Dynamic Composition of Electronic Institutions for Teamwork

Mario Gómez[1] and Enric Plaza[2]

[1] Department of Computing Science, University of Aberdeen
mgomez@csd.abdn.ac.uk
[2] Artificial Intelligence Research Institute, Spanish National Research Council
enric@iiia.csic.es

Abstract. We present a framework for teamwork based on a requirement's driven dynamic composition approach to electronic institutions, which builds on an existing formalism for agent-mediated electronic institutions. In the presented framework, agent teams are designed and deployed on-the-fly so as to met the requirements of the problem at hand. The result is a new form of electronic institution that is created dynamically out of existing components to provide ad-hoc communication and coordination support for teamwork. This approach combines a requirements driven configuration of a team in terms of the structure, competencies and knowledge required (team design) to fulfill problem requirements; and a dynamic negotiation of the communication and coordination components to use for every team role (team formation).

1 Introduction

Cooperative problem solving (CPS) is a form of social interaction in which a group of agents work together to achieve a common goal. Several models have been proposed to account for this form of interaction from different perspectives: distributed artificial intelligence, economics, philosophy, organization science and social sciences. From the artificial intelligence perspective there are two main approaches to cooperation: a micro-level –agent-centered– view, which is focused on the internal architecture or the decision-making model of individual agents, and a macro-level –social– view, which is focused on the societal and organizational aspects of cooperation.

Some of the most challenging issues faced by the MAS community are related to the creation of open MAS [17]. Closed systems are typically designed by one team for one homogeneous environment, while in open MAS the participants (both human and software agents) are unknown beforehand, may change over time and may be developed by different parties. Therefore, those infrastructures that adopt a social view on cooperation seem more appropriate that those adopting a micro-level view, for the former do not enforce a particular agent architecture.

Some aspects of complex system development become more difficult by adopting an agent-centered approach: since agents are autonomous, the patterns and the effects of their interactions are uncertain, and it is extremely difficult to predict the behavior of the overall system based on its constituent components, because of the strong possibility of emergent behavior [16]. These problems can be circumvented by restraining interactions and imposing preset organizational structures, which are characteristic of the social view.

The Agent-Mediated Electronic Institutions (EI) approach was proposed [19,23,8] to address the issues stated above (openness and predictability) by introducing a social control mechanism. However, a main issue arises when trying to use preset organizational structures to operationalize CPS: the need for different team structures to deal with different types of problem. The EI approach was originally intended to model static organizations of agents; therefore, at first glance it seems inadequate to use such an approach for dealing with flexible teamwork. In this paper we introduce a proposal that uses the EI formalism in a novel way: on-the-fly institutions created out of existing components that capture the communication and coordination aspects of teamwork. These institutions are created on demand, according to the requirements of each problem being solved, and are able to reconfigure themselves to deal with changes in the environment.

The paper is structured as follows: Section 2 reviews related work, Section 3 puts our institutional model of teamwork in context by introducing the framework this model is part of, Section 4 describes our proposal to model teamwork based on the EI formalism, and finally, Section 5 summarizes our contributions.

2 Related Work

The notion of Agent Mediated Electronic Institutions (EI) was first proposed in [19] taking fish-auctions as an inspiring metaphor. Since then, it has become a main research topic of several projects, which have further refined and formalized it, as for example in [23,8]. An Electronic Institution (EI) refers to a sort of "virtual place" that is designed to support and facilitate certain goals to the human and software agents concurring to that place by establishing explicit conventions. Since these goals are achieved by means of the interaction of agents, an EI provides the social mediation layer required by agents to achieve a successful interaction: interaction protocols, shared ontologies, communication languages and social behavior rules. Formalization of electronic institutions [11] underpins the use of structured design techniques and formal analysis, and facilitates development, composition and reuse.

Other early frameworks based on social and organization notions are:

– The *Civil Agent Societies* [5]: a framework for developing agent organizations which follows the metaphor of civil human societies based on social contracts, and is oriented towards marketplaces and B2B e-commerce. This framework uses the Contract Net interaction protocol, social norms, notary services and exception handling services.
– The organization of sociality presented in [21]: it is based on a conception of cognition, both at the individual and the collective level, examined in relation to contemporary organization theory.

– The organizational model presented in [6]: this model describes rules of behavior for individual agents using concepts from organization theory such as roles and norms.

Other models and frameworks can be found in the proceedings and post-proceedings of the COIN International Workshop Series on Coordination, Organizations, Institutions, and Norms [1,20]. The framework we propose here is based on the EI approach, and more specifically, we adopt the formalism described in [10,9] as a starting point to our own work. The main contribution of our proposal is the notion of dynamic institutions created on-the-fly by selecting and combining reusable institutional components on-demand, so as to meet stated problem requirements.

The are some related works around the ideas of dynamic organization and coordination:

– In [12] a decision making framework is proposed that enables agents to dynamically select the coordination mechanism that is most appropriate to their circumstances.
– In [26] the authors address some of the aspects that must be considered in order to incorporate norms in agents, and propose a set of strategies to be used by agents in norm-based systems and analyze.
– In [7] the authors discuss reorganization issues in agent societies: they present a classification of reorganization situations, based on the focus of the reorganization, the authority to modify the organization, and how reorganization decisions are taken. This work proposes requirements for agents to allow for the automatic adaptation to a reorganized system.
– The analysis of reorganization requirements has yielded a model for adjustable autonomy [24] as a way to achieve dynamic coordination. This research describes the relation between types of coordination and the autonomy of actors.
– The concept of adjustable autonomy is also explored in [4], in the context of mixed human-agent teams. This work proposes a policy-based capability for adjustable autonomy based on the multiple dimensions of the problem.

A commonality of the former works, which differentiates them from our own work, is the focus on individual agents: mechanisms for autonomous agents to select coordination mechanisms and adapt to the changing environment, while our focus in on the organization itself: how to select the best organization for accomplishing some specific goals.

Also relevant to our own approach is the work presented in [3,2]. This work takes the notions of self-organization and self-configuration from Autonomic Computing, and applies them to Agent Mediated Electronic Institutions, which brings about the notion of Autonomic Electronic Institutions (AEI). In particular, the authors are exploring the the use of Genetic Algorithms and Case Based Reasoning to modify some aspects of an electronic institution to better fulfill its goals as the environment changes. However, our approaches are quite different: one the one hand, the AEI approach addresses the adaptation of norms specified in a parametric way, by learning the parameters that bring about a better global behavior; on the other hand, we address the structure and configuration of the institution itself, in terms of its organizational structure and allowed communication protocols; more specifically, our approach is to configure and reconfigure a new institution by selecting and composing reusable components so as to satisfy stated problem requirements.

Another line of research that is related to our own line is described in [18]. In that paper, the authors use a notion of dynamic electronic institution (DEI) as a temporary organization of agents that is constituted, dissolved and reformed on-the-fly, and is able to adapt its norms dynamically, dynamically in relation to its present members (agents). There are several differences between former proposal and our own one: one the one hand, their approach is driven by the goals of the agents willing to form a coalition and deciding to adopt a common set of norms, while in our approach there exists a previous meta-institution that helps agents form new institutions for solving specific problems, by reusing existing components; on the other hand, the central element of their proposal is the adoption of norms, while our approach gives more importance to the communication and coordination aspects of teamwork, that we use as building blocks of the institution.

Nest section introduces the ORCAS framework, a multi-layered framework for cooperative MAS that embraces the institutional model discussed in this paper.

3 The ORCAS Framework

In this paper we present an institutional approach to CPS that is part of the ORCAS framework for developing and deploying cooperative MAS [13]. The main contributions of this framework are:

– An Agent Capability Description Language (ACDL) that supports all the activities required to cooperate in open environments, from the discovery and invocation of capabilities, to their composition and coordination.
– A model of CPS that is driven by the specification of requirements for every particular instance of a problem to be solved.
– An agent platform for developing and deploying cooperative MAS in open environments.

Figure 1 depicts the main elements of the ORCAS ACDL. A *capability* is able to accomplish some *task*, and may require specific domain knowledge fulfilling some properties or assumptions. These properties assumed for the domain knowledge are specified as *domain models*. There are two types of capability: *skill* and *task-decomposer*. Skills are primitive, non decomposable capabilities, while task-decomposers decompose a problem (a task) into more elementary problems (subtasks), so as to solve complex problems that primitive capabilities cannot accomplish alone. Any capability presents a *knowledge-level description* that specifies what the capability does from a functional view: input, output, preconditions, and postconditions. This functional description can be used by

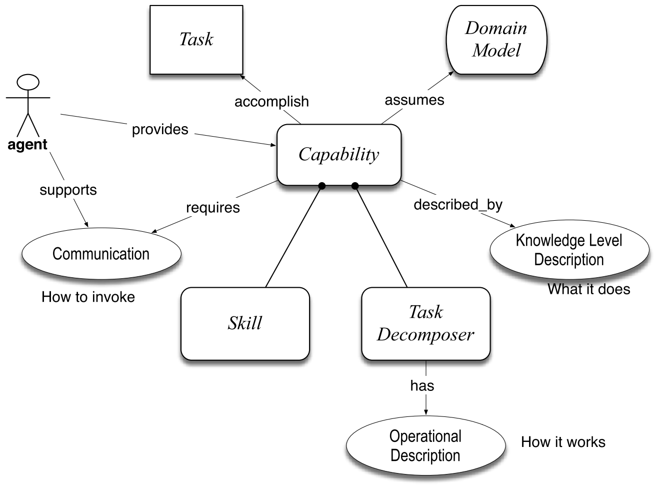

Fig. 1. Overview of the ORCAS ACDL

middle agents to discover and compose capabilities. However, in order to invoke a capability and interact with its provider, a requester agent must use an interaction protocol that is supported by the capability of interest. In ORCAS the information required to invoke a capability is referred to as the *communication* of a capability. Finally, the information required to coordinate multiple agents that are cooperating to solve a problem together is specified by the *operational description* of a task decomposer, which describes the control flow among subtasks (sequencing, parallelism, choices, etc.) in terms of agent roles.

The ORCAS platform provides all the infrastructure required by agents to successfully cooperate according to the ORCAS model of CPS. This model of the CPs process is sketched in Figure 2. The *problem specification* process produces a specification of problem requirements to be met by a team, including a description of the application domain (a collection of domain models) and the problem data to be used during teamwork. The *team design* process uses the problem requirements to build a *task-configuration*, which is a knowledge-level specification of: (1) the tasks to solve, (2) the capabilities to apply, and (3) the domain knowledge required by a team of agents in order to solve a given problem according to its specific requirements. The resulting task-configuration is used during *team formation* to allocate tasks and subtasks to agents, and to instruct agents on how to play their assigned team-roles: capabilities and knowledge to apply, as well as communication and coordination requirements. Finally, *teamwork* is the execution stage where team members try to solve the problem together by following the instructions received during team formation, thus complying with the specific requirements of the problem at hand.

Note that the ORCAS model for CPS should not be understood as a fixed sequence of steps, instead, we have implemented strategies that interleave team design and team formation with teamwork. These strategies enable the

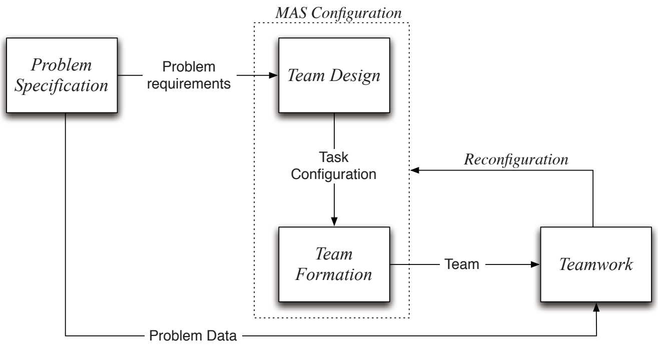

Fig. 2. The ORCAS model for the cooperative problem solving process

reconfiguration of agent teams dynamically so as to react to agent failure and other changes in the environment.

It should be remarked that, within the ORCAS framework, the EIs formalism is used in two ways: on the one hand, we use concepts adapted from the EI formalism described in [10,8] for specifying some elements of the ORCAS ACDL (the communication and the operational description), formalism; on the other hand, the ORCAS agent platform is itself an EI that provides mediation services for both providers and requesters of problem solving capabilities to successfully cooperate (this platform is actually a meta-instution where team-specific institutions are constituted). From now on, to avoid confusion we will sometimes refer to this formalism as ISLANDER, which is the name of a software tool to edit and verify institutions according to the formalism described in the EI formalism adopted here.

The knowledge-level description of a capability and the mechanisms used in ORCAS to discover and compose capabilities (which are part of the team design process) have been described elsewhere [14]. The ORCAS agent platform is described in [15]. In this paper we focus on those aspects of the ORCAS ACDL that are based on ISLANDER, namely the communication and the operational description, and how are these elements used to represent the interaction and coordination requirements of teamwork. These are the subjects of the following section.

4 Dynamic Institutions for Hierarchical Teamwork

The ORCAS ACDL specifies the communication and operational description of capabilities using elements from the ISLANDER formalism in a novel way, so it seems appropriate to briefly review the main concepts of this formalism before describing their use in ORCAS:

1. *Agent roles:* Agents are the players in an EI, interacting by the exchange of speech acts, whereas roles are standardized patterns of behavior required by agents playing part in given functional relationships.

2. *Dialogic framework:* Determines the valid illocutions that can be exchanged among agents, including the vocabulary (ontology) and the agent communication language.
3. *Scenes:* A scene defines an interaction protocol among a set of agent roles, using the illocutions allowed by a given dialogic framework.
4. *Performative structure:* A network of connected scenes that captures the relationships among scenes; a performative structure constrains the paths agents can traverse to move from one scene to another, depending on the roles they are playing.

In ORCAS the specification of capabilities at the knowledge level enables the automated discovery and composition of capabilities, without taking into account neither the communication aspects required to invoke a capability, nor the operational aspects required to coordinate the behavior of several agents. These features are specified respectively in the communication and operational description of a capability:

Communication: Specifies one or several interaction protocols that can be used to interact with an agent to invoke a given capability and get back the result of applying it. This feature is specified using the notion of *scene* from ISLANDER.

Operational Description: Specifies the control flow among the subtasks introduced by a task-decomposer, using a restricted version of the *performative structure* concept from ISLANDER.

A team in ORCAS is designed to solve a problem represented by a knowledge-level structure referred to as a *task-configuration* (the reader is referred to [14] for a more detailed description). Figure 3 shows an example of a task-configuration for a task called *Information-Search*. This task is decomposed into four tasks by the *Meta-search* task-decomposer: *Elaborate-query*, *Customize-query*, *Retrieve* and *Aggregate*, which is further decomposed by the *Aggregation* capability into two subtasks: *Elaborate-items* and *Aggregate-items*. The example includes some skills requiring domain knowledge: the *Query-expansion-with-thesaurus* requires a thesaurus (e.g. *MeSH*, a medical thesaurus), and the *Retrieval* and *Query-customization* skills require a description of information sources.

Any ORCAS team follows the hierarchical structure of a task-configuration, with one team-role per task. Each team role represents a position to be played in the team organization, and includes the following elements: a team-role identifier[1], the identifier of a task to be solved, the identifier of a capability to apply, the domain knowledge to be used by the selected capability (if needed), and optionally, if the capability is a task decomposer, the information required to delegate subtasks to other team-members, which includes, for each subtask: the identifier of a subordinated team-role, the team members assigned to that team

[1] The same task may appear multiple times in the same task-configuration, so a unique team-role identifier is required.

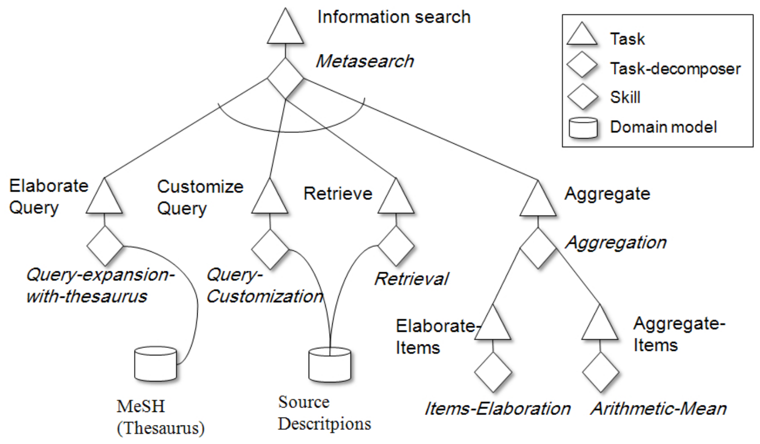

Fig. 3. Task-configuration example

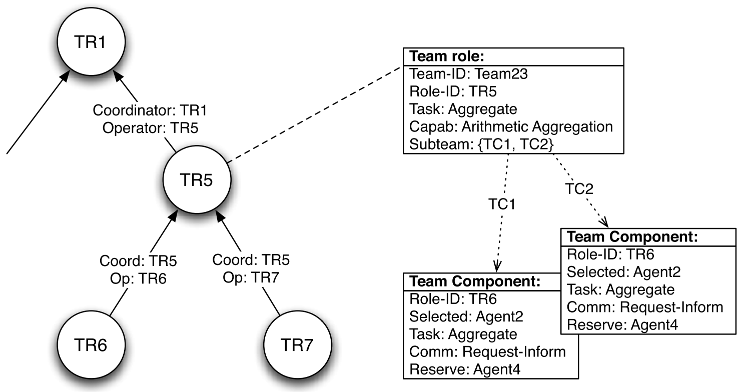

Fig. 4. Team roles

role[2], a collection of reserve agents to use in case that some of the selected team members fail, and a communication protocol that is compatible with the selected capability and shared by both the agent assigned to the parent task, and the agent or agents assigned to the subtask.

Figure 4 depicts a partial example of a team based on the task-configuration showed in Figure 3. Essentially, a team is hierarchical organization of team-roles. In particular, we see 4 team roles –TR1, TR5, TR6 and TR7– corresponding to tasks *Information-Search*, *Aggregate*, *Elaborate-Items*, and *Aggregate-Items*. A

[2] Usually, a task (and the corresponding team role) only needs a team member to be achieved, but some tasks may have to be performed multiple times in parallel, thus they can be served by a number of agents working simultaneously.

team is organized around subordination relations that are established in accordance to the top-down task-decomposition. These relations are specified in terms of two generic roles: the coordinator, which has to be adopted by an agent applying a task-decomposition; and operator, which has to be played by the agents selected to solve some subtask.

A *team-role* specifies the requirements for agents to play a specific position within a team, which includes: a task to be solved, a capability to be applied, and, if the capability is a task-decomposer, then the team-role can include information about team members selected for solving each subtask, the communication elements required to delegate each subtask to the selected agent, and optionally a group of agents to keep in reserve.

A **Team-Role** is a tuple $\pi = \langle R, I, T, C, M, Com, S, A_S, A_R\rangle$ where R is a unique team-role identifier, I is a unique team identifier, T is a task, C is a capability, M is a set of domain-models, Com is a specification of communication requirements, A_S is a set of selected agents, A_R is a set of reserve agents, and S is a subteam, specified as a set of team-components.

A *subteam* is specified as a set of *team-components*, where each team-component holds information about a team-role associated to one subtask. More formally, a **Team-Component** is defined as a tuple $\xi = \langle R, T, A_S, A_R, Com\rangle$ where R is a unique team-role identifier, T is a task A_S is a set of selected agents, A_R is a set of reserve agents, Com is a specification of communication requirements.

A team-component is defined for each subtask introduced by a task-decomposer. The *team-role identifier* (R) determines the precise position of the team-component in the team hierarchy. There is a set of agents selected (A_S) to carry out the team-role, and there is a set of agents to keep in reserve (A_R) for the case that some of the selected agents fail during the Teamwork process. Finally, a team-component includes a specification of the *communication* (Com) required to interact with the agent playing the team-component's team-role (R).

Figure 4 shown an example of a team-role that has to apply a task-decomposer, TR5. The agent selected to play TR5 has to apply the *Aggregation* task-decomposer, which introduces two subtasks: *Elaborate-Items* and *Aggregate-Items.* These subtasks are associated to subordinated team roles TR6 and TR7. The information required by TR5 to cooperate with the agents playing TR6 and TR7 is specified as team-components TC1 and TC2. For example, team component TC1 is associated to TR6, that is allocated to agent AG2, with agent AG4 in reserve, and the communication between TR5 and TR6 has to use a Request-Inform protocol.

We define a team as a structure made of interrelated team-roles and team-components, based on a *subordination relation* $\mathbb{S}$ among team-roles: a team-role is subordinated to another, denoted by $\mathbb{S}(\pi, \pi')$, if the first team-role is bound to a team-component contained in the subteam of the second team-role.

$\mathbb{S}(\pi, \pi') \Leftrightarrow \exists \xi^i \in \pi_S \mid \xi^i_R = \pi'$ where $\pi, \pi' \in \Pi$ are team-roles, $\pi_S \subseteq \Xi$ is the subteam of π (a set of team-components), $\xi^i \in \Xi$ is the i-th element of π_S, and $\xi^i_R \in \Pi$ is the team-role associated to ξ^i.

Noting $\mathbb{S}^*$ the closure of $\mathbb{S}$ we can now define a *team* as follows:

A **Team** is defined as a function of a particular a task-configuration $Team(Conf(\mathcal{K})) = \{\pi \in \Pi | \mathbb{S}^*(\pi^0, \pi) \wedge (head(\mathcal{K}) = \pi^0_T)\}$; where $\pi^0 \in \Pi$ is the team leader's team-role, which is not subordinated to any other team-role, $Conf(\mathcal{K})$ is a task-configuration, $head(\mathcal{K})$ is the root task of the task-configuration $(\mathcal{K})$, and π^0_T is the task allocated to the team leader π^0.

A *team* is a collection of interrelated team-roles, starting from the team-leader π, that is assigned to the root task of a task-configuration. This team model provides an abstract view of the competence required by a group of agents to solve a particular problem. Teams are instantiated during the Team Formation process by selecting a set of agents to play each team-role, and a set of agents to keep in reserve.

Next subsections address, respectively, the specification of the communication and operational description of a capability in ORCAS.

4.1 Communication

Agent capabilities should be specified independently of other agents in order to maximize their reuse and facilitate their specification by third party agent developers. In the general case, agent developers do not know a priori the tasks that could be achieved by a particular capability, neither the domains they could be applied to. As a consequence, the team roles an agent could play using a capability are not known in advance, thus the scenes used to specify the communication requirements of an agent over certain capability cannot be specified in terms of specific team-roles, but in terms of abstract, generic problem solving roles. Since ORCAS teams are designed in terms of a hierarchical decomposition of tasks into subtasks, teamwork is organized as a hierarchy of team-roles.

Some team-roles are bound to a task-decomposer, thus the agents playing those team-roles are responsible of delegating subtasks to other agents, receiving the results, and performing intermediate data processing between subtasks. In such an scenario, we establish an abstract communication model with two basic roles: *coordinator*, which is adopted by an agent willing to decompose a task into subtasks, and *operator*, which is adopted by the agent having to perform a task on demand, using the data provided by another agent that acts as coordinator of a top-level task

Figure 4 depicts some team roles, including the subordination relations that are established between roles, and the generic roles to be assigned when communicating between an agent applying a task-decomposer, and the agents playing the subordinated team-roles. For example, the agent playing TR5 will have to adopt the coordinator role to communicate with the agents playing TR6 and TR7, which will adopt the operator role. Each of these communications will follow the protocol decided during the Team Formation and specified in a team-component object.

Figure 5 shows a scene depicting the communication requirements of an agent over a capability by using a typical request-inform protocol in terms of our two generic roles: *Coordinator* and *Operator*. Symbol *?* denotes a new bind for a

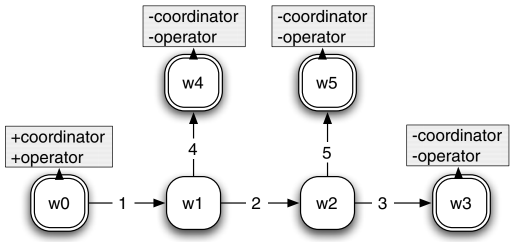

Fig. 5. Example of a communication scene

variable, while *!* denotes an already bound variable. States with double border line enable agents to either join (+) or leave (-) the scene at that point, according to the role they play. In the example, there is an initial state in w_0, where agents enter the scene, and three final states w_3, w_4 and w_5, where agents leave.

We adopt the formal definition of a scene in ISLANDER, so for the reader interested in the technical details, we refer to the papers describing that formalism, as for example [8]. Next section introduces our approach to specify the operational description of a task-decomposer.

4.2 Operational Description

The operational description of a task decomposer is used to specify the coordination among agents in terms of the role-flow policy and the control flow among subtasks. Figure 6 depicts some of the control flow constructions allowed by a performative structure: (a) tasks performed consecutively, in sequence; (b) choice between alternative courses of action; (c) tasks performed in parallel; and (d) tasks that can be executed multiple times.

In ORCAS the operational description of a task-decomposer is based on performative structures, with some distinctive features: as in the EIs formalism,

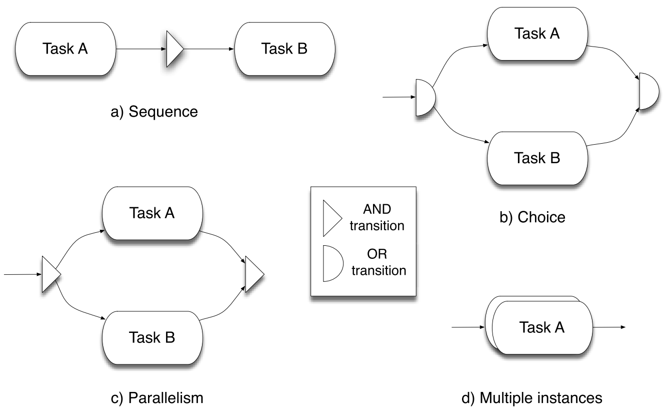

Fig. 6. Control flow among subtasks used in operational descriptions

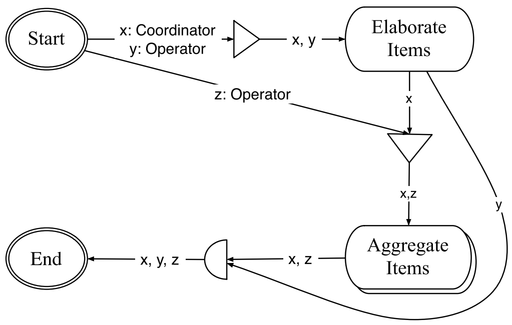

Fig. 7. Example of an operational description

each ORCAS scene within a performative structure must be instantiated by a communication protocol (except the *Start* and *End* scenes). However, in ORCAS the scenes within a performative structure are not instantiated beforehand; that is to say, they are not bound to a specific communication protocol. Instead, the scenes of an operational description are instantiated during team formation, using as a source the set of communication protocols shared by the agents having to interact.

After instantiation, each scene in an operational description corresponds to the communication required to solve a subtask, which implies an agent acting as coordinator invoking the capability provided by another agent acting as operator (or several operators in the case of multiple-instantiated tasks). The coordinator and the operators must use the same communication protocol in order to successfully communicate. Consequently, the instantiation of the scenes in an operational description is done using only those communication protocols shared by the agents involved in a scene. To note that team members are selected during team formation, and thus the set of shared communication protocols is not known until the team members are decided.

Figure 7 shows an example of an operational description for a task-decomposer called Aggregation. This task-decomposer introduces two subtasks: Elaborate-items (EI) and Aggregate-items (AI). Thus, the operational description has two main scenes, one for each subtask, and three role variables: x is a coordinator role, to be played by the agent applying the task-decomposer; y and z are both operator roles; y participates in EI, and z participates AI. Notice that the coordinator (x) is the same in both scenes; it enters EI first and moves to AI only after EI ends.

We adopt the formal definition of a performative structure in ISLANDER, so for the reader interested in the technical details, we refer to the papers describing that formalism, as for example [8].

Since each task-decomposer has an operational description, and the ORCAS organization of a team follows the hierarchical decomposition of tasks into

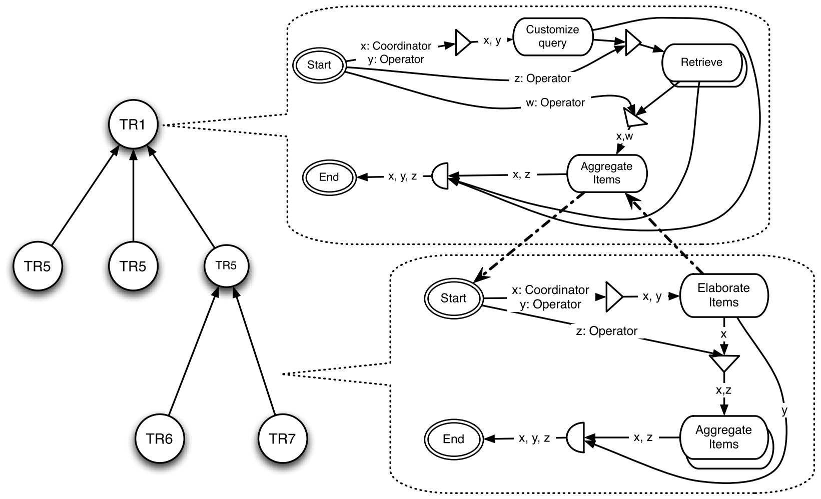

Fig. 8. Teamwork as a nested structure of operational descriptions

subtasks that results of applying task-decomposers, we can model the operational description of a complete team as nested structure of operational descriptions.

Figure 8 depicts the operational description of a team. The top team-role, TR1, is associated to task Information Search, and is bound to a task-decomposer that introduces three subtasks: Customize Query, Retrieve and Aggregate. Therefore, the agent playing TR1 will follow an operational description that contains three scenes, one for every subtask. In addition, the last of these subtasks is bound to another task-decomposer, Aggregation, which in turn introduces a new operational description. The new operational description is nested to the team leader's operational description, and has two scenes, one for Elaborate-Items and another for Aggregate-Items.

Teamwork follows the control flow and the communication scenes established by the nested structure of operational descriptions associated to task-decomposers (already instantiated during team formation). Each scene within an operational description refers to a communication protocol to be played by two agents, one applying a task-decomposer and playing the coordinator role, and one assigned to the corresponding subtask playing the operator role. When an agent playing an operator role has to apply itself a task-decomposer, it will follow the associated operational description playing itself the coordinator role. The execution of an operational description does not finish until all the nested operational descriptions are executed.

Each time a new team is formed according to a task-configuration, a new structure of nested operational descriptions is composed and their scenes instantiated. We regard this structure as a dynamic institution, since it is configured on-the-fly, out of the communication protocols and the operational descriptions supported by the selected team members.

5 Conclusions

In this paper, we have presented a novel approach to teamwork specification using concepts adapted from the EI formalism. In this approach the communication and coordination aspects required for teamwork are reusable components that are used by agents to specify their problem solving capabilities. By doing so, middle agents such as brokers and matchmakers can reason about the communication and coordination aspects of individual agents to dynamically create an EI that is adapted to the particular requirements of every problem to be solved.

While EIs are supposed to be static structures characterized by a predefined network of scenes (a performative structure), we conceive teamwork as a dynamic institution that is build on-the-fly out of existing components: operational descriptions and communication protocols. The operational description of a task-decomposer describes the control flow among subtasks using a specific kind of performative structure in which the communication scenes are not instantiated beforehand. The instantiation of these scenes is done at runtime by selecting communication protocols that are shared by the agents involved in every scene. The result is a hierarchical model of teamwork that is specified as a nested performative structure instantiated and composed on-the-demand, according to the requirements of each problem to be solved by a team of agents. This model supports also the reconfiguration of an institution at runtime, which allows teams to reorganize dynamically to better cope with changes in environments.

By adapting the EI formalism for teamwork, we aim at bringing in some of the benefits of the social-approach in general, and the benefits of the EI approach in particular: promoting openness by avoiding the imposition of a specific agent architecture and favoring reuse; increasing the degree of control over the global system behavior, thus making a MAS more predictable and fostering trustiness; and enabling formal verification tools and automated sofware-generation techniques (e.g. generation of agent-skeletons [25]).

Acknowledgements

This research was sponsored by the Spanish Council for Scientific Research under the MID-CBR (TIN 2006-15140-C03-01) project.

References

1. Boissier, O., Padget, J.A., Dignum, V., Lindemann, G., Matson, E., Ossowski, S., Sichman, J.S., Vázquez-Salceda, J. (eds.): ANIREM 2005 and OOOP 2005. LNCS (LNAI), vol. 3913. Springer, Heidelberg (2006)
2. Bou, E., López-Sánchez, M., Rodríguez-Aguilar, J.A.: Adaptation of autonomic electronic institutions through norms and institutional agents. In: O'Hare, G.M.P., Ricci, A., O'Grady, M.J., Dikenelli, O. (eds.) ESAW 2006. LNCS (LNAI), vol. 4457, pp. 300–319. Springer, Heidelberg (2007)

3. Bou, E., López-Sánchez, M., Rodríguez-Aguilar, J.A.: Towards self-configuration in autonomic electronic institutions. In: Noriega, P., Vázquez-Salceda, J., Boella, G., Boissier, O., Dignum, V., Fornara, N., Matson, E. (eds.) COIN 2006. LNCS (LNAI), vol. 4386, pp. 229–244. Springer, Heidelberg (2007)
4. Bradshaw, J.M., Jung, H., Kulkarni, S., Johnson, M., Feltovich, P., Allen, J., Bunch, L., Chambers, N., Galescu, L., Jeffers, R., Suri, N., Taysom, W., Uszok, A.: Kaa: Policy-based explorations of a richer model for adjustable autonomy. In: AAMAS 2005: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems, pp. 214–221. ACM Press, New York (2005)
5. Dellarocas, C.: Contractual Agent Societies: Negotiated shared connote and social control in open multi-agent systems. In: Proceedings of the Workshop on Norms and Institutions in Multi-Agent Systems, ICMAS 2002 (2000)
6. Dignum, V., Meyer, J.-J., Weigand, H., Dignum, F.: An organization-oriented model for agent societies. In: Proceedings of International Workshop on Regulated Agent-Based Social Systems: Theories and Applications (2002)
7. Dignum, V., Sonenberg, L., Dignum, F.: Towards dynamic reorganization of agent societies. In: Proceedings of CEAS: Workshop on Coordination in Emergent Agent Societies at ECAI 2004 (2004)
8. Esteva, M.: Electronic Institutions: From Specification to Development. Spanish National Research Council. IIIA Monographies, vol. 14 (2003)
9. Esteva, M., de la Cruz, D., Sierra, C.: Islander: an electronic institutions editor. In: Proceedings 1th International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 1045–1052 (2002)
10. Esteva, M., Padget, J., Sierra, C.: Formalizing a language for institutions and norms. In: Meyer, J.-J.C., Tambe, M. (eds.) ATAL 2001. LNCS, vol. 2333, pp. 348–366. Springer, Heidelberg (2002)
11. Esteva, M., Rodriguez, J.A., Sierra, C., Garcia, P., Arcos, J.L.: On the formal specifications of electronic institutions. In: Sierra, C., Dignum, F.P.M. (eds.) AgentLink 2000. LNCS (LNAI), vol. 1991, pp. 126–147. Springer, Heidelberg (2001)
12. Excelente-toledo, C.B., Jennings, N.R.: The dynamic selection of coordination mechanisms. Autonomous Agents and Multi-Agent Systems 9(1–2), 55–85 (2004)
13. Gómez, M.: Open, Reusable and Configurable Multi-Agent Systems: A Knowledge-Modelling Approach. Spanish National Research Council. IIIA Monographs CSIC, vol. 23 (2004)
14. Gómez, M., Plaza, E.: Extending matchmaking to maximize capability reuse. Proceedings of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems 1, 144–151 (2004)
15. Gómez, M., Plaza, E.: The ORCAS e-Institution: A Platform to Develop Open, Reusable and Configurable Multi-Agent Systems. International Journal on Intelligent Control and Systems. Special Issue on Distributed Intelligent Systems 12(2), 130–141 (2007)
16. Jennings, N.R.: On-agent-based software engineering. Artificial Intelligence 117, 227–296 (2000)
17. Klein, M.: The challenge: Enabling robust open multi-agent systems (2000)
18. Muntaner-Perich, E., de la Rosa, J.: Towards dynamic electronic institutions: From agent coalitions to agent institutions. In: Hinchey, M.G., Rago, P., Rash, J.L., Rouff, C.A., Sterritt, R., Truszkowski, W. (eds.) WRAC 2005. LNCS (LNAI), vol. 3825, pp. 109–121. Springer, Heidelberg (2006)
19. Noriega, P.: Agent-Mediated Auctions: The Fish-Market Metaphor. PhD thesis, Universitat Autònoma de Barcelona (1997)

20. Noriega, P., Vázquez-Salceda, J., Boella, G., Boissier, O., Dignum, V., Fornara, N., Matson, E. (eds.): COIN 2006. LNCS (LNAI), vol. 4386. Springer, Heidelberg (2007)
21. Panzarasa, P., Jennings, N.R.: The organisation of sociality: A manifesto for a new science of multiagent systems. In: Proceedings of the Tenth European Workshop on Multi-Agent Systems (2001)
22. Plaza, E.: Cooperative reuse for compositional cases in multi-agent systems. In: Muñoz-Ávila, H., Ricci, F. (eds.) ICCBR 2005. LNCS (LNAI), vol. 3620, pp. 382–396. Springer, Heidelberg (2005)
23. Rodríguez-Aguilar, J.A.: On the Design and Construction of Agent-mediated Electronic Institutions. IIIA Monographs, CSIC. vol. 14 (2001)
24. van der Vecht, B., Dignum, F., Meyer, J.-J.C., Neef, M.: A dynamic coordination mechanism using adjustable autonomy. In: Sichman, J.S., Padget, J., Ossowski, S., Noriega, P. (eds.) COIN 2007. LNCS(LNAI), vol. 4870, pp. 83–96. Springer, Heidelberg (2008)
25. Vasconcelos, W.W., Sabater, J., Sierra, C., Querol, J.: Skeleton-based agent development for electronic institutions. In: Proceedings UKMAS (2001)
26. y López, F.L., Luck, M., d'Inverno, M.: Constraining autonomy through norms. In: AAMAS 2002: Proceedings of the first international joint conference on Autonomous agents and multiagent systems, pp. 674–681. ACM Press, New York (2002)