A new transformation approach for complex systems modelling and simulation: application to industrial control system

Article *in* International Journal of Simulation and Process Modelling · January 2021

DOI: 10.1504/IJSPM.2021.113073

CITATIONS
3

READS
49

5 authors, including:

Noureddine Seddari
Université 20 août 1955-Skikda
16 PUBLICATIONS **52** CITATIONS
SEE PROFILE

Sofiane Boukelkoul
Université Constantine 2
11 PUBLICATIONS **32** CITATIONS
SEE PROFILE

Abdelghani Bouras
Ecole Centrale Casablanca
31 PUBLICATIONS **141** CITATIONS
SEE PROFILE

Mohamed Belaoued
Université 20 août 1955-Skikda
25 PUBLICATIONS **117** CITATIONS
SEE PROFILE

A new transformation approach for complex systems modelling and simulation: application to industrial control system

Noureddine Seddari*

LICUS Laboratory,
Department of Computer Science,
University of 20 August 1955-Skikda,
21000 Skikda, Algeria
and
LIRE Laboratory,
Department of Software Technologies and Information Systems,
University of Abdelhamid Mehri-Constantine 2,
25000 Constantine, Algeria
Email: seddarinoureddine@yahoo.fr
*Corresponding author

Sofiane Boukelkoul

LIRE Laboratory,
Department of Software Technologies and Information Systems,
University of Abdelhamid Mehri-Constantine 2,
25000 Constantine, Algeria
Email: sofiane.boukelkoul@univ-constantine2.dz

Abdelghani Bouras

Department of Industrial Engineering,
Alfaisal University,
Riyadh 12714, Saudi Arabia
Email: abouras@alfaisal.com

Mohamed Belaoued and Mohammed Redjimi

LICUS Laboratory,
Department of Computer Science,
University of 20 August 1955-Skikda,
21000 Skikda, Algeria
Email: belaoued.mohamed@gmail.com
Email: medredjimi@gmail.com

Abstract: Due to the increasing of importance of complex systems, the problem of modelling and simulation (M&S) of these systems, their approaches and solutions have been studied extensively these last years. In this paper, we presented a novel approach to modelling and simulation of discrete event dynamic systems (DEDS), based on the combination of discrete event systems specification (DEVS) and agent group role (AGR). In this study, the DEVS atomic models were transformed into agents and the DEVS coupled models into groups of agents. Thus, we provided a set of procedures and functions that allow the systematic transformation from DEVS into AGR models in order to be implemented in multi–agent platforms.The proposed approach was validated by a case study, an application in the water supply used in natural gas liquefaction process.

Keywords: complex systems; discrete event dynamic systems; DEDS; discrete event systems specification; DEVS; modelling and simulation; M&S; agent group role; AGR model.

Reference to this paper should be made as follows: Seddari, N., Boukelkoul, S., Bouras, A., Belaoued, M. and Redjimi, M. (xxxx) 'A new transformation approach for complmex systems modelling and simulation: application to industrial control system', *Int. J. Simulation and Process Modelling*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes: Noureddine Seddari is an Associate Professor of Computer Science at the University of 20 August 1955-Skikda, Algeria. He holds MS and PhD in Computer Science from the same university in 2011 and 2015, respectively. He was an associate teacher and researcher at the Department of IFA, University of Abdelhamid Mehri, Constantine during October 2015 to December 2017. He is currently member of the LICUS Laboratory at the Department of Computer Science of the University of Skikda. His current research interests include agent oriented software engineering, artificial intelligence, complex systems, DEVS formalism, discrete event dynamic systems, multi-formalism modelling and computer simulations.

Sofiane Boukelkoul is a PhD student in Computer Science at the University of Abdelhamid Mehri, Constantine, Algeria. He obtained his Magister in Computer Science from the University of Skikda, Algeria. His research focuses on modelling and simulation of complex discrete event systems, business processes and their verification and validation. He is also interested in formal methods and formalisms transformation.

Abdelghani Bouras holds a PhD in Operations Research from the Joseph Fourier University, Grenoble. He is an engineer and Master of Operations Research. He is currently a Professor of Industrial Engineering at the Al Faisal University, Riyadh. He was formerly a Professor of Industrial Engineering at Ecole Centale Casablanca and King Saud University. He worked as an Associate Professor of Operations Management and Quantitative Methods at the School of Business and Administration at Al Akhawayn University in Ifrane (AUI), and as an Assistant Professor of Production Management at the School of Business and Management at Liège University. He worked in industry as an operations research analyst at Usinor-Arcelor Group (steel industry), as a modeler-analyst for Electrabel-Suez Group (Belgian electricity producer), and finally as a consultant for Pechiney (aluminium industry). He is involved in many researches in mathematical modelling, operations research, operations and supply chain management, and intrusion detection systems.

Mohamed Belaoued completed his Master's and PhD in Computer Science at the University of Skikda in 2011 and 2016, respectively. In 2016, he joined the University of Constantine 1, where he held the position of assistant then Associate Professor for three years. He is currently an Associate Professor at the University of Skikda, Algeria, and a researcher with LICUS Laboratory at the Department of Computer Science of the same university. He is also a member of the Global Foundation for Cyber Studies and Research (GFCYBER). His research interests include malware analysis and detection, intrusion detection, network and IoT security. He also collaborates with researchers in several other disciplines of computer science, particularly computer modelling and simulation, social networks related topics, etc. He also served as a reviewer for international journals, as well as a conference technical committee member, and a workshop organiser.

Mohammed Redjimi is a PhD Full Professor in Computer Science at the University of 20 August 1955-Skikda. He is currently heading the team 'modelling of industrial systems' at the LICUS Laboratory, University of 20 August 1955-Skikda. He is an author of several scientific communications and publications. His current areas of research include wireless sensor networks, software engineering and systems modelling.

1 Introduction

The modelling and simulation (M&S) approach of a problem, can be defined as an activity that makes it possible to study a system using objects called models (Vangheluwe, 2008). As it is not based directly on the system, but on a simplification of it (a model), some precautions must be taken to ensure the relevance of the answers obtained by the model. This is the purpose of validation and verification activities.

When the M&S approach is applied to the study of complex systems, the used models become more complex, which will lead to a set of specific problems the model of a complex system will indeed correspond to a multi-model describing the target system as a system of systems: a set of heterogeneous systems in interaction. At each stage of the M&S approach, we are concerned with heterogeneity in the representations of the system, the formalism used, and the software implementing the multi-model. The M&S of complex systems require the implementation of a set of specific solutions to integrate this heterogeneity into a coherent whole (Camus et al., 2015)

At this stage, we can consider, for certain systems, hybrid approaches in which the methods of the modelling/simulation would be combined. Our work is conducted within this context since we have opted for a hybrid method that combines discrete event systems specification (DEVS) formalism (Zeigler et al., 2018) and agent group role (AGR) model (Gutknecht and Ferber, 2000).

The DEVS formalism is a concept introduced by Zeigler et al. (2000). This formalism allows defining a model mathematically. In other words, DEVS determines the structures and the behaviours (i.e., the dynamic evolution) of the entities constituting the system, and then use atomic and coupled models. Both models' structure and behaviour are encapsulated by the DEVS representation, and the mathematical representations, generated from DEVS models can be formally verified and validated.

For their part, multi-agent systems (MAS), provide a strong theoretical and practical tool in the simulation field. A MAS includes tools for managing the agents (creation, destruction, interactions between agents) and makes the implementation of the system easily realised. The idea developed here is focused basically on the transformation of a model checked and validated in DEVS towards an agent one, to be implemented in multi-agent platforms based on AGR model. Indeed, a series of procedures allowing DEVS-AGR models transition will be provided and integrated into the multi-agent development kit (i.e., MADKIT) platform. The latter, which has been developed by the team of Gutknecht and Ferber (2000), is based on the agent, group and role (i.e., AGR) concepts. Those algorithms are detailed in section 4. Moreover, we present the controlling of a water supply of gas liquefaction process as a case study of this approach.

This paper is organised as follows: Section 2 covers the key concepts related to our contribution namely: the DEVS formalism and MASs. Section 3 presents the existing approaches and tools for M&S. Both proposed approach and transformation algorithms are adduced in Section 4. In Section 5, we depict the implementation of the approach. Section 6 concludes our work and highlights some of its key perspectives.

2 Overview

Simulation modelling provide safe and efficient solutions, understand complex actions that may be impossible to study in real situations. In this section, we will present some key concepts related to our work, namely, the DEVS formalism and MASs.

2.1 DEVS formalism

The DEVS formalism (Zeigler et al., 2016; Van Tendeloo and Vangheluwe, 2018; Vangheluwe, 2001) is an approach of modelling based on the overall theory of the systems. More precisely, it is a modular and a hierarchical formalism for modelling based on the concept of state. This approach was adopted and developed by an international community of researchers (Sarjoughian and Zeigler, 1999; Kofman et al., 2000; Hild, 2000; Anglani et al., 2000; Filippi et al., 2002; Hamri et al., 2006; Mostefaoui and Dahmani, 2019; Wainer, 2016). This work is based on software architecture development, allowing to facilitate, on one hand, the steps of modelling, simulation and validation. On the other hand, the same environment for modelling in order to analyse systems resulting from different fields and to generate automatically the algorithms of simulation. DEVS can be considered as an environment of multi modelling, which allows gathering, in coherent way, other formalism of modelling based themselves on the global theory of the systems. It is, in fact, a formalism adapted to significant number of application (Barros, 1995; Uhrmacher, 2001; Ntaimo and Zeigler, 2004; Troccoli and Wainer, 2003; Seddari et al., 2014; Wainer, 2015; Bazoun et al., 2016; Mokaddem et al., 2018; Dahmani et al., 2020).

2.1.1 DEVS atomic models

A DEVS atomic model is described by the following equation:

$$AtomicDEVS :< X; Y; S; \delta int; \delta ext; ta; \lambda > \quad (1)$$

where

- $X = \{(p, v)|p \in InPorts, v \in R\}$ is the set of input port and value.
- $Y = \{(p, v)|p \in OutPorts, v \in R\}$ is the set of output port and value.
- S: Is a set of states.
- $\delta int : S \to S$: The internal transition function which makes evolving the system from a state to another one in an autonomous way. It depends on the time elapsed in the current state.
- $\delta ext : Q \times X \to S$: The external transition function which occurs when the model receives an external event. It returns the new state of the system based on the current state.
- $Q = \{(s, e)|s \in S, 0 \leq e \leq ta(s)\}$ is the total state set, e is the time elapsed since last transition.
- $\lambda : S \to Y$: Is the output function of the model. It is activated when the elapsed time in a given state is equal to its life.
- $ta(s)$ shows the life of a state 's' of the system. It is the time during which the model will remain in this state if no external event occurs. We find in Figure 1 a symmetric structure of DEVS atomic model.

2.1.2 DEVS coupled models

In equation (2), we describe a DEVS coupled model:

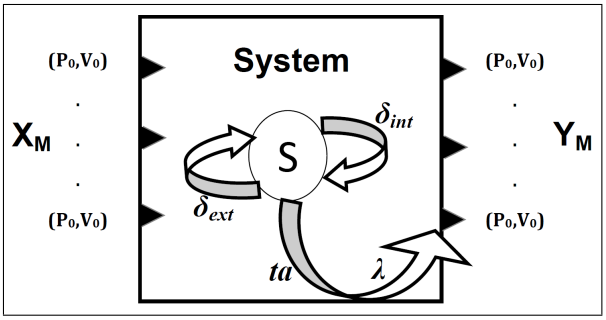
$$CoupledDevs : \\ < X; Y; D; \{Md/d \in D\}; EIC; EOC; IC > \quad (2)$$

where

- X: Is the set of inputs events.
- Y: Is the set of outputs events.

- D: Is the set of names of sub-components.
- $\{Md/d \in D\}$: Is a set of atomic DEVS models or coupled DEVS models.
- EIC: Is a set of external input couplings. They connect the model inputs coupled to those of its own components.
- EOC: Is a set of external output couplings. They connect the outputs of the components to those of coupled.
- IC: Is a set of internal coupling. It connects the outputs of components with entries from other components in the same coupled model. However, no direct feedback loops are allowed, that means that an output port of a component (model) cannot be connected to an input port of the same component.

Figure 1 Atomic DEVS



Source: Seddari et al. (2014)

2.2 Multi-agent systems

2.2.1 Presentation

The multi-agent approach is at the intersection of various domains, such as artificial intelligence (AI), software engineering and distributed computing, and systems. It is a field of study that focuses on collective behaviours that are produced by the interactions between a set of autonomous and flexible entities called agents. Thus, a MAS can be seen as a distributed system composed of a set of agents (Seddari et al., 2017). The representation and the formalisation of an agent can be done in several ways. Ferber and Weiss (1999), Ferber et al. (2003) and Michel et al. (2018) defines MAS system as a system composed of two subsystems. He represents a mono agent system by the couple $< A, W >$, where A is an agent and W is a world as described in equation (3), (4), and Figure 2.

$$A = (P_a, Percept_a, F_a, Infl_a, S_a) \tag{3}$$

where

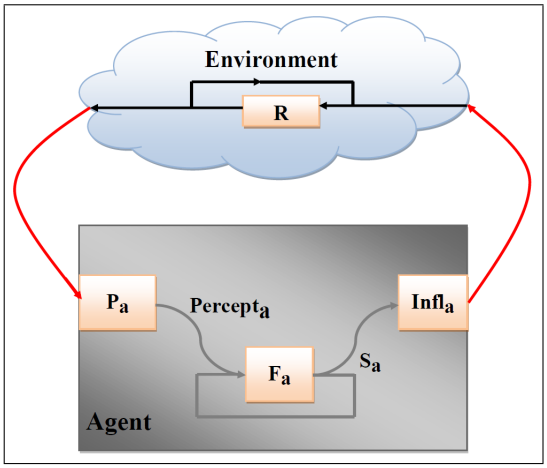
- P_a: The function of perception of the agent. It is the function that allows an agent, being in a state, to discern the set of data that can influence his behaviour.
- $Percept_a$: The set of stimuli and sensations that an agent can receive.
- F_a: The agent's behaviour function that determines its perceptions and its previous state.
- $Infl_a$: The function of action of the agent, that tends to modify the evolution of the world by producing influences from an internal state to the agent.
- S_a: The set of the agent's internal states.

$$W = (E, \Gamma, \Sigma, R) \tag{4}$$

where

- E: The space in which the agent evolves.
- Γ: The influence space an agent produces.
- Σ: The state of the world.
- R: The law of the world evolution.

Figure 2 Influence reaction model (see online version for colours)



3 Related works

There are several approaches in the M&S field that offer interesting solutions to the challenges of handling complex systems.

3.1 DEVS

DEVS formalism has given rise to numerous implementations ranging from simple simulation libraries, to real virtual laboratories to specify a formal multi-model and its simulation (Zeigler et al., 1998; Kim et al., 2009; Quesnel et al., 2009; Zeigler et al., 2013). These last approaches are important for the multi-M&S approach, since they allow to automate the connection between the formal model and the numerical model, and thus to ensure the verification of the model.

In addition, in DEVS/high-level architecture (HLA) (Zeigler et al., 1998), the DEVS wrapping technique was used in conjunction with HLA-based simulation middleware. This approach makes it possible to reuse heterogeneous models already implemented and validated by managing the integration of their formal models and the interoperability of their simulation tools (Zeigler et al., 1999).

The advantages of DEVS are numerous for multi-modelling: it allows to define formal, executable and numerical multi-models, by integrating heterogeneous formalisms and executable models. In addition, the approach can be combined with simulation middleware to reuse models already implemented. However, DEVS has a major drawback when it comes to multi-modelling: it does not provide an explicit solution to bridge the gap between heterogeneous conceptual models (Seck and Honig, 2012).

3.2 Virtual laboratory environment

Virtual laboratory environment (VLE) (Quesnel et al., 2009) is a multi-M&S platform based on DEVS formalism. This platform grants powerful simulation supporting the use of different formalism for models specification. VLE is based on the principle of component-oriented programming, and more precisely application components. Each application component is intended to a specific treatment. This technique makes permits the transformation of a gross monolithic application into a multitude of easily small assembled bricks (Versmisse, 2008). VLE is above all a DEVS engine written in C++ but that is not all, it also allows to take charge of the different stages of the M&S life-cycle. This platform is oriented towards the integration of heterogeneous formalism in one coherent simulation model. It is well adapted for complex models where the coupling of different formalism is required.

3.3 MIMOSA

MIMOSA (Müller, 2007) is a meta platform for M&S of complex systems based on the notions of component (elements), compound (sets of elements), and relationships (between components or compounds) for the structural part, state and events for the dynamic part. In particular, the user can describe the AGR model by introducing on the one hand, populations (compound) of agents (components) sending messages (events) and on the other hand, groups (composed) as sets of roles (components). Population groups relationships specify which agents play which roles in which groups. A discrete event scheduler implements the operational semantics of the entity descriptions and their dynamics described using MIMOSA. It is therefore a fusion of knowledge representation models and dynamic models.

MIMOSA presents an obvious interest for the multi-M&S approach, in the sense that it grants, thanks to Ontologies, to integrate heterogeneous representations of the same complex system. Moreover, since it relies on the DEVS formalism, MIMOSA accepts integrating heterogeneous formalism. Finally, MIMOSA systematises the transition from one stage of the approach to another one, and thus limits the errors that may occur. However, MIMOSA is incompatible with model reuse because its methodology presupposes that thematists unify their representations before their models are formalised and implemented. The approach therefore does not support the management of the interoperability of simulation tools (Müller, 2007).

3.4 SEdit-MADKIT

MADKIT (Gutknecht and Ferber, 2000) is a MAS platform written in JAVA which offers highly successful tools for the management of the agents (creation, scheduling, interactions with messages passing, etc). MADKIT provides a fully agent-oriented environment based on AGR concepts. Each agent may have different roles within different groups. The agents are launched by the MADKIT core, which offers group management and communication services between these groups. Therefor, it is possible to exchange messages directly between agents or to a whole group of agents.

MADKIT contains the multi-formalism graphic editor structure editor (SEdit). This editor can be represented as graphs (unified modelling language-UML, petri nets, interaction diagrams, ...). Formalism are defined externally, in the form of an extensible markup language (XML) description associated with additional classes if one wants to control finely its appearance or its behaviour (for example to allow the simulation of a Petri net). However, this editor does not authorise the modelling of complex systems because its toolbox does not have notions to describe the interactions between models.

3.5 Agents and artefacts for multi-modelling

Agents and artefacts for multi-modelling (Siebert, 2011) aims to build and simulate the heterogeneous multi-model of a complex system from a set of pre-existing models already implemented in simulation software. AA4MM is based on the multi-agent paradigm agents and artefacts (Ricci et al., 2007) to represent a multi-model. Its formalism is based on DEVS, which it uses as an execution model. The particularity of the AA4MM meta-model is the totally decentralised approach to a multi-model, based on DEVS algorithms promoting experimentation as part of the multi-M&S approach (Siebert et al., 2010). However, AA4MM also has several limitations for multi-modelling. First of all, the approach is limited by its field of application: the study of ad-hoc mobile networks. The second limit lies in its capacity to integrate heterogeneous formalism, and in particular cannot integrate event-based formalism. Then, the transition, from a formal multi-model to a numerical multi-model, is not systematic in AA4MM: once the multi-model has been formalised, it must be implemented manually. This phase could include errors raising from model verification as part of an M&S approach (Müller, 2007).

3.6 Summary and contributions

Most of the existing approaches have been axed on natural processes (biological or physical), ecosystems, agro-ecosystems, artificial processes and seldom on industrial processes. Moreover, they provided either a formal to formal models, or MAS to formal models' transformations. In this work, we propose a new combined DEVS-agent-based transformation approach in order to model discrete event dynamic systems (DEDS). This system is a dynamic, asynchronous, in which its states are initiated by its events that cause changes at discrete time points, and are characterised by a complex and hierarchical structure (Ben-Naoum et al., 1995). The main steps of our methodology will be adapted to the industrial control system case. Therefore, our contributions can be summarised in the following points:

- The use of the DEVS formalism for defining the atomic and coupled models based on the abstractions levels of the DEDS components, which is adequate to represent this type of systems (Zeigler et al., 2000).
- Choosing the AGR model for transformation in order to benefit from its advantages, namely: genericity, not being dependent on a field of application, accessibility and opening for DEVS formalism.
- create an algorithmic tool that insures, on the one hand, the passage of DEVS to AGR models. On the other hand, these tools guarantee the interactions between these models, and also guarantee the coordination between the various constituents of the system.

4 The proposed approach

4.1 Formalisation of the AGR model

AGR is an underlying conceptual model of AALAADIN method that focuses on the design and formalisation of MASs in an organisational perspective. AGR is based on three primitive concepts of agents, groups and roles and uses a high level of metaphors for the definition of MASs. However, these metaphors need to be matched within the structures of the DEVS specification in order to define a complete specification.

4.1.1 Formalisation of the agent

An agent is considered as a communicating entity that plays one or several roles within one or several groups. These may overlap and form dynamically. Each agent can belong to several groups. The model places no constraints on the internal architecture of the agents. In an AGR model, agents are not isolated but communicate and interact with other agents through message manipulation. These messages are translated in the DEVS formalism in the form of events, which can contain the information from the message. This specification proposes to use the structure of atomic models for the representation of the agent entity. We formalise this entity in the form:

$$A = (ADEVS|ADEVS = (X; Y; S; \delta int; \delta ext; ta; \lambda)) \quad (5)$$

where

- X: Is the set of stimuli ($Percept_a$) that an agent A can perceive through his mailbox.
- Y: Is the set of ports for output messages to the effectors of an agent A ($effector_a$) for sending messages to other agents through the event sending box.

Perception is specified by the arrival of external events on the ports of atomic DEVS model. It is defined by the events arriving on all external inputs X. The definition of this set specifies the type of messages exchanged by the agents and implies a change of state in the atomic model. Either an agent A formalised by an atomic model ADEVS, the perception Pa of A is defined as all external transition functions such as:

$$P_a = \{\delta ext|\delta ext(Q, (p, v))\} \quad (6)$$

The evaluation of an external transition function implies the change of state of the agent. The internal state of the agent can be formalised as:

$$S_a = \{S|S(phase, \sigma)\} \quad (7)$$

In our specification, the state of the model is characterised by the variable σ, and the phase in which it is located. The phase is decomposed into two parties, the first phase, active, where he receipts a message from the agent. The second phase, passive, takes place upon wait for this message.

Actions are represented as generated events by an agent, i.e., the emission of an external event from the output function output of atomic models. By considering A an agent formalised by an ADEVS, the $Infl$ action of A is defined as follows as well as all output functions such as:

$$Infl_a = \{\lambda|\lambda(S) = (p, v)\} \quad (8)$$

Behaviour, the internal transition function of the agent's atomic model allows to modify the state of models and thus participates in the definition of the agent's behaviour by determining its state. Let F be the behaviour of an agent A, we can formalise Fa as the set of internal transition functions such as:

$$F_a = \{\delta int|\delta int(S)\} \quad (9)$$

The role in AGR model, is an abstract representation of the function, service or identifier of an agent within a group. an agent can play one or several roles. Each role is local to a group. Agents can only establish communication between them through the roles they play. Therefore, control over communications is carried out by the group. The role manipulates the agent's behaviour by adding or subtracting

goals, beliefs, desires or intentions. In our formalisation, a role is associated with an agent's behaviour. However, a change in role is not necessarily attached to a change in the structure of the agent's atomic model. It may simply affect the states of the atomic models. Thus, formalisation to simplify this part remains to be developed since it depends on the systems to be modelled.

4.1.2 Formalisation of the group

In an AGR model, hroup is defined as atomic sets of agent sharing some common characteristic. A group is used for partitioning organisations. An agent may belong to one or several groups. These groups are translated in the DEVS formalism in the form of a coupled CDEVS model. This specification proposes to use the structure of coupled models for the representation of groups. We formalise group in the form:

$$G = (CDEVS|CDEVS = < X; Y; D; M; EIC; EOC; IC >) \quad (10)$$

where

- $Xself$: Is the set of ports for messages arriving respectively from the other models of groups.
- $Yself$: Is the set of ports for sending messages to the other models of agents.
- EIC, EOC, IC define the set of interconnections between agents and groups of agents:

  where

  $$EIC = \{(Xi, Xj)|Xi \subset CDEVSi, Xj \subset Mj, (i,j) \in N\} \quad (11)$$

  with

  a $Mj \in CDEVS$, if the communication is between two groups.

  b $Mj \in ADEVS$, if the communication is between a group and an agent.

  $$EIC = \{(Yi, Yj)|Yi \subset Mi, Yj \subset CDEVS, (i,j) \in N\} \quad (12)$$

  with

  a $Mi \in ADEVS$, if the interconnection is between an agent and a group.

  b $Mi \in CDEVS$, if the interconnection is between two groups.

  $$IC = \{((Xi, Yj), (Yi, Xj))|(Xi, Yi) \subset Mi, (Xj, Yj) \subset Mj, (i,j) \in N\} \quad (13)$$

  with

  a $Mi \in ADEVSMj \in ADEVS$, if the communication is between two agents.

  b $Mi \in ADEVSMj \in CDEVS$, if the communication is between an agent and a group.

  c $Mi \in cDEVSMj \in CDEVS$, if the communication is between two groups.

Once the formalization phase is realised, we proceed to introduce the transformation process. The following algorithms describe the proposed transformation of the DEVS coupled model (CDEVS) to AGR model (Seddari et al., 2019; Seddari, 2015).

4.2 Transformation algorithms

Algorithm 1 allows the passage of DEVS formalism to AGR model. The defined function CDEVS_G (CDEVS) creates the groups Gi (see Algorithm 2, line 2). Then, the procedures CDEVS_EIC, CDEVS_IC and CDEVS_EOC define the different interconnections between the agents AGA (corresponding to the atomic ADEVS model created by ADEVS_AGA (ADEVS) function), AGC_R (representing the coupled model CDEVS input ports), and AGC_E (representing the coupled model CDEVS output ports) on the one hand, and the groups Gi according to coupling type CDEVS (see Algorithm 1, lines 3, 4 and 5), on the other hand.

Algorithm 1 Transformation CDEVS_AGR

Input: CDEVS
Output: AGR
1: **for** all $CDEVS_i$ **do**
2: Call CDEVS_G ($CDEVS_i$) // Agents and groups creation
3: Call CDEVS_EIC ($CDEVS_i$) // Definition of interconnections for the $CDEVS_i.EIC$
4: Call CDEVS_IC ($CDEVS_i$) // Definition of interconnections corresponding to $CDEVS_i.IC$
5: Call CDEVS_EOC ($CDEVS_i$) // Definition of interconnections corresponding to $CDEVS_i.EOC$
6: **end for**

The CDEVS_G ($CDEVS_i$) function: The group Gi which includes several agents is created by this function, for each coupled model $CDEVS_i$ (see Algorithm 2, line 2). So, it adds both receiver coordinator agent (AGC_i_R), and the transmitter coordinator agent (AGC_i_E) (see Algorithm 2, line 8).

Algorithm 2 CDEVS_G transformation function

1: **function** CDEVS_G (CDEVS: CoupledDEVS)
2: Create a group G_i // corresponding to $CDEVS_i$
3: **for** all $ADEVS_j$ **do**
4: call ADEVS_AGA ($ADEVS_j$) // Creation of agents AGA_j
5: add AGA_j within G_i // Regrouping of agents
6: **end for**
7: call CDEVS_AGC ($CDEVS_i$)
8: add AGC_i_E and AGC_i_R within G_i
9: **return** G
10: **end function**

Algorithm 3 ADEVS_AGA transformation function

1: **function** ADEVS_AGA (ADEVS: AtomicDEVS)
2: Create AGA_j // Agent model corresponding to $ADEVS_j$
3: Create a mailbox Mb of size = $ADEVS_j$.Inports.Size
4: Create a sending box Sb of size = $ADEVS_j$.Outports.Size
5: **if** an external event appeared in port: $ADEVS_j$.Inports **then**
6: Run $AGA_j.Pa$ // corresponding to $ADEVS_j.\delta extC$
7: **end if**
8: **if** the life time ta of state S elapsed **then**
9: Run $AGA_j.Fa$ // corresponding to $ADEVS_j.\delta in$
10: Run $AGA_j.Infl_a$ // corresponding to $ADEVS_j.\lambda$
11: **end if**
12: **return** AGA
13: **end function**

Algorithm 4 CDEVS_AGC transformation function

1: **function** CDEVS_AGC (CDEVS: CoupledDEVS)
2: Create AGC_i_R // Receiver agent for $CDEVS_i$.Inports
3: Create a mailbox Mb of size = $CDEVS_i$.Inports.Size
4: Create a sending box Sb of size = $CDEVS_i$.EIC.Size
5: **if** an external event appeared in port $CDEVS_i$.Inports **then**
6: Run $AGC_i\_R.P_a$
7: **end if**
8: **if** the mailbox Mb is not empty **then**
9: Run $AGC_i\_R.F_a$
10: **end if**
11: **if** the sending box is not empty **then**
12: Run $AGC_i\_R.Infl_a$
13: **end if**
14: Create AGC_i_E // emitter agent for $CDEVS_i$.Outports
15: Create a mailbox Mb of size = $CDEVS_i$.EOC.Size
16: Create a sending box Sb of size = $CDEVS_i$.Outports.Size
17: **if** an external event appeared in port:$CDEVS_i$.Outports **then**
18: Run $AGC_i\_E.P_a$
19: **end if**
20: **if** the mailbox Mb is not empty **then**
21: Run $AGC_i\_E.F_a$
22: **end if**
23: **if** the sending box is not empty **then**
24: $RunAGC_i\_E.Infl_a$
25: **end if**
26: **return** AGC
27: **end function**

The ADEVS_AGA (ADEVS) function: grants to create agents (AGA_j) that represent the atomic DEVS models $ADEVS_j$ models (see Algorithm 3, line 2). Thus, both receptors and effectors of each agent correspond respectively to input events ($ADEVS_j.X_i$) and output events ($ADEVS_j.Y_i$) (see Algorithm 3, lines 3 and 4). The agent's internal state represents the state of the atomic model $ADEVS_j.S$. The agent's behaviour function $AGA_j.Fa$ corresponds to internal transition function $ADEVS_j.\sigma int$ (see Algorithm 3, lines 5 and 6). The function of perception of the agent $AGA_j.Pa$ represents the external transition function $ADEVS_j.\sigma ext$. the function of action of the agent represents the output function $ADEVS_j.\lambda$. The time advance mechanism on the behaviour of an agent corresponds to time advance function $ADEVS_j.ta$.

Algorithm 5 CDEVS_EIC transformation procedure

1: **procedure** CDEVS_EIC(CDEVS: CoupledDEVS)
2: **for** m = 1 until the number of $CDEVS_i.Inports$ **do**
3: **for** j = 1 until the number of internal models M **do**
4: **for** k = 1 until the number of M_j.Inports **do**
5: **if** $CDEVS_i.M_j = ADEVS_j$ **then**
6: **if** $CDEVS_i.X_m = ADEVS_j.X_k$ **then**
7: Define the set of messages between $G_i.AGC_i\_R$ and $G_i.AGA_j$
8: **end if**
9: **end if**
10: **if** ($CDEVS_i.M_j = CDEVS_j$) **then**
11: **if** ($CDEVS_i.X_m = CDEVS_j.X_k$) **then**
12: Define the set of messages between $G_i.AGC_i\_R$ and $G_j.AGC_j\_R$
13: **end if**
14: **end if**
15: **end for**
16: **end for**
17: **end for**
18: **end procedure**

Algorithm 6 CDEVS_IC transformation procedure

1: **procedure** CDEVS_IC(CDEVS: CoupledDEVS)
2: **for** j = 1 until the number of internal models M **do**
3: **for** k = 1 until the number of internal models M **do**
4: **if** $j \neq k$ **then**
5: **for** n = 1 until the number of M_j.Outports **do**
6: **for** m = 1 until the number of M_k.Inports **do**
7: **if** ($CDEVS_i.M_j = CDEVS_j$) and ($CDEVS_i.M_k = CDEVS_k$) **then**
8: **if** ($CDEVS_j.Y_n = CDEVS_k.X_m$) **then**
9: Define the set of messages between $G_j.AGC_j\_E$ and $G_k.AGC_k\_R$
10: **end if**
11: **end if**
12: **if** (($CDEVS_i.M_j = ADEVS_j$) and ($CDEVS_i.M_k = ADEVS_k$) **then**
13: **if** (($ADEVS_j.Y_n = ADEVS_k.X_m$) **then**
14: Define the set of messages between $G_i.AGA_j$ and $G_i.AGA_k$
15: **end if**
16: **end if**
17: **if** (($CDEVS_i.M_j = ADEVS_j$) and ($CDEVS_i.M_k = CDEVS_k$) **then**
18: **if** (($ADEVS_j.Y_n = CDEVS_k.X_m$) **then**
19: Define the set of messages between $Gi.AGA_j$ and $G_k.AGC_k\_R$
20: **end if**
21: **end if**

```
22:                 if ((CDEVS_i.M_j = CDEVS_j) and (CDEVS_i.M_k = ADEVS_k) then
23:                   if ((ADEVS_j.Y_n = ADEVS_k.X_m) then
24:                     Define the set of messages between Gj.AGC_j_E and Gi.AGA_k
25:                   end if
26:                 end if
27:               end for
28:             end for
29:           end if
30:       end for
31: end for
32: end procedure
```

The CDEVS_AGC ($CDEVS_j$) function: two coordinators agents AGC, namely the receiver coordinator agent (AGC_i_R), and the transmitter coordinator agent (AGC_i_E), are created by this function, for each coupled model CDEVSi (see Algorithm 4, line 14).

The procedure CDEVS_EIC: define the various interconnections between the AGA_j agent and AGC_i_R agent and AGC_i_R and AGC_j_R.

The procedure CDEVS_IC: defines the various interconnections between the AGC_j_E and AGC_k_R, AGA_j and AGA_k, AGA_j and AGC_k_R, AGC_j_E and AGA_k (see Algorithm 6, lines 9, 14, 19 and 24).

The procedure $CDEVS\_EOC$: defines the various interconnections between the AGA_j agent and AGC_i_E, AGC_j_E and AGC_i_E (see Algorithm 7, lines 6 and 11).

Algorithm 7 CDEVS_EOC transformation procedure

```
1: Procedure CDEVS_EOC (CDEVS: CoupledDEVS))
2: for j = 1 until the number of internal models M do
3:   for p = 1 until the number of Mj.Outports do
4:     for q = 1 until the number of CDEVS_i.Outports do
5:       if CDEVS_i.Mj = ADEVS_j then
6:         if ADEVS_j.Ym = CDEVS_i.Yq then
7:           Define the set of messages between Gi.AGAj and Gi.AGCi_E
8:         end if
9:       end if
10:      if (CDEVS_i.M_j = CDEVS_j) then
11:        if (CDEVS_j.Ym = CDEVS_i.Yq) then
12:          Define the set of messages between Gj.AGC_j_E and Gi.AGC_i_E
13:        end if
14:      end if
15:    end for
16:  end for
17: end for
18: end procedure
```

5 Case of study

In order to validate our approach, we chose to work on industrial control processes used in the petroleum complex SONATRACH[1] and in particular the liquefied natural gas (LNG) unit located in Skikda in the East of Algeria, which has been the subject of previous studies (Seddari and Redjimi, 2013; Seddari et al., 2013). The LNG unit is made up of the following: feed gas metering and compression, BASF CO_2 removal, fractionation of ethane, propane, butane and natural gasoline recovery, dehydration/mercury removal, APCI-liquefaction process and water suply process.

5.1 The water supply section

The supply water, is the major component that composes the natural gas liquefaction process. It provides appropriate quantities of water as requested by a boiler which generates the steam as required for other industrial processes (turbo-compressors, turbines, etc.).

This system is illustrated in Figure 3, and composed by:

- the water tank (D118)
- the condenser (E212).

The exchanges of water supply in water station are performed between two circuits, namely: the compression circuit, and the supply circuit.

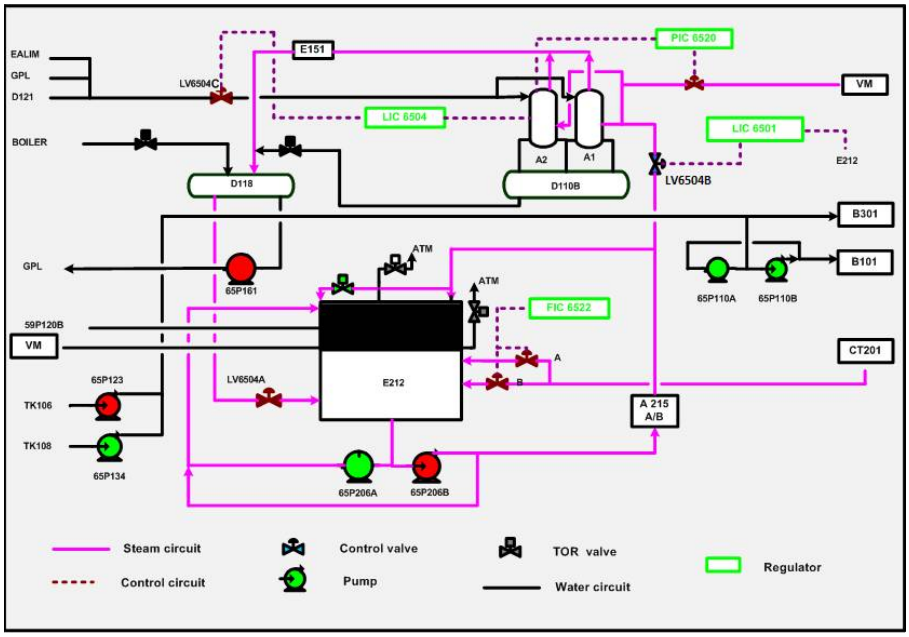
- *Compression circuit:* The extracted water will feed the heaters while passing by the LV6504B towards the degasser. This valve is controlled by the LIC6501. The level of water in the condenser (E212) is regulated to 65% by a loop of regulation LIC6501.
- *Supply circuit:* The LV6504C and LV6504A are controlled by the LIC6504 in order to maintain a constant level in the feed water tank (D118). The extraction water is then conducted towards the controller E151, and then goes to feed the water tank. The level of water in the tank (D118) is regulated to 65% by a loop of regulation LIC6504.

5.2 DEVS model of the system

To model our system, we have used primarily the DEVS formalism; therefore, we have split up our process into two coupled model. Namely, the supply coupled model and the compression coupled model. Each coupled model is composed of a set of atomic models.

- *The supply coupled model:* The decomposition of the supply coupled model to ensure the operation of water supply:
  - a *The model (LIC6504):* Shows the regulator (LIC6504), its role consists of regulating the tank. LIC6504 sends the orders of opening and closing (o_fw) towards the LV6504A valve until the measured level is equal at the desired level (65%).
  - b *The model (D118):* Shows the tank (D118), its role consists of checking the level of water (t_l) in the tank.

Figure 3 Water supply section (see online version for colours)



Source: Seddari et al. (2017)

Figure 4 Formal specification of the compression group (see online version for colours)

Compression (Com)

$G_{compression} = \{ D_{compression} \mid D_{compression} = (M_{AGC\_COM\_R}, M_{LIC6504}, M_{LV6504B}, M_{E212}, M_{AGC\_COM\_E})\}$

where :

$EIC = \{((Com, InPort_i), (LIC6501, InPort_i)), ((Com, InPort_i), (E212, InPort_i)), ((Com, InPort_j), (LV6504B, InPort_i))\}$

$EOC = \{((LV6504B, OutPort_i), (Com, OutPort_i)), ((E212, OutPort_i), (Com, OutPort_j))\}$

$IC = \{((LIC6501, OutPort_i), (LV6504B, InPort_i)), ((E212, InPort_i), (LIC6501, InPort_j)), ((LV6504B, OutPort_i), (LIC6501, InPort_k)), ((LV6504B, OutPort_i), (E212, InPort_j))\}$

LIC6504

$Percept_{LIC6504} = \{X_{LIC6504} = ((InPort_a c\_l), (InPort_b c\_w\_f), (InPort_k, t\_w\_f)) \mid X \in M_{LIC6504}$ and $M_{LIC6504} \in D_{compression}\}$

$Effector_{LIC6504} = \{Y_{LIC6504} = (OutPort_a\ o\ cn) \mid Y \subset M_{LIC6504}$ and $M_{LIC6504} \in D_{compression}\}$

$P_{LIC6504} = \{\delta\ ext\ (phase,\sigma,e,X) = (\text{“active”}, \text{compression time})$ *if phase =“ active”*
$(phase,\sigma\text{ -}e)$ *if phase =“ passive”* $\}$

$F_{LIC6504} = \{\delta\ int\ (phase,\sigma) = (\text{“passive”}, \infty)$ *if* $M_{LIC6504}.X.V = \varnothing$
(“active”, compression time) *if* $M_{LIC6504}.X.V \neq \varnothing\ \}$

$\lambda\ (\text{“active”}) = (OutPort_a\ o\ cn)$

LV6504B

$Percept_{LV6504B} = \{\ X_{MLV6504B} = ((InPort_i, t\_l), (InPort_j,), (InPort_k, o\_cn)) \mid X \in M_{LV6504B}$ and $M_{LV6504B} \in D_{compression}\}$

$Effector_{LV6504B} = \{\ Y_{MLV6504B} = (OutPort_a, t\_w\_f) \mid Y \subset M_{LV6504B}$ and $M_{LV6504B} \in D_{compression}\}$

$P_{LV6504B} = \{\delta\ ext\ (phase,\sigma,e,X) = (\text{“active”}, t\_w\_f\ calculation\ time)$ *if phase =“ active”*
$(phase,\sigma\text{ -}e)$ *if phase =“ passive”* $\}$

$F_{LV6504B} = \{\delta\ int\ (phase,\sigma) = (\text{“passive”}, \infty)$ *if* $M_{LV6504B}.X.V = \varnothing$
(“active”, compression time) *if* $M_{LV6504B}.X.V \neq \varnothing\ \}$

$\lambda\ (\text{“active”}) = (OutPort_a, t\_w\_f)$

E212

$Percept_{E212} = \{\ X_{ME212} = ((InPort_i, c\_w\_f), (InPort_j, t\_w\_f)) \mid X \subset M_{E212}$ and $M_{E212} \in D_{compression}\}$

$Effector_{E212} = \{\ Y_{ME212} = (\ OutPort_i, c\_l) \mid Y \subset M_{E212}$ and $M_{E212} \in D_{compression}\}$

$P_{E212} = \{\delta$ ext (phase,σ ,e,X)=(“active”, c_l calculation time) if phase =“ active”
(phase,σ -e) if phase =“ passive”

$F_{E212} = \{\delta$ int (phase,σ) = (“passive”,∞) if E212.X.V=$\varnothing$
(“active”, compression time) if E212.X.V$\neq\varnothing\ \}$

λ (“active”)= ($OutPort_i$, c_l)

c *The model (LV6504A):* Shows the valve (LV6504A), this valve controls water flow (c_w_f) entering into the condenser.

- *The compression coupled model:* Decomposition of the compression coupled model in a set of atomic DEVS models to carry out the compression operation of water.

  a *The model (LIC6501):* Shows the regulator (LIC6501). Its role consists of regulating the condenser. The regulator (LIC6501) sends the orders of opening and closing system (o_cn) towards the LV6504B valve until this measured level is equal to the desired level (65%).

  b *The model (E212):* Shows the condenser (E212). Its role consists of checking the variation of water level (c_l) in the condenser.

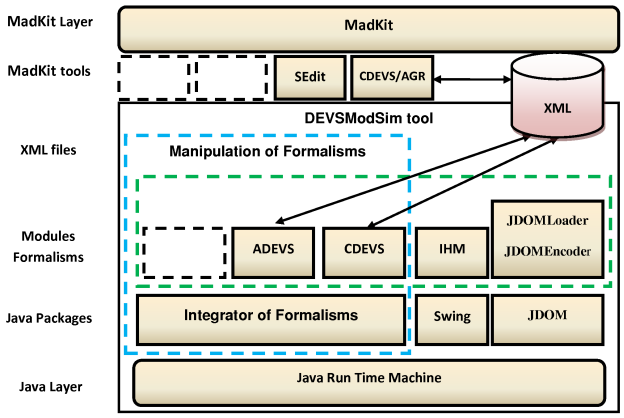
  v *The model (LV6504B):* Shows the valve (LV6504B). Its role is to control the water flow (t_w_f) coming out towards the tank.

Figure 4 presents the formal specification of the compression group according to equations shown in our proposal.

6 Proposed system

The proposed system is characterised by a multi-layer structure whose modularity is a principle to be respected in all development phases. This aspect is a natural need in the case of multi-modelling platforms. Thus, the integration of any formalism must be a systematic task. In Figure 5 we illustrate the overall architecture of the proposed system.

Figure 5 Multi-layer architecture of the proposed system (see online version for colours)



- *The MADKIT layer:* It includes the set of Java class packages that implement a core of agents. Several message and query libraries, as well as graphical development environment management tools and standard agent templates are included in this layer. The latter, comes with a set of tools which may be used to help the development of multi-agent applications.
- *The XML file layer:* Our system stores its models under XML file form for possible reuse and even for ensure portability to the MADKIT platform. Direct manipulation is provided by classes of the Java Document Object Model (JDOM) library. The latter, is a Java API that allows the manipulation of XML files. JDOM uses a set of classes rather than interfaces. Thus, the creation of a new element is the instantiation of a class. JDOM thus facilitates the manipulation of XML documents: reading a document, representation in the form of a tree structure, manipulation of this tree, the definition of a new document, export to several target formats. The parser defined via JAXP by default is used by JDOM. JDOM allows you to export an object tree from an XML document into a stream, a DOM tree or a set of SAX events.
- *The formalism layer:* It includes all the classes containing the rules and constraints of the formalism that the modeler must use to create his model through a user-friendly interface HMI developed by sawing objects. The formalism classes are equipped with plugins (schematised by two-way arrows in Figure 6) so that they can interact with JDOM, and therefore, with XML files.
- *Java packages:* Our system includes a set of libraries. A subset of libraries is imported such as JDOM and Swing. Other libraries are application-specific such as the formalism integrator, represented by a set of classes that ensures the operation of the formalism to be added to the application. The Swing package provides the graphic functionalities.
- *The Java run time machine:* This is the layer where Kava programs are executed and managed. This layer guarantees the implementation independence of operating systems. Thus Therefore any Java program will be able to run on any machine with any operating system.

For the proposed system, the models are saved in XML format, and are generated using DEVSModSim tool we developed and integrated (see Figures 6 and 7).

In the control part, we distinguish two lists to visualise the atomic and coupled DEVS models. They allow, for both models, editing and initialising (Figures 6 and 7).

The generated XML files, will be exploited and simulated by MADKIT. Figures 8 and 9 present an overview of source code of the LV6504A model and the compression group.

DEVS models saved in XML format have a certain structure. Note that a file can contain several models whether they are related or independent of each other.

Figure 6 Interface for editing an atomic model (see online version for colours)

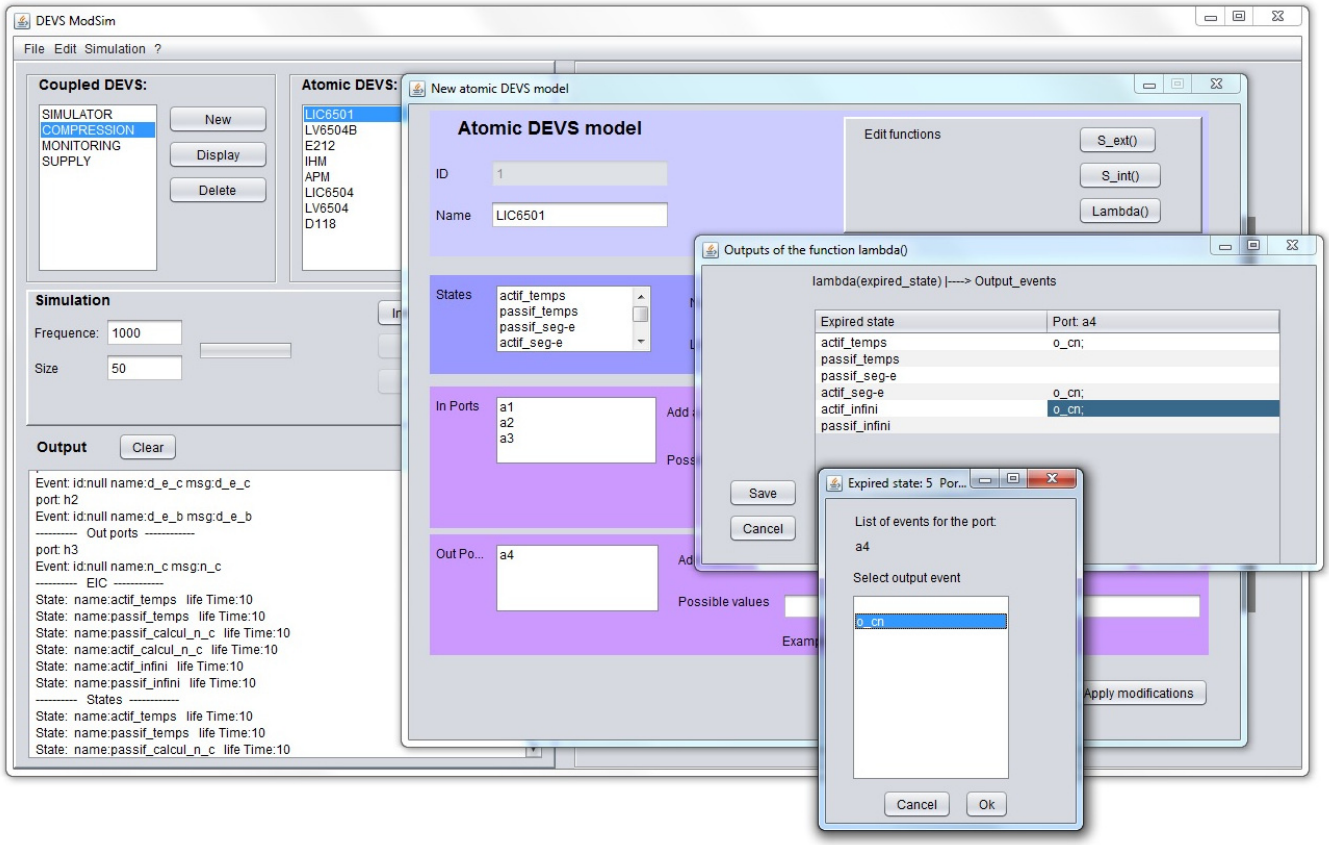


Figure 7 Interface for displaying and editing a coupled DEVS model (see online version for colours)

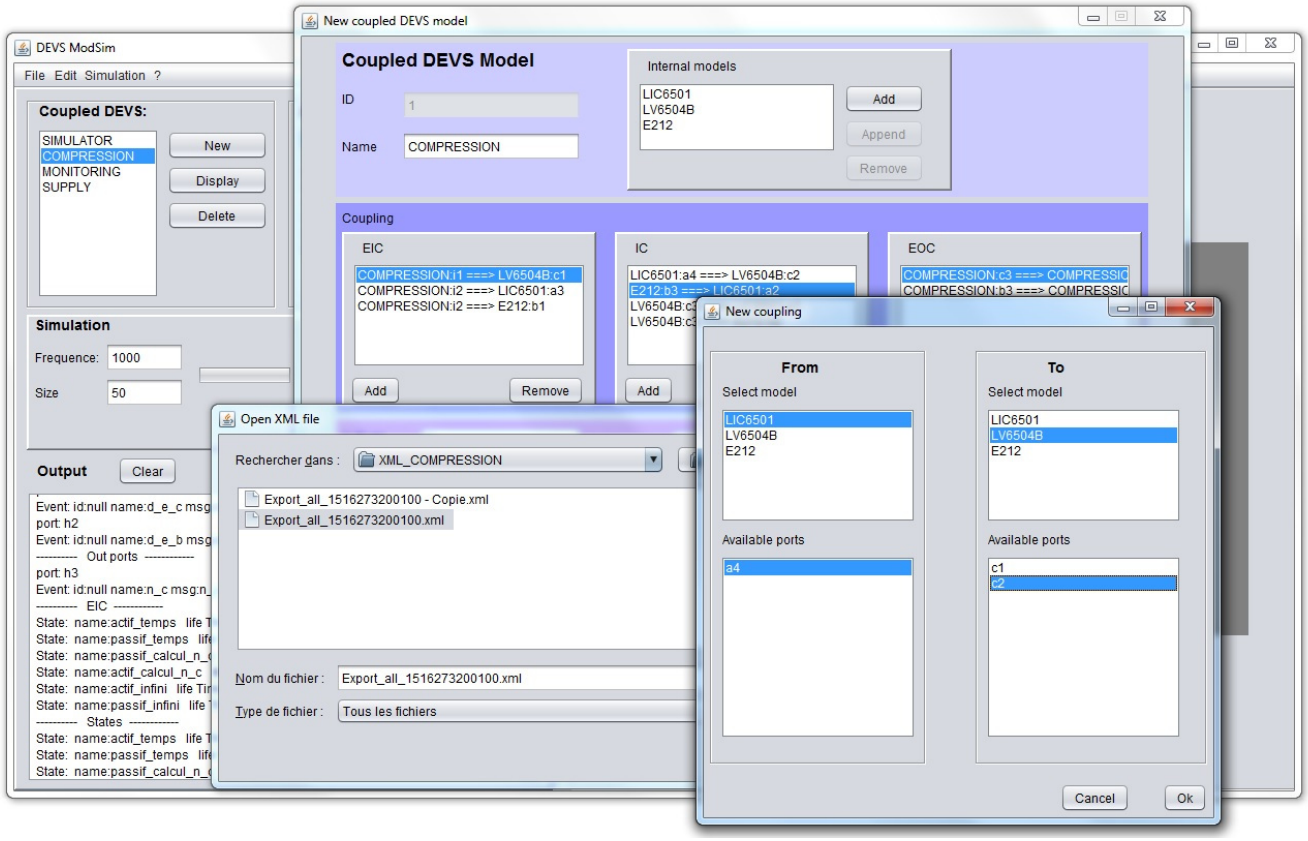
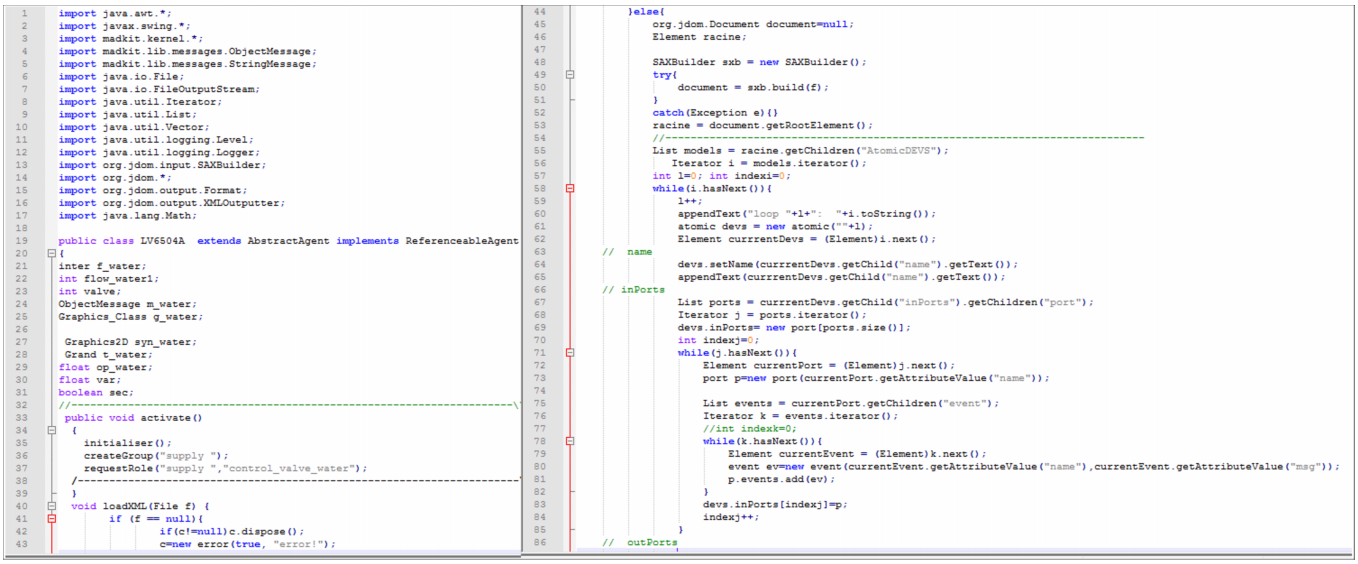


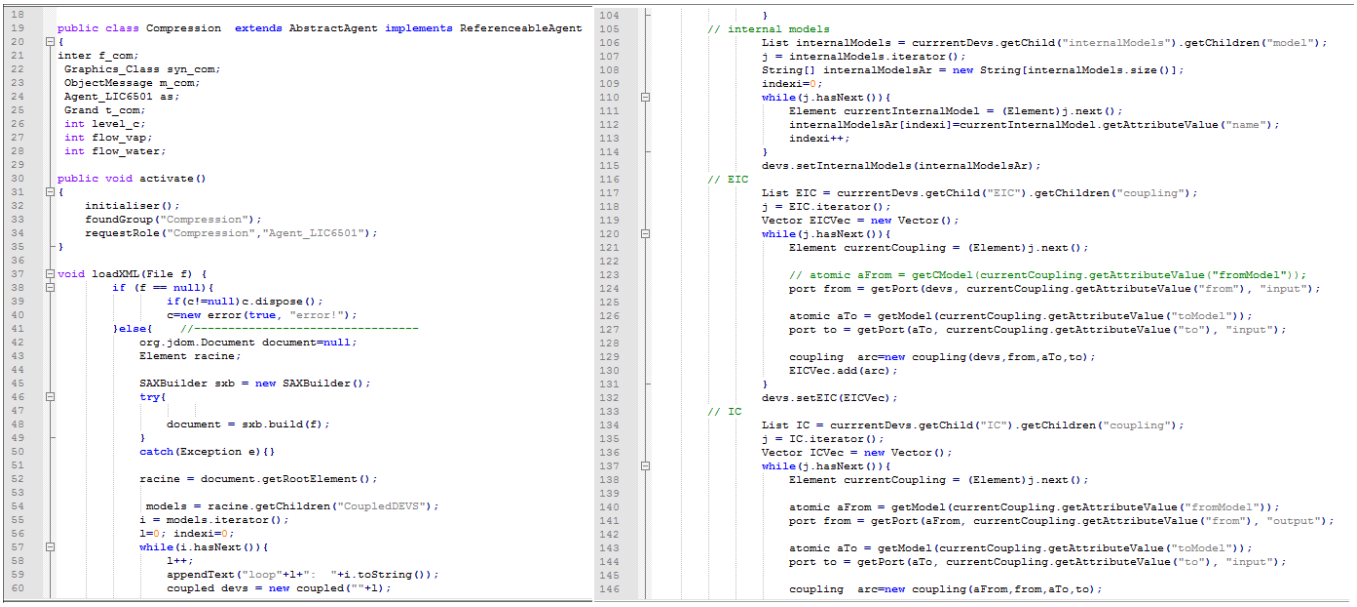
Figure 8 Overview of the LV6504A source code (see online version for colours)



```
import java.awt.*;
import javax.swing.*;
import madkit.kernel.*;
import madkit.lib.messages.ObjectMessage;
import madkit.lib.messages.StringMessage;
import java.io.File;
import java.io.FileOutputStream;
import java.util.Iterator;
import java.util.List;
import java.util.Vector;
import java.util.logging.Level;
import java.util.logging.Logger;
import org.jdom.input.SAXBuilder;
import org.jdom.*;
import org.jdom.output.Format;
import org.jdom.output.XMLOutputter;
import java.lang.Math;

public class LV6504A  extends AbstractAgent implements ReferenceableAgent
{
inter f_water;
int flow_water1;
int valve;
ObjectMessage m_water;
Graphics_Class g_water;

 Graphics2D syn_water;
 Grand t_water;
float op_water;
float var;
boolean sec;
//-------------------------------------------------------------------\
 public void activate()
  {
    initialiser();
    createGroup("supply ");
    requestRole("supply ","control_valve_water");
  /*-------------------------------------------------------------------
  }
 void loadXML(File f) {
       if (f == null){
              if(c!=null)c.dispose();
              c=new error(true, "error!");
```

```
    }else{
        org.jdom.Document document=null;
        Element racine;

        SAXBuilder sxb = new SAXBuilder();
        try{
            document = sxb.build(f);
        }
        catch(Exception e){}
        racine = document.getRootElement();
        //--------------------------------------------------------------------
        List models = racine.getChildren("AtomicDEVS");
            Iterator i = models.iterator();
        int l=0; int indexi=0;
        while(i.hasNext()){
            l++;
            appendText("loop "+l+":  "+i.toString());
            atomic devs = new atomic(""+l);
            Element currrentDevs = (Element)i.next();
//  name
            devs.setName(currrentDevs.getChild("name").getText());
            appendText(currrentDevs.getChild("name").getText());
// inPorts
            List ports = currrentDevs.getChild("inPorts").getChildren("port");
            Iterator j = ports.iterator();
            devs.inPorts= new port[ports.size()];
            int indexj=0;
            while(j.hasNext()){
                Element currentPort = (Element)j.next();
                port p=new port(currentPort.getAttributeValue("name"));

                List events = currentPort.getChildren("event");
                Iterator k = events.iterator();
                //int indexk=0;
                while(k.hasNext()){
                    Element currentEvent = (Element)k.next();
                    event ev=new event(currentEvent.getAttributeValue("name"),currentEvent.getAttributeValue("msg"));
                    p.events.add(ev);
                }
                devs.inPorts[indexj]=p;
                indexj++;
            }
//  outPorts
```

Figure 9 Overview of the compression group source code (see online version for colours)



```
public class Compression  extends AbstractAgent implements ReferenceableAgent
{
inter f_com;
 Graphics_Class syn_com;
 ObjectMessage m_com;
 Agent_LIC6501 as;
 Grand t_com;
 int level_c;
 int flow_vap;
 int flow_water;

public void activate()
{
    initialiser();
    foundGroup("Compression");
    requestRole("Compression","Agent_LIC6501");
}

void loadXML(File f) {
      if (f == null){
            if(c!=null)c.dispose();
            c=new error(true, "error!");
      }else{  //-----------------------------------
          org.jdom.Document document=null;
          Element racine;

          SAXBuilder sxb = new SAXBuilder();
          try{

              document = sxb.build(f);
          }
          catch(Exception e){}

          racine = document.getRootElement();

           models = racine.getChildren("CoupledDEVS");
          i = models.iterator();
          l=0; indexi=0;
          while(i.hasNext()){
              l++;
              appendText("loop"+l+":  "+i.toString());
              coupled devs = new coupled(""+l);
```

```
          }
      // internal models
              List internalModels = currrentDevs.getChild("internalModels").getChildren("model");
              j = internalModels.iterator();
              String[] internalModelsAr = new String[internalModels.size()];
              indexi=0;
              while(j.hasNext()){
                  Element currentInternalModel = (Element)j.next();
                  internalModelsAr[indexi]=currentInternalModel.getAttributeValue("name");
                  indexi++;
              }
              devs.setInternalModels(internalModelsAr);
      // EIC
              List EIC = currrentDevs.getChild("EIC").getChildren("coupling");
              j = EIC.iterator();
              Vector EICVec = new Vector();
              while(j.hasNext()){
                  Element currentCoupling = (Element)j.next();

                  // atomic aFrom = getCModel(currentCoupling.getAttributeValue("fromModel"));
                  port from = getPort(devs, currentCoupling.getAttributeValue("from"), "input");

                  atomic aTo = getModel(currentCoupling.getAttributeValue("toModel"));
                  port to = getPort(aTo, currentCoupling.getAttributeValue("to"), "input");

                  coupling  arc=new coupling(devs,from,aTo,to);
                  EICVec.add(arc);
              }
              devs.setEIC(EICVec);
      // IC
              List IC = currrentDevs.getChild("IC").getChildren("coupling");
              j = IC.iterator();
              Vector ICVec = new Vector();
              while(j.hasNext()){
                  Element currentCoupling = (Element)j.next();

                  atomic aFrom = getModel(currentCoupling.getAttributeValue("fromModel"));
                  port from = getPort(aFrom, currentCoupling.getAttributeValue("from"), "output");

                  atomic aTo = getModel(currentCoupling.getAttributeValue("toModel"));
                  port to = getPort(aTo, currentCoupling.getAttributeValue("to"), "input");

                  coupling  arc=new coupling(aFrom,from,aTo,to);
```

7 Conclusions

The work presented in this paper is a contribution to the M&S of DEDS. Firstly, our contribution consisted of simulating, in a methodological way, complex systems combining a formal model with an agent-based model that can be used in MADKIT platform. This platform contains a multi-formalism tool SEdit, that supports the graphical representation of different formalism and their simulations. However, the latter tool does not allow simulating of DEVS models. For this purpose, we proposed a simulation system characterised by a multilayer architecture whose modularity is principle and respected in all development phases. Thus, the integration of DEVS formalism can be a systematic task. Secondly, our method is significant: by formalisation of AGR model using DEVS equations on one hand, and defining a set of rules that systematically allows for a passage between them on the other hand; at this stage, we proceeded by conducting a formal specification of the proposed approach. Then, we opted for a transformation algorithmic tool to establish correspondence between DEVS and AGR. This correspondence preserves the properties between atomic and coupled models on one side, and agent and group concepts on other side. It also conserves other aspects like consistency, completeness, functional and safety requirements. Moreover, a practical example including the industrial control system of gas liquefaction process (water supply section) is presented and simulated by our approach. Finally, we guarantee a good performance of the proposed system witch enriches multi-agent platform by involves adding additional formalism to pre-existing formalism.

References

Anglani, A., Caricato, P., Grieco, A., Nucci, F., Matta, A., Semeraro, Q. and Tolio, T. (2000) 'Evaluation of capacity expansion by means of fuzzy-DEVS', in *ESM*, pp.129–133.

Barros, F.J. (1995) 'Dynamic structure discrete event system specification: a new formalism for dynamic structure modeling and simulation', in *Proceedings of the 27th Conference on Winter Simulation*, IEEE Computer Society, pp.781–785.

Bazoun, H., Ribault, J., Zacharewicz, G., Ducq, Y. and Boye, H. (2016) 'SLM ToolBox: enterprise service process modelling and simulation by coupling DEVS and services workflow', *Int. J. Simul. Process. Model.*, Vol. 11, No. 6, pp.453–467.

Ben-Naoum, L., Boel, R., Bongaerts, L., De Schutter, B., Peng, Y., Valckenaers, P., Vandewalle, J. and Wertz, V. (1995) 'Methodologies for discrete event dynamic systems: a survey', *Journal A*, Vol. 36, No. 4, pp.3–14.

Camus, B., Bourjot, C. and Chevrier, V. (2015) 'Combining DEVS with multi-agent concepts to design and simulate multi-models of complex systems (WIP)', in *Symposium on Theory of Modeling & Simulation – DEVS Integrative M&S Symposium (TMS/DEVS 15)*, Society for Computer Simulation International, Alexandria, VA, USA, France, April.

Dahmani, Y., Ali, H.N.B. and Boubekeur, A. (2020) 'XML-based devs modelling and simulation tracking', *International Journal of Simulation and Process Modelling*, Vol. 15, Nos. 1–2, pp.155–169.

Ferber, J. and Weiss, G. (1999) *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*, Vol. 1, Addison-Wesley Reading, Boston, MA, USA.

Ferber, J., Gutknecht, O. and Michel, F. (2003) 'From agents to organizations: an organizational view of multi-agent systems', in *International Workshop on Agent-Oriented Software Engineering*, Springer, pp.214–230.

Filippi, J.B., Bernardi, F. and Delhom, M. (2002) 'The JDEVS environmental modeling and simulation environment', in *IEMSS, Integrated Assessment and Decision Support*, pp.283–288.

Gutknecht, O. and Ferber, J. (2000) 'The MADKIT agent platform architecture', in *Workshop on Infrastructure for Scalable Multi-Agent Systems at the International Conference on Autonomous Agents*, Springer, pp.48–55.

Hamri, M.E-A., Giambiasi, N. and Frydman, C. (2006) 'Min-max-DEVS modeling and simulation', *Simulation Modelling Practice and Theory*, Vol. 14, No. 7, pp.909–929.

Hild, D.R. (2000) *Discrete Event System Specification Distributed Object Computing Modeling and Simulation*, PhD thesis.

Kim, S., Sarjoughian, H.S. and Elamvazhuthi, V. (2009) 'DEVS-suite: a simulator supporting visual experimentation design and behavior monitoring', in *Proceedings of the 2009 Spring Simulation Multiconference*, Society for Computer Simulation International, p.161.

Kofman, E., Giambiasi, N., Junco, S. et al. (2000) 'FDEVS: a general DEVS-based formalism for fault modeling and simulation', in *Proceedings of European Simulation Symposium*, pp.77–82.

Müller, J-P. (2007) 'MIMOSA: using ontologies for modelling and simulation', in *INFORMATIK 2007: Informatik trifft Logistik. Band 1. Beiträge der 37. Jahrestagung der Gesellschaft für Informatik e.V. (GI)*, Bremen, Deutschland, 24–27 September, pp.227–231.

Michel, F., Ferber, J. and Drogoul, A. (2018) 'Multi-agent systems and simulation: a survey from the agent community's perspective', in *Multi-Agent Systems*, CRC Press, pp.17–66.

Mokaddem, M., Atmani, B., Boularas, A. and Mokaddem, C.E. (2018) 'DEVSServer: ambient intelligence and devs modelling-based simulation server for epidemic modelling', *International Journal of Simulation and Process Modelling*, Vol. 13, No. 6, pp.557–581.

Mostefaoui, K. and Dahmani, Y. (2019) 'NB-DEVS: a hybrid approach for modelling and simulation of imperfect systems', *International Journal of Simulation and Process Modelling*, Vol. 14, No. 4, pp.377–388.

Ntaimo, L. and Zeigler, B.P. (2004) 'Expression of a forest cell model in parallel DEVS and timed cell-DEVS formalisms', in *Proceedings of the 2004 Summer Computer Simulation Conference*, pp.25–29.

Quesnel, G., Duboz, R. and Ramat, É. (2009) 'The virtual laboratory environment – an operational framework for multi-modelling, simulation and analysis of complex dynamical systems', *Simulation Modelling Practice and Theory*, Vol. 17, No. 4, pp.641–653.

Ricci, A., Viroli, M. and Omicini, A. (2007) 'Give agents their artifacts: the A&A approach for engineering working environments in MAS', in *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, ACM, p.150.

Sarjoughian, H.S. and Zeigler, B.P. (1999) 'The role of collaborative devs modeler in federation development', in *Proceedings of the 99 System Interoperability Workshop*.

Seck, M.D. and Honig, H.J. (2012) 'Multi-perspective modelling of complex phenomena', *Computational and Mathematical Organization Theory*, Vol. 18, No. 1, pp.128–144.

Seddari, N. (2015) *Outils Formels et Opérationnels Pour la Modélisation et la Simulation des Systèmes Complexes (Formal and Operational Tools for Modeling and Simulation of Complex Systems)*, PhD thesis, University of Skikda, Algeria.

Seddari, N. and Redjimi, M. (2013) 'Multi-agent modeling of a complex system', in *2013 3rd International Conference on Information Technology and e-Services (ICITeS)*, IEEE, pp.1–6.

Seddari, N., Redjimi, M. and Belaoued, M. (2019) 'A DEVS/MAS-based framework for modeling/simulation of complex systems', in *Advanced Intelligent Systems for Sustainable Development (AI2SD'2019)*.

Seddari, N., Redjimi, M. and Benoudina, L. (2013) 'Operational approach for modeling and simulation of an industrial process', in *2013 International Conference on Computer Applications Technology (ICCAT)*, IEEE, pp.1–6.

Seddari, N., Belaoued, M. and Bougueroua, S. (2017) 'Agent/group/role organizational model to simulate an industrial control system', *World Academy of Science, Engineering and Technology, International Journal of Social, Behavioral, Educational, Economic, Business and Industrial Engineering*, Vol. 11, No. 10, pp.2376–2385.

Seddari, N., Redjimi, M. and Boukelkoul, S. (2014) 'Using of DEVS and MAS tools for modeling and simulation of an industrial steam generator', *Journal of Computing and Information Technology*, Vol. 22, No. 3, pp.171–189.

Siebert, J. (2011) *Approche Multi-Agent Pour la Multi-Modélisation et le Couplage de Simulations. Application à L'étude des Influences Entre le Fonctionnement des Réseaux Ambiants et le Comportement de Leurs Utilisateurs*, PhD thesis, Université Henri Poincaré-Nancy I.

Siebert, J., Ciarletta, L. and Chevrier, V. (2010) 'Agents and artefacts for multiple models co-evolution: building complex system simulation as a set of interacting models', in *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, International Foundation for Autonomous Agents and Multiagent Systems, Vol. 1, pp.509–516.

Troccoli, A. and Wainer, G. (2003) 'Implementing parallel cell-DEVS', in *Proceedings of the 36th Annual Symposium on Simulation*, IEEE Computer Society, p.273.

Uhrmacher, A.M. (2001) 'Dynamic structures in modeling and simulation: a reflective approach', *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, Vol. 11, No. 2, pp.206–232.

Van Tendeloo, Y. and Vangheluwe, H. (2018) 'Discrete event system specification modeling and simulation', in *2018 Winter Simulation Conference (WSC)*, IEEE, pp.162–176.

Vangheluwe, H. (2001) 'The discrete event system specification (DEVS) formalism', *Course Notes, Course: Modeling and Simulation (COMP522A)*, McGill University, Montreal, Canada, Vol. 13.

Vangheluwe, H. (2008) 'Foundations of modelling and simulation of complex systems', *Electronic Communications of the EASST*, Vol. 10.

Versmisse, D. (2008) *Gestion de la Complexité Formelle et Opérationnelle des Systèmes Complexes: Application Aux Anthroposystèmes Marins*, Littoral.

Wainer, G.A. (2015) 'The cell-DEVS formalism as a method for activity tracking in spatial modelling and simulation', *International Journal of Simulation and Process Modelling*, Vol. 10, No. 1, pp.19–38.

Wainer, G.A. (2016) 'Real-time simulation of DEVS models in CD++', *Int. J. Simul. Process. Model.*, Vol. 11, No. 2, pp.138–153.

Zeigler, B.P., Ball, G., Cho, H., Lee, J.S. and Sarjoughian, H. (1999) 'Implementation of the DEVS formalism over the HLA/RTI: problems and solutions', in *Simulation Interoperation Workshop (SIW)*, Vol. 99.

Zeigler, B.P., Cho, H., Lee, J. and Sarjoughian, H. (1998) *The DEVS/HLA Distributed Simulation Environment and its Support for Predictive Filtering*, DARPA Contract N6133997K-0007, ECE Dept., UA, Tucson, AZ, Vol. 18.

Zeigler, B.P., Sarjoughian, H.S., Duboz, R. and Souli, J-C. (2013) *Guide to Modeling and Simulation of Systems of Systems*, Vol. 516, Springer-Verlag, London.

Zeigler, B.P., Kim, T.G. and Praehofer, H. (2000) *Theory of Modeling and Simulation*, Academic Press, USA.

Zeigler, B.P., Muzy, A. and Kofman, E. (2018) *Theory of Modeling and Simulation: Discrete Event & Iterative System Computational Foundations*, Academic Press, USA.

Zeigler, B.P., Nutaro, J. and Seo, C. (2016) 'Combining DEVS and model-checking: concepts and tools for integrating simulation and analysis', *International Journal of Simulation and Process Modelling*, Vol. 12, No. 1, pp.2–15.

Notes

1 SONATRACH is the Algerian national company for the research, production, transport, processing and marketing of hydrocarbons.