# Organisational Abstractions for the Analysis and Design of Multi-Agent Systems

Franco Zambonelli* Nicholas R. Jennings† Michael Wooldridge‡

* Dipartimento di Scienze dell'Ingegneria
Università di Modena e Reggio Emilia
Via Campi 213-b – 41100 Modena, Italy
franco.zambonelli@unimo.it

† Department of Electronics and Computer Science
University of Southampton, Southampton SO17 1BJ, United Kingdom
nrj@ecs.soton.ac.uk

‡ Department of Computer Science, University of Liverpool
Liverpool L69 7ZF, United Kingdom
M.J.Wooldridge@csc.liv.ac.uk

**Abstract.** The architecture of a multi-agent system can naturally be viewed as a computational organisation. For this reason, we believe organisational abstractions should play a central role in the analysis and design of such systems. To this end, the concepts of agent roles and role models are increasingly being used to specify and design multi-agent systems. However, this is not the full picture. In this paper we introduce three additional organisational concepts — organisational rules, organisational structures, and organisational patterns — that we believe are necessary for the complete specification of computational organisations. We view the introduction of these concepts as a step towards a comprehensive methodology for agent-oriented systems.

## 1 Introduction

Autonomous agents and multi-agent systems (MASs) are rapidly emerging as a powerful paradigm for designing and developing complex software systems. However, as is the case with any new software engineering paradigm, the successful and widespread deployment of MASs requires not only new models and technologies, but also new *methodologies* to support developers engineer such systems in a robust, reliable, and repeatable fashion. In the last few years, there have been several attempts to develop such methodologies. However, most of this work is either tuned to specific systems and agent architectures [9, 4] — thus it lacks generality — or it is defined as an extension of existing object-oriented methodologies [14] — thus it exploits abstractions that are unsuitable for modelling agent-based systems.

Against this background, only a few proposals exist that attempt to define complete and general methodologies, specifically tailored to the analysis and design of MASs. One such methodology is Gaia [32]. Gaia views the process of analysing and designing multi-agent systems as one of constructing computational organisations. Thus, multi-agent systems are viewed as being composed of a multitude of autonomous interacting entities (an *organised society* of individuals) in which each agent plays one (or more) specific *roles*. In particular, Gaia, like a few other agent-oriented methodologies [9, 7, 16], suggests defining the structure of a MAS in terms of a *role model*. This model identifies the roles that agents have to play within the MAS and the interaction protocols in which the different roles are involved.

The adoption of a role model as the main organisational abstraction makes the above mentioned methodology mostly targetted at MASs in which the agents are cooperative and in which the system is closed. However, in order to deal with systems that involve self-interested agents operating in an open environment, we believe that additional organisational abstractions have to be introduced in a methodology [33]. In particular, we believe that *organisational rules*, *organisational structures*, and *organisational patterns* must also play a primary role in the analysis and design of MASs. Organisational rules express general, global (supra-role) requirements for the proper instantiation and execution of a MAS. An organisational structure defines the specific class (among the many possibilities) of organisation and control regime to which the agents/roles have to conform in order for the whole MAS to work efficiently and according to its specified requirements. Organisational patterns express pre-defined and widely used organisational structures that can be re-used from system to system (in a manner similar to the way catalogues of patterns are widely exploited in the design of object-oriented systems) [11].

In this paper, we show, with the aid of two application examples, that adoption of the above organisational abstractions can lead to a methodology that is applicable to a wide spectrum of agent systems. We also believe that the introduction of high-level organisational abstractions can lead to more clean, manageable, and re-usable MAS designs. Specifically, the paper is organised as follows. Section 2 introduces the basic concepts underlying agents and multi-agent systems. Section 3 introduces the additional organisational abstractions that are needed for a methodology to apply to open systems and motivates their adoption. Section 4 briefly sketches how our organisational abstractions can be exploited during the analysis and design of MASs. Section 5 discusses related work in this area and section 6 concludes by outlining the open issues and the future research directions.

## 2 Multi-Agent Systems and Organisations

Agents are software entities that exhibit *autonomous* and *proactive* goal-directed behaviour — their activities are not subject to a global flow of control and they can take the initiative where appropriate — and that are *reactive* to changes in the environment in which they are situated [31, 19]. These characteristics make agents useful as stand-alone entities that are delegated to accomplish a given task on behalf of a user (e.g., personal digital assistants, e-mail filters, or simple robots). However, in the majority of cases, agents exist in the context of *multi-agent software systems*, whose global behaviour derives from the interaction among the constituent agents [13]. In these cases, agents also exhibit *social* behaviour; they interact with one another: either to cooperate to achieve a common objective or because this helps each of the interacting agents to achieve their own objectives.

Here, we distinguish between two main classes of multiple agent system: *(i) distributed problem solving systems* in which the component agents are explicitly designed to cooperatively achieve a given goal, and *(ii) open systems* in which agents, not necessarily co-designed to share a common goal, can dynamically leave and enter the system. In the former case, all agents are known a priori, and all agents are supposed to be benevolent to each other and, therefore, they can trust one another during interactions. In the latter case, the dynamic arrival of unknown agents needs to be taken into account, as well as the possibility of self-interested behaviour in the course of the interactions.

### 2.1 The Organisational Metaphor

The design of parallel and distributed applications, as well as of distributed object systems, usually relies on an architecture that derives from the decomposition of the functionalities and data required by the system to achieve its goal, and on the definition of their inter-dependencies [2]. In MASs, however, the autonomous and proactive behaviour of the constituent agents suggests that applications can be designed by mimicking the behaviour and structure of human organisations. Thus each agent is assigned a specific role in the system. That is, a well-defined task/responsibility in the context of the overall system, that the agent has to accomplish in an autonomous fashion, without any centralised control. In this model, interactions are no longer merely an expression of inter-dependencies, rather they are viewed as a means for an agent to accomplish its role in the organisation. Therefore, interactions are well-identified and localised in the definition of the role itself, and they help characterise the position of the agent in the organisation.

An organisational perspective can also make the design of the system less complex and easier to manage than more traditional metaphors for concurrent systems. Firstly, each agent becomes a separate *locus* of control, in charge of accomplishing its role and being fully responsible for it. Secondly, since agents typically embed most of the functionality they need to accomplish their role, inter-dependencies between the system components are likely to be reduced. When taken together, these points ease the design process because they lead to a cleaner separation between the component-level (i.e., intra-agent) and system-level (i.e., inter-agent) design dimensions.

A final advantage relates to the fact that, in many cases, MASs are intended to support and/or control some real-world organisation. For example, MASs can be adopted to support the workflow management in a team or to help control the activities of an Internet auction. In such cases, an organisation-based MAS design reduces the conceptual distance between the software system and the real-world system it has to support. Consequently, this simplifies the development of the system.

### 2.2 An Organisational Characterisation of Multi-Agent Systems

The organisational perspective leads to a general characterisation of a MAS as depicted in figure 1 [7, 15]. Although some simpler systems can be viewed as a single organisation, as soon as the complexity increases, *modularity* and *encapsulation* principles suggest splitting the system into different sub-organisations. Thus, in most cases, a complex multi-agent system can be viewed as several *interacting organisations*. Naturally a given agent can be part of multiple organisations.

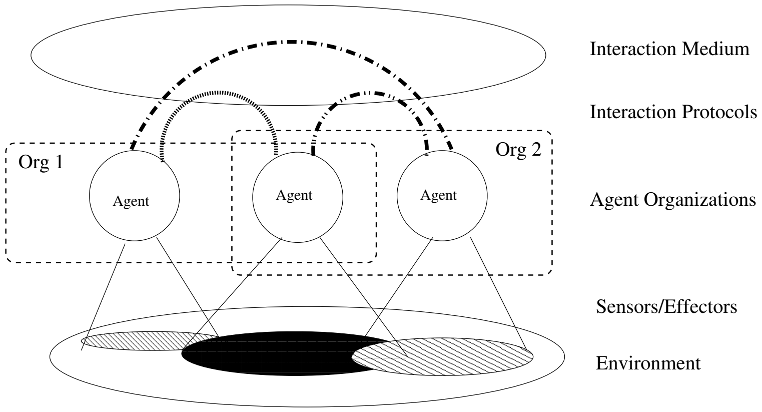


**Fig. 1.** Characterisation of a Multi-Agent System

In each organisation, an agent can play one or more *roles*. The role is what the agent is expected to do in the organisation: both in cooperation with the other agents and in respect of the organisation itself.

Often, the role of an agent is simply defined in terms of the specific task that the agent has to accomplish in the context of the overall organisation. However, in our work the notion of a role is much more precise, in that it gives an agent a well-defined position in the organisation, with a set of associated expected behaviours.

To accomplish their role in the organisation, agents typically need to *interact* with each other in order to *exchange knowledge* and *coordinate* their activities. Therefore, the concept of inter-agent interactions is strictly related to the role of an agent. It is the role that requires a given form of interaction. Even more precisely, an agent, by the very fact that it plays a given role and has a well-defined position in the organisation, is committed to certain interaction protocols with the other agents in the organisation. Of course, the need to interact according to specific protocols requires the presence of a communication medium between agents. This can be either a traditional network infrastructure, typically enforcing a message-passing interaction model, or another infrastructure possibly enforcing a different interaction model (e.g., a shared dataspace enforcing an indirect, data-oriented interaction model [12]).

Generally speaking, a MAS is immersed in a given *environment* with which the agents may need to interact in order to accomplish their role. This interaction occurs via *sensors* and *effectors*, i.e., mechanisms that enable agents to sense and effect a selected portion of the environment. That portion of the environment that an agent can sense and effect is determined by the agent's specific role, as well as on its current status.

### 2.3 Exemplar Multi-Agent Systems

To illustrate our points on the need for organisational abstractions, we will consider two sample problems that will act as running examples throughout this paper.

*Manufacturing Pipeline:* As an example of a MAS that belongs to the class of distributed problem solvers, we will consider a system for the control of a manufacturing process. For example, let us consider the process of assembling, painting and packing metal hardware. Typically, such a control system can be delegated to a multiplicity of agent organisations, each devoted to the control of a well-defined portion of the overall manufacturing process (e.g., the assembling section or the painting section). Within each section, agents can then be associated with the control of a specific tool in the control system or to the control of a specific condition that must be assured to guarantee the correctness of the process.

In this context, we specifically consider a manufacturing pipeline in which items are transformed/augmented (e.g., a pipeline in which metal items are painted). Here, different agents can be devoted to the control of different stages of the pipeline (e.g., an agent is devoted to control the paint spraying, another is devoted to control the heat treatment of the paint, another of controlling the cooling process). Agents interact both indirectly through the environment and directly, through various forms of interaction protocol. In such an organisation, the role of each agent is that of "stage of the pipeline", in charge of ensuring that a specific portion of the pipeline works properly (e.g., that the oven maintains a constant temperature and that the cooling system does not cool items too fast). To this end, agents need to sense and effect that portion of the environment which represents the stage of the pipeline of which they are in charge. In addition, the agents need to interact to achieve a proper global functioning of the pipeline (for instance, by guaranteeing a uniform flux of items throughout the pipeline and by guaranteeing that the global flux of item does not exceed the "processing capabilities" of each of the stages).

*Conference Management:* As an example of an open system we will consider an agent-based system for supporting the management of an international conference. Setting up and running a conference is a multi-phase process involving several individuals and groups. During the submission phase, authors of submitted papers need to be informed that their papers have been received and they need to be assigned a submission number. Once the submission deadline has passed, the program committee (PC) has to handle the review of the papers; contacting potential referees and asking them to review a number of the papers. After a while, reviews are expected to come in and be used to decide about the acceptance/rejection of the submissions. Authors need to be notified of these decisions and, in case of acceptance, must be asked to produce the camera ready version of their revised papers. Finally, the publisher has to collect the camera ready versions from the authors and print the whole proceedings.

The conference management problem naturally leads to a conception of the whole system as a number of different organisations, one for each phase of the process. In each organisation, the corresponding MAS can be viewed as being made up of agents associated to the persons involved in the process (authors, PC Chair, PC Members, Reviewers). The roles played by each agent reflect the ones played by the associated person in the conference organisation. They may require agents to interact both directly with each other and indirectly, via an environment composed of papers and review forms. Since an agent is directly associated with a person, and its behaviour can be influenced by that person, opportunistic behaviour can emerge in the application. For example, an author could attempt to review their own paper or a PC Member could try to deal with fewer papers than they should. In addition, as the natural environment for the MAS is the Internet — due to the world-wide nature of the conference organisation — interactions with agents external to the MAS itself are likely to occur. For instance, a reviewer can decide to exploit its own personal agent to interact with the other agents of the organisation.

## 3 Organisational Abstractions

Organisational role models precisely describe all the roles that constitute the computational organisation; in terms of their functionalities, activities, and responsibilities, as well as in terms of their interaction protocols and patterns, which establish the position of each role in the organisation [32, 9, 7] . However, such role models cannot be considered as the sole organisational abstraction upon which to base the entire development process. Rather, before the design process actually defines the role model and, consequently, the whole organisation, a number of other steps need to be performed. Firstly, the analysis phase should identify *how* the organisation is expected to work. Secondly, the design phase should define *which* kind of organisation best fits the requirements identified in the analysis phase. Thirdly, it needs to be determined whether any re-use of available components can be exploited in some part of the organisational design. When taken together, this necessitates the introduction of three further organisational abstractions: organisational rules (section 3.1), organisational structures (section 3.2), and organisational patterns (section 3.3).

### 3.1 Organisational Rules

The analysis phase aims to collect all the specifications and requirements for building the MAS. To this end, it is possible to identify the basic skills (functionalities and competences) required by the organisation, as well as the basic interactions that are required for the exploitation of these skills. However, until the design phase has decided *which* organisation is most appropriate for the system, the identified skills and interactions cannot fully define the roles and the interaction protocols that will be played in the system (i.e., at defining a complete *role model*): this would imply an early commitment to a specific form of organisation. Instead, what the analysis phase can further identify — even in the absence of a complete role model — are the constraints that the actual organisation, once defined, will have to respect.

The implementation and/or execution of a computational organisation will have to respect a number of constraints, whose identification can either: *(i)* spread horizontally over all the roles and protocols (or, which is the same in this context, over the identified preliminary roles and protocols), or *(ii)* express relations and/or constraints between roles, protocols, or between roles and protocols. For example, in the case of human organisations: *(i)* social conventions define a set of implicit rules that moderate the interactions between all members (e.g., a clerk cannot contradict or ignore the commands of his manager), *(ii)* company specific conventions might impose constraints on how different roles have to be played in each of its organisations (e.g., a clerk cannot assume a role that would imply a member of the managing staff to be somehow subordinated to his clerk).

In both cases, such global constraints cannot easily be expressed in terms of individual roles or individual interaction protocols. Nevertheless, their identification is important for the correct development of the system and, therefore, they must be taken into account by the designer when actually defining the organisation of the system. To capture this type of information we use the concept of *organisational rules*.

The explicit identification of organisational rules is of particular importance in the context of open agent systems. With the arrival of new, previously unknown, and possibly self-interested agents, the overall organisation must somehow enforce its internal coherency despite the dynamic and untrustworthy environment. The identification of global organisational rules allows the system designer to explicitly define: *(i)* whether and when to allow newly arrived — possibly unknown — agents to enter the organisation, and, once accepted, what their position in the organisation should be; and *(ii)* which behaviours should be considered as an expression of self-interest, and which among them must be prevented by the organisation. In this context, organisational rules may also drive the designer towards the definition of the specific organisation that most eases the enforcement of the organisational rules and, for instance, can facilitate preventing undesirable behaviours of unknown and self-interested agents.

In the manufacturing pipeline example, all the different stages have to maintain the same speed of flow of items in the pipeline. This requirement can be more easily expressed in terms of a global organisational rule, rather than replicating it as a requirement for each and every role in the organisation. In the conference management system, there are a number of rules that drive the proper implementation of the organisation. As notable examples: an agent should be prevented from playing both the role of author and reviewer of the same paper and PC Members should not be in charge of collecting the reviews for their own papers. Neither of these constraints can easily be expressed in terms of properties/responsibilities associated to single roles and protocols. Instead, they represent global organisational rules.

### 3.2 Organisational Structures

A role model describes all the roles of an organisation and their positions in that organisation. Therefore, a role model also implicitly defines the *topology* of the interaction patterns and the *control regime* of the organisation's activities. That is, it defines the overall *organisational structure*. For example, a role model describing an organisation in terms of a "master role" and "slave roles" — where the former is in charge of assigning work to the latter and of load balancing their activities — implicitly defines an organisational structure based a hierarchical topology and on a load partitioning control regime. Other exemplar organisational structures include collectives of peers, multi-level and multi-divisional hierarchies [10], and they can all be modelled in term of a role model.

However, it is conceptually wrong to think of a role model as something that actually defines the organisational structure. Instead, in the design of a MAS, as well as in the design of any organisation, the role model should derive from the organisational structure that is explicitly chosen. Thus organisational structures should be viewed as first-class abstractions in the design of MASs.

The definition of the system's overall organisational structure can derive from the specifications collected during the analysis phase, as well as from other factors, related to efficiency, simplicity of application design, and organisational theory [10]. In any case, a methodology cannot start the analysis phase by attempting to define a complete role model that implicitly sets the organisational structure. Rather, the definition of the organisational structure is a design choice that should not be anticipated during the analysis phase. In fact:

- starting from the organisational structure — by pretending to know in advance what it should be or by committing a priori to a given organisational structure — may prevent subsequent optimization and change;
- although, in several cases, the organisational structure of a MAS is directly driven by its counterpart in the real-world system that the MAS is supposed to support, automate or monitor, this should not automatically imply that the organisation of the software system should mimic that of the real counterpart. Instead, the MAS may be better adopting a different organisational choice. There are several reasons why this could happen:
  - the real world organisation may not be well structured and the analysis phase could highlight several shortcomings;
  - the software, in itself, may change the way of working. Thus, the mere presence of the software introduces changes in the real organisation and these changes need to be reflected in the MAS;
  - the efficiency issues that may have driven a human organisation towards the adoption of a particular organisational structure may not necessarily apply to the agent organisation.
- the organisation, once defined, has to respect the organisational rules. Starting from a pre-defined organisational structure can make it difficult to have the organisational rules respected and enforced by the organisation. Instead, the choice of the organisation has to follow the identification of the organisational rules and have to be possibly driven by them.

In the manufacturing pipeline example, the most natural choice is to have an organisational structure in which all of the stages in the pipeline are peers, and in which they directly interact with their neighbours as needed. For instance, with reference to Figure 2, the stages Stage1, Stage2, Stage3, and Stage4 are controlled by agents R1, R2, R3 and R4, respectively, and each of these agents directly interacts with its neighbours. This closely mimics the structure of the real-world pipeline. However, this is not the only possible choice. Moreover, it may not necessarily be the best one. For instance, due to the real-time nature of the pipeline control problem, it may happen that

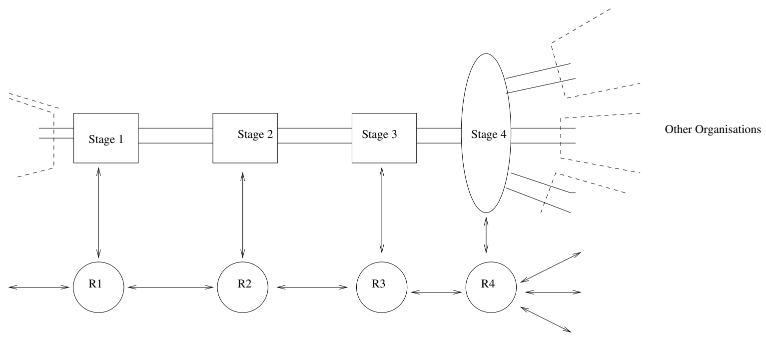


**Fig. 2.** A Manifacturing pipeline: pipeline organisation

a problem that requires global coordination between all the agents cannot be solved in due time, because of the high coordination costs associated with peer-based systems. In such cases, the designer can adopt a different organisational structure: for example, as sketched in Figure 3, it can introduce a global coordinator agent RC in charge of controlling and mediating the interactions for all the other agents, thus leading to a hierarchical organisation.

In the conference management example, the overall structure of the organisation can generally be derived from the structure the conference organisers have decided to adopt. However, it is often the case that the same conference varies its organisational structure from year to year, depending on both the size of the conference and the organisers' attitudes. For example, a small conference usually relies solely on the PC Members for the review process, and the PC Chair acts as a global coordinator, in a single-level hierarchy, for the work of the PC Member (see Figure 4). In contrast, a big conference usually has to involve external reviewers. This may require the PC Chair to partition the papers among the PC Members, and the PC Members to be in charge of seeking the appropriate number of reviews for their assigned partition. In other words, the organizational structure is a multi-level hierarchy based on a work partitioning control regime at the highest level (the one of the PC Chair) and on a global coordination control regime at the PC Member level (see Figure 5).

If the analysis phase commits the system to a specific organisational structure, the designer of the associated MAS will find it difficult to adapt the system, year after year, to the changing needs. For instance, it is very likely that a conference changes its dimensions in different editions and, consequently, its organizational structure. Thus, if the analysis phase simply describes the system's requirements, abstracting away from any specific organisational structure, the designer can reuse it to produce a new design according to the conference's new organisational structure.

*Organisational Relationships* The obvious means by which to specify an organisation is by the inter-agent relationships that exist within it. We emphasise that there is no universally accepted ontology of organisational relationships: different types of organisations make use of entirely different organisational concepts. For example, notions such as "command and control", which may be widely accepted in military organisa-

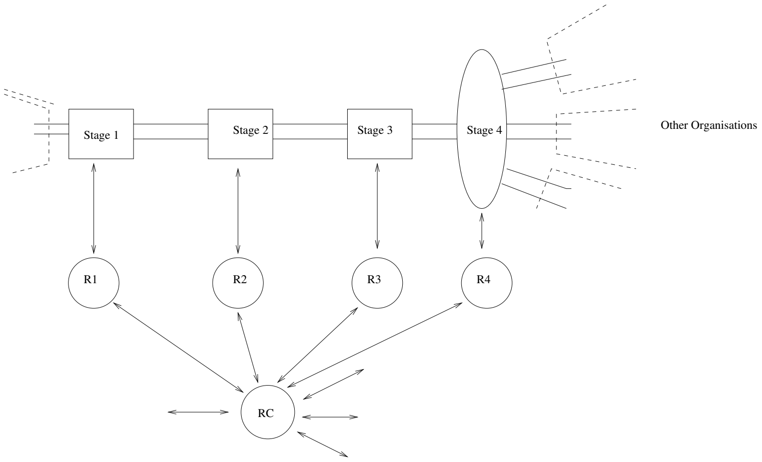


**Fig. 3.** A Manifacturing pipeline: hierarchical organisation

tions, tend not to be used in (most) academic organisations. Nevertheless, as a first pass towards more complete characterisations and formalisations, we can identify certain types of relationships that frequently occur in human and other organisations:

- *control* — which identify the authority structures within a system;
- *peer* — which identify agents of equal status;
- *benevolence* — which identify agents with shared interests;
- *dependency* — which identify the ways in which one agent may rely on another;
- *ownership* — which delimit organisational boundaries.

Note that these (binary) relationships exist between *roles* within a system — let $\mathcal{R}$ be the set of all such roles. In what follows, we give the intuition behind each type of relation. We then go on to give a precise formal definition of the semantics of these relationships.

Perhaps the paradigm example of an organisational relationship is that of one agent *controlling* another. Intuitively, if a role $r$ controls another role $r'$, then $r'$ will perform any service demanded of it by $r$. If $r$ controls $r'$, then as far as $r$ is concerned, the role $r'$ is a resource to be used as desired. Any control relationship $\mathcal{C} \subseteq \mathcal{R} \times \mathcal{R}$, must satisfy the following properties:

- (Reflexive): $(r, r) \in \mathcal{C}$, for all $r \in \mathcal{R}$.
  Any role controls itself.
- (Transitive): if $(r, r') \in \mathcal{C}$ and $(r', r'') \in \mathcal{C}$ then $(r, r'') \in \mathcal{C}$.
  If Ann controls Bob, and Bob controls Charles, then Ann controls Charles.
- (Anti-symmetric): if $(r, r') \in \mathcal{C}$, then $(r', r) \notin \mathcal{C}$.
  If Ann controls Bob, then Bob does not control Ann.

*Peer* relationships capture the notion of "equal status" within organisations. For example, consider two professors in the same university, but in different departments.

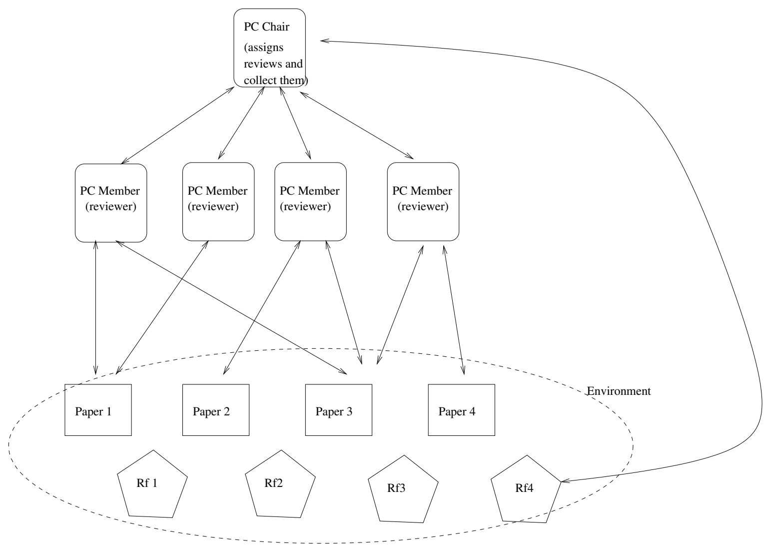


**Fig. 4.** Conference management: in a small conference, the PC Chair and assigns the reviews directly to PC Members and, possibly, to itself

These professors have equal status, even though they may not interact with one-another in the normal course of events. Status relationships have implications for how agents should interact with one-another. Any peer relationship $\mathcal{P} \subseteq \mathcal{R} \times \mathcal{R}$ must be an equivalence relation: it must be reflexive, symmetric, and transitive.

*Benevolence* is the classic assumption made in research on distributed problem solving (DPS) [8]. Put simply, an agent $i$ is said to be benevolent to another agent $j$ if $i$ will offer its services to $j$ whenever it is able to do so. Note that this is not the same as control. If Ann is benevolent to Bob, then Ann is inclined to help Bob wherever possible, *except* where helping Bob would prevent one of her own goals being satisfied. Formally, a benevolence relation $\mathcal{B} \subseteq \mathcal{R} \times \mathcal{R}$ must be reflexive and symmetric. Note that a benevolence relation is not (necessarily) transitive. Thus it is entirely possible for $r$ to be benevolent to $r'$, and for $r'$ to be benevolent to $r''$, without $r$ being benevolent to $r''$. To see why this is the case, consider for example benevolence relations between countries: it is entirely possible for the USA to be benevolent to (for example) Switzerland, and for Switzerland to be benevolent to Ruritania, without the USA being benevolent to Ruritania. (Situations like this are common in international relations!)

*Dependency Relationships* exist between agents primarily because of resource restrictions. For example, Ann controls some resource, (for example a piece of information), and Bob requires this information to satisfy one of his goals, then Bob is dependent on Ann. There are in fact many sub-classes of dependence relation that may exist between agents (see, e.g., [27]). For example, Ann and Bob may be mutually dependent on one-another; Bob may be dependent on Ann but Ann does not know it, or he may be dependent on Ann but he does not know it, and so on. Dependency relations are reflexive and transitive, but need not be symmetric.

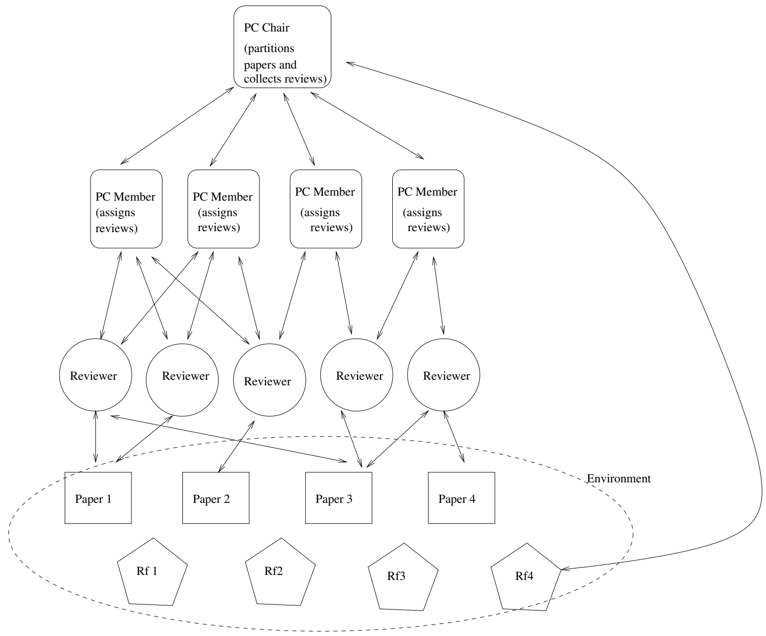


**Fig. 5.** Conference management: in a big conference, the PC Chair partitions the papers among the PC Members that, in their turns, are in charge of finding the appropriate referees for their assigned papers and of collecting the reviews

Finally, turning to ownership relations, the idea is to delimit boundaries of common ownership — thus all the agents belonging to organisation $o$ are grouped together, as are all the agents belonging to $o'$, and so on. Every agent is required to be the member of at least one ownership group, which may of course be a singleton set. Formally, any ownership relation $\mathcal{O} \subseteq \mathcal{R} \times \mathcal{R}$ must be an equivalence relation.

### 3.3 Organisational Patterns

There are numerous potential organisational structures, both in terms of topology of the interactions and control regimes [10]. However, we believe that a (comparatively) small subset of these structures are likely to be used most of the time. Thus, only rarely will peculiar structures be adopted (typically when the organisation has a very specific and unusual set of requirements).

Any methodology that encourages re-use of pre-defined components and architectures will ease and speed-up the work of both designers and developers. Object-oriented technology has recognised this need and increased the potential for re-use via design-patterns [11]. In this case, the most widely-used patterns of composition and interaction of object-oriented systems have been catalogued, and precisely described in terms of extent of applicability, sample implementation, and use cases. A software designer can then rely on these catalogues, and build applications by composing and re-using not only single objects, but whole pieces of the software architecture.

In the area of agent-based systems, we envisage something similar with respect to the most widely used organisational structures. Thus with the availability of catalogues of *organisational patterns*, designers can recognise in their MASs the presence of known patterns, and re-use definitions from the catalogue. In addition, designers can also be guided by the catalogue in the choice of the most appropriate organisational patterns for their MAS. Of course, for patterns to be properly exploited, the organisational structure must have been explicitly identified in the design phase.

In the pipeline example, the pipeline organisation between agents expresses an organisational pattern that is likely to re-appear in many applications (and which is already widely exploited as an architectural patterns in traditional software systems). The same can also be said of the hierarchical pipeline structure. In both cases, if a catalogue of patterns was available, the designer could rely on it to help define the system structure.

In the conference management example, the various organisational structures that conferences of different sizes tend to adopt are all fairly typical: from single hierarchies, to multi-level and divisional ones. Therefore, also in this case, it is expected that a methodology that makes explicit use of organisational patterns would ease the application design.

It is worth mentioning that several attempts to analyse and catalogue organisational agent patterns currently exist [28, 16, 17]. However, in most cases, this work abstracts away from any specific methodology for MAS analysis and design which should encourage and facilitate the re-use of these patterns. This, in turn, makes re-use more difficult.

## 4 Towards an Organisation-Oriented Methodology

The exploitation of the organisational abstractions we have introduced naturally promotes an organisation-oriented methodology for the analysis and design of MASs.

The analysis phase is tasked with collecting all the specifications from which the design of the computational organisation can start. This includes the identification of:

- the overall goals of the organisation and its expected global behaviour;
- the basic skills required by the organisation and the basic interactions required for the exploitation of these skills (that is, a preliminary role model);
- the rules that the organisation should respect and/or enforce in its global behaviour.

The output of the analysis phase should therefore be a triple: $\langle PR, PP, OL \rangle$, where *PR* are the preliminary roles of the system (derived from the identification of the basic skills), *PP* are the preliminary protocols (which have already been discovered to be necessary for the preliminary roles), and *OL* are the organisational rules. It is worth noting that the analysis phase should not committ to any specific organisational structure. Instead, its output should be (and be expressed in terms) independent of any specific organisational structure.

The design phase builds on the output of the analysis phase and produces a complete specification of the MAS. To this end, design can be decomposed into the following phases:

- definition of the organisational structure; by choosing the topology and the control regime. This involves considering: *(i)* the overall organisational efficiency, *(ii)* the need to respect and enforce the organisational rules, and *(iii)* the corresponding (if any) real-world organisation;
- completion of the preliminary role model; based upon the adopted organisational structure, and by keeping the organisational-independent aspects (detected from the analysis phase) and the organisational-dependent ones (deriving from the adoption of a specific organisational structure and from the insertion of roles and protocols in it) as separate as possible;
- exploitation of well-known organisational patterns on the basis of the system's identified organisational structure.

As in the Gaia methodology, we view the output of the design phase as a specification that can be picked up by using a traditional method (such as object orientation or component-ware) or that could be implemented using an appropriate agent-programming framework should one be available.

## 5 Related Work

Traditional analysis and design methodologies, such as object-oriented ones [2], are poorly suited to MASs because of the fundamental mismatch between the abstractions they provide [32]. Consequently, we believe that those efforts that attempt to simply extend object-oriented methodologies to MAS [18, 16] will inevitably fall short. Moreover, traditional compositional methods for object-oriented software architectures [24, 3] also have limited applicability in the definition of organisations for MASs. On the one hand, the defined interaction models are too static when compared to the dynamic interaction model defined by agents. On the other hand, the functionality-oriented modelling of the interactions between the system components clashes with the role-oriented perspective of MASs.

A number of agent-specific modelling techniques and development methodologies have been proposed in recent years (see [14] for a survey), several of which attempt to exploit the idea of a MAS as a computational organisation. In most of the cases, organisation-oriented systems and modelling techniques define an organisation as a collection of roles (i.e., a role model), without introducing any higher-level organisational abstractions. This is precisely what happens, for example, in the ALAADDIN system [9] where "the group structure" is simply the collection of roles that compose the organisation. Analogously, in the ToolKit approach [7], an organisation is defined simply by the set of roles that compose it and by the interaction protocols that have to occur between roles. Neither of these approaches incorporate the notions of organisational rules or organisational structures and, for the reasons we have outlined, will be limited in the range of agent systems they can deal with. In addition, these proposals do not attempt to define a complete and clear methodology for the development of agent system organisations.

Gaia starts from the organisational metaphor and defines a complete methodology [32] for the development of multi-agent systems. It also provides a clean separation between the analysis and design phases. However, it suffers from several limitations that are caused by the incompleteness of its organisational abstractions. The objective of Gaia's analysis phase is to define a fully elaborated role model, derived from the system specification, together with an accurate description of the protocols in which the roles will be involved. This implicitly assumes that the overall organisational structure is known a priori. However, as already stated, this is not always the case. In addition, by focusing exclusively on the role model, the analysis phase fails to identify any global organisational rules (making Gaia unsuitable for modeling open systems and for controling the behaviour of self-interested agents).

Similar shortcomings also effect most of the recently proposed organisation-oriented methodologies. For example, the MASE (Multi-Agent Systems Engineering) methodology [30] provides clean guidelines for developing multi-agent systems, based on a well-defined six-step process. This process drives developers from analysis to implementation. However, once again, the design process fails to identify any organisational abstraction other than the role model.

From a different perspective, some work in the area of coordination models and languages [12, 5] does explicitly address the problem of defining global rules ("coordination laws") to specify the behaviour and the interaction of agent ensembles. In this work, all interactions have to occur via specific "coordination media", whose internal behaviour can be programmed so as to implement specific policies for governing agent interactions. However, only recently have coordination models been recognised as useful abstractions upon which to define methodologies for the analysis and design of those systems. To achieve this, the coordination media are exploited as both the conceptual and physical repository of the organisational rules [6, 26, 25]. A somewhat similar approach has driven the implementation of the Fishmarket system for agent-mediated auctions [23]. In Fishmarket, the need to force agents to act in accordance with the "social conventions" that rule the organisation of an auction is recognised. To enact social conventions, the system dynamically associates a "controller agent" with each agent in the auction. Controller agents act as a coordination media, in charge of mediating all the interactions and of making agents respect the auction's conventions.

## 6 Conclusions and Future Work

This paper has discussed a number of issues related to the analysis and design of multi-agent systems. Specifically, we have considered the view of developing multi-agent systems as a process of constructing computational organisations. To date, the organisational concepts of agent roles and role models have become an important research area in the field of agent-based systems. However in this paper we have introduced three further organisational abstractions: organisational rules, organisational structures, and organisational patterns. These concepts, although neglected by the current methodologies for agent-oriented software engineering, are nevertheless of fundamental importance in multi-agent systems, and we therefore believe they should play a central role in any methodology. Having introduced and motivated these organisational abstractions, we sketched some general guidelines for a new methodology for the analysis and design of multi-agent systems that is centered around organisational abstractions.

Further work is needed to detail the proposed methodology, by:

- fully formalising the concepts of organisation rules and organisational structures. This can possibly be achieved by refining the formalism that we have already introduced in Subsection 3.2 with respect to the organizational structures.
- providing suitable notations for expressing the expected outputs of the analysis and design phases. We expect standard notations, such as UML, to be rapidly adapted to the needs of agent-based software engineering [1], as well as new agent-specific methodologies to emerge;
- identifying guidelines that assist the designer in the identification of suitable organisational structures for the system. Here analytical methods, experimental results, and case study experiences are likely to be helpful in supporting the choice.

For all of the above topics, we expect significant cross-fertilisation of models, formalisms and experiences from a number of different research areas. Among others, the research area of requirements engineering [22] can provide useful guidelines towards the identification and the modelling of organisational rules; the research results of both coordination, organizational and management sciences [20, 29, 21], which have widely studied the structures of human organisations and their most common patterns, are also expected to play a significant role.

## References


1. B. Bauer, J. P. Muller, and J. Odell. Agent uml: A formalism for specifying multiagent software systems, 2000. In this volume.
2. G. Booch. *Object-oriented Analysis and Design (second edition).* Addison Wesley, Reading (MA), 1994.
3. F.M.T. Brazier, B.M. Dunin-Keplicz, N.R. Jennings, and J. Treur. Desire: Modelling multi-agent systems in a compositional formal framework. *Journal of Cooperative Information Systems*, 6(1):67–94, 1997.
4. S. Bussmann. Agent-oriented programming of manifacturing control tasks. In *Proceeding of the 3rd International Conference on Multi-Agent Systems (ICMAS 98)*, pages 57–63. IEEE CS Press, June 1998.
5. P. Ciancarini. Coordination models and languages as software integrators. *ACM Computing Surveys*, 28(2), June 1996.
6. P. Ciancarini, A. Omicini, and F. Zambonelli. Multiagent systems engineering: the coordination viewpoint. In *Intelligents Agents VI (ATAL99)*, volume 1767 of *LNAI*, pages 250–259. Springer-Verlag, 2000.
7. Y. Demazeau and A. C. Rocha Costa. Populations and organizations in open multi-agent systems. In *1st National Symposium on Parallel and Distributed AI (PDAI'96)*. 1996.
8. E. H. Durfee. *Coordination of Distributed Problem Solvers*. Kluwer, 1988.
9. J. Ferber and O. Gutknecht. A meta-model for the analysis and design of organizations in multi-agent systems. In *Proceeding of the 3rd International Conference on Multi-Agent Systems (ICMAS 98)*. IEEE CS Press, June 1998.
10. M. S. Fox. An organizational view of distributed systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(1):70–80, January 1981.
11. E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns*. Addison Wesley, Reading (MA), 1995.

12. D. Gelernter and N. Carriero. Coordination languages and their significance. *Communications of the ACM*, 35(2):97–107, February 1992.
13. M. H. Huhns. Interaction-oriented programming, 2000. In this volume.
14. C. Iglesias, M. Garijo, and J. Gonzales. A survey of agent-oriented methodologies. In A. S. Rao J.P. Muller, M. P. Singh, editor, *Intelligents Agents IV (ATAL98)*, LNAI. Springer-Verlag, 1999.
15. N. R. Jennings. Agent-based computing: Promises and perils. In *International Joint Conference on Artificial Intelligence (IJCAI 99)*, pages 1429–1436, 1999.
16. E. A. Kendall. Role modelling for agent system analysis, design, and implementation. In *1st International Symposium on Agent Systems and Applications*. IEEE CS Press, October 1999.
17. E. A. Kendall. Agent software engineering with role modelling, 2000. In this volume.
18. D. Kinny and M. Georgeff. A methodology and modelling technique for systems of bdi agents. In *Workshop on Modelling Autonomous Agents in a Multi-Agent World, LNAI 1038*, pages 56–71. Springer-Verlag, 1996.
19. J. Lind. Issues in agent-oriented software engineering, 2000. In this volume.
20. T. W. Malone and K. Crowston. The interdisciplinary study of coordination. *ACM Computing Surveys*, 26(1):87–119, March 1994.
21. H. Mintzberg. *The Structuring of Organizations: A Synthesis of the Research*. Prentice Hall, Englewood Cliffs, N.J., 1979.
22. J. Mylopoulos, L. Chung, and B. Nixon. Representing and using nonfunctional requirements: A process-oriented approach. *IEEE Transactions on Software Engineering*, 18(6):483–497, June 1992.
23. P. Noriega. *Agent-mediated Auctions: The Fishmarket Metaphor*. Ph.D Thesis, Universitat Autonoma de Barcelona, Barcelona (E), 1997.
24. J. Odell, H. Van Dyke Parunak, and C. Bock. Representing agent interaction protocols in uml. In *OMG Document ad/99-12-01*. Intellicorp Inc., December 1999.
25. A. Omicini. Soda: Societies and infrastructures in the analysis and design of agent-based systems, 2000. In this volume.
26. A. Omicini and F. Zambonelli. Coordination for Internet application development. *Journal of Autonomous Agents and Multi-Agent Systems*, 2(3):251–269, 1999.
27. J. S. Sichman, R. Conte, C. Castelfranchi, and Y. Demazeau. A social reasoning mechanism based on dependence networks. In *Proceedings of ECAI94*, pages 188–192, Amsterdam, 1994.
28. Y. Tahara, A. Ohsuga, and S. Honiden. Agent system development based on agent patterns. In *International Conference on Software Engineering*, pages 356–367. ACM, 1999.
29. James D. Thompson. *Organizations in Action*. McGraw-Hill, New York, 1967.
30. M. Wood and S. A. DeLoach. An overview of the multiagent systems engineering methodology, 2000. In this volume.
31. M. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995.
32. M. Wooldridge, N. R. Jennings, and D. Kinny. The Gaia methodology for agent-oriented analysis and design. *Journal of Autonomous Agents and Multi-Agent Systems*, 3(3):285–312, 2000.
33. F. Zambonelli, N. Jennings, A. Omicini, and M. Wooldridge. Agent-oriented software engineering for internet applications. In A. Omicini, F. Zambonelli, M. Klusch, and R. Tolksdorf, editors, *Coordination of Internet Agents: Models, Technologies and Applications*. Springer-Verlag, 2000.