Scientific Research Publishing

Open Journal of Modelling and Simulation, 2022, 10, 219-253
https://www.scirp.org/journal/ojmsi
ISSN Online: 2327-4026
ISSN Print: 2327-4018

StochSD: A Full Potential CSS Language for Dynamic and Stochastic Modelling, Simulation and Statistical Analysis

Leif Gustafsson[1], Erik Gustafsson[2], Magnus Gustafsson[2]

[1]Signals and Systems Group, Department of Electrical Engineering, Uppsala University, Uppsala, Sweden
[2]Uppsala University, Uppsala, Sweden
Email: leif.gunnar.gustafsson@gmail.com

How to cite this paper: Gustafsson, L., Gustafsson, E. and Gustafsson, M. (2022) StochSD: A Full Potential CSS Language for Dynamic and Stochastic Modelling, Simulation and Statistical Analysis. *Open Journal of Modelling and Simulation*, **10**, 219-253.
https://doi.org/10.4236/ojmsi.2022.102012

Received: February 18, 2022
Accepted: April 26, 2022
Published: April 29, 2022

Abstract

It is vital that a well-defined conceptual model can be realized by a macro-model (e.g., a Continuous System Simulation (CSS) model) or a micro-model (e.g., an Agent-Based model or Discrete Event Simulation model) and still produce mutually consistent results. The Full Potential CSS concept provides the rules so that the results from macro-modelling become fully consistent with those from micro-modelling. This paper focuses on the simulation language StochSD (Stochastic System Dynamics), which is an extension of classical Continuous System Simulation that implements the Full Potential CSS concept. Thus, in addition to modelling and simulating continuous flows between compartments represented by "real" numbers, it can also handle transitions of discrete entities by integer numbers, enabling combined models to be constructed in a straight-forward way. However, transition events of discrete entities (e.g., arrivals, accidents, deaths) usually happen irregularly over time, so stochasticity often plays a crucial role in their modelling. Therefore, StochSD contains powerful random functions to model uncertainties of different kinds, together with devices to collect statistics during a simulation or from multiple replications of the same stochastic model. Also, tools for sensitivity analysis, optimisation and statistical analysis are included. In particular, StochSD includes features for stochastic modelling, post-analysis of multiple simulations, and presentation of the results in statistical form. In addition to making StochSD a Full Potential CSS language, a second purpose is to provide an open-source package intended for small and middle-sized models in education, self-studies and research. To make StochSD and its philosophy easy to comprehend and use, it is based on the System Dynamics approach, where a system is described in terms of stocks and flows. StochSD is available for Windows, macOS and Linux. On the StochSD homepage, there is exten-

sive material for a course in Modelling and Simulation in form of PowerPoint lectures and laboratory exercises.

Keywords

Stochastic Simulation, Combined Simulation, Consistent Modelling, Open-Source, System Dynamics

1. Introduction

To model and simulate a system under study, you can choose between creating *a micro model* [e.g., an Agent-Based Model (ABM) or a Discrete Event Simulation (DES) model] by describing *each entity of interest*, or a *macro model* [e.g., a Continuous System Simulation (CSS) model] by only describing *the number* of different types of entities or *the number* of entities in defined stages (e.g., Susceptible, Sick, Immune).

However, a major problem in simulation has been that a micro model and a macro model of the same well-defined conceptual model of a system under study often produce mutually inconsistent (*i.e.*, contradictory) results. Based on a series of papers that theoretically solved this problem, the authors of this paper have constructed a CSS program package that contains the facilities to create and simulate CSS models that are fully consistent with a well-defined conceptual model.

1.1. Research Background

The consistency problem was investigated in a number of studies to identify why micro and macro realisations of the same conceptual model often produce inconsistent results. A cornerstone was here the method to transfer a number of discrete entities between stocks (compartments) during a time step in CSS, published by L.G. in 2000 [1]. (This idea is the same as used in Gillespie's τ-leap (or tau-leap) method published the following year [2].)

Then the fundamental problem of queuing, which is essential in micro modelling, could be addressed so that queues could be implemented into a stochastic CSS model [3]. In papers [4] [5] [6] the consistency problem was further investigated with focus on different types of stochasticities, stage-to-compartment expansion in order to preserve the sojourn time distribution in the stage, and the inclusion of attributes into a CSS model.

Finally, it became possible to transform a well-defined conceptual model into an ABM, then to a DES model, and further into a CSS model by a series of transformations without creating inconsistencies. This was published by Gustafsson and Sternad in OJMSi in 2016 [7].

The following year, the conditions required for a CSS model to preserve the properties of a conceptual model were studied in details. This resulted in the "*Full Potential CSS*" theory [8], which in short involves: 1) describing discrete

quantities as discrete and continuous matter as continuous; 2) preserving the sojourn time distribution of a stage; 3) correctly implementing attributes; 4) including different types of uncertainties in an appropriate way.

It was also shown that the Full Potential CSS approach would provide a simple and straightforward solution to combined continuous and discrete simulation.

However, even though the theoretical problems were solved, existing CSS languages did not support and simplify the Full Potential CSS approach. These problems and necessary and practical extensions and tools are discussed and exemplified in Section 2.

The authors of this paper have now developed the open-source, Full Potential CSS package named StochSD (Stochastic System Dynamics) by complementing a classical CSS language with a number of facilities identified in the previous papers.

In short, the modelling and simulation package, StochSD, solves the following problems:

- It supports Full Potential CSS modelling and simulation to remove inconsistencies in results.
- It provides a simple way to combined continuous and discrete simulation.
- It provides StochSD for free to use in all kinds of courses, self-studies and research with CSS.
- In addition, extensive material for Full Potential CSS courses in form of lectures, laboratory exercises and instructive examples are also provided for free on the StochSD homepage (https://stochsd.sourceforge.io/homepage/#/).

In this paper, the simulation language StochSD and its use and possibilities are presented and demonstrated.

1.2. Purpose and Overview of StochSD

StochSD is an open-source, Full Potential CSS language[1] that has two main purposes:

1) To enable Full Potential CSS modelling and simulation within the CSS paradigm. This enables results consistent with those from Discrete Event Simulation or Agent-Based simulation of the same system.

2) To provide an open-source CSS language based on the System Dynamics philosophy for construction and simulation of small and medium-sized models in education, self-studies and research. Therefore, pedagogic aspects, ease of use and understanding are prioritised.

StochSD was developed by the authors during 2016-2021. To make StochSD and its philosophy easy to comprehend and use, it is based on the System Dynamics approach [9] [10] [11], where the system under study is described in terms of stocks and flows. StochSD complements classical CSS with a number of facilities to enable discrete modelling, in particular for discrete and stochastic transfers into, out of or between stocks. It also includes software for stochastic

[1]StochSD can be run within a browser (Google Chrome, Mozilla Firefox or Microsoft Edge) or downloaded from the StochSD homepage. Try it!

modelling, multiple runs, statistical analyses of the results and result presentation in statistical terms.

Technically, StochSD is written in JavaScript. It is based on the System Dynamics part of the modelling and simulation language Insight Maker [12] [13] [14] and a number of other packages that are all open-source. Technical aspects of StochSD and its structure are described in Appendix A.

StochSD is available for Windows, macOS and Linux and as a web application, which means that it can also be run under other operating systems.

StochSD has been tested in a number of university courses. Great effort has been spent on debugging, pedagogic aspects, flexibility and easy handling. To facilitate courses in Model Building and Simulation, nine lectures in form of 260 PowerPoint slides and five laboratory exercises are uploaded for free use on the StochSD homepage (https://stochsd.sourceforge.io/homepage/#/). The contents of the lectures and laboratory exercises are shortly presented in Appendix B.

2. The StochSD Simulation Package

A CSS model is in principle based on differential and algebraic equations (rewritten in numerical form), which in StochSD are structured in terms of stocks and flows into, out of or between stocks, in accordance with the concept of System Dynamics [9] [10] [11].

In the System Dynamics approach, the model is constructed by describing the system under study in terms of *primitives*: Stock: □ (state variable, compartment), Flow: ⇒ (continuous flow or transition of entities), Auxiliary: O (for algebraic calculation), Parameter: ◇ (for describing impact from the system's environment) and Link: → (to transfer causal information from one primitive to another). The StochSD model structure is built in a click-and-draw manner using these primitives. Thereafter, each primitive (except for Link) is opened to define a value or algorithm.

The System Dynamics approach has a number of advantages. The underlying bathtub analogy of a stock filled and drained by inflows and outflows makes the underlying "integration over time" concept intuitive. Placing the attention first on the model structure of interacting elements, and then on the algorithm of each element, leads to a focused and efficient way of modelling. Further, the various types of stochasticity are closely related to the different primitives used in System Dynamics.

In StochSD, continuous amounts are represented by *rational numbers* (called "*real*" or "*float*" in computer jargon) and the numbers of discrete entities are represented by *integer numbers* (or by "real" numbers if the entities are so many that they can be regarded as a continuous amount, e.g. water molecules in a river). However, representation of discrete entities in CSS has consequences that require a sequence of issues to be considered.

For a continuous flow, a fraction of the content in a stock will pass through a flow during a time step, but this has no counterpart for the transition of discrete

entities. The transition of entities between stocks must respect the integrity of the entities. (Also, for chemical processes where the entities become redefined, integrity is to be preserved). The transfer of entities can be *regular* or *irregular*.

- Transitions that occur *regularly* or are scheduled are straightforward to model, e.g. production of items by a machine can be described by a pulse function or a table function.
- *Irregular* transitions, on the other hand, occur at random points in time that cannot be foreseen. For example, radioactive decay, encounters between prey and predator, or arrival of customers at a shop are processes that usually have to be described by statistical means. This is often done using a Poisson process with a rate (intensity) of λ events per time unit (usually varying over time). Such an irregular process is realised by stochastic transitions in a flow. We denote this *transition stochasticity*. Later in this paper, we show how StochSD simplifies the modelling of a Poisson process by a new function: *PoFlow* (λ).

The possibility to draw random numbers from statistical distributions of various kinds (also empirical derived from observations) is also essential.

However, transition stochasticity, like other stochasticities, in a simulation model causes the results to vary between simulation runs (*replications*). For a deterministic model, *a scalar value* of a studied quantity is obtained from one simulation, but for a stochastic model *a distribution of values* is obtained that will require many replications to be well reproduced. Therefore, the results from these replications must be collected and statistically analysed, to provide statistical measures. For this reason, StochSD also includes a statistical tool called StatRes (Statistical Results), which collects results from a specified number of replications, performs statistical analysis and displays the statistical results in the form of means, standard deviations, confidence intervals (CI), min & max values, percentiles of selected quantities and correlations between these quantities. These statistical results can also be graphically presented in histograms and scatter plots.

An overview of issues supported by StochSD to become Full Potential is presented in **Figure 1**. In the following, we assume basic understanding of classical CSS and only discuss and exemplify the additional possibilities associated with StochSD following the numbered items, ① to ⑦, in this figure. These issues are then discussed in Sections 2.1 to 2.7.

2.1. Inclusion of Discrete Entities (① in Figure 1)

The representation of entities is technically simple and consists of initiating the stocks by integers and restricting the transitions during each time step (*DT*·*Flow*) to take integer values. The transitions may be regular (which is straight-forward to describe), but are usually irregular, e.g. the number of cars passing, customers arriving, accidents occurring, etc. during a time step. When such events are random and independent, we have a Poisson process. Then the

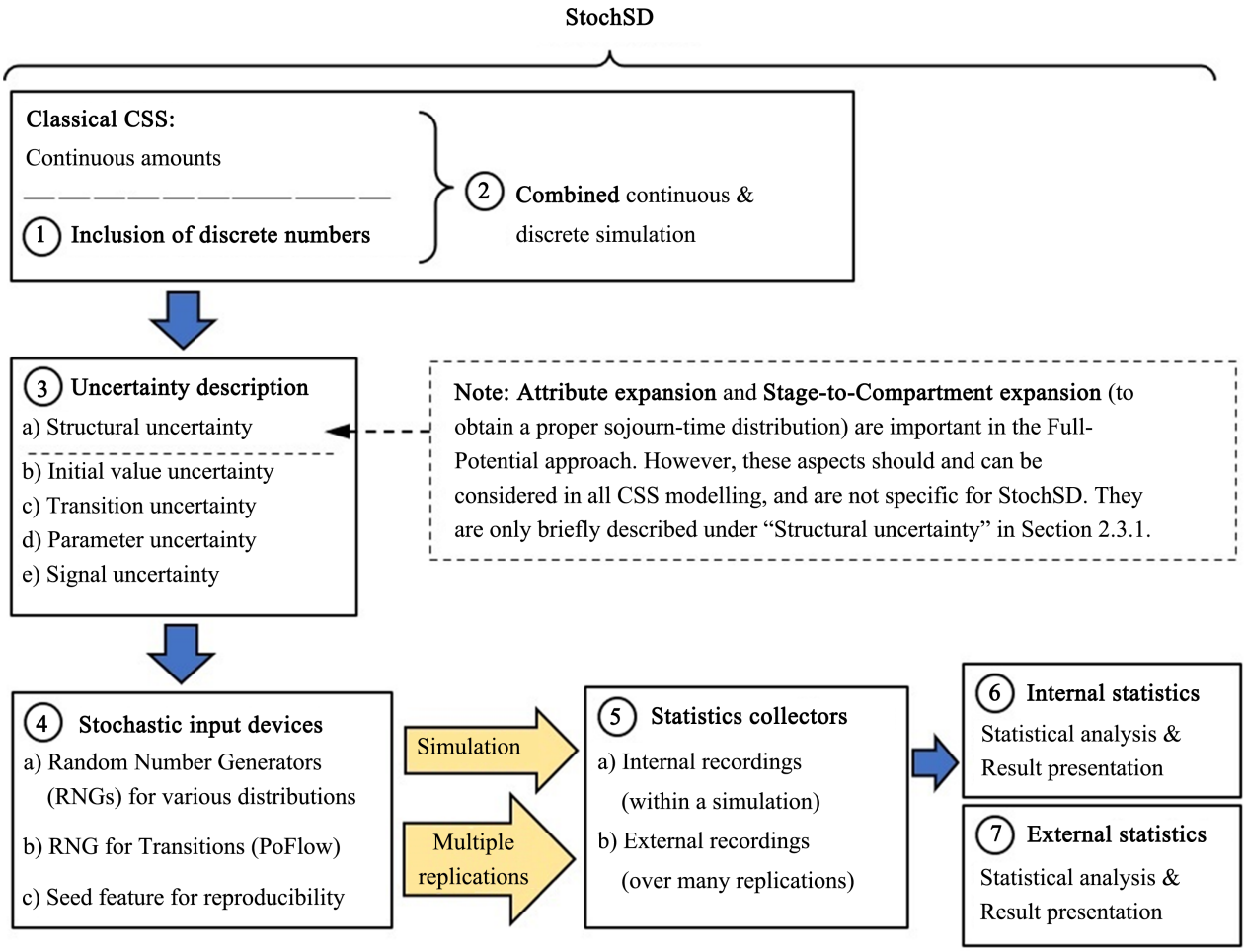

Figure 1. Extensions of classical Continuous System Simulation (CSS) to become a Full Potential CSS language through StochSD.

number per unit time interval is Poisson-distributed according to *Poisson* (λ), where the rate λ is the only argument. This argument may be constant, but will often vary with the content of a stock or values of other primitives (x, y, z), or over time, or all of these at the same time as λ (x, y, z, t). The number of transitions during the time interval DT will then be *Poisson* ($DT{\cdot}\lambda$) distributed (where DT is a sufficiently short time step not to distort the dynamics).

An advantageous property of the Poisson distribution, is that the distributions of *Poisson* ($DT{\cdot}\lambda$) and of *Poisson* ($1/2{\cdot}DT{\cdot}\lambda$) + *Poisson* ($1/2{\cdot}DT{\cdot}\lambda$) are the same, so the step-size, DT, is freely adjustable to fit the dynamic requirements, just as in classical CSS.

In CSS, the Poisson process handles randomness over time. However, the events may also have a size. For example, arrivals at a restaurant may occur as singles, pairs or groups, or accidents have consequences of different sizes. We are then dealing with a compound or mixed Poisson process [15] [16], where randomness over time uses the Poisson distribution and the size uses another proper distribution. This can also be handled in StochSD without further complications.

As an example, let us discuss the inclusion of a Poisson process for a model of radioactive decay where we only know the expected rate $\lambda(X(t))$, where X is the number of undecayed atoms.

Example 1: Continuous and discrete models of radioactive decay

A system of radioactive atoms decays independently and under the same condition. This is by definition a (time variable) Poisson process.

A simplified deterministic and continuous model of this is $\mathrm{d}X/\mathrm{d}t = -X/T$. In a numerical form this can be written:

$$X(t+DT) = X(t) - DT \cdot F(t) \tag{1}$$

$$DT \cdot F(t) = DT \cdot X(t)/T \tag{2}$$

where the output $DT{\cdot}F$ is the "real" fraction that decays during a time step, DT. This fraction is proportional to the number of remaining radioactive atoms $X(t)$ and inversely proportional to the time constant T.

In a stochastic model $DT{\cdot}F$ must be an integer number that is Poisson distributed with the same expected outcome. The model then becomes:

$$X(t+DT) = X(t) - DT \cdot F((t)) \tag{3}$$

$$DT \cdot F(t) = Po\left[DT \cdot X(t)/T\right] \tag{4}$$

where $Po[\sim]$ means that a Poisson distributed random number generator with ~ as argument is called at each time step.

As seen, only the flow statement has to be adjusted to make the deterministic model stochastic and remain integer-based. This is done by including the Poisson random number function, which draws an integer random number for each DT in the transition flow. The deterministic expression here becomes an argument to the Poisson distribution.

A programming comment is that you may not have $DT{\cdot}F$ on the left-hand side of an assignment statement (which was used to illustrate the similarity between the deterministic and stochastic flows). In programming, you must code: $F(t) = X(t)/T$ and $F(t) = Po[DT{\cdot}X(t)/T]/DT$, respectively.

In Figure 2, we show the two models and their behaviours for the case $X(0) = 30$ radioactive atoms with a time constant of $T = 10$ time units. □

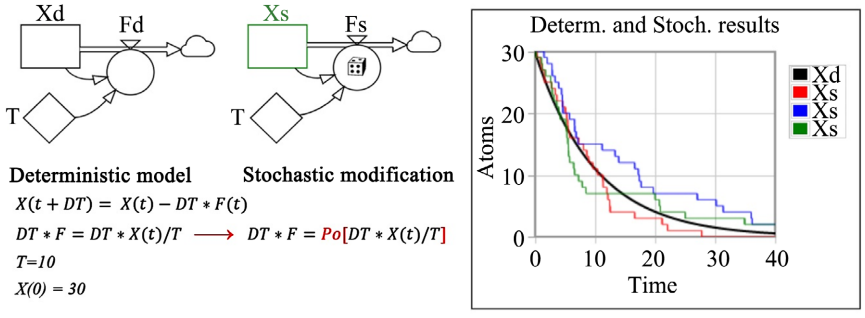

Figure 2. Deterministic and stochastic models of a radioactive decay process with the time constant T. Po [·] stands for a Poisson-distributed random number drawn at each time step of length DT. Results from one deterministic (bold & black) and three stochastic (coloured) replications.

To simplify the introduction of transition stochasticity into a flow, StochSD introduces the stochastic transition function *PoFlow*(λ) for $Po(DT\cdot\lambda)/DT$, which also more clearly illustrates the close relationship between $F(t) = X(t)/T$ and $F(t) = PoFlow(X(t)/T)$. (This falls under points ③c and ④b in **Figure 1**, but is mentioned here to simplify the examples to come.)

2.2. Combined Continuous & Discrete Simulation (② in Figure 1)

A powerful bonus is that combined continuous and discrete simulation becomes easy to perform. The following is a slightly adjusted example from [8].

Example 2: Combined continuous and discrete prey-predator model

Conceptual model: The Lotka-Volterra equations describe a prey-predator system for two species, X and Y, by differential equations [17] [18] [19]. The prey breeds at a rate proportional to its size X, and is reduced because of encounters with predators, Y, which is proportional to $X\cdot Y$. We also include competition among the 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 deaths of the predators are proportional to their number Y. The continuous Lotka-Volterra model then takes the mathematical form:

$$\mathrm{d}X/\mathrm{d}t = a\cdot X - b\cdot X\cdot Y - c\cdot X^2 \tag{5}$$

$$\mathrm{d}Y/\mathrm{d}t = d\cdot X\cdot Y - e\cdot Y \tag{6}$$

where a and d are fertility constants, b and e mortality constants, and c is a proportionality constant for competition.

Now, we want the prey to be a continuous amount of biomass X (e.g. of grass), while the predators Y are represented by a discrete number of entities (e.g. sheep).

The combined continuous-discrete model is easily obtained by making the inflows and outflows to and from Y stochastic, according to the Poisson distribution, whereby a random and integer number of entities transfers into and out of Y at each time step. Initialising "sheep" to an integer number will thus keep the number of sheep integer. Below, the combined model is shown where the modifications from the continuous to discrete form are shown in Equations (12) and (13):

$$X(t+DT) = X(t) + DT\cdot\left[F1(t) - F2(t) - F3(t)\right] \quad (\text{Cont. grass}) \tag{7}$$

$$F1(t) = a\cdot X(t) \tag{8}$$

$$F2(t) = b\cdot X(t)\cdot Y(t) \tag{9}$$

$$F3(t) = c\cdot X^2(t) \tag{10}$$

$$Y(t+DT) = Y(t) + DT\cdot\left[F4(t) - F5(t)\right] \quad (\text{Discrete sheep}) \tag{11}$$

$$F4(t) = PoFlow\left[d\cdot X(t)\cdot Y(t)\right] \tag{12}$$

$$F5(t) = PoFlow\left[e\cdot Y(t)\right] \tag{13}$$

A single replication of the combined continuous-discrete model, exemplified by $a = 0.2$, $b = 0.005$, $c = 0.001$, $d = 0.005$ and $e = 0.3$, and starting at the equilibrium $X(0) = 60$ tons of grass and $Y(0) = 28$ sheep is shown in **Figure 3**. In the replication shown, the discrete predators became extinct at around 265 time units (e.g. months). The continuous "prey" then increases logistically to an equilibrium without stochastic variations.

Comparing the continuous model with the combined model reveals that the continuous model, starting at the equilibrium state, will only produce two straight horizontal lines: $X(t) = 60$ tons of grass and $Y(t) = 28$ sheep. Starting the continuous model outside the equilibrium state will make it approach the equilibrium for both species without lasting variations. Further, a phenomenon such as extinction cannot occur for the continuous model. □

Note how easily a combined CSS model can be constructed in StochSD. The alternative of constructing a combined continuous and discrete model, with its mixture of disparate micro and macro concepts, time-handling methods and requiring a special combined simulation language, is not an attractive option for a macro study.

2.3. Uncertainty Description (③ in Figure 1)

Uncertainties can be of two main types, *structural* and *statistical*. We first briefly describe the possible uncertainties with the help of an example and then discuss each of them in more detail, referring to this example.

Example 3: An epidemic SIR model (to be discussed throughout Section 2.3)

An epidemic SIR model, named after its three sequential stages (Susceptible, Infectious and Recovered) of an infectious disease process, is shown in **Figure 4** [18] [19] [20]. To make it simple, each stage is represented by a single stock. An *Authority* that will react when the number of infectious individuals becomes too high is also included.

This example is used to demonstrate where and how different types of uncertainty can be described. Uncertainty may be about: **1)** the *model structure*, **2)** the

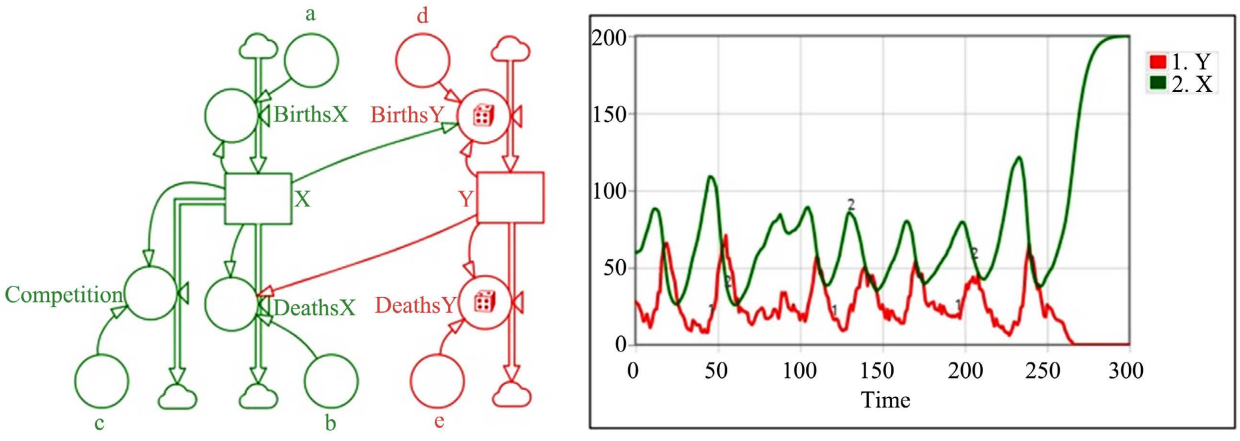

Figure 3. The combined prey-predator model with continuous prey (X = Grass) and discrete predators (Y = Sheep), and a replication of this model. (Note that the colouring of the stocks X and Y is preserved in the time plot.)

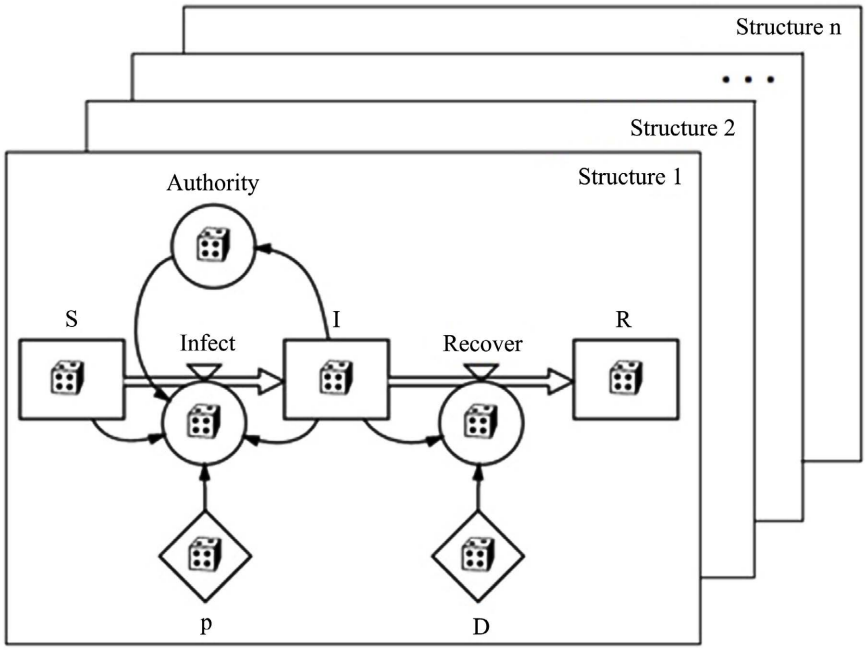

Figure 4. Diagram of epidemic SIR models with an Authority, showing where different types of uncertainties can be modelled. These uncertainties may relate to the model structure or to any algorithm or value defined in a primitive (represented by a die symbol in the primitive).

initial value of a stock, **3)** the *transitions* in a flow, **4)** the *value* or *sequence of values* in a parameter, and **5)** the *information* (*signal*) affecting the process. (To illustrate the signal uncertainty, an authority is added that monitors the epidemic and will intervene by recommendations or restrictions if the number of infectious individuals becomes too high. The uncertainty in this last case refers to information transmitted via a link, but the uncertainty involved is then handled in an auxiliary controlling the link.)

The five types of uncertainty to be discussed (*structural*, *initial value*, *transition*, *parameter*, and *signal*) all refer back to this example. Of these five types, structural uncertainty refers to the overall structure of the model or to how a stage or attribute should be represented, and can be handled by alternative model structures or sub-structures. The other four uncertainties refer to different types of primitives. These are handled by describing the uncertainty by using a specified statistical distribution from which to draw random numbers.

2.3.1. Structural Uncertainty (③a in Figure 1)

What model structure is most appropriate for the study? This issue is the same whether the model is continuous or discrete, deterministic or stochastic.

The structure of the SIR model in **Figure 4** can be questioned. One possibility is that an infected person becomes infectious after a latent period. This can be modelled by including an Exposed (E) stage between the Susceptible and the Infectious stages. (The model is then called a SEIR model.) Another possibility is that Recovered (immune) persons will lose their immunity after some time, thus

returning to the Susceptible stage (making a so-called SIRS or SEIRS model). In particular, it is important to understand the difference between a stage and a stock, and to consider whether it is important to include further attributes.

Stage-to-compartment expansion

On a more detailed level, the expansion of a stage into a structure of compartments (in parallel and/or series) to generate a proper sojourn time distribution in the stage is another example of structural uncertainty. (By modelling the infectious stage by a single compartment, the sojourn-time distribution is implicitly assumed to be exponential, which is very untypical for e.g. a pregnancy or a trip from A to B.)

Attribute expansion

When introducing an important attribute, e.g. a risk attribute to model disease behaviour and progression for high- and low-risk individuals, two or more separate (but interacting) chains of compartments and flows may be necessary.

Structural uncertainty is about exploring alternative model structures to find that generating the most similar behaviour to the studied system.

2.3.2. Statistical Uncertainties (③b, c, d and e in Figure 1)

The reason for describing uncertainties in a model is that *uncertainties contain information*. If one can describe the distribution of observations and the value of its parameters (e.g. mean, standard deviation, min, max), then more information is included. This, when correctly done, makes the model behave more realistically and display different possible behaviours.

Uncertainties are often handled by using an estimated average value in the model to keep the model deterministic. This may work in some cases, but may be disastrous in others. For example, a model of a queuing system using an average time, a, between arriving customers and an average service time, s, (where $s < a$) will never make a queue, while a stochastic model (using the same a and s) can cause large queues and waiting times.

Each type of statistical uncertainty is associated with a specific primitive: Initial uncertainty with stocks, transition stochasticity with flows, parameter uncertainty with parameters, and signal uncertainty with links controlled by an auxiliary. These uncertainties are all treated in a similar way, by first describing the uncertainty by a statistical distribution and then drawing random numbers from this distribution.

Initial value uncertainty (③b in Figure 1)

Information about the initial number of individuals in the three stocks (S, I, R) shown in Figure 4 may be uncertain. However, assume that we know that the total population under study is $N = 1000$ people and that three of these individuals have returned from abroad bringing home a new disease, but we do not know whether one, two or all three of these returned infectious. The initial value for the stock $I(0)$ is then 1, 2 or 3. Assuming that the probability is 60% for $I(0) = 1$, 30% for $I(0) = 2$ and 10% for $I(0) = 3$, a uniformly distributed random number (here called [*Rand*]) between 0 and 1 can be drawn and used to initialise

the *I*-stock at time zero. In StochSD, this can be obtained by the multi-row function:

$$\begin{aligned}&\text{If } [\text{Rand}] < 0.6 \text{ Then}\\&\quad 1\\&\text{Else If } [\text{Rand}] < 0.6 + 0.9 \text{ Then}\\&\quad 2\\&\text{Else}\\&\quad 3\\&\text{End If}\end{aligned} \tag{14}$$

(Note that if you place the call to the RNG within the algorithm, you will obtain two calls giving the algorithm a different meaning.)

Further, for the remaining 1000 − $I(0)$ people, we assume that the statistics indicate that an individual has a 40% chance of being immune (belonging to the *R*-stock in Figure 4). Then $R(0) = RandBinomial(1000 - I(0), 0.4)$ and $S(0) = 1000 - I(0) - R(0)$ are a proper initialisations of the *R*- and *S*-stocks that utilises the available information. □

Note: Because initial stochasticity only affects the stocks at time zero, the initialisation will not be influenced by the choice of time step, *DT*.

Transition uncertainty (①, ③c and ④b in Figure 1)

Transition of discrete entities in a flow was partly covered in Section 2.1, because it is intrinsically connected with discreteness.

There are systems where entities are transferred in a deterministic way (which is straightforward to model). However, often the random number of events occurring during a time step is unknown and thus must be handled stochastically. It is therefore important to be able to handle transition stochasticity within the flows into, out of and between stocks.

In the deterministic case of the SIR model shown in Figure 4, the flows "*Infect*" and "*Recover*" are defined by: $Infect = p \cdot S \cdot I$ and $Recover = I/D$, where p is the infectious parameter and D is the duration parameter for the sojourn in the infectious stage. The stochastic transitions may then be obtained by *PoFlow* (·) of the deterministic expressions, *i.e.*:

Deterministic model		Stochastic model	
$Infect = p \cdot S \cdot I$	$\rightarrow$	$Infect = PoFlow(p \cdot S \cdot I)$	(15)
$Recover = I/D$	$\rightarrow$	$Recover = PoFlow(I/D)$	(16)

$DT \cdot PoFlow$ will always produce an integer number of transitions that on average amounts to $DT \cdot p \cdot S \cdot I$ and $DT \cdot I/D$ events per time step, respectively. Thus, if one starts with integer numbers in the stocks, then the transitions add or subtract integer numbers, so the stocks will remain integer. □

As already mentioned in Section 2.1, the choice of time step does not affect the number of expected events per time unit when using the Poisson distribution.

Parameter uncertainty (③d in Figure 1)

Unexplained impact from the environment on the system under study can be modelled by stochastic parameters. In this case, there is no a priori form to describe such uncertainties and only statistical observations or other knowledge about the variations can act as guides.

In the SIR model in Figure 4, the infectious parameter, p, and the parameter describing the sojourn time as infectious, D, may vary with the weather, behaviours of the individuals and other conditions, in an unpredictable way.

A problem with parameter stochasticity is that its modelled influence will change with the size of the time step, DT. If DT is reduced, more random numbers will be drawn per time unit. The effect of this is smoothing, because of the law of large numbers. One way to handle this problem is to change a parameter after fixed periods of time. This allows DT to be less than or equal to the specified time interval, without affecting the parameter stochasticity if DT is changed. □

Signal uncertainty (③e in Figure 1)

Signals, *i.e.* communication of information in a system under study, can be *distorted* and/or *delayed*.

In Figure 4, this is exemplified by an authority that collects information about the number of infectious persons and analyses the data (which takes time and is never exact). The authority may then issue recommendations about using sanitary measures and avoiding contacts, which later on when the message is received may affect the rate of the "*Infect*" flow.

The time taken to collect, analyse and distribute the information to the receivers may vary in an unpredictable way (information may be sent by post; it may arrive during the weekend but be noticed first on Monday morning, etc.). Information may also be distorted because of imperfect data, misinterpretation, not reaching all subjects, etc.).

The length of a delay can be generated by drawing a random number from one proper distribution, and the distortion of the number of infectious individuals may be modelled by another stochastic distribution. In this way, uncertainty about signals can be handled by the sending and/or receiving primitive (e.g. an auxiliary).

For signal stochasticity too, the effects of drawing numbers from the statistical distributions are usually sensitive to the time step used for the simulation, so it will be wise to keep the random values constant for fixed periods of time.

2.4. Stochastic Input Devices (④ in Figure 1)

To enable construction of a stochastic CSS model, random number generators (RNGs) of various statistical distributions from which random numbers can be drawn are required. A way to make a replication or series of replications of a stochastic model *reproducible* is also a valuable asset.

2.4.1. Random Number Generators (RNGs) (④a in Figure 1)

Stochasticity is obtained by a function call to a RNG, which returns a sample

from the specified distribution for each time step. Some CSS languages have few, and others more, RNGs. StochSD has inherited a large number of RNGs from Insight Maker. In particular, it is important to be able to customise an RNG from empirical data (e.g. from measurements). In StochSD, this RNG is called *RandDist*. Some of the 13 RNGs in StochSD are:

- *Rand* (*Min*, *Max*) (Uniform distribution)
- *RandBernoulli* (*Probability*) (Bernoulli distribution)
- *RandBinomial* (*Count*, *Probability*) (Binomial distribution)
- *RandPoisson* (*Rate*) (Poisson distribution)
- *RandExp* (*Rate*) (Exponential distribution)
- *RandNormal* (*Mean*, *StdDev*) (Normal distribution)
- *RandDist* ({{*x1*, *y1*}, {*x2*, *y2*}, …, {*xn*, *yn*}}) (Custom distribution)

2.4.2. The RNG for Discrete Transitions: PoFlow (④b in Figure 1)

In order to simplify the rather ugly flow expression $F = RandPoisson(DT*\lambda)/DT$ for handling random transitions of entities in a flow equation, a new function doing exactly the same is: $F = PoFlow(\lambda)$, which is more readable. This function also illustrates the close relationship between the deterministic and stochastic cases (just include the deterministic flow expression as an argument of *PoFlow* ()). See Sections 2.1 and 2.3.2, above.

2.4.3. Making a Stochastic Model Reproducible (④c in Figure 1)

Making a stochastic model reproducible may seem to be a contradiction. However, this is a legitimate technique that has many advantages. It is possible because stochasticity is generated by pseudo-RNGs, where each RNG produces a very long sequence of numbers that has virtually all the statistical properties of the specified statistical distribution. When the sequence finally ends (which will not happen in practice), it starts over and repeats the cycle. When a seed is defined, it indicates where in the cycle to start, which gives the opportunity to repeat a simulation run exactly.

Reproducibility can be used in the following cases:

- To demonstrate or reconstruct a simulation run because something of particular interest happens, e.g. a species becomes extinct or a dam bursts, in order to understand why or to demonstrate how this happened.
- When comparing two models, they must be tested under *similar conditions*. For example, when comparing different designs of a queuing system, having the same sequences of arriving customers for the two cases would eliminate one source of variation in the results. This technique is called *variance reduction* using *common random numbers*. Unfortunately, use of variance reduction (except for very simple cases) requires the possibility to assign *separate seeds* to each RNG, which is not possible in StochSD.

StochSD makes a stochastic simulation model reproducible by locking the seed for the random number generators in the model. Since this feature is global (rather than belonging to each RNG), it is defined as a macro. In the macro, the

seed is set to e.g. 17 by writing: *SetRandSeed* (17).

By changing the argument, another (reproducible) simulation run is obtained.

2.5. Statistics Collectors (⑤ in Figure 1)

A stochastic model produces outcomes that vary stochastically *within a replication* and *between replications*. Therefore, a large number of observations within a replication must be recorded, or the results over many replications must be assembled to provide good statistical estimates.

It is therefore necessary to distinguish between these two kinds of statistics which we denote *internal* and *external*. Although the internal and external statistical measures are almost the same, the tools for obtaining them are different.

2.5.1. Collecting Internal Statistics (⑤a in Figure 1)

This concerns the stochastic variations within a replication, where relevant data are collected (usually for each *DT*) during the replication. It is then convenient to include statistical functions in the model to update the statistics during the replication by counting events, summing up quantities, updating variances, testing if X is the largest or smallest value so far, etc.

StochSD has statistical functions, inherited from Insight Maker, for min, max, mean, median, standard deviation and correlation, for collecting and calculating internal statistics from a part or the whole simulation:

- *PastMin* (…)
- *PastMax* (…)
- PastMean (…)
- *PastMedian* (…)
- *PastStdDev* (…)
- *PastCorrelation* (…)

In special cases, for example when queues are involved, case-specific devices for average number waiting for a resource, average waiting time, utilization of resources such as taxi cabs, doctors, etc. can be built. This is shown in Example 4 in Section 2.6, below.

2.5.2. Collecting External Statistics (⑤b in Figure 1)

This concerns the stochastic variations over many replications, where the end results from each replication are collected. Note that functions, which *at the end of a replication* show the overall maximum, the value when something interesting happens, the cumulated or average performance during the replication, etc. can be used or easily be constructed.

The collection of external results over many replications is taken care of by a StochSD tool named StatRes, see Section 2.7, below.

2.6. Internal Statistics (⑥ in Figure 1)

In the internal case, the statistical estimates are calculated in the model during the simulation, where the functions mentioned in Section 2.5 present the results.

Because the observations come from a dynamic (and stochastic) process, there is *dependence* between $X(t)$ and $X(t+T)$, and between $X(t)$ and another variable $Y(t)$. This means that the samples are not independent, so the Central Limit Theorem does not apply. Thus, confidence intervals based on assumptions of a normal distribution cannot be derived in the internal case (unless it is possible to first divide the sequence into independent time intervals).

In the following example, taken from [8] and simplified, we demonstrate how a queuing system can be modelled and how devices for queuing measures, such as average queue length, average waiting time and utilisation of a server, can be constructed.

Example 4: Queuing model and constructed statistical devices

A shop that is open for eight hours a day has an expected arrival rate of $\lambda = 8$ customers per hour, and the customers are served by a single assistant with a service rate of $\mu = 10$ customers per hour. A stochastic model, where QS denotes the actual number of queuing and served customers, can be:

$$QS(t+DT) = QS(t) + DT(In - Out) \tag{17}$$

$$In = PoFlow(lambda) \tag{18}$$

$$Out = MIN[PoFlow(mu), QS/DT] \tag{19}$$

Figure 5 shows the queue model with a number of devices for internal statistics.

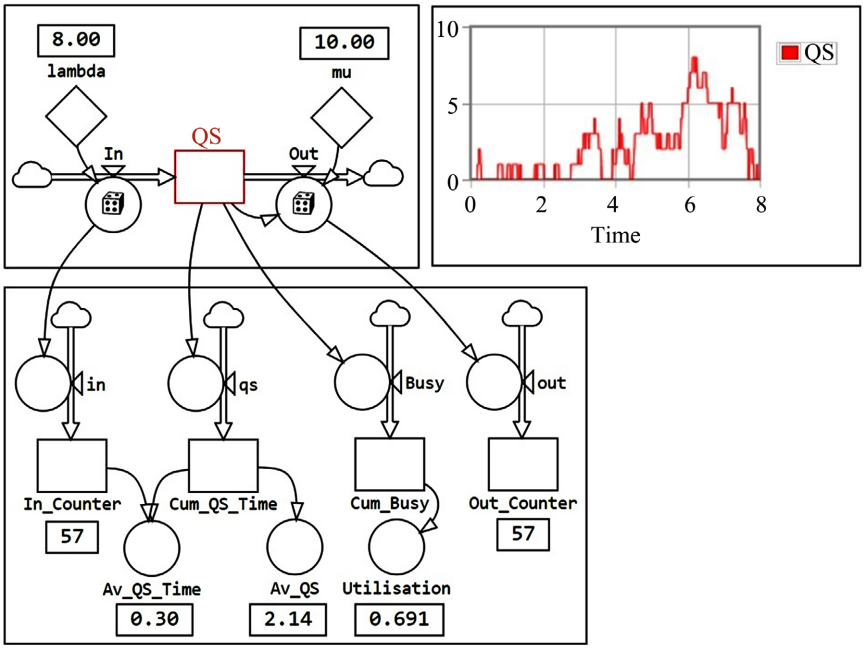

Figure 5. The upper part of the diagram shows a so-called *M*/*M*/*1* queuing system [21], where the stock QS holds the number of "Queuing and Served" customers and a replication showing how QS might vary over time. The lower part of the diagram shows devices for counting arrivals and departures, and for calculating average queue-time, average queue-length and server utilisation. (Time unit: hour.)

The collectors of internal statistics on arriving and departing customers, queuing customers, waiting times and utilisation of the server are smoothly modelled in StochSD as follows:

In_Counter and *Out_Counter* count the customers entering and leaving the shop, respectively.

Cum_QS_Time accumulates *QS* over time to obtain the total waiting time in *QS*.

Dividing *Cum_QS_Time* by *Time* + *eps* gives the average number in the queuing system *Av_QS*, and dividing it by *In_Counter* + *eps* gives the average waiting time *Av_QS_Time*.

Finally: *Busy* = *Min* ([*QS*], 1) accumulates the server's activity over time into *Cum_Busy*. The average utilisation of the server is then: *Utilisation* = *Cum_Busy*/(*Time* + *eps*).

(The machine epsilon, *eps*, is the smallest number that can be represented. It prevents "division by zero" before the counters are incremented.) □

Note: A queuing model requires transition stochasticity of discrete entities to be meaningful. A corresponding continuous and deterministic model will erroneously not generate any queues or waiting times.

2.7. External Statistics (⑦ in Figure 1)

In the external case, the statistical estimates are collected from many replications. Note that there is a principal difference from the internal case in calculation of a statistic. In the internal case, the conditions change dynamically during the replication, so the samples are not independent. In the external case, the repeated replications of the same model and under the same experimental conditions are independent and identically distributed experiments. Therefore, the Central Limit Theorem applies, so calculation of confidence intervals based on the assumption of normal distribution is legitimate when the number of replications is not too small. (In StochSD, at least 20 replications are required.)

StochSD contains the powerful tool StatRes (Statistical Results), which connects to the current model, orders *N* replications, collects results of specified quantities and calculates statistics on average, standard deviation, confidence interval, min and max values, and percentile for all selected quantities. Testing of hypotheses can also be done, based on confidence intervals.

StatRes can also presented the statistics graphically in the form of histograms and by scatter plots (*X*-*Y* plots), with coefficient of correlation between *X* and *Y* calculated. It can also control the seed-of-seeds used to produce a new seed for each replication in order to make the study, based on multiple replications, reproducible.

The communication between StatRes and the model is restricted to before and after full replications, and updating of graphics in the model is disconnected to make StatRes fast.

Example 5: Lanchester's combat model analysed by StatRes

Lanchester's model of warfare from 1916 describes the battle between two forces X and Y as a *deterministic*, *linear* model. The casualties in X are inflicted by Y according to $Fx = c \cdot Y$. Similarly, $Fy = c \cdot X$, where the hitting power c = 0.1 (here assumed to be the same for both forces) is the number of eliminated entities per time unit that each member can inflict on the enemy. Initially $X(0)$ = 5 and $Y(0)$ = 4 entities. The battle ends when the last entity of the losing force is eliminated, see [18] [19] [22]. Below, the code for the continuous, deterministic model and for the modifications to a corresponding discrete, stochastic model are shown:

Deterministic model		Stochastic modifications	
$X(t+DT) = X(t) - DT \cdot Fx$			(20)
$Fx = c \cdot Y$	$\rightarrow$	$Fx = PoFlow(c \cdot Y)$	(21)
$Y(t+DT) = Y(t) - DT \cdot Fy$			(22)
$Fy = c \cdot X$	$\rightarrow$	$Fy = PoFlow(c \cdot X)$.	(23)

The StochSD model and the results from two replications are shown in **Figure 6**.

We are now interested in comparing the results from the deterministic and stochastic models: *Which force will win? What number of entities will remain after the battle? How long will the battle take?*

For the *deterministic model*, a single simulation gives that X will always win, the remaining force is X = 3.00 units and the battle will always take 11.00 time units. To test whether these results are correct and representative, StatRes was used for a sequence of 10,000 replications of the stochastic model. The results are shown in **Figure 7**.

Summing up the results obtained from 10,000 replications shows that the results from the deterministic model were all wrong: 1) X wins on average 71% of the battles (95% CI: 0.70 - 0.72)—not all battles, and Y wins 29%. 2) The expected number of X entities after battle is 2.57 entities (95% CI: 2.53 - 2.61), which deviates significantly from the deterministic model result of X = 3 entities. 3) The expected number of Y entities after a battle is 0.77 (95% CI: 0.74 - 0.80),

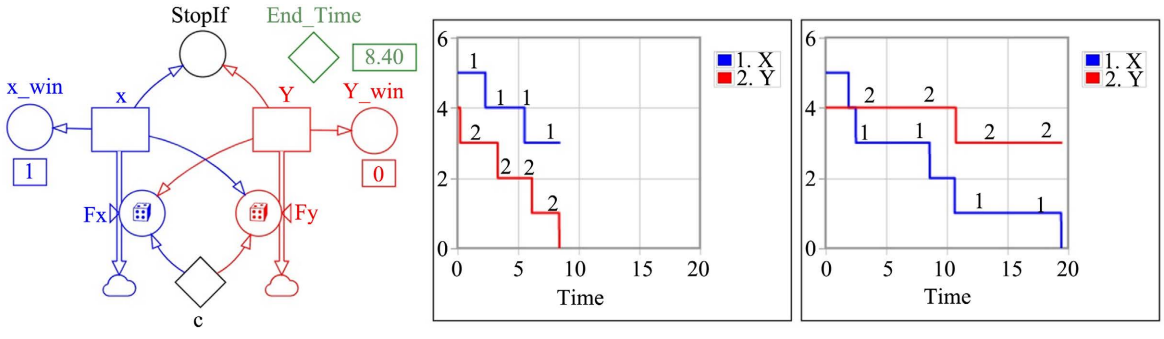

Figure 6. Lanchester's model where random hitting is introduced, and the results from one replication where X wins and one where Y wins. The *StopIf* auxiliary ends the simulation when the last X or Y is eliminated, which saves a lot of execution time, and *End_Time* shows the length of the battle.

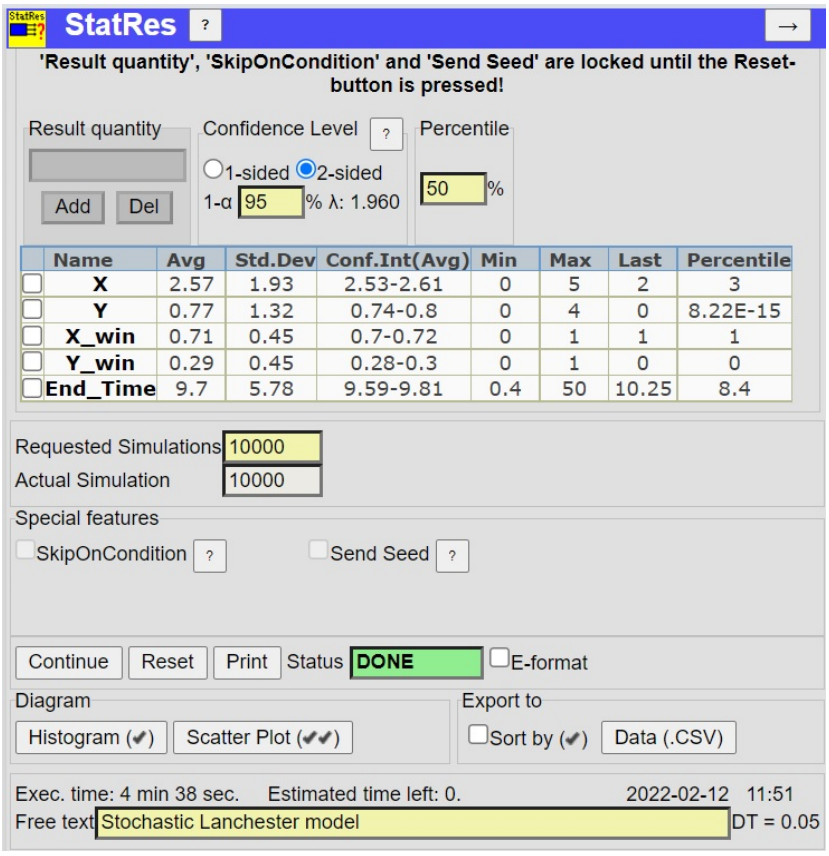

Figure 7. Results obtained using the StatRes tool. The result quantities *X*, *Y*, *X_win*, *Y_win* and *End_Time* were selected and 10,000 replications of the Lanchester model were ordered from StatRes. With $DT = 0.05$ time units, the replications and statistical analyses took 4 min 38 s on a Core I5 PC under Windows 10.

not zero. 4) The average length of a battle is 9.70 time units (95% CI: 9.59 - 9.81), which is significantly shorter than the 11.00 time units obtained from the deterministic model.

This example also illustrates that *linearity* does not ensure unbiased results for a deterministic model. For a deeper discussion, see [6]. □

Comment: Do not confuse the confidence interval (CI) around an average estimate (given above) with the spread of results for a quantity. For example, the average length of a battle was 9.70 time units (95% CI: 9.59 - 9.81), while the interval covering 95% of the battles varied between 2.30 and 24.25 time units. (This was easily obtained by setting the "Percentile" percentage to 2.5 and 97.5, respectively and reading the percentile values for *End_Time*.) Histograms and correlations can also be used in StatRes, and the raw data for each replication can be exported to e.g. a spreadsheet or a statistical program as Comma Separated Values (CSV).

3. Using StochSD

In this chapter, we discuss what StochSD looks like, how a model is constructed and executed, the philosophy behind StochSD, what tools are included, and what

a StochSD model can reveal.

3.1. The StochSD Graphical User Interface

When StochSD is opened, the graphical user interface (GUI) shown in **Figure 8** appears.

On the upper part of the screen, there is a title bar, a panel with a row of menus and a row of buttons. Below the panel is the modelling window, where a model can be built in a click-and-draw manner.

The Title bar shows the file name with its path, and the time of the last saving (for a saved model file).

The Menu bar consist of eight pull-down or pop-up menus named: File, View, Print (Print Diagram or Print Equations), Simulation Settings (Start Time, Length, Time Step and Integration Method), Macros (for own macros or setting a seed), Colour, Tools (Optim, Sensi, StatRes and ParmVar), and Help (About StochSD, Licenses and Manuals for StochSD and its Tools).

The Button bar includes buttons for Construction (Stock, Auxiliary, Parameter, Converter—a table look-up function, Flow, Link, and Ghost), for Styling (Rotate name, Move valve, Straighten link, Text, Rectangle, Ellipse, and Arrow/Line), for Results presentation (Number box, Table, Time Plot, Compare Simulations Plot, XY-Plot, and Histogram), and for Execution of the model (Run/Pause, Step, and Reset).

To the right of the Execution buttons is a Progression field, where an orange bar expands over the field during execution and becomes green when the simulation is finished. This field also displays Start Time, Current Time, Integration method and Time Step used. To the right of the Progression field there is a button for the Time Unit, which must be specified before any model building can start.

To the left under the buttons, there is an Inspection field that shows the "equation" of a marked primitive without having to open it. To the far right, there are links to the homepages of StochSD and Insight Maker.

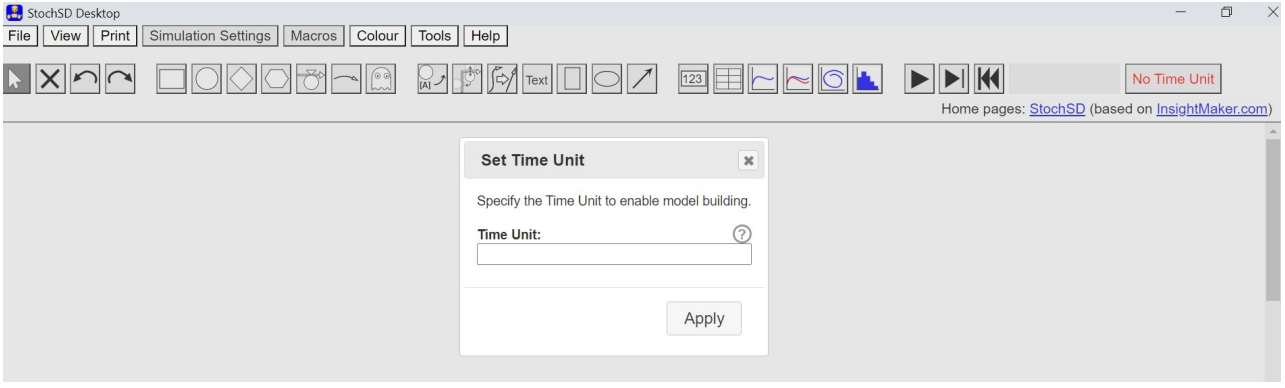

Figure 8. The StochSD graphical user interface, which is composed of a title bar, menu bar, button bar and modelling window. At start, the time unit to be used throughout the modelling must be specified (e.g. "hour", "day", "year"; or just "Time Unit" or "t.u." for a generic model where the aim is to demonstrate a principle without limiting it to a time scale). No modelling can begin before this is specified.

Under the panel is the Modelling window ("canvas"), where the model is constructed and Tables, Plots, Number boxes, texts, frames, lines, etc. can be added.

3.2. Construction and Execution of a Model

StochSD applies the System Dynamics approach, where the model is constructed by the primitives Stock: □, Auxiliary: O, Parameter: ◇, Converter: (hexagon), Flow: ⇒, Link: →, and Ghost (duplicate of a primitive with a ghost symbol inscribed).

As soon as the time unit is specified (then shown to the right of the button bar), model building can begin. The model structure is built in a click-and-draw manner by placing and connecting the primitives on the canvas. Thereafter, each primitive is opened to define an algorithm or a value. This also applies for the Plots, Tables and Number boxes. Specification of length, time step, etc. is done in the Simulation Settings menu. Then the simulation can be started by pressing the Run button (▶).

Figure 9 shows a model of radioactive decay similar to that in Example 1 in Section 2.1. Here the *Decays* flow primitive is opened for definition.

Naming and defining the primitive

StochSD's syntax is inherited from Insight Maker. The name and the equation (or more correctly "algorithm in an assignment statement") of a primitive are defined in the dialogue box of a stock, flow, auxiliary, parameter or converter.

The name of a primitive is given in the Name field. It must start with a bracket, [, followed by a string that may only contain the characters A to Z, a to z and _ (anywhere) and 0 to 9 (if not the first character), and ends with a bracket,], e.g. [*Stock1*], [*Water_Flow*], [*c*]. However, if the name is defined without brackets, StochSD will include them.

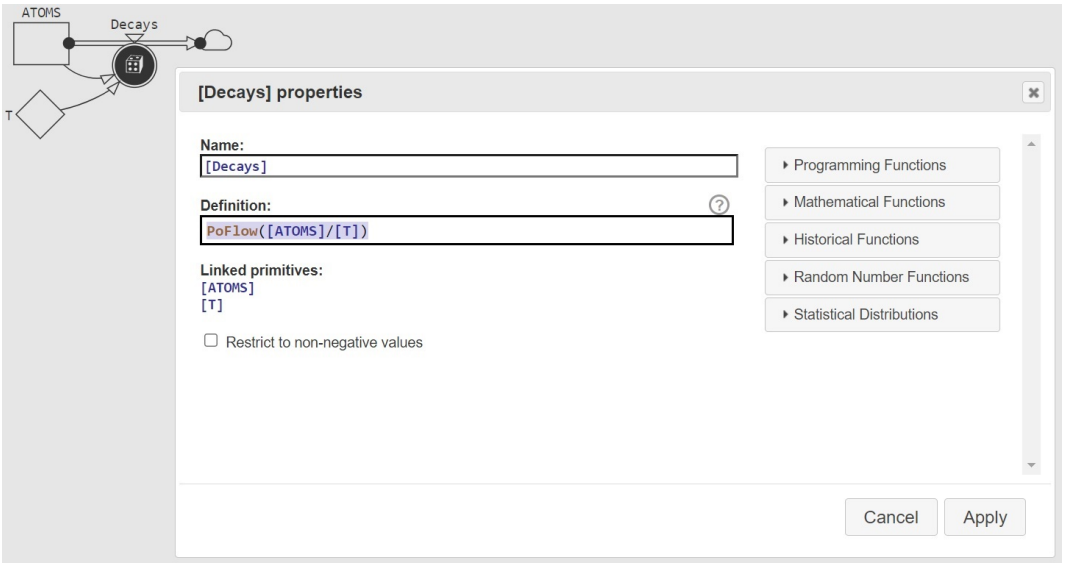

Figure 9. The dialogue box for the flow named "Decays" is opened. In the Name and expandable Definition fields, the flow can be given a name and the algorithm can be written (e.g. as *PoFlow* ([*ATOMS*]/[*T*]). A rich function library is also available.

In the Definition field, the algorithm is composed of primitives, functions, numbers and operators. Here, brackets must be included around the primitives. The easiest way to include a primitive is to click it in the list "Linked primitives:" (see **Figure 9**). (On the canvas, the brackets are hidden and in this paper we usually exclude the brackets around the primitives when explaining a model.)

Further, a function ends with parentheses "()", with or without parameters. A number may include a decimal point and e-format, e.g. 1.7e−12 (meaning 1.7×10^{-12}) is also allowed.

Primitives, functions and numbers are combined by the arithmetic operators + − * / ^ () for addition, subtraction, multiplication, division, power (x^n) and parentheses. As always, the calculation order is: power, then multiplication and division, and last addition and subtraction. Parentheses can be used to alter this order. After a # sign, a comment on the algorithm can be added.

If the definition is empty, contains unmatched brackets or parentheses, not includes all linked primitives, or includes unlinked primitives then the primitive is marked with "**?**" to show where there are more to specify or correct.

The *StochSD User's Manual and Tutorial* [23] provides a detailed description about construction and execution of a StochSD model. This manual is included under the Help menu and can also be downloaded from the StochSD homepage.

3.3. The Philosophy behind StochSD

The lesson learnt from several decades of teaching is that it is crucial to give the student a realistic insight into what modelling and simulation is, and what it is not. A model is a huge simplification of reality. "*Everything should be made as simple as possible, but no simpler.*" (attributed to Albert Einstein). "*All models are wrong, but some are useful.*" (attributed to George Box). These statements should be central insights communicated in every course in modelling and simulation. However, it is also important to understand that a model includes *more* than a simplified description of a system. Therefore, all model constituents should be understood. A StochSD model can be described as an onion with several layers, as shown in **Figure 10**.

The Core layer of the model describes the dynamic and algebraic relations of the system under study in terms of stocks, flows, auxiliaries, etc. (but no parameters). In the Subjective layer, one can define concepts that have no counterpart in the system under study, for example biomass, market price or an objective function (e.g. for optimisation where each unit of Stock 1 is valued at 30 Euros and each unit of Stock2 at 10 Euros.)

It is also important to consider that a system under study seldom is unaffected by its environment, *i.e.* by factors that are outside the system borders (in space, time or details). The Environmental layer of the model therefore includes *Initial values* (that depend on what has happened before the study period), as well as *unexplained influences* (from the environment). In StochSD, parameters (◇) are used for this. Parameters can be constants or functions, but they cannot have incoming links since they are unexplained quantities.

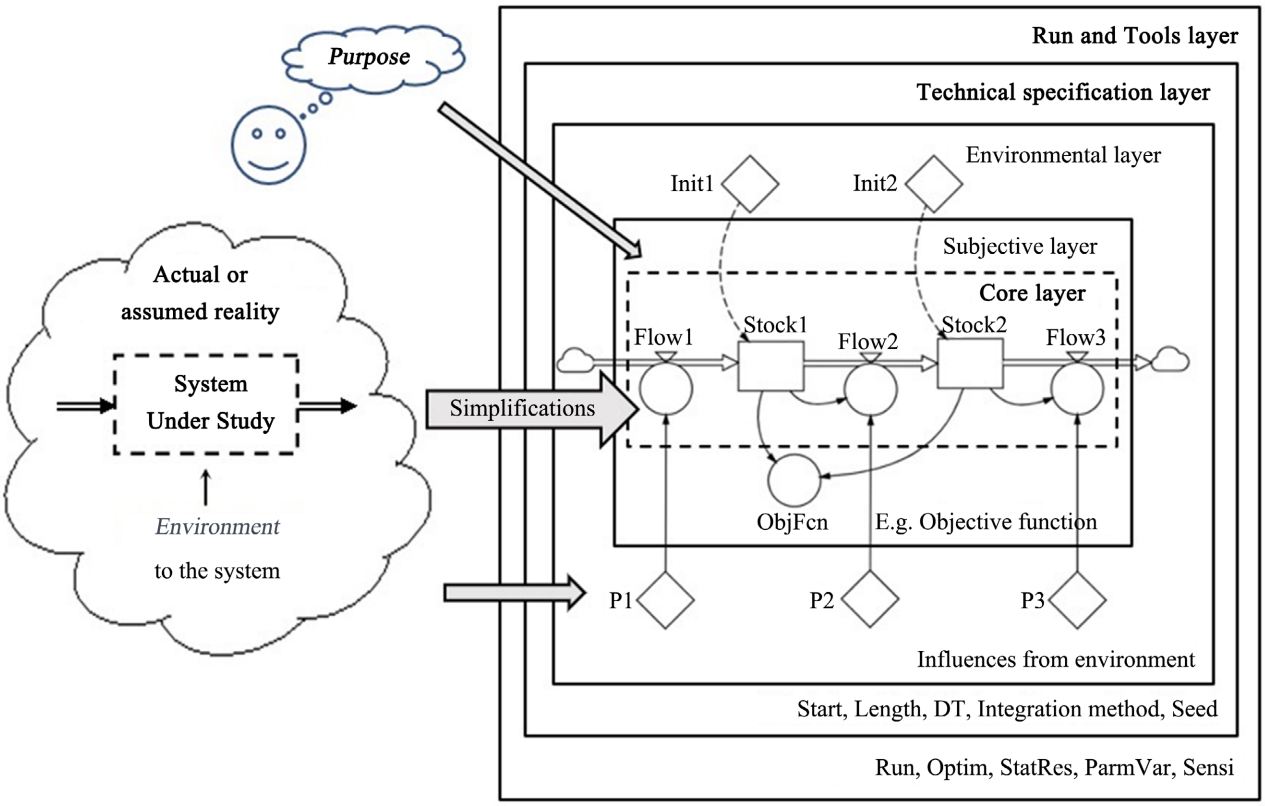

Figure 10. A model is *less* and *more* than a description of a system: The "Core layer" is a huge simplification of the system under study, focusing on dynamic and algebraic relations. In the "Subjective layer", one can create subjective concepts to fit the purpose. In the "Environment layer", external influences on the system under study are described. In the "Technical specifications" layer, start time, length of simulation, step size etc. are specified. Finally, a "Run and Tools layer" allows model behaviour to be generated and investigated.

In practical model building in StochSD, the three inner layers are mixed on the canvas, but the cloud symbols of a flow show that the sources and sinks are located outside the system borders. Further, the parameter symbol (◇) tells that it is an unexplained quantity.

In the Technical specifications layer, start time, length of simulation, integration method, time step and the seed for initiating random number generators are all artificial concepts.

Finally, there is a Run and Tools layer, from where model behaviour can be generated, analysed or optimised in a controlled way. Together, these five layers constitute the model to be studied.

3.4. Four Integrated Tools

StochSD has integrated tools for optimisation, sensitivity analysis, statistical analysis, and analysis of parameter variations. These tools require multiple replications to collect results, followed by analysis and result presentation. They are:

- **Optim**—for optimisation or parameter estimation of a deterministic model. This tool is based on a simplex optimiser, which means that it is reasonably efficient and can also deal with constraints [24] [25] [26].

- **Sensi**—a tool for studying the sensitivity of one or several specified quantities on changing the value of e.g. a parameter or an initial value [27].
- **StatRes**—collects results from multiple replications, makes statistical analyses and presents statistical results (average, standard deviation, max, min, confidence interval, correlation, etc.). The statistical results can also be presented in graphical form [28].
- **ParmVar**—a hybrid of StatRes and Optim for parameter estimation of stochastic models. ParmVar performs multiple replications of a stochastic model, and for each of these replications parameter estimation is made. These multiple estimations of model parameters show the variations in parameter estimates in statistical terms [29].

These tools are described briefly in [23], and in detail in [26] [27] [28] [29].

3.5. Comparing Models to Show What Stochasticity Can Reveal

To show the power and versatility of StochSD as a Full Potential CSS language, we conclude this section with an example comparing three deterministic and stochastic models.

Example 6: Continuous modelling hides important information

Three simple deterministic models with very different structures are presented:

a) Logistic model (an open, first-order, non-linear model) with a population P that increases in proportion to its size ($a \cdot P$) and decreases because of competition where everyone competes with all the others ($b \cdot P^2$),

b) SI model (a closed, second-order, bilinear model) where infectious individuals, I, transfer the disease on meeting susceptible individuals ($c \cdot S \cdot I$),

c) Pruned Prey-Predator model (an open, second-order, bilinear model), where encounters between prey X and predator Y bring death to the prey and promote an increase in Y (according to $d \cdot X \cdot Y$ and $e \cdot X \cdot Y$).

These three deterministic models and their behaviours are shown in **Figure 11**.

Now we introduce transition stochasticity in each flow, because number of births and deaths, number of infected, and number of prey deaths and predator births are independent, random events. Therefore, each "deterministic flow statement" is now replaced by *PoFlow* (*deterministic flow statement*), see **Figure 12**.

The randomly occurring events make the three models reveal their differences in behaviours. In the Logistic model (12a) the population, P, can go up and down, and in one replication P becomes extinct. In the SI model (12b), the number of infectious individuals, I, increases monotonically, but always ends at the same level. In the Pruned Prey-Predator model (12c) the number of predators, Y, also increases monotonically, but ends at different levels. This is described in more detail in [8]. □

In **Figure 12**, the "Compare Simulations Plot", where results from *different replications* can be displayed together, is used. (This type of plot is also useful to

display the effect of different values of a parameter, or when searching for an appropriate time step.)

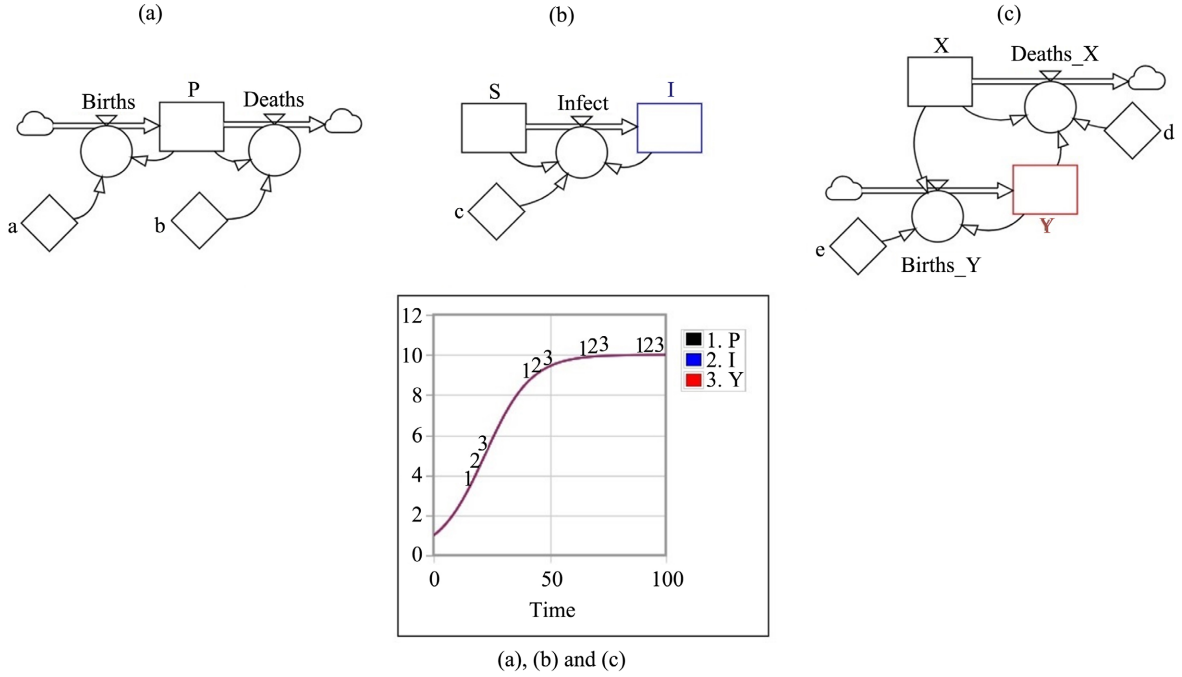

Figure 11. (a)-(c) Deterministic Logistic, SI and pruned prey-predator models, which show identical behaviours despite their different structures. (a) Logistic model, P(0) = 1, a = 0.1, b = 0.01; (b) SI model, S(0) = 9, I(0) = 1, c = 0.01; (c) Pruned Prey-Predator model, X(0) = 9, Y(0) = 1, d = 0.01, e = 0.01.

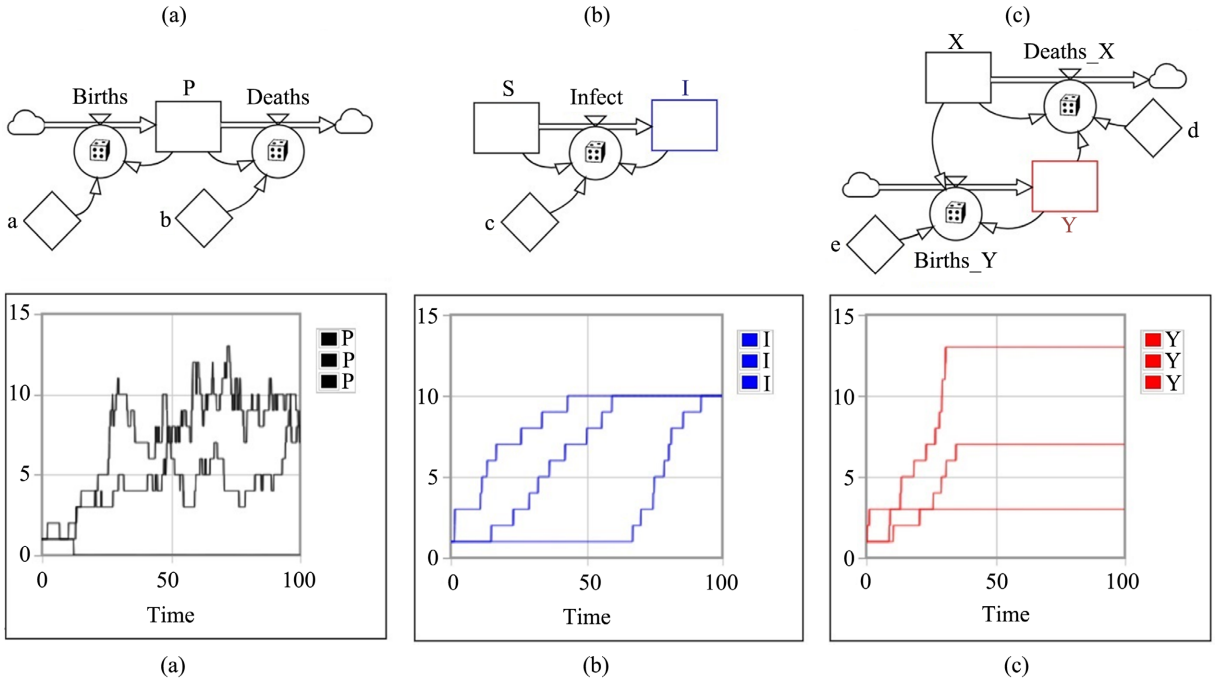

Figure 12. (a)-(c) Stochastic version of the Logistic, SI and Pruned Prey-Predator models shown in **Figure 11**, and their behaviours in three replications each. (a) Logistic model, P(0) = 1, a = 0.1, b = 0.01; (b) SI model, S(0) = 9, I(0) = 1, c = 0.01; (c) Pruned Prey-Predator model, X(0) = 9, Y(0) = 1, d = 0.01, e = 0.01.

4. Discussion and Conclusions

StochSD has two main purposes: 1) To enable Full Potential CCS modelling and simulation, and 2) To provide an open-source CSS language based on the System Dynamics philosophy for construction and simulation of small and medium-sized models in education, self-studies and research, where, pedagogic aspects, ease of use and understanding is prioritised.

4.1. Practical Use of Full Potential CCS Modelling and Simulation

The underlying Full Potential CSS theory is presented in [8], and is only briefly discussed and exemplified in this paper. In short, StochSD supports the "Full Potential CSS" requirements by:

- Enabling modelling of continuous matter as "real" numbers and discrete entities as integers.
- Recommending stage-to-compartment expansion when required.
- Recommending attribute expansion when required.
- Enabling means for structural, transition, initial value, parameter and signal stochasticities.
- Providing tools for collecting both *internal statistics* (within a replication) and *external statistics* (over multiple replications).
- Providing tools for multiple runs, statistical analyses and results presentation.

Full Potential CSS is a break from the old tradition where CSS modelling is used to describe both continuous matter and discrete entities as *continuous*, often with disastrous results.

The old view of simulation in general is well formulated by Wolfgang Kreutzer in his book: *System Simulation*, *Programming Styles and Languages* ([30], p. 18): "*Modelling styles are paradigms in this sense. Monte Carlo methods*, *continuous*, *discrete-event and combined simulation all have their own methodologies*, *tools*, *traditions*, *user communities and prototypical applications*." With the Full Potential CSS approach, all four of these "paradigms" can be consistently handled in a unified and smooth way on macro level [7] [8].

StochSD is powerful enough to be used in practical studies in epidemiology, ecology, traffic flows, queuing systems, production and many other fields. In epidemiology, the dominant approach is currently to make (randomised) studies with well-defined start and end points analysed by statistics. However, here the underlying process is dynamic and may involve different types of uncertainties that cannot be handled within a static statistical approach. A scientific study in epidemic modelling based on StochSD is presented in [31].

4.2. Use in Education and Self-Studies

To support the use in education and self-studies, we have created the StochSD homepage [32], where we supply free material to enable courses in classical CSS and Full Potential CSS modelling and simulation. This material is based on lectures and laboratory exercises developed by Professor Leif Gustafsson during his

40 years of experience in research and teaching modelling and simulation on all levels at Swedish universities.

The resources provided at the StochSD homepage include:

- The StochSD software package for Windows, Mac OS X and Linux or as a web application. It also means that StochSD is available to each student.
- StochSD User's Manual and Tutorial [23].
- Manuals for the tools Optim, Sensi, StatRes and ParmVar [26] [27] [28] [29].
- Lectures L1 - L9 in form of 260 PowerPoint slides for courses in modelling and simulation, and L0 discussing their use and legal aspects.
- Laboratory exercises for Classical CSS (Lab 1 & Lab 2) and Full Potential CSS (Lab 3 - Lab 5).
- Instructive example models in StochSD as .ssd files.
- A link to the paper "The Full Potential of Continuous System Simulation Modelling" [8].

A Full Potential CSS course is best conducted in two steps:

Step 1: Basic deterministic CSS modelling

A course in modelling and simulation should be based both on theory and on practical model building and experimentation. Therefore, lectures teaching the basics about a system under study and a model of it, describing how a modelling project should be performed, as well as the CSS language StochSD are all crucial. Further, two full-day laboratory exercises based on deterministic modelling and simulation where the student does the modelling, experimentation and thinking is also recommended.

After introducing basic concepts such as system under study, model, purpose, etc., it is appropriate to start with deterministic modelling with the focus on how structure causes behaviour (especially effects of positive and negative loops, oscillations, control, etc.). An excellent introduction to this is "*Thinking in Systems*" by Donella Meadows [11].

It is also important to understand the difference between stage and compartment (stage-to-compartment expansion) to obtain realistic sojourn time distributions, and how addition of attributes sometimes is necessary but will create a larger model, see [8].

Deterministic models can be based on *amounts* and sometimes also on *fractions*. Fractions make a model scalable.

Step 2: Implementing and handling discreteness and stochastics

When the student understands deterministic modelling and is familiar with the concepts of stocks and flows, it is time to address how discreteness and uncertainties should be modelled. In particular, it is important to understand how discrete entities are represented and how this often is connected to transition stochasticity. For a discrete model, *the actual number of entities* has to be modelled (here fractions cannot be used).

Different types of uncertainty (structural, transition, initial value, parameter, and signal) must be understood and should be covered in practical modelling.

The effects of stochasticity should also be taught and practised, by building corresponding deterministic and stochastic models and comparing their behaviours. It is also essential to understand the danger of using classical CSS to model discrete entities as continuous.

Further, a stochastic model requires multiple replications, so the student must understand how to analyse stochastic variations within a replication and over multiple replications in statistical terms. Basic statistics (distribution, mean, standard deviation and confidence interval) are required to understand and use statistical functions and the StatRes tool. Two laboratory exercises will consolidate basic knowledge about stochastic modelling and simulation.

In Appendix B, a suggested schedule for an extensive course in "Model building and Simulation using StochSD" gives an overview of the provided course material. This schedule also indicates the content of each lecture and laboratory exercise. However, the teacher is free to use or modify, change, translate, etc. the material to form his or her own course (with exceptions for six slides from Pixabay [33] and one from OpenStreetMap [34], where their generous licenses must be respected—see Lecture 0: Introduction & License at the StochSD homepage).

4.3. Conclusions

The CCS Language StochSD has the two purposes: 1) To make macro modelling consistent with micro modelling by enabling use of Full Potential CSS. 2) To provide a free, open-source, pedagogic tool for courses in macro modelling and simulation.

StochSD was mainly developed and tested in a number of university courses during 2019-2021, and was officially released in January 2021. In January 2022, an improved version was released. Until now (mid-February, 2022) more than eleven hundred downloads of StochSD in 58 countries have been done, as well as an unknown number of the web version of StochSD. The Full Potential aspects have also been thoroughly tested, e.g., in [31]. Our conclusion is that StochSD works well both technically and pedagogically.

StochSD is meant to be easy to use and understand, why we have not included vector modelling (although this is possible—but not graphically supported), nor have we included animation of e.g., the content in stocks, values of auxiliaries, etc. One desirable feature would be to rebuild the random number generators (RNG) so that each RNG could have an individual seed instead of sharing a global one. This would enable more powerful variance reduction. Future work will focus more on correctness and pedagogical issues than on making StochSD more advanced.

We also expect the free, extensive course material just included to be helpful, especially to initiate high-quality education in macro modelling and simulation.

Although we find that StochSD works well, it is not perfect. We regard it as the first serious attempt to design a Full Potential CSS software package for stochastic and dynamic macro modelling and simulation. The open-source philosophy, where all code is available for use, modification, improvements or new

software, is important to us. We hope that this will be utilised in the future to make CSS modelling and simulation an even more powerful tool based on a stable scientific foundation.

Acknowledgements

We thank Scott Fortmann-Roe for his excellent work with the development of Insight Maker, which constitutes the base for StochSD, and for his kind and fast replies and help on several occasions. We also thank the other providers of open-source facilities used in StochSD.

The software development was partly funded by the Swedish Cancer Society, Contract number: 150846. The Karolinska Institute, Stockholm, Sweden also provided partial economic support for the development of StochSD.

Further development and testing of StochSD were performed in two Master's projects at Uppsala University, Sweden. StochSD was also tested in several courses during its development at the Swedish University of Agricultural Sciences supervised by Sven Smårs.

The authors want to thank Mikael Sternad for valuable discussions and suggestions.

We also thank Mary McAfee (Scantext) for spell checking, improving the language and suggesting various other improvements of the manuscript.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Gustafsson, L. (2000) Poisson Simulation—A Method for Generating Stochastic Variations in Continuous System Simulation. *Simulation*, **74**, 264-274. https://www.researchgate.net/publication/220164720

[2] Gillespie, D.T. (2001) Approximate Accelerated Stochastic Simulation of Chemically Reacting Systems. *The Journal of Chemical Physics*, **115**, 1716-1733. https://doi.org/10.1063/1.1378322

[3] Gustafsson, L. (2003) Poisson Simulation as an Extension of Continuous System Simulation for the Modeling of Queuing Systems. *Simulation*, **79**, 528-541. https://doi.org/10.1177/003759703040234

[4] Gustafsson, L. and Sternad, M. (2007) Bringing Consistency to Simulation of Population Models—Poisson Simulation as a Bridge between Micro and Macro Simulation. *Mathematical Biosciences*, **209**, 361-385. https://doi.org/10.1016/j.mbs.2007.02.004

[5] Gustafsson, L. and Sternad, M. (2010) Consistent Micro, Macro and State-Based Population Modelling. *Mathematical Biosciences*, **225**, 94-107. https://doi.org/10.1016/j.mbs.2010.02.003

[6] Gustafsson, L. and Sternad, M. (2013) When Can a Deterministic Model of a Population System Reveal What Will Happen on Average? *Mathematical Biosciences*, **243**, 28-45. https://doi.org/10.1016/j.mbs.2013.01.006

[7] Gustafsson, L. and Sternad, M. (2016) A Guide to Population Modelling for Simulation. *Open Journal of Modelling and Simulation*, **4**, 55-92. https://doi.org/10.4236/ojmsi.2016.42007 http://file.scirp.org/pdf/OJMSi_2016042717425486.pdf

[8] Gustafsson, L., Sternad, M. and Gustafsson, E. (2017) The Full Potential of Continuous System Simulation Modelling. *Open Journal of Modelling and Simulation*, **5**, 253-299. https://doi.org/10.4236/ojmsi.2017.54019 http://www.scirp.org/JOURNAL/PaperInformation.aspx?PaperID=80104

[9] Forrester, J.W. (1961) Industrial Dynamics. MIT Press, Cambridge, MA.

[10] Forrester, J.W. (1968) Principles of Systems. Wright/Allen Press Inc., Cambridge, MA.

[11] Donella Meadows, H. (2008) Thinking in Systems: A Primer. Chelsea Green Pub., White River Junction, VT.

[12] Fortmann-Roe, S. (2014) Insight Maker: A General-Purpose Tool for Web-Based Modeling & Simulation. *Simulation Modelling Practice and Theory*, **47**, 28-45. http://www.sciencedirect.com/science/article/pii/S1569190X14000513 https://doi.org/10.1016/j.simpat.2014.03.013

[13] Fortmann-Roe, S. (2022) Manual for Insight Maker (Providing More Features, e.g. Functions, that Are Supported but Not Described in StochSD. Note that Only the System Dynamics Part of Insight Is Supported in StochSD). Insight Maker. https://insightmaker.com/manual

[14] Fortmann-Roe, S. (2022) Insight Maker API. Dokumentation of Built-In Functions. https://dokumen.tips/documents/insight-maker-api.html

[15] Bratley, P., Fox, B.L. and Schrage, L.E. (1983) A Guide to Simulation. 2nd Edition, Springer-Verlag, New York. https://doi.org/10.1007/978-1-4684-0167-7

[16] Grandell, J. (1997) Mixed Poisson Processes. Chapman & Hall/CRC, London.

[17] Volterra, V. (1926) Fluctuations in the Abundance of a Species Considered Mathematically. *Nature*, **118**, 558-560. https://doi.org/10.1038/118558a0

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

[19] Braun, M. (1993) Differential Equations and Their Applications. 4th Edition, Springer-Verlag, New York.

[20] Kermack, W.O. and McKendrick, A.G. (1927) A Contribution to the Mathematical Theory of Epidemics. *Proceedings of the Royal Society of London. Series A*, **115**, 700-721. https://doi.org/10.1098/rspa.1927.0118 https://royalsocietypublishing.org/doi/pdf/10.1098/rspa.1927.0118

[21] Gross, D. and Harris, C.M. (1998) Fundamentals of Queueing Theory. 3rd Edition, John Wiley & Sons, New York.

[22] Lanchester, F.W. (1916) Aircraft in Warfare, the Dawn of the Fourth Arm. Tiptree, Constable & Co., Ltd., London.

[23] Gustafsson, L. (2022) StochSD User's Manual and Tutorial. https://stochsd.sourceforge.io/manuals/StochSD_User_Manual.pdf

[24] Nelder, J.A. and Mead, R. (1965) A Simplex Method for Function Minimization. *The Computer Journal*, **7**, 308-313. https://doi.org/10.1093/comjnl/7.4.308

[25] Press, W.H., Flannery, B.P., Teukolsky, S.A. and Vetterling, W.T. (1989) Numerical Recipes (in FORTRAN, Pascal or C). The Art of Scientific Computing. Cambridge University Press, Cambridge.

[26] Gustafsson, L. (2019) Optim—An Optimiser for Deterministic StochSD Models.

https://stochsd.sourceforge.io/manuals/StochSD_Optim.pdf

[27] Gustafsson, L. (2019) Sensi—A Sensitivity Analyser for StochSD Models. https://stochsd.sourceforge.io/manuals/StochSD_Sensi.pdf

[28] Gustafsson, L. (2019) StatRes—A Tool for Statistical Analysis of Stochastic StochSD Models. https://stochsd.sourceforge.io/manuals/StochSD_StatRes.pdf

[29] Gustafsson, L. (2019) ParmVar—A Tool for Studying the Variations of Parameter Estimates in StochSD Models. https://stochsd.sourceforge.io/manuals/StochSD_ParmVar.pdf

[30] Kreutzer, W. (1986) System Simulation: Programming Styles and Languages. Addison-Wesley Publishing Company, Inc., Boston.

[31] Gustafsson, M. (2020) Evaluation of StochSD for Epidemic Modelling, Simulation and Stochastic Analysis. Master's Thesis, Uppsala University, Uppsala. http://www.diva-portal.org/smash/get/diva2:1503898/FULLTEXT01.pdf

[32] StochSD's Homepage (2022). https://stochsd.sourceforge.io

[33] Pixabay (2022) A Community of Creatives, Sharing Copyright Free Images, Videos and Music. All Contents Are Released under the Pixabay License. https://pixabay.com/

[34] OpenStreetMap (2022) A Free, Editable Map of the Whole World that Is Being Built by Volunteers Largely from Scratch and Released with an Open-Content License. https://www.openstreetmap.org/

[35] JavaScript (2022) The Programming Language of HTML and the Web. http://www.javascript.com/

[36] jqPlot (2022) A Plotting and Charting Plugin for the jQuery JavaScript Framework for Line, Bar and Charts with Many Features. http://www.jqplot.com

[37] jQuery (2022) A JavaScript Library Designed to Simplify HTML DOM Tree Traversal and Manipulation, as Well as Event Handling. https://www.w3schools.com/jquery/

[38] jQuery UI (2022) Contains User Interface Components Built on Top of the jQuery JavaScript Library. It Handles Pop-Up Windows. https://jqueryui.com/

[39] Normalize (2022) Makes the User Interface Look More Similar across Browsers. https://www.npmjs.com/package/normalize.css/v/3.0.2

[40] CodeMirror (2022) A Text Editor. Used for Styling and Brackets Matching. https://codemirror.net/

[41] NW.js (2022) A JavaScript Desktop Wrapper (Previously Known as Node-Webkit). Runs JavaScript without a Browser. https://nwjs.io/

Appendix A: Technical Aspects of StochSD

Specification

The StochSD project started by defining necessary and desirable properties of a an open-source, "Full Potential CSS language" based on the System Dynamics approach [8] [9] [10] [11]. The project was performed during 2016-2021 under the supervision of Professor Leif Gustafsson, based on his 40 years of experience in research and teaching micro and macro modelling and simulation on all levels at Swedish universities, in cooperation with his sons Erik and Magnus Gustafsson who carried out the programming and selection of proper open-source packages.

Choice of a host CSS language

As a host package for the development of StochSD, the choice was to use the System Dynamics part of Insight Maker [12] [13], developed by Scott Fortmann-Roe. Insight Maker is a well-structured simulation package of high quality, and with a few exceptions open source. It is written in JavaScript [35] and thus platform independent.

From Insight Maker, StochSD inherited a platform that includes the main System Dynamics elements, the difference equation solvers for Euler and RK4, the large library of appropriate statistical functions and random number generators, the macro facility, the error checking facility, the imperative environment with general programming functions and the API. However, Insight Maker's graphical user interface is dependent on non-open-source packages. In StochSD, these packages are replaced with open-source alternatives to make all of StochSD open source.

Design and implementation of StochSD—some major additions and changes

StochSD inherited several third-party open-source packages with Insight Maker. We also added the open-source library packages jqPlot [36], jQuery [37], jQuery UI [38], Normalize [39] and CodeMirror [40]. A large number of additions and changes were made in construction of the StochSD package, including:

Open-source changes and additions

- Non-open-source packages from Insight Maker were replaced.
- A new graphical user interface is constructed.
- A new file handling facility is constructed.
- New Dialogue boxes for primitives, plots, tables, etc.
- New result presentation objects: Plots, table, histogram, and number boxes.

Full Potential CSS additions

- Insight Maker was selected for its statistical features (and for quality reasons).
- A tool for optimization and model fitting is developed.
- A tool for sensitivity analysis is developed.
- A tool for statistics from multiple replications is developed.
- A tool for studying the variation in parameter estimates in a stochastic model is developed.

- New or adjusted functions are added, e.g.: *PoFlow* (..) that simplifies transitions and visualises the relation between the deterministic and stochastic cases. *StopIf* (*Condition*) that stops the execution after the actual time step is completed. This often saves a lot of execution time for stochastic models.

Functional additions

- Checking that all primitives from incoming links are included in the definition, and that no primitive without an incoming link is included.
- Checking for unmatched brackets and parentheses in definitions of primitives.
- Changing a primitive's name will automatically enter all algorithms involved.
- A Print menu is provided for model documentation: Print Diagram and Print Equations.
- Giving a colour to a primitive implies that it takes that colour in a time plot.

Pedagogical and aesthetical improvements

- Time unit specification replaces the dimension specification of Insight Maker.
- Primitives are redesigned to meet the de facto standard of System Dynamics.
- A parameter primitive is introduced as an auxiliary that cannot take incoming links.
- The flow can be broken in right angles, and the valve symbol can be flipped or moved to a proper segment of a broken flow.
- Flows with loose ends automatically start and/or end with a cloud symbol.
- The link is displayed as a cubic Bézier curve, so it can be nicely shaped.
- A primitive with an undefined or faultily defined algorithm is marked with "**?**".
- A die symbol shows that stochasticity is included in a primitive.
- A "*SetRandSeed*" button is inserted in the macro form.
- User's manuals for StochSD and the tools are both provided online (under the Help menu) and on the StochSD homepage.
- Lectures in form of PowerPoint slides are provided on the StochSD homepage.
- Laboratory exercises are provided on the StochSD homepage.
- Instructive StochSD models are provided on the StochSD homepage.

Furthermore, technical improvements, conformity, reliability, speed, protection, warnings, etc. have been made, and errors found during extensive testing, debugging and laboratory exercises have been corrected.

The structure of the StochSD package

An overview of the structure of the StochSD package is presented in **Figure A1**.

As shown in **Figure A1**, StochSD is available in two versions: StochSD-Desktop, which can be downloaded to a computer and includes NW.js [41] to interpret the JavaScript code; and StochSD-Web, which is run within a browser (Google Chrome, Mozilla Firefox or Microsoft Edge). The desktop version is recommended because models can more easily be locally stored in a file structure de-

cided by the user. On a computer where this is not possible, for example Chromebook, the web application is a way to run StochSD.

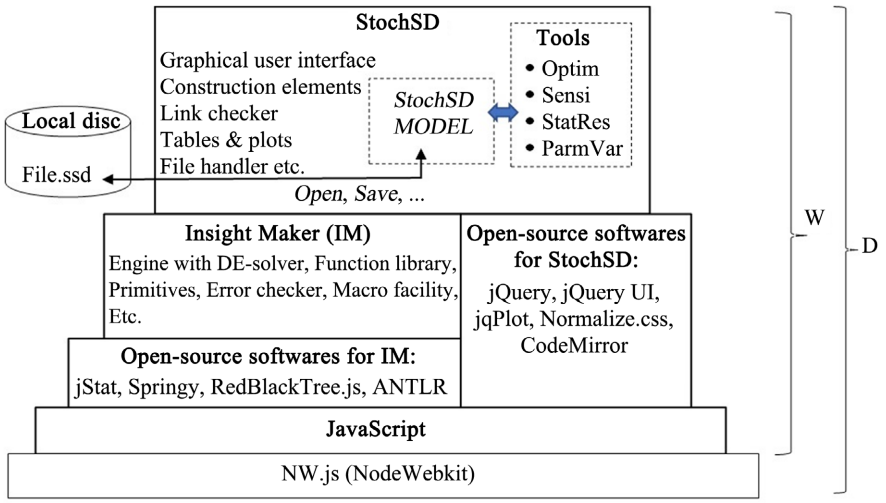

Figure A1. Components of StochSD and its use of third-party software. D: StochSD-Desktop, meaning that StochSD includes NW.js. W: StochSD-Web, meaning that StochSD must be run within a browser.

Appendix B: Suggested Schedule

Table B1 informs about the content of the uploaded material for a course in Modelling and Simulation. You can pick the parts of interest by downloading it from the StochSD homepage.

Table B1. Suggested schedule for a course in Modelling and simulation. (Each Lx contains material for several hours, Lab x about one day).

	Section	Comments
L1	Basic concepts	System under study, model, purpose, etc.
L2	Modelling in StochSD	Model building and preparations for Lab1
Lab 1	**Structure and Behaviour**	You model the structure and the simulation shows the model behaviour.
L3	Important techniques	Sensitivity analysis, Optimization, Model fitting. The tools Sensi & Optim do this.
Lab 2	**Model fitting, Optimization and Sensitivity analysis**	Practicing L3.
L4	Randomness, Statistical distributions and Random numbers	Basic statistics and Random numbers.
L5	Stochastic modelling	Including different kinds of uncertainties into the model.
Lab3	**Deterministic vs. Stochastic Model Building and Simulation**	Practicing L4 and L5. Showing that deterministic modelling can be misleading and insufficient.
L6	Statistical output analysis and the tool StatRes	The outcomes from a stochastic model varies, why many replications and statistical analysis are required. StatRes does this.

Continued

Lab 4	**Stochastic Modelling of Uncertainties**	Modelling different types of uncertainties. Using the StatRes tool.
L7	Experimental Design and Variance Reduction	Some more advanced issues.
Lab 5	**Time handling and its problems**	Brings insight to how the model is updated timestep by timestep, and some problems with this approach.
L8	The Modelling project	Describes the strict requirements for a modelling and simulation project.
L9	Simulation vs. Speculation	Discusses the use and limitations of modelling and simulation. The use of simulation as a crystal ball to predict the future must be condemned, not to cause false, dangerous and costly conclusions.
Exam.	**Examination** (if required). The main examination is completed laboratory exercises.	A possible examination should focus on basic concepts, understanding, how to perform a modelling project, the use and limitations of simulation. Further, the strict distinction between system under study and model, and the understanding where information comes from is crucial.
Book	**Suggested book**: "Thinking in Systems: A Primer" by Donella Meadows [11].	
Project	In a postgraduate course, a Modelling Project can be included.	