MAS2DES-Onto: Ontology for MAS-based Digital Ecosystems

Solomon Asres Kidanu* **Richard Chbeir***
**LIUPPA, University of Pau and Adour Countries*
Anglet, France
Email: {solomon.asres,richard.chbeir}@univ-pau.fr

Yudith Cardinale
Dpto. de Comput. y Tecnología de la Información
Universidad Simón Bolívar, Venezuela
Email: ycardinale@usb.ve

Abstract—**Multi-Agent Systems (MASs) have received much attention in recent years because of their advantages on modeling complex distributed systems, such Digital Ecosystems (DESs). Many existing modeling languages that support the design of such systems are based on ontologies to assist the representation of agents knowledge. However, in the context of DESs, there is still a need for more general conceptual models to represent the specific characteristics of DESs in terms of win-win interaction, engagement, equilibrium, and self-organization. Then, concepts such behavior, roles, rules, and environment are needed. This paper describes an ontology-based approach by proposing MAS2DES-Onto, as the conceptual model, which considers the essential static and dynamic aspects of MASs by a clear representation of their concepts and relationships to support the design and development of DESs. To validate and conduct experimental tests, we integrate MAS2DES-Onto into a framework to automatically generate MAS-based DESs. Results show the efficiency and effectiveness of our approach.**

1. Introduction

Multi-Agent Systems (MASs) have become one of the most promising technologies used in complex applications, especially in collaborative systems to increase the efficiency and effectiveness of working groups in distributed environments [1]. MASs are complex systems that integrate a collection of autonomous agents that have their own goals/actions and are able to interact, collaborate, and exchange knowledge [2]. Digital Ecosystem (DES) is one of the areas in which MASs are appropriated as a means for modeling such complex systems. DESs provide the basis for an open environment where agents interact with each other to reach their individual or shared goals in an evolving environment. Agents in DESs are being capable of acting autonomously, making decisions, and fulfilling responsibilities.

In order to help knowledge representation in MASs, a battery of modeling languages and methodologies proposed in the literature are based on ontologies [3], [4]. Ontologies enables the agents to fully understand the knowledge domain and to identify possible links between different ontology concepts [5]. However, in the context of DESs, there is still a need for more general conceptual models to represent the specific characteristics of DESs in terms of win-win interaction, engagement, equilibrium, and self-organization. Thus, a common and shared vocabulary is also required to define the environment and organizations of the ecosystem, as well as the communication ways, structure, behaviors, roles, and rules of agents in the DES.

Therefore, the objective of this work is to describe an ontology-based approach for MAS-based DESs modeling and development, considering the simplicity, efficiency, and speed in code generation, that an ontology offers to develop MASs. Our approach mainly relies on a layered framework, presented in a previous work [6] and on a core ontology, called MAS2DES-Onto, which is the focus of this work. MAS2DES-Onto provides the essential static and dynamic aspects of MASs by a clear representation of their concepts and relationships to support the design and development of DESs. With MAS2DES-Onto as the core ontology, the framework enables an easy and fast means to instantiate and automatically generate MAS-based DESs from the core ontology. To validate and evaluate MAS2DES-Onto, we conducted experimental tests with the framework. Results show the efficiency and effectiveness of our ontology-based approach to automatically generate DESs.

This work is organized as follows. Related work is presented in Section 2. Section 3 describes MAS2DES-Onto. Experimental tests are presented in Section 4. Conclusions are in Section 5.

2. Related Work

Agents are the natural way to design DESs, because their ability of autonomous problem-solving, making decisions, and fulfilling responsibilities [7]. We identify the following key issues that agents require to operate in MAS-based DES environments: (i) *structure*: agents are coming to DES environments with their own interests; thus, they should provide information about their properties, characteristics, resources; (ii) *behavior*: agents in MASs as well in DESs are supposed to be proactive and reactive; the representation of their internal behavior and reasoning mechanism is vital; accordingly, agents update their beliefs/knowledge based on their precepts, information received, knowledge about the environment; (iii) *roles*: agents come to MAS and

DES environments with various motives being provider, consumer, or both; the agents roles must be clearly defined to facilitate their sharing in the system; (iv) *reasoning and rules*: agents react based on specific set of rules (that dictates their behaviors) and engage in a deliberation process, which allows them to adjust their goals, plans, actions; (v) *communication*: interaction is a mandatory task for agents in MASs and DESs, the communication aspect is a means to express the interaction with the environment and deal with messages exchange; (vi) *environment*: the system should be represented with its constituent elements, allowing to perceive the environment of the MAS-based DES as a whole. Hence, in order to create an agent model, these aspects should be taken into consideration so that an agent could meet the requirements to exist, survive, and benefit in MAS-based DESs. In the following we present some works on agent concept modeling from MAS and DES point of views.

Agent Concept Representations in MASs. The Conceptual Agent Model framework (CAM) [8], comprises the entities of an agent categorized into three models: (i) static, which describes the structural components of a conceptual agent and their relationships; (ii) dynamic, which provides concepts to represent agent behaviors; and (iii) interaction model to describe how agents interact with each other in a domain. In CAM, dynamic entities have the ability to change the world through actions, while static entities do not have action capabilities. Dynamic entities have rules which govern actions. An agent is a dynamic entity that has beliefs, perceptions, and actions (perceiving, learning, reasoning). This work proposes the agent concept from its structure, behavior, and its interaction to the external entities. However, there is nothing about the agent roles in terms of resource and service provision and consumption.

Based on the Belief-Desire-Intention (BDI)[1] model [9], authors in [10] introduce Architecture Description Language (ADL) concepts for specifying MAS architectures. They categorize the main concepts in two models to capture a set of structural and behavioral concepts: (i) the internal model captures the agent mental states and its behavior; the agent knowledge comprises a set of beliefs the agent has about the environment and a set of goals it follows; the intentional behavior of an agent is represented by its ability to react to events; an event is generated by an action of an agent or by services provided by another agent; a plan defines the sequence of actions chosen by the agent to achieve a goal; and (ii) the global model presents the interaction among agents that conform the MAS and is composed of: agent, configuration, architecture, interface, effector, sensor, and service. This work displays concepts at individual and global agent levels, while interacting each other to bear MAS. However, it misses role and rule definitions.

1. The BDI model uses the Belief, Desire, and Intention concepts that are used correspondingly to symbolize and model an agent information state (what agent knows about itself and its environment), motivational state (what agent tries to achieve), and deliberative state (a plan to achieve agent desired state of affairs).

Authors in [11] propose an Ontology Driven-Procedural Reasoning System (O-PRS) agent model. PRS substitutes the abstract notions of desires and intentions in BDI to concrete concepts of goals and plans. The proposed O-PRS agent model represents the basic knowledge that PRS-like agents need to act, which are Believes, Plans, and Events. Each agent may have one or more believes which may correspond to certain events and plans. When an event occurs for the agent, the corresponding plan for the same believe will be executed. This work does not clearly provide all elements to define agent structure, role, and rule definitions; and it provides nothing about how agents interact and communicate each other.

In [12], authors describe the Emotional Belief-Desire-Intention Model by combining concepts of BDI with emotional aspect of agents. Then, agents would have both cognitive and reactive behavior. To do so, the following concepts are combined: Percepts, Beliefs, Desires, Options, Intentions, Emotions, Personality, and Resources. Percept represents anything that comes from the environment and Belief is acquired from it. Desires is a goal achieved by an agent and Option is an alternative to accomplish the desires. Intention is an option that an agent has committed. Emotion represents an agent instinct behavior. Personality states the set of emotional qualities that make the agent different form others. Resource concept has a central role in agent decision making. According to this work, emotions influence the available resources the agent can use. However, role and interaction aspects of agents have been left.

ANote is a modeling language to offer a standard way to describe concepts related to the agent-oriented modeling process at high level and level of individual agents [13]. Concepts such as goal, interaction protocol, environment, resource, and organization are defined at high level. Action, communication, and plan are concepts that characterize individual agents. With these concepts, ANote defines seven views based on its conceptual meta-model: goal, agent, scenario, planning, interaction, environmental, and organizational views. A goal provides an initial identification of a tree of goals that outlines the system functions. An agent specifies the roles that will perform the goals elicited in the goal view. A scenario captures agent behavior that shows how goals are achieved by agents. A planning shows the agent internal actions and their sequences. An interaction is used to represent the set of messages that agents exchange while executing an action plan. This work gives more emphasis to interaction and external environment. Less attention pays to representation of agent structure and reasoning concepts.

In [5], authors provide a conceptual view for MAS modeling that takes into account the three main dimensions in MAS development: agents, environment, and organizations. Authors specify agents and their characteristics and roles. The agent concept is described further in terms of: Percept, Action, Belief, Message, and Plan. The organizational view indicates the role an agent could play in a group to accomplish its mission and achieve a goal. Besides graphical display of the three views, further details are not provided.

Agent Concept Representations in DESs. Authors in [14] propose the concepts of DESs and its subclass concepts: species and environment. Species are individual or organizations that participate in the ecosystem which come from certain domain, play roles, and follow rules. They are driven by own profit and carried out tasks that relate to the profit. Hence, species ontology is presented with the combination of elements that describe them – i.e., Domain, Task, Profit, Rule, and Role (Supplier (Available Service) and Requestor (Requested Service)). Environment is the second constituent of a DES which supports services to species. Thus, the DES ontology has species and environment as sub components with specifications of those components. The internal structure and behavioral states of an agent are disregarded in this model.

Discussion. Table 1 compares the models described with respect to our identified aspects of agent concept. Both the structural and behavioral aspects of agent enable interoperability among agents. However, most of the existing works focused on representing the internal state (behavior) in a form of belief, plan, and action. The reasoning/strategy aspect of an agent makes it intelligent in its decision making by properly analyzing the existing knowledge; whereas very few works give attention to this. Agents might be required to define and follow some constraints/rules while they are interacting with others. However, existing works forgot this aspect except one work. Role definition is very important as agents exist as a community and all agents could not play the same role; only half of the reviewed works address it. Since, communication is vital for efficient agents interaction, majority of the works address it though in different way. The way the environment is represented has an impact on agent modeling in its communication. This environment could be a MAS or a DES. Thus, considering this fact, most of the works address this. To sum up, little attention has been given to the structure, reasoning, role, and rules. Moreover, the representation of each model of the agent varies from work to work, which shows that there is no comprehensive agent concept model. Therefore, it recalls a need to come up with a comprehensive and generic agent concept model that will support interoperability, facilitate communication, express agent behaviors for performing their actions in a strategic way, and allow defining constraints/rules.

3. MAS2DES-Onto

Our ontology-based approach aims to support the modeling and automatic generation of MAS-based DESs, relying mainly on a layered framework, presented in a previous work [6], and on a meta-model, called MAS2DES-Onto. MAS2DES-Onto integrates concepts to determine the elements to specify a MAS and what should be present in DESs. We integrate MAS2DES-Onto into the framework to provide an Ontology-based Integrated Development Environment (O-IDE) for automatic generation of MAS without requesting advanced programming skills. As shown in Figure 1, the framework contains three main components [6]: (i) *Designer*, in which MAS2DES-Onto is integrated, it allows developers to specify/tune end-user/domain DES requirements (agent concepts and their relationships: roles, rules, behaviors) by using OJ language (for details see [6]); thus, a specific ontology model of the requested DES is derived from the MAS2DES-Onto core ontology; (ii) *Generator*, which automatically generates all software components of the DES from the derived ontology and according the user desired platform and language; and (iii) *Deployer*, to instantiate the generated DES while considering configuration aspects such as release, logo, application (web, mobile applications, desktop applications). In this section, we detail MAS2DES-Onto.

The core concept of MAS2DES-Onto is *agent* (see Figure 2), all other concepts are structured into five modules: structural, species, reasoning, interaction, and system modules. We highlight the intention of each module as follows.

Structural Module. This module represents concepts describing basic components and properties of the agent structure (see Figure 3). It is composed of: *agent*, *property*, *profile*, *resource*, and *rule* concepts. An *agent* is an atomic autonomous entity that is capable of performing functions and represents a participant in the system. A *property* describes attributes that characterizes an agent which can be either *common* to all agents or dedicated (*personal*) to a specific agent. A *profile* is a set of information describing an agent with its *preferences*. A *resource* is a concept which represents the set of assets that an agent possesses and manages. A *rule* defines conditions or constraints set by a specific agent that others should follow while interacting. This module mainly provides the structure to fulfill the requirements of MASs. In fact, to meet also the requirements of an agent in DES environments, *profiles* and *rules* are incorporated. These concepts are needed to provide agent interests and *preferences* for better collaboration and *rules* are essential to keep benefits of an agent in the DES, while interacting with others.

Species Module. As Figure 4 shows, this module consists of *agent* types with their *roles*. In DESs, we can have both *moral* and *digital agents* unlike MAS, which only has digital agents [15]. The *role* concept defines the part played by an *agent*. Depending on the context, an *agent* is capable of playing several *roles*, such as *consumer* (who sends *requests* to access services/resources), *provider* (who sends *responses* to *requests*), and *orchestrator* (who coordinates activities in the system). This is decisive for agents to determine to whom to communicate and set their expectations. Also multiple *agents* may be able to play the same *role*. This module focuses on addressing requirements of DESs.

Reasoning Module. Reasoning is vital for agents to act intelligently in the environment where they exist. Indeed, agents which are either in MASs or DESs are assumed to crown characteristics of being proactive and

TABLE 1: Summary of Agent Concept Models

Work	Structure	Behavior	Role	Reasoning	Rule	Comm.	Env.
CAM [8]	Yes	Yes	No	Partial	No	Yes	No
ADL [10]	No	Yes	No	Partial	No	Yes	Partial
O-PRS [11]	No	Yes	No	No	No	No	Partial
EAM [12]	No	Yes	No	Partial	No	No	Partial
ANote [13]	No	Yes	Yes	No	No	Yes	Yes
Onto for MAS [5]	No	Yes	Yes	No	No	No	Yes
DES [14]	No	No	Yes	No	Yes	Partial	Partial

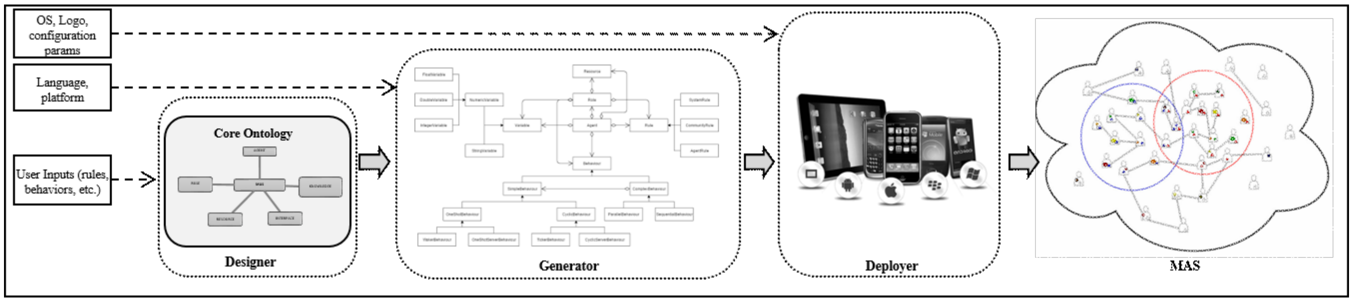
OS, Logo, configuration params
Language, platform
User Inputs (rules, behaviors, etc.)
Core Ontology
Designer
Generator
Deployer
MAS

Figure 1: MAS2DES-Onto: Conceptual Model for MAS-based Digital Ecosystem

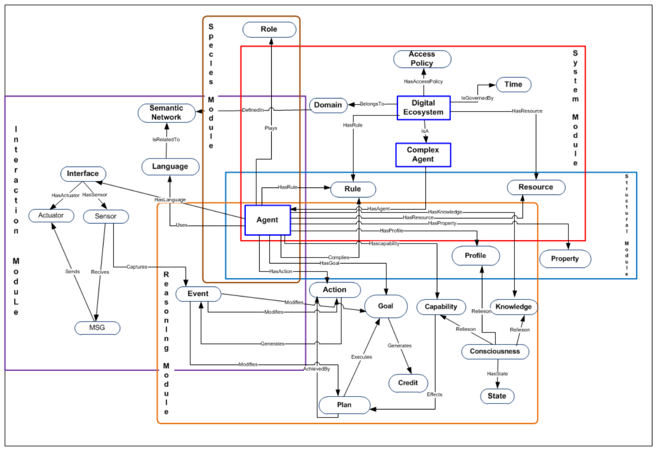

Figure 2: MAS2DES-Onto: Conceptual Model for MAS-based Digital Ecosystem

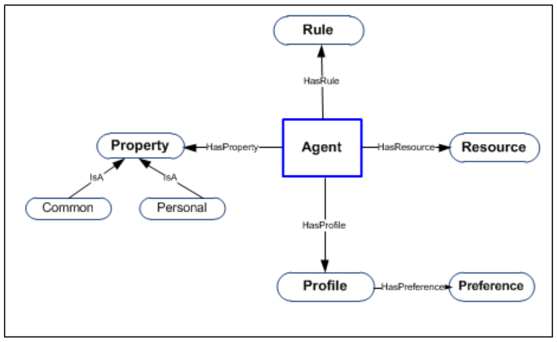

Figure 3: MAS2DES-Onto Structural Module

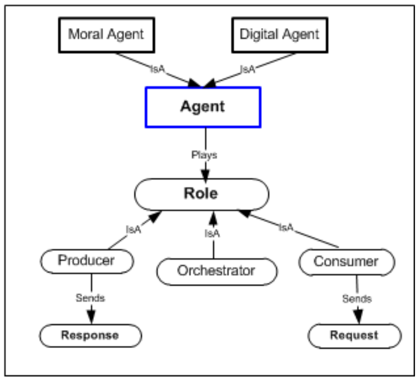

Figure 4: MAS2DES-Onto Species Module

reactive. Obviously, the level of expected reasoning skill in DES agents must be better than agents in MAS, since agents in DES should react in responsible manner for the safety of their environment. The reasoning module, shown in Figure 5, consists of mental states of an agent and dependencies among them. An agent makes decisions based on its beliefs, desires, intentions (specified in its *profile* and *preferences*), and *capabilities*. An agent needs *knowledge* about its environment in order to make good decisions. However, *knowledge* about the current *state* of the environment is not always enough to decide what to do. In addition to a current *state* description, the agent needs *goal* information, which represents aims and desires that an agent wants to achieve. *Goal* achievement requires commitment. Thus, to maximize the *credit* of success, an agent need to acquire a *capability*, which represents the intention and ability of an agent to react to its *plans*. A *plan* defines the sequence of *actions* or services to be chosen by the agent to fulfill a *goal*. An *event* is generated by an *action* or by services provided by another agent; which may modify *plan*, *action*, and *goal*. Agents must comply *rules* which govern *actions* while communicating to others. *Consciousness* is cumulative effect that arises from agent *knowledge*, *capability*, and *profile*. These concepts are important to have proactive and reactive agents in the system so that they can quickly deal with unexpected events [12].

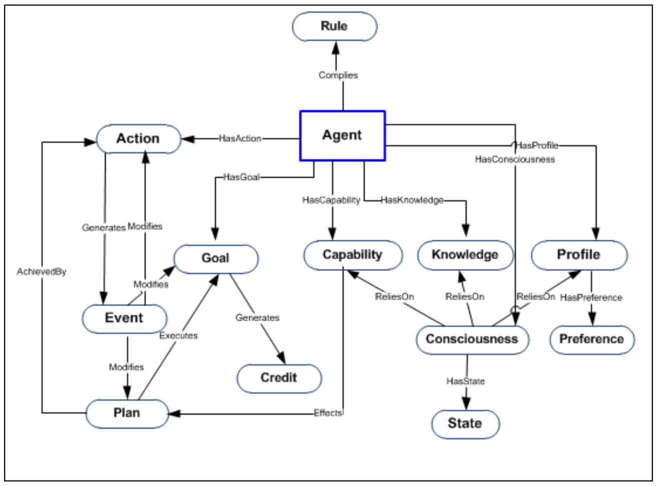

Figure 5: Reasoning Module of MAS2DES-Onto

Interaction Module. One of the basic essences of agents in MASs and DESs is interaction, which serves as basic element to support communication and the construction of the system [16]. Figure 6 displays the concepts of this module. An agent interacts with its environment through an *interface*, that specifies how an agent appears to the rest of the system. It is composed of a set of *actuators* and *sensors*. An *actuator* provides a service that is available to other agents. Each *sensor* requires a service from another agent and captures *events*. To effect this, a *message* (*MSG*) is the means; which is an object communicated between agents. The correspondence between a *request* and a *response* defines an interaction through a *message* (*MSG*). Agents can interact by asking for or sharing *goals*, *plans*, *resources*, and semantic meanings (*terms*). Agents in DESs are heterogeneous and domain-clustered. In order to support and bring semantic interoperability among agents, common communication vocabulary (*language*) must be defined with their semantic relationships (*semantic network*).

System Module. This module essentially specifies concepts that compose a DES environment. Only agents are found in MAS environments, whereas a DES environment has agents and other important elements that facilitate collaboration among participants. Figure 7 shows that a *digital ecosystem* consists of *agents*, *complex agents*, *resource*, *rule*, *time*, *access policy*, and *domain*. An *agent* represents a participant in the *digital ecosystem*. *Complex agents* represents an agent which consists of other agents. In DES, all agents come to the environment with some *resources* and should obey the system level *rules*. These agents are also free to define their resource sharing and usage policy (*access policy*). Every *digital ecosystem* is assumed to have one or more application *domains* in a given period of *time*.

Discussion. MAS2DES-Onto models agent as an entity with purposes (represented as agent roles, goals, and actions) and an entity with internal control (represented as agent

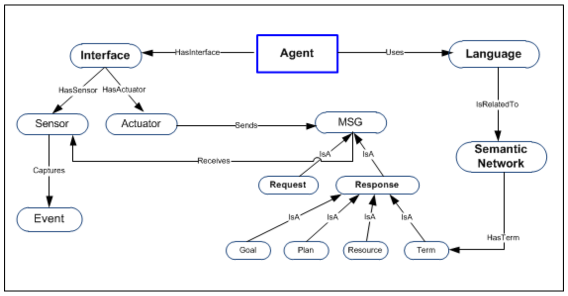

Figure 6: MAS2DES-Onto Interaction Module

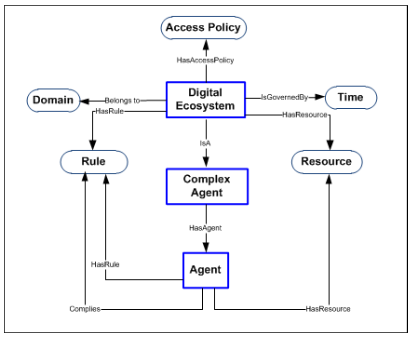

Figure 7: MAS2DES-Onto System Module

knowledge, plans, capability, and rules) to assures its autonomy. The model considers consciousness (among other required capabilities of an agent) to deal with events to own its reactivity character. MAS2DES-Onto also supports the modeling of agent acquaintances and interaction protocols which is significant to implement cooperative behavior of agents. Moreover, our MAS2DES-Onto represents the DES itself as one of the main concepts. Apart its advantages in support for interoperability, knowledge representation and reasoning, semantically consistent communication between agents, MAS2DES-Onto model is a key step in fully defining the design of agents to the point where a higher degree of code generation is possible.

4. Experimental Testing and Results

To validate and evaluate the capacity of MAS2DES-Onto as the core ontology of a framework for modeling and automatic development of MAS-based DESs, we conducted performance evaluation and agent interaction modeling tests. These tests were executed with OnToJade (ONtology TO JADE) [6], a prototype of the framework which uses JADE[2] as the implementation platform and in which we integrate MAS2DES-Onto.

Three different kinds of tests are conducted. The first test is to show the performance of the proposed framework. The second test show the suitability of our approach for modeling agent interaction in the system. Finally, the third test is to demonstrate how to model a specific behavior in the system: enforcing to follow system level rules.

2. http://jade.tilab.com

4.1. Performance Tests

The goal of this test is to prove that the MAS2DES-Onto based framework supports developers better than manual approach in generating MAS-based DESs. To show this, we compared the process of automatically generating MAS-based DESs using OnToJade against the usual manual way by using platforms such as JADE and Java. We create three DESs: (i) **DES1**, which contains the minimum possible number of agents (two agents); (ii) **DES2**, with ten agents; and (iii) **DES3**, which has fifteen agents. We measure the number of code lines and the time required to create the Java files and to instantiate the system. Each DES in OnToJade was generated 100 times and the average measure is taken as the final result.

Table 2 presents the results, in which Total Time considers both the DESs creation and instantiation times. Regarding the number of code lines, there is an incremental growth for both approaches. However, in all scenarios, for OnToJade the number of code lines generated is only nearly to 20% of the code lines generated with the manual approach. In terms of time, OnToJade takes almost 50% more time than the manual approach to instantiate DESs. This is because, besides pure codes, agent behaviors and relations are expressed using the OJ Language and cause more time to parse. However, the total time for OnToJade is less than the manual approach in all scenarios. This is due to the time to create files in OnToJade is less than 20% than the manual one.

We also conducted a test to automatic generate a larger DES, with 1000 agents (**DES4**). Due to the size of the system, we omit the manual test in this case. As shown in Table 3, only few code lines are added and the time remains less than one minute even for such huge DES. To conclude, the complexity of the system has insignificant impact on the time to generate the DES using our MAS2DES-Onto based approach.

4.2. Agent Interaction Modeling

This test focuses on showing the modeling of *Agents* interaction with MAS2DES-Onto. For this purpose, we define:

- three categories of agents: *Requester, Provider, Moderator*; they represent different *roles* from the Species Module;
- three types of requests: *content, processing, support*; they represent different *resources* from Structural and System Modules and the respective *Interfaces* and *MSGs* from Interaction Module to offer/access resources);
- two DESs, from the System Module, with the following specifications: (i) **DES1** consists of three agents (*Agent A, Agent B, Moderator Agent*); *Moderator Agent* is a special agent with coordination *actions* and *capabilities* (from Reasoning Module), *Agent A* represents a *Requester*, and *Agent B* is a *Content Provider*; *Agent A* asks *Agent B* for contents; this DES1 is for testing the One-to-One agents interaction; and (ii) **DES2** consists of five agents (*Agents A, B, C, D, Moderator*); *Agent A* is a *Requester*; *Agent B* and *Agent C* represent *Content Providers* (with different type of resources) and *Agent D* represents a *Processing Provider*; this DES2 is to test the One-to-Many interaction among one and $N > 1$ agents.

Figure 8 shows the derived ontology, after the needed concepts for the tests were taken from the core MAS2DES-Onto ontology. All *actions* are modeled from the Reasoning Module by using our OJ Language [6]. Our OnToJade prototype offers a simple graphic visualization of actions happened during the life of the system. Figure 9 displays the view of the graphic interface of the generated DES1 and shows that a unicast communication between *Agent A and B* has made successfully. Edges between the agents and the *Moderator Agent* shows the subscription of the agents to the *Moderator* to join to the system[3].

The graphical view of the second experiment is shown in Figure 10. We also show below an extract of OJ Language statements for illustrative ends. In order to effect a multicast communication, it is a must for agents to get registered by the *Moderator Agent* (below we show the OJ instructions in (1)), who takes care of the fulfillment of *rules* (modeled from the Structural and System Modules, see Section 4.3). Thus, *Agent A* asks first the *Moderator Agent* for agents that match the service type it is interested on (corresponding OJ instructions are marked as (2) below), *Moderator Agent* responds that *Agents B and C* offer that service, then *Agent A* contacts them (OJ instructions in (3)).

OneShotbehavior (1)

```
RegisterToModertor action
 "REGISTER AS "content-provider"" ProcessingRegistration
 "REGISTER AS "processing-provider""}
```

Wakerbehavior (2)

```
Agent A action "SEND REQUEST @TYPE:content-provider"
RequestWaker action "SEND REQUEST
    @TYPE:content-provider":string
ProcessingRequest action "SEND REQUEST
    @AgentD RESOURCE WHERE NAME EQUALS "hello.jar""
```

Cyclicbehavior (3)

```
ProcessingRequestServer action "SEND INFORM
    @sender RESULT OF EXEC MSG.CONTENT"
ProcessingRequestServer reacts to "REQUEST":string
RequestServer action "SEND INFORM
    @sender RESOURCE":string
RequestServer reactsTo "REQUEST":string
```

To conclude, the test results and the displayed graphs show that with MAS2DES-Onto it is possible to define different type of agents with different resources, behaviors,

3. AMS (Agent Management System) and DF (Directory Facilitator) shown in Figures are agents generated by JADE for platform control purposes.

TABLE 2: Measures to generate DESs

Measures	DES1		DES2		DES3	
	OnToJade	*JADE/Java*	*OnToJade*	*JADE/Java*	*OnToJade*	*JADE/Java*
N. of code lines	∼10	∼50	∼15	∼70	∼20	∼90
Time to create files	20 sec.	2 min.	30 sec.	3 min.	45 sec.	4 min.
Time to instantiate	1.1184 sec.	0.621 sec.	1.197 sec.	0.629 sec.	1.214 sec.	0.648 sec.
Total Time	**<1 min.**	**<3 min.**	**<1 min.**	**<4 min.**	**<1 min.**	**<5 min.**

TABLE 3: Measures to generate DESs

Measures	DES4
	OnToJade
N. of code lines	∼25
Time to create files	45 sec.
Time to instantiate	3.766 sec.
Total Time	**<1 min.**

and roles. It is also noted that the involvement of special agents is important to facilitate the smooth communication among agents and for proper coordination and management of the system.

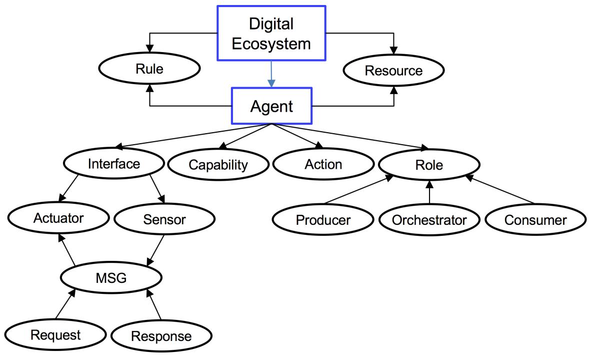

Figure 8: Ontology for the experimental DESs derived from MAS2DES-Onto

4.3. Rules Modeling

The objective of this test is to illustrate how we can model specific behaviors in the system. We do so by showing how it is possible to model the validation of system level rules, such as: Rule for minimal number of agents, Rule for contribution, Rule for collaboration. These rules are integrated through the *Moderator Agent*. We created different DES at each level of the test.

Rule 1: minimum number of agents. In a Digital Ecosystem, there must be at least two agents. For this test, we created a system with only one *Agent*. When the *Moderator Agent* observes that the amount of agents in the system is not enough to function as a DES, it automatically ceases the DES after a short delay. Here is below the trace of the *Moderator Agent* in this situation:

```
Moderator trace: System currently running with a
number of agents running on the platform (1)
inferior to the constant "noOfAgents", system
will shutdown automatically if no agents comes
to the platform!
```

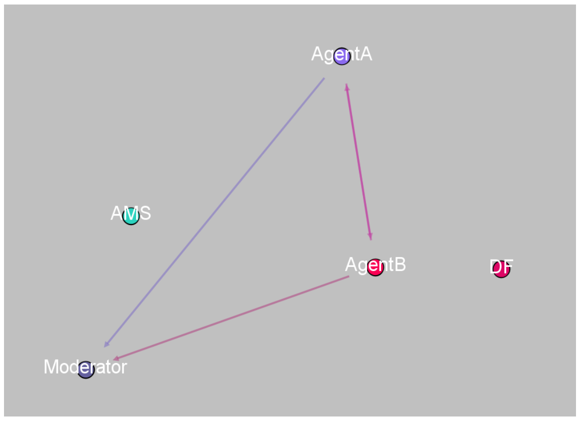

Figure 9: Unicast communication in DES1

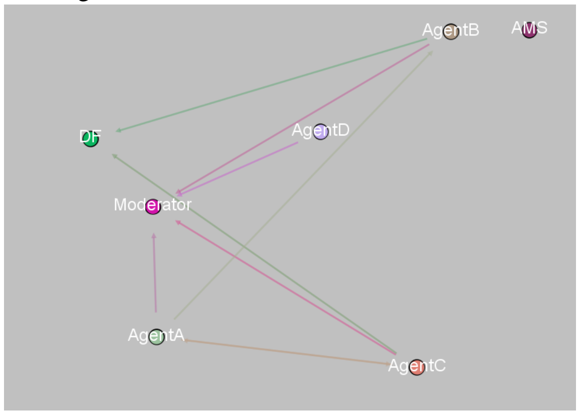

Figure 10: Multicast communication in DES2

This result indicates that there must be at least two agents so that the DES exists and the platform supports it.

Rule 2: minimum number of contents. For this test, we create a DES with two different agents, *AgentA* and *AgentB*. *Agent A* is linked to a resource folder; whereas *AgentB* have no any resource. The *Moderator* realizes that *AgentB* does not fulfill the minimum threshold for resources to stay in the system and it automatically add it to the list of agents to kill. The console trace below for the Rule 2 test gives more details:

```
Moderator trace: new agent asks to join the
                 platform, name is: AgentA
Moderator trace: new agent asks to join the
                 platform, name is: AgentB

AgentA Trace:Sending message(s) to: Moderator,
             performative is SUBSCRIBE, content is
             C:/Users/Pictures/Images/screenshot0.png
```

```
Moderator trace: AgentB have a number of resources
                 inferior to the constant
                 "contentMinimumAmount" name
                 added tokickList!
```

The result tells that each agent in the DES should contribute so that mutual benefit could ensure. otherwise the *Moderator Agent* will be forced to take actions. This is important to keep equilibrium of the systems and punish free-rider agents.

Rule 3: TimeToQuit. This test shows that the *Moderator Agent* will automatically kill an agent that does not interact with its environment, contribute, and collaborate in a given time frame. For the purpose of this test, we created a DES containing four different agents. *AgentA* sends a message every three seconds to every agent on the platform; *AgentB* has resources but does not share them; *AgentC* is an idle agent (no resources and behaviors); and *AgentD* sends a message every three seconds to every agent but it has no resources.

According to the rules for interaction, contribution, and collaboration, the *Moderator* did not kick *AgentA* and *AgentC* based on their resources amount. However, it took actions on the rest of the agents. The console trace below provides the details about this test:

```
Moderator trace: new agents ask to join the
                 platform, name are: AgentA,
                 AgentB, AgentC, AgentD
Moderator Trace: AgentA declaring a new resource!
                 Name is /screenshot0.png
Moderator Trace: AgentB declaring a new resource!
                 Name is /hello.jar
Moderator trace: Agent B TimeToQuit = 0! Name
                 added to kickList!
Moderator trace: Agent AgentC have a number  of
                 resources inferior to the
                 constant "contentMinimumAmount",
                 name added to kickList
Moderator trace: Agent AgentD have a number of
                 resources inferior to the
                 constant "contentMinimumAmount",
                 name added to kickList
```

To summarize, the tests on rules has shown it is possible to define and implement different kinds of rules at system and individual agent levels to ensure the benefits of all participants and for the balance of the system. With this, the results displayed that the Moderator Agent is capable of enforcing system level rules. And at last, any agent which is not ready to comply system level rules cannot survive in the system.

5. Conclusions and Future Work

We present MAS2DES-Onto, an ontology that can be used to model MAS-based Digital Ecosystems. The key concepts in our approach were categorized in five modules that can be developed at various levels of detail, allowing to design and develop MAS in a modularized way. MAS2DES-Onto captures all the requirements for agent and system environment representation. This is done by providing a precise definition of agents in terms of its static structure, dynamic behavior, interactions with the environment, and system structure. We show that integrating our model into existing MAS methodologies is not only possible, but is a positive step towards more complex system models. We plan to validate MAS2DES-Onto in other application DES domains.

References

[1] U. Pakdeetrakulwong, P. Wongthongtham, and et al., "An ontology-based multi-agent system for active software engineering ontology," *Mobile Networks and Applications*, vol. 21, no. 1, pp. 65–88, 2016.

[2] A.-M. Talib, R. Atan, R. Abdullah, and M. A. Murad, "Security ontology driven multi agent system architecture for cloud data storage security: Ontology development," *Journal of Computer Science and Network Security*, vol. 12, no. 5, pp. 63–72, 2012.

[3] D. Choinski and M. Senik, "Ontology based knowledge management and learning in multi-agent system," in *Symp. on Agent and Multi-Agent Systems: Technologies and Apps*, 2012, pp. 65–74.

[4] V. Mascardi, D. Briola, and et al., "A holonic multi-agent system for sketch, image and text interpretation in the rock art domain," in *Agents and Multi-agent Systems*, 2012, pp. 81–100.

[5] A. Freitas, L. Hilgert, S. Marczak, F. Meneguzzi, R. H. Bordini, and R. Vieira, "A multi-agent systems engineering tool based on ontologies," in *Internat. Conf. on Conceptual Modeling*, 2015, pp. 1–4.

[6] C. Donzelli, S. A. Kidanu, R. Chbeir, and Y. Cardinale, "Onto2MAS: An Ontology-Based Framework for Automatic Multi-Agent System Generation," in *Conf. on Signal Image Tech. & Internet based Systems*, 2016, pp. 381–388.

[7] P. Balaji and D. Srinivasan, "An introduction to multi-agent systems," in *Innovations in multi-agent systems and applications-1*, 2010, pp. 1–27.

[8] K. Monu, "A conceptual modeling method to use agents in systems analysis," *Lecture Notes in Computer Science*, vol. 5829, pp. 374–386, 2009.

[9] B. Pell, M. Pollack, M. Tambe, M. Woolridge, and M. Georgeff, "The belief-desire-intention model of agency," in *International Conference on Multi-Agent Systems*, 1999, pp. 1–10.

[10] S. Faulkner, M. Kolp, Y. Wautelet, and Y. Achbany, *A Formal Description Language for Multi-Agent Architectures*, 2008, pp. 143–163.

[11] A. Mousavi, M. Nordin, Z. A. Othman *et al.*, "An ontology driven, procedural reasoning system-like agent model, for multi-agent based mobile workforce brokering systems," *Journal of Computer Science*, vol. 6, no. 5, pp. 557–565, 2010.

[12] M.-A. Puică and A.-M. Florea, "Emotional belief-desire-intention agent model: Previous work and proposed architecture," *Journal of Advanced Research in Artificial Intelligence*, vol. 2, no. 2, pp. 1–8, 2013.

[13] J.-A. Sardinha, R. Choren, V. T. Da Silva, R. Milidiú, and C. de Lucena, "A combined specification language and development framework for agent-based application engineering," *Journal of Systems and Software*, vol. 79, no. 11, pp. 1565–1577, 2006.

[14] H. Dong and F. K. Hussain, "Digital ecosystem ontology," in *Internat. Conf. on Emerging Technologies and Factory Automation*, 2007, pp. 814–817.

[15] J. V. Rauff, "Multi-agent systems: An introduction to distributed artificial intelligence," *Mathematics and Computer Education*, vol. 39, no. 1, p. 81, 2005.

[16] S. A. DeLoach and J. L. Valenzuela, "An agent-environment interaction model," in *International Workshop on Agent-Oriented Software Engineering*, 2006, pp. 1–18.