SIMULATING CONTINUOUS TIME PRODUCTION FLOWS IN FOOD INDUSTRY BY MEANS OF DISCRETE EVENT SIMULATION

Fabio Bursi[(a)], Andrea Ferrara[(b)], Andrea Grassi[(c)], Chiara Ronzoni[(d)]

[(a,b,c)] Dipartimento di Scienze e Metodi dell'Ingegneria
Università di Modena e Reggio Emilia
Via Amendola 2, 42122 Reggio Emilia, Italy

[(a,b,c,d)] Logistics & Automation Consulting srl
Via G. V. Catullo 22, 42124 Reggio Emilia, Italy

[(a)]fabio.bursi@unimore.it ,
[(b)]andrea.ferrara@unimore.it,
[(c)]andrea.grassi@unimore.it,
[(b)]chiara.ronzoni@lac-consulting.eu

ABSTRACT

The paper presents a new framework for carrying out simulations of continuous-time stochastic processes by exploiting a discrete event approach. The application scope of this work mainly refers to industrial production processes executed on a continuous flow of material (e.g. food and beverage industry) as well as production processes working on discrete units but characterized by a high speed flow (e.g. automated packaging lines).
The proposed model, developed adopting the DEVS formalism, defines a single generalized base unit able to represent, by means of an event scheme generated by state changes, the base behaviors needed for the modeling of a generic manufacturing unit, that is, (i) breakdowns and repairs, (ii) speed and accumulation, and (iii) throughput time. Moreover, the possibility to keep trace of additional measures of parameters related to the process and the flowing material (i.e. temperature, concentration of pollutant, and so on) is also considered. Since these parameters can change over time in a continuous manner with respect to some laws that depend on contingent conditions, the possibility to transmit those laws as functions is introduced in the model.

Keywords: continuous flow simulation, DES, DEVS.

1. INTRODUCTION

It is widely known that simulation is effectively applied in industry to address design and management issues in complex production systems that cannot be easily represented mathematically.

The use of simulation delivers added value to customers both in the deployment of a new production plant as well as in analyzing existing ones. Typical applications regards the definition of work centers capacity and buffers dimensioning and location, of process control rules, of layout configuration, accomplishing with specific design target in terms of system performances also considering different scenarios. Specially, Discrete Event Simulation (DES) has been widely adopted in the industrial context thanks to the advantages deriving from the discretization of time, that is, the possibility to speed up computation time and to ease model building activity.

As a consequence, in last decades the birth and the development of numerous discrete event simulators has been seen. However, there are industrial contexts of great relevance for which the discrete event simulation is not the best approach to represent the system since approximations are typically requested to keep the simulation time in line with industrial requirements. These sectors are, for instance, fluid processing (e.g. food and beverages industry) or high-speed automated lines (e.g. packaging lines) that, for the high processing speed, the system behaves as the same as it were a fluid process.
While a large number of works addressing the discrete event simulation of manufacturing systems have been produced in years by scientists (see for a comprehensive review Jahangirian et al. 2010), only recent papers have focused on the problem of defining simulation models able to consider the production flow as a if it were a fluid, that is, continuous (Praehofer 1991, Tamani et al. 2009). Hence, further studies to develop new approaches for the simulation analysis of production system adopting a continuous flow approach are of interest for both academics and practitioners.

The aim of this paper is to present a new modeling framework for the simulation of flow manufacturing processing following an approach that aims to reproduce the behavior of a continuous-time stochastic process. The innovation of the proposed paper resides in the definition of a generalized model able to represent a continuous-time process by using a discrete event

approach, in which the events are signals related to state changes of the simulation units. The structure of the basic model is so that all of the minimum requirements needed for modeling an industrial process are met and, moreover, the possibility to consider continuous functions for process parameters is also introduced. This results in a very efficient and scalable modeling framework.

The Discrete Event system Specification (DEVS) formalism, firstly introduced by Zeigler (Zeigler 1976, Zeigler 1984, Zeigler et al. 2000), is used in this paper to define the base unit model. DEVS allows the development of robust model representation based on the concept of atomic models and on the concept of higher-level models coupling. Several applications of DEVS for the definition of models for simulating manufacturing systems have been presented in literature. Among them, interesting contributions are Giambiasi and Carmona (2006) and Pujo et al. (2006).

The remaining of the paper is organized as follows. Section 2 defines the problem statement, while Section 3 develops the model for the base unit. Finally, Section 4 provides concluding remarks.

2. PROBLEM STATEMENT

The aim of this paper is to present a new modeling framework for the simulation of flow manufacturing processing following an approach that aims to reproduce the behavior of a continuous-time stochastic process.

This aspect is of particular interest for modeling manufacturing processes acting on a continuous flow of material (i.e. food and beverage industry), as well as processes working on discrete units but flowing at a high rate (e.g. packaging lines). For the latter, discrete event simulation is typically adopted to carry out performance analysis and to address design tasks, involving a huge overhead in terms of computation time. In fact, simulating a high capacity production line by means of a discrete event approach involves the need to manage a large number of events just to represent the flowing of the units.

Conversely, a continuous-time stochastic approach determines states and transitions probabilities, considering the manufacturing process as it were working on a continuous flow. Typically, mathematical modeling is used to model the system and to obtain the closed form solutions. The base unit model is the so-called two-machines one-buffer building block, whose first models were proposed by Zimmern (1956), Gershwin and Schick (1980), Yeralan and Tan (1997), and that obtained several improvements in years to enhance its capability to represent the behavior of real systems (Tan and Gershwin 2009, Tolio 2011, Tan and Gershwin 2011, Gebennini et al. 2011, Gebennini and Gershwin 2013). This building block is able to represent a simple series of two machines (a simple line) in which the decoupling effect of a buffer is also considered, and then it is the minimal requirement need to model a flow based manufacturing system. To model more complex systems, i.e. lines with more than one buffer, decomposition techniques have been introduced (Tan and Yeralan 1997, Gershwin and Burman 2000, Levantesi et al. 2003).

The approach proposed in this paper uses a discrete event mechanism to reproduce the behavior of a continuous-time stochastic process. To reach this goal, the base unit model (see Figure 1) has been conceived so as to manage signals, coming from the other connected units, that are delivered following a discrete event scheme. A signal transmits information about a state change in the upstream or in the downstream, then producing changes in the internal states and parameters of the unit itself. In this way, the need to model the very production flow is avoided, and then computational time is saved, while the accurate behavior of the system is granted by the transmission of the only signals needed to determine state changes.

To be able to represent production processes in industry, the basic modeling unit has been engineered so as to be able to represent three basic behaviors with which the most general real-world working unit can be modeled. In particular, those behaviors are:

1. failures and repairs;
2. working speed and accumulation;
3. throughput time.

Failures and repairs represent the operational state of the unit and are related to the Time-To-Failure (TTF) and the Time-To-Repair (TTR) profiles, that typically are random variables. Those random variables determine the occurrence of breaking and repair events, then putting the unit in down and up states, respectively. Working speed and accumulation make it possible to model changes in working speed as a consequence of state changes in the upstream and the downstream, on one side, and in the internal accumulation level on the other side. Moreover, internal accumulation level can involve changes in working speed, and this is the reason why those two aspects have to be jointly considered. It has to be pointed out that accumulation is here related to decoupling capability, so the capability of the unit to vary its content of material from zero to a maximum value. The throughput time represents a delay that has to be applied to a signal exiting from the downstream and generated as a consequence of an signal entering from the upstream.

Referring to Figure 1 the three aforementioned basic behaviors have to be considered in the reported sequence (1, 2, and 3), given the implicit interdependence among them. By means of those three basic behaviors, we are able to produce a general base unit model that can represent the main categories of working units found in real applications, such as:

- work centers continuously operating on the flow, characterized by a specific maximum production speed, zero accumulation and zero throughput time;

- buffers, characterized by a maximum speed, an accumulation greater than zero, and a throughput time depending on the buffering strategy (FIFO, LIFO, mixing, etc.);
- conveying units (i.e. conveyors, belts, pipes, etc.), characterized by a fixed speed, zero accumulation, and a throughput time depending on the length and the speed.

Summarizing, the basic behavior allows for the correct representation of the basic operation of a generic line, making also possible the computation of the classical performance measures such as throughput, efficiency, stay time in different status, and so on.

Moreover, the modeling approach here proposed provides the capability to include additional parameters whose values need to be tracked during the simulation and along the production flow. Classical examples, referring to the food industry, are the temperature of the product, or the concentration of pollutant substances that can be generated by some unwanted situations in a process unit and then propagate in some way along the production flow. The need to track those kind of parameter is clear since the use of a flow simulator in food industry is mainly related to the definition of control policies in processes and of product traceability strategies, as well as product waste and net efficiency estimation.

Hence, the modeling approach proposed in this paper allow for the addition of every parameter to monitor the user need to trace, thanks also to the a general scheme to define interactions among the monitored parameter and the basic behaviors. Moreover, the model has been conceived so as to allow the exchanging of functions between units, rather than simple values, by means of which parameter values can be calculated as a function of time without the need to generate additional events for updating parameter values.

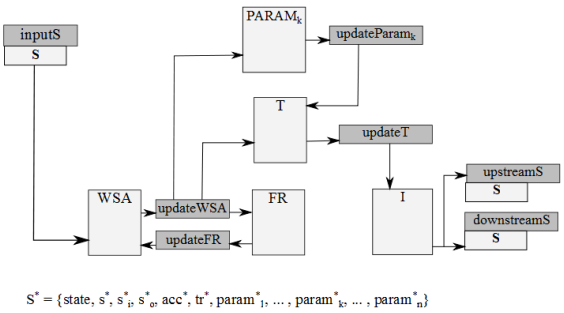

Figure 1: The Base Unit Model.

In other words, we have also adopted the continuous-time approach to model parameters value variations.

To understand the importance of this last aspect we can refer to a typical case that can be found in food industry, that is, an accumulation unit working with a pure mixing strategy (i.e. a tank) having an entering product flow coming from an upstream process and an exiting flow sent to a downstream process. If, at a certain time, the upstream process start to fail producing a product characterized by a constant concentration of pollutant, the concentration of pollutant inside the accumulation unit begins to change over time following a law that depends on the amount of product present in the unit and the flow of pollutant entering the unit itself. As a consequence, the product sent to the downstream by the accumulation unit will be characterized by a concentration of pollutant following the same law, thus continuously varying over time. From here the need to allow the possibility to transmit functions among modeled units. If we can transmit functions, we have only to generate events related to a state change and then transmitting the new functions to trace parameter values, thus avoiding the need to generate polling events just to update the values of those parameters that are varying over time following continuous laws.

3. THE BASE UNIT MODEL

3.1. Base unit object

3.1.1. Informal description

The base unit model is composed by three atomic models representing the behaviors (i) failures and repairs, (ii) working speed and accumulation, (iii) throughput time, plus one interface and n additional parameter models. This purpose of this section is to illustrate the interaction between them.

The Figure 1 shows the interactions between external signals, coming from the upstream and the downstream flows, and internal signals. The base unit is composed by standard objects and a interface whose task is to generate output signals. Considering a simple flow system, the base unit is provided with one input and two output ports for external signals, in order to send system variations both upstream and downstream the flow.

Forward signals processed among internal objects follows a static logical scheme represented by a matrix of dependencies (see Figure 2) composed by:

1. the minimal matrix managing signals among standard objects;
2. the external signals;
3. the set of rows and columns managing additional parameters signals.

	FR	WSA	T	Interface	Output	Param$_1$	...	Param$_k$	...	Param$_n$
Input	0	1	0	0	0	0	0	0	0	0
FR	0	1	0	0	0	0	0	0	0	0
WSA	1	0	1	0	0	1	1	1	1	1
T	0	0	0	1	0	0	0	0	0	0
Interface	0	0	0	0	1	0	0	0	0	0
Param$_1$	0	0	1	0	0	-	-	-	-	-
...	0	0	1	0	0	-	-	-	-	-
Param$_k$	0	0	1	0	0	-	-	-	-	-
...	0	0	1	0	0	-	-	-	-	-
Param$_n$	0	0	1	0	0	-	-	-	-	-

Figure 2: The Dependency Matrix.

The vector S^* represents the internal set of data that are used for local storing purposes.

The signal generated by the failures and repairs object (FR) is related to a state change due to the

occurrence of a TTF or of a TTR, that is, a change of the operative conditions of the unit. A signal (updateFR) is then sent to the working speed and accumulation object (WSA) for updating purposes. The opposite signaling direction (updateWSA) is also introduced, since a change in the speed of the unit can imply a variation in the way the TTF is consumed (i.e. if the failures are operation dependent or not). Updates in WSA conditions are also sent to the additional parameter objects (ADD_k) to allow the update of their values and functions.

The throughput time object (T) acts as a sort of final gateway by which every signal has to pass before being sent to the external units. The reason is that the object T is devoted to the computation of the delay with which external units located in the downstream and in the upstream realize that something has changed in the considered unit. Finally, the interface object (I) takes care of the broadcasting of the signal to the downstream and the upstream.

3.1.2. Formal description

$$\text{'base unit'} = < X_{bu}, Y_{bu}, T_{bu}, \delta_{ext}, \delta_{int}, \lambda, t > \quad (1)$$

Input event variable Xbu = (inputS) where:

- 'inputS' = S where S = { f , s, $param_1$, ... , $param_k$, ... , $param_n$}
  - f ={d,u} is the flow parameter, where ? f = 'd' represents a message coming from a downstream unit while ? f = 'u' represents a message from an upstream unit;
  - s is the working speed;
  - *paramk, for k = 1, ... ,n, is the function describing the variation of the additional parameter k over time.*

Output event variables Y_{bu} = (*downstreamS,upstreamS*) where:

- 'downstreamS' = {'u', s*,param1, ... ,paramk, ... , paramn }
  - 'u' indicates the unit in the downstream that the signal is coming from an unit in its upstream;
  - s^* is the value of the actual working speed;
  - $param_k$, for k = 1, ... ,n, is the function describing the actual law of variation of the additional parameter k over time.
- 'upstreamS' = {'d', s^*, $param_1$, ... , $param_k$, ... , $param_n$}
  - 'd' indicates the unit in the upstream that the signal is coming from an unit in its downstream;
  - s^* is the value of the actual working speed;
  - $param_k$, for k = 1, ... ,n, is the function describing the actual law of variation of the additional parameter k over time.

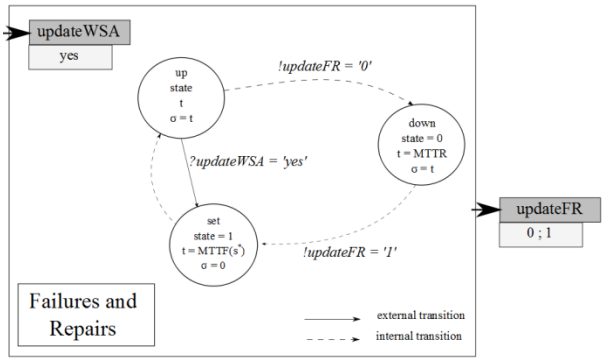

Figure 3: The Availability Object.

3.2. Failures and repairs object model

3.2.1. Informal description

The failures and repairs object (FR) models the base unit availability, switching the system from 'up' state to 'down' state and vice-versa (see Figure 3).

Let us describe the behavior in the case of the unit entering the 'up' state. Considering the 'set' phase, the base unit is set in a working state, meaning state variable set to '1', t variable set to the result of TTF profile function and s to t. After that initialization, the system switches to the 'up' state. Since then, two cases can happen:

1. a working speed chage signal arrives from the WSA object;
2. an internal state change happens.

In the first case, the incoming signal represents a notification of a speed change and it makes the system jump from the state 'up' to the state 'set' in order to update the TTF function. Since this is a transitional state, the system returns immediately to the previous state, until the amount of time s (the TTF) has passed. After this time, an internal transition from the 'up' to the 'down' state happens.

In the second case, the state is set to '0' and σ is set to the TTR value. An internal update message is created in order to notice this operative change. After σ has passed, the system makes an internal transition from the 'down' state to the 'set' state where a new TTF is computed and the state variable is set to '1'. After that, the system switches to the 'up' state for a time equal to σ. By changing the state from 'down' to 'set' an internal output signal is created. All internal signals generated are sent to the WSA object.

3.2.2. Formal description

$$\text{'}\boldsymbol{failures\ and\ repairs\ object}\text{'} = < X_{fr}, Y_{fr}, T_{fr}, \delta_{ext}, \delta_{int}, \lambda, t > \quad (2)$$

Input event variable: X_{fr} = (*updateWSA*) where:

- 'updateWSA' = {yes} indicates a work speed and accumulation variation.

State variables: T_{fr} = (phase, state, t, σ) where:

- 'phase' = {up,down, set} is a name representing the situation in the real world;
- 'state' = {0,1} represents the availability of the base unit, where ?state = '0' means the base unit is down and needs to be repaired, and ?state = '1' means the base unit is up;
- ' f ' ={TTR,TTF} represents the time to repair and the time to failure functions, that are probability distribution functions also considering the speed s^* for TTF consuming computations;
- $\sigma \in \mathbb{R} + 0 \cup \infty$ is the life time of the current state.

Output event variable: Y_{fr} = (updateFR) where:

- 'updateFR' = {0,1} indicates a variation in the system operativity.

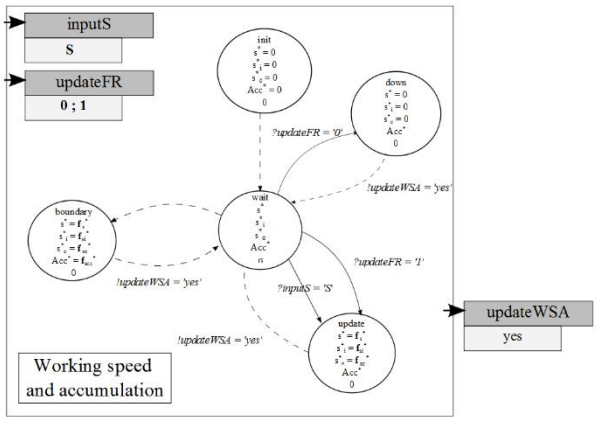

Figure 4: The Working Speed And Accumulation Object.

3.3. Working speed and accumulation object model

3.3.1. Informal description

The purpose of this object (see Figure 4) is to model the operative conditions of the base unit system. Considering the 'init' state as initialization of the object variables, all parameters are set to '0' and, by means of an internal transition, the system jumps in the 'wait' state. Thus, the working speed and accumulation object receives an !updateFR='1' and the system switches to the 'update' state: s^*_i , s^*_o , s^*,and Acc^* variables are updated to the new calculated values obtained by reading previously stored values of s*i , s*o , s*,and Acc*. In this case, the base unit restarts calculating parameters using the stored ones.

After the update have been done, the system switch to the 'wait' state creating an output signal !updateWSA = 'yes' in order to notice the changes made. The same path is followed also when there is an external signal ?inputS = 'S' entering the unit, where information about speed changes, both upstream and downstream, are stored and then used to update the unit operative condition.

As a transitional state, the system switch to the 'wait' state and remain in this state for a time that is the minimum value between infinite and t_b, the time when the base unit will be at one accumulation boundary. Boundaries are possible only when the base unit is configured as a buffer, so that boundaries are the empty level of the buffer and a level of the buffer equal to its capacity.

At the boundary state, the actual level of the buffer is calculated (empty or full). The 'boundary' state is transitional, so that the system sets its parameters, creates an internal output signal to notice the changes and then switch into the 'wait' state in order to wait that a change signal occurs. After receiving an !updateFR = '0' the system switches from the 'wait' state to the 'down' state and sets all its parameters to '0'. As a transitional state, the system switches to the 'wait' state after the creation of an internal output in order to notice an operative change.

3.3.2. Formal description

$$\textit{'working speed and accumulation object'} = < X_{WSA}, Y_{WSA}, T_{WSA}, \delta_{ext}, \delta_{int}, \lambda, t > \quad (3)$$

Input event variable: X_{WSA} = (inputS,updateFR) where:

- 'inputS' = S where S = { f, s, $param_1$, ... , $param_k$, ... , $param_n$}
  - f ={d,u} is the flowparameter, where ?f ='d' represents amessage coming froma downstream unit while ?f = 'u' represents a message from an upstream unit;
  - s is the working speed;
  - $param_k$, for k = 1, ... ,n, is the function describing the variation of the additional parameter k over time.
- 'updateFR' = {0,1} the internal signal that is generated as output by the failures and repairs object, where ?updateFR = '0' indicates that the state of the base unit is down and ?updateFR = '1' indicates that the state of the base unit is up.

State variables: T_{WSA} = (phase, s^*, s^*_i, s^*_o, acc^*, s) where:

- 'phase' ={init,down,wait,update,boundary} is a name representing the situation in the real world;
- 's*' = fs () represents a function which calculates the base unit working speed

considering the upstream and downstream base units working speed, the maximum working speed of the base unit, and the accumulation level.
- 's*i' = fsi () represents a function which determines the upstream base unit working speed.
- 's*o' = fso () represents a function which determines the downstream base unit working speed.
- 'acc∗' = facc () represents a function which determines the accumulation level considering s*, s*i, s*o and the previous accumulation level.
- $\sigma \in \mathbb{R} + 0 \cup \infty$ is the life time of the current state.

Output event variable: Y_{WSA} = (updateWSA) where:

- 'updateWSA' = {yes} indicates a working speed and accumulation variation.

3.4. Throughput time object model

3.4.1. Informal description

By referring to Figure 5, in the initial state 'init', the throughput time function tr is set for the first time. This function depends on the base unit speed, accumulation and the base unit state. Later, the object waits for an input signal in order to update its function. This function can be updated only when the object receives at least one of the two input signals. These two signals are sent by an additional parameter object or by the working speed and accumulation object. Finally, the object is able to send a signal to the Interface object.

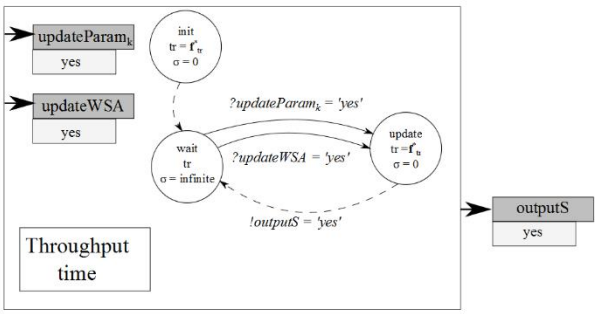

Figure 5: The Throughput Time Object.

3.4.2. Formal description

$$\text{'throughput time object'} = < X_{tr}, Y_{tr}, T_{tr}, \delta_{ext}, \delta_{int}, \lambda, t > \quad (4)$$

Input event variable: X_{tr} = (updateParamk ,updateWSA) where:

- 'updateParamk' = {yes} indicates a variation in the k-th additional parameter function;
- 'updateWSA' = {yes} indicates a work speed and accumulation variation.

State variables: T_{tr} = (phase, t_r, σ) where:

- 'phase' = {init,wait,update} is a name representing the situation in the real world;
- 'tr' = f ∗tr (s∗,acc∗, state) is the function that sets the throughput time and depends on:
  - the base unit speed s*;
  - the current accumulation acc*;
  - the base unit state.
- $\sigma \in \mathbb{R} + 0 \cup \infty$ is the life time of the current state.

Output event variable: Y_{tr} = (outputS) where:

- 'outputS' = {yes} indicates a signal to be sent to the Interface object.

3.5. Interface object model

3.5.1. Informal description

By referring to Figure 6, in the initial state 'init', the output signal is set to zero. When the interface receives a signal from the throughput time object, S is updated. For each input signal received, two different signals are generated: one is sent to upstream and the other one to the downstream. The first signal is S = {'d', s*, param*}, while the latter is S = {'u', s*, param*}. It means that the Interface object sends:

- 'flow' = {'d', 'u'}, that is, the information to the upstream and the downstream with the position reference of the considered base unit;
- the speed of considered base unit;
- the additional parameter functions of the considered base unit.

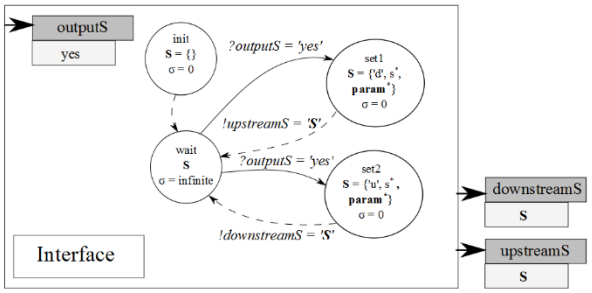

Figure 6: The Interface Object.

3.5.2. Formal description

$$\text{'}\boldsymbol{interface\ object}\text{'} = < X_{int}, Y_{int}, T_{int}, \delta_{ext}, \delta_{int}, \lambda, t > \quad (5)$$

Input event variable: X_{int} = (outputS) where:

- 'outputS' = {yes} indicates the reception of a signal from the throughput time object.

State variables: T_{int} = (phase, S, σ) where:

- 'phase' = {init,wait, set1, set2} is a name representing the situation in the real world;
- S = { f low, s*, param*} is the output signal and it consists of:
  - 'flow' = {'u', 'd'}, 'u' indicates that the considered base unit is the upstream of the destination base unit, while 'd' indicates that the considered base unit is the downstream of the destination base unit;
  - s*, the speed of the considered unit;
  - param*, the additional parameter functions of the considered base unit.
- $\sigma \in \mathbb{R} + 0 \cup \infty$ is the life time of the current state.

Output event variable: Y_{int} = (upstreamS, downstreamS) where:

- 'upstreamS' = {yes} indicates a signal to be sent to the upstream base unit;
- 'downstreamS' = {yes} indicates a signal to be sent to the downstream base unit.

3.6. Additional parameter object

3.6.1. Informal description

Looking at Figure 7, in the initial state 'init', the parameter of the additional function 'tparam' is set for the first time. This function depends on the base unit speed, accumulation, the base unit state and on the additional parameter itself. Later, the object waits for an input signal in order to update its function. This function can be updated only when the object receives a signal by the working speed and accumulation object. Finally, the object is able to send a signal to the throughput time object.

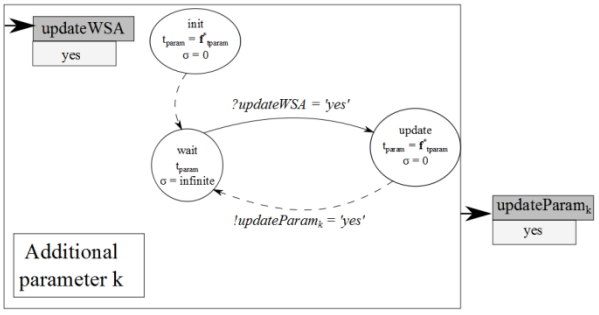

Figure 7: The generic additional parameter object.

3.6.2. Formal description

$$\text{'}\boldsymbol{additional\ parameter\ object}\text{'} = < X_{param}, Y_{param}, T_{param}, \delta_{ext}, \delta_{int}, \lambda, t > \quad (6)$$

Input event variable: X_{param} = (updateWSA) where:

- 'updateWSA' = {yes} indicates a work speed and accumulation variation.

State variables: T_{param} = (phase, t, σ) where:

- 'phase' = {init,wait,update} is a name representing the situation in the real world;
- 't' = $\{f^*_{tparam}\}$ represents the function associated to the additional parameter object.
- $\sigma \in \mathbb{R} + 0 \cup \infty$ is the life time of the current state.

Output event variable: Y_{param} = (updateParamk) where:

- '$updateParam_k$' = {0,1} indicates a variation in the additional parameter k function.

4. CONCLUSIONS

A new framework for addressing simulation of continuous-time stochastic processes by exploiting a discrete event approach is presented in the paper. The core of the framework is constituted by a base unit model able to represent the minimum set of behaviors required for the modeling of a generic unit working in a real manufacturing system (i.e. workcenter, accumulator/buffer, conveyor/pipe).

The base unit is modeled by adopting the DEVS formalism and contains a set of other atomic objects whose interactions determine the sequence of events. The base concept is that events are generated by state changes in objects and propagates both in the upstream and in the downstream to notify connected objects that something has changed. In this way, the performances of each unit is determined by the staytime in states, while events are only related to state changes, thus saving a lot of computational time.

Moreover, the possibility to keep trace of additional measures of parameters of interest for the production is also added. As the model is conceived, parameters undergoing variations over time defined by continuous laws is possible without generating overheads on the event generation.

The presented work is particularly valuable for simulating manufacturing processes executed on a continuous flow of material (e.g. food and beverage industry) as well as production processes working on discrete units but characterized by a high speed flow (e.g. automated packaging lines).

Being the presented base unit model general, its implementation in simulation systems is desirable, while further extensions can be easily developed.

REFERENCES

Gebennini, E. and Gershwin, S. B., 2013. Modeling waste production into two-machine one-buffer transfer lines. *IIE Transactions* 45 (6): 591–604.

Gebennini, E., Grassi, A., Fantuzzi, C., Gershwin, S. B. and Schick, I. C., 2011. Discrete time model for two-machine one-buffer transfer lines with restart policy. *Annals of Operations Research*: 1–25.

Gershwin, S. B. and Burman, M. H., 2000. A decomposition method for analyzing inhomogeneous assembly/disassembly systems. *Annals of Operations Research* 93:91–115.

Gershwin, S. B. and Schick I. C., 1980. *Continuous Model of an Unreliable Two-Stage Material Flow System with a Finite Interstage Buffer*. Technical Report LIDS-R-1039, OSPNo. 87049, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, USA.

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

Jahangirian, M., Eldabi, T., Naseer, A., Stergioulas, L. and Young, T., 2010. Simulation in manufacturing and business: A review. *European Journal of Operational Research* 203 (1): 1–13.

Levantesi, R., Matta, A. and Tolio, T., 2003. Performance evaluation of continuous production lines with machines having different processing times and multiple failure modes. *Performance Evaluation* 51:247–268.

Praehofer, H., 1991. System theoretic formalisms for combined discrete-continuous system simulation. *International Journal of General Systems* 19:226–240.

Pujo, P., Pedetti, M. and Giambiasi, N., 2006. Formal DEVS modelling and simulation of a flow-shop relocation method without interrupting the production. *Simulation Modelling Practice and Theory* 14 (7): 817–842.

Tamani, K., Boukezzoula, R. and Habchi, G., 2009. Intelligent distributed and supervised flow control methodology for production systems. *Engineering Applications of Artificial Intelligence* 22 (7): 1104– 1116.

Tan, B. and Gershwin, S. B., 2009. Analysis of a general Markovian two-stage continuous-flow production system with a finite buffer. *International Journal of Production Economics* 120 (2): 327–339.

Tan, B. and Gershwin, S. B., 2011. Modelling and analysis of Markovian continuous flow systems with a finite buffer. *Annals of Operations Research* 182 (1): 5–30.

Tan, B. and Yeralan, S., 1997. Analysis of multistation production systems with limited buffer capacity. Part II: The decomposition method. *Mathematical and Computer Modelling* 25 (11): 109–123.

Tolio, T., 2011. Performance evaluation of two-machines line with multiple up and down states and finite buffer capacity. *In Proceedings of the 8th International Conference on Stochastic Models of Manufacturing and Service Operations*, 117–127. 2011, Kusadasi, Turkey.

Yeralan, S. and Tan, B., 1997. Analysis of multistation production systems with limited buffer capacity. Part I: The subsystem model. *Mathematical and Computer Modelling* 25 (7): 109–122.

Zeigler, B., 1976. *Theory of modelling and simulation*. New York: John Wiley.

Zeigler, B., 1984. *Multifacetted Modelling and Discrete Event Simulation*. London: Academic Press.

Zeigler, B., Praehofer, H. and Kim, T., 2000. *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems*. Second ed. New York, NY: Academic Press.

Zimmern, B., 1956. ´Etudes de la propagation des arrˆets al´eatoires dans les chaˆınes de production. *Rev. Statist. Appl*. 4:85–104.