Agent Environments for Multi-Agent Systems – A Research Roadmap

Danny Weyns[1], Fabien Michel[2], H. Van Dyke Parunak,
Olivier Boissier, Michael Schumacher, Alessandro Ricci
(Organizers E4MAS – 10 Years Later)
Anarosa Brandao, Carlos Carrascosa, Oguz Dikenelli, Stépane Galland,
Ander Pijoan, Patrick Simo Kanmeugne, Juan A. Rodriguez-Aguilar,
Julien Saunier, Visara Urovi, Franco Zambonelli
(Section Coordinators)

[1] Linnaeus University Sweden, KU Leuven Belgium
danny.weyns@cs.kuleuven.be
[2] Université de Montpellier, France
fmichel@lirmm.fr

Abstract. Ten years ago, researchers in multi-agent systems became more and more aware that agent systems consist of more than only agents. The series of workshops on Environments for Multi-Agent Systems (E4MAS 2004-2006) emerged from this awareness. One of the primary outcomes of this endeavor was a principled understanding that the agent environment should be considered as a primary design abstraction, equally important as the agents. A special issue in JAAMAS 2007 contributed a set of influential papers that define the role of agent environments, describe their engineering, and outline challenges in the field that have been the drivers for numerous follow up research efforts. The goal of this paper is to wrap up what has been achieved in the past 10 years and identify challenges for future research on agent environments. Instead of taking a broad perspective, we focus on three particularly relevant topics of modern software intensive systems: large scale, openness, and humans in the loop. For each topic, we reflect on the challenges outlined 10 years ago, present an example application that highlights the current trends, and from that outline challenges for the future. We conclude with a roadmap on how the different challenges could be tackled.

Keywords: agent environment, multi-agent systems, middleware, large-scale systems, open systems, human in the loop.

1 Introduction

Ten years ago, the awareness grew among researchers in the multi-agent systems community that agent systems consist of more than only agents. The Environments for Multi-Agent System workshop (E4MAS [2]) that was organized in conjunction

with AAMAS 2004 emerged from this awareness. The driver for E4MAS 2004 was the following statement:

> *There is a general agreement in the research community that agent environments are essential for multi-agent systems, yet researchers neglect to integrate the agent environment as a primary abstraction in their models and tools for multi-agent systems*.

During three successful editions of the E4MAS workshop [2][3][4] and various additional activities, a substantial group of researchers worked intensively on the subject of agent environments. One of the primary outcomes of this endeavor was a principled understanding that the agent environment should be considered as a primary design abstraction, equally important as the agents. Different models and architectures have been proposed to design agent environments, and these designs have been validated in a variety of application domains. A special issue devoted to agent environments in multi-agent systems in the Journal on Autonomous Agents and Multi-Agent Systems in 2007 [5] included a set of influential papers that define the role of agent environments, describe their engineering, and outline challenges in the field that have driven numerous follow-up research efforts.

At AAMAS 2014 in Paris, researchers in the E4MAS domain organized a workshop on "E4MAS—10 Years Later," and this paper builds upon discussions at that workshop. The goal of this paper is:

- To reflect on the past 10 years of research and engineering on agent environments for multi-agent systems;
- To investigate to what extent the challenges identified a decade ago have been tackled;
- To outline challenges for future research on a short and longer term.

Instead of taking a broad perspective, we focus on three particularly relevant topics of modern software intensive systems: the large scale of systems, the openness of systems to deal with parts that enter and leave the system dynamically, and humans in the loop that interact with the system. Evidently, we focus on these topics from the viewpoint of agent environments for multi-agent systems. For each topic, we explain the topic and highlight challenges outlined 10 years ago, we present an example application illustrating the current trends, and then we outline challenges for the future. We conclude the paper with a roadmap to tackle the different challenges.

The remainder of this paper is structured as follows. Section 2 focuses on the impact on agent environments for large-scale multi-agent systems. Section 3 zooms in on open agent environments for multi-agent systems. Section 4 discusses the impact of humans in the loop on agent environments. Finally, Section 5 outlines a possible roadmap for future research in this important field.

2 Agent Environments for Large-Scale Multi-Agent Systems

Many real world problems are of high dimension (lots of interacting features), large in size and often stochastic by nature [22]. Such Large-Scale Systems (LSSs) are

intricately multifarious, with multiple objectives that can lead to conflicts among the multiple decision makers present in these systems. A system can be considered as an LSS if one (or both) of the following perspectives holds [23]: *(i)* It can be decomposed into a number of interconnected subsystems, either for practical reasons (design) or because computation needs to be distributed (performance); (ii) Its high dimensionality leads to a combinatorial explosion in its space of possible behaviors, so that the usual methods for modeling analyzing, controlling or designing cannot find a solution in a reasonable amount of time. As a result, these systems require that the control of the data and/or computation be decentralized over the subsystems. The engineering of LSSs has been subject of extensive research, including approaches proposed for dealing with complexity within the field of multi-agent systems.

In particular, MASs are a natural approach for modeling and implementing LSSs because they rely on decentralized loci of control by means of agents [22]. A Large Scale MAS (LSMAS) is a MAS that is hard to (i) *engineer* (e.g. coordination among thousands of agents) or (ii) *deploy* (e.g., real-time interaction may no longer hold due to the computational requirements). Bottlenecks in an LSMAS are usually related to the size of the system in terms of the number of agents and the amount of data in the system. Indeed, regardless of the application domain, each additional agent requires some computational resources. Moreover, the MAS should be able to accept new agents without compromising its functioning. This section discusses the crucial role of agent environments for an LSMAS, i.e., a MAS with a large number of agents evolving in application environments that potentially involve a huge amount of data.

2.1 Large scale and Agent Environments

The agent environment is now broadly recognized as a first class abstraction for building a MAS, especially because it mediates interactions between agents and their access to resources [7]. In an LSMAS, the number of interactions and resources could be very large, hence the design of the agent environment is even more crucial because it directly impacts scaling issues and plays an important role in managing potential bottlenecks. We put forward four requirements that are central to engineering agent environments for LSMAS:

1. *Scalable Structure:* refers to distributing the computation and state of the agent environment. The agent environment may have different structures: multi-level or hierarchical, multi-stage or dynamic. For example, the agent environment may be structured in segments, each representing a local view on the physical environment; segments may be connected via a P2P network.
2. *Access to Resources:* Typically, LSMASs are composed of heterogeneous agents deployed in an agent environment, which defines laws that regulate access to resources. At a large scale, monitoring, trust, and security aspects are to be carefully designed so that the cost induced by managing access to resources does not become a bottleneck.
3. *Scalable Communication*: When coordination among thousands of entities is required, the agent environment should provide means for communication between agents that do not involve any central point of access or control.

4. *Interaction model:* to achieve scalable agent environments, it is important to provide agents with efficient means for perceptions, actions, and interactions. Central here are suitable abstractions, e.g., the agent environment should offer high-level primitives to agents for perception, coordination etc., that support efficient processing by the agents.

2.2 Challenges on Large Scale Agent Environments in Retrospect

While scalability was not a prominent topic in the past E4MAS efforts, the four requirements mentioned in the previous section have been partly identified or addressed in different contexts during the period 2004-2007.

Scalable Structure and Access to Resources. Several researchers showed that the structural scalability of the agent environment is strongly related to the ability to achieve decentralized control over the environmental data and dynamics, so that one can move easily from a monolithic structure to a distributed one. In [24], the agent environment is decomposed into independent *interaction spaces,* each of which defines explicitly local environmental rules. In the domain of large-scale traffic simulation, [25] applies a holonic modeling of the agent environment so that the environmental processes apply only locally. These examples show that decentralizing the structure of the agent environment and managing access to resources based on the principle of locality are already identified as key principles for achieving scalability of agent environments.

Scalable Communication and Interaction Model. A decade ago, considering dynamics in the agent environment as a efficient means for achieving communication and coordination in an (LS)MAS was already a topic of interest in the E4MAS community. Especially, nature-inspired mechanisms supporting indirect communications through the agent environment, such as digital pheromones and force fields, were considered to scale better than direct message exchanges (see e.g. [28][29][26]), thus providing scalable interaction models for achieving coordination among numerous agents using stigmergic principles. Nevertheless, it is interesting to note that the mechanisms for engineering the agent environment discussed in [8] do not consider explicitly scalability as a main feature of interest. Since then, the dramatic evolution of the technological context, especially with respect to the exponential increase of smart mobile devices, has put scalability on the agenda as a major topic. Nowadays, scalability is no longer an option, but a requirement for many MAS applications.

2.3 Example Application

We illustrate a recent effort on agent environments for LSMASs in the context of *Personalized Health Systems* (PHSs). PHSs are systems that support patient-centered healthcare by assisting patients in self-managing their medical conditions. Using a PHS, patients and caregivers are connected so that health data are accessible

independently from their geographical location. Since the patient's data is generated in a distributed setting, these systems require reliable, scalable and interoperable models of information flow. For example, [30] models the discovery and exchange of health records with a dynamic interoperable MAS network. A high-level model of this system is illustrated in Fig.1.

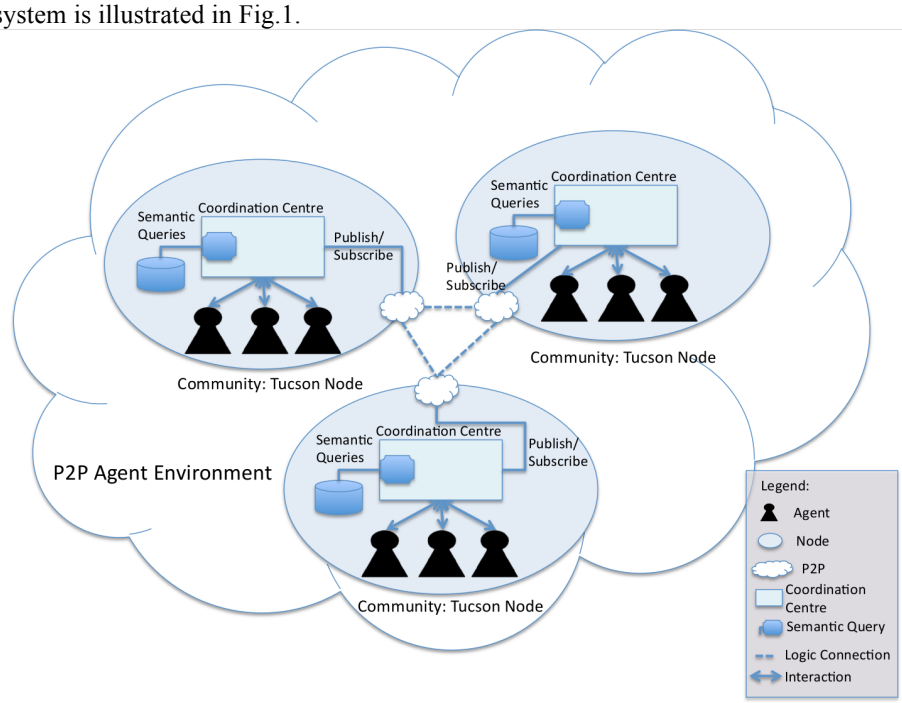

Fig. 1. SemHealthCoord: An agent-based LSMAS model for health data exchange

Different health communities (i.e. hospitals) store the patient's data. A Peer-to-Peer (P2P) architecture connects these communities dynamically and at large scale. Fig. 1 shows how the agent environment is organized. Health communities are connected as *Nodes* in a *P2P* network. A set of coordination rules *(Coordination Center)* defines how agents can find patients' data and how they can propagate updates in the network of communities. Since the data in different communities may be organized differently, the querying of data follows a semantic knowledge base (*Semantic Queries*). In this model, the agents specialize on performing specific tasks (i.e. finding the data about a patient) while the agent environment itself defines how the interactions can take place across communities. More specifically, the *Coordination Center* specifies how data can be queried in a distributed level and how new data can be propagated to different communities. The *Agent Environment* uses the TuCSoN coordination model [32] where agents retract, write or read (called *in-out* or *rd primitives)* data in the *Coordination Center* using specific tuple templates. These actions trigger reactions that coordinate the tasks of different agents, despite these agents may not share the same space, may not know each other's reference, and may not be synchronized.

2.4 Challenges Ahead

Realizing the requirements of LSMASs outlined in Section 2.1, namely: (1) making the structure scalable, (2) ensuring efficient access to resources, providing (3) scalable communication means and (4) interaction models, are still major challenges. In this respect, previous research emphasizes the crucial role of locality and decentralization when engineering the agent environment's structure and mechanisms. Not addressing these aspects puts more responsibility on the agents, which leads to complex agents and hampers scalability. However, achieving locality and decentralization is not sufficient if the system cannot be adapted and evolved over time. Therefore, future research on scalable agent environments is about addressing the different aspects in an integrated manner. We outline two key aspects for future efforts.

As we move toward LSMASs that have to deal with huge amounts of data, elaborating efficient structures and dynamics is not only a solution for achieving scalable communication and interaction, but also a key to more effective processing of data and information. To that end, we see two important challenges that agent environments have to address: *(i)* Preprocessing data: data should be modeled and structured so that they can be easily managed and evolved using large scale dynamics compliant to the underlying environment (e.g., by taking inspiration from map reduce approaches), and *(ii)* Post-processing data: data should be synchronized with the agents' needs. In other words, the agent environment could anticipate requests by processing data accordingly, through internal dynamics.

Another central challenge lies in designing agent environment structures and dynamics in an integrated way; e.g., design agent environment dynamics so that they accommodate the underlying physical infrastructure. Considering this aspect, one can take inspiration from the General-Purpose computing on Graphics Processing Units (GPGPU) community (High Performance Computing). In this context, computation and data models are explicitly considered so that they can benefit from the underlying physical infrastructure of the GPU (a massively parallel architecture). Performance and scalability are directly influenced by how the data model accommodates the underlying hardware. So, it is possible to design scalable agent environment dynamics very efficiently because they are modeled matching the physical infrastructure. One recent example is the use of digital pheromones in LSMAS simulations [34].

3 Open Agent Environments for Multi-Agent Systems

Living in an environment, perceiving it, and being affected by it intrinsically imply openness. Software systems are no longer isolated, but become permeable sub-systems, whose boundaries permit reciprocal side effects. The reciprocal influence between system and environment is often extreme and complex, making it difficult to identify clear boundaries between the system and its environment.

In several cases, to achieve their objectives, software systems must interact with external software components, either to provide services and data, or to acquire them. More generally, different software systems, independently designed and modeled, are likely to "live" in the same environment and interact explicitly with each other. These

open interactions call for common ontologies, communication protocols, and suitable broker and coordination infrastructures to enable interoperability.

A major advance in engineering multi-agent systems has been the recognition of the importance of the agent environment in which the agents are situated, and through which they interact, as a first-class abstraction. However, current environment-based multi-agent systems rely on a fixed, a priori definition of the agent environment, and only agents that conform to that definition can exploit it. A powerful next step is the notion of an open agent environment, one that adapts in response to the agents that inhabit it.

This section explores the theme of open agent environments for multi-agent systems. We start by explaining the viewpoint we take on openness of agent environments in this paper. Then we look at challenges of open agent environments that have been identified earlier and reflect on these. We continue by illustrating a typical existing approach to deal with openness in multi-agent systems. Finally, we reflect and outline challenges ahead.

3.1 What is Openness?

The concept of openness of software systems is not well defined in the literature. [12] refers to open software systems as systems that are specifically built to allow for extensions. [13] considers openness as a property of software systems that are subject to decentralized management and can dynamically change their structure. [17] refers to openness as the system's ability to deal with entities leaving and entering the system. [14] refers to openness of a MAS as "the ability of introducing additional agents into the system in excess to the agents that comprise it initially." He categorizes openness in three levels: (1) off-line openness, which allows addition of new agents only off-line, e.g., by halting the system, adding agents, updating some connection information, and re-starting the system, (2) static openness where agents can be added to the system without re-starting it, but all of the agents either are notified of such an addition, or they hold in advance a list of prospective additional agents, and (3) dynamic openness that allows agents to leave or enter the system dynamically, during run time, without explicit global notification.

Our particular interest here is in dynamic openness, which enables a system to adjust itself dynamically to uncertainty in the environment, tasks, and availability of resources. As outlined by numerous researchers, this type of uncertainty is particularly relevant for systems that are deployed in environments with high levels of dynamicity and change, which are nowadays the rule rather than the exception [14][15][16].

3.2 Challenges on Openness of Agent Environments in Retrospect

In the period 2004 to 2007, several researchers pointed out challenges on the openness of agent environments. [1] poses the following question:

> *What responsibilities does the agent environment have and what services can it provide to increase its openness to heterogeneous agents?*

Openness of agent environments was primarily seen as an engineering challenge. For example, [7] identifies the need for suitable software architectures for the agent environment, while [18] argues for suitable abstractions and infrastructures to support agent environment design. [19] stresses the need of suitable mechanisms for the agent environment to support social interactions. On a more concrete level, [17] poses the question whether electronic institutions can be further exploited to handle openness. The emphasis on openness of agent environments has been primarily on the need for architectures and infrastructures that allow different agents to join or leave a multi-agent system at will. The uncertainty in the deployment context, tasks, the availability of resources and changing system requirements, and its impact on the openness of multi-agent systems was not of primary concern a decade ago. This is not surprising, as the dramatic change of operating conditions in which software intensive systems are expected to operate has only become clear over the years.

3.3 Example Application

We illustrate the efforts on openness in engineering agent environments with an example in the domain of supply chain management. Modern supply chain management requires the collaboration of distributed and heterogeneous systems of multiple companies, which naturally maps to open multi-agent systems. However, developing such collaborative applications and building the supporting information systems poses several engineering challenges. [20] presents Macodo, an architectural approach that aims to address the problem of managing the design complexity of collaborative applications.

Central to Macodo are five abstractions: actor, collaboration, role, behavior, and interaction. Macodo offers a middleware infrastructure that supports these abstractions at the levels of design and implementation. An *actor* is an entity that has access to the collaboration environment and is capable of participating in collaborations by playing roles. In a concrete system, actors can be business entities, software agents, services, or even people. A *collaboration* is a controlled process, taking place in the collaboration environment, of a group of actors working together towards a set of goals. A collaboration consists of a set of roles, representing the different actors and their responsibilities in the collaboration, and a set of interactions among the actors of these roles. Collaborations are reusable and can be created and destroyed by the manager of the collaboration. A *role* is the embodiment of the participation of an actor in a collaboration that defines the actor's responsibilities in that participation. When an actor enters a collaboration, a new role instance is created. When the actor leaves the collaboration, the corresponding role instance is destroyed. The distinction between role and role instance is similar as in [55] that distinguishes between role types and role instances. Within the context of a role, an actor can execute behaviors and participate in interactions with other actors in the collaboration. A *behavior* is a coherent unit of reusable functionality that is executed in the context of a role. A behavior is typically application-specific and can encapsulate the execution of a task or the participation in an interaction. Finally, an *interaction* is a controlled exchange of information between the actors of a set of roles in a collaboration. An interaction can have an application-specific protocol.

Macodo offers a set of architecture views that support engineers in modeling applications using these abstractions. The *Collaboration View* models collaborations as reusable modules and shows how they are decomposed into reusable submodules (i.e., roles, interactions, and behaviors). The Collaboration View is used to describe the collaborations in a system in terms of implementation units. The *Collaboration & Actor View* models the actors in a system and the concrete collaboration instances among them. In this view, actors are represented as components, and collaborations as connectors. The Collaboration & Actor View is used to describe the runtime architecture of a system in terms of actors and the collaborations between them, assigning responsibilities to actors, while making abstraction of collaboration details. The *Role & Interaction View* models the internal runtime architecture of a collaboration in detail. This view allows documenting the concrete role and interaction instances in a collaboration, the active behaviors of roles, and how roles delegate the participation in interactions to behaviors. A behavior is executed in the context of a role, giving the actor of the role access to the interfaces of the behavior.

The Macodo abstractions and architectural views allow the modeling and documentation of collaborative applications. The Macodo middleware provides an agent environment to design and implement collaborative applications that are modeled in the Macodo architectural views. The platform supports the Macodo abstractions as programming abstractions by mapping them to concrete technology. Fig. 2 shows the primary elements of the Macodo middleware.

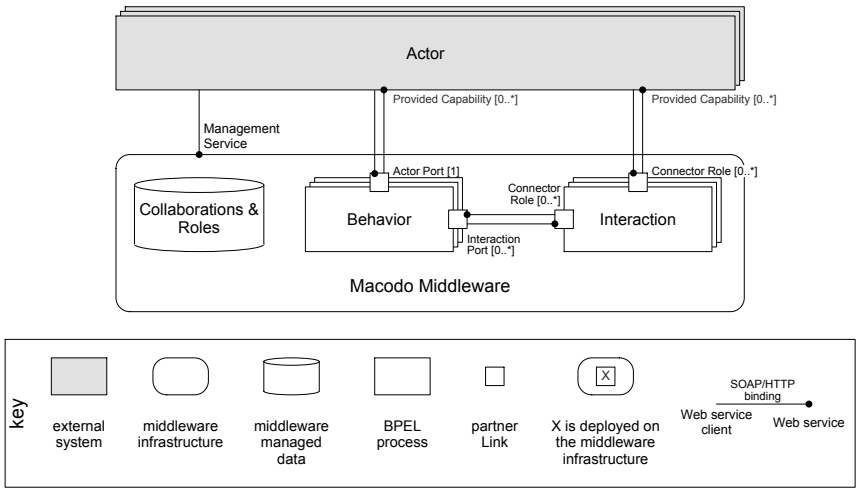

Fig. 2 Macodo middleware

[20] presents a concrete realization of the Macodo middleware using Web Services technology. Once specified, collaboration modules can be loaded in the Macodo middleware. The management service of the middleware can then be used to register actors and to manage the life-cycle of concrete collaboration and role instances. After a role has been assigned to an actor, the actor can 'play' the role. To play a role, an actor uses interactions and behaviors. The information flow between the actors, interactions, and behaviors is mediated by the middleware, which routes messages to

the correct interactions, behaviors, and actors. Messages between the middleware and actors contain additional Macodo data, which uniquely identifies the role to which a message belongs. By decoupling actors from the roles they play, the Macodo middleware offers an open agent environment where different agents can join and leave collaborations at will.

In a concrete supply chain, the supply chain partners are the actors that can play the roles of vendor, warehouse, retailer, and transporter. Each supply chain network can be modeled as a collaboration. For example, in a vendor-managed inventory (VMI), the vendor is responsible for managing the inventory. Products in the inventory, kept in an intermediate warehouse, remain property of the vendor until consumed, or called-off, by the retailer. The warehouse regularly reports inventory levels to the vendor. Based on these inventory levels, the vendor replenishes the warehouse. The retailer can call-off products from the warehouse, after which it reports the consumption to the vendor. To model these collaborations, we can define roles, behaviors, and interactions. For example, we can define an Inventory Reporting Behavior for the Warehouse role to collect inventory levels and pass it to another role using an Inventory Reporting Interaction (send inventory levels to interested parties).

The architecture views then support modeling concrete applications. For example, with the Role & Interaction View can be used to model runtime qualities, such as throughput of interactions or robustness of behaviors. We can, for example, specify that the Call-Off Fulfillment Behavior should always reply to a Call-Off Interaction, even if the actor of the Warehouse role is not reacting. The specifications can then be implemented using the Macodo middleware programming abstractions and concrete instanced can be loaded in the Macodo middleware. At runtime, actors can dynamically enter, participate, and leave collaborations, and new actors can join. For example, a new transporter can enter a collaboration and supply chain partners may switch the transport service dynamically taking into account ongoing agreements.

3.4 Challenges Ahead

In previous research, openness of agent environments has primarily been approached from an engineering perspective, emphasizing the ability of different agents to join or leave a multi-agent system at will. As illustrated with the example above, the main focus has been on identifying suitable modeling abstractions, architectures and infrastructures to support open agent environments. However, the ever-growing complexity of software systems introduces a variety of uncertainties that need to be handled at runtime, including dynamics in operating conditions that are difficult to predict and the need to handle changing system requirements that may not be anticipated at design time. Several researchers have pointed out that traditional engineering approaches may not be sufficient to deal with these uncertainties, and call for new engineering solutions. To support openness, we see the following key challenges for the next generation of agent environments:

- Handling uncertainty as a first-class citizen to deal with the inherent dynamics of the context in which multi-agent systems are deployed.

- Reducing seamless integration of online runtime adaptation and offline evolution.
- Support for agents to form sustainable ecosystems (e.g., infrastructure that enables integration of mobile applications developed by different vendors).
- Efficient integration of a wide spectrum of services, from integrating 'things' to supporting intelligent cooperation between and among agents and humans.

4 Agent Environments and Humans in the Loop

Emerging technologies such as wireless sensor networks, Internet of Things (IoT), and smart and wearable devices, provide the basis for new types of applications where the physical world can be accessed or modified by computational systems. Examples of such systems are energy management, health care, and traffic systems. These applications are characterized by *humans in the loop*, i.e., humans are an essential part of the realization of the rich functionalities of such systems. Humans can have the role of users of the system, where they are in continuous interaction with the system through computational devices (PC, tablet, smartphones, etc.), or with the physical environment itself, as in IoT. Humans can also have a role as being integral parts of the system itself, i.e., socio-technical systems. Examples include incorporating users to perform security-critical functions, and incorporating activity models in smart homes to improve the independence of elderly people.

Multi-agent systems are an effective approach for modeling and designing systems with humans in the loop, given their characteristics of autonomy and sociality. In particular, the notion of agent environment can play a crucial role, since the environment is a natural place to model the shared distributed physical and social world with which systems and people interact, and it offers rich forms of communication, either explicit or implicit, temporary or persistent, with manageable levels of coupling.

In this section, we explore the role of the agent environment in the design of multi-agent systems with humans in the loop. We start by outlining the position of humans in the loop in computing systems. Then we look at challenges that have been identified earlier and reflect on these. We provide a recent example application that shows how humans are integrated in the loop in a multi-agent system, and conclude with challenges ahead in this promising area for future research.

4.1 Humans in the Loop

Based on a cursory review of the literature we identified several levels of involvement of humans in the loop in computing systems. We noticed a particular interest for humans-in-the-loop systems in the control systems community; see for example [36][38][35]. Example efforts in the context of MAS are [42][41][40][43][49].

1. Humans-in-the-loop monitoring. This level is characterized by a system that monitors humans and takes appropriate actions when needed. An example is AlarmNet [37], which is a smart home health care application that monitors

activities of daily living by using environmental and wearable sensors and creates a continuous medical history. Authorized health care providers are allowed to monitor activity patterns to determine if the residents need immediate attention or new healthcare services.

2. Humans-in-the-loop interaction. This level is characterized by humans that are in continuous interaction with the system through computational devices. An example is a mobile application that supports users to find each other based on particular criteria such as locality, preferences, social contacts etc.
3. Minimizing human intervention. This level is characterized by a system that only invokes a human operator when necessary, and does so in a minimally intervening manner. An example is a human who is responsible for security-related configuration decisions and enacting particular policies [39]. Such tasks require knowledge that may be very hard to codify.
4. Humans-in-the-loop supervisory control. This level is characterized by intermittent human operator interaction with a remote, automated system in order to manage a controlled process or task environment. Examples include air traffic control, military and space command and control, crises response management, and unmanned vehicle operations.

Our interest in this section is on the different levels of human involvement in the loop in multi-agent systems.

4.2 Challenges on Humans in the Loop in Retrospect

In the period 2004 to 2007, humans-in-the-loop in the context of agent environments has not been explored very well in E4MAS research. [7] distinguishes between three levels of support provided by the agent environment in MAS:

1. Providing support to agents for accessing the deployment context. Agents have low-level knowledge to directly access hardware and software resources.
2. Providing agents an abstraction level to the deployment context. The abstraction level bridges the conceptual gap between the agent abstraction and low-level details of the deployment context.
3. Providing support to agents for interaction-mediation. The interaction-mediation level offers support to regulate the access to shared resources, ensure restrictions are met and mediate interaction between agents.

The three levels of support of the agent environment represent different degrees of functionality that agents can use to achieve their goals. The obvious question in the context of this section is: where are humans situated in this three level reference model? Given the different levels of involvement of humans in the loop in MAS, bringing humans in this picture is not a simple task. Straightforward modeling of humans as either part of the deployment context or “special agents” will not be satisfactory for the different responsibilities of humans in the loop in MAS. The key point is to understand how the agent environment as first-class abstraction can support different levels of involvements of humans in the loop in MAS.

4.3 Example Application

We illustrate current research on humans in the loop in MAS with an example application from the domain of pervasive and ubiquitous computing that is called a *sociotechnical superorganism* [44]. Pervasive and ubiquitous computing is a well-known and obvious case where humans are in the loop. In these kinds of systems, the infrastructure is used ubiquitously to access and deploy new services for interacting with the surrounding physical world and with the social activities occurring in it.

A sociotechnical superorganism comprises networks of entities -- ICT devices and citizens – that continuously and seamlessly cooperate in highly decentralized activities. Entities can be involved in participatory sensing activities, and the results of real-time sharing of knowledge at city scale enables a shared understanding, via machine-based computing and humans-based reasoning, of urban issues of interest and their dynamics. This in turn makes it possible to plan and direct responses or fix problems with collective actions. Consequently, intelligent, coordinated responses to city-scale problems emerge from a closed feedback loop involving collective sensing activities, understanding and sharing of ideas, and collaborative actions.

In the SAPERE approach [45] pervasive service environments are modeled and architected as a non-layered spatial substrate, laid above the actual pervasive network infrastructure, on top of which human users act as prosumers continuously producing and consuming data. Fig. 3 shows the SAPERE Reference Model. The agent environment (MAS Environment) abstractions support the design of agents' activities and interactions.

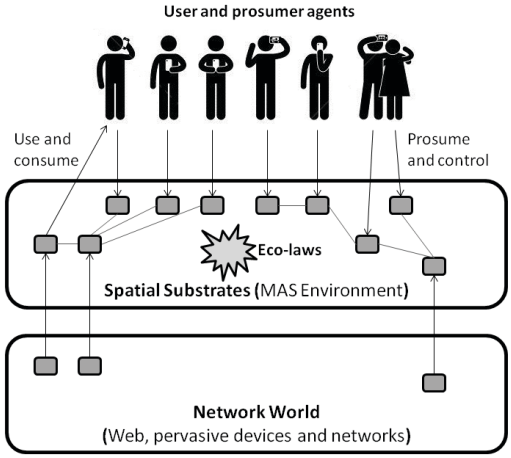

Fig. 3. SAPERE Reference Model [56]

The substrate embeds the basic eco-laws that rule the activities of the system. There, individuals of different species -- agents/services, data, and devices -- interact and combine with each other (in respect of the eco-laws and typically based on their spatial relationships), so as to serve their own individual needs as well as the sustainability of the overall ecology. In this data-centric approach, the agent

environment supports human/agent interaction and coordination by providing an open distributed set of data spaces, hosting streams of tuples -- generated by sensors, actuators, human actions and reactions -- semantically combined, aggregated, manipulated, and diffused according to the eco-laws.

[56] proposes "In good company", a distributed application for the food court of a shopping mall, that is based on SAPERE, The application enables people to spend some time with friendly persons or anyhow sharing common affinities. A typical use case scenario is the following: 1) a user running the application on its mobile phone approaches the mall's food court willing to launch "in good company"; 2) user's request for friendly locations is shared between the displays associated to a food provider of the court; 3) for each given restaurant, the display takes care of polling its costumers (using the app) to provide a measure of friendship affinity towards the requesting user; 4) each display aggregates such measures and pushes back the answer to the requesting user; 5) given such information the user can decide in which restaurant to have lunch and which group of people to join.

For this application, a SAPERE node with the app code is running on users' smartphones and restaurants' display stands. Different agents running on different devices interact with one another by sharing data via the spatial structure (see Fig. 3). For example, the restaurant Agent propagates the affinity query (AQ) – with a gradient indicating the number of hops and decay time – to surrounding displays. The agent environment regulates the distribution of data through *spread eco-laws* and *aggregation eco-laws*. This example shows how the agent environment can provide support for humans-in-the-loop interaction.

4.4 Challenges Ahead

Bringing humans-in-the-loop in MAS applications through a supporting agent environment is an open research topic. These kinds of systems pose complex challenges for an agent environment such as to how model humans, how to design a humans-aware communication infrastructure, how to provide decision and co-ordination support, and how to implement regulation mechanisms. We conclude this section by listing some of the key challenges we see in this exciting research area:

- Obtaining a comprehensive understanding of the spectrum of different types of human-in-the-loop functions in MAS. The levels of human involvement in computing systems provide a starting point.
- Defining and incorporating human models into agent environments to support humans in the loop in MAS, incl. positioning these models into the levels of support of the agent environment [7], or revision or extending the levels.
- Defining agent environment mechanisms and effective means for enabling interaction, coordination, cooperation not only among agents, but among humans and agents too.
- Understand the engineering implications of bringing humans in the loop in agent environments. This challenge includes identifying methodologies for designing and developing scalable agent environments for human-agent MAS, that integrate with the technology stack, e.g., Internet-of-Things and the cloud.

- Take an inter-disciplinary perspective, by bringing together researchers and expertise from both the human and the agent side, with the objective of designing mixed agent environments with agents and humans.

5 Roadmap

Fig. 4 shows the typical progressing levels of maturity to solve problems of computing systems over time [46]. Software/system engineers typically start by solving specific problems in a specific way. When problems recur, the expertise is turned into reusable solutions, for example in the form of frameworks or libraries. In the next stage, engineers abstract from concrete realizations and document design knowledge in the form of architectural approaches to solve the problems, such as tactics, patterns and reference solutions. Then, the knowledge is often consolidated in stable middleware solutions, offering developers programming abstractions and supporting infrastructure. Finally, language support is developed that provides an integrated solution to software developers.

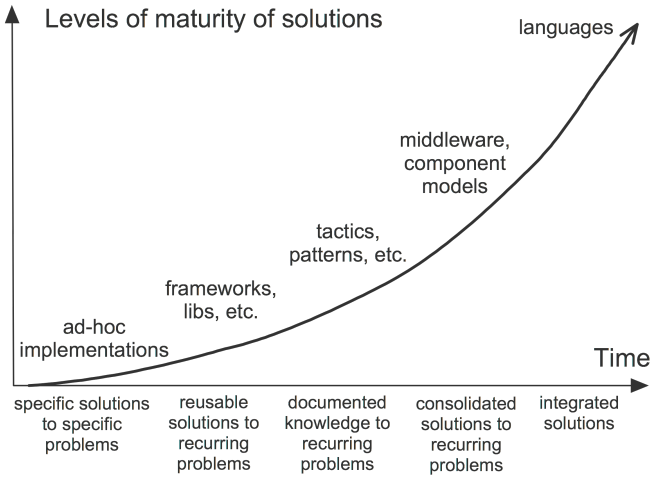

Fig. 4. Maturity levels of computing system solutions

In terms of Fig. 4, researchers and engineers have explored solutions for the different agent environment aspects we have discussed in this paper – large scale, openness, humans in the loop – at different levels. Most efforts have focused on solving specific problems with specific solutions, as testified in [2][3][4]. Some of these solutions have been consolidated in reusable frameworks, e.g., [53][54]. A few researchers have presented patterns to solve problems related to agent environment; examples are [47] with a set of patterns for self-organizing systems, and [48] presenting the results of a recent systematic survey of patterns applied in MAS. [52] presents an architecture framework for collective intelligence systems, comprising three viewpoints that support architects with designing agent environment for knowledge

sharing platforms that are based on stigmergic principles. Different middleware solutions and a few component models have been developed. Prominent examples are electronic institutions [49] and coordination artifacts [50]. Recently, some initial efforts have been done on programming support for agent environments, e.g., [51].

A closer look at existing work shows that most efforts are at lower levels of solution maturity, in particular ad-hoc implementations and frameworks. This is a natural situation for research that has been in an explorative stage. However, we believe that the time has come to balance exploration with consolidation. In Section 4 we have presented a variety of opportunities for exploratory research on agent environments for MAS. We conclude with complementary opportunities to consolidate research efforts:

- Perform empirical research to validate the claims of existing solutions of agent environments for multi-agent systems.
- Consolidate existing knowledge on agent environments for multi-agent systems; one effective way to do so is by performing a systematic survey of the state of the art in the field;
- Consolidate existing know-how on agent environments for multi-agent systems by documenting recurring solutions in the form of patterns, reference models and reference architectures;
- Define model problems and exemplars to drive and communicate research advances, establish research agendas, and compare and contrast alternative approaches.

Computing systems are increasingly intertwined with the surrounding world in which they are deployed and used. Furthermore, the growing dynamics, integration, and expanding scale of software-intensive systems calls for decentralization. The agent environment lies at the intersection of these two evolutions and will be more relevant for future computing systems than ever before. We hope that both the opportunities for further exploration and suggestions for consolidation may be a stimulus to further study, development and maturation of the field of agent environments in multi-agent systems.

References

[1] Weyns, D., Parunak, H.V.D., Michel, F., Holvoet, T., Ferber, J. Environments for multiagent systems state-of-the-art and research challenges, Volume 3374 of Lecture Notes in Computer Science., Springer (2005)

[2] Weyns, D., Parunak, H.V.D., Michel, F.: Environments for Multi-Agent Systems, First International Workshop, E4MAS 2004, New York, NY, USA, July 19, 2004, Revised Selected Papers. Volume 3374 of Lecture Notes in Computer Science., Springer (2005)

[3] Weyns, D., Parunak, H.V.D., Michel, F.: Environments for Multi-Agent Systems II, Second International Workshop, E4MAS 2005, Selected Revised and Invited Papers. Volume 3830 of Lecture Notes in Computer Sci- ence., Springer (2006)

[4] Weyns, D., Parunak, H.V.D., Michel, F.: Environments for Multi-Agent Systems III, Third International Workshop, E4MAS 2006. Selected Revised and Invited Papers. In: E4MAS. Volume 4389 of Lecture Notes in Computer Science., Springer (2007)

[5] Parunak, H.V.D., Weyns, D.: Guest editors' introduction, special issue on environments for multi-agent systems. Autonomous Agents and Multi-Agent Systems 14(1) (2007) 1–4
[6] Weyns, D., Omicini, A. Special Issue Engineering Environments in Multi-Agent Systems. Multiagent and Grid Systems 5(1) (2009) 1–131
[7] Weyns, D., Omicini, A., Odell, J.: Environment as a first class abstraction in multiagent systems. Autonomous Agents and Multi-Agent Systems 14(1) (2007) 5–30
[8] Platon, E., Mamei, M., Sabouret, N., Honiden, S., Parunak, H.V.D.: Mechanisms for environments in multi-agent systems: Survey and opportunities. Autonomous Agents and Multi- Agent Systems 14(1) (2007) 31–47
[9] Viroli, M., Holvoet, T., Ricci, A., Schelfthout, K., Zambonelli, F.: Infrastructures for the environment of multiagent systems. Autonomous Agents and Multi-Agent Systems 14(1) (2007) 49–60
[10] Valckenaers, P., Sauter, J.A., Sierra, C., Rodrıguez-Aguilar, J.A.: Applications and environ- ments for multi-agent systems. Autonomous Agents and Multi-Agent Systems 14(1) (2007) 61–85
[11] Helleboogh, A., Vizzari, G., Uhrmacher, A., Michel, F.: Modeling dynamic environments in multi-agent simulation. Autonomous Agents and Multi-Agent Systems 14(1) (2007) 87–116
[12] Buckley, J., Mens, T., Zenger, M., Rashid, A., and Kniesel, G. 2005. Towards a taxonomy of software change. Journal on Software Maintenance and Evolution: Research and Practice, 309–332.
[13] F. Zambonelli and V. Parunak, Signs of a revolution in computer science and software engineering. In *Proceedings of the 3rd International Workshop on Engineering Societies in the Agents World*. Lecture Notes in Computer Science, vol. 2577. Springer, 2003
[14] O. Shehory, Software architecture attributes of multi-agent systems. In: Proceedings of Agent Oriented Software Engineering, pp 77–90, 2000
[15] D. Weyns, Architecture-based design of multi-agent systems. Springer, Heidelberg 2010
[16] B. Cheng et al., Software Engineering for Self-Adaptive Systems: A Research Roadmap, Lecture Notes in Computer Science, vol. 5525, 2009
[17] P. Valckenaers, J. Sauter, C. Sierra, J. A. Rodriguez-Aguilar, Applications and environments for multi-agent systems, International Journal on Autonomous Agents and Multi-Agent Systems 14 (1), 2007
[18] M. Viroli, T. Holvoet, A. Ricci, K. Schelfthout, F. Zambonelli, Infrastructures for the environment of multiagent system, International Journal on Autonomous Agents and Multi-Agent Systems 14 (1), 2007
[19] E. Platon, M. Mamei, N. Sabouret, S. Honiden, H. Van Dyke Parunak, Mechanisms for environments in multi-agent systems: Survey and opportunities, International Journal on Autonomous Agents and Multi-Agent Systems 14 (1), 2007
[20] R. Haesevoets, D. Weyns, T. Holvoet, Architecture-Centric Support for Adaptive Service Collaborations, ACM Transactions on Software Engineering and Methodology (TOSEM), 23(1), 2014
[21] Weyns D., Architecture-Based Design of Multi-Agent Systems, Springer 2010
[22] P. Scerri, R. Vincent, and R. Mailler, Comparing three approaches to large-scale coordination. Coordination of Large-Scale Multiagent Systems, Springer US, 2006.
[23] M. Jamshidi. Large-Scale Systems: Modeling and Control. North-Holland Series in System Science and Engineering. North-Holland, 1983.
[24] A. Gouaïch, F. Michel and Y. Guiraud, MIC*: A Deployment Environment for Autonomous Agents, Environments for Multi-Agent Systems, Lecture Notes in Computer Science, volume 3374, Springer 2005
[25] S. Rodriguez, V. Hilaire and A. Koukam, Holonic modeling of environments for situated multi-agent systems. Environments for Multi-Agent Systems II, Lecture Notes in Computer Science, volume 3830, Springer 2006.

[26] D. Weyns, K. Schelfthout, and T. Holvoet, Exploiting a Virtual Environment in a Real-World Application, Environments for Multi-Agent Systems II, Lecture Notes in Computer Science volume 3830, Springer 2006
[27] H. Van Dyke Parunak. A survey of environments and mechanisms for humans-human stigmergy. Environments for Multi-Agent Systems II, Lecture Notes in Computer Science volume 3830, Springer 2006
[28] M. Mamei and F. Zambonelli, Motion coordination in the quake 3 arena environment: a field-based approach. Environments for Multi-Agent Systems, Lecture Notes in Computer Science, volume 3374, Springer 2005
[29] H. Van Dyke Parunak, Sven A. Brueckner, John Sauter, Digital Pheromones for Coordination of Unmanned Vehicles, Environments for Multi-Agent Systems, Lecture Notes in Computer Science, volume 3374, Springer 2005
[30] V. Urovi, A. C. Olivieri, S. Bromuri, N. Fornara, and M. I. Schumacher. A peer to peer agent coordination framework for IHE based crosscommunity health record exchange. 28th ACM Symposium On Applied Computing, SAC 2013
[31] V. Urovi, A. C. Olivieri, S. Bromuri, N. Fornara, and M. I. Schumacher, Secure P2P cross-community health record exchange in IHE compatible systems, International Journal on Artificial Intelligence Tools, IJAIT 2013
[32] A. Omicini and E. Denti, From tuple spaces to tuple centres, Science of Computer Programming, 41(3):277-294, 2001
[33] S. Puricel, S. Bromuri, J. Krampf, L. Diolosa, J. Puder, C. Montreuil, M. Schumacher, and J. Ruiz, Telemedical outpatient monitoring and management of gestational diabetes mlitus by the g-demande system: A randomized controlled feasibility study (tele-gdm). In Diabetes Technology and Therapeutics, volume 16, 2014
[34] F. Michel, Translating agent perception computations into environmental processes in multi-agent-based simulations: A means for integrating graphics processing unit programming within usual agent-based simulation platforms. Systems Research and Behavioral Science, 30(6), 2013.
[35] S. Munir and J. Stankovic and C. M. Liang and S. Lin, Cyber Physical System Challenges for Humans-in-the-Loop Control, 8th International Workshop on Feedback Computing, 2013
[36] M. Cumming, Supervising automation: humans on the loop, Aero-Astro, MIT Aeronautics and Astronautics Department, Massachusetts Institute of Technology 2008
[37] A Wood, J. Stankovic, G. Virone, L Selavo, Z. He, Q. Cao, T Doan, Y. Wu, L. Fang, and R Stoleru, Context-Aware Wireless Sensor Networks for Assisted Living and Residential Monitoring. IEEE Network 22, 4, 2008
[38] W. Li, D. Sadigh, S. S. Sastry, and S. A. Seshia, Synthesis for Humans-in-the-Loop Control Systems,
[39] L. F. Cranor, A framework for reasoning about the human in the loop. *Conference on Usability, Psychology, and Security*, UPSEC 2008
[40] F. Lancelot, M. Causse, N. Schneider, M. Mongeau, Humans-in-the-Loop Multi-agent Approach for Airport Taxiing Operations, Trends in Practical Applications of Agents, Multi-Agent Systems and Sustainability, Advances in Intelligent Systems and Computing Volume 372, 2015
[41] Claes, R., Holvoet, T., Weyns, D.: A decentralized approach for anticipatory vehicle routing using delegate multiagent systems. IEEE Transactions on Intelligent Transportation Systems 12(2), 2011
[42] N. Schurr, J. Marecki, M. Tambe, P. Scerri, The Future of Disaster Response: Humans Working with Multiagent Teams using DEFACTO, AAAI Spring Symposium on AI Technologies for Homeland Security, 2005
[43] J. M. Bradshaw, P. Feltovich, M. Johnson, Humans-Agent Interaction, In The Handbook of Humans-Machine Interaction: A Humans-Centered Design Approach, 2011

[44] Zambonelli, F.: Toward sociotechnical urban superorganisms. IEEE Computer 45(8), 76–78 (2012)
[45] F. Zambonelli, G. Castelli, L. Ferrari, M. Mamei, A. Rosi, G. Di Marzo, M. Risoldi, A. Tchao, S. Dobson, G. Stevenson, J. Ye, E. Nardini, A. Omicini, S. Montagna, M. Viroli, A. Ferscha, S. Maschek, B. Wally, Self-aware Pervasive Service Ecosystems, Procedia Computer Science, Volume 7, 2011
[46] D. Weyns, M. Caporuscio, B. Vogel, A. Kurti, Design for Sustainability = Runtime Adaptation U Evolution, Sustainable Architecture: Global collaboration, Requirements, Analysis, SAGRA 2015
[47] L. Gardelli, M. Viroli, A. Omicini, Design Patterns for Self-organising Systems, Lecture Notes in Computer Science Volume 4696, 2007
[48] J. Juziuk, D. Weyns, T. Holvoet, Design Patterns for Multi-Agent Systems: A Systematic Literature Review, Research Directions in Agent-Oriented Software Engineering, Springer, 2015
[49] D. De Jonge, B. Rosell, and C. Sierra. Human interactions in electronic institutions. In *AT*, 2013
[50] A. Omicini, A. Ricci, M. Viroli, C. Castelfranchi, and L. Tummolini, Coordination Artifacts: Environment-Based Coordination for Intelligent Agents. Third International Joint Conference on Autonomous Agents and Multiagent Systems, 2004
[51] A. Ricci, M. Piunti, and M. Viroli, Environment programming in multi-agent systems: an artifact-based perspective. *Autonomous Agents and Multi-Agent Systems* 23, 2, 2011
[52] J. Musil, A. Musil, D. Weyns, S. Biffl, An Architecture Framework for Collective Intelligence Systems Working International Conference on Software Architecture, WICSA 2014:
[53] J. Sauter, R. Matthews, H. Van Dyke Parunak, and S. A. Brueckner, Performance of digital pheromones for swarming vehicle control. Fourth international joint conference on Autonomous agents and multiagent systems, AAMAS 2005
[54] A. Ricci, M. Viroli, and A. Omicini, CArtAgO: a framework for prototyping artifact-based environments in MAS. Environments for multi-agent systems III, E4MAS 2006
[55] James Odell, H. Van Dyke Parunak, Mitchell Fleischer: The Role of Roles, Journal of Object Technology, vol. 2, no. 1, 2003
[56] G. Castelli, M. Mamei, A. Rosi, F. Zambonelli, Developing Social Applications in SAPERE, IEEE 10th International Conference on Ubiquitous Intelligence & Computing and IEEE 10th International Conference on Autonomic & Trusted Computing, 2013