Generalized Discrete EVent System specification (G-DEVS): A State of the Art

Grégory Zacharewicz

Preprint of DOI: 10.1177/0037549718777626 journals.sagepub.com/home/sim
To appear in Sage Simulation Journal

Generalized Discrete Event System specification (G-DEVS) A State of the Art Study

Greg Zacharewicz*

** Univ. Bordeaux, IMS, UMR 5218, F-33400 Talence, France*
(e-mail: gregory.zacharewicz@ims-bordeaux.fr)

Abstract: The Generalized Discrete Event System specification (G-DEVS) language was introduced by Norbert Giambiasi in the 1990s. This paper first examines the specification of G-DEVS and gives an historical view of N. Giambiasi's works that contributed to this concept. The paper particularly focuses on the extension of G-DEVS to distributed simulation. An example of a G-DEVS model of a ski chairlift system with chair and skier is proposed to show the accuracy gained by using G-DEVS instead of other classical discrete event modeling formalisms. Then, the paper presents how G-DEVS has been extended for interoperability with other components in the context of supply chain M&S coping, with the possibility to compose different model formats at simulation time. Next, the focus is on a G–DEVS editor tool and its extensions: LSIS_DME. The distributed simulation and the HLA standard were used to support the interoperability of various models and simulators.

Keywords: Discrete Event Modeling and Simulation, G-DEVS, DEVS, Distributed Simulation, HLA.

1. INTRODUCTION

The Generalized Discrete Event System specification (G-DEVS) language was proposed by N. Giambiasi in the middle of the 1990s (Giambiasi 2000). He created G-DEVS to more accurately capture the signals and states of complex systems using polynomial approximations of real data. This formalism has been used in several works. It has been originally discussed in B. Escude's PhD thesis. Section 2 presents an overview the G-DEVS formalism from a theoretical point of view. This includes a recall of the formalism, compiled with different contributions. Section 3 presents an historical perspective situating various contributions to G-DEVS. In Section 4, these contributions are illustrated by developing a simple case study of a ski chairlift that exposes the interest of G-DEVS. This academic example shows the accuracy gains regarding the data handled by the model in a real system. Section 5 focuses on the use of G-DEVS and HLA in the context of interoperability demand with other tools. Section 6 presents the LSIS_DME modelling and simulation tool. This tool is compliant with the HLA standard. Finally, the paper presents the interoperability between G-DEVS modelling and simulation with other modelling languages and simulators in reference to model transformation and distributed simulation standards.

2. G-DEVS OVERVIEW

2.1. Formal description of G-DEVS

Traditional abstraction with discrete events approximates input-output signals as a piecewise-constant trajectory. Nevertheless, this abstraction seems limited when data relationships are not constantly linear and even more complex. In the middle of the 1990s, Norbert Giambiasi proposed extending DEVS to deal with more complex information using polynomial approximation. In detail, G-DEVS defines the abstraction of a signal with a piecewise-polynomial trajectory (Giambiasi, 2000). *Fig. 1* illustrates a comparison between a piecewise-constant segment and a piecewise-polynomial segment. Giambiasi's idea was to transmit not the unique value of an event, but rather the coefficient values that describe the polynomial that approximates real information.

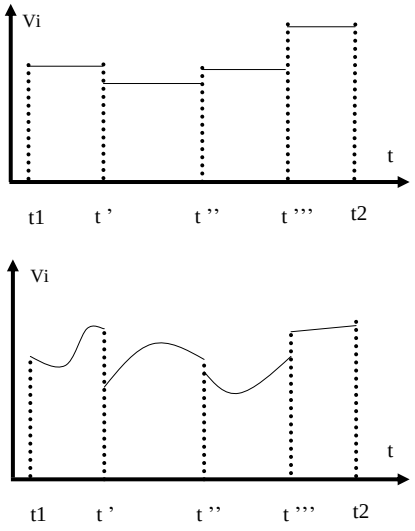

Fig. 1. Piecewise constant vs. polynomial Trajectories

Basically, in G-DEVS an input event is a list of coefficient values of the polynomial describing the continuous signal. The degree of the polynomial is the degree of the event. Therefore, when modeling a signal, DEVS can be considered as a particular case of G-DEVS, i.e., a 0-order G-DEVS. The representation of an original continuous signal is thus more accurate with G-DEVS, since many mathematical functions can be approximated with the Taylor function into polynomial functions. Then, to recreate as an output a continuous information, it can be used, with a sensible amount of data, mathematical methods such as Lagrange-Hermite interpolation approaches in order to recreate an appropriate polynomial level. However, this kind of methodology is beyond this paper's scope and are not developed here.

We recall in this section the G-DEVS formalism. As introduced previously, it finds its main interest by the fact that, from a mathematical point of view, classical discrete event abstraction approximates observe input-output signals as a piecewise-constant trajectory. As presented earlier, G-DEVS defines abstractions of signals with piecewise-polynomial trajectories. Thus, G-DEVS defines an event as a list of values. These values represent the polynomial coefficients that approximate the input-output trajectory. Formally, G-DEVS represents a dynamic system as an n-order discrete event model with:

$$DES_N = <XM, YM, S, \delta_{int}, \delta_{ext}, \lambda, D>$$

The following mappings are required:

XM = A^{n+1}, where A is a subset of integers or real numbers

YM = A^{n+1}

S = $Q \times (A^{n+1})$

For a total state S defined by the following set of state variables $(q, (a_n, a_{n-1},....., a_0), 0)$ and a continuous polynomial input segment $w : <t_1, t_2> \rightarrow X$, are defined:

The internal transition function:

$\delta_{int}\ (q, (a_n, a_{n-1},....., a_0)) =$

$$(Straj\ q, x\ ((t1+D((q, (a_n, a_{n-1},....., a_0)), x))))$$

with $x = a_n t^n + a_{n-1} t^{n-1} + \ldots\ldots + a_1 t + a_0$ and Straj model state trajectory

$\forall$ q de Q et $\forall$ $w : <t_1, t_2> \rightarrow X$,

Straj $q,w : <t_1, t_2> \rightarrow Q$

The external transition function:

$\delta_{ext}\ (q, (a_n, a_{n-1},....., a_0), e, (a_n', a_{n-1}',....., a'_0)) =$

$$(Straj\ q, x(t_1+e), x')$$

with: Coef (x) = $(a_n, a_{n-1},....., a_0)$

and Coef (x') = $(a_n', a_{n-1}',....., a'_0)$

Coef: function to associate n-coefficients of all continuous polynomial function segments over a time interval <ti, tj> to the constant (n+1) values $(a_n, a_{n-1},....., a_0)$, such as:

$w(t) = a_n t^n + a_{n-1} t^{n-1} + \ldots\ldots + a_1 t + a_0$

InCoef: $\lambda\ (q, (a_n, a_{n-1},....., a_0)) = \Lambda(q, x)$

The output function:

$\lambda : S \rightarrow A^{n+1}$

The function defining the life time of the states:

D$(q, (a_n, a_{n-1},....., a_0)) =$

MIN(e/Coef (Otraj q, $x(t_1)) \neq$

Coef (Otraj q, $x(t_1 + e)$)

with Otraj model output trajectory:

Otraj q,w:$<t_1, t_2> \rightarrow Y$

2.2. DEVS Coupled model

Zeigler (1976) introduced the concept of the coupled model. Each basic model of a coupled model interacts with the other models to produce global behavior. The basic models are either atomic models or coupled models stored in a library. G-DEVS model coupling is using same hierarchical approach as DEVS. In consequence, a G-DEVS coupled model is defined by the following structure (reused from DEVS):

MC = < X, Y, D, $\{M_d/d \in D\}$, EIC, EOC, IC, Select>

X: set of external events.

Y: set of output events.

D: set of components names.

M_d: DEVS models.

EIC: External Input Coupling relations.

EOC: External Output Coupling relations.

IC: Internal Coupling relations.

Select: defines priorities between simultaneous events intended for different components.

2.3. DEVS and G-DEVS Simulator

Ziegler (2000) proposed the concept of a DEVS abstract simulator to define the simulation semantics of a formalism. The architecture of the simulator was based on the hierarchical model structure. G-DEVS respects and uses the concepts of the DEVS abstract simulator.

The processors involved in a hierarchical simulation are: ***Simulators,*** which ensure the simulation of the atomic models; ***Coordinators***, which ensure the routing of messages between coupled models; and the ***Root Coordinator***, which ensures the global management of the simulation (*Fig. 2*).

The simulation runs due to the exchange of specific messages between the different processors:

XMessage: Represents an external event (e.g., coefficient-event vector in G-DEVS).

***message**: Represents an internal event.

Ymessage: Represents an output event.

Imessage: Initializes the model with all the default values chosen by the user.

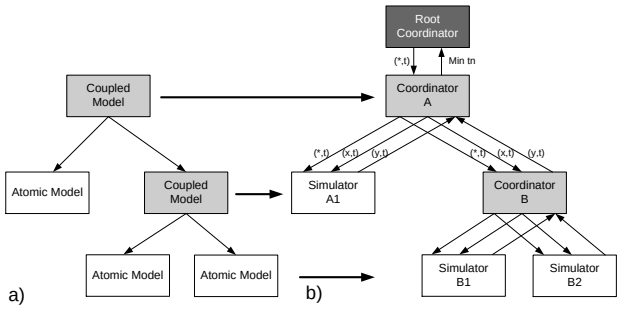

Fig. 2. From model to simulation

3. CHAIRLIFT EXAMPLE

This section presents the example of a detachable 1-seater chairlift system in which a skier is moved on a suspended chair along a cable sustained by pylons. This example is attempting to continue the series of academic examples to illustrate GDEVS, such as the bottling chain system, introduced by Giambiasi (2001), in which barrels were moved horizontally on a conveyor belt. Here, a chairlift loads and unloads skiers on a chair. Then chair moves skier from starting point to the top of a ski track, it is released by a commanded system. *Fig. 3* schematically describes this system. For educational purpose, it is assumed that chairs are detachable and released one by one and we consider in this example that only one at a time can be on the route. We use a chair speed input value to command the speed of the chairlift. The chair can be stopped in case the skier falls down at departure or arrival step, slowed down if the skier is in a difficult situation or speeded up to reduce the lost time due to previous situations. The system precisely situates the skier trough the chair position computed as a continuous value. Localizing the skier can be useful for restarting the system after a breakdown, carefully handling skier arrival or in case of emergency.

The model here was realized using the G-DEVS formalism, with the goal of showing accuracy gains compared to other formalisms like classical DEVS.

A hybrid system is one in which the model specification includes both continuous descriptions in the form of differential equations and discrete events. Praehofer (1991) developed a formalism to synthesize hybrid model specification utilizing continuous and discrete event sub-sets and illustrated the idea for a barrel generator system. Under G-DEVS, the hybrid system may be specified uniformly utilizing only discrete events and will yield similar quality results as in Praehofer's work.

The system is characterized through a continuous input (*chair speed*), a continuous output (*chair position*), and a discrete output (*skier*). *Chair position* also serves as a state variable. The chair speed causes the value of chair position to change and the first derivative of chair speed is equal to chair speed. A skier is considered reaching his destination when the chair position variable reaches the position value + 100 m; it is then outputted through the discrete output port, skier that can be used to count the number of skiers reaching the top of chairlift. Next, chair position is reset to 0. Given that chair position represents the integration of chair speed over time, its behavior is piecewise linear and its individual segments may be expressed in G-DEVS in the form of discrete coefficient-events. Thus, the chairlift system may be modeled uniformly through discrete events in G-DEVS. In contrast, in Praehofer (1991), the M&S approach mandate a traditional hybrid specification.

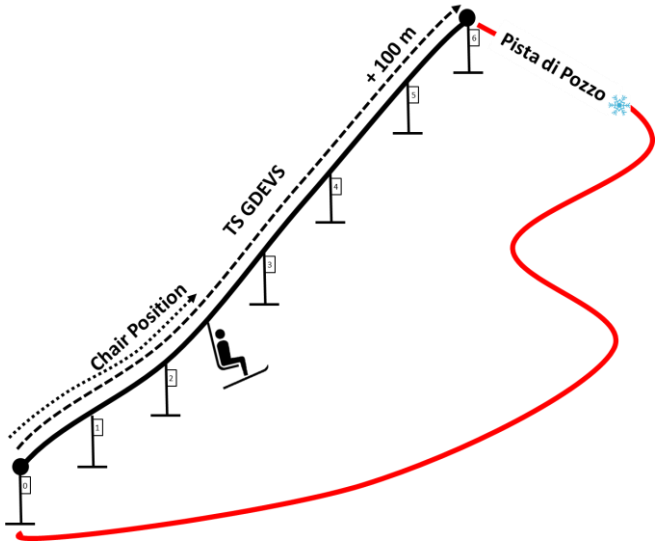

Fig. 3. Chairlift, chair (and skier) position

The progress of execution of the G-DEVS simulation is traced through the execution of the state transition and output functions. Assume that the tuple (*ac, bc*) are state variables, that represents the coefficients of the linear function: *chair position = 10 . ac . e + bc* that models the linear skier position progress with *e* the elapsed time. The state variable *sigma* specifies how the simulation time advances, so it defines the model state life time. The state variable *in* represents the memorization of the event occurrence of a new flow rate given by *chair speed.* It distinguishes if the skier is being moved at the time the event is received or not.

δext (((ac, bc), sigma, in), e, chair speed)

```
bc := 10 . ac . e + bc
ac : = chair speed
in := true
sigma := 0
```

δint (((ac, bc), sigma, in))

```
if in = false
then    bc  := 0
        sigma := 10 / ac
else    if ac = 0
        then    sigma := infinity
        else    sigma := (10- bc) / ac
        endif
```

in:= false
endif

λ (((ac, bc), sigma, in))
if in = false
then send out the skier variable to output skier
and send out (ac, 0) to port-chair position
else send out (ac, bc) to port-chair position
endif

This example reflects the use of Mealy representation in the modeling, where the generation of an output event is triggered by an input event. An output event is assumed to result at the same time the causal input event is intercepted by the model. Thus, while synthesizing a coupled model, if loops characterized by zero delays are encountered, the simulation process will fail, since an infinite number of events may be generated at the exact same time. Clearly, this is an artifact of the zero delay assumption, stemming from approximating delay values to zero. To address the problem effectively, a small but finite delay, Δt, may be introduced between the input and output events.

Fig. 4 describes graphically the evolution of the input, chair speed, output skier, and the coefficients – ac and bc – as the simulation progresses and time advances. The continuous variable chair position is determined from the discrete coefficient value utilizing the "**Coef^{-1}**" function, as presented in *Fig. 5*.

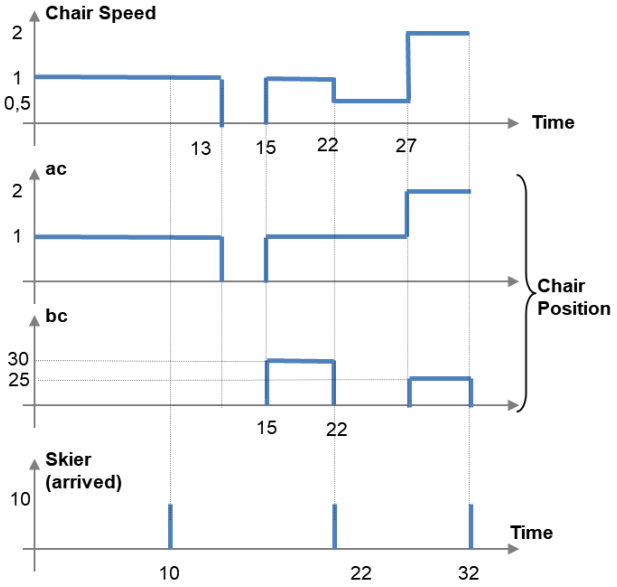

Fig. 4. Evolution of Chair Speed, Skier, and the coefficients – ac and bc – as a function of time for Chair Position.

The hybrid system serves as a good vehicle to intuitively understand the power of generalization inherent in the G-DEVS approach. Consider that the chair position variable is an Nth order function of the input inflow. To represent the state of the model would require N+1 coefficients and a G-DEVS specification may be developed along the same lines as described earlier. An input event associated with inflow will cause a state change in at least one of the N+1 coefficients, thereby triggering execution of the model.

If we focus on the chair position value state variable of the studied system, we can see that using G-DEVS can give a more precise value of the chair (and skier) position. *Fig. 5*, gives the variation of this state variable during simulation. Depending on some incidents (e.g., delay, slow down) that can occur while moving; we can observe that the skier and chair position is not the same, due to the slope of the polynomial functions that describe it. In detail, in the first case the chair position is purely linear. In the second case, a stop occurs while lifting; it follows a short delay of 3 time units. And in the last case, the chair starts more slowly and then needs to be accelerated to reach the deadline to make the skier arriving at destination on time. The result is that at any time during the simulation the skier (chair) position can be questioned to give its current value more precisely than can piecewise-constant values according to ac, bc and e. In comparison, the position is much more precise than only considering discrete steps reached such as considering, for instance, the numbers of pylons reached.

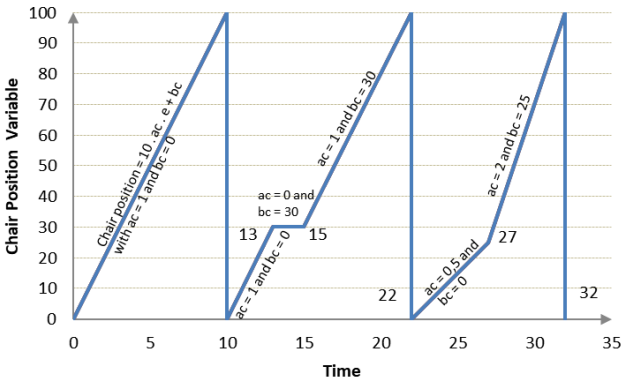

Fig. 5. Continuous representation of the "Chair Position" variable

4. HISTORICAL PERSPECTIVE OF G-DEVS' CONTRIBUTIONS

This section presents a non-exhaustive historical perspective of the different contributions proposed by Norbert Giambiasi and his team at the DIAM lab and, later, at the LSIS lab.

Bruno Escude started his PhD work with the statement that systems whose input/output are dynamic functions of time are nevertheless modelled by classical discrete event specification (e.g. DEVS) that approximates the input, output, and state trajectories through piecewise constant segments, where these segments correspond to discrete time intervals that are not necessarily equal in length. For systems that defy accurate modelling through piecewise-constant segments, B. Escude, mentored by S. Ghosh and N. Giambiasi (2000, 2001), presented G-DEVS for the first time in a journal paper as "a generalized discrete event specification", wherein the trajectories are organized through piecewise-polynomial segments. They argued that the use of polynomial functions for segments promises higher accuracy in modelling

continuous processes rather than discrete event abstractions. They also compared and observed that discrete event systems, including DEVS and G-DEVS, executed faster on computers because executions denoted significant abstractions in the system, unlike in continuous simulations where execution is continuous and exhaustive. Moreover, they observed in practice that G-DEVS' superiority over DEVS lies in its ability to discretize a system characteristic. A key contribution of G-DEVS is that it permits the development of a uniform simulation environment for hybrid, i.e., both continuous and discrete, systems. They concluded by illustrating G-DEVS on both a first-order system and a hybrid system, with piecewise linear segments. These two representative systems were modelled under G-DEVS and executed on a simulator developed for G-DEVS execution.

At the same time, Armand Damiba, a PhD student, under the supervision of N. Giambiasi, and Aziz Naamane (2001), proposed an approach combining a bond-graph with G-DEVS' formalism for the modelling and simulation of complex systems using discrete event methods. They showed how to build discrete-event simulation models for bond-graph elements, using either piecewise linear input-output trajectories or any kind of polynomial trajectories. One of the main advantages of their method is the reduction in the number of simulation steps, and therefore the possibility of studying dynamic-hybrid systems using only the discrete event paradigm.

Then, Jean Claude Carmona, N. Giambiasi, and A. Naamane (2004) described the fundamental integrator operator, the time delay operator, and the concept of output feedback. Thus, they proposed an integrator under piecewise-linear input trajectories, thus, under trajectories described by a sequence of general order polynomials (G-DEVS), ensuring a user-given accuracy. Also, they obtained smart behavior in the case of input discontinuities contrasting with the unsatisfactory examples of classical numerical solvers. Furthermore, a detailed comparison with discrete-time simulation techniques, such as Euler, allowed the assessment of an important computational gain using the proposed techniques. Finally, the complete treatment of a hybrid system not only illustrated the relevance of this approach, but underlined the interest of its application in the more general contexts of mixed-mode simulation and distributed simulation.

Later, Giambiasi and Carmona (2006) proposed to model basic continuous components of dynamic systems in a way that facilitated the transposition to a G-DEVS model, which thus offered the ability to develop a uniform approach to model hybrid systems (abstraction closer to real systems), i.e., systems composed of both continuous and discrete components. The approach was obviously a discrete-event approach, in which the choice of the time interval between two steps of calculation was based on the behavior changes of the process and no longer constant and/or a priori given. The underlying objective was to strictly satisfy a given accuracy with a low computational cost. More precisely, they presented a G-DEVS model of an integrator using polynomial descriptions of input-output trajectories. They showed its great capability of easily handling the delicate problem of input discontinuities, and made a detailed comparison with classical discrete time simulation methods, thus demonstrating its relevant properties. Several examples, including a complete hybrid system, illustrated their results.

Gabriel Wainer and N. Giambiasi (2004a) and (2005) introduced the Cell-Discrete Event System Specification (Cell-DEVS) formalism that allows defining asynchronous cell spaces with explicit timing delays (based on the specifications of the DEVS formalism). They used Cell-DEVS to solve different applications and go one step further in the definition of complex continuous systems by combining Cell-DEVS and Generalized DEVS (G-DEVS). In particular, they proposed a model describing the electrical behavior of the heart tissue, as previous research in this field has thoroughly studied this problem using differential equations and cellular automata. They showed that they can provide adequate levels of precision at a fraction of the computing cost of differential equations. They demonstrated that the use of the G-DEVS formalism is perfectly suited to deal with this category of problems, thus improving complex systems analysis. As a conclusion, Wainer and Giambiasi showed that their approach permits extending easily models to provide different actions in different cells, while not affecting performance (2004b).

Gregory Zacharewicz (2008a), as a PhD student under the supervision of Claudia Frydman and N. Giambiasi, presented a Workflow environment allowing distributed simulation based on DEVS/G-DEVS formalisms. A description language for Workflow processes and an automatic transformation of a Workflow into a G-DEVS model were defined. They introduced a new distributed Workflow Reference Model with HLA-compliant Workflow components, detailed the HLA objects shared between Workflow federates, and presented the publishing/subscribing status of each of these federates. Finally, they illustrated the use of this distributed environment with an example from Microelectronic production Workflow.

Recently, Amine Hamri, A. Naamane, and N. Giambiasi (2015) have presented and demonstrated using G-DEVS to build precise discrete-event models of logic gate design and analysis in order to get more accurate and faster simulations. In this work, states were represented with linear piecewise trajectories (G-DEVS of order 1), contrary to the classical Boolean logic models where states have constant piecewise trajectories (0 and 1). With G-DEVS models, the transition from a low level to a high one and vice versa is a linear trajectory, which is more realistic than the instantaneous transitions of classical logic gate models. They also demonstrated that this accurate representation does not require any more computations than does the DEVS model.

Several works in different application domains have been developed by using and citing G-DEVS such as Barhen (2004). They cannot be all developed here. We can cite Wainer (2004b) that accurately modeled and simulated Heart Tissue and Zacharewicz and T. Alix (2012) that modeled and simulate joint product and service design. In Le Goc (2003), the knowledge about the behavior of a continuous process has been formalized in terms of relation between discrete events so that a recursive recognition process of signatures was used to design monitoring cognitive agents. At the end, B. Zeigler

has listed G-DEVS (Zeigler, 2003) as one of the significant contribution in the recent advances of DEVS.

5. G-DEVS' DISTRIBUTED SIMULATION AND INTEROPERABLITY

For interoperability purposes, G-DEVS has been proposed to extend the specification by giving the possibility to couple not only distributed G-DEVS models, but also G-DEVS and non-G-DEVS simulation components. Inspired by Zeigler et al. (1998), Zacharewicz (2008a), during his PhD thesis under the supervision of Giambiasi and Frydman, proposed to open G-DEVS to distributed simulation. At that time, the standard for High Level Architecture (HLA) (IEEEa 2000) was rising; it was reviewed and adopted as a concrete potential frame to develop G-DEVS distributed models. The objectives of these works were to optimize resource use, work on remote resources, and/or reuse existing simulations, and more generally systems, by interconnecting them. A distributed processing must ensure interoperability, confidentiality, integrity, and causality using temporal synchronization algorithms.

5.1. Distributed Simulation

Different implementation of the RTI (Run Time Infrastructure) of HLA was realized and choice was made between open source, freeware, and commercial RTI solution implementation. Data need to be synchronized and interoperable between the different applications that exchange them. The RTI selected for these purposes in data exchange was poRTIco RTI (poRTIco, 2010). The main reason for this choice comes from the Java open source development of this 1.3-certified and 1516-compliant tool. Portico was set and parameterized under Eclipse. The FOM file includes variable objects to be shared. These objects are mainly product attributes, along with their geolocation and crossings points. The different federates involved in the platform are either G-DEVS Models or other heterogeneous software applications. The platform has been tested and used in supply chain modelling and simulation (Zacharewicz et al., 2011).

The distributed simulation was chosen to ensure the exchange of information between IS, because it can handle data from heterogeneous distributed systems without interpreting them; moreover, it has mechanisms for exchanging synchronized messages. This is a means to convey and orchestrate the exchange of data between IS, as an alternative to SOA. It is robust (running at low levels with local behavior commonly expressed by discrete event models, such DEVS (Zeigler, 1976; Zacharewicz, 2008a)) and, finally, it is completely explicit (using synchronization algorithms (Fujimoto, 2000)). The performance aspect of the messages exchanged, i.e., the interpretation of messages for the simulation, was left to the IS and was not addressed directly by the distributed simulation.

5.2. High Level Architecture Overview

The Architecture of High Level, High Level Architecture (HLA) (IEEEb, 2000), is a specification of software architecture that defines a normative framework to create global execution software consisting of distributed simulations and applications. This standard was introduced by the Office of Defense Modeling and Simulation (DMSO) (1998) of the Department of Defense (DoD). The original goal was the reuse and interoperability of military applications, simulations, and sensors. In HLA, each participating application is called federated. An HLA federate interacts within an HLA federation (Federated Group). HLA definitions have been formalized in Standard HLA 1.3 in 1996 and HLA 1516 (IEEEa, 2000) in 2000.

The report on interoperability solutions implementations (Zacharewicz, 2008a) attests that significant enterprise interoperability solutions use HLA standards to support the distributed implementation of enterprise interoperability between components at a "run time" level.

6. LSIS_DME AND ITS EXTENSIONS

6.1. LSIS-DME: DEVS (and G-DEVS) Modeling Editor

Creating G-DEVS models in an editor was a challenge in the late 1990s. In 2000, the first G-DEVS editor was proposed and released under the name of DIAM-SIM (Giambiasi, 2000). Then, in 2005, using a more recent programming language, a new editor was proposed called LSIS_DME (Zacharewicz, 2008a). This last one was chosen to recode the models recalled in the previous section and to perform simulations of these G-DEVS models. The main reason for this reimplementation came was the deprecation of the code developed in the 1990s. The elected language was Java and the eclipse platform for portability aptitudes, ability to develop lightweight web applications, and to exchange information in XML format. This new Java-based development environment has allowed the tool and G-DEVS Model created inside to be more open and interoperable with other tools and has also supported a better user-friendly graphical editor.

Also, the runtime target was improved and tailored for G-DEVS simulation formalism (Giambiasi, 2000).

As an illustration, many examples, such as the models proposed by Naamane and al. (2004), workflow models, and the conveyor example in section 3, have been graphically modelled with LSIS_DME. *Fig. 6* depicts a G-DEVS atomic model screenshot captured in the software. The software create, save, and edit G-DEVS or DEVS atomic and coupled models with only a couple of clicks and almost no lines of code required from the model maker. This permits non-specialists, including industry practitioners and students, to get into the G-DEVS modeling phase within a very short time.

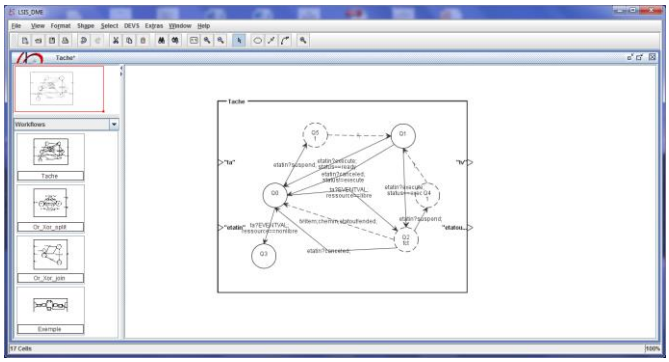

Fig. 6. G-DEVS Model in LSIS_DME

In addition, LSIS_DME has been extended to run distributed simulations. It was completed to become HLA compliant, according to the methodology provided in Zacharewicz's PhD thesis (2008a). These models are therefore potentially able to be simulated in a distributed environment according to the HLA standard.

This implements the class FederateAmbassador and callback functions from RTIAmbassador to communicate with the Run Time Infrastructure. LSIS_DME has been initially configured to connect a commercial RTI in the frame of a European project. Then, it was migrated to an open source framework (details in section 6). LSIS_DME has permitted performing distributed simulations connected with other components (client orders event generator, data bases, guidance emulator of crossing, etc.). The LSIS_DME extensions solutions for the platform was coded using the software development environment Eclipse.

6.2. LSIS_WME: Workflow Component editor

LSIS_DME's original features mostly apply to users in the field of M&S. To tackle a production chains' design, the use of G-DEVS and associated tools was not easy entry point. To overcome this problem, Zacharewicz et al. (2008b) introduced a more conceptual language associated with a graphical tool with a reduced number of concepts appropriate for non-specialists. In detail, the production data exchange defines the flow of information (like sequence planning or tracing/tracking data) between tasks to be performed by resources. In more detail, this data flow between the chain steps and the logistic partners, the products, and the client pass through products that can be equipped, for example, with RFID tags, RFID readers, and any mobile station. This flow must be orchestrated by a technical component able to manage routing, sequencing, and aggregating information. This component is in fact composed of a workflow engine that describes the organization of the proper sequence of information to exchange regarding causality and, in addition, a HLA RTI that sustains the time synchronization specified in this tool.

The WfMC proposed an XML representation of Workflow currently accepted as a standard in the Workflow community (WfMC, 1999; 2005). The XML Workflow process model structure correctness can be certified by referring to a Workflow Document Type Definition (DTD). This XML representation is not fully convenient for the XML specification of production or logistic Workflow. On the one hand, the specificities of data transiting in a flow of production need to be identified to be handled by production software and exploited at the end of flow. On the other hand, some definitions of this DTD are relative to administrative Workflow and are not required for the kind of Workflow under our scope, which can overcast the description for non-Workflow expert users.

Thus, Zacharewicz (2008a) proposed a simple graphical language to represent the components involved in Workflow dedicated to the representation of production systems. A XML Workflow process model is composed of tasks components (*Fig. 4* square items) that treat items and controllers' components (*Fig. 4* round items) that route items between tasks. Items (information, e.g., product, routing data, etc.) pass over a sequence of these components. The items are performed by resources (e.g., Valve, Conveyor, etc.). *Fig. 7* details the Workflow model of a bottling and packing chain process with the graphical Workflow Model Editor tool (LSIS_WME) developed at LSIS University Aix-Marseille (Zacharewicz, 2008a). It represents the high-level Workflow model of the information exchange (data flow) between different components. This model consists of an initialization task (Play symbol), an end task (Stop symbol) (in the figure, they are initiated and finished by server component), and several tasks for data treatment (the tasks are allocated to the components; the swim lines distinguish the responsibilities). In addition, the model contains controllers (OR-join, OR-split, etc.) to route the data between the tasks.

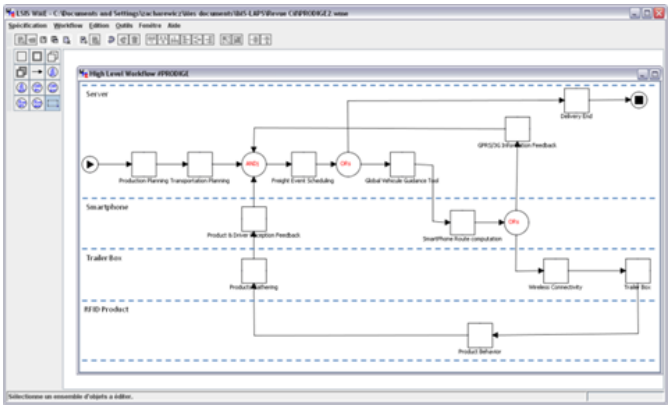

Fig. 7. Supply Chain project Workflow

Workflow models are high level models, they need to be refined for simulation. Several models have been used to develop simulation engines for Workflow. For example, Yasper and Yawl (2017) based their model on Petri nets (Van Hee, 2005).

Zacharewicz (2008b) presented the ability of LSIS_WME to generate G-DEVS specification using the XML XSLT model transformation. In the M&S framework, some parts of the Workflow model were transformed into a G-DEVS coupled

model by coupling G-DEVS atomic models (*Fig. 5*). The coupled model coupling relation was generated and a library of logistic models was proposed. This G-DEVS model takes advantage of formal properties. In conclusion, the workflow contains both G-DEVS models and other software components. HLA interoperability permits performing distributed simulation to validate the coherence of information flow to be exchanged between the partners before real execution.

6.3. Hybrid Simulation Platform

The LSIS_DME and WME editors have been involved with previous work using HLA to ensure interoperability, as mentioned in Zacharewicz (2008b) (2010). They have been reused for building an application platform dedicated to a simulation case study of supply chain model partners. The accomplishment of mixing several simulation tools and other software demonstrated that formal execution can be composed with other data handling tools such as data bases systems, ERP, etc.

Fig. 8 presents the distributed components of the platform that used HLA to communicate in a System of System approach of the announced logistic project (Zacharewicz et al., 2011). Here, the G-DEVS models were coupled with RFID databases, GIS maps, and constraint solver software.

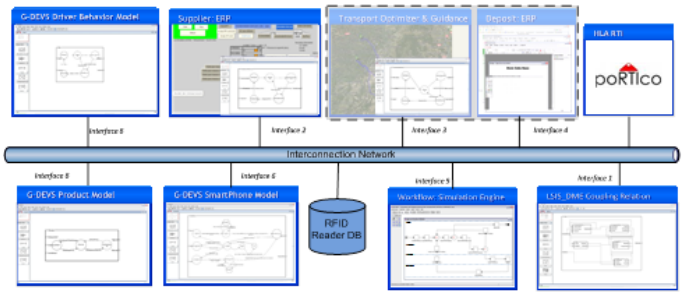

Fig. 8. G-DEVS HLA platform

7. ONGOING WORK AND PERSPECTIVES

G-DEVS workflow models have been exploited in industrial contexts to represent several industrial processes. At the same time, industrial enterprises have gradually moved their goals towards production of physical products supplemented by intangible services to differentiate themselves in a competitive market. The study of these services, their set up, and the evaluation of their efficiency is a rising research domain. In the frame of Model Driven Service Engineering Architecture (MDSEA), a service system is modeled from different points of view (static and dynamic) at different MDSEA levels: Business Service Model (BSM), Technology Independent Model (TIM), and Technology Specific Model (TSM). Simulation is a dynamic feature of MDSE, which explains the need for coherent M&S formalisms for simulation activities.

Accordingly, Hassan Bazoun (2014), in his PhD work supervised by Zacharewicz, presented the simulation of service systems based on DEVS and G-DEVS models. These works inherit the transformation of workflow to G-DEVS. Because workflow modelling is now mostly represented according to the BPMN standard, the authors defined a transformation approach of BPMN 2.0 models into DEVS and G-DEVS simulation models based on the metamodel approach, particularly with the need for data. They described the methodology for enrichment of obtained DEVS or G-DEVS models through performance indicator settings (time and costs).

Other approaches are still under development for using G-DEVS as the convergence formalism for many paradigms, both conceptual or not. G-DEVS is an unambiguous formalism ready for simulation. These works often deal with multi modeling and model transformation. Thus, this demonstrates the continuing interest in G-DEVS.

8. CONCLUSIONS

This paper proposed a state-of-the-art G-DEVS formalism. It also outlined a history of different contributions. Then, it provided the example of a chairlift system. In addition, it introduced the use of G-DEVS models in distributed simulation with the objective of interoperability with other systems. The HLA standard can be used not only for interoperability between distributed G-DEVS models, but also potentially with other simulation formalisms. Recent research has consisted in bringing missing features to G-DEVS regarding its interoperability with other software. We keep in mind that polynomial abstractions are still controlled by experts to define appropriate thresholds, for instance. One open issue is that some domain-dedicated software agents can use the semantics of the domain to determine the shape of the polynomial function to be used in the model and simulation.

9. ACKNOWLEDGEMENT

This paper is dedicated in the memory of Norbert Giambiasi, who was the initiator, director, and mentor of these works. I have special thoughts for Claudia Frydman, his wife, who has contributed to this research work by bringing the vision of a computer scientist, including the consideration of performance computing that has always been perfect complement to Norbert's vision focused on digitalization and system specification. For myself, I was mentored by both of them, particularly during my PhD study and the works presented in section 2.4 and the specific study case developed in section 2.5 at the LSIS Lab and Aix-Marseille University.

10. REFERENCES

Alix, T., Zacharewicz, G., (2012). Product-service systems scenarios simulation based on G-DEVS/HLA: Generalized discrete event specification/high level architecture. *Computers in Industry*, Elsevier, 63 (4), pp.370-378.

Barhen, S., Barhen, J., & Protopopescu, V. (2004). Asynchronous Discrete Event Systems and Emergence of Computational Chaos. *Mediterranean Multiconference on Modeling and Simulation*, Genoa, Italy October 29-31

Bazoun H., Bouanan Y., Zacharewicz G., Ducq Y., and Boye H. (2014). Business process simulation: transformation of BPMN 2.0 to DEVS models (WIP). *In* Proceedings of the Symposium on Theory of Modeling & Simulation - DEVS Integrative *(DEVS '14). Society for Computer Simulation International, San Diego, CA, USA, Article 20, 7 pages.*

Carmona, J.C., Giambiasi, N. & Naamane. (2004). Generalized Discrete Event Abstraction of Continuous Systems: Application to an Integrator, *A. Journal of Intelligent and Robotic Systems* 41: 37.

DMSO (1998). High Level Architecture, *DMSO, https://www.msco.mil/MSReferences/HLATechnicalSpecifications.aspx*, accessed November 2017.

Fujimoto R. M., (1997). "Zero lookahead and repeatability in the high level architecture", *Spring Simulation Interoperability Workshop, Orlando, FL, 3-7 March.*

Hamri, M. E. A., Giambiasi, N., & Naamane, A. (2015). Generalized discrete events for accurate modeling and simulation of logic gates. *In Concepts and Methodologies for Modeling and Simulation* (pp. 257-272). Springer International Publishing.

Giambiasi, N., & Carmona, J. C. (2006). Generalized discrete event abstraction of continuous systems: GDEVS formalism. *Simulation Modelling Practice and Theory, 14(1),* 47-70.

Giambiasi N., Escude B., Ghosh S., (2000). G-DEVS A Generalized Discrete Event Specification for Accurate Modeling of Dynamic Systems. *SCS Transactions Volume* 17, 3, p.120-134.

Giambiasi N., Escude B., Ghosh S. (2001). Generalized Discrete Event Simulation of Dynamic Systems, in*: Issue 4 of SCS Transactions: Recent Advances in DEVS Methodology-part* II, Vol. 18, pp. 216-229, Dec 2001.

IEEE std 1516-2000 (2000a). IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Framework and Rules *The Institute of Electrical and Electronic Engineers, ISBN: 0738126217, March 2001.*

IEEE std 1516.2-2000 (2000b). IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Federate Interface Specification *The Institute of Electrical and Electronic Engineers, ISBN: 0738126217, March 2001.*

Le Goc, M., & Bouche, P. (2004). Towards a Discrete Event Formalization of Sachem's Perception Based Monitoring. *IFAC Proceedings Volumes, 37(15), 203-208.*

Naamane, A., Giambiasi N., Damiba A. (2001). Generalized Discrete Event Simulation of Bond Graph. *Simulation 77(1-2): 4-22.*

poRTIco, (2017). HLA RTI Software, http://porticoproject.org, accessed November 2017.

Praehofer, H. (1991). Systems theoretic formalisms for combined discrete continuous system simulation. *Int. J. Gen. Systems, 19(3), pp. 219–240.*

Van Hee, K., Oanea, O., Post, R., Somers, L., & van der Werf, J. M. (2006). Yasper: a tool for workflow modeling and analysis, *ACSD 2006. Sixth International Conference* on *Application of Concurrency to System Design, 2006* (pp. 279-282). IEEE.

Wainer, G. A., (2004a). Performance analysis of continuous cell-DEVS models. *In Proceedings of 18th European Simulation Multiconference.*

Wainer, G. A., Giambiasi, N., (2004b). Accurate Modeling and Simulation of Heart Tissue with GDEVS/Cell-DEVS. *In MSV/AMCS (pp. 150-156).*

Wainer G. A., Giambiasi N. (2005) Cell-DEVS/GDEVS for Complex Continuous Systems. *Simulation 81(2): 137-151.*

WfMC, Workflow Management Coalition. (1999). Terminology & Glossary. *WfMC-TC-1011, 3.0, Feb.*

WfMC, Workflow Management Coalition. (2005). Workflow Process Definition Interface -- XML Process Definition Language (XPDL). *WFMC-TC-1025, Oct.*

YAWL: Yet Another Workflow Language, website, *(2017) http://yawlfoundation.org/, accessed November 2017*

Zacharewicz, G., C. Frydman, N. Giambiasi (2008a). G-DEVS/HLA Environment for Distributed Simulations of Workflows. *Simulation, 84(5), pp. 197-213.*

Zacharewicz, G., D. Chen, B. Vallespir, (2008b). HLA Supported, Federation Oriented Enterprise Interoperability, Application to Aerospace Enterprises. *EuroSISO, Edinburgh, Scotland, 08E-SIW-074, July.*

Zacharewicz G., M. E. A. Hamri, C. Frydman, N. Giambiasi, (2010). A Generalized Discrete Event System (G-DEVS) Flattened Simulation Structure: Application to High-Level Architecture (HLA) Compliant Simulation of Workflow, *Simulation,* Vol 86, Issue 3, pp. 181 - 197

Zacharewicz G., J-C. Deschamps, François J. (2011). Distributed simulation platform to design advanced RFID based freight transportation systems. *Computers in Industry* 62(6): 597-612.

Zeigler B. P. (1976) *Theory of Modelling and Simulation.* Wiley & Sons, New York, NY.

Zeigler B. P., Lee J. S., (*1998).* Theory of quantized systems: formal basis for DEVS/HLA distributed simulation environment, *Proc. SPIE Vol. 3369, p. 49-58, Enabling Technology for Simulation Science II; Alex F. Sisti; Ed.* Aug.

Zeigler B. P., Praehofer H., Kim T. G., *(2000). "Theory of Modeling and Simulation." 2nd Edition, Academic Press,* New York, NY.

Zeigler B. P., (2003). DEVS today: recent advances in discrete event-based information technology, *11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems, MASCOTS 2003. pp. 148-161.*