transactions on Software Engineering, Special Issue on Managing Inconsistency in Software Engineering.

Milner, R. and Parrow, J. and Walker, D. (1992). A Calculus of Mobile Processes. Information and Computation, vol 100.

Yu, E., Du Bois, P., Dubois, E., Mylopoulos, J. (1997). From Organization Models to System Requirements: A Cooperating Agents Approach. In: Cooperative Information Systems: Trends and Directions, M. P. Papazoglou and G. Schlageter (eds), Academic Press, 1997. pp. 293-312.