Proceedings of the 2017 International Conference on Industrial Engineering and Operations Management (IEOM) Bristol, UK, July 24-25, 2017

Synchronizing Discrete Event Simulation Models and System Dynamics Models

Magdy Helal
Industrial Engineering Department
American University of the Middle East, Kuwait
magdy.helal@aum.edu.kw

Luis Rabelo
Industrial Engineering and Management Systems Department
University of Central Florida, Orlando, FL, USA, 32816
luis.rabelo@ucf.edu

Abstract

This paper describes a framework to integrate and synchronize discrete event simulation (DES) models and system dynamics (SD) simulation models. SD is used to model the total manufacturing enterprise system while DES models are used to provide detailed simulations of the manufacturing processes in the system. The SD and DES models are synchronized to reach a comprehensive simulation that accommodate all management levels in a single structure, while recognizing the differences in terms of scope and frequency of decision making, and levels of details suitable at each management level. The current approach can maintain the integrity of the two simulation paradigms and can use existing/legacy simulation models without requiring learning new simulation skills. We discuss certain technical details in synchronizing the two simulation paradigms: the types and format of data exchanged, and the need for replications in DES. We propose dividing the DES model run into a number of mini runs (segments), and describe the resumption algorithm that is meant to ensure statistical validity of the data obtained from each DES segment.

Keywords:
Hybrid discrete-continuous simulation, System Dynamics, Discrete Event Simulation, Synchronization

1. Introduction

Discrete Event Simulation (DES) and System Dynamics (SD) simulation are the two main simulation paradigms in use. They are fundamentally different; following two different perspectives in viewing the world. DES models are made of sets of processes, resources, and queues where entities flow among them; competing for the resources at the processes, to have certain activities done, and cause events. Events are timeless occurrences that highlight the time consuming activities. DES has been dominating simulation modeling at the operational levels of management, in particular in the manufacturing systems applications where the main concerns are at the level of individual resources, resources, units of product, or steps of processing. At the higher management levels; with the global views of the system, DES faces difficulties. DES is a stochastic, data-driven approach, in which the state of the system is updated at the points in time when events occur. SD, on the other hand is a successful system thinking approach targeting top management levels with comprehensive integrative perspectives. With relatively minimal data requirements, it has been most effective in addressing decision makers' needs at the strategic management levels for policy analysis and design. Yet, it failed to offer the needed granularity at the operational levels (Rabelo et al. 2015; Brito et al. 2011; Helal, 2008; Helal et al. 2007; Goddeing at al. 2003).

Integrating/synchronizing/coordinating the two paradigms in a hybrid approach has been promoted for long, and several frameworks have been proposed (Morgan et al, 2017; Rabelo et al., 2015, Brito et al, 2011; 2008, Helal; Helal et al., 2007; Venkateswarana and Son 2005; Rabelo et al. 2005).). Contrasting SD and DES explicitly indicates the potential of the two paradigms complementing each other in hybrid simulation approaches to develop simulation models of the current complex business systems, where management levels are significantly overlapping. Hybrid SD-DES simulations can offer comprehensive simulation models that encompass all management levels and

recognize the differences between them in terms of scope and frequency of decision making as well as the levels of details preferred and used at each level. Conceptual differences between SD and DES can be summarized as in Table 1, (Morgan el at. 2017; Brito et al. 2011; Tako and Robinson, 2009; Helal, 2008; Morecroft and Robinson, 2005). The comparisons easily justify the development of hybrid simulations, and explain the suitability of each method for the kind of applications it is commonly used for.

Table 1. Contrasting DES and SD

	DES	SD
Perspective	Micro, analytic, emphasizing detail complexity	Macro, holistic, emphasizing dynamics complexity
World view	Network of queues, resources, and activities through which entities flow	Interrelated; overlapping feedback loops
Underlying philosophy	Vague	Well-defined
Resolution	Individual entities, attributes, events	Homogenized entities in continuous flows
Relationships	Mainly linear	Mainly non-linear
Nature of model	Stochastic	Deterministic
Level of model complexity	Increases exponentially	Increases linearly
Data needs	Data-demanding, numerical with some judgmental elements	Broadly drawn; soft variables included
Problem scope	Operational	Strategic
Time advance	Unequal time slices matching events	Equal time steps
State evolution	At discrete points in time when events occur; randomness drives behavior	Continuously with time advance; system structure drives behavior
Outputs	Statistically valid estimates and detailed performance measures, randomness vital in system performance analysis	Understanding of sources of structure behavior modes, randomness not commonly significant
Output interpretation	Statistical/mathematical background needed	Easily interpreted; no particular mathematical skills needed
System stability	Not characterized	Explicit in the feedback structure
Randomness	Explicitly included	Implicitly based on feedback delays

2. Hybrid discrete-continuous simulations

There are two basic approaches to develop hybrid simulations: the hybrid state transition machine (Maler et al., 1992; Harel, 1987) and the DEVS&DES formalism (Ziegler et al., 2000). Both organize the interaction between discrete and continuous variables, by allowing continuous variables to schedule discrete events if they cross preset threshold levels. On the other direction, discrete events can cause abrupt changes in the values of the continuous variables. In the hybrid state machine, the state diagram represents the potential states of the system and the discrete transitions among them. Some system variables are incorporated in the discrete state diagram, and modeled as continuous by differential/integral equations. Threshold values for these variables are defined. During the simulation run when a variable reaches a threshold, a *discrete event* is triggered at the state diagram and a transition from a state to another may be taken. Only events can update the system state. Upon the execution of an event in the discrete part, a continuous variable can be assigned a new value regardless of its mathematical formulation. The commercial simulation software package (http://anylogic.com/) is an implementation of the state machine system. It allows the

building of SD and DES models that can coexist and interact according to the mechanics of the hybrid state machine described above (Helal, 2008, Borshchev and Fillippov, 2004).

The DEV&DESS formalism combines Ziegler (1976)'s Discrete Event System specification (DEVS) formalism and Differential Equations System Specification (DESS) formalism, to describe hybrid systems. DEV&DESS combines the sets of inputs, outputs, states, transition, output, and rate of change functions of the two original formalisms into a unified format to specify the hybrid system. Additionally, the DEV&DESS uses the *condition function* to connect the continuous variables to the discrete components of the system. Once a threshold is crossed, the condition function is activated to schedule an event at the discrete part and an update of system state may be executed.

The two approaches are fundamentally similar; both allow the discrete component to control the continuous component. Events drive the simulation model and only events update the system state. Continuous calculations are bounded by the discrete events. The impact of the continuous calculations is communicated to the discrete components by generating a special type of events (state event) based on the values of the continuous variables as compared to predefined threshold values. The behavior of a continuous variable over time, according to that can be depicted as in Figure 1. The continuous variable behaves as continuous between discrete events. The (x, y) indicates an event number and its time of occurrence. The continuous variable must accept abrupt changes in its value by the occurrence of the discrete events. And a segment of continuous calculations between two events is not a continuation of the previous segment.

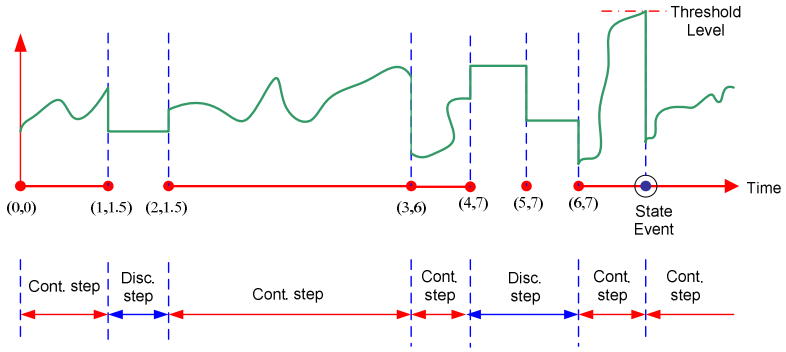

Figure 1: Intermittent continuous behavior in control-based hybrid simulation

3. The SDDES Framework

An approach to integrate and synchronize the SD and DES has been described in Helal et.al. (2007), Helal (2008), Pastrana et al. (2010), and Rabelo et. al. (2015). It is meant to develop hybrid simulations of the manufacturing enterprise systems. Called SDDES, it proposes to build a single, comprehensive SD model of the total enterprise system. Then a number of DES models for selected sub-systems of the enterprise are built to be integrated into the SD models. In Figure 2, a single DES model is connected to a causal loop diagram (CLD) portion of a SD model. The Production Start Rate is the rate at which raw materials are removed from Parts Inventory to start processing, and it is also the rate at which raw materials are converted into Work in Process. Work in Process Adjustment, Finished Inventory Adjustment, and Parts Inventory Adjustment are the amounts of materials that should be made available to maintain equilibrium. The Indicated Materials Ordering is the estimated rate of ordering new materials to satisfy the adjustment needs and the scheduled production rate (not shown). Three balancing feedback loops in the diagram interact to maintain stable materials supply and production rate.

The causal links of the Work in Process and Production Rate are cut in the SD model, to have these two variables receive their values from the DES model of the production processes. DES allows full details of all equipment, workers, and working conditions at the production process that cannot be modeled easily by SD. The SD model sends the Production Start Rate as the input to drive the DES model. At each SD computational step, the CLD of SD reacts to the values of Work in Process and Production Rate by an increase or decrease in Finished Goods Inventory level and the appropriate inventory adjustments, in addition to updating its Production Start Rate. The reaction is communicated to DES by exporting the new Production Start Rate in expectation of updated Work in process and Production Rate from DES.

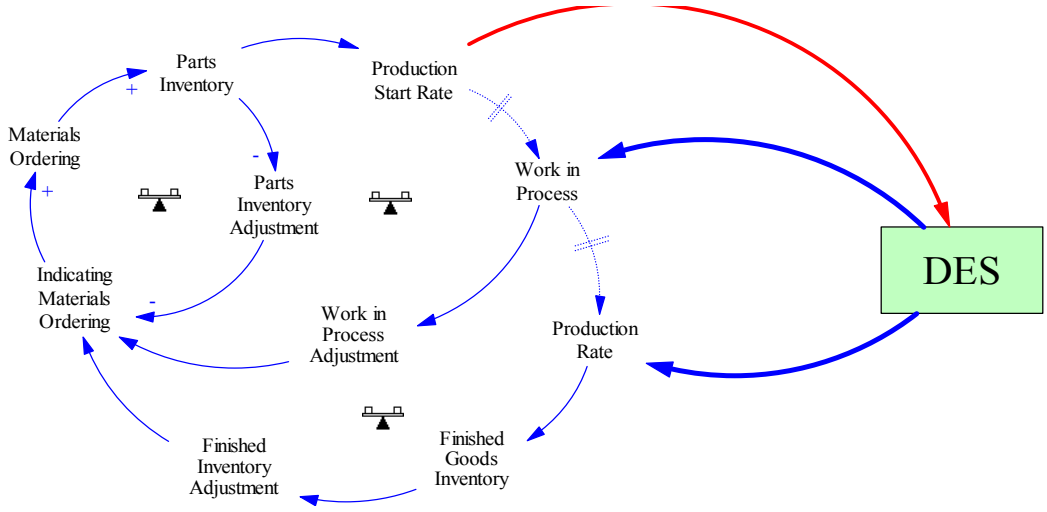

Figure 2. A DES model integrated into an SD model

In synchronizing the SD and DES models, the models are run in parallel, as represented in Figure 3. There are one SD models and several DES models interacting with SD and among each other. The planning horizon is defined by the SD model run length (L). Continuous simulation in SD is based on evaluating sets of integral equations that are evaluated recursively at specified time steps (step size ∇t). In SDDES, the DES models are run in run segments. The analysis/planning horizon of the simulation will be divided into several segments for the DES models. Each segment is a separate DES simulation run. Each DES model's run length is set equal $n\nabla t$ where n is appositive integer. We call the DES run length the time bucket (TB). The TB concept is borrowed from the traditional material requirements planning systems, which assume that time is divided into equal-length TB (weeks, days, etc.) and the system is observed at the end of each TB. An aggregate plan, for instance, could be for the upcoming three months and then it is detailed such that orders are assigned on a weekly basis to the available resources. The TB concept has been used in distributed simulations to synchronize the participating models (Lee and Wysk, 2004; Fujii et al. 2000; Ma et al. 2001; Fujii et al. 1999).

The need for segmenting the DES runs is to allow having replications in running the DES models. DES models are typically run in a number of replication in order to develop statistical estimates of the system performance indicators. SD models are deterministic and replications are not needed. To communicate with the SD model, the inputs from the DES models must be statistically valid, not based on a single model run. For instance if the DES model is to run for a month, using say 10 replications, then the one month run is divided into four runs each of one week length, or possibly into 30 runs each of a single day length. Each segment is a typical DES run, which uses the same 10 replications. At the end of the segment all data are collected as typical DES output statistical data. They are exported to the SD model and they are also used to initialize the following segment of the DES model run. In that way, we offer statistically sound data from DES models to the SD model. The exchange of data between SD and DES is made at the end of the TBs for each DES model. Since TB is equal to $n\nabla t$, where ∇t is the SD time step, and n is a positive integer, then the end of any DES run segment matches the calculation step in the SD model.

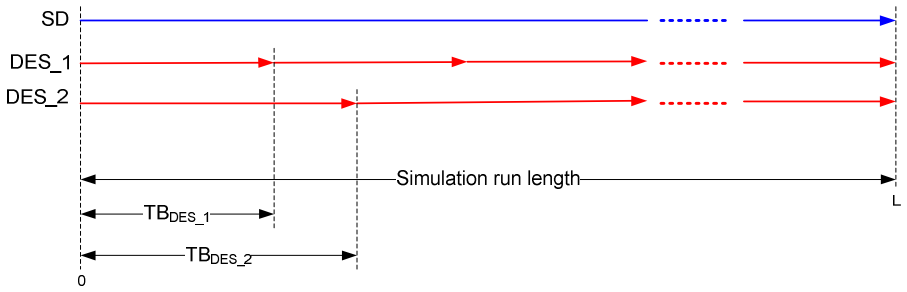

Figure 3 organizing the simulation runs using TB

This indicates an advantage of the SDDES approach over the typical state machine or the DEVS&DESS frameworks. SDDES allows the two simulation paradigms to run freely while exchanging valid data. No one paradigm will dominate the other, as it is the case in the state machine systems and in the DEVS&DESS formalisms.

In implementing that synchronization approach, two main issues arise:

1. The types of data from each type of simulation and how to make them usable in the other type.
2. How to implement the run segmenting of the DES runs and maintain the statistical validity of the DES models results.

3.1 Data in the SD and DES models

DES models generate two types of data: observational and time-persistent (Law and Kelton, 2000). Observational data are values of random variables that do not have a life time (e.g. number of units leaving the system). These values are observed at the occurrence of the relevant events then they expire as the associated entities are discarded from the system or moved to another state. They can be counted and averaged using simple arithmetic averaging. A time persistent value is a value of a random variable that is valid from the event time that generated it until the next event that updates it. An example is the number of entities waiting in a queue.

SD models are constructed of two types of variables: stocks and flows. Stocks are accumulators whose values are functions in the time (e.g. inventory, cash balance) and they continue to exist even when the system is brought to a frozen state. Flows are the rates of increase or decrease of the stock levels. Flows cannot exist if the system is stopped. Flows represent the activities and policy actions while stocks represent the outcomes of these activities. In SDDES, data from DES are used in SD and data from SD are used in DES as depicted in Figure 4 and explained afterward.

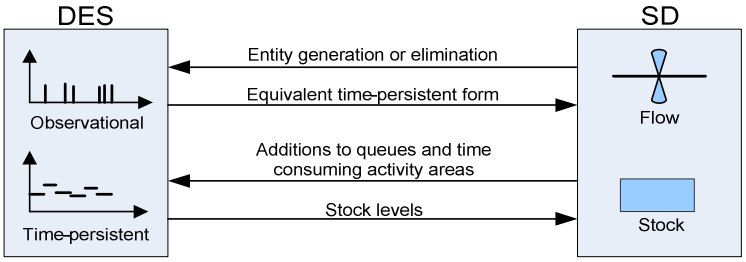

Figure 4 Correspondence between SD and DES data in SDDES

3.1.1 Using SD stocks and flows data in DES

SD stocks accumulate units of products, people, orders, etc. SD stocks correspond to DES' queues, buffers, storage areas, and where entities can be held for non-zero time intervals in DES. For example, the work in process stock in the SD module corresponds to all unfinished product units in the DES model that are waiting in queues or undergoing processing (delays). The contents of a SD stock can be exported to queues and delay objects in the DES modules and they are directly usable in DES.

Figure 5 (Top) shows a SD model with a single stock of the level of work in process (WIP) and two flow rates (e.g. materials arrival rate and production rate). The figure shows a DES model of two processes and two queues before them. The DES is the details of the process modeled implicitly by the WIP stock in the SD model. The WIP level in SD is translated into the number of entities in the DES model, at the queues of the two processes (assuming the two processes are the only contributors to WIP in this part of the system).

In the opposite direction, the SD's WIP data is collected from DES. The number of units in the queues and the number of units being processed are given to the WIP stock as shown in Figure 5 (Bottom). When implemented in SDDES, the SD stock receives more accurate values of the flow rates based on the DES model data. The inflow and outflow of the WIP stock are exported and imported to and from DES. That is the DES model has replaced the WIP stock in the SD model with the full details of processing activities. When the end of the DES run segment matches the calculation steps of the SD model, then we provide the SD with more realistic values while maintain the the integrity of the SD feedback loops.

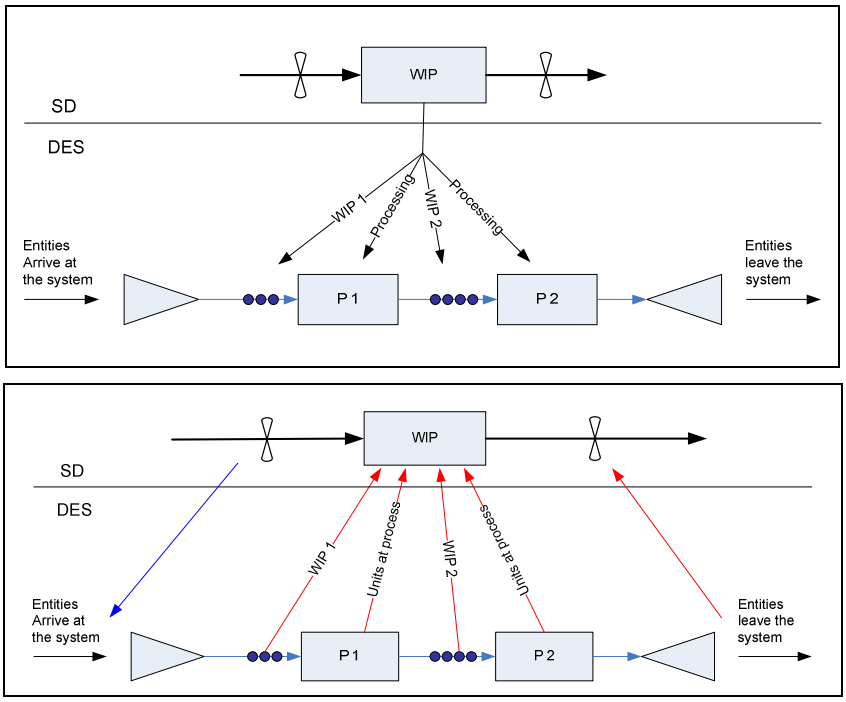

Figure 5 Exporting SD stock contents to DES (Top) and importing SD stock contents from DES (Bottom)

3.1.2 Using DES observational data in SD

Observational data exist in DES models only for the duration of the event that created them, which is zero. Observational data can be mapped to flow rates in SD. For dimensional correctness in SD, observational data should be transformed into equivalent quantities that can be measured in the same units of the flow rates they map to. Counted over the duration of the DES run segment, the DES observational data can map to an equivalent SD flow rate using the following relationship.

$$\text{SD equivalent of DES counted data} = \text{DES count}\left(\frac{n\Delta t}{SD\ Base\ Time\ Unit}\right)$$

Referring to Figure 6 (left), let X be the random variable of the number of entities removed from a system modeled by DES. At the event times of removing the entities, X may take the values of x_1, x_2, x_3, … etc. at the event times t_1, t_2, t_3, … respectively. To estimate a removal rate from DES, the number of entities over the in-between events times is divided by the time interval length. The rate estimates are denoted by X_{SD}. If the DES model had started idle and empty at time zero, then the system needed t_1 time units to deliver x_1 entities. The equivalent rate over t_1, X_{SD1} is estimated as the ratio of x_1 to t_1. Rates over other periods are estimated similarly. When exported to SD, the estimated equivalent value over each time interval is used for the corresponding SD flow rate variable.

3.1.3 Using DES time-persistent data in SD

DES time persistent data are usable in SD as they are generated in DES and they can map to the SD stocks. Both time-persistent data and SD stock contents represent quantities that stay valid over nonzero time intervals. The time-persistent values are recorded at the times of the events that affect them. More than one event can occur during the DES run segment that affect the time-persistent value. The end-of-run DES value of the time-persistent variable maps to the appropriate SD stock level at the same point in time.

It is possible to estimate a time-weighted average of the DES time-persistent variable over the DES run and export it to the SD. In this case the average value may not be usable for a stock level. Such a DES value can add more information to the SD model. For instance, if the work in process data is collected from DES, the level of work in process at the end of the DES run corresponds to the relevant SD stock. Whereas a weighted average value of the work in process over the DES run can be used to indicate the level of system crowdedness, which can be used to estimate a labor satisfaction measure, or an indicator of the shop floor quality of environment.

Assuming a DES segment length of $n\Delta t$, Figure 6 (Right) illustrates these two approaches. Some accuracy analysis will be needed since the weighted-average value will correspond to somewhere between the beginning and the end of the SD computational step. The SD methodology calculates the stock level at the end of the time step.

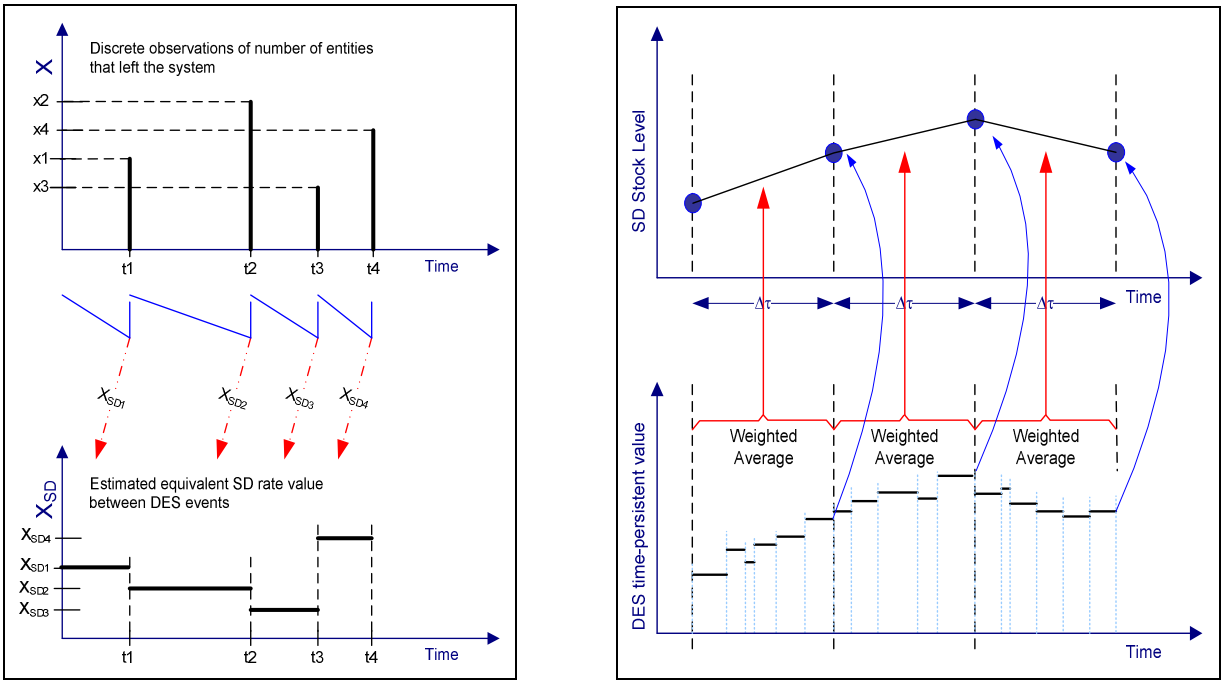

Figure 6 Mapping observational data from DES to equivalent rate variables in SD (Left) and mapping DES time-persistent data to SD stock levels (Right)

3.2 The DES Run Segment Resumption

The SDDES synchronization mechanism requires DES models to run not for the entire planning horizon for which the SDDES simulation is run, but for run segments. A run segment is a complete discrete simulation run with the necessary number of replications. At the end of the segment the DES models export outputs to the SD and other DES models and receive inputs from them. The choice of the TB size is a modeling decision that is made considering the desirable levels of accuracy and efficiency as well as the function modeled in the particular DES models. The SDDES simulation run length L is divided into integer number of segments for each DES model. The minimum size of TB is the SD computational time step; Δt. For a DES model m the length of its run segment is $TB_m = L/n\Delta t$, where n is a positive integer.

Figure 7 shows the synchronization sequence actions (numbered as shown on the arrows) of advancing each model' simulation time and performing the data exchanges. Three DES models with TB sizes are assumed. The base TB is equal to the SD computational time step Δt, for simplicity. The first DES model ($TB_1 = \Delta t$) requires exchanging data at each SD computational time step. The second DES model ($TB_2 = 2\Delta t$) is slower and requires exchanging data every 2 SD time steps. The third ($TB_3 = 5\Delta t$) is even slower and requires data exchanges every 5 time steps.

3.2.1 The Resumption Algorithm

Consider a basic production line modeled in DES, which will be synchronized with a comprehensive SD model of the enterprise system. The production line consists of a number of processing workstations in a general flow line design. Equipment may fail randomly. Transportation between equipment is included in processing times. The SDDES synchronization sequence requires dividing each DES model run into segments. At the end of each segment the state of the part of the system modeled by DES is saved. And at the start of the following segment the saved state is used to initialize the run, thus the DES model resumes from where it stopped at the end of the previous segment. The resumption is vital for the statistical validity and correctness of the results obtained from each DES model. In the DES model of this basic production line, resumption ensures that:

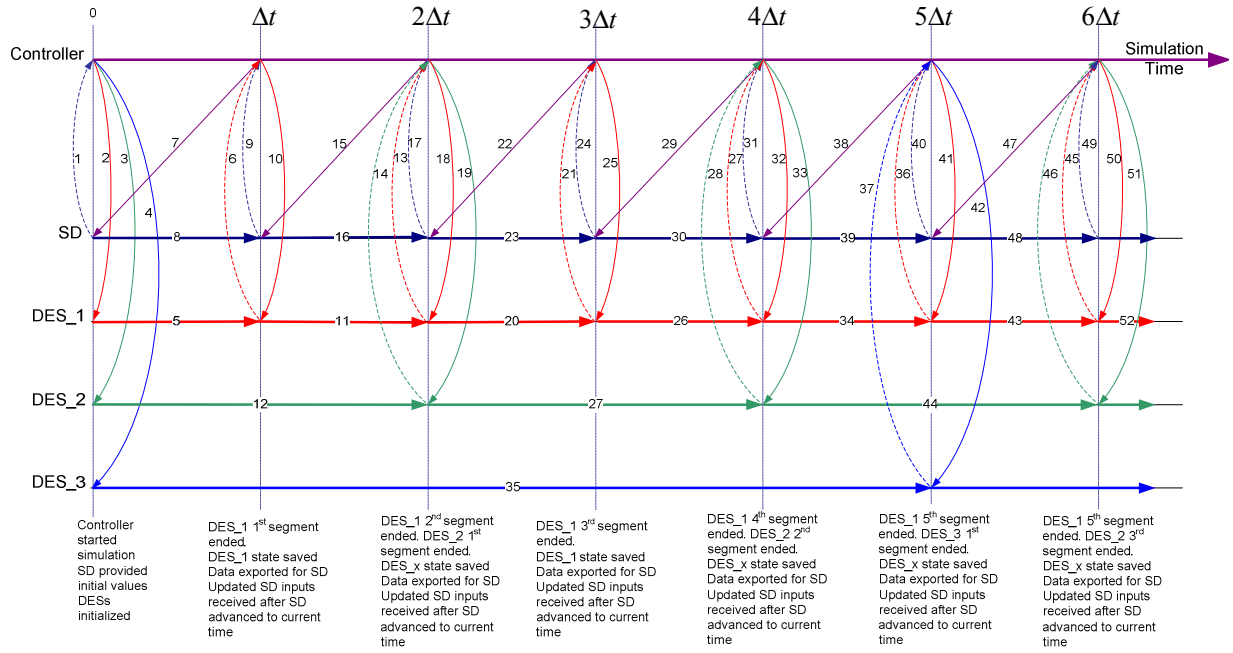

Figure 7 SDDES controller' synchronization action sequence

- Each queue at the start of a segment has the same number of entities it had at the end of the previous segment.
- Each resource at the start of a segment is in the same state it was at when the previous segment ended.
- Entities seizing resource units at the end of a segment continue to seize the same resource units at the start of following segment for the remaining delay times
- Entities that were in processing at the end of the segment and could not finish processing before the segment ends, will continue processing at the start of following segment using the unused portion of the processing time
- All system variables and attributes are at the same values they had when the previous system ended.

We propose the *resumption algorithm* to execute the segmenting of the DES runs. For the basic system highlighted above, the resumption algorithm has been customized as in the listing below, which shows how the resumption algorithm is performed for each two consecutive DES segments; *k* and *k+1*.

1. During segment *k*:
   a. When an entity e seizes a resource unit *r* to start processing, do the following:
      i. Record the entity's identification number
      ii. Record the resource unit's identification number
      iii. Record the processing time; *pt* assigned to the entity at the resource unit
      iv. Record the time of the processing start event; pt_{ps}
   b. Purge all recorded values in Step 1-a if entity e releases resource unit *r* after processing is completed before the end of the segment
   c. When the state of a resource unit *r* that is not seized for processing or is idle, changes to an unavailable state, do the following:
      i. Record the resource unit's identification number
      ii. Record the state of the resource
      iii. Record the time interval; t_{un}, assigned to r for that unavailable state
      iv. Record the time of the event of entering the state; t_{Sun}
   d. Purge all recorded values in Step 1-c if time interval t_{un} has been elapsed before the end of the segment

e. When the state of a resource unit *r* that is being seized by an entity e for processing, changes to an unavailable state, do the following:
   i. Record the entity's identification number
   ii. Record the resource unit's identification number
   iii. Record the state of the resource
   iv. Record the time interval; t_{un}, assigned to r for that unavailable state
   v. Record the time of the event of entering the state; t_{Sun}
   vi. Record the remaining processing time with the entity; $Pt_{e\mathrm{Re}m}$.

f. Purge all recorded values in Step 1-e if time interval t_{un} has been elapsed before the end of the segment

2. At the end of segment *k*:
   a. For each entity undergoing processing, save all recorded values in Steps 1-a through 1-e.
   b. For each resource unit that is not idle, save all recorded values in Steps 1-a through 1-e.
   c. For each queue that has a length greater than zero save the number of waiting entities and their attributes
3. At the start of segment *k+1*:
   a. If saved resumption information in Step 2 are not null then do the following:
      i. Assign each in-processing entity to the appropriate resource unit using the information saved in Step 2, and let the processing time equal to the remaining processing time; $pt-(L-pt_{ps})$ where L is the segment k length
      ii. Set each resource unit in an unavailable state into the same state using the information saved in Step 2-b, and let the time interval for that state equal to $t_{un}-(L-t_{Sun})$ where L is the segment k length
      iii. Insert entities into the queues they were waiting in at the end of segment *k*, using the information saved in Step 2.
   b. Clear all information saved in Step 2
   c. Designate the current segment as segment *k* and return to Step 1.

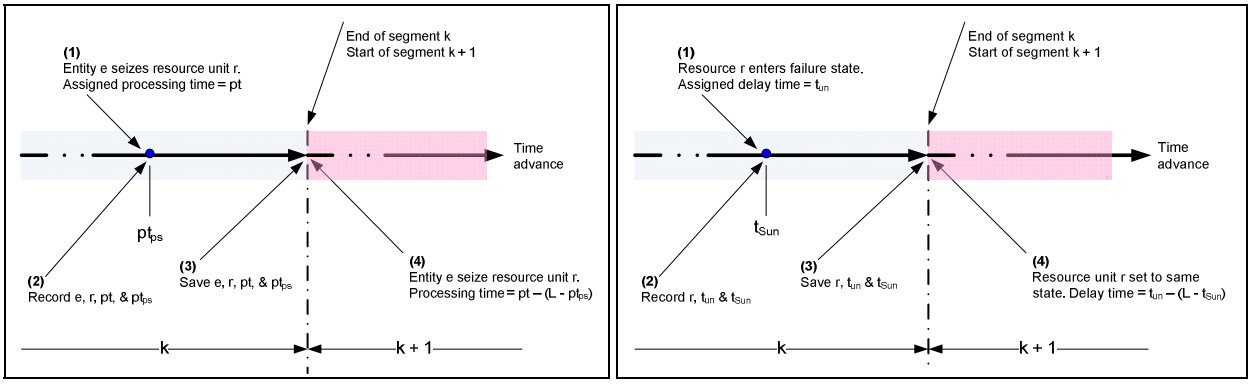

Figure 8 Resuming processing between two consecutive DES segments (Left) and resuming the unavailable state of a resource unit between two consecutive DES segments (Right)

For instance, Figure 8 (Left) describes the recording and saving of the resumption data from segment *k*, and use of that data in segment *k+1*. This is presented for an entity that starts processing in segment k and does not finish processing, and thus needs to continue processing in segment *k+1*. Figure 8 (Right) describes the resumption in segment *k+1* of a resource unit state that was in an unavailable state (e.g. failure state) in the previous segment *k*.

Processing resumption can accommodate the following cases in the DES models:

1. A single entity undergoing processing by a single-unit resource
2. A single entity undergoing processing by one unit of a multiple-unit resource
3. A representative entity (batched entities) undergoing processing by a single-unit resource
4. A representative entity (batched entities) undergoing processing by one unit of a multiple-unit resource

Table 2 summarizes these four cases regarding the information to record and save during and at the end of each segment. Representative entities may represent entities of the same or different attributes. Batched entities are separated before recording their resumption information. Attributes are recorded along with the entities' identification numbers as described in the recoding-saving procedure above.

Table 2 Different cases of processing resumption between consecutive DES run segments

Case	To record at end of segment k	To do at start of segment k+1
Single entity on a single-unit resource	• Entity id number; e • Resource unit id number; r • Processing time; pt • Processing start time; pt_{ps}	• Assign e to r • Processing time = $pt-(L-pt_{ps})$

Single entity on a multiple-unit resource	• For each seized resource unit; i:
	  • Entity id number; e_i
	  • Resource unit id number; r_i
	  • Processing time; $pt_{ei,ri}$
	  • Processing start time $pt_{ps,ei,ri}$
		• For each seized resource unit; i:
		  • Assign e_i to r_i
		  • Processing time = $pt_{ei,ri}-(L-pt_{ps,ei,ri})$

Representative entity on a single-unit resource	• Batch id number; b • Batch size; s_b • For each entity in b, indexed by i: ○ Entity id number; e_{bi} • Resource unit id number; r • Processing time; pt • Processing start time; pt_{ps}	• Given s_b, assign e_{bi} e_{bi} to b • Assign b to r • Processing time = $pt-(L-pt_{ps})$

Representative entity on a multiple-unit resource	• For each seized resource unit; i:
	  • Batch id number; b_i
	  • Batch size; s_{bi}
	  • For each entity in b_i, indexed by j:
	  ○ Entity id number; e_{bij}
	  • Resource unit id number; r_{bi}
	  • Processing time; $pt_{bi,rbi}$
	  • Processing start time; $pt_{ps,bi,rbi}$
		• For each seized resource unit; i:
		  • Given s_{bi}, assign e_{bij} to b_i
		  • Assign b_i to r_{bi}
		  • Processing time = $pt_{bi,rbi}-(L-pt_{ps,bi,rbi})$

4.0 Experimental Analysis

The synchronization approach described above has been implemented for experimentation and a case study involving a real manufacturing enterprise has been conducted and results were logical and meaningful. See Rabelo et al. (2015) for details on that case application, which involved a single SD model and two DES models of two production lines. A Visual Basic (VB) application was developed that communicated with the Arena and Vensim software packages. The VB implementation of the SDDES framework used the ActiveX Data Object (ADO) technology to connect to the simulation engine in Arena to gain access to and control the simulation models. The functions and messaging tools of the Vensim's Dynamic Link Library (DLL) were utilized to communicate with the SD model while establishing a Dynamic Data Exchange (DDE) link to have the software acting as a server/client application for the data exchange operations. To use it as the centralized data repository, the controller used MS Excel object to perform the data preparation operations and appropriately authorize the simulation engines to perform data read/write operations. The use of spreadsheets in implementing SDDES provides a cheap approach to integrate the discrete and continuous simulations in a scalable comprehensive model, making use of existing (legacy) simulations and not requiring learning new simulation skills.

5.0 Summary and Conclusions

We have described a framework to synchronize SD and DES models to develop hybrid simulation of the total enterprise system. A comprehensive SD model of the total enterprise system is synchronized with a number of DES models; built to represent selected sub systems. We have discussed the exchange of data between the SD and DES models, and the necessary data preparation and formatting. The SD and DES models run in parallel and their runs are managed by the SDDES controller. It is needed that the DES simulation runs be divided into segments, where each segment is treated as a typical DES simulation run with replications. We presented the resumption algorithm, which monitors the run segments of each DES model and carry data from a segment to the next in order to maintain the statistical integrity of the DES run and outputs, such that autonomous SD and DES model can run in parallel and share statistically valid DES data.

Combining SD and DES accommodates the coexistence of the continuous and discrete behaviors and the deterministic and stochastic natures in the manufacturing enterprise system. They also accommodate the different perspectives in decision making and views of the system being focused on the dynamic complexity of the business unit or on the detail complexity in managing it. The comprehensive and scalability dimensions in the simulation models of the manufacturing systems were achieved by using a modular structure that regarded autonomous SD and DES models as modules contributing to the same simulation. This can simplify the model building process as several SD or DES models can be built with well defined, relatively narrow scopes to model the various business units and then be interfaced. The addition and deletion of the DES models as the system evolves over time is not restricted. Further, we can utilize existing simulation models without the need to learn new simulation skills.

It is acknowledged that more experimentation is necessary to emphasize the value of the proposed approach. In particular more investigations of the resumption algorism are needed to evaluate its effectiveness for more complex DES model configurations.

References

Brito, T., Botter R., Trevisan, E., A conceptual comparison between discrete and continuous simulation to motivate the hybrid simulation methodology, *The Winter Simulation Conference*, Dec 11-14 Phoenix, AZ, 2011

Fujii, S., Morita, H., Tanaka, T., A basic study on autonomous characterization of square array machining cells for agile manufacturing, *The Winter Simulation Conference,* Dec, 10-13, Orlando, FL, 2000

Fujii, S., Ogita, A., Kidani, Y., Kaihara, T., Synchronization mechanisms for integration of distributed manufacturing systems. *Simulation* vol. **72 No.** 3:187-197, 1999

Godding, G., Sarjoughian, H., Kempf, K., Semiconductor supply network simulation, *The Winter Simulation Conference*, Dec 7-10, New Orleans, LA, 2003

Haler, D., Statecharts: A visual formalism for complex systems. *Science of Computer Programming* Vol. **8:** 231-274, 1987

Helal, M., *A hybrid system dynamics-discrete event simulation approach to simulating the manufacturing enterprise*. PhD Dissertation, University of Central Florida, FL, USA, 2008

Helal, M., Rabelo, L., Sepulveda, J., Jones, A., A methodology for integrating and synchronizing the system dynamics and discrete event simulation paradigms, *International Conference of the System Dynamics Society*, July 29 – Aug 2, Boston, MA, 2007

Law, A., Kelton, W., *Simulation modeling and analysis,* McGraw Hill, USA, 2000

Lee, S., Wysk, R. A resource reconciliation mechanism for a manufacturing federation coordinated using MRP/ERP system. *The Winter Simulation Conference*, Dec 5-8, 2004

Ma, Q., Judd, R., Lipset, R., Distributed manufacturing simulation environment. *Proceedings of the Summer Computer Simulation Conference*, July 15-17, Orlando, FL, 2001

Maler, O., Manna, Z., Pnueli, A., From timed to hybrid systems. In Bakker, J, Huizing, C, Roever, W, Rozenberg, G. (Eds.) *Real-Time: Theory in Practice*. Springer-Verlag, Germany, 1992

Morgan, J., Howick, S., Belton, V., A toolkit of designs for mixing Discrete Event Simulation and System Dynamics. *European Journal of Operational Research* Vol.257 No. 3: 907–918, 2017

Morecroft, J., Robinson, S., Explaining puzzling dynamics: comparing the use of system dynamics and discrete event simulation. *The 23rd International Conference, System Dynamics Society*, July 17-21, MA, USA, 2005

Pastrana, J., Marin, M., Rabelo-Mendizabal, L., Helal, M., Enterprise Scheduling: Hybrid, Feedback, and Hierarchical issues. *The Winter Simulation Conference,* Dec 5 – 8, Baltimore MD, 2010

Borshchev, A., Fillippov, A., From system dynamics and discrete event to practical agent-based modeling, *The 22nd System Dynamics Conference*, July 25-29, Oxford, England, 2004

Rabelo, L., Sarmiento, A., Helal, M., Jones, A., Supply chain and hybrid simulation in the hierarchical enterprise. *International Journal of Computer Integrated Manufacturing* Vol. **28 No.** 5: 488-500, 2015

Rabelo, L., Helal, M., Jones, A., Min, H.,. Enterprise simulation: A hybrid system approach. *International Journal of Computer Integrated Manufacturing* Vol. **18 No** 6: 498-508, 2005

Tako, A., Robinson, S., Comparing discrete event simulation and system dynamics: user's perceptions, *Journal of Operations Research Society* Vol 60: 296-312, 2009

Venkateswarana, J., Son, Y., Hybrid system dynamics discrete event simulation based architecture for hierarchical production planning, *International Journal Production Research*. Vol. 43 No. 20: 4397-4429, 2005

Zeigler, B., *Theory of modeling and simulation,* Wiley, NY,1976

Zeigler, B., Praehofer, P., Kim, T., *Theory of modeling and simulation: Integrating discrete event and continuous complex dynamic systems*. 2nd Ed., Academic Press, USA, 2000

Biography

Magdy Helal is an Assistant Professor of Industrial Engineering and the American University of the Middle East in Kuwait, and Benha University in Egypt. Dr. Helal earned B.Sc. and M.Sc. Mechanical and Production Engineering From Benha University, Egypt; and PhD in Industrial Engineering from The University of Central Florida in the USA. Dr. Helal has accumulated broad academic and practical experiences, and developed solid sets of skills in data analysis, simulation modeling, and performance assessment; developing and applying quantitative mathematical models (TQM-based, operational research, and simulation) of industrial, service, and educational systems.

Luis Rabelo is an Associate Professor of Industrial Engineering And Management Systems Department at the University of Central Florida. Dr. Rabelo received dual degrees in Electrical and Mechanical Engineering from the Technological University of Panama and Master's degrees from the Florida Institute of Technology and the University of Missouri-Rolla. He received a Ph.D. in Engineering Management from the University of Missouri-Rolla in 1990, where he also did Post-Doctoral work in Nuclear Engineering and Artificial Intelligence in 1990-1991. He holds dual MS degrees in Aerospace Systems Engineering & Management from the Massachusetts Institute of Technology (MIT). He has over 240 publications, three international patents, and graduated 14 Doctoral students as advisor. His experience includes Ohio University, BF Goodrich Aerospace, Honeywell Laboratories, the National Institute of Standards and Technology, NASA, and MIT. He has received many awards among them ONE NASA and Fulbright.