Autonomous Agents in Organized Localities Regulated by Institutions

Michaela Huhn, Jörg P. Müller, Jana Görmer, Gianina Homoceanu, Nguyen-Thinh Le, Lukas Märtin,
Christopher Mumme, Christian Schulz, Niels Pinkwart, and Christian Müller-Schloer
Niedersächsische Technische Hochschule
Braunschweig, Clausthal-Zellerfeld, Hannover, Germany
Email: {Michaela.Huhn|Joerg.Mueller|Jana.Goermer|Christopher.Mumme|Nguyen-Thinh.Le|Niels.Pinkwart}@tu-clausthal.de,
Cms@sra.uni-hannover.de, Schulz@rts.uni-hannover.de, G.Homoceanu@iti.cs.tu-bs.de, Maertin@ips.cs.tu-bs.de

***Abstract*—This paper proposes a new metaphor for constructing systems of systems: Autonomous Agents in Organized Localities (AAOL). An agent-based approach is used for modeling structure and behavior of complex systems that consist of (semi-)autonomous systems, where goals, resources, capabilities are described locally while a need for superordinated "global" regulation exists. The notion of organized localities is used to describe spatially or logically constrained spheres of influence of regulation bodies. Agents inhabit – and can move across – localities; regulation rules are modeled via computational norms and enforced by electronic institutions. A key objective of our work is to explore and advance applicability of AAOL to constructing mechatronic systems with (at least soft) real-time constraints. We describe requirements for modeling systems of systems, and outline the key pillars of AAOL: a conceptual architecture and a metamodel providing the basic constructs for describing AAOL-type systems. A case study of a decentrally organized airport transportation infrastructure illustrates the concepts and the feasibility of AAOL-based systems of systems design.**

Meta model; Organized Locality; Regulations mechanisms (key words)

I. Introduction

Recent advances in coupling heterogeneous computing devices and software applications have stimulated several innovative fields like autonomous robotics, ambient assisted living, or enhanced civil infrastructure. Such *systems of systems* are characterized by loosely coupled, software-controlled subsystems that cooperate to achieve joint goals. Each subsystem operates semi-autonomously in order to pursue individual tasks, but it also obeys the current constraints within its local environment. We assume subsystems are developed independently of the purposes and unifying requirements of an entire system.

Digital Ecosystems [1] address similar issues in the areas of enterprise application integration, social networks, and virtual communities. Digital Ecosystems combine features from Multi-Agent Systems (MAS), Autonomic Computing and Service-Oriented Architectures to develop and evolve robust, scalable systems with a high degree of self-management. However, with the integration of *physically acting* mechatronic subsystems and the growing complexity of joint tasks, the need for "semantically rich abstract levels of description" [2] increases. The computing entities shall temporarily co-operate in a certain scenario; in addition, they shall be deployable in another context with different environmental constraints and tasks in a plug'n'play like manner. To meet these challenges, social concepts like organizations, institutions and norms have been proposed in the area of MAS [3], [4]. Social concepts are a means of explicit representation of global objectives and constraints and of their relation to the level of interacting groups and even to individuals with their beliefs, desires, and intentions (BDI). They build a basis for self-insight, self-management, and finally self-adaptation of such systems of systems. But until now, the different aspects of social interaction and regulation have been mainly considered in isolation. So, open questions are 1) Which are key concepts needed to balance between the global objectives and the autonomy of individual actors within a system of systems? 2) What are the specific requirements induced by the mechatronic subsystems that have to be incorporated to pave the way towards cyber-physical systems? 3) What are adequate architectural means to found the design of subsystems upon?

Here, we present a conceptual metamodel and architecture for Autonomous Agents in Organized Localities (AAOL) to facilitate the model-based development of cyber-physical systems of systems. Localities capture the idea of a restricted sphere of influence and environmental constraints in which semi-autonomous agents cooperate under the control of centralized regulation bodies, called institutions. As a first proof of concept, an airport transportation scenario has been designed according to the AAOL concepts and realized on a simulation platform. The paper is structured as follows: In Sec. II, key viewpoints of systems of systems are discussed. Our approach is introduced in Sec. III, the AAOL metamodel is presented in Sec. IV. Sec. V describes the airport departure scenario. Sec. VI concludes and discusses future research directions.

II. Dimensions of Systems of Systems

In this section, we discuss four distinct aspects of systems of systems, resulting requirements, and related work.

This work was funded by the NTH Focused School for IT Ecosystems (www.it-ecosystems.org). NTH (Niedersächsische Technische Hochschule) is a joint university of Technische Universität Braunschweig, Technische Universität Clausthal, and Leibniz Universität Hannover.

A. Representation of the Environment

As we are interested in the application of our approach to technical systems featuring physical components, below the requirements and structure of a typical environment for embodied participants (e.g., robots) are described. In order to perceive the environment, the participants require sensors (e.g., cameras). In general, the scanned data underlies some uncertainties, which have to be considered in the preprocessing (data reduction, filtering) of the raw data. Moreover, reactive behavior (e.g., obstacle avoidance) is a very important requirement for accomplishing tasks in the environment. Complex behaviors (e.g., navigation) require an internal representation of the environment. It is possible either to provide predesigned maps or to create maps from sensor data at runtime (e.g., SLAM) [5]. Different kinds of maps are e.g. grid-based, feature-based, and topological maps. To perform navigation tasks (e.g., path planning), a participant needs to know its relative position in the environment. Since the uncertainty of sensor data adds up over time, proper localization algorithms (e.g., Monte-Carlo localization [6]) become necessary. All processing steps underlie real time requirements, the steps need to be finished during a fixed time interval, as they finally lead to the control of motors, which result in a movement of driving wheels, manipulators, sensor rotation, etc.

As participants can manipulate the environment and every change may have significant effects on the whole system, these issues cannot be handled by predefined, constant mechanisms [3]. Since these difficulties transfer to digital ecosystems, and a centralized regulation and control is not applicable, our approach aims at regulation mechanisms, which are centrally triggered but accomplished in a decentralized fashion.

B. Autonomous Systems and Agents

The environment surrounds all moving participants and static objects (e.g., roads) in a system. Participants can be modeled by the concept of *agents*. Agents are capable of carrying out actions autonomously to achieve goals [7]. In [8], [9] architectures to model agents have been proposed.

An agent perceives selective parts of the its environment to gather significant information and to interpret this as actual *world state*. Based on its world state the agent draws conclusions about a situation and makes decisions to select appropriate actions for achieving its goals. The execution of actions by the agent's actuators will change the environment; these changes have to be re-perceived. This continuous process of perceiving, analyzing, selecting and executing can be realized by *self-management*. Hence, autonomy and self-management capabilities enable agents to act solely in unknown environments. However, the autonomy of agents counteracts the controllability of groups of agents. To meet this challenge, we have to consider adequate decentralized coordination mechanisms.

C. Coordination Mechanisms

In order to solve complex tasks in a shared environment, agents need to coordinate their activities, form groups and cooperate within organizations through appropriate interaction mechanisms. Due to decentralized system design and possible different roles of each agent, conflicts between agents or organizations can arise. MAS [7] are a promising paradigm for the construction of systems of systems. A MAS is a system consisting of multiple agents having a large number of structural variations (heterogeneous) or multiple identical agents (homogeneous) that interact with each other. Interaction often uses communication and implies that agents coordinate their activities to solve a problem or to reach common goals. A common model of communication is speech-act based: (1) Agents are able to make a decision for action selection, (2) communication is a type of action, and (3) it carries a semantic meaning that has to be understood by agents (e.g., KQML or FIPA ACL). Therefore agents have a relationship with each other and form an organization [10]. Roles are the main focus that agents take within the organization and those associate agents with one another. A role comprises the constraints (requirements, skills) that an agent has to satisfy when adopting a role behavior, the benefits (abilities, authorization, profits) that an agent receives in playing that role, and the responsibilities associated with it [10].

External agents can see an organization as a single entity. An organization can therefore be part of another organization, resulting in a hierarchical structure of organizations and agents [11]. Coordination entails temporal "harmonization" of activities. It takes place independent of individual goals of the agents and comprises control mechanisms for working together. Coordination mechanisms avoid ineffective behavior by reducing competition for resources [7] and by dealing with different types of conflicts.

D. Regulation Mechanisms

Even with autonomous or loosely coupled adaptive systems, the purposes, guarantees, and restrictions are usually considered from a global or regional perspective. Central questions are: 1) How are the global objectives achieved via the participating systems? 2) How are agents designed (a priori), or selected or adapted (at runtime) that are compliant to the system-level rules valid in a restricted situation in their structure and behavior? 3) How can adaptations on global or regional objectives be achieved? These issues are answered differently in distinct areas:

In Organic Computing [12] and swarm intelligence, individual subsystems cooperate locally to solve global optimization problems without central control. Self-optimizing mechatronic systems [13] propose a layered architecture called *Operator Controller Module* with a central *cognitive controller* that is able to react on changes in the environment or the system objectives by reconfiguring the subsystem controllers. Distributed constraint satisfaction [14] and game theory express the global objectives via consistency predicates or equilibria.

Organizational MAS [4], [15] aim at handling compound tasks like search and rescue scenarios. As the subtasks and situations differ significantly, an *organization* adopts the structural and behavioral interaction patterns for agent coopera-

tion. The organization possesses the capability and workflow models that allow to reason on how global constraints and objectives can be met in particular locally, time-based or resource-restricted situations. They assign roles to agents and monitor them.

Normative or institutional MAS [16] is a new direction that transfers social and judicial concepts to open communities of software agents. Trusted institutions administer norms, which specify the rules the agents shall obey to, and impose them by enforcement or incentives. Norms are a powerful and flexible means to achieve guarantees on both, the global level in terms of preferred equilibria status, as well as on the individual level like safety, fairness or progress assertions.

III. Autonomous Agents in Organized Localities

Based on the requirements from Sec. II, in the following we elaborate an approach to organize complex systems.

A. Representation of the Environment

In this work, we consider our approach in a simulated environment. Therefore, the complexity of the data provided by the sensors as well as the control of the actuators is considerably reduced as compared to a real world scenario (see Sec. II). Here, the sensor perception corresponds to the knowledge of a small part of agents' proximity (e.g. content of adjacent cells in a grid map) and we are able to choose the proper abstraction level of the actuators, such as "move one cell north". Additionally, communication can be implemented with a guaranteed exchange of messages, which avoids various difficulties compared to a real hardware implementation.

Either way, due to the complexity of the system, the environment in which the agents are situated is not uniform in terms of its functional structure and its requirements. Different areas of the environment may require different agent behavior. This division could be spatial (due to e.g. different environmental configurations) but also semantic (e.g., different communication protocols for the interaction between the agents or with the objects). For this purpose we decompose the locality in to several scenes. Each scene is characterized by constraints which may take effect on different levels of the system. These requirements lead to our approach of AAOL.

An organized locality can be understood as a physical or virtual place offering a number of opportunities. It has a scope defining a boundary, so systems may enter, leave, and return later to the locality. Further a locality may provide organizations to foster coordination. It is associated with institutions to regulate the interaction of autonomous, heterogeneous agents beyond physical and technical constraints. Institutions regulate the agent behavior in order to balance between different interests and to establish and sustain certain notions of stability. Organizations structure the grouping and collaboration of agents within the locality. Both are introduced in Sec. III.C and III.D.

Examples of localities are an airport (see Sec. V), a fair, or a social network. The infrastructure is part of a local ontology representing domain knowledge. A locality sets the stage for a possibly hierarchically structured collection of scenes describing pre-defined interaction templates for specific coordinated activities or to achieve certain subgoals.

B. Autonomous Systems and Agents

AAOL defines the locality as a virtual infrastructure to be used by the agents to achieve goals related to the subject of the locality.

We propose an *agent architecture* with four layers: *Social Context Layer* (SCL), *Individual Context Layer* (ICL), *Execution Layer* (EL), and *Mechatronic Layer* (ML) (see Fig. 1). Agents perform predefined atomic or sequenced (plans) actions related to their goals. Goals and plans are potentially spread among multiple agents (joint goals/plans). Each layer

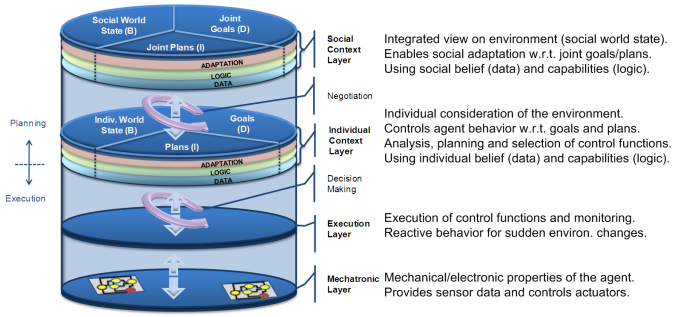

Fig. 1. AAOL Agent Architecture

has an authority. If this authority cannot handle a case, a request to an upper layer is performed. In case the EL detects an unresolvable problem, a request to the ICL is issued. For multi-agent problems the ICL has to activate the SCL to preform communication for negotiation. The upper layers solve the problem and reactivate the lower layers again, e.g., the EL will be re-activated to perform the actions. Hence, information flow for planning in bottom-up and for execution in top-down direction. The ICL is in focus of the decision-making process. Decisions are based on the information from the EL reflected to the individual world state and the goals of an agent. Such feedback control problems can be implemented, e.g., by the MAPE cycle approach [17].

The ICL and SCL are partitioned into three segments according to a BDI architecture (cf. [8]). *Belief* is perceived as the world state. In the SCL, this state is restricted by norms and in the ICL it is derived from the behavior of other agents in the environment. Goals of agent groups (social context) or of single agents (individual context) are classified as *Desires*. *Intention* includes specific plans for realizing individual/joint goals, respecting the environmental restrictions.

If multiple agents act in the same locality, joint tasks have to be coordinated and resource conflicts need to be solved.

C. Organizations in Multi-Agent Systems

In the agent architecture described in Fig. 1 organizations are located in the SCL and can be seen as computational methods inspired by concepts from economy and sociology that appear as one entity in the locality based upon social

and functional distinctions and roles amongst individuals. Organizations can also be structured hierarchically e.g. by providing certain agents with more authority than others through role definitions. A peer-to-peer architecture is any distributed network composed of agents that make a portion of their resources directly available to other agents, without the need for central coordination instances. Peers are both suppliers and consumers of resources, in contrast to the traditional client-server model. The fully connected architecture has a general form of a chief director usually forming the single well-informed element, the so-called "voice" of an organization to the outside, to sub-division managers and to the workers. A group can be seen as a specialized entity (or subsystem) of an organization, usually consisting only of one leader and workers to reach a common goal or achieve a joint plan. For this, communication, negotiation and conflict resolution is connecting the individual with the social context layer. The connections between the agents with different roles imply interaction guaranteeing the service of the localities, but this may lead to conflicts between agents which need to be handled. In conflict cases, agents should be able to detect, analyze and solve conflicts. In addition, they should be able to learn how a conflict occurred to avoid future conflicts.

D. Regulation Mechanisms

As stated in Sec. III.A an institution is associated with a locality. It provides normative regulations (norms) and mechanisms to establish or to ensure their compliance. It acts through an organization that executes institutional tasks. The tasks contributing to norm compliance are: 1) An *information service* administers the identities of agents currently present in the locality and provides them with knowledge about the current norms, 2) *Norm monitors* monitor whether the agents behave according to the norms based on the information gathered from *observers*, 3) A *norm enforcement* guarantees that control is imposed on the agents participating in the locality in such a way that they will behave norm-compliant to assure vital global objectives and the safety of individuals.

Norms are an explicit description of the regulations that govern the agents' behavior in the locality for the benefit of the community and itself as a member of it. Norms are defined in a top-down manner by the institution. Our approach assumes that agents are able to understand them.

IV. AAOL Metamodel

In this section, we describe a metamodel which provides the constructs (and their relationships) for modeling systems based on the AAOL concept.

A. Representation of the Environment

In order to provide the structure of the localities, we need a representation of the environment, which enables a proper association between the specific regulation mechanisms and the localities. As stated in Sec. 3A, our approach is to divide a locality into several scenes (e.g. different areas in an airport, see Sec. V). The institutions which are associated with a locality, provide regulation mechanisms within the scope of a specific scene (see Fig. 2). The division into

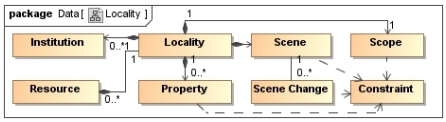

Fig. 2. Metamodel of a Locality

scenes can be motivated by various tasks rules, processes, requirements, properties, constraints or resources (e.g. sensor properties, movement constrains or energy resources). Within these scenes, associated sets of norms are used to regulate the behavior and interaction of the agents. According to this, the agents need an internal representation of the context which is relevant in the specific scope.

B. Autonomous Systems and Agents

For behavioral specifications for agents we introduce a metamodel regarding to our architecture (see Fig. 3). The

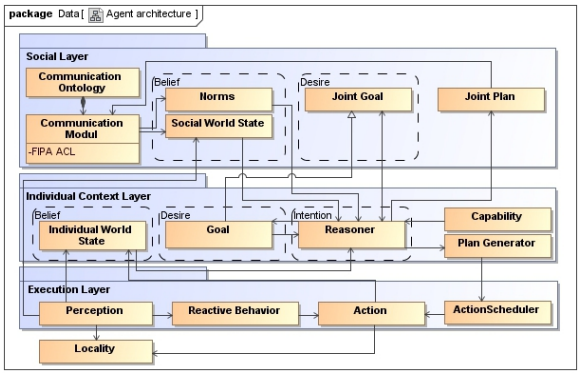

Fig. 3. Metamodel of the Agent Architecture

metamodel breaks down the concepts of layers and BDI to classes. Classes are responsible for specific parts of the agents' behavioral specifications. The reasoner is the central decision instance in the ICL, responsible for decision making based on agent capabilities, (joint) goals, (joint) plans, norms, and the individual/social world states.

From this point on, the ML is neglected, since the behavioral specification is seen in context of a virtual environment in a simulation. Instead, the ECL is directly linked to the locality. Raw sensor data and control commands for actuators are preprocessed in an appropriate way.

We extend our metamodel to describe further details of simulation platforms. An agent consists of an agent architecture, capabilities and behaviors (see Fig. 4).

Further relations of an agent in the metamodel are described according to [18]: an agent has accesses to a set of resources (information, knowledge, ontologies, etc.) from its environment, i.e., the locality. Furthermore, an agent has goals and is able to take on locality roles (to act in accordance to a plan) and behaviors, which are represented by the agents'

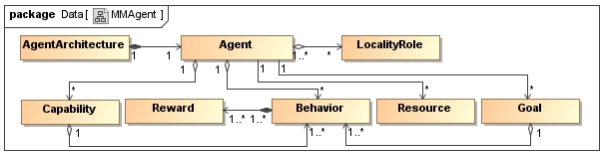

Fig. 4. Metamodel of an Agent

capabilities. The coordination of agents in organizations is also reflected in our metamodel.

C. Metamodel of Organizations

Fig. 5 is an extension of [18] and shows the metamodel of organizations. It includes the concept of an *Organization* and its *Structure*, *Group* and its *Context*, *Institution* and *Norm*, *Binding*, *InteractionUse*, *ActorBinding*, *Interaction* and its *Protocols* for *Communication* and *Coordination*, *LocalityRole*, *Actor* and *Agent* as well as *Capability* and *Resource* (from the agent aspect). An organization is derived from the agent perspective and it inherits characteristics of an agent [18], i.e. capabilities which can be performed by its members. A *Group* is a special kind of an organization that is bound by a *Group context*. The *Structure* defines the pattern of the organization. It can bind agents or organizations to the *LocalityRole*. *Interaction* in an organization has internal protocols that specify how its members communicate with each other and coordinate their activities. For interaction, *LocalityRole*s are bound to *Actor*s (by *ActorBinding*) that can be considered as representative entities within the corresponding interaction protocols. Thus, an actor can be seen as an agent (or organization) with a *Role* and a task. A role defines the behavior of an agent in

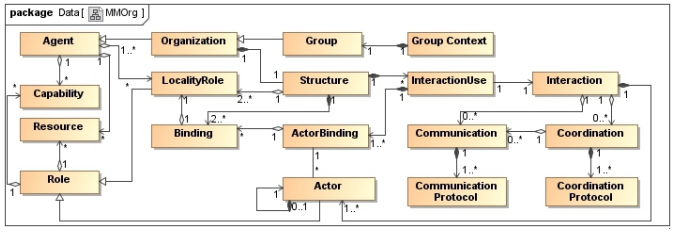

Fig. 5. Metamodel of Organizations and Roles

a given context (e.g., an organization). Therefore it provides an agent with capabilities and a set of resources it has access to. An actor can be considered as a generic concept and either binds instances directly or through the concepts LocalityRole and Binding. The set of bound entities could be further specialized through the subactor (specialization of the superactor) reference that refers again to an actor.

D. Regulation Mechanisms

The conceptual metamodel for the normative aspect of an institution is illustrated in Fig. 6. A norm consists of a *normative statement* expressing the regulation imposed, a set of subjects to which the normative statement applies, and the scope that may restrict its area of applicability to particular scenes or situations. The norm also has a specification of *fulfillment/violation* for monitoring norm compliance, and the consequences of norm compliance in terms of positive/negative *rewards*. A set of norms is structured in a *normative structure* based on which the classification scheme is realized. It classifies the norms with respect to abstraction levels, a priority scheme or context information. We distinguish different types of norms, namely *obligations*, *prohibitions*, and *permissions*. Permissions specify the allowed actions to be performed by the agent while obligations and prohibitions restrict its behavior. The agent interprets prohibitions (restrictions) as guidelines which it may decide to violate. Obligations are used in case norm-compliant behavior is considered mandatory, due to its high criticality either with respect to the system-level objectives or to the dependability issues of individuals. Thus, obligations must never be violated and they are enforced on the agent. The norm enforcement in MAS is

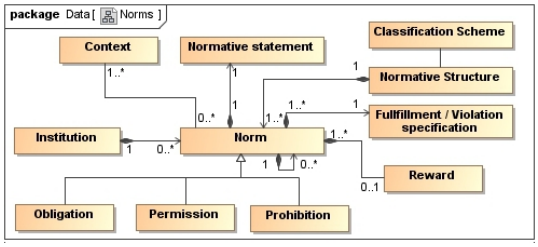

Fig. 6. Conceptual Model of Norms

done under one of two assumptions: the designer can control the actions agents realize in the system and can stop forbidden actions before they take place by implementing a trusted component mediating all interactions [19] or only the agent controls its actions and the regulation is done after the action has been performed by using sanctions/rewards [16]. The main disadvantage of the first assumption is the compelling amount of communication overhead for the trusted component (agent governor) for mediating all agent's interactions. The second assumption allows agents to act selfishly and to violate norms. Thus, obligations cannot be enforced strictly. For our approach we have combined elements from both assumptions. We use institutional agents (IAs), which act preemptively on agents only in case of obligations. At each step, the IAs compute a list of candidates of agents, for which an obligation applies. For each candidate, the IAs then identify forbidden actions, from the list of possible actions defined at design time. Only at this moment the IA acts and restricts the candidates from performing the forbidden actions. The other types of norms are handled by means of rewards and sanctions due to the second assumption.

E. Process (Integration of the Four Dimensions)

The integration of the four dimensions is illustrated in Fig. 7. 1) The *Environment* is represented as the *Locality* which is scanned by the agent and he performs action inside. 2) The *Agent* has an architecture with an *Execution Layer* where the agent is connected to the *Locality* and *LocalityRole* is

allocated to the agent. (3) The *Organization* is the connection between the *Agent* and the *MAS* where it is embedded together with *Institution*. Both refer to 4) our *Regulation mechanisms* where *Conflict handling* and *Norm handling* are listed as first implementations.

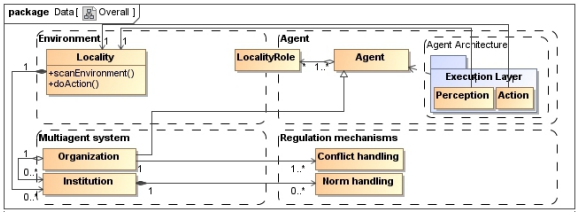

Fig. 7. Overview of the four Dimensions

V. The Airport Scenario: An AAOL Case Study

Next we show how the AAOL approach is used to model and implement a prototypical simulation of a transport infrastructure at an airport.

A. Scenario Description

We consider autonomous transport vehicles (ATVs) in the departure area of an airport. The ATVs transport passengers between stopovers/stations (passenger entrances, check-in counters, gates, and plane parking positions). ATVs consume energy while driving on roads, which have different capacities (one/two lanes) and speed limits, so they have to recharge their batteries at a charging station (CS). (see Fig. 8). All ATVs

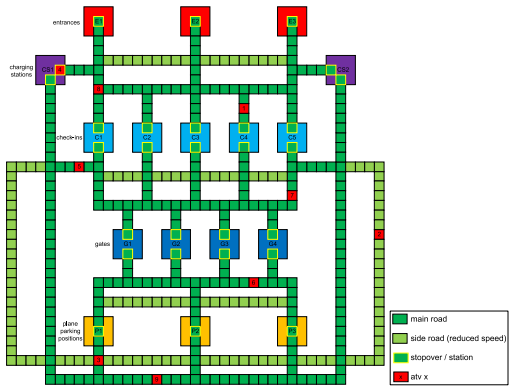

Fig. 8. Road Grid of the Airport

know the airport map and stopovers/stations. Furthermore they can perceive other ATVs that are within their horizon and have to make decisions based on gathered information. According to our agent architecture, ATVs perform (joint) tasks/plans (e.g., transport passengers) and have (joint) goals (e.g., save energy). On successful/failed transport task executions, ATVs are getting positive or negative rewards.

Vehicles are signed in a service that collects passenger orders and offers tickets (pickup/drop positions, times) to the ATVs. This leads to competition on tickets, roads and CSs, so ATVs have to negotiate. A communication grid and conflict solving mechanisms address these problems. To prevent the occurrence of unsafe situations caused by a selfishly acting ATV we need to implement a mechanism to balance agent autonomy and system controllability according to our meta-model (see Sec. V.E).

B. Simulation Platform

The airport scenario is implemented on a scalable simulation platform integrating JADE and Repast Simphony called JRep. Repast targets agent-based simulation and an efficient visualization of effects on the system of system-level, whereas JADE is meant for building complex interacting agent communities. JADE supports FIPA(-ACL) and interaction protocols.

The airports objects (roads, entrances, check-ins, gates, plane parking positions) are implemented in REPAST, whereas the agents (ATV, CSs) are realized in JADE using behavior components to build an agent architecture according to Fig. 3.

C. Representation of the Environment

The virtual infrastructure is called context in Repast and is defined by a set of environment elements (e.g., roads). These elements are arranged at design time. In contrast to a real world scenario, a registry is necessary where active agents (ATVs) have to sign in/off while entering/leaving the scenario at runtime. Additionally, some static elements (stations, stopovers) register as static agents at initiation of the simulation to enable interactions.

The registry stores references of agents to access their properties (e.g., positions). With a request to the registry this replaces the process of environmental scanning. Thus, an agent can recognize close by agents. If there is another agent, the registry provides its reference to enable a connection establishment between the agents. The registry does not coordinate the agents centrally, but coordination and conflict handling are done by direct communication between agents. Manipulations of the environment are based on interactions among agents. Complex interactions between agents (e.g., for negotiations) presuppose communication. Based on the JADE framework, we implemented an ontology for the airport scenario to support ACL-based communication.

In the registry, localities and scenes are represented as predefined or spontaneous mappings of agents to sets of norms and infrastructural restrictions. The locality role of an agent restricts some behavioral aspects of it. Organization can be formed by subsets of agents (grouping).

Institutions are also realized as static agents, but not visible in the infrastructure. An institutional agent defines norms for its scene in a locality. It got unrestricted access to the agent references in the registry to observe behavior, enforce norms, and give rewards.

The representation of the environment and allocation of agents is possible in our simulation platform with the help of the registry. Our concept of localities is implemented by flexible assignments of agents to organizations. Further work is necessary to support changes of static objects to restructure the infrastructure at runtime.

The abstract agent class is extended comprehensively with the implementation for active agents in the system.

D. Autonomous Systems and Agents

The implementation of an agent is based on the metamodel from Sec. IV.B. Layers are represented by packages, the BDI structure is reflected to sub packages, and elements (e.g., the reasoner) are classes.

Our scenario is limited to human-controlled and autonomous agents. The implementation for human-controlled agents allows for the direct control of an agent at runtime.

The reasoner is primarily responsible for the agent behavior. For modeling and validation of the reasoner specification, we apply a variant of UML 2 Activity Diagrams [20]. The structure of the reasoner builds on the MAPE cycle [17]. We provide tool support for simulation of the airport scenario that allows us to monitor and validate the agent behavior (e.g., negotiation processes) in selective views upon the abstract level of graphical models. Our implementation is expendable by further functionality for automated testing and validation.

E. Conflict Model

In the airport scenario, many types of conflicts might occur. At present, we only focus on resource conflicts, i.e., two or more ATVs compete for a resource. The following conflicts can happen in this scenario: 1) two ATVs are approaching a road junction, 2) an ATV needs to be recharged while many others are waiting to be served by the CS, and 3) ATVs have to take passengers to unoccupied check-in counters. The resource which an ATV will consume in these three examples is a road junction, a CS, or a check-in counter. All of them have limited capacity. To detect potential resource conflicts, the approach followed in this paper makes use of agents' belief information about its world state (cf. Sec. 2B). That is, each ATV has a *Belief* about the last, current, and next possible position of other ATVs existing within its scope.

Let B be an ATV which exists in the scope of an ATV A. The next possible position of B can be determined using information about its current and last positions. The calculation of the next position is based on the assumption that when an ATV is moving, it is probable that this ATV will move straight forward. Using X and Y coordinates to describe the position of elements within an airport, the difference between the last and the current X coordinate is determined as follows: $diff\_X(B)$=$|last\_X(B) - current\_X(B)|$. As a result, the next X coordinate of B will be $next\_X(B) = current\_X(B) + diff\_X(B)$, and similarly, the next Y coordinate of B can be determined. The next possible position of B is composed of $next\_X(B)$ and $next\_Y(B)$. To enable the calculation of next possible positions for other ATVs within the scope of ATV A, the *belief* of A about the current and last position of these ATVs needs to be updated dynamically. For this purpose, A keeps trace of all known ATVs, i.e., ATVs which have been in the scope of A once, using a collection. If an ATV B occurs in the scope of A at the first time, B will be added to the collection and the last position will become the current position. Other ATVs, which already exist in this collection, will be updated with its last and current positions.

Let $\langle X, Y \rangle$ be the position of the airport map element which an agent A will move to, (i.e., consume). Based on the *Belief* about next possible positions of other ATVs, A is able to collect the number of other ATVs in its collection which will leave/enter this map position. ATV_{Env}, ATV_{Leave}, and ATV_{Enter} are the sets of ATVs which are occupying, will leave, and will enter the map position required by A, respectively. A has a potential conflict iff $|ATVs_{Env}| - |ATV_{Leave}| + |ATV_{Enter}| + 1 > C$, where C is the capacity of the airport map element $\langle X, Y \rangle$. Since A also has to be considered an ATV that will consume the map element being observed, the condition for checking potential conflicts must include the added value 1. In order to collect ATVs which are occupying, will leave, and will enter the map position required by A, we apply first-order unification. This mechanism allows us to process the *Belief* of a huge amount of ATVs with less computational resource than having to iterate through each of them. Our conflict detection mechanism can be used to detect the mentioned types of conflicts and is the base for conflict handling (regional regulation mechanisms, cf. Fig. 7).

F. Norms for ATVs

We implemented one institution so far that provides traffic norms and mechanisms for regulating the ATV's behavior. The norms are stored in a database and checked a priori for consistency. The institution provides the following services: 1) *Information service* is a JADE agent that provides methods to send norms established at the systems' design time and uses a protocol class to play the initiator role (JADE behavior) while the ATVs use a responder class which has (as a parameter in the constructor) a *MessageTemplate* used to select the protocol initiation message from the initiator. For each registered agent, its *AgentRepresentation* (belief of how the agent behaves) is updated regularly with the actions the agent performs, as observed by an IA, as well as the next possible actions, 2) *Norm monitor* checks the compliance of the agent's actions according to the applied norms. At the moment, the scope of norms can only be restricted by sets of agent's IDs, 3) *Norm enforcement* directs agents to comply with norms.

We illustrate the norm monitoring on a small example: Whenever a CS is busy charging an ATV, it broadcasts its busy state. The obligation is *stop FOLLOW receiveBusy*. It states that within the neighborhood of a CS, an ATV that wants to recharge, receiving the busy message has to perform *stop* next. Thus, *receiveBusy* is added to its *AgentRepresentation*. The *NormMonitor* verifies that all but the *stop* action are forbidden and directs the agent to comply (using FIPA-Request protocol). We extended the formal norm definition from [21] by a new operator called $FOLLOW$. We assume that a mapping of the concrete actions present in our locality to the abstract ones described by the norms is provided (see [21]). For now, the set of concrete actions Act an agent can perform is assigned at design time. Thus, an abstract norm like $\beta\ FOLLOW\ \alpha$, is understood as $b\ FOLLOW\ a$ which means that b is performed

immediately after a, where a and b are concrete instances of the abstract actions β and α. This is translated in the form of forbidden actions, i.e. the agent must not do any action but b.

Applying this to an ATV approaching a busy charging station, the institution eliminates all but the stop action from its set of enabled actions, thereby enforcing the norm.

The current approach provides a good basis which can be easily extended to define other types of contexts or norms.

VI. Conclusion and Outlook

We proposed AAOL, a new metaphor for constructing systems of systems. We used an agent-based approach for modeling structure and behavior of complex systems that consist of (semi-)autonomous systems, where goals, resources, capabilities are described locally while a need for superordinated "global" regulation exists. The notion of organized localities was introduced to describe spatially or logically defined and restricted spheres of influence of regulation bodies. Agents inhabit – and can move across – localities; regulation rules are modeled via computational norms and enforced by electronic institutions. A major objective is to explore and advance the applicability of AAOL to constructing technical systems with (at least soft) real-time constraints. This focus sets our work apart from existing work on normative MAS.

The main contributions of this paper are threefold. First, we described a conceptual architecture for AAOL; second, a metamodel was defined that provides designers with constructs to describe AAOL-type systems, and that, together with the architecture, lays the foundation for an AAOL runtime infrastructure. Third, we illustrated the use of the AAOL metaphor and concepts to model a decentrally organized airport transportation infrastructure.

A main benefit of our approach is that its concepts (localities, institutions, and norms) provide designers with instruments for flexible modeling of different control topologies of systems of systems, ranging from centralized and homogeneous to decentralized and heterogeneous settings. Further, the multi-agent based approach in conjunction with the localities concept supports well decentralized systems design scenarios, where the different parts evolve independently while having to obey certain invariants or rules constraining the overall structural or behavioral development of the system of systems.

The AAOL approach aims at mechatronic agents acting in physical environments, but our current prototype has been implemented on a simulation platform only. Thus, the challenges of a mechatronic layer controlling sensors and actuators, its interplay with the execution layer, and whether the layered architecture is feasible to meet the real-time constraints need further investigation in future. Within the airport scenario implementation, the AAOL concepts are still not fully explored: In particular, major structuring constructs like locality, institution, and organizations are implemented statically.

Based on the proof-of-concept described in this paper, future work will provide more elaborate design-time and runtime models of the core concepts, as well as a more powerful and flexible execution/simulation environment; we will also develop AAOL engineering methodologies and corresponding tools, providing semi-automatic mapping of higher-level models to technical platforms. Finally, models and algorithms for analyzing formal properties of AAOLs, including functional properties but also non-functional aspects, such as safety and performance, are of high interest.

References

[1] P. Dini, N. Rathbone, M. Vidal, P. Hernandez, P. Ferronato, G. Briscoe, and S. Hendryx, "The Digital Ecosystems research vision: 2010 and beyond," 2005.

[2] T. Heistracher, T. Kurz, C. Masuch, P. Ferronata, M. Vidal, A. Corallo, G. Briscoe, and P. Dini, "Pervasive service architecture for a Digital Ecosystem," in *1st WS on Coordination and Adaptation Techniques for Software Entities*, 2004.

[3] H. L. Cardoso and E. Oliveira, "Electronic Institutions for B2B: Dynamic normative environments," *Artificial Intelligence and Law*, vol. 16, no. 1, pp. 107–128, 2008.

[4] J. Vázquez-Salceda, V. Dignum, and F. Dignum, "Organizing Multiagent Systems," *Autonomous Agents and Multi-Agent Systems*, vol. 11, pp. 307–360, 2005.

[5] S. Thrun, "Robotic Mapping: A Survey," in *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002.

[6] O. Wulf, M. Khalaf-Allah, and B. Wagner, "Using 3d data for Monte Carlo Localization in Complex Indoor Environments," in *European Conf. on Mobile Robots*, 2005.

[7] G. Weiss, Ed., *Multiagent Systems - A modern approach to distributed Artificial Intelligence*. MIT Press, 1999.

[8] A. S. Rao and M. P. Georgeff, "Modeling rational agents within a BDI-architecture," in *2nd Intern. Conf. on Principles of Knowledge Representation and Reasoning*, J. e. a. Allen, Ed., 1991, pp. 473–484.

[9] J. P. Müller, *The design of intelligent agents - A layered approach*, ser. LNAI. Springer, 1996, vol. 1177.

[10] J. Ferber, O. Gutknecht, and F. Michel, "From agents to organizations: An organizational view of Multi-Agent Systems," in *4th Intern. WS on Agent-Oriented Software Engineering*, ser. LNCS, vol. 2935, 2003, pp. 214–230.

[11] N. R. Jennings, "On agent-based Software Engineering," *Artificial Intelligence*, vol. 177, no. 2, pp. 277–296, 2000.

[12] H. Schmeck, C. Mller-Schloer, E. Cakar, M. Moez, and U. Richter, "Adaptivity and self-organisation in organic computing systems," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 5, no. 3, pp. 10:1–10:32, 2010.

[13] T. Hestermeyer, O. Oberschelp, and H. Giese, "Structured information processing for self-optimizing mechatronic systems," in *1st Intern. Conf. on Informatics in Control, Automation and Robotics*, H. A. et al., Ed., 2004, pp. 230–237.

[14] M. Yokoo and K. Hirayama, "Algorithms for distributed Constraint Satisfaction - a review," *Autonomous Agents and Multi-Agent Systems*, vol. 3, pp. 185–207, 2000.

[15] S. A. DeLoach, "OMACS: A framework for adaptive, complex systems," in *Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models*, V. Dignum, Ed. IGI Global, 2009, pp. 76–104.

[16] C. Sierra, J. Thangarajah, L. Padgham, and M. Winikoff, "Designing institutional Multi-Agent Systems," in *7th Intern. WS on Agent-Oriented Software Engineering*, ser. LNCS, vol. 4405, 2007, pp. 84–103.

[17] J. O. Kephart and D. M. Chess, "The vision of Autonomic Computing," *IEEE Computer*, vol. 36, pp. 41–50, 2003.

[18] C. Hahn, C. M. Mora, and K. Fischer, "A platform-independent metamodel for Multiagent Systems," *Autonomous Agents and Multi-Agent Systems*, vol. 18, no. 2, pp. 239–266, 2009.

[19] M. Esteva, B. Rosell, J. A. Rodriguez-Aguilar, and J. L. Arcos, "AMELI: An agent-based middleware for Electronic Institutions," in *3rd Intern. Joint Conf. on Autonomous Agents and Multiagent Systems*, 2004, pp. 236–243.

[20] C. Knieke and U. Goltz, "An executable semantics for UML 2 Activity Diagrams," in *ECOOP 2010 Intern. WS on Formalization of Modeling Languages*, 2010.

[21] H. Aldewereld, J. Vázquez-Salceda, F. Dignum, and J. C. Meyer, "Norm compliance of protocols in electronic institutions," in *AAMAS*, 2005, pp. 1291–1292.