Coordination Artifacts: A Unifying Abstraction for Engineering Environment-Mediated Coordination in MAS

Alessandro Ricci and Mirko Viroli
DEIS, Alma Mater Studiorum–Università di Bologna
47023 Cesena (FC), Italy
E-mail: {a.ricci,mirko.viroli}@unibo.it

Keywords: Multi-Agent System Coordination, Multi-Agent System Environment, Multi-Agent System Engineering, Coordination Artifacts

Received: May 10, 2005

Similarly to human organizations, where the environment plays a fundamental role in supporting social activities, the environment of a multi-agent system (MAS) is the natural place where understanding and designing agent coordination. Accordingly, we propose the notion of coordination artifact as a unifying abstraction for engineering environment-based coordination of agents. This is meant to capture at the MAS level abstractions and concepts like services, tools, and artifacts*, which are typically shared and exploited by the collectivity of individuals for achieving individual as well as global objectives. In this work we describe this framework, by defining a model for the coordination artifact abstraction, and discussing the infrastructures and technologies currently available for engineering MAS applications with coordination artifacts.*

Povzetek: Zajema enotno abstrakcijo inženirskega okolja v MAS z namenom koordinacije.

1 Introduction

Direct interaction and explicit communication are not always the best approaches to achieve coherent systemic behaviour in the context of MAS and agent societies. This is quite evident when taking into account the main approaches dealing with environment-based coordination such as stigmergy and, more generally, mediated interaction frameworks and infrastructures based on forms of coordination / cooperation without direct communication (see [43] for a recent survey).

Mediated interaction and environment-based coordination are highly debated also in other research fields outside MAS and CS, where collaborative and cooperative activities are studied in complex social contexts: notable examples are CSCW (Computer Supported Cooperative Work) and HCI Ê(Human Computer Interaction) [36], recently focussing on cognitive and social theories which explicitly take into account the role of environment in coordination, such as Distributed Cognition [15] and Activity Theory [19]. There, a relevant issue is to understand what makes an environment a good place for actors to work together: Ê

How to design the agent environment to suitably support the social activities of a possibly open agent society?

This question can be considered of primary importance also in MAS, and it involves issues that are not fully considered by current approaches dealing with coordination through the environment. In particular:

"Not only ants" | Approaches dealing with environment-based coordination typically consider *reactive* agents, either embedding all the intelligence into the environment or obtaining it as emergent phenomenon (well known examples are stigmergy coordination and swarm intelligence [28, 38, 3]). Here instead we are interested on the one side devising an environmental support that can be useful to amplify the intelligence of individual agents, possibly exploiting their cognitive capabilities. On the other side, we are interested in considering intelligence not only as an emergent phenomenon, but promoting the engineering of intelligence by designing and building suitable environmental abstractions.

"Not only special-purpose coordination" | Existing environment-based approaches to coordination — such as stigmergy — typically provide solutions only to specific coordination problems, without the abstraction required to use and systematise coordination in the wide range of social activities. Here instead we are interested in conceiving general purpose environment abstractions that could be suitably specialised and dynamically configured / tuned for addressing specific and heterogeneous coordination activities.

"Toward engineering" | Frequently, investigations in literature only concern simulation and abstract models (a notable exception can be found in [13], where a model for situated MAS is provided for the engineering of systems). Here we are interested instead in methodologies and infrastructures, i.e. in identifying models, languages, architectures and middleware

technologies to be exploited at the design stage in agent oriented software engineering, as well as for development and online management of MAS.

In this paper we describe the conceptual and engineering framework based on the notion of *coordination artifact*, which aims at addressing the above issues. The framework provides a systematic view of environment-based coordination for general coordination problems, and extends the scope of applicability to heterogeneous, *cognitive / intelligent* agents. Coordination artifacts are runtime abstractions encapsulating and providing coordination services, to be exploited by agents within a given social context. They can be exploited then as basic building blocks for designing and developing suitable working environments for heterogeneous multi-agent systems, supporting their coordination for collaboration or competition. Accordingly, coordination artifacts can be considered a kind of *first-order environmental abstractions or modules* as defined in the paper [42], part of this special issue.

We here gather the main results of our previous investigations [43, 24, 40], and provide a self-contained description of the role of coordination artifacts in the engineering of MAS environments. In particular, the remainder of the paper is organised as follows. Sect. 2 recalls the conceptual framework inspired by Activity Theory, as a background for the approach described in the paper, focussing on the importance of the environment in supporting social activities. Sect. 3 presents in detail the coordination artifact abstraction, along with its main properties and Sect. 4 remarks the impact of the framework on MAS engineering. Then, Sect. 5 discusses the framework as a unifying tool for understanding environment-based approaches in general, and in particular focuses on TuCSoN as a model / infrastructure / technology supporting the main features of the coordination artifact approach. Finally, related works are discussed in Sect. 6 and conclusions in Sect. 7.

2 Environment and Activity Theory

The environment support for both the analysis and the development of activities in complex systems — such as human society — is among the main issues studied by socio-psychological approaches such as Activity Theory (AT) and Distributed Cognition.

Activity Theory, defined also Cultural-Historical Activity Theory, is a social psychological theory initiated with the work of Lev Vygotsky (1926–62) in the context of Soviet Psychology (SP) [41]. From its origins, AT was furthered in the Soviet Union by Vygotsky's students — Alexey Leontiev in particular — in the first half of the 20th century. It then spread also outside the Soviet Union, first to Scandinavia and Germany and finally — at the end of the 1990s — to the United States. Nowadays it has been applied also in the context of computer science related fields, such as Computer Supported Cooperative Work (CSCW) and Human Computer Interaction (HCI) (see [19] for a survey).

AT is a very general framework for conceptualising human activities — how people learn and society evolves — based on the concept of human *activity* as the fundamental unit of analysis. The approach was developed in contrast to purely cognitive approaches which were dominating the first years of the 20th century: according to them, human individual and social activities could be analysed and understood focussing only on the internal (mentalistic) representation of the individuals, in other words on the individual information-processing capabilities. On the contrary, the basic inspiration principle of AT is the *principle of unity and inseparability of consciousness (human mind) and activity*: human mind comes to exist, develops, and can only be understood within the context of a meaningful, goal-oriented, and socially determined interaction between human beings and their material environment. From the beginning, a fundamental aspect for AT was the *interaction* between the individuals and the *environment* where they live, in other words their *context*. After an initial focus on the activity of the individuals, the AT research has evolved toward the study of human collective work and social activities, then facing issues such as the coordination and organisation of activities in human society.

Here the investigation of AT is of particular relevance because it remarks the fundamental role of the environment in the development of complex systems. According to AT any activity carried on by one or more components of a systems — individually or cooperatively — cannot be conceived or understood without considering the tools or *artifacts* mediating the actions and interactions of the components. Artifacts on the one side mediate the interaction between individual components and their environment (including the other components), on the other side embody the part of the environment that can be designed and controlled to support components' activities. Moreover, as an observable part of the environment, artifacts can be monitored along the development of the activities to evaluate overall system performance and keep track of system history. In other words, mediating artifacts become first-class entities for both the analysis and synthesis of individual as well as cooperative working activities inside complex systems.

The complexity of the activities of the social systems focussed by AT can be found nowadays in MAS and agent societies. With analogous consideration, we consider it fundamental to frame the role of the environment for the analysis and synthesis of social activities inside MAS, and in particular of the artifacts mediating such activities. In this work we describe the framework of *coordination artifacts* as an approach to systematise this vision and make it effective for the engineering of systems as MASs — from design to development and runtime, including their dynamic observation and management.

3 The Coordination Artifact Abstraction

Coordination artifacts can be conceived as persistent entities specialised in providing a coordination service in a MAS [33, 24]. The term coordination should be here understood in its most general sense, as the management of dependencies among separate activities [16], shaping and constraining the (agent) interaction space [5]. Coordination artifacts are *infrastructural* abstractions meant to improve coordination activities automation; they can be considered then as basic building blocks for creating effective shared collaborative working environments, alleviating the coordination burden for the involved agents. Human society is full of entities like coordination artifacts, engineered by humans in order to support and automate coordination activities: examples range from blackboards, maps, schedulers and paper trays, to traffic lights, clocks, and so on. These and other kinds of computerised artifacts are currently under investigation in the context of CSCW and cognitive sciences, where their importance in supporting human individual and cooperative activities is being recognised [37, 32].

Basically, a coordination artifact *(i)* entails a form of mediation among the agents using it, and *(ii)* effectively embeds and enact some coordination policy. Accordingly, two basic aims can be identified: *(i) constructive*, as an abstraction essential for creating and composing social activities, *(ii) normative*, as an abstraction essential for ruling social activities.

3.1 A First Model

Also taking inspiration from our society, a basic abstract model can be devised, where a coordination artifact features:

- a *usage interface*, defined in terms of a set of *operations*. Agents *use* coordination artifacts by executing operations provided by the artifact, and by eventually perceiving information about the operation completion. Notice that due to the nature of coordination artifacts and their interaction schema, agent actions executing operations are more similar to *practical acts* rather than *communicative acts* — which makes our approach sensibly different from direct, ACL-based interaction;
- a set of *operating instructions*. This information describes (formally) how to use the artifact in order to exploit its coordination service. For instance, operating instructions might specify the protocol of interactions to be used, and the mentalistic semantics of actions and perceptions [40];
- a *coordinating behaviour specification*. This information describes (formally) the coordinating behaviour of the artifact, in terms of coordination rules required for enacting the coordination service.

In particular, taking the agent viewpoint, to exploit a coordination artifact simply means to follow its operating instructions, on a step-by-step basis. It is worth noting that, since a considerable coordination burden can be charged upon the artifact and be hidden from the agents, operating instructions are generally quite simple when compared to the interactive behaviour required in the case of direct communication (protocols). Hence, our approach to interaction can be fruitfully leveraged by intelligent agents, which can exploit an artifact through its operating instructions so as to take part to complex coordination scenarios.

A simple but effective example of coordination artifact is a *task scheduler* in cooperative working environments, which can be found in concurrent systems as well as in human society. The coordination problem concerns ruling the order of execution of a dynamic set of tasks taken in charge by some agents, according to some scheduling policy. A coordination artifact can be designed to provide one such scheduling service. A possible usage interface would consist — for instance — in two basic operations[1]:

- *taskStart(-Token)*, to manifest agent intent to start executing the task. The completion of the operation means that the agent can start the task according to the scheduling policy of the artifact. A token is returned to the agent for identifying its activity;
- *taskCompleted(+Token)*, to signal the completion of the task.

Operating instructions simply consist in: first, invoking the *taskStart* operation to manifest the intention to start a task; then, invoking *taskCompleted* to signal the completion of the task. The coordinating behaviour of the artifact concerns the enactment of the scheduling policy, queueing requests and serving them according their position in the queue — for instance using a FIFO policy.

We conclude this section remarking both the philosophical / conceptual and engineering difference between agents and coordination artifacts. Agents are *goal-governed / goal-oriented* entities, and accordingly agent models / languages / architectures are suitable for defining pro-active and autonomous behaviour of agents. Coordination artifacts are *function-oriented* entities, i.e. entities designed to provide some kind of functionality or service. From this point of view, coordination artifacts are much more similar to *objects* as found in the object-oriented paradigm: differently from agents, they have a well-defined interface, providing operations that can be invoked by agents. On the contrary, agents do not provide interfaces with operations that can be invoked from external entities. More on this point can be found in [35, 39], where a generalisation of the notion of coordination artifact is introduced — the *artifact* abstraction —, representing any device populating agent working environments and that can provide functionalities other than coordination.

[1] The basic Prolog notation is adopted for describing argument of operations: + means an output argument, - an input argument, ? an input / output argument.

3.2 Basic Properties

Generally speaking, as devices exploited by agents to support their coordination activities, coordination artifacts have some basic properties which are indeed different from autonomy, pro-activeness, reactivity, and social ability which characterise instead the agent abstraction [44]:

Focus on interaction management | Coordination artifacts are specialised in automating coordination activities. For this purpose, they typically adopt a computational model suitable for effective and efficient interaction management, whose semantics can be easily expressed with concurrency frameworks such as process algebras [2] or Petri nets [31], see e.g. the work in [40].

Encapsulating coordination | Coordination artifacts encapsulate a coordination service, allowing user agents to abstract from how the service is implemented. As such, a coordination artifact is perceived as an individual entity, but actually it can be distributed on different nodes of the MAS infrastructure, depending on its specific model and implementation. Encapsulation is the key to achieve reuse of coordination. Agent society engineers can create and exploit handbooks or catalogs of coordination artifacts, embodying the solutions to general coordination problems in organisations, analogously to an handbook of handbook of organisation/coordination processes [17]. Also, a coordination artifact provides a certain *quality of coordination*, in particular in terms of the scalability with respect to the dimensions identified by Durfee in [11], which are related to performance, robustness, reliability, and so on. The description of such dimensions is important to identify the range of applicability of the artifact in the engineering of agent societies.

Malleability | Coordination artifacts are meant to support coordination in open agent systems, characterised by unpredictable events and dynamism. For this purpsose, their coordinating behaviour can be adapted and changed dynamically, either *(i)* by engineers (humans) willing to sustain the MAS behaviour, or *(ii)* by agents responsible for managing the coordination artifact, with the goal of flexibly facing possible coordination breakdowns or improving the coordination service provided.

Inspectability and controllability | A coordination artifact typically supports different levels of inspectability: *(i)* inspectability of its operating instructions and coordinating behaviour specification, in order to let user agents to be aware of how to use it or what coordination service it provides; *(ii)* inspectability of its dynamic state and coordinating behaviour, in order to support testing and diagnosing (debugging) stages for the engineers and agents responsible of its management. Controllability is also fundamental for runtime management of a coordination artifact, by making it possible to freeze its behaviour, to trace it, supporting step-by-step execution while watching its state, to restart it, and so on. So, from an operational point of view, a coordination artifact can be understood as a sort of *virtual machine of coordination*, executing some form of coordination specification, fully inspectable and controllable by coordination artifact administrators [21].

Linkability | This term is borrowed from coordinative artifacts studied in the context of CSCW [36]. It refers to the capability of linking artifacts together, in order to support a dynamic form of composition useful to scale with coordination activity complexity. This property is fundamental for supporting also the scenarios depicted in the paper [14] in this special issue, where multiple environments are considered and an environment can act as a medium for agents to interact with an other environment. Analogously, a coordination artifact can be used by an agent as a (coordination) medium to interact with other artifacts, which are linked to the first one.

Spatial extension | Differently from agents, coordination artifacts can have a spatial / topological extension, meaning that — in a MAS model with some topological structure — the same artifact can be present (simultaneously) in different nodes of the topology. In other words, while typically in a MAS a single agent cannot be distributed (it is located on a specific node of the MAS), on the contrary a single artifact can be distributed among different nodes of a MAS.

Summing up, coordination artifacts are conceived to be engineering abstractions used for designing, building and supporting at runtime coordination in agent societies, suitably instrumenting their dynamic working environment. Also, they can be useful to support forms of scientific investigation of collective behaviours. As mediating entities, coordination artifacts typically reify and manage agent communication events; accordingly, they can be used to trace and log the overall interaction behaviour of the agent societies exploiting them. Thus, they can act as kinds of *social memory*, which can then be inspected for possible scientific analysis about global behaviours.

4 Engineering Social Activities

The introduction of coordination artifacts impacts the methodology adopted for engineering social activities in agent societies. Taking inspiration from Activity Theory, we can identify three different stages characterising any social activities supported by coordination artifacts (see Fig. 1):

Co-construction | In this stage, engineers and scientists understand and reason about the social objectives of the society, and define a model of the social tasks

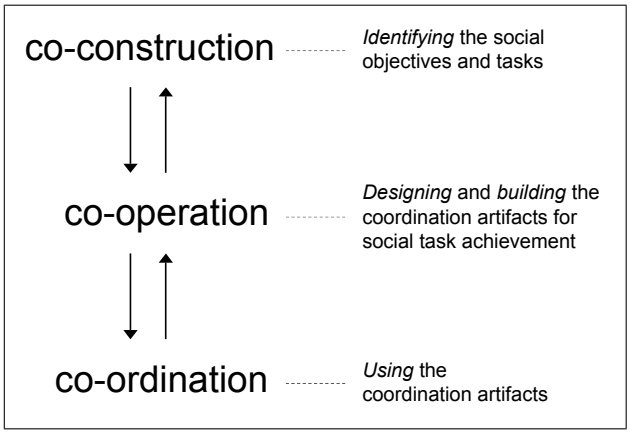


Figure 1: Levels of a social activities

required to achieve them. This implies understanding the shape of the agent interaction space, by possibly identifying also the dependencies that need to be managed (dependency detection is a fundamental aspect of coordination, according to the theory of coordination [16] and to cognitive theories of agent societies [6]).

Co-operation | In this stage, society engineers — and possibly intelligent agents — design and build the coordination artifacts according to the objective identified in the previous stage (co-construction). This implies understanding how to manage the dependencies previously identified, and defining a coordinating behaviour useful for that purpose. A model of coordination artifact must be chosen, according to its ability of embedding and enacting the required coordinating behaviour.

Co-ordination | In this stage, coordination artifacts are exploited, supporting the execution of the social activity. Here, the focus is on the efficient execution and automation of the coordination activities.

As in the case of AT, the three levels are distinct analytical moments that can be applied continuously, since a social activity is considered to be always under development, given the intrinsic openness of the environment and the dynamism of organisations.

4.1 Activity Levels as Engineering Stages

Activity Theory is primarily used as an analytical tool for understanding collaborative work in complex organisational contexts, and as a design tool to improve them. In such contexts, AT makes it possible to face the social complexity first by separating individual and collective activities, then by identifying and designing the artifacts required to support both of them.

Along this line, we can devise a correspondence between the three collaborative stages in Fig. 1, and the engineering stages as typically found in (agent-oriented) software engineering methodologies, i.e., analysis, design, development and deployment / runtime. Generally speaking, individual and social tasks are identified and described in the analysis and design stages of such methodologies. Each individual task is typically associated with one specific competence of the system. Each agent in the system is assigned to one or more individual tasks, and assumes full responsibility for their correct and timely completion. From an organisational perspective, this corresponds to assigning each agent a specific role in the organisation. Conversely, social tasks represent the global responsibilities of the agent system. In order to carry out such tasks, several possibly heterogeneous competences usually need to be combined. The design of social tasks leads to the identification of global *social laws* that have to be respected / enforced by the society of agents, to enable the society itself to function properly and in accordance with the expected global behaviour.

Given this picture, it is possible to identify a correspondence between the analysis stage (where individual and social tasks are identified) and the co-construction level, where the social objectives of the activities are shaped. Then, the identification of the social laws required to achieve the social tasks can be seen as a first step in the co-operation level. This level roughly corresponds to the design and development stages of the engineering process: coordination artifacts are the abstractions which make it possible to design and develop social tasks. At the co-operation level such artifacts are designed and developed to embody and enact — as governing abstractions provided by the infrastructure — the social laws and norms previously identified. Finally, the deployment and runtime stages correspond to the co-ordination level, when the coordination artifacts are instantiated and exploited.

The dynamism among the levels, that are compared here to the engineering stages of a system, promote then a new approach in the engineering of MASs that we can call here *online engineering*: coordination artifacts can be redesigned, manipulated, tested, debugged, analysed dynamically, at runtime. In order to support the online engineering methodology two aspects are essential: first, working with abstractions featuring suitable properties such as inspectability, controllability and malleability, which are necessary for their online analysis and synthesis; second, designing and building infrastructures that support services enabling, access and exploitation (co-ordination stage), and tools for their inspection, control, adaptation (co-operation stage) — see Sect. 4.3.

4.2 The Organisation Perspective: Structuring the Working Environment

Coordination artifacts can be suitably used in a structured and ruled organisation. Coordination artifacts become the entities around which the social activities are built, inducing a natural form of organisation structuring and modelling. By abstracting from details, several independent collaborative and cooperative activities are carried over inside an organisation, each one charged upon a group of

agents and a suitable coordination artifact. The same coordination artifact can be used in different ways according to the *roles* of the agents: moreover, operating instructions can be in principle partitioned according to the role of the agent using the artifact.

Following the organisation perspective, coordination artifacts are the key to shape agent working environment, as *(i)* tools for pure coordination, and *(ii)* interfaces mediating and coordinating agent access to the resources and the services provided by the environment itself. As mediating interfaces, coordination artifacts can encapsulate the policies for resource management, involving the coordination of both the users and the resources or the providers of the services.

The two issues above point out the fundamental role of artifacts in the design and construction of an effective working environment, supporting agent activity toward the achievement of their individual and social tasks. This is particularly relevant in the context of cognitive theories applied for CSCW, such as Distributed Cognition [15]. In the design and construction of a good working environment for the organisation, the tension between subjective and objective approaches emerges again in terms of the dichotomy between *flexibility* — the capability of individuals to adapt to contingent situations — and *automation* — the capability of making the execution of activities fluid. On the one side, given the complexity and the openness of agent organisations, a working environment keeps evolving and requires flexibility in order to allow for supporting changes and adaptations. The lack of flexibility dramatically impacts on all system activities. On the other side, a good working environment should assist workers as much as possible in their coordination, providing services to alleviate their coordination burden and let them focus on their individual work. The lack of system coordination typically makes organisations unable to govern the complexity of the activities: the final result is typically a weak control of activities, and poor performances in their execution.

4.3 Toward Infrastructures for Coordination Artifacts

Coordination artifact infrastructures (or middlewares) provide services for their access and use, effectively supporting the co-operation and co-ordination levels, and the reflection / reification transitions. Services range from artifacts creation and discovery to inspection and dynamic adaptation of their state and coordinating behaviour. Referring to the 3-Layer Model defined in [42], a coordination artifact infrastructure is part of the Execution Platform layer — at the Middleware level — while coordination artifacts themselves are specialised and used at the MAS Application Layer, programmed according to the application specific logic.

In the overall, coordination artifacts can be seen then as a fundamental abstraction for *governing* infrastructures [22], i.e. infrastructures providing flexible and robust abstractions to model and shape the agent interaction space, according to the social and normative objectives of systems.

Infrastructures also represent an effective approach to the general problem of formalisability of complex systems, which may come either for pragmatical or theoretical issues. By their very nature, infrastructures intrinsically encapsulate key portions of systems — often in charge of the critical system behaviour. In this case, governing infrastructures encapsulate agent interaction and coordination through coordination artifacts. As a result, providing well-specified infrastructures, and in particular formally-defined coordination artifacts promotes the discovery and proof of critical system properties. Most notably, a system property can be assessed at design-time through the formal definition of some design abstraction. Then, by ensuring compliance of the corresponding run-time abstraction provided by the infrastructure, such a property can be enforced at execution time and be automatically verified for any system based on the infrastructure.

5 A Unifying Abstraction for MAS Environment-based Coordination

The notion of coordination artifact can be considered as a unifying abstraction from different point of views. On the one side, one of the main roles of coordination artifacts is as engineering tools for directly designing and developing building blocks, specialised to provide coordination functionalities (the glue) — general-purpose enough to be suitably programmed and configured according to the specific coordination problems to be solved. On the other side, as the agent abstraction is meant to unify all the specific approaches dealing with autonomous, pro-active and goal-governed / oriented behaviour, the coordination artifact abstraction can be used to represent any first-class device supporting interaction through agents. Accordingly, any device could be described in terms of a coordination artifact with a specific usage interface, a coordinating behaviour and possibly some operating instructions.

Among the main example, the pheromone infrastructure in stigmergy-based coordination approaches ([29] for instance) can be described as a coordination artifact, providing as a usage interaface operations for deposit and sensing pheromones, and with a coordinating behaviour defined by the diffusion / aggregation / evaporation law of pheromones. The notion of coordination artifact can be used as a theoretical foundation to these approaches, identifying and generalising environmental entities which are not described as agents.

Another example is given by e-Institutions [12], that are middlewares where agent interaction is governed and ruled by norms imposed by the *Institution*, as an entity external to the agents. The *institution* can be modelled as a coordination artifact, with the coordinating behaviour specified by the norms ruling agent communication.

Coordination media introduced in the context of coordi-

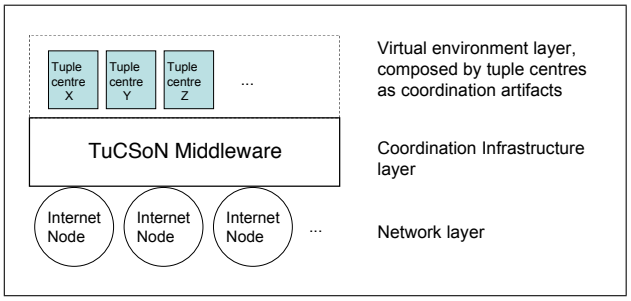


Figure 2: A logical view of TuCSoN infrastructure. The view is analogous to the 3-Layer model defined in [42]

nation models and languages (examples are tuple spaces, channels, etc. see [27] for a survey) can be described at the agent level as coordination artifacts: coordination primitives define the usage interface and the coordinating behaviour is defined by the coordination law defining the semantics of coordination media.

It is worth noting that these examples do not provide any explicit idea of operating instructions, which is instead a main property of coordination artifacts and fundamental for supporting intelligent agent coordination in open environment. This reveals an intrinsic inadequacy of existing approaches in filling the gap between agent rationality and environment-based coordination — see [40].

5.1 A concrete example: TuCSoN

As a concrete example of a model / infrastructure bringing some of the main principles that characterise the coordination artifact framework, here we consider the TuCSoN coordination infrastructure for MASs [26] [2].

The infrastructure enables agent interaction and coordination by means of *tuple centres*, which can be considered as a kind of coordination artifact. Technically, tuple centres are *programmable tuple spaces* — reactive, logic-based blackboards that agents associatively access by writing, reading, and consuming *tuples* (ordered collections of heterogeneous information chunks represented as first-order logic terms) via simple communication operations (*out*, *rd*, *in*, *inp*, *rdp*) [21]. While the behaviour of a tuple space in response to communication events is fixed, the behaviour of a tuple centre can be tailored to the application needs by defining a set of *specification tuples* expressed in the ReSpecT language, which define how a tuple centre should react to incoming / outgoing communication events. So, unlike tuple spaces, tuple centres can be programmed with reactions so as to encapsulate coordination laws directly in the coordination media. From the topology point of view, tuple centres are collected in infrastructure nodes, distributed over the network, organised into articulated domains (see Fig. 2 for a logical view).

So, tuple centres can be conceived as general-purpose coordination artifacts, which can be customised (programmed, tuned) dynamically to entail a specific coordinating behaviour. Generally speaking, tuple centres exhibit the properties that characterise coordination artifacts. First, they provide different levels of inspectability, since both the communication and the coordination state can be inspected at runtime. Second, different levels of malleability and controllability can be provided — both by allowing to dynamically change the artifact coordinating behaviour, and to control its execution by means of proper infrastructure tools [10]. The linkability property is supported by a primitive (*out_tc*) of the ReSpecT language, which makes it possible to directly insert a tuple from a tuple centre to another [34]. Also, we can identify the basic elements that characterise the abstract model of coordination artifacts: the usage interface is composed by the basic coordination primitives plus the primitives to inspect and change tuple centre behaviour (*set_spec* and *get_spec*). The coordinating behaviour specification is given by the ReSpecT specification. The notion of operating instructions is not directly supported in tuple centres, even if the ReSpecT specification tuples implicitly contain a description of how to exploit the tuple centre in order to obtain the coordinating service.

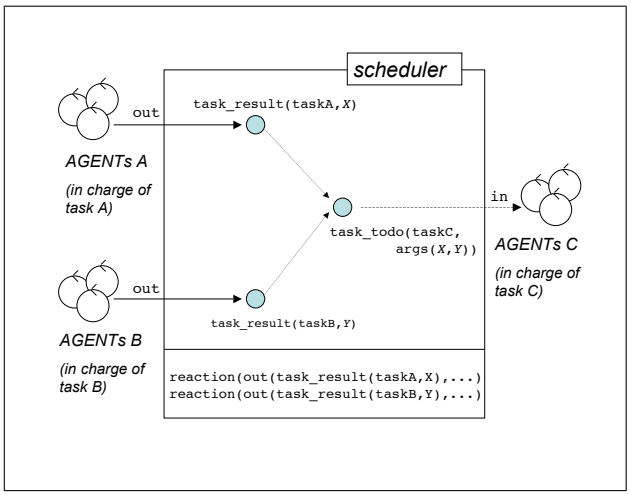
5.2 Coordination Artifacts in TuCSoN

Coordination artifacts can be considered as units of reuse for engineering cooperative working environments: as agents encapsulate skills and competences concerning the execution of some task, the achievement of some goal or the solution of some problem, coordination artifacts encapsulate strategies for constructing and ruling coordination activities. Tuple centre can be then suitably programmed to realise coordination artifacts with different coordinating purposes, such as flexible communication, knowledge mediation, resource sharing, and so on.

As a specific and representative example, here we consider workflow management, which is characterised by different kinds of coordination issues. Distributed workflow management concerns the automated integration and coordination of heterogeneous and independent distributed activities involved in the same global business process. Among the others, it includes activity scheduling and synchronisation, information and control flow management, exception management, and so on. In the context of service-oriented architectures — in particular Web Services — the workflow management idea is applied to the so-called *orchestration* [30].

Typically, special purpose languages — examples are XPDL and BPEL — can be used to define the workflow specification; their specification is executed by the *workflow engine*, the core component of a Workflow Management System. A workflow engine — also called orchestration engine — can be framed here as a general purpose coordination artifact, which is dynamically programmed to enact a coordinating behaviour according to the workflow specification.

[2] The TuCSoN technology is available as an open source project at the TuCSoN web site `http://tucson.sourceforge.net`




```
reaction(out(task_result(taskA,X)),(
  in_r(task_result(taskA,X)),in_r(task_result(taskB,Y)),
  out_r(task_todo(taskC,args(X,Y))) )).

reaction(out(task_result(taskB,Y)),(
  in_r(task_result(taskB,Y)),in_r(task_result(taskA,X)),
  out_r(task_todo(taskC,args(X,Y))) )).
```

Figure 3: Scheduler tuple centre *(top)* with its coordinating behaviour expressed in ReSpecT *(bottom)*

In the context of MASs, a tuple centre then can be programmed to provide the services from a simple task scheduler up to a full-fledged general purpose workflow engine. As an example, here we consider the realisation of a simple scheduler of three activities — A, B and C — coordinated according to a join pattern: task C can only start when both tasks A and B have been completed. Tasks are executed by independent agents, typically unaware of the global workflow and focussed on the achievement of their specific job. The tuple centre `scheduler` shown in Fig. 3 is an example of a coordination artifact providing such a scheduling service. The operation of the usage interface can be:

- `in(task_todo(+TaskName,-TaskInfo))`, for taking in charge the execution of a task. The presence of a tuple `task_todo` manifests the fact that a specific task has to be done, according to current workflow.
- `out(task_result(TaskName,TaskResult))`, for communicating the result of the execution of a task, signaling its completion.

In the example, *TaskName* can be `taskA`, `taskB` or `taskC`. The operating instructions of this coordination artifact, to be followed by agents in charge of task execution, would consist first in getting information about the task, then in providing the result. Fig. 3 shows also the ReSpecT specification realising the scheduling behaviour: basically, a suitable `task_todo` tuple is automatically generated in the tuple set as soon as the results of the execution of both tasks A and B are available.

6 Related Work

The coordination artifact abstraction brings in MAS ideas and concepts that have played a central role in other (un)related fields. From concurrent and distributed systems, coordination artifacts can be considered the generalisation of traditional coordination abstractions, from low level ones such as semaphores, monitors, to high-level ones, such as tuple spaces and, more generally, coordination media as found in coordination models and languages [27].

In particular, the notion of coordination artifact is strictly related to the *programmable coordination medium* abstraction defined in [9], on which the tuple centre model is based. According to the frequently adopted meta-model described in [4], a coordination model can be described by identifying the *coordinables* — the entities participating to coordination activities —, and the *coordination media* — the entities enabling and managing agent communication according to some coordination laws defining the semantics of the coordination activities. Programmable coordination media extend the basic notion of coordination medium with the idea of programming the internal behaviour with some specific language, so as to flexibly specify the coordination rules according to the application needs. So, programmable coordination media share some properties which characterise coordination artifacts, such as encapsulation of coordination and malleability of the behaviour. Instead, differently from programmable coordination media and coordination media in general, coordination artifacts do not necessarily manage *communications* among agents, but — more generally — *interactions*, caused by the execution of operations provided by the usage interface. Also, the coordination artifact framework introduces some structural properties — such as operating instructions — which are new with respect to the classic coordination meta-model, and which are indeed important in the context of open agent societies.

Blackboards as defined in Distributed Artificial Intelligence context can be framed and modelled in MAS as coordination artifacts, toward the integration of the two different points of view (traditional multi-agent and blackboard systems) in designing collaborating-software engineering space [7].

Actually, coordination artifacts can be exploited as an analytical tool for describing existing approaches based on some form of mediated / environment-based interaction. For instance, the environment provided by the pheromone infrastructure in [29] supporting stigmergy coordination can be interpreted as a coordination artifact exploited by ants to coordinate with each other: as such, it provides operations for depositing and sensing pheromones, and the coordinating behaviour is given by the environmental laws ruling the diffusion, aggregation and evaporation of pheromones. Analogously, the *field* abstraction in the co-field approach [18] — a recent approach for engineering of swarm intelligent systems — can be seen as a coordination

artifact, mediating mobile agents interaction and supporting their coordinated navigation inside some kind of space.

Also some coordination and organisation approaches developed in the context of intelligent / cognitive agents can be framed in terms of artifacts. A main example is is given by electronic institutions ([12] is an example), where agent societies live upon an infrastructure (middleware) which governs agent interaction according to the norms established for the specific organisation, representing both organisation and coordination rules. The institution then can be framed as a kind of shared coordination artifact, characterised by an interface with operations that agents use to communicate, and providing a normative function on the overall set of agents.

7 Conclusions

In the context of human activities and CSCW, Activity Theory and Distributed Cognition remark the importance of the environment — and in particular of the tools available in the environment — for governing the complexity of cooperative / social work, in particular for its analysis and construction. Analogously, the framework of coordination artifacts aims at providing an engineering key for instrumenting a MAS working environment with first-class abstractions which could help agents of a MAS to cooperate and coordinate. Such first-class abstractions are meant to be exploited in the various stages of the MAS engineering process: at the design stage, as modelling entities for designing social activities; at development and runtime stage, as runtime abstractions — supported by suitable infrastructures — to be used by agents to execute the social activities; and at runtime stage also for online engineering of systems, as inspectable, malleable abstractions which can be dynamically observed, controlled, adapted — by human as well as by intelligent agents — to support online debugging and evolution of the activities.

Recently, the coordination artifact concept has been generalised toward the notion of *artifact*, as first-class abstraction representing tools or objects (devices) that agents can either individually or collectively *use* to support their activities, and that can be designed to encapsulate and provide different kind of functionalities [35, 39, 23]: coordination artifacts can be framed then as artifacts designed to specifically provide coordination services. Artifacts are currently investigated as basic building blocks for programming MAS [35], engineering MAS environment [39], and — more generally — to re-frame the notion of intelligent agents as goal-oriented / driven users of artifacts [23]: as happen in the human case [20], artifacts can act not only as amplifiers of agent (human) capabilities, but as entities that can significantly change the nature of the tasks to be done, enhancing the overall performances.

In conclusion, the notion of (coordination) artifact and related conceptual / modelling / engineering frameworks seem to be one promising way to put the environment in-the-loop when modelling and engineering agent-based systems. Indeed, the work can be still considered in its infancy and many aspects need to be further explored and developed: from (formal) theories including artifacts in agent cognition and reasoning models, to models and languages for designing and developing artifacts, to full-fledged infrastructures supporting artifacts and related services at runtime (such as creation, discovery, management, etc.), possibly integrated with existing agent-based platforms. First investigations about the integration between artifacts and existing agent models / platforms can be found in [25] and in [35], which discuss the use of TuCSoN tuple centres in the context of JADE FIPA-compliant platform [1] and of 3APL agents [8], respectively.

References

[1] F. Bellifemine, A. Poggi, and G. Rimassa. JADE: a FIPA 2000 compliant agent development environment. In *AGENTS '01: Proceedings of the fifth international conference on Autonomous agents*, pages 216–217, New York, NY, USA, 2001. ACM Press.

[2] J. A. Bergstra, A. Ponse, and S. A. Smolka, editors. *Handbook of Process Algebra*. North-Holland, Amsterdam, The Netherlands, 2001.

[3] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, New York, NY, 1999.

[4] P. Ciancarini. Coordination models and languages as software integrators. *ACM Computing Surveys*, 28(2):300–302, June 1996.

[5] P. Ciancarini, A. Omicini, and F. Zambonelli. Multiagent system engineering: The coordination viewpoint. In N. R. Jennings and Y. Lespérance, editors, *Intelligent Agents VI. Agent Theories, Architectures, and Languages*, volume 1757 of *LNAI*, pages 250–259. Springer-Verlag, 2000.

[6] R. Conte and C. Castelfranchi. *Cognitive and Social Action*. University College London, London, UK, 1995.

[7] D. D. Corkill. Collaborating software: Blackboard and multi-agent systems & the future. In *Proceedings of the International Lisp Conference*, New York, NY, USA, 2003.

[8] M. Dastani, F. de Boer, F. Dignum, and J.-J. Meyer. Programming agent deliberation: an approach illustrated using the 3APL language. In J. S. Rosenschein, T. Sndholm, M. Wooldridge, and M. Yokoo, editors, *2nd international Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2003)*, pages 97–104, New York, USA, 14–18 July 2003. ACM Press.

[9] E. Denti, A. Natali, and A. Omicini. Programmable coordination media. In D. Garlan and D. Le Mé-

tayer, editors, *Coordination Languages and Models – Proceedings of the 2nd International Conference (COORDINATION'97)*, volume 1282 of *LNCS*, pages 274–288, Berlin (D), 1–3 Sept. 1997. Springer-Verlag.

[10] E. Denti, A. Omicini, and A. Ricci. Coordination tools for MAS development and deployment. *Applied Artificial Intelligence*, 16(9/10):721–752, Oct./Dec. 2002.

[11] E. H. Durfee. Scaling up agent coordination strategies. *IEEE Computer*, 34(7), July 2001.

[12] M. Esteva, B. Rosell, J. A. Rodríguez-Aguilar, and J. L. Arcos. Ameli: An agent-based middleware for electronic institutions. In N. R. Jennings, C. Sierra, L. Sonenberg, and M. Tambe, editors, *3rd international Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, volume 1, pages 236–243, New York, USA, 19–23 July 2004. ACM.

[13] J. Ferber and J.-P. Müller. Influences and reaction: a model of situated multiagent systems. In *2nd International Conference on Multi-Agent Systems (ICMAS'96)*, Kyoto, Japan, 1996.

[14] A. Gouaich and F. Michel. Towards a unified view of the environment(s) within multi-agent systems. In this issue.

[15] D. Kirsh. Distributed cognition, coordination and environment design. In *Proceedings of the European Conference on Cognitive Science (ECCS '99)*, pages 1–11, 1999.

[16] T. Malone and K. Crowston. The interdisciplinary study of coordination. *ACM Computing Surveys*, 26(1):87–119, 1994.

[17] T. W. Malone, K. Crowston, J. Lee, B. Pentland, C. Dellarocas, G. Wyner, J. Quimby, C. S. Osborn, A. Bernstein, G. Herman, M. Klein, and E. O'Donnell. Tools for inventing organizations: Toward a handbook of organizational processes. *Management Science*, 45(3):425–443, 1999.

[18] M. Mamei, F. Zambonelli, and L. Leonardi. Cofields: Towards a unifying approach to the engineering of swarm intelligent systems. In P. Petta, R. Tolksdorf, and F. Zambonelli, editors, *Engineering Societies in the Agents World III*, volume 2577 of *LNCS*, pages 68–81. Springer-Verlag, Apr. 2003.

[19] B. A. Nardi. *Context and Consciousness: Activity Theory and Human-Computer Interaction*. MIT Press, 1996.

[20] D. A. Norman. Cognitive artifacts. In J. Carroll, editor, *Designing Interaction: psychology at the human-computer interface*, pages 17–38. Cambridge University Press, New York, NY, USA, 1991.

[21] A. Omicini and E. Denti. From tuple spaces to tuple centres. *Science of Computer Programming*, 41(3):277–294, Nov. 2001.

[22] A. Omicini and S. Ossowski. Objective versus subjective coordination in the engineering of agent systems. In M. Klusch, S. Bergamaschi, P. Edwards, and P. Petta, editors, *Intelligent Information Agents: An AgentLink Perspective*, volume 2586 of *LNAI: State-of-the-Art Survey*, pages 179–202. Springer-Verlag, Mar. 2003.

[23] A. Omicini, A. Ricci, and M. Viroli. *Agens Faber*: Toward a theory of artefacts for MAS. *Electronic Notes in Theoretical Computer Sciences*, 2005. 1st International Workshop "Coordination and Organization" (CoOrg 2005), COORDINATION 2005, Namur, Belgium, 22 Apr. 2005. Post-proceedings.

[24] A. Omicini, A. Ricci, M. Viroli, C. Castelfranchi, and L. Tummolini. Coordination artifacts: Environment-based coordination for intelligent agents. In N. R. Jennings, C. Sierra, L. Sonenberg, and M. Tambe, editors, *3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, volume 1, pages 286–293, New York, USA, 19–23 July 2004. ACM.

[25] A. Omicini, A. Ricci, M. Viroli, M. Cioffi, and G. Rimassa. Multi-agent infrastructures for objective and subjective coordination. *Applied Artificial Intelligence*, 18(9/10):815–831, Oct./Dec. 2004.

[26] A. Omicini and F. Zambonelli. Coordination for Internet application development. *Autonomous Agents and Multi-Agent Systems*, 2(3):251–269, Sept. 1999.

[27] G. A. Papadopoulos and F. Arbab. Coordination models and languages. *Advances in Computers*, 46:329–400, 1998.

[28] H. V. D. Parunak, S. Brueckner, M. Fleischer, and J. Odell. A preliminary taxonomy of multi-agent interactions. In J. S. Rosenschein, T. Sndholm, M. Wooldridge, and M. Yokoo, editors, *2nd International Joint conference on Autonomous Agents and Multiagent Systems (AAMAS 2002)*, pages 1090–1091. ACM Press, 2003.

[29] H. V. D. Parunak, S. Brueckner, and J. Sauter. Digital pheromone mechanisms for coordination of unmanned vehicles. In C. Castelfranchi and W. L. Johnson, editors, *1st International Joint Conference on Autonomous Agents and Multiagent Systems AAMAS'02*, pages 449–450. ACM Press, 2002.

[30] C. Peltz. Web services orchestration and choreography. *IEEE Computer*, 36(10):46–52, Oct. 2003.

[31] J. L. Peterson. Petri nets. *ACM Computer Surveys*, 9(3):223–252, 1977.

[32] J. Rambusch, T. Susi, and T. Ziemke. Artefacts as mediators of distributed social cognition. In *Proceedings of the 26th Annual Meeting of the Cognitive Science Society*, Mahwah, NJ, 2004. Erlbaum.

[33] A. Ricci, A. Omicini, and E. Denti. Activity Theory as a framework for MAS coordination. In P. Petta,

R. Tolksdorf, and F. Zambonelli, editors, *Engineering Societies in the Agents World III*, volume 2577 of *LNCS*, pages 96–110. Springer-Verlag, Apr. 2003.

[34] A. Ricci, A. Omicini, and M. Viroli. Extending ReSpecT for multiple communication flows. In *The 2002 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'02)*, Las Vegas, USA, 24–27,June 2002.

[35] A. Ricci, M. Viroli, and A. Omicini. Programming MAS with artifacts. In *Workshop on Programming Languages for Multi-Agent Systems (PROMAS), 4th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'05)*, Utrecht, The Netherlands, 2005.

[36] K. Schmidt and C. Simone. Coordination mechanisms: Towards a conceptual foundation of CSCW systems design. *International Journal of Computer Supported Cooperative Work (CSCW)*, 5(2–3):155–200, 1996.

[37] K. Schmidt and I. Wagner. Ordering systems: Coordinative practices and artifacts in architectural design and planning. *International Journal of Computer Supported Cooperative Work (CSCW)*, 13(5–6):349–408, 2004.

[38] L. Steels. The artificial life roots of Artificial Intelligence. *Artificial Life Journal*, 1(1):89–125, 1994.

[39] M. Viroli, A. Omicini, and A. Ricci. Engineering MAS environment with artifacts. In D. Weyns, H. V. D. Parunak, and F. Michel, editors, *2nd International Workshop "Environments for Multi-Agent Systems" (E4MAS 2005)*, AAMAS 2005, Utrecht, The Netherlands, 26 July 2005.

[40] M. Viroli and A. Ricci. Instructions-based semantics of agent mediated interaction. In N. R. Jennings, C. Sierra, L. Sonenberg, and M. Tambe, editors, *3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, volume 1, pages 286–293, New York, USA, 19–23 July 2004. ACM.

[41] L. S. Vygotsky. *Mind and Society*. Harvard University Press, 1978.

[42] D. Weyns and T. Holvoet. On the role of the environment in multiagent systems. In this issue.

[43] D. Weyns, H. V. D. Parunak, and F. Michel, editors. *Environments for MultiAgent Systems*, volume 3374 of *LNAI*. Springer-Verlag, Feb. 2005. 1st International Workshop (E4MAS 2004), New York, NY, USA, 19 July 2004. Revised Selected Papers.

[44] M. J. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995.