12
Towards a Notion of Agent Coordination Context

Andrea Omicini

Abstract

On the one hand, recent studies on the history of human societies suggest that the role of the *environment* has to be taken explicitly into account in order to understand evolution of individuals and groups in any non-trivial setting. On the other hand, the notion of *context* is well-known and relevant to several research areas such as natural language, philosophy, logic, and artificial intelligence. In these areas, contexts are typically used to model the effect of the environment (in its most general sense, including the spatial and temporal interpretation of the term) on the communication occurring amongst active entities, such as humans or artificial agents.

Generalising upon the recently introduced notion of *context-dependent coordination*, in this seminal paper we propose the notion of *agent coordination context* as a means to model and shape the space of agent interaction and communication. From a theoretical perspective, agent coordination contexts can serve the purposes of (i) enabling agents to model the environment where they interact and communicate (the *subjective* viewpoint), and (ii) providing a framework to express how the environment affects interpretation of agent communication acts (the *objective* viewpoint). From an engineering perspective, the notion of agent coordination context enables in principle agents to perceive the space where they act and interact, reason about the effect of their actions and communications, and possibly affect their environment to achieve their goals. Also, agent coordination contexts allow engineers to encapsulate rules for governing applications built as agent systems, mediate the interactions amongst agents and the environment, and possibly affect them so as to change global application behaviour incrementally and dynamically.

12.1 EVOLUTION OF SOCIAL SYSTEMS

12.1.1 A Look Back to Human Societies

The complexity of artificial systems is growing day by day, and their dynamics in the era of the Internet and Ubiquitous Computing [36] is slowly but steadily approaching the dynamics of biological and social systems. So, the first (obvious, but maybe the most relevant) question to be answered is: Where are we heading to? To have some clues about where this rapid (r)evolution will take us as computer scientists and engineers of complex systems, it might be worthwhile to take a look back to the evolution of human societies. As a matter of fact, recent theories interpret biological systems as information-based systems evolving in complexity through a handful of fundamental transitions [18]. On the other hand, artificial, human-built systems are roughly taking a similar path, growing in complexity through quantum leaps – Ubiquitous Computing possibly being one of them. It is then easy to think that some portions of the history might repeat: that some of the striking forces driving biological and social evolution might take new forms and drive the evolution of artificial systems as well; that some of the results of biological and social evolution might find their counterparts in the entities and structures of forthcoming human-built systems.

Latest results from both biological [18] and historical sciences [14] tell us that, after millions years of biological evolution, cultural and *social evolution* has become predominant – where social evolution basically means organisational changes in the social structure. Also, the role of the *environment* is reckoned as a dynamic and active force that drives the evolution of human societies, and also affects the ability of both individuals and social structures to survive, grow, and possibly overcome the others. According to [14], the environment where individuals and societies live determines which kinds of individuals survive and which kinds of societies / organisations prevail, typically by making resources more or less available, or by imposing constraints, limitations, and (vital) needs. While in the short / medium term, a more or less successful interaction of individuals and societies with the environment determines their success or failure, in the long term the environment is to be seen as one of the main factors (if not the main one) determining the successful evolution of individual and societies in the global setting.

On the other hand, the environment may evolve over time, and its evolution is partially under human's control. Depending on their understanding of the environment and of its dynamics, individuals and societies might plan and act so as to drive environment changes over time, and possibly survive and grow according to their ability in modeling and building / modifying their own environment.

So, while physical laws and phenomena (as environment emerging dynamics) are out of the human reach, the ability to model the environment and its dynamics, along with some goal-oriented activity and planning capability, enables humans and societies to induce some controlled change making the environment fit more their needs and desires, thus partially driving the environment evolution.

So, some preliminary questions already call for an answer: Who plays individuals and societies in the artificial systems designed and built by humans – today and tomorrow? What is environment, in the era of the Internet and Ubiquitous Computing?

12.1.2 A Look Forward to Agent Societies

To face the ever-growing complexity of artificial systems, increasingly powerful abstractions have been introduced in computer science (and related areas), which have gradually taken the software engineering (SE) field, and become best practice in the years: structured programming languages, modules, objects, components. Today, research on multiagent systems (MAS) is emerging as a fertile ground for powerful and expressive abstractions and tools, making it possible to face the new challenges in terms of system complexity. Agents are autonomous, goal-oriented, pro-active entities, organized in societies, and situated within an environment: notions like *agent*, *role*, *organization*, *society*, *environment* are the basic bricks of what is called Agent-Oriented Software Engineering (AOSE) [9, 38]. Even though MAS might seem now far from becoming SE mainstream, agent technology is already somehow ubiquitous, and the continuous progress in agent research and systems seems to encourage us to think that MAS are the next step in the evolution of artificial systems.

There seems to be no apparent need to advocate any anthropomorphic principle to reason about agents and their role in the future of artificial systems. Analogy is enough to drive comparisons and possibly help us foreseeing the evolution of MAS. More than ten thousands years have seen humans as individuals autonomously acting in an unpredictable and possibly hostile environment, trying to achieve their own goals by interacting with other individuals – cooperating, competing, coordinating – according to complex interdependency patterns. Humans have evolved as situated, speaking, specialisable, and context-aware entities, co-existing and interacting within societies, deeply immersed within their environment, which they represent, learn from, and try to affect / change for their own purposes. Also, according to [14], (successful) societies built by humans are open to change, feature some form of "social learning", and react to environment pressure and changes. Their organization fits a precise scale, and does not scale up: e.g., it can be observed that peer-to-peer organisations scale up to a maximum of 80-100 members, then fail. So, social organization is forced to dynamically evolve over time, to face the challenges posed by its own growth, by other competing human groups, and by the environment as well.

In agent arena, the above considerations still basically apply: the above sentences roughly holds for humans as well as for agents, and for human societies as well as for MAS. Analogy is not so strong, instead, when we take into account the notion of environment. In fact, it is still true that only a limited part of the environment falls within the reach of the control of individuals (agents/humans) and societies, and that in order to partially control the environment evolution over time the ability to model the environment and its dynamics is required, along with some goal-oriented activity and planning capabilities. Also, in agent systems, too, there exist non-controllable forces (e.g., the market, determining the survival of a MAS application) and maybe hostile, evil entities (e.g., virus, ill-driven organisations), that make agent environment a partially unpredictable factor in its very essence. On the other hand, however, we as humans are by definition the designers and builders of artificial systems – agent ones, in particular. So, the main difference to be pointed out here is our role as humans with respect to human and agent societies. While human beings play the role of individuals within human societies, with limited control on their environment, humans might act as *gods* with respect to agent societies. There, part of the environment is still out of the individual (agent) control, but is mostly defined and built according to computer scientists' and engineers' intentions and specifications: what is out of the control of the agent could be within the reach of the human.

The need for suitably powerful abstractions for modeling and engineering the agent environment as a first-class entity is then clear. What might not be so clear is that representing the environment and its dynamics in agent systems (and in artificial systems in general) is a twofold problem, depending on which viewpoint we take: either the agent, or the human one. So, the following fundamental questions at least require an answer: Which abstractions are the most expressive and manageable for computer scientists and engineers to model, build, and control the environment for agent systems? And, which abstractions are the best suited for agents (in particular intelligent ones) to enable them to perceive, understand, and possibly affect their environment according to their needs, desires and goals?

12.2 CONTEXTS EVERYWHERE

Literally speaking, a *context* is what comes along (both in a spatial and temporal sense) with a "text", that is, a collection of signs or symbols, and possibly conveys further elements to interpret and give meaning to the text itself. In general, the notion of *context* is relevant and used within several research areas such as language, philosophy, logic, and artificial intelligence. In the field of natural language, *context-dependent interpretation* is a notion of outmost importance, encompassing the dependency of interpretation on both the linguistic and the non-linguistic environment where words and sentences occur. In the same field, *computational models of context* have been developed, for instance for natural language generation. Furthermore, several different notion of context have been defined over time in logic, computer science, and artificial intelligence. The collection [3] reports on contextual deontic logic, contextual learning, contextual interpretation of objects in a semantic network, and many other examples.

In the area of computer science, contexts have often be studied as a means to encapsulate knowledge (whether factual or procedural) or beliefs, and to combine them incrementally and dynamically to model many different notions related to knowledge, beliefs, and communication. For instance, in the programming language field, *contextual logic programming* have promoted the notion of context to a first-class entity, as the dynamically evolving set of the axioms determining the truth value of logic formulae [12].

Most recently, the issue of *context-aware computing* has become mainstream in the human-computer interaction (HCI) area, where it has been recognized that (i) there is a need for models, methods and tools to capture the setting where interaction between humans and computers occurs, (ii) the notion of context is the most obvious candidate to provide the required conceptual foundation, but (iii) the notion of context is still ill-defined [13] – or, at least, no consensus can be reached about what a context precisely is [37].

In general, contexts are typically used to model the effect of the environment – in its most general acceptation, including the spatial and temporal interpretations of the term – on the interaction and communication occurring amongst active (and typically intelligent) entities, such as humans or artificial agents. Also, contexts typically account for *situatedness* of the interaction and communication, coming to say that interaction and communication cannot be understood outside of the environment where they occur – out of their context, in other words. So, in spite of the many different acceptation of the term within the many heterogeneous research areas where it is commonly used, the notion of context is the most suitable candidate to be used as the first-class abstraction for modeling and engineering the environment in agent systems.

12.3 OBJECTS VS. AGENTS: THE ISSUE OF CONTROL

Since the notion of context occurs in so many different areas, and in computer science as well, one may argue that in a field like object-oriented languages and systems, for instance, the need for explicitly modeling the environment seemingly did not emerge so far, at least not so clearly. So, what are “object societies”, and why are they so different from agent societies, that a notion of “object context” has not been developed, yet?

Among the many possible answers, the most convincing one relates to the issue of control. In object-oriented (OO) systems, control flows along with data: as the reification of message passing, method invocation couples control and data, providing a single way to drive control along an OO system. Roughly speaking, design is *control-oriented* in OO systems, where control is outside objects, and the (human) designer works as a sort of “central control authority” – the *control god*, in other words.

If this simplifies system design at the small scale, it nevertheless makes design a non-trivial effort at the large scale, requiring engineering discipline to be effectively managed. The notion of object *interface* provides the form of discipline required, by constraining inter-object interaction, in terms of both control and data passing. Though the interface, control is passed to an object in a controlled way, and crosses object boundaries from outside in – when a method of the interface is invoked –, then from inside out – when the invoked method returns control to the caller. By describing the observable behaviour of objects, interfaces give discipline to object interaction and altogether provide the designer with a complete description of the interaction environment at design time. So, “social interaction” among objects is then limited and disciplined by interfaces, and governed by the human designer. In some sense, the environment surrounding an object in an application consists of all the other application objects, as described by their interfaces – the collection of the object interfaces could represent the context where the object lives and interacts. As a result, roughly speaking, while interfaces are enough for human designers to govern OO systems, objects are not powerful enough to require the development of explicit notions for object society and environment.

Instead, agent systems are not control-oriented, but *goal-oriented*. Agent systems have no central control authority (no control god), instead each agent is an independent *locus* of control: control is inside agents, and the agent goal drives the control – each agent works as its own control god. So, since data do not necessarily flow with control, control and data are uncoupled in agent systems. As any non-trivial form of decoupling, this makes system engineering simpler. In fact, the human designer is to some extent free of the burden of explicitly managing control in agent systems, and can in principle focus on designing agent system as information-oriented at the highest level of abstraction.

However, given that AOSE promotes agents as useful abstractions for modeling and engineering large complex systems [28, 27], the need for a disciplined environment for agent systems emerges clearly in the same way as in the case of OO systems.

This is even more true for applications in the Ubiquitous Computing scenario [36]. There, in fact, the huge number, diversity, and dynamics of the environments where agents would be called to (inter)act could give raise the same sort of complexity that already emerged in the robotics research – like the *frame problem*, or the *qualification problem*. That is, roughly speaking: Which actions is the agent actually allowed to perform at any place and time? Which effect on the surrounding environment will agent actions really have? Which knowledge about the environment is relevant for an agent to effectively plan and act? How and to what extent can an agent be ensured that its knowledge about the environment is at any time consistent with its actual state?

As a result, when we try to devise out an agent notion that could roughly correspond to object interface, we are led back to some agent-related notion of *context*, which could work as a limited yet effective representation of the agent environment, and also allow agent interaction to be properly disciplined. Also, the required engineering discipline essentially concerns agent interaction – within societies, and with the environment. So, since by definition *coordination* is managing [17] and constraining [35] interaction, it seems almost clear that the notion of context we need in complex agent systems is essentially that of an *agent coordination context*.

Even though it would play a role corresponding to object interfaces in object systems in some sense (that is, according to the line of reasoning developed in this section), an agent coordination context is obviously not an "agent interface" – no way. In its most general acceptation, an agent cannot be invoked: control is within agents, and does not cross the agent boundaries. An agent could indeed be designed as a service provider, but it is autonomous by definition: in principle, it would provide services on its own deliberation, and could then at any time refuse to do so – differently than objects, "agents can say no" [21].

However, object systems of today could tell us more about what an agent coordination context could be – more precisely, component-based systems could. In fact, component-based systems typically come along with a notion of *infrastructure*, providing *services* to application objects, like transparent access to distributed resources, directory services, name services. These infrastructures (like CORBA [22], DCOM [33], Jini [34]) embody some implicit definition of the environment for components and applications, by defining how components could enter a new environment, how could they identify themselves and the other participants, how could they locate, access or make available resources, and so on. Notions like interface description language and inspectable interface suggest us that an agent coordination context should be conceived not only as a tool for human designers, but also as a run-time abstraction provided as a service to agents by a suitable infrastructure.

Finally, the ever-growing capabilities of the abstractions in artificial systems is changing the roles they play. Expert systems are used in the diagnosis of errors and critical situations, objects are used to automatically balance loads in parallel systems, agents are used for network management, just to quote some. So, while we take as granted the meta-level roles that humans play in un artificial system (as designers, developers, deployers, maintainers, actors, final users), we often do not take into account as well that some of these roles – and perhaps in some future all of them – could be played by the components of an artificial system. This is for instance one of the concerns of the Semantic Web project [2], which basically considers software agents as alternative (to humans) users of Web pages. As a consequence, features of abstractions and tools for the engineering of artificial systems require more and more to be accessible to both humans and artificial system components: so, for instance, if we require that an agent coordination context be inspectable, this should mean that can be inspected by a human (say, the system manager) and by an intelligent agent as well.

12.4 AGENT COORDINATION CONTEXT

Generalizing upon the recently introduced notion of context-dependent coordination [6], this paper suggests the notion of *agent coordination context* as a means to model and shape environment in agent systems. As discussed in the previous section, this notion is expected to play in some sense the same role within agent systems as the notion of object interface in object systems: that is, providing a discipline for interaction, and some pattern for managing control. Since, roughly speaking, control is outside objects, but inside agents, an agent coordination context is indeed meant to provide a sort of "boundary description" as an object interface does – but from inside out (an agent), rather than from outside in (an object). As an interface, however, a coordination context provides decoupling between agents and their environment, so that the agent can in principle be designed and developed independently of other agents, resources, and services populating the MAS environment.

More precisely, a coordination context should

- work as a model for the agent environment, by describing the environment where an agent can interact, and
- enable and rule the interactions between the agent and the environment, by defining the space of the admissible agent interactions

As a tool to model the environment, an agent coordination context can serve a twofold purpose, according to the viewpoint we take. From the agent viewpoint (more generally called the *subjective* viewpoint [32]), an agent coordination context should provide agents with a suitable representation of the environment where they live, interact, and communicate. Such a representation should obviously include some notion of *locality*, in both space and time – since a global and synchronous environment is not realistic in today application scenarios –, implicitly defining the where and when of agent interactions. Obviously, a coordination context should contain all the information about the entities (agents, services, resources) an agent could interact with, along with a description of the admissible way of interaction. This obviously assumes that a *context representation language* is defined and used: it could be a first-order logic language, an XML-based ones, or whatever – until it is expressive enough to fully describe an agent environment and all the admissible agent interactions.

From the human designer viewpoint (more generally called the *objective* viewpoint [32]), an agent coordination context should provide a framework to express the interaction within a MAS as a whole. More precisely, coordination contexts define the space of MAS interaction, that is, the admissible interactions occurring among the agents of a MAS, and between the agents of a MAS and the MAS environment. In particular, as far as inter-agent communication is concerned, coordination contexts model how the environment affects interpretation of agent communication acts.

As a tool to enable and rule agent interaction with the environment, an agent coordination context can again serve a twofold purpose. From the agent viewpoint, the coordination context enables in principle agents to perceive the space where they act and interact, reason about the effect of their actions and communications, and possibly affect the environment to accomplish their own goals. From the viewpoint of a human engineer, coordination contexts would allow engineers to encapsulate rules for governing applications built as agent systems, mediate the interactions amongst agents and the environment, and possibly affect them so as to change global application behaviour incrementally and dynamically. Coordination contexts allow then a form of *prescriptive coordination* to be enforced, constraining from the design stage to the run time the space of agent interaction. By the way, it should be observed that only the capability to ensure that harmful behaviors could be prevented, and that only some admissible courses of action could be taken by agents, would make a host infrastructure provide open spaces where agents can act according to their own deliberation and will, free to accomplish their own goals, without constraining their model or behavior *a priori*. From this viewpoint, then, prescriptive coordination seems to be a sort pre-condition to agent autonomy, rather than an obstacle to it.

As observed in the previous section, this also calls for a suitable run-time support, embodying coordination contexts as run-time abstractions provided by the agent infrastructure. Even more, by their very nature, agent coordination contexts could be used to mediate between the needs of a MAS application and its host, by actually modeling agent infrastructures (their structure and organization, the resources and services they make available to agents) in terms of coordination contexts. When suitably supported as run-time abstractions, coordination contexts should be dynamically *configurable* and *inspectable* by both agents and humans. Configurability would allow a MAS to evolve at run time, by suitably adapting its behavior to changes – expectedly a major requirement in complex and highly dynamic scenarios such as the Ubiquitous Computing one [36]. Inspectability would allow both humans and intelligent agents to reason about the current laws of coordination as represented and embodied within coordination contexts, and to possibly change them by properly reconfigure coordination contexts according to new application needs. Finally, since coordination models have been recognized as a basis to enforce social rules [8], and that contexts are easily interpreted as social constructs [1], coordination contexts have the potential to be exploited for the engineering of social order within agent societies [7].

12.5 EXAMPLES

Given the seminal nature of this notes, it is obvious that the notion of agent coordination context has lots of elements and details to be defined and made more precise before it becomes actually implementable and usable. As a result, cases of agent coordination contexts can be given only in terms of either a conceptual example, or a model of coordination where aspects of a coordination context can be suitably devised out at the proper level of abstraction.

12.5.1 The Control Room Metaphor

The *control room metaphor*, as defined in the following, provide a conceptual example of an agent coordination context. According to this metaphor, an agent entering a new environment is assigned its own control room, which is the only way in which it can perceive the environment, as well as the only way in which it can interact. There, *admissible agent inputs* are represented as *lights* (for time-discrete inputs) and *screens* (for time-continuous inputs), while *admissible agent output* are represented as *buttons* (for time-discrete outputs) and *cameras* (for time-continuous outputs). Thus, for instance, bi-directional continuous communication with another agent could be carried on through a dedicated pair screen-camera.

How many input and output "devices" are available to an agent, of what sort, and for how much time – this is what defines the control room *configuration*, that is, the specific agent coordination context. Such a configuration, representing a sort of environment interface for the agent, is at any time prescriptive, in that it fully describes all the admissible interactions for an agent: however, it might change over time whenever needed. In fact, the initial control room configuration is subject to preliminary negotiation between the agent and the hosting node. Then, it could be dynamically modified according to changes in the needs of either the agent or the hosting node providing the coordination context. For instance, due to sudden lack in resources, a control room could be reduced by the hosting infrastructure to the empty context (no interaction allowed, in other words), implicitly forcing the agent to move elsewhere in order to be able to perform some meaningful action. Also, one may think to associate a cost (per time, per use) to every devices made available by a configuration, so that negotiations for the initial control room configuration, or for a subsequent change in the configuration, would also take the economic issue into account.

12.5.2 Coordination Contexts in the TuCSoN Coordination Model

A coordination model as defined in [31] provides a framework and a technology to shape and govern the space of agent interaction. A meaningful example of a coordination model and technology is TuCSoN [29]: there, the coordination space is defined as a collection of local interaction spaces, and agent interaction is mediated through programmable tuple spaces called *tuple centers* [25]. Each TuCSoN hosting node is equipped with an infrastructure providing agents with a set of tuple centers as run-time coordination abstractions to be used to interact with other agents and with the local environment as well. Tuple centers are Linda-like tuple spaces [16] enhanced with the notion of behavior specification, expressed in terms of the first-order logic specification language ReSpecT [24]. Each tuple center is independently programmed through its own ReSpecT specification, defining its behavior in response to communication events. As a result, a tuple center encapsulates the laws of coordination in terms of a ReSpecT specification, which are both inspectable and dynamically configurable.

In terms of a coordination context, TuCSoN provides each agent with a view of the space of interaction that includes a collection of coordination media, namely the tuple centers – but is not necessarily limited to that. On the one hand, in fact, the TuCSoN infrastructure is not prescriptive at all: any interaction pattern other than the tuple center-mediated one is in principle available to agents. So, for instance, two agents could choose to interact directly and bypass tuple center mediation. On the other hand, since tuple centers are provided as *coordination services* by the TuCSoN infrastructure, the deliberate choice of the agents to exploit them is enough to bound and constrain their interaction: until agents interact through tuple centers, they will follow the coordination laws that tuple centers embody. Also, the coordination context provided by TuCSoN is in general inspectable and dynamically configurable by both humans and (intelligent) agents: humans can access and modify tuple centers through inspectors, made available by the infrastructure as development and deployment tools, whereas agents are provided with a set of special meta-level primitives to inspect and change tuple center behavior specifications. Since tuple centers encapsulate both the state of the communication and the laws of coordination as first-order logic clauses, they can be in principle exploited by both humans and agents to monitor a MAS evolution over time, and to possibly change MAS behavior dynamically by suitably affecting their ReSpecT behaviour specification.

12.6 RELATED APPROACHES & OPEN ISSUES

The notion of *context-dependent coordination* was first introduced in [6]. There, as in TuCSoN [29], a multiplicity of different programmable tuple spaces (tuple centers [25]) are used to provide mobile agents with different coordination contexts, independently programmable, and configurable by agents according to their own application needs. Here too, however, prescriptiveness of coordination is bound to either the choice of agents to interact through tuple centers only, or the ability of the infrastructure to super-impose tuple center-mediated interaction to agents – however, this issue is not explicitly discussed in the paper.

A better defined notion of coordination context is instead introduced by *law-governed interaction* [19]. There, a set of policy-independent trusted controllers enforces coordination laws as access policies to tuple spaces locally to agents. Coordination laws are then embodied locally to controllers, which basically work as a sort of coordination contexts.

Even though they do not discuss explicitly a notion of context, agent-oriented methodologies are more and more stressing the role of the environment in the engineering of complex systems. Methodologies like Gaia [39] and SODA [23] emphasize the idea that both agent societies and environment have to be handled as first-class entities. In particular, SODA is explicitly meant to exploit coordination models and technologies for the engineering of the MAS aspects related to agent societies and environment.

Given the seminal nature of these notes, several issues remain open to further research. The first concerns the problem of situatedness and explicit representation of context [40]. While the notion of context representation language naturally suggests a symbolic approach to environment representation, it is not yet clear how much the very notion of situatedness would cope well with symbolic representation of the environment [4]. Also, it is not clear whether communication should be handled and represented as a special kind of action, or as an independent modality of interaction [15].

The notion of prescriptive coordination discussed in previous sections raise the issue of security, in particular in the context of open systems. Given that the main goal of security is, roughly speaking, to manage agent/environment interactions so as to prevent possibly harmful behaviors, security can in principle be intepreted as a coordination issue [10, 5]. So, how can an agent coordination context represent and embody the level of security required by today systems? And to what extent can security policies be defined and enforced, without harming the fundamental notion of agent autonomy? Some work in this direction already began, even though it is still quite preliminary [11].

Last but not least, coordination contexts could provide systems with an economy-oriented view of the environment. Information has a cost, interaction and communication have a cost, too: access to resources and services, as well as to communication media, is not to be taken as free to agents and applications in general. Also, coordination adds a value per se, which should be properly accounted for. So, a coordination context could in principle be used also to provide agents with an economy model of the environment, even though it is not clear how costs should be represented in general. For instance, an agent entering a new environment could initially contract to buy a cheap and "weak" coordination context (with only limited access to resources), and only after a preliminary exploration phase deliberate whether to leave the place (having found it not interesting at all) or to stay there, possibly extending its local interaction capabilities by negotiating (and paying for) a more powerful coordination context.

12.7 CONCLUSIONS

The study of the evolution of human societies tells us that we cannot even understand the evolution of individuals and groups in any non-trivial setting without explicitly taking the environment into account. The notion of context is already extensively used in several research fields to model the effect of the environment upon individuals and organizations. Also, a notion corresponding to object interface is required in agent systems to discipline and engineer the space of agent interaction – both agent-to-agent and agent-to-environment interactions. Then, the notion of *agent coordination context* was introduced here as both a representation tool and a run-time abstraction meant to handle agent environment as a first-class entity in the modeling and engineering of complex agent systems.

It is anyhow clear that any feasible and manageable notion of agent coordination context requires ontologies, abstractions, and languages to represent agent interaction within an environment, as well as run-time technologies, tools, and methodologies to handle the environment as a first-class entity in the modeling and engineering of agent systems. Most of these issues are likely to be addressed in the research work that will follow these seminal notes.

12.8 ACKNOWLEDGEMENTS

First, I would like to thank MIUR (the Italian "Ministero dell'Istruzione, dell'Università e della Ricerca") and Nokia Research Center, Burlington, MA, USA, for partially supporting this work. Also, let me thank the organizers of the Workshop on Internet Process Coordination and Ubiquitous Computing, and all the attendants as well, that stimulated the development of the concepts and ideas presented here.

Finally, I am deeply grateful to Alessandro Ricci, whose continuous flow of materials, ideas, remarks and criticisms was of invaluable help in improving the various version of this article.

REFERENCES

1. Varol Akman. Rethinking context as a social construct. *Journal of Pragmatics*, 32(6):743–759, 2000.

2. Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, May 2001.

3. Pierre Bonzon, Marcos Cavalcanti, and Rolf Nossum, editors. *Formal Aspects of Contexts*, volume 20 of *Applied Logic Series*. Kluwer Academic Publishers, 2000.

4. Rodney A. Brooks. *Cambrian Intelligence: The Early History of the New AI*. MIT Press, 1999.

5. Ciarán Bryce and Marco Cremonini. Coordination and security on the Internet. In Omicini et al. [30], chapter 11, pages 274–298.

6. Giacomo Cabri, Letizia Leonardi, and Franco Zambonelli. Context-dependency in Internet-agent coordination. In Omicini et al. [28], pages 51–63.

7. Cristiano Castelfranchi. Engineering social order. In Omicini et al. [28], pages 1–18.

8. Paolo Ciancarini, Andrea Omicini, and Franco Zambonelli. Multiagent system engineering: The coordination viewpoint. In Nicholas R. Jennings and Yves Lespérance, editors, *Intelligent Agents VI. Agent Theories, Architectures, and Languages*, volume 1757 of *LNAI*, pages 250–259. Springer-Verlag, 2000. 6th International Workshop, ATAL'99, Orlando, FL, USA, 15–17 July 1999, Proceedings.

9. Paolo Ciancarini and Michael J. Wooldridge, editors. *Agent-Oriented Software Engineering*, volume 1957 of *LNCS*. Springer-Verlag, 2001. 1st International Workshop, AOSE 2000, Limerick, Ireland, 10 June 2000, Revised Papers.

10. Marco Cremonini, Andrea Omicini, and Franco Zambonelli. Multi-agent systems on the Internet: Extending the scope of coordination towards security and topology. In Francisco J. Garijo and Magnus Boman, editors, *Multi-Agent Systems Engineering*, volume 1647 of *LNAI*, pages 77–88. Springer-Verlag, 1999. 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW'99, Valencia, Spain, 30 June – 2 July 1999, Proceedings.

11. Marco Cremonini, Andrea Omicini, and Franco Zambonelli. Coordination and access control in open distributed agent systems: The TuCSoN approach. In António Porto and Gruia-Catalin Roman, editors, *Coordination Languages and Models*, volume 1906 of *LNCS*, pages 99–114. Springer-Verlag, 2000. 4th International Conference, COORDINATION 2000, 11–13 September 2000, Limassol, Cyprus, Proceedings.

12. Enrico Denti, Antonio Natali, Andrea Omicini, and Francesco Zanichelli. Robot control systems as contextual logic programs. In Christoph Beierle and Lutz Plümer, editors, *Logic Programming: Formal Methods and Practical Applications*, volume 11 of *SCSAI*, pages 343–379. Elsevier Science, 1995.

13. Anind K. Dey, Daniel Salber, and Gregory D. Abowd. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. In *Human-Computer Interaction* [20], pages 97–166.

14. Jared Diamond. *Guns, Germs, and Steel: The Fates of Human Societies*. W.W. Norton & Company, 1^{st} edition, March 1997.

15. Peter Gärdenfors. The pragmatic role of modality in natural language. In Paul Weingartner, Gerhard Schurz, and Georg Dorn, editors, *The Role of Pragmatics in Contemporary Philosophy*, pages 78–91. Holder-Pichler-Tempsky, Vienna, 1998. 20th International Wittgenstein Symposium, Proceedings.

16. David Gelernter. Generative communication in Linda. *ACM Transactions on Programming Languages and Systems*, 7(1):80–112, January 1985.

17. Thomas Malone and Kevin Crowstone. The interdisciplinary study of coordination. *ACM Computing Surveys*, 26(1):87–119, 1994.

18. John Maynard Smith and Eörs Szathmáry. *The Origins of Life: From the Birth of Life to the Origins of Language*. Oxford University Press, 1^{st} edition, May 1999.

19. Naftaly H. Minsky, Yaron M. Minsky, and Victoria Ungureanu. Safe tuplespace-based coordination in multiagent systems. In *Applied Artificial Intelligence* [26], pages 11–34.

20. Thomas P. Moran and Paul Dourish. Context-aware computing. *Human-Computer Interaction*, 16(2-4), 2001. Special Issue.

21. James J. Odell, H. Van Dyke Parunak, and Bernhard Bauer. Representing agent interaction protocols in UML. In Ciancarini and Wooldridge [9], pages 121–140.

22. OMG. CORBA home page. `http://www.omg.org/corba/`.

23. Andrea Omicini. SODA: Societies and infrastructures in the analysis and design of agent-based systems. In Ciancarini and Wooldridge [9], pages 185–193.

24. Andrea Omicini and Enrico Denti. Formal ReSpecT. In Agostino Dovier, Maria Chiara Meo, and Andrea Omicini, editors, *Declarative Programming – Selected Papers from AGP'00*, volume 48 of *Electronic Notes in Theoretical Computer Science*, pages 179–196. Elsevier Science B. V., 2001.

25. Andrea Omicini and Enrico Denti. From tuple spaces to tuple centres. *Science of Computer Programming*, 41(3):277–294, 2001.

26. Andrea Omicini and George A. Papadopoulos. Coordination models and languages in AI. *Applied Artificial Intelligence*, 15(1):1–103, 2001. Special Issue.

27. Andrea Omicini, Paolo Petta, and Robert Tolksdorf, editors. *Engineering Societies in the Agents World II*, volume 2203 of *LNAI*. Springer-Verlag, 2001. 2nd International Workshop, ESAW'01, Prague, Czech Republic, 7 July 2001, Revised Papers.

28. Andrea Omicini, Robert Tolksdorf, and Franco Zambonelli, editors. *Engineering Societies in the Agents World*, volume 1972 of *LNAI*. Springer-Verlag, 2000. 1st International Workshop, ESAW'00, Berlin, Germany, 21 August 2000, Revised Papers.

29. Andrea Omicini and Franco Zambonelli. Coordination for Internet application development. *Autonomous Agents and Multi-Agent Systems*, 2(3):251–269, 1999. Special Issue: Coordination Mechanisms for Web Agents.

30. Andrea Omicini, Franco Zambonelli, Matthias Klusch, and Robert Tolksdorf, editors. *Coordination of Internet Agents: Models, Technologies, and Applications*. Springer-Verlag, March 2001.

31. George A. Papadopoulos. Models and technologies for the coordination of Internet agents: A survey. In Omicini et al. [30], chapter 2, pages 25–56.

32. Michael Schumacher. *Objective Coordination in Multi-Agent System Engineering – Design and Implementation*, volume 2039 of *LNAI*. Springer-Verlag, 2001.

33. Roger Sessions. *COM and DCOM: Microsoft's Vision for Distributed Objects*. John Wiley & Sons, December 1997.

34. Jim Waldo. The Jini architecture for network-centric computing. *Communications of the ACM*, 42(7):76–82, 1999.

35. Peter Wegner. Why interaction is more powerful than computing. *Communications of the ACM*, 40(5):80–91, 1997.

36. Mark Weiser. Hot topic: Ubiquitous computing. *IEEE Computer*, 26(10):71–72, October 1993.

37. Terry Winograd. Architectures for context. In *Human-Computer Interaction* [20].

38. Michael J. Wooldridge, Paolo Ciancarini, and Gerhard Weiss, editors. *Agent-Oriented Software Engineering II*, volume 2202 of *LNCS*. Springer-Verlag, 2002. 2nd International Workshop, AOSE 2001, Montreal, Canada, 29 May 2001, Revised Papers & Invited Contributions.

39. Franco Zambonelli, Nicholas R. Jennings, Andrea Omicini, and Michael J. Wooldridge. Agent-oriented software engineering for Internet applications. In Omicini et al. [30], chapter 13, pages 326–346.

40. Tom Ziemke. Embodiment and context. In *European Conference on Cognitive Science, ECCS'97, Workshop on Context*, Manchester, UK, 9–11 April 1997.

Author(s) affiliation:

- **Andrea Omicini**

  DEIS, Università di Bologna, Sede di Cesena
  via Rasi e Spinelli 176, 47023 Cesena, FC, Italy
  Email: aomicini@deis.unibo.it