JOURNAL OF INFORMATION SCIENCE AND ENGINEERING XX, XXXX-XXXX (2016)
DOI: XX.XXX.XXXX.XX

A New Formal Multi-Agent Organization Based on the DD-LOTOS Language

SAMRA SABEG [1,+], TOUFIK MESSAOUD MAAROUK [1] AND MOHAMMED EL HABIB SOUIDI [1]
[1] *Faculty of Science and Technology*
University Abbes Laghrour Khenchela
ICOSI Lab, BP 1252 EL Houria
E-mail: {samra.sabeg; maarouk.toufik; souidi.mohammed}@univ-khenchela.dz

A multi-agent organizational model represents a coordination mechanism that allows tasks to be shared among agents to perform complex tasks. While the Agent-Group-Role organizational model (AGR) provides a concise methodological framework for designing multi-agent systems, it is expressed in informal language and lacks formal semantics. Consequently, designers of multi-agent systems have been unable to exploit this model for analysing and checking the behaviour of their systems. Some works propose investigating the issue of model transformation; unfortunately, no effort has been made to transform AGR models using a formal language defined on the semantics of true concurrency. The DD-LOTOS Language is one of the promising alternatives to this problem, as it is based on true concurrency semantics and supports the distributed aspect of the system. This paper proposes a formal approach that generates DD-LOTOS specifications from AGR models. This formalization permits the analysis, verification, and validation of the important properties of an organization. Model-to-text (M2T) transformation uses the Xpand tools to implement the approach. The e-commerce case study is used to illustrate our approach.

Keywords: Multi-agent systems, Coordination mechanism, Organizational model, Model transformation, Formal semantics, DD-LOTOS

1. INTRODUCTION

In recent years, multi-agent systems (MAS) are considered a promising paradigm for developing complex, distributed, and dynamic applications [1]. Therefore, a particular interest nowadays is given to multi-agent systems because they heavily rely on efficient methodologies, new techniques, and recent engineering methods that designers should exploit when designing open and heterogeneous systems [2].

Multi-agent systems are applied to different domains, such as robotics, distributed systems, e-commerce, communication protocols, networks, etc.[3]. Multi-agent systems engineering may be summarized in two viewpoints [4]:

1) Agent-oriented approaches : in which a special attention is initially paid to agents' behaviour and their architectures [5]. In [6] authors declared that agent-oriented systems need no structure in advance and that their organization is implicitly emergent as an outcome of agents' interactions. The main problems of this type of MAS are uncertainty and

[+]Corresponding author

unpredictability. This approach may lead a system to unsuitable situation. As a result, agent-oriented approaches cannot be used to engineer large-scale systems.

2) Organization-oriented approaches: this approach has been recently adopted to overcome the major weaknesses of the first approach (uncertainty, unpredictability). An important element of that approach is the concept of organization. An organization is a set of agents and can be partitioned into partitions (groups). It minimizes the scope of interactions and reduces unpredictability [7]. The adoption of organizations is a new manner for defining the structure and the interactions between agents in MAS.i.e. specify the structural and dynamical aspects of multi agent system. In this approach, the designer describes the organisation and patterns of agents' activities in advance [8] [9]. In addition, the organization imposes a set of constraints that agents respect to achieve their global objective easily. These constraints describe an organizational model that aims to model coordination in MAS and guarantee a higher level of efficiency [10].

Several organizational models are currently adopted for modelling coordination in organization oriented MAS, such as AGR [11], AGRE [12], MOISE [13], AGRMF [14, 15], and others. An organizational model is a coordination mechanism used in a development methodology to control agents' behaviour and ensure the ability of organization to efficiently carry out its tasks, [16]. It is defined by means of new organizational concepts (role, group, organization, goal, etc.).

Despite the wide use of these organizational models in the design of complex MAS, they are usually expressed in an informal language and suffer from a lack of formal semantics. This renders the essential step of verification and validation organizational model difficult. Therefore, we cannot ensure that the system design contains no errors.

The AGR model is one of the most familiar and widely adopted organizational models for building MAS. However, it is described in intuitive and natural language. This lack of formal semantics can hinder the correct interpretation and verification of the model [17]. Thus, integrating formal languages with the AGR model in the modelling process seems necessary to create verifiable and rigorous specifications.

To remedy this issue, several works have investigated the problem. Typically, the proposed approaches depend on formal methods to design and verify complex systems. The majority of these approaches are based on process algebras [18, 19], rewriting logic and Maude [20], and Category Theory [21, 22] as specification languages.

In the context of this work, we propose a formal approach based on model transformation to transform the AGR model into a DD-LOTOS specification. The choice of AGR model among the existing organizational models is justified by several reasons. Firstly, the AGR model is a simple but very generic and powerful organizational model for the design of complex applications. Secondly, it conforms to the general principles of organization-oriented systems in which we are interested. Thirdly, the AGR model uses several notations (cheeseboard diagram, organizational structure, and organizational sequence diagrams) to represent both static and dynamic aspects of an organization. Lastly, and most importantly, the AGR model is integrated with multi-agent systems development methodology to complete the analysis and design phases of the development process.

The main contribution of our work is to define formal semantics for the AGR model using the formal language DD-LOTOS [23]. The novelty of our approach compared to existing works is that the DD-LOTOS language supports various aspects of complex systems.

The DD-LOTOS language represents a timed extension of the LOTOS description language. However, DD-LOTOS is defined with true parallelism semantics, which avoids assuming structural and temporal atomicity of actions. In other words, in DD-LOTOS, actions are considered non-atomic and have a non-zero duration. In contrast, approaches based on process algebras and timed Petri nets use interleaving semantics, where actions are assumed to be atomic, indivisible, and have zero duration.

Furthermore, DD-LOTOS offers the advantage of allowing the explicit specification of sites or localities. In the context of multi-agent systems, these sites can represent the notion of a group. Another fundamental feature of DD-LOTOS is its capability to verify quantitative properties of actions, owing to the assumption of non-atomicity in their structure and timing.

This transformation gives a rigorous specification for the AGR model. The proposed approach consists of two main steps. Firstly, we define a meta-model of the AGR model using the EMF (Eclipse Modelling Framework) tool. Secondly, we use the Xpand tools to generate the DD-LOTOS specification from the AGR model. The transformation is done by a model-to-text transformation (M2T).

This transformation aims to enable model checking of AGR models using the model checker UPPAAL. The choice of UPPAAL is motivated by two main reasons: firstly, UPPAAL takes the timed automata model as input, and secondly, it supports the verification of timed properties such as bounded liveness.

When the formal DD-LOTOS specification is complete, a semantic model specific to the DD-LOTOS language called C-DATA is generated, which is an extension of the timed automatons; the generation approach is presented in [24]. Finally, we can use the UPPAAL model checker to check the properties of the smooth behaviours; these properties are expressed in a temporal logic such as the TCTL logic. The UPPAAL tool gives us a yes answer for each property if it is satisfied and a no answer with a diagnosis for each property that is not satisfied. Several properties, including the absence of deadlock, the respect of time constraints by agents, the absence of non-violation of confidentiality properties between agents, and compliance with the specification, can be verified in the AGR model.

The rest of this paper is structured as follows: Section 2 reviews the related work. Section 3 describes the main concepts of both the AGR model and the DD-LOTOS language. Section 4 presents the proposed formal approach. Section 5 details the implementation of the suggested approach using the Xpand language. We illustrate our approach with the e-commerce example in Section 6. Section 7 presents the conclusion of the paper and future work.

2. RELATED WORK

Currently, few approaches have been proposed for the formalization of organizational models in MAS. They aim to specify and verify formally multi-agent systems based on organizational notions to deal with the lack of solid semantics of different models. This lack hinders the verification, analysis, and correct interpretation of these models.

Many formal methods have been adopted to deal with this issue, such as the Maud language, LOTOS, Petri net notation, and category theory. DD-LOTOS has not been

largely applied to multi-agent systems, especially not organizational models. Therefore, the proposed approach differs from previous studies in that it addresses the transformation of the AGR organizational model by using the DD-LOTOS language defined on maximality semantics.

In [20], the authors proposed a formal approach based on rewriting logic to prototype the AGR model. The approach is implemented by the Maude language. This formal specification allows the simulation, verification, and validation of the AGR organizational model and is evaluated through a supply chain management case study.

Several new works, such as [21, 22, 25], have used category theory to formalize the systems based on organisational notions.

In [21], the authors proposed an idea for formalizing organization in MAS. The formalization is based upon collective phenomena using category theory. This study aims to have a categorical model that enables studying some organizational properties such as adaptation and stability.

In [22], the authors have exploited category theory to model the organization in multi-agent systems. They transform the AGR model into a formal model by using category theory. The obtained mathematical model enables the analysis and verification of organizational proprieties. Firstly, they explored the concepts of organization: Agent, Group, and Role. Second, they established categories and morphisms using these concepts.

In [25], the authors proposed a new approach for verifying MAS properties. They proposed a formal model of MAS by using categories, and morphisms between agents in category theory without focusing on the agent's architecture.

In [18, 19, 26] the authors have developed a formal specification approach for the organizational model among MAS. In [18], the same authors presented a formal approach to specifying, validating, and verifying MAS based on organizational models. They have chosen the OZS notation, which allows verification of formal specifications. Their approach is illustrated through the satisfaction-altruism application. The benefit of this approach is that you can reuse a range of models in different applications by decomposing the model into different reused formal concepts.

In [19], the authors developed a formal specification approach based on the organizational model RIO (Role-Interaction-Organization). The approach manages the issue of dynamic roles within organizational multi-agent systems. The authors used the OZ notation and illustrated their approach through the Satisfaction Altruism application. Finally, they analysed the final specification to check the behaviour of the agents.

In [26], the authors developed a formal prototyping approach of multi-agent systems designed by an organizational model. This formal specification is given by using a multi-formalism, which incorporates state charts into Object-Z to describe both reactive and transformational aspects and to facilitate the analysis operation of the specification. Finally, they examined the final specification to ensure that the agents behaved correctly.

Guerrouf and Chaoui [27] proposed an approach allowing a MAS to reorganize at run time. The authors have used the graph grammar to specify their systems. First, they described the organization using a type graph to model its structural aspects. Then, they specified the dynamic aspect of the system by defining a list of rules using the AGG tool. When an event occurs in the system, the equivalent rule is triggered to reorganize the system at runtime. Finally, the approach is validated by the supply chain management

application. In [28], the authors proposed a formal specification language called AgLOTOS to express agent plans from a set of agents' intentions. Each agent plan is described as a set of processes executed concurrently. In addition, the AgLOTOS semantics permit updating the plans according to the new agent's intentions. The approach aims to revise the agent's plan when their intentions change.

Another work [29] proposed a formal approach for modelling and analysing mobile agent-centred systems. The authors used the Mobile UML to model the system. Then, they proposed an automatic transformation approach to translating the concerned diagrams to the pi-calculus specification. The latter is analysed with analysis tools. Finally, the approach is implemented using the AToM3 tool.

In [30], the authors have proposed an algorithm to generate a specification of MAS from a Petri net model. A language named Maude is used to automatically generate the Maude specification. First, a MAS is designed with Petri nets as the input of the algorithm to generate its specification. Finally, they used this specification to verify their system.

Regarding the recent applications of the AGR model in complex problems, we can note the work proposed in [31]. Specifically, the authors presented a multi-agent system development framework, known as MaSMT, which is specially introduced to process English to Sinhala agent-based machine translation. The main reason for using this organizational model in MaSMT is the fact that AGR provides an agent's infrastructure, a communication process for the agents, and agent status controlling, as well as a tool for agent monitoring. About blockchain systems, AGR was recently used in [32] to propose a generic multi-agent organizational modelling for studying blockchain systems, known as AGR4BS. Precisely, the authors used AGR organizational model to identify and represent the generic entities that are common to blockchain systems. Moreover, the authors demonstrated via the use of four real case studies how this generic model can be introduced to model different blockchain systems. They also show how AGR4BS can be used to model three well-known attacks on blockchain systems.

3. BACKGROUND

3.1 Overview of AGR organizational model

The choice of an organizational approach to designing MAS is based on the specification of some structural and functional constraints that agents should adopt to perform their tasks easily within the system [33]. These constraints are generally described as an organizational model, such as the AGR model. With an organizational model, the system may manage many weaknesses such as uncertainty, complexity, and efficiency [34]. An organizational model is a coordination mechanism that controls the interactions and behaviour of the agents. Such features make this approach appropriate for developing distributed and complex systems.

The AGR model is proposed by Ferber in [11]. It complies with the organizational system principles . The AGR model is a new extension of the AALAADIN model [35]. It provides a good solution for building organization-based multi-agent systems.

The AGR model relies on three organizational concepts: agents, groups, and roles [36].

Agent: an agent is an autonomous and active entity that plays several roles in different groups. Each agent may be a member of different groups to play one or several roles. One of the AGR model principles is that no assumption is imposed on the agent's language and architecture.
Group: a group is a partition of a system. It contains agents with the same characteristics. Two agents are able to communicate if they belong to the same group. A group is an instance of a group structure, which describes it. This group structure defines interaction patterns between roles within a group.
Role: a role is a generic representation of the agent's behaviour within a group. A role may be held by several agents, and an agent must at least holed a role in a group.

In the AGR model, two types of constraints have been proposed: *dependence* and *correspondence*. A correspondence constraint means that when an agent holds one role will automatically hold another role. This constraint defines a representative agent between two groups. A dependency constraint between two roles R1 and R2 expresses that the agent acting R2 requires playing R1.

Several diagrams have been proposed to represent the organization. The organizational

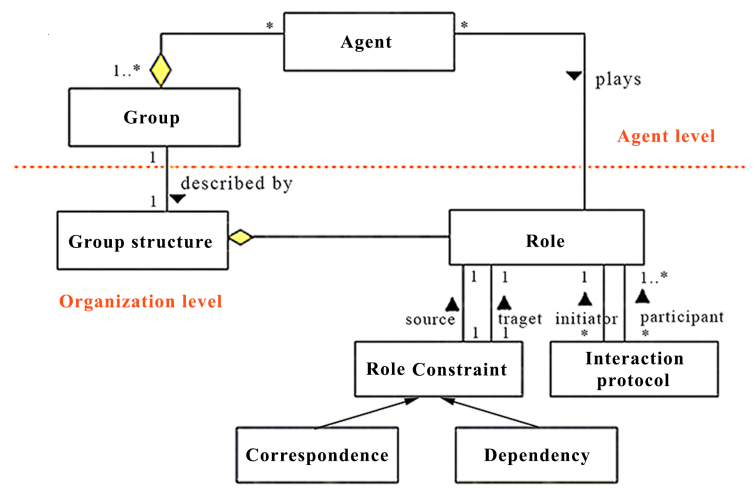

Fig. 1. The UML meta-model of Agent-Group-Role [11]

structure notation describes the organization (i.e. abstract level). In this diagram, groups are described by group structures, which define a set of interactions between roles. The two types of constraints are expressed as arcs between roles. The cheeseboard diagram is very suitable for representing the concrete organization (i.e. agent level). It is possible to instantiate several concrete organizations from one organizational structure. In addition, graphical elements are proposed to describe this diagram; ovals, skittles, and hexagons that represent respectively groups, agents, and roles. Lastly, organizational sequence diagrams are used to describe the organizational activities dynamics (dynamic aspect). The AGR meta-model is depicted in Fig. 1.

3.2 The DD-LOTOS formal language

The DD-LOTOS (Distributed Durational Language Of Temporal Ordering Specification) formal language is a powerful technique that permits the specification and the verification of the behaviour of distributed and complex systems. [37, 38, 39]. The DD-LOTOS language extends the D-LOTOS language [40] and is based on maximality semantics to support the duration of action. It takes into consideration three aspects: distribution, remote communication, and mobility of processes.

The distribution aspect is ensured by the existence of localities, and the communication between localities is carried out by the exchange of messages on communication channels. In its turn, the DD-LOTOS also allows for the mobility of processes between localities. Fig. 2 shows a distributed system that is composed of localities l and k that communicate via the communication channel b. Q, P, and E are processes, Q and P reside in the locality l and are synchronized through gate a. $l(E)$ denotes the behavioural expression E at the locality l. Therefore, the system is modelled as a set of processes that are executed in several localities. This system is represented by the following behavioural expression:

$$l(Q \mid [a] \mid P) \mid k(E)$$

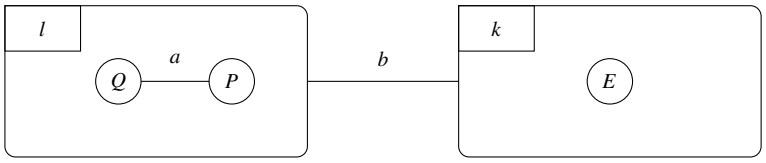

Fig. 2. Distributed system with DD-LOTOS.

3.2.1 Syntax of DD-LOTOS language

The syntax of DD-LOTOS uses some notations:

- Let $X, Y, Z...$ are processes identifiers.
- $Act = \mathscr{G} \cup \{i, \delta, go, create\}$, $\mathscr{G}$ is a range of observable actions a, b. $i \notin \mathscr{G}$ is a silent action, and $\delta \notin \mathscr{G}$ is the successful termination action, *go* and *create* provokes respectively the migration and the creation process.
- L is a part of $\mathscr{G}$
- $\mathscr{B}$ presents a range of behavioural expressions E, F,...
- $\mathscr{D}$ presents the time, $d \in D$ is a value of time.

Table 1 shows the essential DD-LOTOS elements:

- The temporal restriction is expressed by $a\{d\}$.

Table 1. Syntax of DD-LOTOS

$E ::=$	**Behaviours**	
	$stop \mid exit\{d\} \mid \Delta^d E \mid X[L] \mid$	
	$g@t[SP];E \mid i@t\{d\};E \mid hideLinE \mid$	
	$E[]E \mid E \mid [L] \mid E \mid E \gg E \mid E[> E \mid$	
	$a!v\{d\};E$	Emission
	$a?x;E$	Reception
	$go(l,\mathrm{E})\{d\}$	Migration
	$create(l,\mathrm{E})$	Creation of locality
$S ::=$	**Systems**	
	$\phi \mid S \mid S \mid l(E)$	

- $\Delta^d E$: this expression permits to delay the execution of the expression E by an unit of time d.
- In the behavioural expression $g@t[SP];E$, SP presents a logical predicate, and t expresses a variable of time. $HideLinE$ presents the hiding operator which hides some actions of the process and considers them internal to it. $E \mid [L] \mid E$ expresses a parallel composition, $E[]E$ expresses a non-deterministic choice, $E[> E$ represents a preemption, and $E \gg E$ expresses a sequential composition.
- $b!m\{d\};E$ presents an expression that sends a message m on the gate b. This transmission must be performed in a period of time [0,d].
- $a?x;E$ states the reception of a message through the gate a.
- $go(l,\mathrm{E})\{d\}$ migrates the behavioural expression E to an other locality l, this operation of mobility must be carried out in a delay d.
- The expression $create(l,\mathrm{E})$ creates a locality l with the behavioural expression E in the locality l.

The system may be a composition of several systems $S \mid S$, a behaviour E in a locality or empty ϕ.

3.2.2 Structured operational semantics

The DD-LOTOS has an operational semantics that is represented by a set of semantic rules:

- $a!v\{d\};E$
  The emission of the message v in the configuration ${}_M[a!v\{d\};E]$ cannot start until each action of the set M has completed its execution, in other words $Wait(M) = false$ as the rule 1 shows. The passing of time is expressed in the rules 2 and 3. The emission of message must be restricted by a certain time, otherwise; it is transformed to $Stop$ as depicted in Rule 4.

1. $$\frac{\neg Wait(M)}{{}_M[a!v\{d\};E] \overset{M^{a!v}{}_x}{\longrightarrow} {}_{\{x:a!v:t\}}[E]} \qquad x = get(\mathscr{M})$$

2. $$\frac{Wait(M^{d'}) or (\neg Wait(M^{d'}) and\ \forall \varepsilon{>}0.\ Wait(M^{d'-\varepsilon}))\ d'{>}0}{{}_M[a!v\{d\};E] \xrightarrow{d'} {}_{M^{d'}}[a!v\{d\};E]}$$

3. $$\frac{\neg Wait(M)}{{}_M[a!v\{d'+d\};E] \xrightarrow{d} {}_M[a!v\{d'\};E]}$$

4. $$\frac{\neg Wait(M)\ and\ d'{>}d}{{}_M[a!v\{d\};E] \xrightarrow{d'} {}_M[stop]}$$

- **distance communication**
  The flowing rule 1 expresses the receiving and sending of messages through the same gate:

1. $$\frac{-}{{}_M[l(a!v\{d\};E1)]|_{M'}[k(a?xE2)] \xrightarrow{\tau} {}_M[l(E1)]|_{M'}[k(E2\{v/x\})]}$$

- **creation of localities** $l(create(k,E))$
  The configuration ${}_M[l(F \mid create(k,E))]$ expresses different evolutions according to the actions of the set M. Rule 1 expresses that the creation of new localities can not perform until all actions in the set M have finished their execution, expressed by $wait(M) = false$. Rule 2 states that if the unit of time d is elapsed before the process of creating a locality is sensitized, then this process cannot evolved.

1. $$\frac{\neg Wait(M)}{{}_M[l(F|create(k,E))] \xrightarrow{M^{create}{}_x} {}_\phi[l(F)]|_{\{x:create:0\}}[k(E)]}$$

2. $$\frac{Wait(M^d) \qquad d{>}0}{{}_M[l(F|create(k,E))] \xrightarrow{d} {}_{M^d}[l(F|create(k,E))]}$$

- **Process of deletion of localities**
  If all processes in the locality are `Stop` then, this locality will be deleted as shown in the rule 1 and rule 2.

1. $$\frac{\neg Wait(M) \qquad and[l(Stop)]}{{}_M[l(Stop)] \xrightarrow{\delta} {}_\phi[\phi]}$$

2. $$\frac{\neg Wait(M) \qquad and \qquad {}_M[l(Stop)]|||_N[k(E)]}{{}_M[l(Stop)]|||_N[k(E)] \xrightarrow{\delta} {}_N[k(E)]}$$

- **Process of migration** ${}_M[(k(go(l,E)\{d\})]$
  The rule 1 expresses that if actions of the set M have not completed their execution, then the process of migration can not occur. Rule 2 expresses that the occurrence of the action go has for the period d, in the contrary case the migration will never. Rule 3, states the passage of time.

1. $$\frac{\neg Wait(M)}{{}_M[k(F|go(l,E)\{d\})] \xrightarrow{M^{go}{}_x} {}_\phi[k(F)]|_{\{x:go:0\}}[l(E)]} \qquad x = get(\mathscr{M})$$

2. $$\frac{\neg Wait(M) \qquad and \qquad d'{>}d}{{}_M[k(go(l,E)\{d\})] \xrightarrow{d'} {}_M[k(Stop)]}$$

3. $$\frac{\neg Wait(M) \qquad and \qquad d'>0}{{}_M[k(go(l,E)\{d+d'\})] \xrightarrow{d} {}_{M^d}[k(go(l,E)\{d'\})]}$$

- **Time evolution on system**

1. $$\frac{E \xrightarrow{d} E'}{l(E) \xrightarrow{d} l(E')}$$

2. $$\frac{S_1 \xrightarrow{d} S'_1 \quad S_2 \xrightarrow{d} S'_2}{S_1 \mid S_2 \xrightarrow{d} S'_1 \mid S'_2}$$

4. The proposed transformation approach

In this paper, we profit from the advantage of the DD-LOTOS language and the AGR model for the formal specification and validation of complex systems. Our approach generates DD-LOTOS specifications from the AGR model. The basic idea is to interpret each element in the AGR model into its equivalent in terms of DD-LOTOS language. As an example, an agent is transformed into a process in DD-LOTOS. In the AGR model, the system is specified using three major concepts: Agent, Group, and Role. Whereas in the DD-LOTOS language, the system is specified by a set of processes. For this reason, we consider in this approach that the system's behaviour is modelled as a set of interacting agents; each agent specifies the behaviour of an object in the system. These agents are transformed into a set of processes in the DD-LOTOS language.

The proposed approach constitutes of three steps. Firstly, all agents are transformed into DD-LOTOS processes. The generated processes in this step will constitute the reserved section for the declaration of processes in the DD-LOTOS specification. Secondly, all groups in the AGR model are transformed into localities in the DD-LOTOS language. A locality is an environment that will contain a set of processes generated in the first step from agents. Thirdly, in this latter step, the global specification of the system is established, and its behaviour is formed from a set of localities generated in the second step.

The global specification will be checked for errors using DD-LOTOS tools. In the following section, we will detail the transformation of each concept in the AGR model to its equivalent concept in the DD-LOTOS language.

4.1 Transformation of agents into DD-LOTOS processes

A role is an important element in the AGR model [11]. It describes the behaviour of an agent. On the other hand, a process in DD-LOTOS is an object defined by a behaviour expression that describes its behaviour. On this basis, an agent is transformed into the process in DD-LOTOS. The behaviour expression of this process is denoted by the role of the agent. Thus, agents playing roles in a group are converted into a set of processes in DD-LOTOS.

Fig. 3 illustrates the structure of the DD-LOTOS process generated from an agent. The name of process <Agent>is the name of the agent in the AGR model. Its gates are the set of agents' interactions. The behavioural expression of this process is the role played by the agent *role_played*. For example, agent A in the system sends a message M to agent B. In terms of DD-LOTOS, agents A and B are transformed into processes. The interaction

between agents is converted into a gate named g. The role played by agent A (sending message) is transformed into behaviour expression of the process A expressed as follow: g! M, which means transmission message M via the gate g.

```
process <Agent>[gate_List] :=
behavior
  role_played[gate_list]
where
  process <role_played>[gate_list]:=....endproc
endproc
```

Fig. 3. Structure of the DD-LOTOS process

4.2 Transformation of groups into localities

The AGR model specifies a system as a set of groups. Each group contains a set of interacting agents. In the second step, each group is transformed into a locality in DD-LOTOS. The name of the locality is the name of the specified group. The behaviour expression E of the locality is formalized by agents of the group. Agents in a group are transformed into processes and composed by the parallel composition operators:

i) These processes are composed using the partial synchronization operator $||[< messages >]||$ if there exist messages (interactions) between two agents. Messages between agents are transformed into synchronization gates of the operator. This operator is used when simultaneous processes synchronize on the gates mentioned in the operator.

ii) If there is no interaction (messages) between agents, we use the interleaving operator $|||$ instead the partial synchronization operator. The interleaving operator is used when two parallel processes run completely independently without synchronization (the list of gates is empty in this operator).

iii) In the case where each agent has to interact with every agent in the same group, the full synchronization operator $||$ must be used. The full synchronization operator means that two processes executed in parallel have to synchronize on every gate. For example, we use the AGR model to design an organization. The organization consists of two groups: Group 1 and Group 2. The first group contains two interacting agents A and B. The formalization of this system by using the DD-LOTOS language will be as follow: first, both groups A and B are transformed into localities: locality 1 and locality 2. Second, agents A and B are transformed into processes respectively named A and B. These processes are composed using the partial synchronization operator for modelling the behaviour expression E of locality 1. Message m between the two agents is transformed into a synchronization gate of the operator as follows: $A|[m]|B$, which means that two processes A and B synchronize on the gate m. The use of the partial synchronization operator is justified by the existence of an interaction between agents.

4.3 Building the global specification

In the third step, the global specification is built and its behaviour expression is obtained from groups. Groups are used to model the behavioural expression E of the system.

Fig. 4 shows the essential elements of the global specification. The name of specification <specification_name>is the name of the system specified by the AGR model. The behaviour expression of this specification is derived from groups that model the system's behaviour. Groups in the AGR model are formalized as localities and composed using the system composition operator | as follow: $Locality1(E) \mid Locality2(F)\ldots. \mid Localityn(Q)$. Knowing that E, F, and Q denote behavioural expressions of localities. Processes generated from agents in the first step are defined in the section *'where'* of the specification.

```
specification <specification_name>:=
behavior
  l(E) | k(F) | p(Z)
where
process E [gate_list]:=
  Agent_1 [gate_list] || Agent_n [gate_list]
where
  process <Agent_1>[gate_list]:=
  ...
  endproc
  ...
  process <Agent_n>[gate_list]:=
  ...
  endproc
endproc
...
endspec
```

Fig. 4. Structure of generated DD-LOTOS specification

Illustrative example

To illustrate our approach, we considered a simple organization of the reviewing process [11]. The organization is established according to the AGR organizational model. Fig. 5 shows part of the organization that consists of two groups of agents.

The first group consists of two agents: the *author* and the *receiver*. The second group also

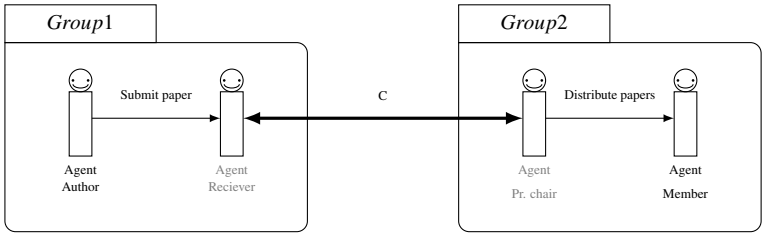

Fig. 5. Agent-Group-Role based organization of reviewing process.

contains *members* and *program chairs*. An *author* agent submits its paper to the *receiver*, who is also the program chair agent in the second group. (A representative agent is defined between groups to coordinate the interactions by the correspondence constraint *C*).

Table 2. The DD-LOTOS specification of a system designed by the AGR model

Specification $Reviewing\_process[subm, distrib] :=$
Behaviour
  $Group1(E) \mid Group2(P)$
Where
Process $E[subm] :=$
  $Author[subm] \mid [subm] \mid Receiver[subm]$
Where
Process $Author[subm] :=$
  $subm\,!\,Submit\,;\,exit$
EndProc
Process $Receiver[subm] :=$
  $subm\,?\,x : Message\,;\,exit$
EndProc
EndProc
Process $P[distrib] :=$
  $Pr.chair[distrib] \mid [distrib] \mid Member[distrib]$
Where
Process $Pr.chair[distrib] :=$
  $distrib\,!\,Distribute\,;\,exit$
EndProc
Process $Member[distrib] :=$
  $distrib\,?\,x : Message\,;\,exit$
EndProc
EndProc
EndSpec

The *program chair* sends the paper to the *member* to be examined. *Submit* and *Distribute* are interactions between agents.To transform this system into a DD-LOTOS specification, we consider that a system consists of two localities: $Group1$ and $Group2$. E and P are their behavioural expressions, respectively. Therefore, The system can be defined by the following behaviour expression $Group1(E) \mid Group2(P)$.

The behaviour expression $E := Author[subm] \mid [subm] \mid Receiver[subm]$ specifies two processes: *author* and *receiver*, executed in locality $Group1$. The processes are generated from agents in $Group1$ and are synchronized on the gate. The message "Submit paper" is transformed into the gate subm in DD-LOTOS. In $Group2$, the behavioural expression $P := Pr.chair[distrib] \mid [distrib] \mid Member[distrib]$ specifies two processes: *Pr.chair* and *Member* that are generated from agents. The message *Distribute papers* between agents is transformed into a synchronization gate. Agents in the Group 2 are transformed into processes and are composed using the synchronization operator $\mid [synch\_gates] \mid$ to form the behaviour of the locality. Interactions between agents are transformed into synchronization gates in the operator. The role enacted by an agent is transformed into sub-process that expresses the the agent's behaviour. The global specification of the system in DD-LOTOS is given in Table 2.

5. Automatic generation of the DD-LOTOS specification

In this section, we explain the automation of the proposed approach to generate DD-LOTOS specifications from the AGR models. To this end, our approach passes on two main phases (Fig. 6): first, we implement a meta-model for the AGR model using the Eclipse Modelling Framework (EMF). Second, a model-to-text transformation is done to generate DD-LOTOS specification by using the EMF model and the Xpand tool (https://eclipse. org/modeling/m2t/?project=Xpand).

The Xpand is a section of the Open Architecture Ware platform (OAW). It is used for a model-to-text transformation (M2T). The selection of Xpand for generating code is justified by its easiness and clarity of use, as well as its integration into the Eclipse framework as a plug-in. It allows us to exploit the XMI format models created by the EMF tool in the first step. It uses templates that control code generation. Our approach helps developers to design multi-agent systems using the AGR model and then generate the DD-LOTOS formal model for validation purposes.

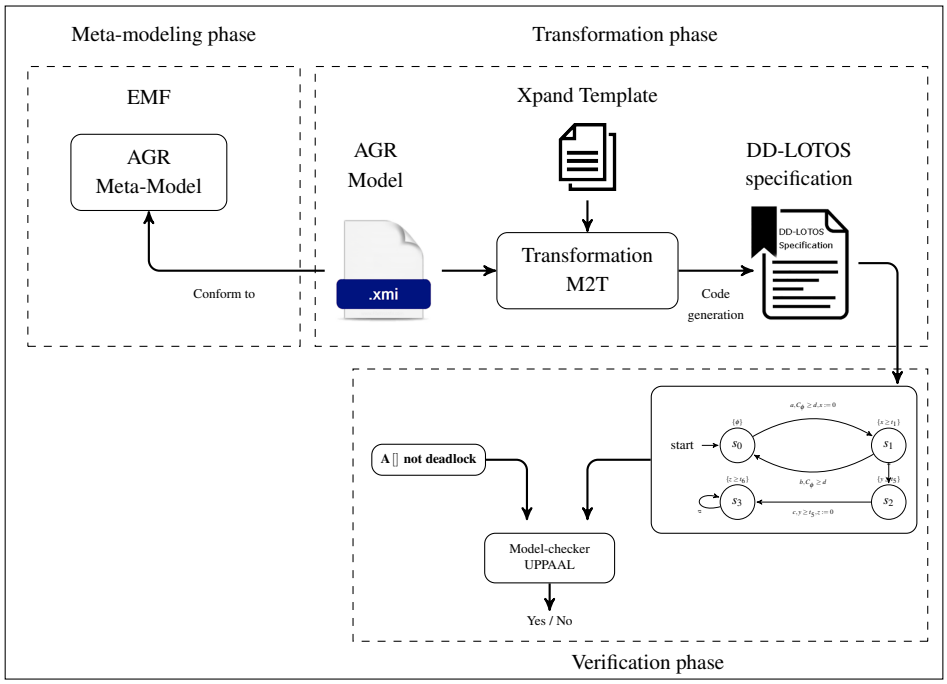

Fig. 6. Proposed transformation approach.

5.1 The AGR meta-model

We have implemented our approach in the Eclipse platform, using especially EMF (Eclipse Modelling framework) to develop the meta-model of the AGR model. EMF is an environment specific for meta-modelling integrated into the Eclipse platform. It facilitates the construction of tools and applications based on structured data models. It is

composed of a set of plug-ins, among these plug-ins we mention the Ecore meta-model, which is used to describe EMF models. The AGR meta-model has an abstract syntax that facilitates the specification of the static and dynamic aspects of the AGR model. By having the AGR meta-model, the developer can generate different models specified in the AGR formalism without understanding this meta-model. Since AGR models consist of agents, groups, and roles, three main classes are proposed to meta-model the AGR: ”Agent”, ”Group”, and ”Role” to describe respectively agents, groups, and roles. A class ”Interaction” to describe interactions between agents. An interaction can occur between different agents and is specified at the organizational level within roles. Each interaction has one initiator role and one or more participating roles, as shown in Fig. 7. Further, two classes ”dependency” and ”correspondence” are defined to describe types of structural constraints that are imposed on agents. They inherit from a class ”RoleConstraint”. A role specifies constraints, which an agent should satisfy to obtain a role. For this reason, a structural constraint is described between roles in the meta-model. A class ”group-Structure” describes groups, because each group is an instance of one group structure that contains a set of roles.

After having developed the AGR meta-model, we are capable now, to generate DD-LOTOS specifications by using the Xpand tool.

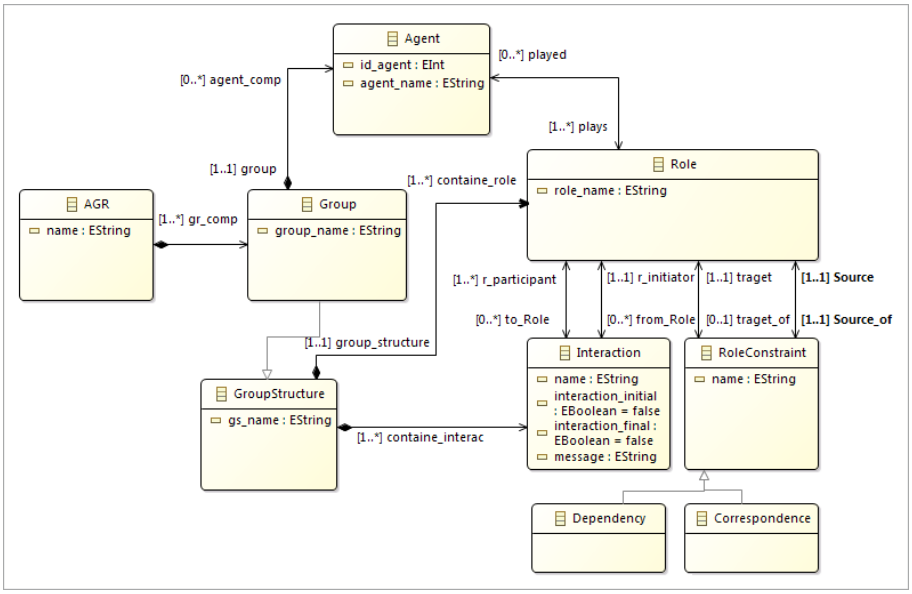

Fig. 7. The meta-model of the AGR model.

5.2 Model-to-Text (M2T) Transformation of the AGR model to DD-LOTOS specification

The Model to Text transformation concentrate on the generation of code from described models. There are two approaches to model-to-code transformation. Approaches

based on the principle of templates are currently the most used. The code generation typically uses templates. The template is a target code contains sections (parts) of the meta-code executed during the instantiation of the source model to calculate variable parts.

In this phase of our approach, we explain code generation from EMF models using Xpand templates. The process passes in three steps. First, we use the AGR model described in the XMI model (an XML format to exchange models), which is conform to the AGR meta-model implemented in Fig. 7 to write specification generation templates (Fig. 8) in the .xpt file. Second, we use the Check language to specify some constraints that the XMI model must satisfy to be correct. Specified constraints in the Check language are stored in the *.chk* file. This file starts with an import of the meta-model according to the format "*import metamodel*;". Each constraint is specified in a context, which is a meta-class of the imported meta-model, to which the constraint applies. Constraints can be of two types: *Warning* or *Error*.

The following example shows an "Error" constraint to verify that a group structure is empty and must contain at least one role, for models conforming to the meta-model "AGRModel".

import AGRModel;
extension metamodel::Extensions;
context GroupStructure ERROR "No roles defined in group-structure:" +gsName :
this.containeRole.exists(e| Role.isInstance(e));

Template.xpt | e-commerce.DDLotos | Checks.chk | AGRModel.ecore | AGR.xmi

```
«IMPORT AGRModel»
«DEFINE principal FOR AGRModel::AGR»
«FILE this.name+".DDLotos" -»
Specification «this.name -»
«" ["-»«FOREACH this.gr_comp.containe_role.from_Role AS gate_out SEPARATOR "," -»«gate_out.name -»«ENDFOREACH -»«"]"-»«":= "»
Behavior
«IF !this.gr_comp.isEmpty-»
«EXPAND locality FOREACH gr_comp SEPARATOR " | "-»
«ENDIF»
Where
«EXPAND locality_proc FOREACH gr_comp-»
Endspec
«ENDFILE -»
«ENDDEFINE -»
«DEFINE locality FOR AGRModel::Group-»
«this.gs_name-»«"("»«group_name-»«")"-»
«ENDDEFINE-»
«DEFINE locality_proc FOR AGRModel::Group -»
Process «group_name -»«"["-»
«FOREACH containe_role.from_Role AS gate_out SEPARATOR "," -»«gate_out.name -»«ENDFOREACH -»«"]"-»«":= " -»
«IF !containe_role.isEmpty -»
«EXPAND agent_comp FOREACH containe_role SEPARATOR " || "»
Where
«EXPAND agent_behavior FOREACH containe_role-»
Endproc
«ENDIF -»
«ENDDEFINE -»
«DEFINE agent_comp FOR  AGRModel::Role -»
«IF !played.isEmpty -»
«"("-»«FOREACH played AS play SEPARATOR "|||"-»
«play.agent_name -»
«"["-»
«FOREACH from_Role AS gate_out SEPARATOR "," -»«gate_out.name -»«ENDFOREACH -»«","-»
«FOREACH to_Role AS gate_in SEPARATOR "," -»«gate_in.name -»«ENDFOREACH -»«"]"-»
«ENDFOREACH-»«")"-»
```

Fig. 8. A template for DD-LOTOS code generation

Finally, we run the template to generate the specification. In the end, we should have a specification generation. The specification is composed of a set of processes executed in a concurrent way using parallel composition operators. Once we have a formal model of the AGR model in the DD-LOTOS language, we will focus on the verification of the

specified system with formal verification. For this reason, a semantic model specific to the DD-LOTOS language called C-DATA is generated, which is an extension of the timed automatons. We can use the UPPAAL model checker to check the properties of the smooth behaviours; these properties are expressed in a temporal logic such as the TCTL logic.

6. Case study

To illustrate the applicability of our approach, we show how a multi-agent system designed by the AGR model is transformed into a DD-LOTOS specification to be checked later using the UPPAAL model checker.

6.1 Description

We consider a well-known e-commerce example [11] that is an abstraction of a travel agency example. Customers try to buy products from an agency via a brokering agent for the agency. The system consists of three interacting group structures (Fig. 9). A group structure of clients named **ClientGS** interacts with a brokering agent. A group structure of providers, named **ProviderGS** interacts with a brokering agent, and a group structure of contracts named **ContractGS** is created to establish a contract between client and provider when a customer decides to buy the product.

To get the product, the client must join a client group and then requests the product from the broker agent that resides in the same group. The broker (an agent that enacts the broker role in both groups client and provider) diffuses a call for proposals to several providers in the second group (Providers). Different proposals are presented to the client, who decides and chooses one of them (this requires many interactions between broker and providers). In that case, a contract group is established with two new roles Buyer and Seller respectively enacted by the client and the selected provider. In this process, we have used the organizational structure diagram (organizational level) to design this organization using only organizational concepts (group structures, roles, and patterns of activity). The use of organizational level is a new manner to describe two static and dynamic aspects of the system. This level is responsible for representing the concrete organization (agent level) abstractly, i.e. specifying the expected patterns of interactions between roles in the organization.

6.2 Code generation

After specifying our e- commerce system example with the AGR model, we are now capable to generate the DD-LOTOS specification by realizing the model-to-text transformation. First, we develop the EMF model representing our system. The model, which is conforms to the AGR meta-model, presents the concrete organization of the system. It is stored in the *.xmi* file. As shown in Fig. 10, the organization consists of three agent groups: the Client group contains the client and broker agents, the Provider group contains provider and broker agents, and the Contract group contains seller and buyer agents. We define a correspondence constraint between the broker role in the Client group and the broker role in the Provider group to identify a representative agent between the two groups (an agent that plays the broker role in the Client group will automatically play

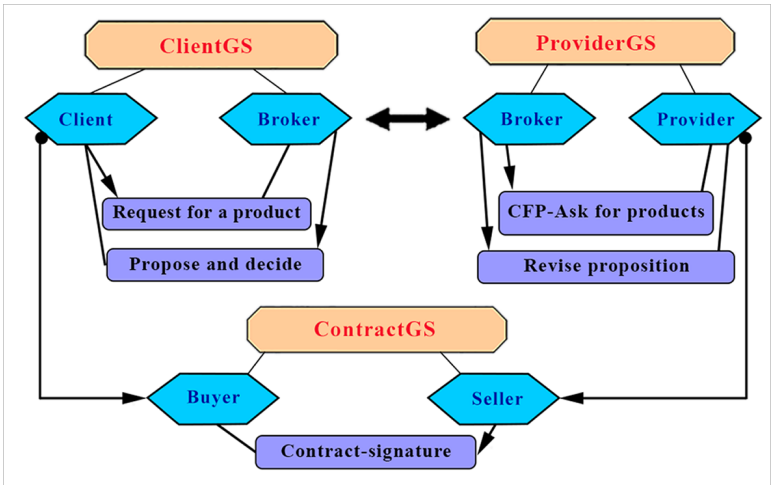

Fig. 9. Organizational structure diagram of an e-commerce example [11]

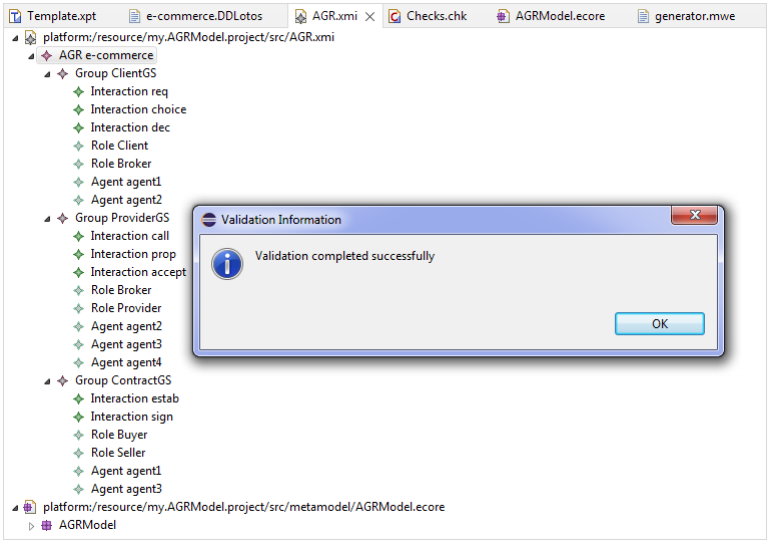

Fig. 10. XMI model of the AGR model

the same role in the Provider group). Second, the model-to-text transformation uses the Xpand templates with the XMI model, generated by the EMF tool, to generate a specification in the DD-LOTOS language. The transformation is obtained by a simple click on the 'run' button. Finally, a file with the *.DDlotos* extension is generated and contains the DD-LOTOS specification (Fig. 11).

Each concept in the XMI model (agents, groups, roles, and interactions) is transformed into its equivalent in the DD-LOTOS specification. In the generated textual spec-

ification, three localities are generated from groups specified in the XMI model. These localities are named Client, Provider, and Contract. The global behaviour expression of the final specification is formalized from the composition of these localities using the system composition operator | as follow: ClientGS(E) | ProviderGS(F) | ContratGS(P).

Agents playing roles are transformed into processes. These processes are composed by using parallel composition operators to formalize the behaviour expressions (E, F, and P) of each locality. Interactions between agents in the XMI model are transformed into gates of generated processes. All roles specified in the XMI model describe the agents' behaviours in the global specification. In our example, five sub-processes (behaviours) are generated from roles that are: client, broker, provider, buyer, and seller. These sub-processes are defined in the processes definitions section after the keyword 'where' in the specification. The final specification consists of a set of concurrent processes.

Once formal specification is generated in DD-LOTOS language, the final step will focus on the verification of the smooth behaviour of the system using formal verification. This last step will be carried out by calling the semantic model C-DATA with the UPPAAL model checker to check some organization properties expressed in a temporal logic such as the TCTL logic.

Template.xpt | e-commerce.DDLotos | AGR.xmi | Checks.chk | AGRModel.ecore | generator.mwe

```
Specification e-commerce [req,dec,choice,call,accept,prop,sign,estab]:=
Behavior
ClientGS(E) | ProviderGS(F) | ContractGS(P)
Where
Process E[req,dec,choice]:= (agent1[req,dec,choice]) || (agent2[req,dec,choice])
Where
Process agent1[req,dec,choice]:= Client[req,dec,choice]
where
Process Client[req,dec,choice]:=
        req!Request for a product;
        choice?X:Message;
        dec!Decide;exit
Endproc
Endproc
Process agent2[req,dec,choice]:= Broker[req,dec,choice]
where
Process Broker[req,dec,choice]:=
        req?X:Message;
        choice!Request-choice;
        dec?X:Message;exit
Endproc
Endproc
Endproc

Process F[call,accept,prop]:= (agent2[call,accept,prop]) || (agent3[prop,call,accept]|||agent4[prop,call,accept])
Where
Process agent2[call,accept,prop]:= Broker[call,accept,prop]
where
Process Broker[call,accept,prop]:=
        call!Call-for-proposal;
        prop?X:Message;
        accept!Accept-proposal;exit
Endproc
Endproc
```

Fig. 11. The DD-LOTOS specification generated for the e-commerce system

7. Conclusion and future work

Several organizational models are used to design multi-agent systems. However, these models lack formal semantics that can hinder the formal verification of these systems. Therefore, it is necessary to use formal methods with organizational models to

specify organizational multi-agent systems.

In the literature, several approaches have been proposed to address the lack of formal semantics in the AGR model. The majority of these approaches define a mapping between the AGR model and process algebras such as LOTOS, CCS, Petri nets, and their extensions, as well as the rewriting logic and its formal language MAUDE. The main limitation of these approaches is that they are defined on interleaving semantics, which does not allow verifying properties concerning the durations of action executions.

In this paper, we defined formal semantics for the AGR organizational model using the DD-LOTOS language that is defined on true concurrency. For this aim, we have proposed an automatic model-to-text transformation approach that generates the DD-LOTOS specification from the AGR model. The suggested approach permits capturing structural and dynamic aspects of organizational multi-agent systems. This formalization enables the creation of a formal model that can be used to validate specific organizational properties. We have used a set of tools in the Eclipse environment to automate this transformation. The EMF tool is used for the meta-modelling and the Xpand language is for the generation of code from the AGR model. Finally, we have illustrated our transformation approach through the e-commerce case study.

In future work, we intend to use this approach as a library to propose a generic approach that can be applied to other organizational models such as the AGRE and MOISE models using new organizational concepts. In addition, we intend to use the C-DATA (Communicating Durational Action Timed Automaton) semantics model, an interpretation model of the DD-LOTOS specifications in the verification and validation stage, to verify some formal properties via model verification tools.

ACKNOWLEDGMENT

REFERENCES

1. N. R. Jennings, "An agent-based approach for building complex software systems," *Communications of the ACM*, Vol. 44, 2001, pp. 35–41.
2. M. Wooldridge, *An introduction to multi-agent systems*. John wiley & sons, 2009.
3. J. Ferber and G. Weiss, *Multi-agent systems: an introduction to distributed artificial intelligence*. Addison-wesley Reading, 1999.
4. G. Picard, J. F. Hübner, O. Boissier, and M. P. Gleizes, "Reorganisation and self-organisation in multi-agent systems," in *1st International Workshop on Organizational Modeling, ORGMOD*, 2009, pp. 66–80.
5. J. Upton, I. Janeka, and N. Ferraro, "The whole is more than the sum of its parts: Aristotle, metaphysical," *Journal of Craniofacial Surgery.*, Vol. 25, 2014, pp. 59–63.
6. M. Wooldridge, N. R. Jennings, and D. Kinny, "The Gaia methodology for agent-oriented analysis and design," *Autonomous Agents and multi-agent systems.*, Vol. 3, 2000, pp. 285–312.
7. K. S. Barber and C. Martin, "Dynamic reorganization of decision-making groups," in *Proceedings of the fifth international conference on Autonomous agents*, 2001, pp. 513–520.

8. M. Kolp, P. Giorgini, and J. Mylopoulos, "Organizational patterns for early requirements analysis," in *International Conference on Advanced Information Systems Engineering*, 2003, pp. 617–632.
9. N. R. Jennings, "On agent-based software engineering," *Artificial intelligence*, Vol. 117, 2000, pp. 277–296.
10. B. Horling and V. Lesser, "Quantitative organizational models for large-scale agent systems," in *International Workshop on Massively Multiagent Systems*, 2004, pp. 121–135.
11. J. Ferber, O. Gutknecht, and F. Michel, "From agents to organizations: an organizational view of multi-agent systems," in *International workshop on agent-oriented software engineering LNCS*, 2004, pp. 214–230.
12. J. Ferber, F. Michel, and J. Báez, "Integrating environments with organizations," in *International workshop on environments for multi-agent systems*, 2004, pp. 48–56.
13. J. F. Hübner, J. S. Sichman, and O. Boissier, "A model for the structural, functional, and deontic specification of organizations in multi-agent systems," in *Brazilian Symposium on Artificial Intelligence*, 2002, pp. 118–128.
14. M. E. H. Souid, P. Songhao, L. Guo, and C. Lin, "Multi-agent cooperation pursuit based on an extension of AALAADIN organisational model," *Journal of Experimental & Theoretical Artificial Intelligence*, Vol. 28, 2016, pp. 1075–1088.
15. M. E. H. Souidi, A. Siam, and Z. Pei, "Multi-agent pursuit coalition formation based on a limited overlapping of the dynamic groups," *Journal of Intelligent & Fuzzy Systems*, Vol. 36, 2019, pp. 5617–5629.
16. E. Argente, V. Julian, and V. Botti, "Multi-agent system development based on organizations," *Electronic Notes in Theoretical Computer Science*, Vol. 150, 2006, pp. 55–71.
17. C. G. Troya and M. A. Vallecillo, "A rewriting logic semantics for ATL," *Journal of Object Technology*, Vol. 150, 2011, pp. 1–29.
18. V. Hilaire, O. Simonin, A. Koukam, and J. Ferber, "A formal approach to design and reuse agent and multiagent models," in *Agent-Oriented Software Engineering V: 5th International Workshop, AOSE 2004, New York, NY, USA, July 19, 2004. Revised Selected Papers 5*, 2005, pp. 142–157.
19. V. Hilaire, P. Gruer, A. Koukam, and O. Simonin, "Formal specification approach of role dynamics in agent organisations: Application to the Satisfaction-Altruism Model," *International Journal of Software Engineering and Knowledge Engineering*, Vol. 17, 2007, pp. 615–641.
20. M. A. Laouadi, F. Mokhati, and B. H. Seridi, " A formal framework for organization-centered multi-agent system specification: A rewriting logic based approach," *Multiagent and Grid Systems*, Vol. 13, 2017, pp. 395–419.
21. S. Abderrahim and R. Maamri, "A Category-theoretic Approach to Organization-based Modeling of Multi Agent Systems on the Basis of Collective Phenomena and Organizations in Human Societies," *Informatica*, Vol. 42, 2018, pp. 563–576.
22. A. Boudjidj, E. Merah, and M. E. H. Souidi, "Towards a formal multi-agent organizational modeling framework based on category theory," *Informatica*, Vol. 45, 2021, pp. 277–288.
23. T. M. Maarouk, D. E. Saidouni, and M. Khergag, "Dd-lotos : A distributed real time language," in *Proceedings 2nd Annual International Conference on Advances in Dis-*

tributed and Parallel Computing (ADPC 2011) Special Track: Real Time Embedded Systems, 2011, pp. 45–50.
24. T. M. Maarouk, D. E. Saidouni, R. Mahdaoui, and H. Houassi, "Interpretation of DD-LOTOS Specification by C-DATA," *Communications in Computer and Information Science*, Vol. 539, 2015, pp. 414–423.
25. O. Ormandjieva, J. Bentahar, J. Huang, and H. Kuang, "A Category-theoretic Approach to Organization-based Modeling of Multi Agent Systems on the Basis of Collective Phenomena and Organizations in Human Societies," *Procedia Computer Science*, Vol. 52, 2015, pp. 538–545.
26. V. Hilaire, A. Koukam, P. Gruer, and J. P. Müller, "Formal specification and prototyping of multi-agent systems," in *International Workshop on Engineering Societies in the Agents World*, 2000, pp. 114–127.
27. G. Fayçal and A. Chaoui, "A graph transformation based approach for multi-agent systems reorganization," *Multiagent and Grid Systems*, Vol. 15, 2019, pp. 375–394.
28. A. C. Chaouche, A. E. Seghrouchni, J. M. Ilié, and D. E. Saidouni, "A dynamical plan revising for ambient systems," *Procedia Computer Science*, Vol. 32, 2014, pp. 37–44.
29. A. Belghiat and A. Chaoui, "A multi-paradigm approach to model and verify mobile agent software systems." *Multiagent and Grid Systems*, Vol. 14, 2018, pp. 337–356.
30. A. Boucherit, A. Khababa, and L. M. Castro, "Automatic generating algorithm of rewriting logic specification for multi-agent system models based on petri nets," *Multiagent and Grid Systems*, Vol. 14, 2018, pp. 403–418.
31. B. Hettige, A. S. Karunananda, G. Rzevski, and A. Meffre, "Masmt4: The agr organizational model-based multi-agent system development framework for machine translation," in *Inventive Computation and Information Technologies: Proceedings of ICICIT 2020. Springer Singapore*, 2021, pp. 691–702.
32. H. Roussille, Ö. Gürcan, and F. Michel, "AGR4BS: A generic multi-agent organizational model for blockchain systems," *Big Data and Cognitive Computing*, Vol. 06, 2021, p. 1.
33. H. A. Abbas, S. I. Shaheen, and M. H. Amin, "Organization of multi-agent systems: an overview," *arXiv preprint arXiv:1506.09032.*, Vol. 4, 2015, pp. 46–57.
34. B. Horling and V. Lesser, "A survey of multi-agent organizational paradigms," *The Knowledge engineering review*, Vol. 19, 2004, pp. 281–316.
35. J. Ferber and O. Gutknecht, "A meta-model for the analysis and design of organizations in multi-agent systems," in *Proceedings international conference on multi agent systems (Cat. No. 98EX160)*, 1998, pp. 128–135.
36. R. C. Patrice, N. Lachiche, P. Gançarski, A. Meffre, and C. Collet, "Generic architecture for ambient intelligence based on an organizational centered multi-agent approach," in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, 2012, pp. 786–789.
37. T. M. Maarouk, E. Merah, S. Ghaoui, and N. Rahabi, "Formal semantics and transformation of BPMN models," *International Journal of Business Process Integration and Management.*, Vol. 9, 2019, pp. 158–169.
38. T. M. Maarouk, M. E. H. Souidi, and N. Hoggas, "Formalization and Model Checking of BPMN Collaboration Diagrams with DD-LOTOS," *Computing and Informatics*, Vol. 40, 2021, pp. 1080–1107.

39. T. M. Maarouk, “Modèles formels pour la conception des systèmes temps réel,” 2012, université Frères Mentouri-Constantine 1.
40. D. E. Saïdouni and J. P. Courtiat, “Prise en compte des durées d’action dans les algèbres de processus par l’utilisation de la sémantique de maximalité,” 2003, proceedings of CFIP.

Sabeg Samra She obtained her degree in Computer Science from the Batna University, Algeria in 2006. She worked in formal verification. She received her MSc in Computer Science from the Khenchela University, Algeria in 2019. She is a PhD student at the Khenchela University, Algeria.

Toufik Messaoud Maarouk is Lecturer in the Department of Mathematics and Computer Science, Faculty of Sciences and Technology, University of Khenchela, Algeria. He received his Ph.D. in computer science from the Constantine University, Algeria, in 2012. His main areas of research include formal methods, concurrency theory, formal semantics and distributed computing.

Mohammed El Habib Souidi is Lecturer in the Department of Mathematics and Computer Science, Faculty of Sciences and Technology, University of Khenchela, Algeria. He received his Ph.D. in computer science from the Harbin Institute of Technology (China), in 2017. His main areas of research include multiagent task coordination, reinforcement learning, game theory and path planning.