The Poisson Simulation Approach to Combined Simulation

Leif Gustafsson
Signals and Systems,
Dept. of Engineering and Sciences
Uppsala University, P.O. Box 534
SE-751 21 Uppsala, Sweden
+4618500061
Leif.Gustafsson@bt.slu.se

Mikael Sternad
Signals and Systems,
Dept. of Engineering and Sciences
Uppsala University, P.O. Box 534
SE-751 21 Uppsala, Sweden
+46184713078
Mikael.Sternad@Signal.uu.se

ABSTRACT

This paper modifies the foundations of combined Discrete Event Simulation (DES) and Continuous Systems Simulation (CSS) by extending the CSS part to the stochastic domain through including discrete (although aggregated) entities. The traditional combined DES/CSS simulation, where the modeller selects between a DES and a CSS description for components of the system under study, lacks a solid theoretical foundation. Model behaviour, results and conclusions may then depend on whether a DES or a CSS description is utilised.

By introducing Poisson Simulation (PoS) as an extension to CSS, several advantages are obtained. First, a theoretically sound foundation is achieved in which aggregation can be performed safely. Second, combined type problems can, in principle, be performed exclusively in PoS, which is often considerably simpler than including DES. The computational complexity is reduced when a large number of discrete entities can be modelled as flows into and out of a relatively small number of state variables. Third, even when it is practical to include DES, the combined DES/PoS approach is more powerful and flexible than a combination of DES and CSS.

Keywords

Aggregation, Modelling, Combined simulation, Continuous System Simulation, Discrete Event Simulation, Hybrid simulation, Poisson Simulation.

1. INTRODUCTION

A dynamic system is composed of so many pieces (e.g. atoms), has so many characteristics and is so complicated that it can never be modelled in all its details. The very essence of modelling is to build a parallel description, called the *model*, which is much simpler but otherwise preserves important characteristics and mechanisms of the system under study. Here we will focus on models where the interest lies at least partly in describing many discrete entities, such as molecules, persons, data packages, vehicles or products on a production line. The system under study can then either be represented as a *micro model*, where the individual entities with their attributes and behavioural logics are individually represented and are identifiable, or as a *macro model*, where the entities are aggregated into a number of state variables (compartments) connected by flows. Then, only the numbers (or mass) of entities of a certain kind or in a certain situation are described.

Micro and macro modelling and simulation are often realised as Discrete Event Simulation (DES) and Continuous System Simulation (CSS), respectively. However, it is sometimes important or practical to utilise both of these approaches in the form of a Combined DES/CSS model[1]. A short and well formulated characterisation of CSS, DES and combined simulation was given by Kreutzer [14]:

"Continuous-system simulation was primarily used in engineering, physics, chemistry, biology, ecology, economics and sociology, where the behaviour of dynamic systems is traditionally captured by differential equations, either because this is considered a 'natural' representation for some process or because of the high level of aggregation at which phenomena are studied. Systems are typically represented at a uniform, fairly low level of detail. Irregularities, expressed by complex interactions do not occur or are abstracted away.

Discrete-event simulation, on the other hand, originated in studies of inventory and manufacturing systems. It has been popular with operations researchers, mathematicians and

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission.

[1] The name 'Combined simulation' is preferred here to 'hybrid simulation', since the latter concept is traditionally used to mean simulation using both analogous and digital computers.

computer scientists. Here, a phenomenon can be studied at any desired level of detail. Arbitrarily complex process interactions and any relevant irregularities may be explicitly modelled."

"Combined simulation tries to integrate both the discrete-event and the continuous-change approach to the study of process behaviour." "Combined models can be driven by a discrete-event simulator, with some components represented as continuous processes. A predominantly continuous model may alternatively be augmented by incorporating a number of special events describing irregularities in its behaviour."

When constructing a model for combined simulation, the modeller has the option of using an aggregated description of subsystems. For example, in the data communication field, a stream of packets can be described either by a DES model that generates individual packets, or by a deterministic macro model, often called a fluid flow approximation. The choice of aggregation level should be governed by the purpose of the study, e.g. whether individual statistics are needed or whether a lumped description is adequate. When aggregation is used, individuality is sacrificed but the model can gain in comprehensibility, simplicity, amount of data required and speed of execution in simulation.

When a (sub)system under study can be modelled either in DES or in CSS fashion, it is essential that the choice of model does not distort the behaviour, results and conclusions of the study. A DES model is in general stochastic and more detailed than a corresponding CSS model, but the results from the two approaches should at least be consistent (contradiction-free).

The fundamental problem with aggregating discrete entities into a deterministic CSS model with real-valued state variables is that both the discreteness of entities and the irregular behaviour of the system under study are lost. These undesired side-effects are harmless only when the number of entities is virtually infinite or (which is the same) the matter is virtually infinitely divisible into a 'fluid'. According to the law of large numbers, the stochastic variations then become negligible. However, for micro- and meso-scale models where the number of entities is not large enough to regard flows as continuous and stochastic variations as negligible, the aggregation into CSS will in general produce undesired phenomena and distort behaviour, results and conclusions.

This paper addresses the inconsistency between Discrete Event Simulation and Continuous System Simulation and demonstrates how a method denoted *Poisson Simulation* (PoS), which is an extension of CSS, removes inconsistencies when entities are aggregated into state variables. Combined simulation can then benefit from the consistency and power of PoS. The paper focuses on the fundamentals of combined modelling and simulation rather than on applied details.

The objectives of this paper are:

- To demonstrate how aggregation of entities should be performed to preserve discreteness and stochasticity, so that the aggregated model is fully consistent with a corresponding non-aggregated model in e.g. DES. This is accomplished by extending the CSS concept to include stochastics and (aggregated) discrete entities. This extended domain is named Poisson Simulation.
- To demonstrate that the PoS domain, alone, can correctly handle many types of combined problems.
- To show that when combined simulation is preferable, the combination of DES and PoS represents a consistent, safe, powerful and practical modelling and simulation approach.

The paper is organised in the following way. In Section 2 and the accompanying Appendix, the problems of aggregation into a CSS model are discussed and examples of inconsistency are given. Poisson simulation is introduced in Section 3 and some examples given in Section 4 illustrate the capacity of PoS to include both discrete and continuous parts in the same model, as well as queues and single actors. In Section 5, the combined DES/PoS approach is discussed. Finally, in Section 6 we briefly summarise and discuss our findings. A number of examples are used to illustrate and underline the messages. These examples are intentionally small and not representative of full-scale combined models.

2. THE PROBLEM OF AGGREGATION

When a system under study, containing a certain number of entities that are changed at irregular instants (events) or with unknown outcomes, is aggregated into a CSS model, the aggregation creates a number of side-effects. First, it has the intrinsic side-effect of grinding the entities into a continuous mass. Second, it replaces the irregularities with deterministic fractions instead of probabilities. CSS aggregation is fine for the case where the numbers *of every type of entities* are very large, but it can otherwise severely distort model behaviour, results and conclusions.

Readers unfamiliar with the side-effects of CSS aggregation are referred to the Appendix, where two illustrative examples are given. The first, Example 7, demonstrates that aggregation into a simple epidemic CSS model may distort results and conclusions beyond all sense. While the population in this example is large, one state variable holds

a small subpopulation. The second example (Example 8) illustrates that trying to restore stochasticity to a deterministic CSS model by adding noise can create a number of artefacts that seriously jeopardise a model study.

These examples demonstrate the lurking danger of aggregation into a deterministic CSS model, and warn against using inappropriate measures to re-create irregularity/stochastic behaviour. What is called for is a conceptual approach that correctly handles aggregation without side-effects.

3. POISSON SIMULATION AS AN EXTENSION OF CONTINUOUS SYSTEM SIMULATION

3.1 Introduction

Poisson Simulation is an extension of CSS designed to handle models where the entities are aggregated as integer numbers of entities in the state variables, and where the stochastics are preserved in a correct way. In the simplest form, events happen randomly and independently. This implies that the number of events during a time interval Δt becomes Poisson-distributed. Therefore, flows into and out of compartments (state variables) during Δt can be generated as integer-valued and Poisson-distributed random outcomes. The method is presented in detail in [4], and is introduced below by a simple example. A number of examples of different applications are given in [7].

Example 1: Radioactive decay in CSS and PoS
The system has one state variable x, representing the number of radioactive atoms, and one outflow rate f, describing the number of radioactive decays per time unit. Therefore, f is proportional to the state value x with a proportionality constant a that describes the proportion of x that is expected to decay during a time unit. A deterministic CSS model (using Euler integration) is then:

```
      x = x0                [Initialisation]
      Goto *
AGAIN: x = x - Δt·f
  *   f = a·x
      time = time + Δt
      if time ≤ Tend then Goto AGAIN .
```

To model the irregularity of the decay process in a realistic way by a stochastic model, we reason thus: The expected outflow still has an intensity of $f=a \cdot x$ per time unit or $\Delta t \cdot f = \Delta t \cdot a \cdot x$ during the time interval Δt. Since the decay instant of an atom is independent of the timing of decays of other atoms, the number of events during the time interval Δt should be Poisson-distributed with the parameter $\Delta t \cdot a \cdot x$. Thus, the reduction of x during Δt has a Poisson-distributed variation denoted: $Po[\Delta t \cdot a \cdot x]$. Therefore, the model is reformulated as:

```
      x = x0
      Goto *
AGAIN: x = x - Δt· f
  *   f = Po[Δt·a·x]/Δt    [The decays are now stochastic]
      time = time +Δt
      if time ≤ Tend then Goto AGAIN .
```

This model correctly describes the stochastic properties with regard to the number of atoms in the state variable x and its variability in repeated trials, without producing any artefacts. When initialised by an integer, $x(t)$ remains integer-valued for all t. This is in contrast to the erroneous result in Example 8 in the Appendix, where an alternative model is obtained by adding noise to the CSS model. ■

3.2 Poisson Simulation

A Continuous System Simulation model represents a deterministic and dynamic system by a set of ordinary differential (and algebraic) equations. Each differential equation can be expressed as:

$$dx_i(t)/dt = f_{i1}(\mathbf{x},t) - f_{i2}(\mathbf{x},t) , \quad i=1,2,...k , \qquad (1)$$

where f_{i1} is the sum of inflows and f_{i2} is the sum of outflows to the state variable x_i during the time interval dt and $\mathbf{x}$ is the vector of state variables $\mathbf{x}(t) = (x_1(t), x_2(t), \ldots x_k(t))^T$. These dynamic equations, together with initial values $x_i(0) = x_{i0}$, completely determine the behaviour of $\mathbf{x}(t)$ over time. In CSS language, this dynamic equation can be written as a difference equation using Euler's approximation:

$$x_i(t+\Delta t) = x_i(t) + \Delta t \cdot f_{i1}(\mathbf{x},t) - \Delta t \cdot f_{i2}(\mathbf{x},t) , \quad i=1,2,...k. \qquad (2)$$

Assume that a system of interest contains discrete entities affected by irregularly occurring events. For the purpose of the study, a *conceptual model* that only contains relevant information is specified, see the upper part of Figure 2. In this conceptual model, discrete entities, their behaviour and their attributes are described at an appropriate level of detail and irregularities are specified by relevant statistical distributions. Of special interest is the intensity of events over time. We now wish to construct an *aggregated* state variable model, similar to (2), where the state variables preserve integer numbers of entities at all times and where the probabilistically specified transition intensities of the conceptual model are preserved. The following conditions must then be fulfilled:

1) The state variables $x_i(t)$ must be initialised with *integer numbers of entities*, whose numbers can change only because of inflows and outflows that transfer integer numbers of entities.

2) The probabilistic properties are preserved by letting the number of entities transferred in each flow be generated as random integer numbers that are consistent with the conceptual model.

When entities represented by a flow occur randomly with an intensity λ, the number of events during a time interval Δt will be Poisson-distributed according to $Po[\Delta t \cdot \lambda]$. The intensity λ may be a function of the state vector $\boldsymbol{x}$ and of time but it is approximated as being constant during the sufficiently short time interval Δt, just as in CSS simulation. The approximation of a stepwise constant intensity makes it possible to obtain a simulation model that can be updated in discrete time, with time-step Δt. The state equations then take the form:

$$x_i(t+\Delta t)=x_i(t) + Po[\Delta t \cdot f_{i1}(\boldsymbol{x},t)] - Po[\Delta t \cdot f_{i2}(\boldsymbol{x},t)],\ i=1,...k \quad (3)$$

and one example is graphically illustrated by Figure 1.

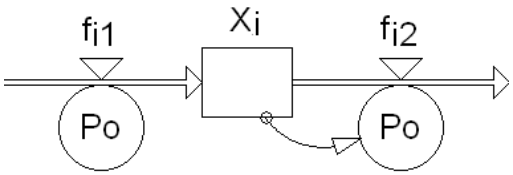

Figure 1. Forrester diagram [2,16] of (a part of) a PoS model. The box x_i contains the current integer number of entities. This can only change by arriving entities f_{i1} or by departing entities f_{i2}. The single-line arrow from x_i to f_{i2} shows that f_{i2} is dependent on the current number of entities in x_i.

Each flow f_{ij} is thus represented by a time sequence of (dynamically varying and time varying) Poisson processes. The only theoretical requirement of independence is that the events behind each flow are mutually independent within each time-step Δt. (In a DES model, corresponding events would be generated one event at a time, with exponentially distributed inter-event times.) The Poisson distribution, with state- and time-dependent intensity, is a flexible mechanism for generating irregularly occurring numbers of events with desired statistical properties. In each particular case, the functions $f_{ij}(\boldsymbol{x},t)$ are obtained, or approximated, from the properties of the original conceptual model.

Here we present the PoS models in Euler form, but it is possible to use other single-step algorithms such as higher order Runge-Kutta algorithms to integrate the equations. The argument is then estimated by a R-K algorithm before the Poisson random number generator is called, see [3].

DES and PoS models also differ in many aspects other than time handling, see [8,10]. In DES various statistical distributions are used as needed to model sojourn times, choices, outcomes of an operation, etc. In PoS (as well as in CSS) the state variable (compartment) is by nature 'an exponential building block'. Therefore, sojourn times other than exponential are obtained by a structure of compartments in series and/or parallel, with the *Po[]* mechanism handling the internal flows of the structure. Also choices and outcomes of an operation are handled with the *Po[]* mechanism, together with appropriate logical functions.

In most aspects PoS is structured and implemented in the same way as a corresponding CSS model. Provided that the PoS model is correctly built, it will produce aggregated results fully consistent with those produced with the DES approach, when both originate from the same conceptual model [8]. For instance the SIR model in Example 7 in the Appendix will produce the same probability density functions for simulation outcomes when realised by either DES or PoS.

Note that when the number of entities $n \to \infty$, then $Po[\Delta t \cdot f(x,t)] \to \Delta t \cdot f(x,t)$, so the PoS model approaches a deterministic CSS model. Thus the *embedded* CSS model (the fluid flow approximation) is obtained by removing the *Po[ ]* parts, but keeping the arguments, of the PoS model. Note also that sub-models describing continuous matter (CSS) and discrete entities (PoS) can be combined and linked within one PoS model, as illustrated by Example 2 in Section 4.

Figure 2 shows how a system under study can be represented as a DES model or aggregated into a PoS model in a consistent way – in both cases preserving discreteness and stochasticity. If quantities can be regarded as continuous, the DES approach can be combined with CSS equations. Using the PoS approach, CSS equations are handled within the PoS concept.

Poisson Simulation can be performed in any CSS language provided that a Poisson-distributed random number generator is accessible, which is usually the case. Alternatively, the complete model can be written in any programming language using a Poisson generator that requires only a few lines of additional code [4,17].

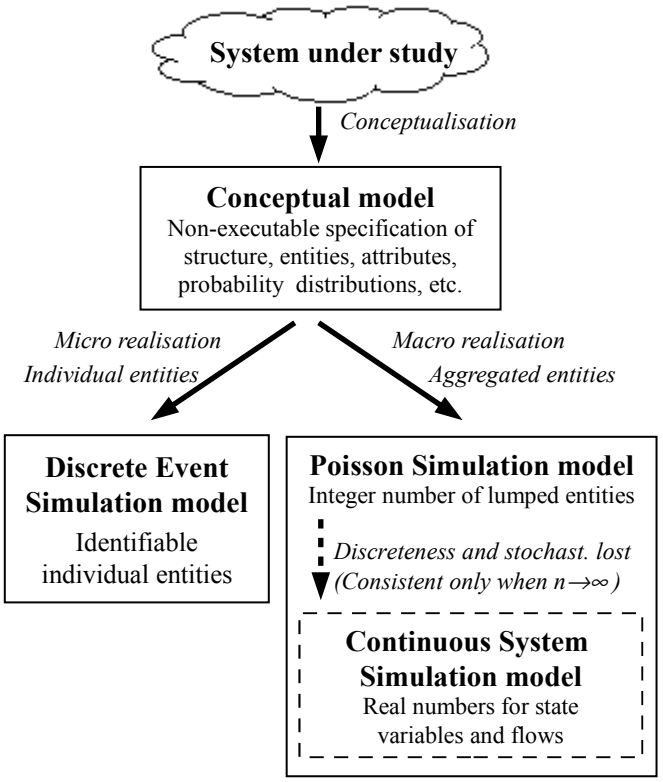

Figure 2. The relationships between System at study, Conceptual model, Discrete Event Simulation, Poisson Simulation and Continuous System Simulation.

4. POISSON SIMULATION AS AN ALTERNATIVE TO COMBINED MODELLING

To demonstrate the capacity and versatility of Poisson Simulation to handle different types of processes, four examples are sketched in this section, followed by a discussion of data collection and statistics within the single replication and over multiple replications.

4.1 Modelling Capacity

PoS not only offers a way to correctly handle aggregation, but also constitutes a flexible and execution-efficient aggregated way to model problems that would otherwise have required a combined DES/CSS approach to preserve both the discreteness and the stochastics. An ecological model of prey-predator type, where discrete predators feed on a continuously varying biomass, illustrates this.

Example 2: A combined discrete and continuous Volterra model

The classical Lotka-Volterra equations describe a prey-predator system for two species X and Y by differential equations [15,18]. The prey population breeds at a rate proportional to its size x. The prey population is reduced because of encounters with predators, which is proportional to $x \cdot y$. There is competition among prey proportional to x^2. The encounters with prey give the predators the energy to breed, so they increase in proportion to $x \cdot y$. Finally, the predators die in proportion to their numbers y. The classical Lotka-Volterra model therefore has the form:

$$\begin{aligned} dx/dt &= ax - bxy - kx^2 \\ dy/dt &= cxy - dy\ , \end{aligned} \qquad (4)$$

where a and c are fertility constants, b and d mortality constants, and k is a proportionality constant for competition. In our setting we now let the prey be a continuous amount of biomass x (e.g. of grass), while the predators y are the number of entities (e.g. deer). The combined discrete-continuous model can then be rewritten as:

$$\begin{cases} x(t+\Delta t) = x(t)+\Delta t \cdot f_1 - \Delta t \cdot f_2 - \Delta t \cdot f_3 \\ \Delta t \cdot f_1 = \Delta t \cdot a \cdot x(t) \\ \Delta t \cdot f_2 = \Delta t \cdot b \cdot x(t) \cdot y(t) \\ \Delta t \cdot f_3 = \Delta t \cdot k \cdot x^2(t) \\ y(t+\Delta t) = y(t)+ \Delta t \cdot f_4 - \Delta t \cdot f_5 \\ \Delta t \cdot f_4 = Po[\Delta t \cdot c \cdot x(t) \cdot y(t)] \\ \Delta t \cdot f_5 = Po[\Delta t \cdot d \cdot y(t)]. \end{cases} \qquad (5)$$

The behaviours of the continuous and the combined discrete-continuous models, with $a=0.2$, $b=0.005$, $c=0.005$, $d=0.3$ and $k=0.001$, are exemplified in Figure 3.

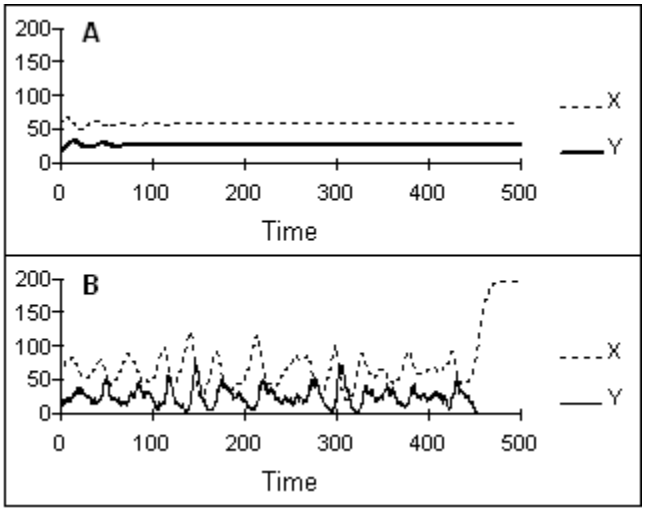

Figure 3. A single replication of the prey-predator model. A) with continuous prey (x) and predators (y) using the CSS model (4). B) with continuous prey (x) and discrete predators (y) using the combined model (5). At around 450 time units, the predators become extinct so the prey increases logistically.

Comparing the deterministic CSS model (4) with the PoS model (5) reveals that the deterministic model rapidly reaches a steady state point for both species without further variations. Furthermore, phenomena such as extinction cannot occur in (4).

Estimating various statistics from this probabilistic model requires repeated simulations. To give an idea of the execution speed of PoS, the execution time for *10 000* replications of the Volterra model (5) over a time span of 500 time units, using $\Delta t=0.25$ time units, was measured. For an ordinary *3* GHz PC it took *74.9* seconds. As a comparison, the same number of replications of (4) (which of course would not be needed for a deterministic model) took *31.9* seconds. Thus, in this case the execution time for the stochastic model is *2.3* times longer than for the deterministic one due to random number generator calls. ■

The following example demonstrates that queues, resources like servers (that can be busy or idle), waiting numbers and waiting times, etc. that are central concepts in DES can be smoothly modelled in PoS.

Example 3: *Queues in PoS*

Implementation of queuing systems in Poisson Simulation is straightforward. Here this is demonstrated for the *M/M/1*, *M/M/c* and *M/M/∞* cases, where M/M/* stands for exponentially distributed inter-arrival and service times, and the third symbol represents the number of servers, see [13]. The input (arrivals) is then a Poisson process, with intensity λ, of the form: $In = Po[\Delta t \cdot \lambda]$.

Let q denote the actual total number of *queuing and served* entities in the state variable Q. Let us introduce the following model for the output (departures after a mean service time of $1/\mu$) and denote it the *kernel* of the queue dynamic model:

$Out = Po[\Delta t \cdot \mu \cdot MIN(1,q)]$ *for* the *M/M/1* case;

$Out = Po[\Delta t \cdot \mu \cdot MIN(c,q)]$ for the *M/M/c* case; and

$Out = Po[\Delta t \cdot \mu \cdot q]$ for the *M/M/∞* case.

Here, *MIN(c,q)* means that at most c entities can be served simultaneously, and if $q<c$ only q entities are served. However, there is a complication. The number of entities q in the state variable must not become negative. CSS and PoS have no automatic mechanism protecting from negative numbers. A guard mechanism should be included to prevent the output from draining the state variable of more than its actual content. This can be accomplished by including another *MIN* function comparing the kernel introduced above to the actual content. Thus $Out = MIN\{kernel, q\}$.

The complete algorithm of a queuing system then is:

$$
\begin{cases}
q(t+\Delta t) = q(t) + In - Out \\
In = Po[\Delta t \cdot \lambda] \\
Out = MIN\{Po[\Delta t \cdot \mu], q\} & [M/M/1 \text{ case}] \\
\quad \text{or} \\
Out = MIN\{Po[\Delta t \cdot \mu \cdot MIN(c,q)], q\} & [M/M/c \text{ case}] \\
\quad \text{or} \\
Out = MIN\{Po[\Delta t \cdot \mu \cdot q], q\} & [M/M/\infty \text{ case}]
\end{cases}
$$

For more information on queuing models in PoS, see [5]. Figure 4 shows an M/M/c queue with a number of statistical devices that are discussed in Section 4.2.

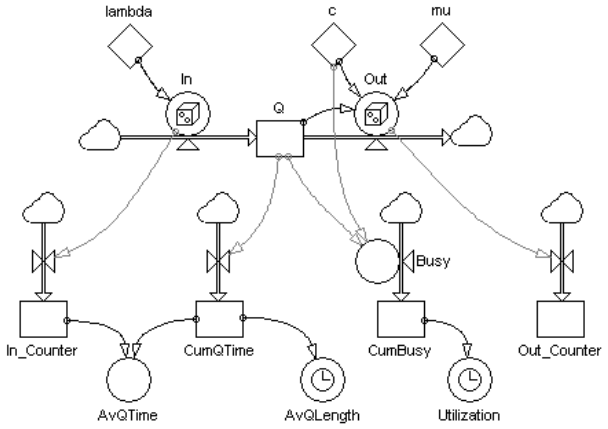

Figure 4. An M/M/c queuing system presented as a Forrester diagram [2,16]. A die in a symbol means that a random number (here Poisson) from some specified distribution is drawn for each time-step. A clock in a symbol means that time is involved in the calculations. Here, *Q* is a state variable holding the total number of queuing and currently served entities. The arrival rate is λ (*lambda*), and the service rate (per server) is μ (*mu*). Of course λ, μ and also the number of servers *c* can be varied during the simulation. The lower part of the figure shows devices for counting arrivals and departures, and calculating queue-time, queue-length and server utilisation, as discussed in Section 4.2. ■

Because PoS can handle any number of discrete entities, it can of course also handle single entities separately. The next example shows how a single entity (actor) can also be modelled in PoS when needed.

Example 4: A single actor in PoS

A single (or a few) entities (physicians, fire trucks, machines, etc.) can be in different states in the same meaning as an actor in a DES model. As an example, see the sketch in

Figure 5, which can be part of a larger PoS model. When the change to a new state is affected by some external event, this is coded within the proper flow equation. When a sojourn time distribution of a stage is required, a number of state variables in series or parallel can be used to generate the appropriate distribution. When there is a random choice between branches (e.g. F23 and F24 in Figure 5), an information link between the flow equations involved is needed (not shown) to prevent an entity from taking more than one branch.

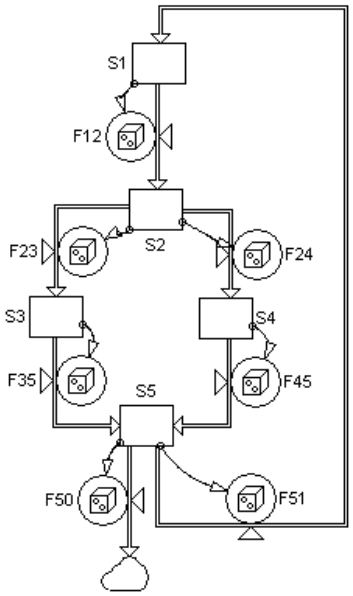

Figure 5. Example of a PoS model where a single entity or a few entities can be in different states *S1, S2, S3, S4, S5* and *S0* (*S0* is a cloud symbol representing that the entity has left). ■

The *Po[ ]*-mechanism handles the randomness describing the numbers of events over time intervals, but various statistical distributions can be used for other purposes.

Example 5: Mixed stochastic processes

Events may happen irregularly. Furthermore, the *consequences* of the events can be described by random distributions other than Poisson. Assume that an insurance company intends to study the variations in the annual number of accidents and in total costs of these accidents during the year. The expected intensity of accidents varies strongly over the year as described by a known intensity $\Delta t\lambda(t)$.

The cost per accident is modelled as exponentially distributed with average $\mu(t)$; most accidents are not expensive but a few are. Each randomly occurring event then has the real-valued attribute: cost, which is obtained from a random number generator named *Expo[*$\mu(t)$*]*. We express the corresponding PoS model in pseudo-code as:

```
Time=0;  CumAccidents=0;  CumCosts=0
 While Time < OneYear do
   N = Po[Δt·λ(t)]        [No. of accidents during Δt]
   CostsInDT = 0
   For i=1 to N do CostsInDT = CostsInDT + Expo[μ(t)]
   CumAccidents = CumAccidents + N
   CumCosts = CumCosts + CostsInDT
   Time = Time + Δt
End
```

Remark: The innermost loop generates an N-Erlang[μ] distributed variable, and can be substituted by a random number generator for that distribution, if available. ■

4.2 Internal and External Statistics

Two kinds of statistics are needed in stochastic simulation, irrespective of whether it is implemented in DES, PoS or otherwise. First, there are *internal* data collection and statistics related to what happens *within a replication* (pdf, average, variations, min, max, etc.). Second, *external* data collection and statistics *from many replications* of a simulation model are needed to explore various behaviours of a stochastic model.

4.2.1 Internal data collection and statistics

Below we exemplify how data and statistics can be handled swiftly within a replication by counters, tallies and other statistical devices implemented as part of the PoS model.

Example 6 (Example 3, continued): *Internal data collection and statistics in PoS*

Counters, tallies, histograms, etc. for averages, standard variations, pdfs, min, max, etc., to be updated for each time-step, are easily constructed within the PoS model. Below the queuing model in Figure 4, devices for internal data collection and statistics within a replication are shown. The flows of arrivals and departures are accumulated over time in In_Counter and Out_Counter, respectively. The queuing customers are also integrated over time to CumQTime. By dividing CumQTime by In_Counter, the average time in queue, AvQTime, is obtained. By dividing CumQTime by time, the average queue length, AvQLength, is obtained. Server utilisation is calculated from the fraction of time the *c* servers are busy. ■

4.2.2 External data collection and statistics

Because PoS models are stochastic, large numbers of replications are needed to estimate the pdfs, confidence intervals, correlations, etc. It is therefore convenient to use a supervisory program that can carry out a specified number of replications, collect specified results and present statistical estimates. Such programs exist for Powersim [6,16] and Matlab [11,19].

5. COMBINED DES/PoS SIMULATION

The aggregation into a PoS model has consequences for size and execution speed. When a large number of entities are aggregated into a small number of state variables, the model will be comprehensive. The size and computational complexity of this model will furthermore not grow with the number of entities in the system, while a DES model grows proportionally with the number of entities (if e.g. they are modelled as objects). Furthermore, in PoS many entities in many flows may be transferred for each time-step, while in DES one event is handled per 'time cycle', which makes the PoS approach significantly faster than DES for large populations.

However, although it is *in principle* possible to model individual entities, queues, behavioural logics, etc. in PoS, it is not always comprehensive, practical or fast. Especially when a system under study contains a heterogeneous population of entities or many entity attributes, the number of state variables may become very large, so DES usually is a better option than PoS. Furthermore, in DES any sojourn time distribution (even empirical) may be used, while such a distribution has to be approximated with a structure of state variables in series and/or parallel in PoS. The modeller should therefore carefully consider when or what parts of a system under study should be modelled in DES or aggregated into PoS.

6. DISCUSSION AND CONCLUSIONS

Poisson Simulation (PoS) constitutes a straightforward way to perform an aggregation of discrete entities without producing undesired side-effects of grinding down the discrete entities into a continuous mass and thereby also eliminating the original variations because of numbers. Only when the number of entities is so large that flows can be considered continuous should the modeller consider further simplifying the PoS model into a (deterministic) CSS setting. We have also seen that PoS alone, in principle, can handle most combined problems. When it is still practical to use a combined approach, the DES/PoS approach is a more structured, versatile and powerful approach than DES/CSS.

Because of limited space, only some illustrative aspects of PoS have been presented. We refer to the literature for more details. In [4] the fundamental rules of PoS are presented. That paper describes e.g. when multiple flows may be superimposed or how to control correlation when flows are dependent. A number of PoS examples are presented in [7]. The full consistency between DES and PoS (despite of e.g. different concepts, building blocks, level of aggregation, time handling, etc.) is demonstrated in [8] and [10]. The modelling of queuing problems is treated in [5]. In [3], stochastic integration algorithms other than Euler are given. Finally, devices for external data collection and statistics are presented in [6] and [11].

In conclusion, the combined DES/PoS concept has a number of advantages compared with the combined DES/CSS concept and provides a consistent, complete and powerful way to combine modelling and simulation.

APPENDIX. UNDESIRED CONSEQUENCES OF CSS AGGREGATION

A.1 Very Different Results from DES and CSS Even For a Large Population

As an example of how CSS can distort results and conclusions even when the studied population is large, consider the following example of an epidemic model.

Example 7: *SIR (Susceptible-Infectious-Recovered) model*

SIR models constitute a class of standard lumped models used for describing the effect of infectious diseases on populations of interacting individuals [1,12]. Here we compare the results when a simple SIR model is realised as a DES and a CSS model. The SIR model to be studied assumes three types of individuals in the population, the Susceptible population of size S, the Infectious of size I, and the persons who Recover and are permanently immune of size R. In a conceptual model, the following transition probability scheme is assumed for the change of subpopulation size at each time-step:

Type of transition:	**Infinitesimal probability:**
$S \rightarrow S\text{-}1,\ I \rightarrow I+1$	$p \cdot S \cdot I \cdot dt$
$I \rightarrow I\text{-}1,\ R \rightarrow R+1$	$(I/T) \cdot dt$.

This model is studied here for a relatively large population of *1001* persons, initially divided into: $S(0)=1000$, $I(0)=1$ and $R(0)=0$ persons. The risk parameter p is set to $p=0.0003$ per individual and time unit and the expected

sojourn time as Infectious is *T=4* time units (exponentially distributed). The quantity to be studied is the number of Susceptible individuals that will eventually become infected, *S(0)-S(End)*.

In a DES setting, the results are a probability density function (pdf) of the total number of infected individuals after the epidemic has run its course. Figure 6 shows the results from *10 000* replications of the DES model.

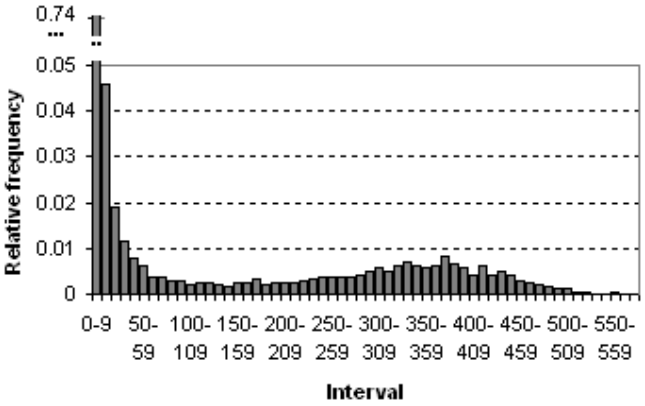

Figure 6. The pdf of the number of susceptible individuals becoming infected during the epidemic [S(0)-S(End)] as calculated from *10 000* replications of the DES model. The bar intervals are *0-9, 10-19, ..., 570-579*. The first bar (*0-9*) is cut and has value *0.7388*.

Now, consider a CSS model of the form:

$$S(t+\Delta t) = S(t) - \Delta t \cdot f_1$$
$$I(t+\Delta t) = I(t) + \Delta t \cdot f_1 - \Delta t \cdot f_2$$
$$R(t+\Delta t) = R(t) + \Delta t \cdot f_2$$
$$f_1 = p \cdot S(t) \cdot I(t)$$
$$f_2 = I(t)/T \, .$$

Using this CSS model causes loss of shadings of the problem by producing a categorical answer that *318.5* individuals of the Susceptible population will become infected. This expected number of *318.5* susceptible persons getting the disease is very far from the DES prediction of *54.6* (*52.2-57.0* for *95%* C.I.). Furthermore, the correct conclusion that in *74%* of the cases less than *10* Susceptibles will become ill is not reflected at all. ■

The reason for CSS producing such a poor estimate of the expected value of infected individuals (close to *600%* too large) is that not all subpopulations were large all the time. The epidemic starts with a single infected individual!

The model considered above was bilinear, but a linear deterministic model will also lose the range of outcomes described by a pdf and it may produce wrong estimates when the number of entities is small, see [9].

A.2 Restoring Stochastic Variations in CSS

CSS models are usually deterministic, but a CSS modelling language usually provides random number generators for different distributions. However, trying to restore stochasticity to a deterministic CSS model by adding or multiplying back noise of a certain distribution is not a good idea, since the stochastics are an intrinsic part of the changing process originally defined by events.

Example 8 (Example 1, continued): *Radioactive decay*

Radioactive decay, where $x(t)$ represents the number of non-decayed radioactive atoms with a decay fraction a per time unit, can be implemented in a deterministic CSS model as $x(t+\Delta t) = x(t) - \Delta t \cdot a \cdot x(t)$, see Example 1. If we try to restore stochasticity by adding random perturbations as:

$$x(t+\Delta t) = x(t) - \Delta t \cdot a \cdot x(t) + b \cdot e(t) \, ,$$

where $e(t)$ is a zero-mean discrete-time white noise, here assumed Gaussian, we would obtain results as in Figure 7.

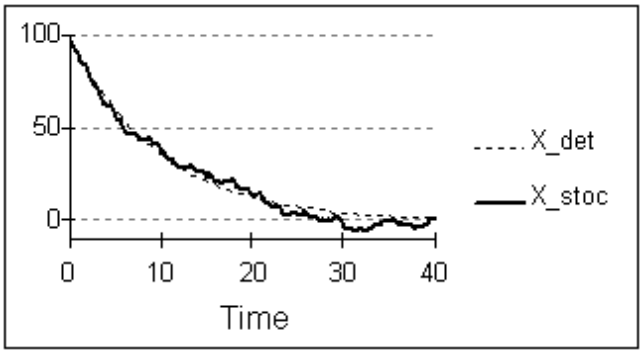

Figure 7. Simply adding noise to a deterministic CCS model may create a number of artefacts.

As illustrated by Figure 7, the stochastic CSS model would then produce a number of unfeasible phenomena such as: 1) Non-integer numbers of (non-decayed) atoms; 2) stochastic variations unrelated to the remaining number of atoms; 3) sudden increases in the number of atoms; 4) continued variations around the equilibrium state even when there are no atoms left; 5) a negative number of atoms may occur. Furthermore, 6) the ensemble of trajectories from many replications not having the correct distribution; and 7) the model behaviour will strongly depend on the time-step used.

The artefacts in this example may seem innocent and obvious, but when part of a larger model, they may generate severe consequences. Variations without appropriate reasons may excite other parts of the model. Negative numbers of entities may trigger various phenomena. ■

The reason for failure is that stochasticity is an intrinsic part of irregular events, and has to be modelled correctly. This is what Poisson Simulation does and, as a consequence, artefacts as exemplified above are eliminated.

REFERENCES

[1] Bartlett M.S. 1960. Stochastic Population Models in Ecology and Epidemiology. John Wiley & sons Inc., New York.

[2] Forrester J.W. 1961. Industrial Dynamics. Cambridge, MIT Press, MA.

[3] Gillespie, D.T. 2001. Approximate accelerated stochastic simulation of chemically reacting systems. *J. Chem. Phys*. 115:1716-1733.

[4] Gustafsson L. 2000. Poisson Simulation – A method for generating stochastic variations in Continuous System Simulation. Simulation 74/5:264-274.

[5] Gustafsson L. 2003. Poisson Simulation as an extension of Continuous System Simulation for the modeling of queuing systems. Simulation 79/9:528-541.

[6] Gustafsson L. 2004. Tools for Statistical Handling of Poisson Simulation: Documentation of StocRes and ParmEst, Dept. of Biometry and Engineering, The Swedish University of Agricultural Sciences.

[7] Gustafsson L. 2005. Studying dynamic and stochastic systems using Poisson Simulation. In: Liljenström H. and Svedin U. (Eds.) Micro – Meso – Macro: Addressing Complex Systems Couplings. World Scientific Publishing Company. Singapore, pp. 131-170.

[8] Gustafsson L. and Sternad M. 2007. Bringing consistency to simulation of population models – Poisson Simulation as a bridge between micro and macro simulation. Mathematical Bioscience, 209:361-385.

[9] Gustafsson L. 2009. Consistency or Not between Deterministic and Stochastic Population Models? Technical Report. Signals and Systems, Uppsala University, 1-19. Available at: www.signal.uu.se/Research/simulation.html

[10] Gustafsson L. and Sternad M. 2009. Consistent Micro, Macro and State-Based Population Modelling. Submitted.

[11] Hedqvist T. 2004. Wanda for MATLAB. Methods and Tools for Statistical Handling of Poisson Simulation, Master of Engineering Thesis, Uppsala University, ISSN: 1401-5749, UPTEC IT0422, Uppsala, Sweden. Available at: www.signal.uu.se/Research/simulation.html

[12] Kermack W.O. and McKendrick A.G. 1927. Contributions to the mathematical theory of epidemics. Proc. Royal Soc. A 115:700-721.

[13] Kleinrock L. 1975. Queueing Systems. Vol. 1: Theory. John Wiley & Sons Inc.

[14] Kreutzer W. 1986. System Simulation: Programming Styles and Languages. Addison-Wesley, Sydney.

[15] Luenberger D.G. 1979. Introduction to Dynamic Systems. Theory, Models and Applications. John Wiley & Sons, Inc. New York.

[16] Powersim Reference Manual. 1996. Powersim Corporation, 1175 Herndon Parkway, suite 600, Herndon, VA 22170, Powersim USA, Powersim Press.

[17] Press W.H., Flannery B.P., Teukolsky S.A. and Vetterling W.T. 1989. Numerical Recipes in Pascal – The Art of Scientific Computing. Cambridge University Press, Cambridge, UK.

[18] Volterra V. 1926. Variazioni e fluttuazioni del numero d'individui in specie animali conviventi. Memoire della R. Accademia Nazionale dei Lincei, anno CCCCXXIII, II.

[19] Using MATLAB. 2002. The MathWorks, Inc. Natick, MA.