Towards a Multi-Agent Model of the Decisional Subsystem of Distributed Industrial Systems: an Organizational and Formal Approach

Stéphane GALLAND, Nicolas GAUD, Abderrafîâa KOUKAM

Multi-Agent System Group,
Systems and Transport Laboratory,
University of Technology of Belfort-Montbéliard, France
{stephane.galland,nicolas.gaud,abder.koukam}@utbm.fr

ABSTRACT

In this paper we propose the formalization of the stage *Design* of our methodological approach $\mathcal{M}_{\mathcal{A}}\mathcal{MA}-\mathcal{S}_{\text{DIS}}$. This methodology allows the modeling of systems from the class of the distributed industrial problems, such as enterprise consortiums and virtual enterprises. The stage *Design* of $\mathcal{M}_{\mathcal{A}}\mathcal{MA}-\mathcal{S}_{\text{DIS}}$ provides a multi-agent model of the system. It uses the organizational method RIO and its formal multi-formalism language OZS.
The proposed formal model focus on the decisional subsystem. We present three organization's definitions and the interactions between their composing roles. Thus we formalize and explain the abstract definitions required for the modeling of the agent organizations. Finally we focus on the *Trader* role from the Trading organization. The abstract definitions illustrate the formal modeling of the agents and their interactions with the agents' environment. The *Trader* example highlights the modeling of an agent interacting with other agents.

Keywords: Multi-Agent System, RIO, OZS, Virtual Enterprise, Formal Model, Design Stage, Object-Z, Statechart

1. INTRODUCTION

Simulation is already recognized as an effective technology to solve modern industrial problems, by allowing to support and study the dynamic behavior of these systems. Even if the simulation is a powerful tool, industrial systems are still difficult to engineer and thus some problems still exist. First, the simulation tools are seldom packaged with a methodology, which should allow an easier modeling of an industrial system. Second, components and modular modeling are still poorly supported. It remains difficult to reuse existing models (or a part of them) without substantial changes or a complete rewriting. The physical, informational and decisional aspects of an industrial system are often imbricated and not clearly separated. For example, simulate a system with a pulled-flow management policy, and then change it to use a pushed-flow policy, often force the designer to completely rewrite the simulation model, even if only the decisional part has changed.

In this paper, we propose the formalization of the stage *Design* of the methodological approach $\mathcal{M}_{\mathcal{A}}\mathcal{MA}-\mathcal{S}_{\text{DIS}}$ [10, 11]. It intends to provide a solution to those problems and allowing the modeling of complex industrial organizations independently of any software platform. The modeling of industrial systems or manufacturing systems with or without multi-agent systems (MAS) was already explored by several works [2, 19, 24]. And some of them focus on formal specification approaches [4, 18, 12].
The formal model of distributed industrial systems[1], that we propose in this paper, is based on the organizational multi-agent approach RIO [16], and on its formal counterpart OZS [13]. RIO is based on three main concepts : Role, Interaction and Organization.

Our formalization of the stage *Design* of $\mathcal{M}_{\mathcal{A}}\mathcal{MA}-\mathcal{S}_{\text{DIS}}$ is intented to complete the initial methodology. It provides an adaptation of the original multi-agent model using an organizational approach, and the associated formalization of the agent roles. This work is part of larger effort to define a well-founded approach for industrial system engineering. The proposed model is a part of the modeling chain in which the model's translations between each ring do not alter the model's semantic.

In the following section, the methodological approach $\mathcal{M}_{\mathcal{A}}\mathcal{MA}-\mathcal{S}_{\text{DIS}}$ is introduced The section 3 presents the used organizational model and a short example about the decisional subsystem. The base classes are detailled in the section 4. They are used to build inherit classes of the industrial system model. And before concluding, an example of a part of the decisional subsystem model is presented.

2. OVERVIEW OF $\mathcal{M}_{\mathcal{A}}\mathcal{MA}-\mathcal{S}_{\text{DIS}}$

As mentioned is the introduction, the methodological approach $\mathcal{M}_{\mathcal{A}}\mathcal{MA}-\mathcal{S}_{\text{DIS}}$ is dedicated to the simulation models of distributed industrial systems. Our goal is to allow the simulation of manufacturing systems, which are complex and distributed (enterprise network, strongly distributed enterprise, etc.). We are interested by the systems which always include a distribution aspect. But, we are not limited to the distribution on a computer network. In fact, our interest is also on the distribution of the information and of the decision-making processes. But, the existing tools do not fully support these two last aspects. Moreover, they are domain dependent, e.g., AReVi for virtual representation of systems [6], SWARM for simulation of articial way of life [3]; or they only include one aspect of the distribution, e.g., HLA is a architecture that supports the interoperability of models. In another point of view, the underlying models for simulation (Petri nets, queuing theory, etc.) and the distributed simulation techniques are partly adapted to the simulation of distributed industrial systems. In fact, even if they permit to realize the simulation process, they do not really support the specific constraints of the distributed industria systems (confidentiality, dynamics, etc.).

Starting from the assets of the software engeneering and the works already completed in the field of simulation, we

1-4244-0451-7/06/$20.00 ©2006 IEEE

[1]Depending on the problem, it could be an Enterprise Consortiums, a Virtual Enterprise, an Extended Enterprise or a Network of Enterprises

Published in the Special Session "Multi-Agent Systems for Supply Chain Management" of the IEEE International Conference on Service Systems and Service Management (IC-SSSM06), pp. 859–865. Troyes, France.

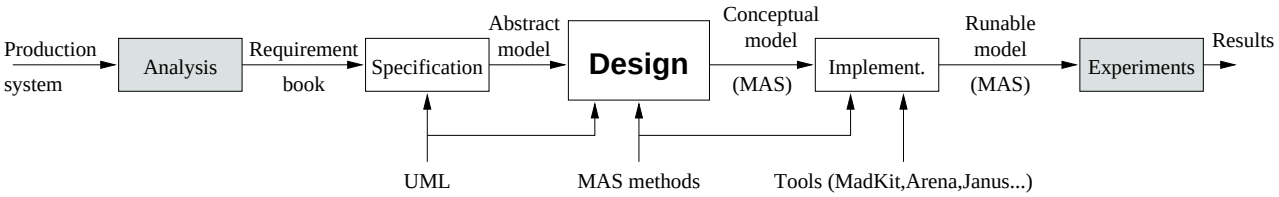


Fig. 1: Life Cycle of the Methodological Approach

propose a life cycle (see figure 1) for our methodological approach that defines the different stages and the different provided models after each of them [10, 11]. The key point of $\mathcal{M}_{\mathcal{A}}\mathcal{MA}\text{-}\mathcal{S}_{\text{DIS}}$ is to use multi-agent models to support the simulation of distributed industrial systems. Indeed, the multi-agent systems are naturally distributed and permits to simulate complex decision-making processes [14, 17, 8].

The four following subsections describes the major stages of the life cycle of $\mathcal{M}_{\mathcal{A}}\mathcal{MA}\text{-}\mathcal{S}_{\text{DIS}}$.

2.1. Specification

The *Specification* within a simulation context is the translation of the information from the requirement book to an abstract model. The formalisms and the methods strongly depend on the methodology: for example, ASCI uses the directed diagrams Entity-Association-Attribute (EAA) and the object diagrams. If we take a look on a methodology resulting from the modeling of enterprise like CIMOSA [1], the *Specification* partly corresponds to the modeling level of the "design specification". In addition, the concepts used to model an industrial system on this level are also specific to the used methodological approach.

According to J.L. Le Moigne an industrial system is the composition of the decisional, physical and informational subsystems. Each of them compose the abstract model in $\mathcal{M}_{\mathcal{A}}\mathcal{MA}\text{-}\mathcal{S}_{\text{DIS}}$.

The decisional subsystem encompasses the whole of the organisational structures and decision-making processes. The UML metamodel of [10] defines a language that permits to describe the interactions and the relationship between the *decision-making centers*. The centers can take operational, tactical and strategical decisions. The relationships between centers can be hierarchically controlled or cooperative. This point of view is issued from the works on the organizational structures in industrial systems [2] and in multi-agent systems [15]. Each decision-making center includes at least one *behavioral model*, which will be defined during the design stage.

The physical subsystem is not modeled in this paper. From our previous works the physical subsystem is composed of the infrastructure elements such as production cells, transportation systems... Many tools already simulate the behavior of this subsystem (ARENA®, SIMPLE++®, queue model based tools...). Thus it can be considered as a black-box on which the rest of the simulation model could be plugged.

The information subsystem is the interaction mean between the decisional subsystem and the physical subsystem. It is composed of information such as the product's nomenclature, the operational sequences, the time and delay specifications... These objects are refered and used by the decisional entities and the physical components during the simulation. In the conceptual model the information subsystem is a subset of the agent's environment.

2.2. Design

The phase of *Design* consists in the building of a data-processing model (also called conceptual) which describes in a more precise way the model resulting from the *Specification*. The objective is not to finalize the execution model. But, it is to build a model independent of any tool and any software platform. Thus, only the decisions that not direct the low-level choices are carried out. In general, this phase is reduced to nothing (eg. Conical Methodology), or is merged with the phase of implementation (eg. ASCI Method).

Within the framework of $\mathcal{M}_{\mathcal{A}}\mathcal{MA}\text{-}\mathcal{S}_{\text{DIS}}$, we consider that the phase of *Design* is significant. Indeed, we decided to not introduce the multi-agent systems (MAS) during the phase of *Specification* because we think that the choices of implementation must be hidden as much as possible to the user of the methodology. We consider that the *Design* is the phase during whish a MAS model could be introduced. The use of a MAS is not in contradiction with the definition of the design resulting from the software engeneering. Indeed, the selected MAS modeling approach (the approach "Vowels" [5], AGR [8] or RIO [16, 21]) is independent of any software implementation. Consequently, we can create a multi-agent model respecting this approach without choosing a platform of execution (MADKIT, JANUS...). Unfortunatly, [10, 11] does not propose a formal model of multi-agent architecture dedicated to our class of problems. Our contribution in this paper tends to provide such a model based on a formal and organizational approach: RIO and OZS.

2.3. Implementation

The *Implementation* is the translation of the model resulting from the *Design* on a particular software platform.

The following sections especially focus on the stage *Design*. The other stages could be studied in [10, 11].

3. DESIGN STAGE: AN ORGANIZATIONAL-BASED APPROACH TO BUILD THE MULTI-AGENT SIMULATION MODEL OF THE DECISIONAL SUBSYSTEM

As mentioned in the previous section, the *Design* permits to build a multi-agent model which is representing the abstract model of a distributed industrial system. In [10, 11] we propose several translation rules to automate the build of the multi-agent model from the abstract model. These rules strongly depend on the proposed agent-oriented ar-

Published in the Special Session "Multi-Agent Systems for Supply Chain Management" of the IEEE International Conference on Service Systems and Service Management (IC-SSSM06), pp. 859–865. Troyes, France.

chitecture. But in our point of view this approach has several problems:

- each entity from the abstract model is directly mapped to a dedicated agent. It means that an agent cannot represent two different abstract entities at the same time;
- the proposed architecture includes different levels of abstraction for the agents: the problem-specific agents and the previously platform-specific agents. The agent-oriented approach does not permits to clearly differenciate these two layers;
- the proposed rules are not based on a formal definition of the agents nor their roles.

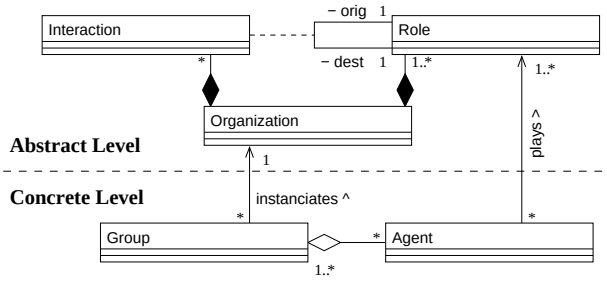


Fig. 2: RIO Meta-Model inspirated by [20]

From the multi-agent research community, several works provide organization-based approaches for the multi-agent modeling. We have selected the RIO methodology [20, 21, 21, 16] because it is based on the concepts of role, interaction and organization which are very closed to the definitions of the abstract entities. The figure 2 illustrates the meta-model of RIO. It details the various relationships between those three main concepts. The role is the abstraction of a behavior or/and a status in an organization. An interaction is a link between two roles such that an action in the first role produces a reaction on the second. And an organization is defined by a set of roles, their interactions and a common context. We have selected this approach since it enables formal specifications, animations and proofs based on the OZS formalism [13]. This method was already successfully used for the holonic modeling of industrial systems and their transportation systems [21].

According to the RIO methodology, the decisional subsystem of an industrial system could be modeled as a set of organizations. The figure 3 illustrates three of the RIO models composing our example of multi-agent model. The agent's roles are inside plain-line boxes and the environmental roles are depicted as white dashed boxes. Interacting roles are linked using plain lines associated to a box describing the name of a interaction and its direction. The environmental roles defines the resources required by the decisional subsystem's entities to tackle their behaviors. The agent's roles are directly mapped from the entities defined inside the abstract model.

In this paper, we are essentially focused on the trading organization and detail the generic interactions between the agent's roles and the environmental roles. The formal specification of these interactions and the trader role are successively described in the two following sections.

4. DEFINITION OF THE ABSTRACT BASE OBJECTS WITH OZS

The roles of the entities composing a virtual enterprise are described in a formal way using the multi-formalisms language OZS which combines the Object-Z and Statecharts notations [13]. The statecharts are used to describe the behavior of the roles, and specify their possible states and how events may change these states. The Object-Z schemas describes operations called by the statechart. Both compose an OZS class which describes an agent's role.

This section describe the generic behaviors (or roles) of an agent interacting with its environment. These definitions are refined and extended to create several problem-specific behaviors. Our vision of the environment is inspired by the Influence/Reaction model, dealing with simultaneous actions in MAS and stipulating that an agent do not directly modify the environment state [7, 9]. The OZS inheritance procedure implies the specialization of the statecharts describing the role behaviors and the overloading of several operations.

Several words on the OZS formalism As mentionned above, the OZS formalism is based on Object-Z and Statecharts. We assumes that you are familiar with the Z notation [22] which is a mathematical typed notation based on sets and functions. In this paper only the Z's key points mandatory to undestand our models are explained. On OZS an object is described inside a box such as the example in the following page. This box is labelled by the object's name (eg. *EnvironmenObject*) and is divided into several sub-boxes which represent the static definitions, the attributes, the operations and the dynamical behavior attached to the corresponding object. Each sub-box is composed of the variable's definitions on the top and by the invariant constraints on the bottom. In an operation box, the variable definition corresponds to the declaration of the interface of this operation. The following operators could be used: $\Xi(v)$ and $\Delta(v)$ respectively indicate if the object's attribute v will be accessed in read-only or read-write mode by the operation; $v?$ and $v!$ respectively denote that v is an input or an output parameter of the operation. If $\Delta(v)$ was specified, the notation v and v' in the bottom part of the operation represent the value of v respectively before and after the call to the operation.

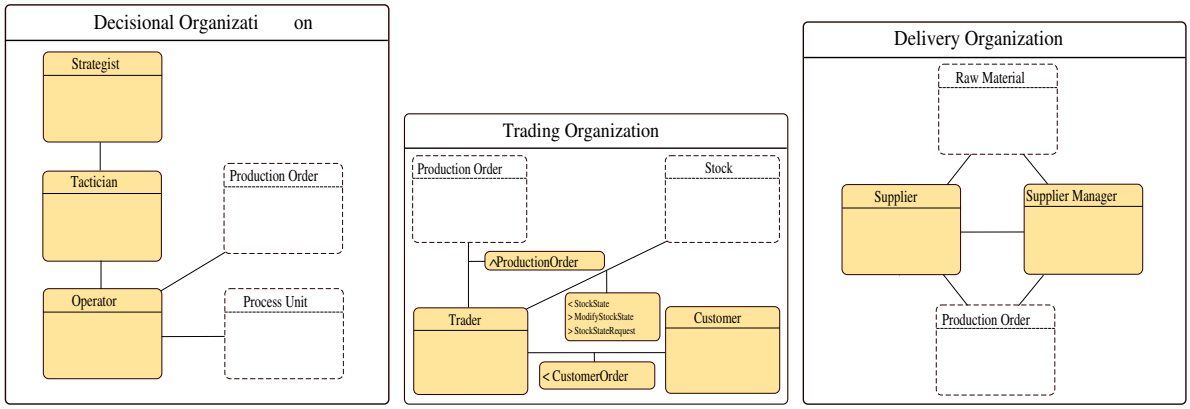
The dynamical behavior is described with a statechart in which each transition was labelled by the name of the operation to execute when passing from a state to another one. The transition could be passed only if a specified condition was reached (this condition is expressed between brackets) or when a message was received by the agent playing this role.

Preliminary Definitions Before describing the OZS schemas, the following notations are introduced :

- Σ is the set of all possible states of the environment,
- *CHAR* is the set of all possible characters,
- *ID* $::=$ seq *CHAR* is the set of possible identifiers for objects and agents,
- *TIME* $::= \mathbb{N}^+$ defines a date of simulation,
- *INFLUENCE* is the set of all possible influences,
- *currentTime* is the current date of the simulation.

As illustrated by the figure 3, some roles are defined inside the multi-agent environment: process unit, raw

Published in the Special Session "Multi-Agent Systems for Supply Chain Management" of the IEEE International Conference on Service Systems and Service Management (IC-SSSM06), pp. 859–865. Troyes, France.



(a) Decisional Organization (b) Trading Organization (c) Delivery Organization

Fig. 3: Several RIO models of industrial system's organizations

material... Because this paper aims to focus on the agent's roles instead of the environmental roles, your environment model is very simplified but inspired by the influence-reaction model [7, 9] and the formal definitions proposed by [23].

Environment Definition The environment is composed by objects defined by the following OZS class. First an environmental object is defined by its state. The definition of a object's state is not significant at this point. But it is dependent of the date of the simulation. The *state* function permits to retain an history of the states along the simulation life. The *transition* function describes all the possible transitions available from one state to another. This function simply returns the successor states of the state in which was the environmental object at the specified time. To understand and formalize the agent's roles the operations *getState* and *influence* are defined. The first, *getState*, permits to obtain the state of objects according to a given date of simulation. The second operation *influence* permits to influence the object with a desired action. Because of our environment simplification the environmental **dynamics** introduced by [23] are not modeled in our system.

From this point it is possible to define the environment as a set of environmental objects. The following OZS specification defines one of these objects. Thus, the state of the environment is the union of the states of all the objects composing it. Because of the static modelling of the environment, no dynamical behavior is included inside the model. We already introduced the notion of influence to prepare future works specifically dealing with environmental dynamics and the modeling of the environment as a MAS.

An influence is defined by all desired states for a set of influenced objects :

$$INFLUENCE \mathrel{\widehat=} [EnvironmentObject \pfun \Sigma]$$

EnvironmentObject

$$transition : \Sigma \pfun 2^{\Sigma}$$

$$1 \upharpoonright transition \neq \varnothing \land 2 \upharpoonright transition \neq \varnothing$$

$$id : ID$$
$$currentState : \Sigma$$
$$state : TIME \pfun \Sigma$$

$$\forall o_1 : EnvironmentObject; o_2 : EnvironmentObject \bullet o_1.id = o_2.id \Rightarrow o_1 = o_2$$

getState

$$\Xi(state)$$
$$t? : TIME$$
$$s! : \Sigma$$

$$t? \in \operatorname{dom} state \Rightarrow s! = state(t?) \land$$
$$t? \notin \operatorname{dom} state \Rightarrow s! = state(max(\operatorname{dom} state))$$

influence

$$\Xi(transition)$$
$$\Delta(state, currentState)$$
$$t? : TIME$$
$$s? : \Sigma$$

$$s? \in \operatorname{ran} transition(currentState)$$
$$state' = state \cup \{t? \mapsto s?\}$$
$$currentState' = s?$$

The following OZS class defines the environment. We consider as in the RIO meta-model, the interaction as a link between two roles such that an action in the first role produces a reaction on the second. So in OZS, a whole interaction between two roles is modeled using two operations, one in each role. The action is modeled using an operation, whose the name starts with the prefix *send*, and the reaction with an operation, whose the name starts with the prefix *receive*. The operations *receive* (resp. *send*) is thus called when an interaction message/object was received from (resp. sent to) another role. An example of a such interaction could be found in the RIO organization diagram illustrated by the figure 3(b).

Published in the Special Session "Multi-Agent Systems for Supply Chain Management" of the IEEE International Conference on Service Systems and Service Management (IC-SSSM06), pp. 859–865. Troyes, France.

Environment

$objects : \mathbb{F}\, EnvironmentObject$
$currentInfluences : ID \pfun \mathbb{F}\, INFLUENCE$
$AllInfReceived : \mathbb{B}$
$influenceComposition : \Sigma \times \mathbb{F}\, INFLUENCE \pfun \Sigma$

$\forall i : ID \mid i \in \mathrm{dom}\, currentInfluences \bullet$
$\exists_1 o : EnvironmentObject \mid o \in objects \bullet o.id = i$

INIT

$objects = currentInfluences = \varnothing$
$AllInfReceived = \mathrm{false}$

receiveInfluence

$\Delta(currentInfluences)$
$i? : ID$
$a? : \mathbb{F}_1\, INFLUENCE$

$\exists_1 o : EnvironmentObject \bullet i? = o.id$
$i? \notin \mathrm{dom}\, currentInfluences \Rightarrow$
$currentInfluences' = currentInfluences \cup \{i? \mapsto a?\} \land$
$NewInfArrived'$

checkAllInf

$\Xi(currentInfluences)$
$\Delta(AllInfReceived)$

$(\#currentInfluences = \#AbstractISRole.\mathrm{playing})$
$\Rightarrow AllInfReceived' = \mathrm{true}$

computeReaction

$\Xi(objects, influenceComposition)$
$\Delta(currentInfluences)$

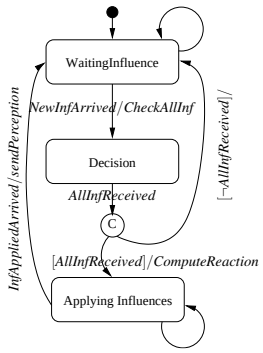
$\forall o : EnvironmentObject \mid$
$o \in objects \land o.id \in \mathrm{dom}\, currentInfluences \bullet$
$o.influence(currentTime,$
$influenceComposition(o.getState(currentTime),$
$currentInfluences(o.id)))$
$currentInfluences' = \varnothing$
$InfAppliedArrived'$

sendPerception

$\Xi objects$
$p! : \mathbb{F}\, \Sigma$

$p! = \{\forall o : EnvironmentObject \mid$
$o \in objects \bullet o.getState(currentTime)\}$

Behaviour



Definition of an abstract agent role To define the OZS classes for the problem-dependent agent's roles, an abstract role specification is proposed to represent a generic role for our application domain. This formal class must then be specialized by the problem-dependent agent's role classes. The abstract role definition is based on a weak sequence of three steps: perception, decision, action. This sequence is modeled using a simple statechart with two states : *Idle* and *Activity*. The activity state will be specialized by each sub-roles. In our model, all agents perceive at the same time, when the environment provide the perception, the operation *receivePerception* is thus called to retrieve the set of percepts. If the agent was in the activity state, it must compute actions based on its knowledge and its percepts. Finally, the agent can emit influences to the environment during the transition between the states *activity* and *idle* and with a called to the operation *sendInfluence*. The class *AbstractISRole* inherits a part of its definition from the RIO class *Role* [16].

AbstractISRole

[*Role*]

$id : ID$
$percepts : \mathbb{F}\, \Sigma$
$actions : \mathbb{F}\, INFLUENCE$

$\forall a_1 : AbstractISRole, a_2 : AbstractISRole \bullet$
$a_1.id = a_2.id \Rightarrow a_1 = a_2$
$\forall f : \mathbb{F}\, \Sigma \mid\in \mathrm{dom}\, filter \bullet filter(f) \subseteq f$

INIT

$percepts = actions = \varnothing$
$filter = \mathrm{id}\, \mathbb{F}\, \Sigma$

receivePerception

$\Xi(filter)$
$\Delta(percepts)$
$p? : \mathbb{F}\, \Sigma$

$\exists_1 f : \mathbb{F}\, \Sigma \pfun \mathbb{F}\, \Sigma \bullet percepts' = f(p?)$
$percepts' \neq \varnothing \Rightarrow PerceptionIn'$

decide

$\Xi(percepts)$
$\Delta(actions)$

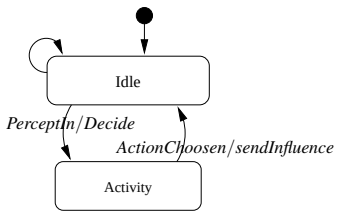
$\exists_1 f : \mathbb{F}\, \Sigma \pfun \mathbb{F}\, INFLUENCE \bullet actions' = f(percepts)$
$ActionChoosen'$

sendInfluence

$\Xi(actions)$
$a! : \mathbb{F}\, INFLUENCE$

$a! = actions$

Behaviour



5. AN EXAMPLE OF THE DECISIONAL SUBSYSTEM MODEL

Let takes the trading organization (see figure 3) role to illustrate our formal model of a distributed industrial system. The trader is the role interacting with the customers of the system. It collects the orders of final products and generates production orders each time the product's

Published in the Special Session "Multi-Agent Systems for Supply Chain Management" of the IEEE International Conference on Service Systems and Service Management (IC-SSSM06), pp. 859–865. Troyes, France.

stocks have not enough elements.

Preliminary Considerations In the following OZS classes, *Product* defines all the possible products in the industrial system and *Customer* is the role played by agents in the trading organization.

CustomerOrder is an interactional object representing the customer's orders. It contains the count of each final products requested by a customer. Interactional objects are data structures exchanged between agents during an interaction [12].

CustomerOrder

$id : ID$
$customer : ID$
$order : Product \pfun \mathbb{N}_1^+$

$order \neq \varnothing \land \exists_1 c : Customer \bullet c.id = customer$
$\forall co_1 : CustomerOrder; co_2 : CustomerOrder \bullet$
$co_1.id = co_2.id \Rightarrow co_1 = co_2$

From the customer order, the trader checks the state of the product stock (representing by the interactional objects *StockStateRequest* and *StockState* defined below) to obtain the count of products not previously affected to a customer. The stock state is based on the definition of a product affectation which indicates the count of free products and the count of products affected to customers.

ProductAffectation

$freeQuantity : \mathbb{N}^+$
$affectedQuantity : ID \pfun \mathbb{N}_1^+$

$\forall i : ID \mid i \in \mathrm{dom}\, affectedQuantity \bullet$
$\exists_1 c : Customer \bullet c.id = i$

The request for the state of the product's stocks could be defined as:

StockStateRequest

$orderId : ID$
$orderProducts : \mathbb{F}\, Product$

$\forall r_1 : StockStateRequest; r_2 : StockStateRequest \bullet$
$r_1.id = r_2.id \Rightarrow r_1 = r_2$

The state of the product's stocks could be defined as:

$$StockState \mathrel{\widehat=} [Product \pinj ProductAffectation].$$

In the case the stock does not containing enough free products to archive the customer order, the trader must sebd a production order to the rest of the industrial system. A production order was used by the operators in the decisional subsystem to start manufacturing. The trader gives the count of requested product, but because he does not control the manufacturing time, the product stock is not directly updated by the trader. This point is the result of the physical system behavior outside the trading organization. In the same way supplying the final products to the customer is archived in the delivery organization. The production order is modeled in OZS as:

$$ProductionOrder \mathrel{\widehat=} [Product \pinj ProductAffectation]$$

Role Definition It is now possible to define the trader behavior inside an OZS class. According to the RIO method, the roles can be taken by agents only if they archive an obtain condition, and must leave the role when another condition was true. In the specifical case of our trader model, the obtain condition was always true and the leave condition avoids the agents to leave the role until all the costumer orders was archeived.

Trader

[*AbstractISRole*]

$orders : \mathrm{iseq}\, CustomerOrder$
$cOrder : CustomerOrder$
$stockStates : ID \pfun StockState$
$enoughStock : \mathbb{B}$
$cModif : Product \pfun \mathbb{N}_1^+$
$cNeed : Product \pfun \mathbb{N}_1^+$

$obtainCondition = \mathrm{true}$
$leaveCondition = (orders = \langle\rangle \land$
$behavior.states.\,\mathrm{active} = behavior.states.Idle)$

INIT

$orders = \langle\rangle$
$cModif = cNeed = stockStates = \varnothing$
$enoughStock = false$

receiveCustomerOrder

$\Delta(orders)$
$c? : CustomerOrder$

$orders' = orders \frown \langle c? \rangle$
$orders' \neq \varnothing \Rightarrow CustomerOrderIn'$

sendStockStateRequest

$\Delta(cOrder, orders)$
$r! : StockStateRequest$

$cOrder' = head\, orders \land orders' = tail\, orders$
$r!.orderId = cOrder'.id$
$r!.orderProduct = \mathrm{dom}(cOrder'.order)$

receiveStockState

$\Delta(stockStates)$
$i? : ID$
$s? : StockState$

$stockStates' = stockStates \cup \{i? \mapsto s?\}$
$StockStateIn'$

checkStock

$\Xi(cOrder)$
$\Delta(stockStates, cOrder,$
$cModif, cNeed, enoughStock)$

---

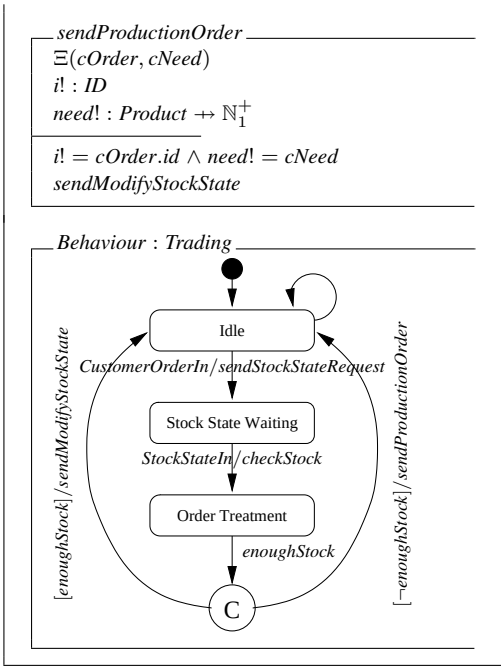
let $instock == stockStates(cOrder.id),$
$moves == \{\forall p \in \mathrm{dom}\, cOrder.order \bullet$
let $free == instock(p).freeQuantity,$
$need == cOrder.order(p), ok == (free \geq need) \bullet$
$(ok \Rightarrow p \mapsto need) \lor (\neg ok \Rightarrow p \mapsto free)\} \bullet$
$cModif' = moves \land cNeed' = \{\forall p \in \mathrm{dom}\, moves \bullet$
$p \mapsto (max\{0, moves(p) - instock(p)\})\} \land$
$enoughStock' = (cNeed \neq cNeed') \land$
$stockStates' = cOrder.id \ndres stockStates$

sendModifyStockState

$\Xi(cOrder, orders, cModif)$
$i! : ID$
$modif! : Product \pfun \mathbb{N}_1^+$

$i! = cOrder.id \land modif! = cModif$
$orders \neq \langle\rangle \Rightarrow CustomerOrderIn'$

Published in the Special Session "Multi-Agent Systems for Supply Chain Management" of the IEEE International Conference on Service Systems and Service Management (IC-SSSM06), pp. 859–865. Troyes, France.



All the other roles defined in the model of the industrial system could be expressed in the same formal way as the *Trader* class.

6. CONCLUSION

In this paper we propose a formal model for multi-agent systems dedicated to industrial systems and in particular to distributed manufacturing systems. This model is used in the stage *Design* of our methodological approach $\mathcal{MAMA}$–$\mathcal{S}_{\text{DIS}}$. Based on the formal method RIO and its foundation language OZS, our approach limits the semantic divergences between the abstract and the conceptual models. In addition, it is an illustration of the capability of RIO and OZS to express manufacturing system models. Indeed the functional approach used in the methodology's stage *Specification* could be directly matched to the role-based approach of RIO.

This work is part of larger effort to define a well-founded framework for modelling and simulation of manufacturing and supply chain systems. Future research will deepen the formal specification of the physicial and informational subsystems, and on the formal translation rules between each models. We also work on multi-scale models based on holonic multi-agents systems. These works will be applied and used inside our methodological approach to archived the simulation of large-scale industrial systems.

REFERENCES

[1] AMICE. *CIMOSA: CIM Open System Architecture, 2nd revised and extended edition*. Springer-Verlag, Berlin, 1993.

[2] C. Berchet. *Modélisation pour la simulation d'un système d'aide au pilotage industriel*. PhD thesis, Université de Savoie, Dec. 2000.

[3] R. Burkhart. The swarm multi-agent simulation system. In *OOPSLA Workshop on "The Object Engine"*, 1994.

[4] I. Castillo and J. S. Smith. Formal modeling methodologies for control of manufacturing cells: Survey and comparison. *Journal of Manufacturing Systems*, 2002.

[5] Y. Demazeau. Steps towards multi-agent oriented programming. In *1st International Workshop on Multi-Agent Systems (IWMAS)*, Boston, 1997.

[6] T. Duval, S. Morvan, P. Reignier, F. Harrouet, and J. Tisseau. ARéVi : Une boîte à outils 3d pour des applications coopératives. *"La coopération"*, 9(2):239–250, June 1997.

[7] M. Fabien. Le modèle influence/réaction pour la simulation multi-agents. In *MFI*, Toulouse, France, May 2001.

[8] J. Ferber. *Multi-agent Systems: Introduction to Distributed Artificial Intelligence*. Addison Wesley, Feb. 1999.

[9] J. Ferber and J. Müller. Influences and reactions : a model of situated multi-agent systems. In *ICMAS*, pages 72–79, 1996.

[10] S. Galland, F. Grimaud, P. Beaune, and J.-P. Campagne. Mama-s: an introduction to a methodological approach for the simulation of distributed industrial systems. *International Journal of Production Economics*, 85(1):11–31, 2003.

[11] S. Galland, F. Grimaud, P. Beaune, and J.-P. Campagne. Simulation of distributed industrial systems - a multi-agent methodological approach. In A. Dolgui, J. Soldek, and O. Zaikin, editors, *Supply Chain Optimisation: Product/Process Design, Facility Location and Flow Control*, volume 94. Springer, 2005.

[12] A. Gouaich and F. Michel. MIC*: a deployment environment for autonomous agents. *LNAI*, 3374:109–126, July 2004. Revised paper, Environments for Multi-Agent Systems, First International Wokshop, E4MAS 2004.

[13] P. Gruer, V. Hilaire, A. Koukam, and P. Rovarini. Heterogeneous formal specification based on object-z and statecharts: semantics and verification. *Journal of Systems and Software*, 70(1-2):95–105, 2004.

[14] Z. Guessoum. *Un Environnement Opérationnel de Conception et de Réalisation de Systèmes Multi-Agents*. PhD thesis, Université Pierre et Marie Curie, LAFORIA-IBP, May 1996.

[15] M. Hannoun, O. Boissier, J. S. Sichman, and C. Sayettat. MOISE : An organizational model for multi-agent systems. In M. Monard and J. Sichman, editors, *Advances in Artificial Intelligence*, volume 1952 of *Lecture Notes in Artificial Intelligence*, pages 156–165, Brazil, Nov. 2000. Springer.

[16] V. Hilaire, A. Koukam, P. Gruer, and J.-P. Müller. Formal specification and prototyping of multi-agent systems. In A. Omicini, R. Tolksdorf, and F. Zambonelli, editors, *Engineering Societies in the Agents' World*, number 1972 in Lecture Notes in Artificial Intelligence. Springer Verlag, 2000.

[17] N. Jennings, K. Sycara, and M. Woolridge. A roadmap of agent reseach and development. *Journal of Autonomous Agents and Multi-Agent Systems*, 1(1):7–38, 1998.

[18] A. Matta, C. A. Furia, and M. Rossi. Semi-formal and formal models applied to flexible manufacturing systems. In C. Aykanat, T. Dayar, and İ. Körpeoğlu, editors, *Computer and Information Sciences - ISCIS 2004: 19th International Symposium*, volume 3280 of *Lecture Notes in Computer Science*, page 718. Springer Berlin / Heidelberg, Kemer-Antalya, Turkey, Oct. 2004.

[19] F. Maturana, W. Shen, and D. Norrie. Metamorph: An adaptive agent-based architecture for intelligent manufacturing. *International Journal of Production Research*, 37(10), 1999. 2159-2174.

[20] S. Rodriguez, N. Gaud, V. Hilaire, S. Galland, and A. Koukam. An analysis and design concept for self-organization in holonic multi-agent systems. In *the Fourth International Workshop on Engineering Self-Organizing Applications (ESOA'06)*, May 2006.

[21] S. A. Rodríguez. *From analysis to design of Holonic Multi-Agent Systems: a Framework, methodological guidelines and applications*. PhD thesis, Université de Technologie de Belfort-Montbéliard, 2005.

[22] J. Spivey. *The Z Notation: A Reference Manual*. Number OX1 4EW. Oriel College, Oxford, England, second edition edition, 1992.

[23] D. Weyns and T. Holvoet. A formal model for situated multi-agent systems. In B. Dunin-Keplicz and R. Verbrugge, editors, *Special Issue of Fundamenta Informaticae*, volume 63 of *Formal Approaches for Multi-Agent Systems*. IOS Press, 2004.

[24] J. Wyns. *Reference architecture for Holonic Manufacturing Systems - the key to support evolution and reconfiguration*. PhD thesis, Katholieke Universiteit Leuven, 1999.

Published in the Special Session "Multi-Agent Systems for Supply Chain Management" of the IEEE International Conference on Service Systems and Service Management (IC-SSSM06), pp. 859–865. Troyes, France.