Simulation Coupling Limitations with Respect to Shared Entities Constraints

Khadim Ndiaye[1], Flavien Balbo[1], Jean-Paul Jamont[2] and Michel Occello[2]
[1]*Mines Saint-Etienne, Univ. Lyon, Univ. Jean Monnet, IOGS, CNRS, UMR 5516 LHC, Institut Henri Fayol, F-42023 Saint-Etienne, France*
[2]*Univ. Grenoble Alpes, Grenoble INP, LCIS, F-26000 Valence, France*

Keywords: Simulation Coupling, Interoperability, Multi-agent based Simulation.

Abstract: Simulation coupling is a mean by which already developed tools are reused and run together for the sake of capitalizing on existing endeavours. A main challenge to microscopic simulation coupling is the synchronization of schedulers, which are in charge of ordering internal actions for their respective simulation. To achieve a consistent execution of the overall simulations, simulation coupling must tackle challenges to interoperability and schedulers' synchronization. In the scope of microscopic simulations, functional coupling objectives can be categorized into different levels from coupled simulations that only exchange aggregated information, to a coupling that highlights novel behaviours. Our goal in this paper is to show that the existing coupling solutions fail to implement the problem where the coupling objective is to combine individual behaviors from diverse microscopic simulations, in order to create new ones. This failure is due to the fact that these solutions consider microscopic simulations to be coupled, as whole components with autonomous schedulers instead of a composite set of behaviors. The limitations are shown using the DEVS formalism to describe coupled microscopic simulation under different coupling objectives, with a formalization of constraints induced by shared components.

1 INTRODUCTION

The study of complex systems is a difficult and time consuming task as it involves various and heterogeneous domains of expertise. Microscopic simulation models have become a widely used tool in many disciplines dealing with systems made up of autonomous entities (Grimm and al, 2006). From domain specific tools, domain experts build independent simulations specific to their interest. The coupling of those simulations is a mean to reuse already made efforts, in order to get a representation of the overall systems. For instance, to study the impact of emergency responses plans and methods on people and infrastructures, a fire propagation simulation, a crowd evacuation simulation and a sensor network simulation are coupled in (Jalali et al., 2011). The simulators used in that study were already developed and each dedicated to a specific task.

A coupling is driven by a set of requirements and constraints so that the end result is meaningful. A same set of simulations can be subject to different coupling objectives. For instance, coupling a macroscopic and microscopic traffic simulations can whether help to calibrate the latter, or allow to study the effects of drivers' behavioural change in traffic flows or congestions.

In this paper, we exhibit a type of coupling problem that most coupling solutions fail to implement. In fact, the coupling objective to this problem constraints scheduling relations on simulations, as a result of shared components between simulations. Examples of shared components are a spatial environment or a stock exchange market, which evolutions depend on the synchronization of actions from the coupled simulations. Unlike *whole component* simulations that exhibit a irreducible behaviour, microscopic simulations have a composite set of distinguishable behaviours. The problem we highlight arises when parts of the simulations are coupled, as opposed to coupling microscopic simulations as *whole components*. There are two issues to consider for the shared components consistency. The first one is the equity between agents activated by different schedulers in accessing the shared component, whilst the second is the consistency of the resulting simulation with regards to

Ndiaye, K., Balbo, F., Jamont, J-P. and Occello, M.
Simulation Coupling Limitations with Respect to Shared Entities Constraints.
DOI: 10.5220/0006859603380346
In *Proceedings of 8th International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH 2018)*, pages 338-346
ISBN: 978-989-758-323-0

coupling objectives. Our focus is targeted on Multi-Agent Based Simulations (MABS) as a microscopic simulation models, in the rest of the paper.

The paper is organised as follows. Coupling solutions are presented along with challenges specific to simulation coupling in Section 2. This section also presents Discrete Time Specified System (DTSS) coupled systems from the DEVS formalism. Section 3 begins by describing MABS as DTSS coupled systems in order to represent the behaviour of coupled microscopic simulations. Then it showcases the limits of current coupling solutions with regard to the shared components challenge by introducing a formalism to describe coupling objectives , before concluding with some insights on our future works.

2 BACKGROUND ON COUPLING SOLUTIONS

In this section, challenges that arise in the scope of simulation coupling are first discussed. Then it presents selected coupling solutions and how they address coupling challenges.

2.1 Simulation Coupling Challenges

Simulation coupling is a mean by which already developed tools are reused and run together for the sake of capitalizing on existing endeavors The issues to coupling simulations that were developed independently, are mainly due to the identification and translation of coupling objectives (Kasputis and Ng, 2000), and to simulation heterogeneity. The heterogeneity may concern the formalism used, the programming language and more generally the domain of interest. We distinguish two family types of challenges: interoperability (data exchange, structure and transformation) and synchronization (time management and scheduling).

(C1) Data Exchange: coupling independently developed simulators requires information exchange between the simulators. An information may consist of data used by the executed models, or may be events shared throughout the simulators for synchronization purposes. Data exchange mechanism should be generic enough to allow the routing of any data structure and should also handle scalability.

(C2) Data Structure and Transformation: simulations from different domains deal with different kind of data, or different representations of the same information. An agreed upon semantic must be shared by the coupled simulations to deal with the exchanged information. The issue is to share information that is meaningful to each simulations.

(C3) Time Management: simulation models may be expressed in different time representation (continuous, discrete event or discrete time) and scales (seconds, days, ...). A consistent coupling solution may ensure a simultaneous execution of the simulations by aligning local clocks according a global time clock: that is the local causality constraint (Fujimoto, 2001).

(C4) Global Scheduling: When simulations are coupled, simulated components may comply to a constraining scheduling policy originating from the coupling objective. So the problem is how to make use of local schedulers to reflect the desired global scheduling.

2.2 Simulation Coupling Solutions

We categorize existing coupling solutions using a classification that reflects the difference between solutions, depending on how they implement the coupling of simulations. The four coupling solution types are described as follows:

Conceptual Model Translation: approaches using model translation to unify the system's conception. The diverse simulation models are wrapped under a common formalism and then composed by means of integration or hierarchical construction. It is the case of the Discrete Event System Specification Bus (DEVS Bus) (Kim et al., 2003), which is a simulation infrastructure developed for heterogeneous simulators to interoperate. Its architecture allows to manage the time advancement (C3) and offers a consistent message passing mechanism between the simulations (C1, C2).

External Medium Use: approaches introducing an external component as a facilitator to the coupling problem. It can also be an interface implemented by the simulation models to sustain their interoperability. The High Level Architecture (HLA) fall under this category. It is a standard composed of an interface specification, the Object Model Template (OMT) and a set of conception rules. The interface specification describes the APIs and services provided by the RunTime Infrastructure (RTI). The RTI, as an external medium, is a middleware that provides time management services (C3) and means for federates to communicate data with each other (C1). The designer decides what and how information are exchanged and used in the federates (C2) by writing a Federate Object Model (FOM).

Common Component Mediation: approaches using a shared component between the coupled models. In order to sustain interoperability, simulation models that some component, can use that component as a proxy to exchange information. The Environment Interface Standard(Behrens et al., 2009) (EIS), is a coupling solution that uses the environment to ensure interoperability among participating agent platforms. It allows to build distributed and heterogeneous Multi-Agent Systems (MAS), by advocating for the separation of agents' execution from their environment. Modeling a trans-platform MAS problem with EIS, means to define independently a shared environment model with controllable entities, by which platforms can indirectly communicate (C1, C2).

Implemented Model Composition: approaches aiming to compose simulators with respect to their implementations. It means that the technical implementation of the simulation model is preserved as opposed to the conceptual model translation approach. The Multi-agent Environment for Complex-SYstem CO-simulation (Camus et al., 2016) (MECSYCO) is an ad hoc solution at simulations' composability. MECSYCO uses the concept of coupling artifact to reify interactions between simulation models. The coupling artifact are in charge of routing and transforming (C1,C2) input and output data from simulations. For coordination between simulation models (C3), a decentralized and conservative time management approach is implemented.

2.3 DTSS Modeling of MABS

2.3.1 Motivation

The theory of modeling and simulation of (Zeigler et al., 2000) introduces a family of formalism (Discrete Event System Specification(DEVS), Discrete Time Specified System (DTSS) and Differential Equation Specified System (DESS)) to model continuous and discrete event systems. It provides rigorous foundations for modeling and simulation, and also introduces concepts of modular and hierarchical model composition. Hence making it a solid groundwork for a MABS coupling formalism.

DEVS's hierarchical modeling focuses on the relations between components. As such, it is suitable to represent multi-agent based simulations resulting from the interactions between agents. This feature explains why many works represent MAS using DEVS formalism (Müller, 2008; Steiniger et al., 2012). However, the features of MABS schedulers are of prime interest when choosing the required formalism. DTSS simulators implement a fixed time step scheduling algorithm that jumps from one simulation step to the next, and computes the next simulated system's state and output values. In this light, DTSS is more in resemblance of how most multi-agent simulation platforms behave.

2.3.2 Coupling DTSS Components

The hierarchy of system specification introduced in (Zeigler et al., 2000) describes the levels at which a system may be known in order to report its complexity, with increasing structural specificity. The level of description used in this hierarchy specifies systems from a behavioral point of view (lowest level), for instance as a black box, to internal structure design (highest level) as in complex systems. The hierarchy specifies two ways DTSS components are coupled depending on the required system description level: multicomponent DTSS (multiDTSS) and discrete time specified network (DTSN).

MultiDTSS: is a system composed by interacting DTSS components (M_d — eq. 2) that influence each other though their state transition functions.

$$multiDTSS = < X, Y, D, \{M_d\}, c > \tag{1}$$

X is the input set of the multiDTSS and Y its output set. D is the index set of the DTSS components, $\{M_d\}$. A multiDTSS time base takes discrete values characterized by the time step constant c. This time step constant is shared by all DTSS components, $\{M_d\}$. For all $d \in D$, M_d is specified by:

$$M_d = < X_d, Y_d, Q_d, I_d, E_d, \delta_d, \lambda_d > \tag{2}$$

X_d and Y_d are respectively the input and output set of M_d. Q_d is the set of all possible states the component can have. δ_d is the state transition function. It processes the current state according to the previous state and the input. λ_d is the output function of the system. $I_d \subseteq D$ is the set of components' indexes on which M_d has read access, meaning their states are variables in M_d's state transition and output functions. $E_d \subseteq D$ is the set of components' indexes on which M_d has write access on, meaning the components on which it can set their state values. Despite the influence relations between components in a multiDTSS, their scheduling is not constrained. A multiDTSS is simulated by iterative scheduling without specific ordering over the components, and by executing their state transition functions.

DTSN: is a higher level of specification where components are coupled through their input and output interfaces rather than interacting directly through their state and state transition functions as in multi-DTSS. A Discrete Time Specified Network is structured as:

$$DTSN_N = < X_N, Y_N, D, \{M_d\}, \{I_d\}, \{Z_d\}, c > \tag{3}$$

N is the index used to designate the DTSN. X_N and Y_N are the external inputs and outputs of the network. c is the time step constant and D the index set of components, $\{M_d\}$. The set of influencers I_d is the index set of components which outputs are coupled to M_d's input through Z_d which is the interface map of M_d. Z_d indicates which of the inputs of a component M_d are associated to which outputs of its influencers in I_d. Components need to have all their inputs available in order to execute. This means that the scheduling is constrained by the influencers. Let us note that I_d may also contain the index of the network DTSN, i.e. $d \in D \cup \{N\}$.

Because of the constraints induced by $\{Z_d\}$, a specific order needs to be observed in the execution of DTSN components. To resolve the concern about input and output interdependencies, the execution of a DTSN uses a *coordinator* which schedules its subordinate components in the right sequence, meaning that only when all components' inputs are available that is the component ready to go (Zeigler et al., 2000).

The choice on the coupling model (multiDTSS or DTSN) is determined by the level of specification needed in the coupling. If we can precisely isolate their states and how those states influence each other (δ_d in eq. 2), the multiDTSS specification can be used. The result is a multiDTSS of multiDTSS and the simulation of such coupling model does not require a specific ordering in how the components are executed. If however this process to capture influences between components turns out insufficient, a DTSN modeling may be used instead. *coordinators* are used to resolve localized scheduling constraints between a component and its influencers. A *root coordinator* is then used to monitor the overall scheduling.

3 SHARED ENTITY CONSTRAINTS

This section highlights the constraints on shared components between coupled microscopic simulations resulting from specific coupling objectives, and how existing coupling solutions lack to address them. Because a MABS is a set of interacting agents, we first describe it using the hierarchy typologies that coupled DTSS provides. Then basic coupling objectives are presented, along with an example for each type of coupling objective. Finally, the constraints from coupling objectives are formalized into relations, in order to measure how they affect the scheduling of coupled MABS.

3.1 MABS as DTSS Coupling

The main components of a MABS architecture are: environment, agents, actions' scheduler, and interactions. The environment is composed by a set of objects and agents with perceivable properties. The scheduler is in charge of activating the agents between time steps. Agents are components with autonomous processes that are capable of acting on the environment and communicating with each other. Agents can have internal properties that are hidden or perceivable ones reachable by others in the environment. Because interactions in the MABS are relevant to coupled DTSS models, only perceivable properties of agents are considered. As such, these terms are defined in the sense of *EASI* (Saunier et al., 2014). The states of an agent, object or message are contextual information that characterize them at any given time. A state of an element d : $s_d = \{(p_i, v_i) : p_i \in P_d, v_i \in Dom_i \in Dom_d\}$, is a set of 2-tuples of properties $P_d = \{p_i\}$ and values from $Dom_d = \{Dom_i\}$ couples (where Dom_i is the domain of p_i). Q_d is the set of all possible states s_d a component can have.

An agent's state transition does not solely depend on its own state, but may also depend on other states it perceives. So the set of agents in a MABS are interacting components, that influence each other through their state. MABS can then be seen as a multiDTSS with agents as DTSS components. As such, we model a MABS as a multiDTSS of agents defined by:

$$mabs_\alpha = < X_\alpha, Y_\alpha, D, \{ag_d\}, c >, \qquad (4)$$

X_α is the set of all perceivable states , i.e. messages, agents' and objects' states in the MABS. Y_α is the overall output of all agents in the MABS. D is the index set of the agents and c is the time step for all agents. ag_d is an agent represented as a DTSS component and its parameters are defined similarly to the specification of M_d (eq. 2):

$$ag_d = < X_d, Y_d, Q_d, I_d, E_d, \delta_d, \lambda_d > \qquad (5)$$

X_d is the set of perceivable states by the agent. Y_d is the set of messages and agent's actions on the environment. Q_d is the set of states the agent can have.

However, the term *influence* used in coupled DTSS, differs from its general meaning in MABS. In fact, agent's autonomy does not allow for others to directly set its states. So the only agent, an agent ag_d can set its state, is itself : $E_d = \{d\}$ for all $d \in D$. Hence the state transition function bears the following form:

$$\delta_d : Q_d \times X_d \rightarrow Q_d$$

$Q_i \subset X_d \subset X_\alpha$, $\forall i \in I_d$. It means that influencing states are passed to agent through its input.

If all components are DTSS, and there is no delayless cycles and all the components have the same time step, then the resulting structure of a coupled DTSS is proved to be equivalent to a DTSS (Zeigler et al., 2000). In other words, DTSS is closed under the coupling operation. Hence the MABS represented as a multiDTSS can also be considered as a DTSS, and be coupled with other MABS in the same way.

Given the nature of agents' interactions, a single MABS is specified as a multiDTSS. But when coupling MABS themselves, new relations can arise between agents from different simulations, depending on the coupling objective.

3.2 Coupling Objective of MABS

A coupling objective is the desired result achieved by observing a set of requirements and constraints. Let us consider the coupling of two independent MABS, $mabs_\alpha$ and $mabs_\beta$ along with their set of agents $\{ag_i^\alpha\}$ and $\{ag_i^\beta\}$, in order to obtain a new simulation noted $mabs_\gamma$. We highlight three types of coupling objectives a designer have, with increasing level of constraints in the relations between agents:

Conjoint Execution of Simulations (Case 1): the designer jointly executes MABS where agents throughout simulations do not influence each other directly. Instead, information at the level of the simulations are shared. It is for instance the case of the functional coupling objective of *studying the effect of weather on traffic"* with $mabs_\alpha$ and $mabs_\beta$ representing respectively the weather and traffic MABS. The simulated traffic flow is used by the weather simulation to compute air pollution and weather conditions will have effects on the traffic simulation. We use the coupling operator $\perp$ to note this kind of coupling: $mabs_\gamma = mabs_\alpha \perp mabs_\beta$.

Merging Agents Into Same Simulation (Case 2): agents in different simulations need to be aware of each other and influence mutually as if they are in the same simulation. For instance, with the objective to *"simulate different vehicle models in the same traffic simulation"*, different models of vehicle (car and truck models for instance) from different simulations need to share a same road network. The respective positions of vehicles in one simulation, will influence the behavior of vehicles in the other. Using the operator $\cup$, we note: $mabs_\gamma = mabs_\alpha \cup mabs_\beta$

Building New Behaviors (Case 3): the designer builds up new behaviors by composing existing agent behaviors in different simulations. In addition to being aware of each other existence, agents through the simulations need to have new interactions in order to reflect the new behaviors to be built up. In the example of the functional coupling objective: *combine a traffic simulation with a convoy simulation to give the latter a realistic behavior*, the concept of vehicle and convoy are already modeled but the behavior of adapting the convoy's movement to the vehicles', exists in neither. We use the notation: $mabs_\gamma = mabs_\alpha \oplus mabs_\beta$

In section 3.3, we describe the relations induced by the existence of shared components into what we call *shared entities*, to evaluate the constraints they impose on the implementation.

3.3 Shared Entity Definition

We define a *shared entity* as any element that is at least represented in two simulations to be coupled, in accordance with a coupling objective. In the case of coupling a *convoy* with *vehicles*, the concept *position of a vehicle* is shared by the *vehicle* and the *convoy* it belongs to, because the convoy's *position* is obtained by mean of aggregation of its constituting vehicles' *positions*. The sharing of such concept implies that the modification of the *position* of a vehicle should directly impact the *position* of its corresponding convoy, whilst the related vehicle and convoy are in different simulations When composition agents' behavior in order to create new ones, agents in different simulations, which were formerly independent, will now influence each other by their perceivable states.

These common properties may belong to specifically simulated objects in the environments or be subpart of agent properties. In this context, we distinguish two types of shared entities: real and abstract ones. Real shared entities are objects in the environment that are explicitly simulated components in at least one simulation, while abstract ones are concepts assimilated to a subset of agent properties. For instance, the *road network* is a real shared entity and the *position* of a vehicle is a abstract shared entity because it is a vehicle property and subpart of a convoy properties. It is important to distinguish the two types, as we will see later in this paper, that they do not impose the same constraints.

A coupling objective requires interoperability and synchronization relations between simulations. So does the shared entity as an instance of the coupling relations. In that sense, its definition is separated into two parts: I_e for data exchange and transformation, and S_e for time management and scheduling. A shared entity e between two agents ag_k^α and ag_l^β is defined by:

$e = (I_e, S_e)$
$I_e = (ag_k^\alpha, ag_l^\beta, P_e, (transform_j^i)_{i,j})$
$with: 0 < i < | P_e |,\ j \in \{k,l\}$
$S_e = (ag_k^\alpha, ag_l^\beta, lifeCycle_e)$
$P_e = \{p_e^i\} \subseteq P_k \bigcap P_l$
Q_e^i *state set of property* p_e^i
$transform_k^i = write_l^i \circ read_k^i$, *and*
$transform_l^i = write_k^i \circ read_l^i$
$write_k^i : Q_k \to Q_e^i$, *and* $write_l^i : Q_l \to Q_e^i$
$read_k^i : Q_e^i \to Q_k$, *and* $read_l^i : Q_e^i \to Q_l$
$lifeCycle_e = (scheduling_e, create_e, destroy_e)$
$scheduling_e(rwAccess_e(k), rwAccess_e(l))$: a permutation on $\{k,l\}$

I_e is the interoperability part of the sharing relation and P_e is the set of shared properties associated to the shared entity e. Both real and abstract shared entities model the sharing relation over some property values. The family of functions $(transform_j^i)_{i,j}$ describes how information on the shared entity is adapted to be perceivable by the agents, where the indexes i and j represent the index of the property to be transformed and the index of the agent which uses it. There are two accesses an agent can have on shared entity: read and write accesses. For instance, a transformation function can be the result of a composition of two operations: translation of agents states into shared entity states($write_j^i$), and the translation of shared entity states into agent states($read_j^i$).

S_e controls how the exchange mechanism defined by I_e is performed, by determining specific actions to occur depending on states in Q_e. It represents the synchronization relation where $lifeCycle_{kl}$ directs the creation, destruction and scheduling of the related agents, upon an update of shared entity's state. For instance, a convoy have a read access to a *position* while the corresponding vehicle have a write access on it. So the state of the shared entity *position* needs to be updated by the vehicle before the convoy is executed. $lifeCylcle$'s *create* and *destroy* functions, define actions to take upon creation and destruction events (e.g. if a vehicle leaves its simulation, the convoy needs not to take into account its position anymore and the corresponding shared entity is destroyed).

The sharing of properties implies changes in the structure in each of these MABS components because the set of influencees in each MABS have to change. In the following, we will see how new influences resulting from coupling objectives, affect the network of coupled MABS using the DTSS modeling introduced in 3.1.

3.4 Constraints on Coupled MABS

To couple MABS, the closure of coupling operations on DTSS can be used by coupling the two resulting multiDTSS to obtain another multiDTSS or DTSN specification. We have also defined shared entities from a coupling objective in two parts corresponding to the interoperability and synchronization challenges. Let us consider the three cases corresponding to the different types of coupling objectives, and discuss how their implementation could be achieved.

Case 1 ($\perp$): this is the case where agents in different simulations $mabs_\alpha$ and $mabs_\beta$ do not influence each other individually. So there is no shared entities between agents of MABS to be coupled, but there is still synchronization and interoperability issues to resolve. We can use the multiDTSS or DTSN specification composed by $mabs_\alpha$ and $mabs_\beta$ to achieve the coupling. The choice is driven by the conditions given earlier in Section 2.3.2. For instance, a coupled MABS system as a DTSN is given by :

$$mabs_\alpha \perp mabs_\beta = < X, Y, D, \{mabs_d\}, \{I_d\}, \{Z_d\}, c > \quad (6)$$

The terms are defined similarly to (3). $D = \{\alpha, \beta\}$. Z_d maps out which output of one simulation is fed to the other: $Z_\alpha(Y_\beta) = X_\alpha$ and $Z_\beta(Y_\alpha) = X_\beta$.

Case 2 ($\cup$): agents from different simulations live in the same environment, which they need to share the state. Hence there must be a non-trivial definition of I_e in their shared entities. But there is no constraint on how simulations are executed. They could run in parallel or sequentially for instance. So the $lifeCycle$ function in their S_e may not be necessarily defined. Since shared entities impose no constraints on the scheduling of sharing agents, the same reasoning from case 1 can be applied with changes in the interface maps $\{Z_d\}$. The information exchange mechanism from I_e will be translated in the definition of Z_d of the MABS. So the states defined in the shared entity e are given to the output of the MABS, which are connected thanks to their interface maps. For instance, if $I_e = (ag_k^\alpha, ag_l^\beta, P_e, (transform_j^i)_{i,j})$, and if ag_k^α have write access on e and ag_l^β read access on e, then $mabs_\alpha$ is added to E_k^α and $mabs_\beta$ to I_l^β.

Case 3 ($\oplus$): coupled agents must share state information when their behaviors are combined to create new ones. So the shared entities must enable interoperation (I_e) and maintain consistency of shared properties on both sides (S_e). Consequently, this changes the structure in each of these MABS components because the set of influencees in each MABS have to change. For instance, for any two agents ag_k^α and ag_l^β sharing an entity e, if ag_k^α have write access on properties in e then the corrected influencees set $E_k^{'\alpha}$ in $mabs_\alpha$ becomes: $E_k^{'\alpha} = E_k^{\alpha} \cup \{l\}$. Coupling the MABS as in (eq. 6) will not be possible because of the new influence sets with specific scheduling constraints. In fact, each coupled component is only able to schedule the behavior of its constituting components, but in the $mabs_\alpha$'s structure, there are elements from $mabs_\beta$ existing in $E_k^{'\alpha}$. To avoid having

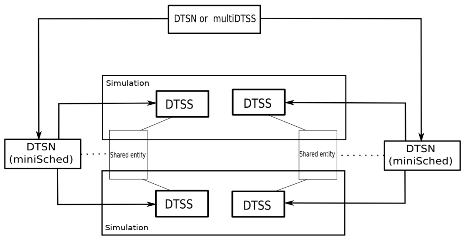

Figure 1: Hierarchical coupling network.

agents scheduled by different MABS in the same multiDTSS, the solution is to rethink the network hierarchy, as presented in Figure 1. Instead of coupling agents from a same simulation into a MABS and then couple the resulting MABS*s*, agents must be grouped with respect to their shared entities. To do so, agents sharing an entity are firstly coupled in a DTSN we call *miniSched*:

$$miniSched_e = < X, Y, D, \{ag_d\}, \{I_d\}, \{Z_d\}, c > \quad (7)$$

With e a shared entity among $\{ag_d\}$ and Z_d constructed from interoperability relation in I_e and scheduling relation in S_e. DTSN is chosen because of the explicit need of scheduling constraints. So the order in which related agents need to be executed is constrained by the *miniSched*. Now since the *miniSched*s are DTSN related to each other by coupling objectives, we can build a DTSN or multiDTSS of *miniSched*s like in (eq. 6):

$$mabs_\alpha \oplus mabs_\beta = < X, Y, D, \{miniSched_d\}, \{I_d\}, \{E_d\}, c > \quad (8)$$

3.5 Coupling Solutions and Shared Entities

The constraints induced by shared entities is that the resulting network is executed by several MABS. In fact, each agents is run in its own simulation by its proper scheduler. We presented 4 categories of coupling solutions to group solutions according to the approach they take in section 2.2. However, since we are interested in the coupling of MABS that are executed independently, solutions that use same conceptual model translation are not discussed. In the following, we will discuss how solutions try to address each of the cases presented in the last section.

External medium use is a way to make simulations interoperate by serving as a message passing and synchronization mechanism. Let us take HLA as an example. HLA's RTI acts as a medium that enables message passing and time coordination. It is analogous to the coordination used in DTSN components. In fact, coupling simulations with HLA comes to execute a DTSN specification obtained by translating the state transition function, output function and interface map into RTI and federate ambassadors. for that reason, HLA is well suited for case 1 and 2. So a HLA federation (ftn) composed of DTSS federates (fdte)is a DTSN where the RTI is the coordinator :

$$ftn = < X, Y, D, \{DTSS\_fdte_d\}, \{I_d\}, \{Z_d\}, c > \quad (9)$$

For HLA to implement the coupling objective in **case 3**, the $DTSS_fdte$s need to be chosen among two possibilities: MABS federates or agent federates. But agents can not be chosen as federates as they are not controlled directly by the RTI but are rather activated by local schedulers. Also MABS can not be considered as federates because the scheduling constraints of the shared entities will not be fulfilled as discussed in Section 3.4.

Then we might consider using HLA to implement the network in eq.8 which respects the scheduling constraints, by using *miniSched*s as $DTSS_fdte$s:

$$ftn = < X, Y, D, \{miniSched\_fdte_d\}, \{I_d\}, \{Z_d\}, c > \quad (10)$$

Since *miniSched*s couple agents in different simulations, they can be seen a federation of *agent* federates:

$$miniSched\_fdte_e = < X_e, Y_e, D, \{ag_d\}, \{I_d\}, \{Z_d\}, c > \quad (11)$$

There is two coupling levels: between components sharing entities and between *miniSched*s. Not only would we need a medium for each *miniSched*, but also a medium to help all these media to interoperate. So the medium should be conceptualized with a hierarchical design. With HLA, the RTI should be able to consider a federation as a federate. (Ahn et al., 2010) and (Kim and Kim, 2006) introduced hierarchy into HLA federations. But with those solutions, the federations' execution are independent. Which is not the case here because agents in different federations are still managed by the same scheduler. Another solution would be to bring the hierarchy into the RTI like coordinators in DTSN abstract simulations. In this case, a federation could be composed of RTI federates. However, that is not the case in the RTI specification, making it impossible for HLA to satisfy constraints in case 3, because of its lack to control the scheduling of agent in federates.

Component mediation solutions are alternatives to resolve the shared entity problem. The common component could manage shared entities through which agents interact. EIS's environment model contains controllable components on which agents act and interact. These controllable components can be used to represent shared entities. Nevertheless, time synchronization on actions is still needed. For instance with EIS, the environment interface does not provide any time synchronization mechanism as it is originally meant for agent platforms and not for simulations in particular. Since the shared entities would live in the environment model, shared entities with synchronization constraints can not be handled. Thus the impossibility of case 3.

Finally, implemented model composition solutions like MECSYCO deals with specific type of simulations, abiding by specific interfaces in their implementations. As such, they only allow to couple simulations as a whole (case 1) and not the coupling of agents across different simulations. Thus not enabling to implement such a network in figure 1.

4 CONCLUSION

This paper presented a challenge to microscopic simulations coupling that is yet to be addressed by existing coupling solutions. This challenge arises when designing the coupling of simulations where components on which synchronization is required, are shared. We defined the concept of shared entities to report of components sharing between coupled simulations. To show the limitations of existing coupling solutions with regards to this challenge, we proceeded to study the constraint resulting from different coupling objective on coupled MABS, with we represented using the DTSS formalism. We showed that the limitations are due to a lack of active scheduling approach by the current solutions.

Our future work is targeted to the proposal of a coupling model that takes into account the shared entity challenge. For genericity purposes, and along side the coupling model, we will work on a formal framework to define coupling objectives in terms of interoperability and scheduling relations between a set of simulations to be coupled. A formalization of coupling objectives will enable to detect inconsistencies before the implementation. It will also enable their structural analysis in order to correspond to a given coupling objective, the suitable coupling solutions.

ACKNOWLEDGMENT

This work was carried out thanks to funds from Rhone-Alpes Region's *ARC 7* program (www.arc7-territoires-mobilites.rhonealpes.fr): *"Innovations, mobilités, territoires et dynamiques urbaines"*.

REFERENCES

Ahn, J. H., Seok, M. G., Sung, C. H., and Kim, T. G. (2010). Hierarchical Federation Composition for Information Hiding in HLA-Based Distributed Simulation. In *2010 IEEE/ACM 14th International Symposium on Distributed Simulation and Real Time Applications*, pages 223–226.

Behrens, T. M., Dix, J., and Hindriks, K. V. (2009). Towards an Environment Interface Standard for Agent-Oriented Programming. Technical Report IfI-09-09, Institut für Informatik, TU Clausthal.

Camus, B., Paris, T., Vaubourg, J., Presse, Y., Bourjot, C., Ciarletta, L., and Chevrier, V. (2016). *MECSYCO: a Multi-agent DEVS Wrapping Platform for the Co-simulation of Complex Systems*. PhD thesis.

Fujimoto, R. M. (2001). Parallel simulation: parallel and distributed simulation systems. In *Proceedings of the 33nd conference on Winter simulation*, pages 147–157. IEEE Computer Society.

Grimm, V. and al (2006). A standard protocol for describing individual-based and agent-based models. *Ecological Modelling*, 198(1):115 – 126.

Jalali, L., Mehrotra, S., and Venkatasubramanian, N. (2011). Interoperability of Multiple Autonomous Simulators in Integrated Simulation Environments. *Spring SIW*, 11.

Kasputis, S. and Ng, H. C. (2000). Model composability: formulating a research thrust: composable simulations. In *Proceedings of the 32nd conference on Winter simulation*, pages 1577–1584. Society for Computer Simulation International.

Kim, J.-H. and Kim, T. G. (2006). Hierarchical HLA: Mapping hierarchical model structure into hierarchical federation. In *International conference on modeling and simulation–methodology, tools, software applications (M&S-MTSA'06). Calgary, Canada*, pages 75–80.

Kim, Y. J., Kim, J. H., and Kim, T. G. (2003). Heterogeneous Simulation Framework Using DEVS BUS. *SIMULATION*, 79(1):3–18.

Müller, J.-P. (2008). Towards a Formal Semantics of Event-Based Multi-agent Simulations. In *MABS*, volume 5269, pages 110–126. Springer.

Saunier, J., Balbo, F., and Pinson, S. (2014). A formal model of communication and context awareness in multiagent systems. *Journal of Logic, Language and Information*, 23(2):219–247.

Steiniger, A., Krüger, F., and Uhrmacher, A. M. (2012). Modeling agents and their environment in Multi-

Level-DEVS. In *Proceedings of the 2012 Winter Simulation Conference (WSC)*, pages 1–12.

Zeigler, B. P., Praehofer, H., and Kim, T. G. (2000). *Theory of Modeling and Simulation, Second Edition*. Academic Press, San Diego, 2 edition edition.