Modelling and Simulation Concepts

Hans Vangheluwe

At a first glance, it is not easy to characterize modelling and simulation. Certainly, a variety of application domains such as fluid dynamics, energy systems, and logistics management make use of it in one form or another. Depending on the context, modelling and simulation is often seen as a sub-set of Systems Theory, Control Theory, Numerical Analysis, Computer Science, Artificial Intelligence, or Operations Research. Increasingly, modelling and simulation *integrates* all of the above disciplines. Recently, modelling and simulation has been slated to become the computing *paradigm* of the future. As a paradigm, it is a way of representing problems and thinking about them, as much as a solution method. The problems span the analysis and design of complex dynamical systems. In analysis, abstract models are built inductively from observations of a real system. In design, models deductively derived from a priori knowledge are used to build a system, satisfying certain design goals. Often, an iterative combination of analysis and design is needed to solve real problems. Though the focus of modelling and simulation is on the behaviour of dynamical (*i.e.,* time-varying) systems, static systems (such as entity-relationship models and class diagrams, described in the Unified Modelling Language UML [RJB99]) are a limit-case. Both physical (obeying conservation and constraint laws) and non-physical (informational, such as software) systems and their interactions are studied by means of modelling and simulation.

1 Basic concepts

In the following, an introduction to the basic concepts of modelling and simulation is given.

Figure 1 presents modelling and simulation concepts as introduced by Zeigler [Zei84, ZPK00].

Object is some entity in the Real World. Such an object can exhibit widely varying behaviour depending on the context in which it is studied, as well as the aspects of its behaviour which are under study.

Base Model is a hypothetical, abstract representation of the object's properties, in particular, its behaviour, which is valid in *all* possible contexts, and describes all the object's facets. A base model is hypothetical as we will never —in practice— be able to construct/represent such a "total" model. The question whether a base model exists at all is a philosophical one.

System is a well defined object in the Real World under specific conditions, only considering specific aspects of its structure and behaviour.

Experimental Frame When one studies a system in the real world, the experimental frame (EF) describes experimental conditions (context), aspects, . . . within which that system and corresponding models will be used. As such, the Experimental Frame reflects the *objectives* of the experimenter who performs experiments on a real system or, through simulation, on a model. In its most basic form (see Figure 2), an Experimental Frame consists of two sets of variables, the Frame Input Variables and the Frame Output Variables, which match the system or model terminals. On the input variable side, a *generator* describes the inputs or stimuli applied to the system or model during an experiment. A generator may for example specify a unit step stimulus. On the output variable side, a *transducer* describes the transformations to be applied to the system (experiment) or model (simulation) outputs for meaningful interpretation. A transducer may for example specify the calculation of the extremal values of some of the output variables. In the above, *output* refers to physical system output as well as to the synthetic outputs in the form of internal model states measured by an observer. In case of a model, outputs may *observe* internal information such as state variables or parameters. Apart from input/output variables, a generator and a transducer,

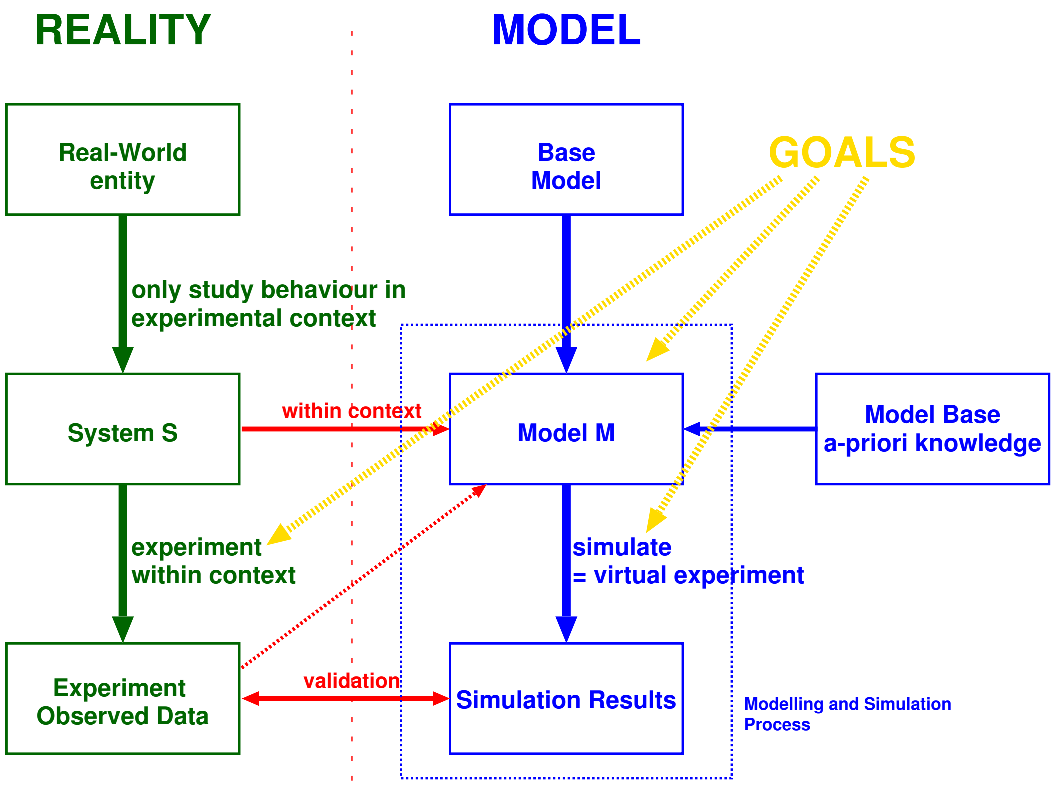
REALITY
MODEL
Real-World entity
Base Model
GOALS
only study behaviour in experimental context
System S
within context
Model M
Model Base a-priori knowledge
experiment within context
simulate = virtual experiment
Experiment Observed Data
validation
Simulation Results
Modelling and Simulation Process

Figure 1: Modelling and Simulation

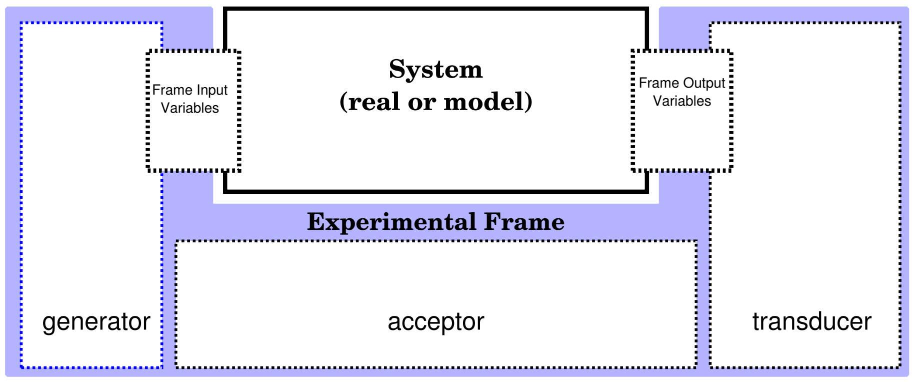

Figure 2: System versus Experimental Frame

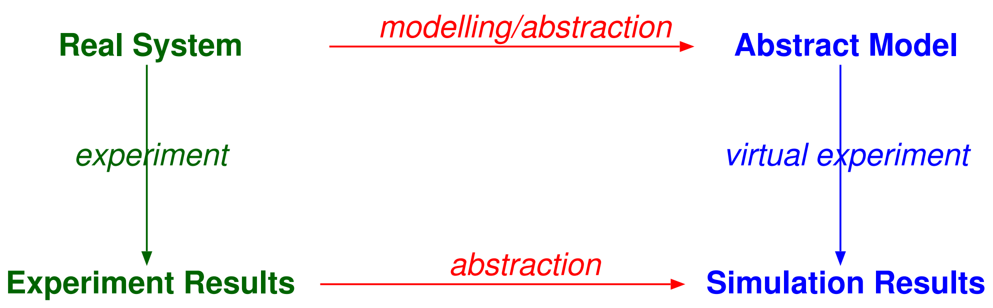

Figure 3: Modelling – Simulation Morphism

an Experimental Frame may also comprise an *acceptor* which compares features of the generator inputs with features of the transduced output, and determines whether the system (real or model) “fits” this Experimental Frame, and hence, the experimenter’s objectives.

(Lumped) Model gives an accurate description of a system within the context of a given Experimental Frame. The term “accurate description” needs to be defined precisely. Usually, certain properties of the system’s structure and/or behaviour must be reflected by the model within a certain range of accuracy. Note: a lumped model is not necessarily a lumped parameter model [Cel91]. Due to the diverse applications of modelling and simulation, terminology overlap is very common.

Experimentation is the physical act of carrying out an experiment. An experiment may interfere with system operation (influence its input and parameters) or it may not. As such, the experimentation environment may be seen as a system in its own right (which may in turn be modelled by a lumped model). Also, experimentation involves observation. Observation yields *measurements*.

Simulation of a lumped model described in a certain formalism (such as Petri Net, Differential Algebraic Equations (DAE) or Bond Graph) produces *simulation results*: the dynamic input/output behaviour. Simulation may use *symbolic* as well as *numerical* techniques. Simulation, which mimics the real-world experiment, can be seen as *virtual experimentation*, allowing one to *answer questions* about (the behaviour of) a system. As such, the particular technique used does not matter. Whereas the goal of modelling is to *meaningfully describe* a system presenting information in an understandable, re-usable way, the aim of simulation is to be *fast and accurate*. Symbolic techniques are often favoured over numerical ones as they allow the generation of *classes* of solutions rather than just a single one. For example, $Asin(x) + Bcos(x)$ as a symbolic solution to the harmonic equation $\frac{d^2x}{dt^2} = -x$ (with A and B determined by the initial conditions) is preferred over one single approximate trajectory solution obtained through numerical simulation. Furthermore, symbolic optimizations have a much larger impact than numerical ones thanks to their global nature. Crucial to the System–Experiment/Model–Virtual Experiment scheme is that there is a *homomorphic* relation between model and system: building a model of a real system and subsequently simulating its behaviour should yield the same results as performing a real experiment followed by observation and codifying the experimental results (see Figure 3). A simulation model is a tool for achieving a goal (design, analysis, control, optimisation, ...) [BO96]. A fundamental prerequisite is therefore some assurance that inferences drawn from modelling and simulation (tools) can be accepted with *confidence*. The establishment of this confidence is associated with two distinct activities; namely, verification and validation.

Verification is the process of checking the consistency of a simulation program with respect to the lumped model it is derived from. More explicitly, verification is concerned with the correctness of the transformation from some intermediate abstract representation (the conceptual model) to the program code (the simulation model) ensuring that the program code faithfully reflects the behaviour that is implicit in the specification of the conceptual model. A model compiler may automate the transformation from conceptual model to simulation model (code). If this compiler can be verified, *all* transformations by the compiler are verified.

Validation is the process of comparing experiment *measurements* with *simulation results* within the context of a certain

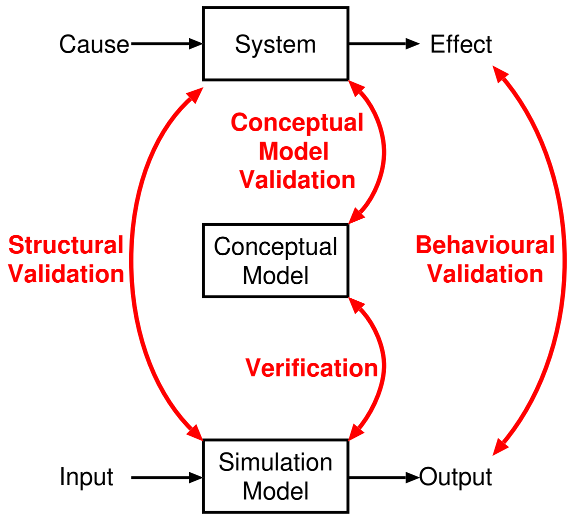

Figure 4: Verification and validation activities

Experimental Frame [Bal97]. When comparison shows differences, the formal model built may not correspond to the real system. A large number of matching *measurements* and *simulation results*, though increasing confidence, does *not prove* validity of the model however. A single mismatch between measurements and simulation results invalidates the model. For this reason, Popper has introduced the concept of *falsification* [Mag85], the enterprise of trying to falsify or disprove a model. A model may state that water in a pot boils at $100^{o}C$. Many experiments will confirm this model, until either the pot is closed or is taking to a different altitude. A falsified model should not lead to an outright rejection of the model. Rather, it should lead to a refinement of the model's experimental frame on the one hand and to an attempt to expand the model to fit the current experimental frame. In the case of the water boiling in a pot, a restricted experimental frame would state that the pressure must be constant ($1atm$). Expending the model would express the relationship between boiling point and pressure and volume.

Various kinds of validation can be identified; *e.g.,* conceptual model validation, structural validation, and behavioural validation. *Conceptual validation* is the evaluation of a conceptual model with respect to the system, where the objective is primarily to evaluate the realism of the conceptual model with respect to the goals of the study. *Structural validation* is the evaluation of the structure of a simulation model with respect to perceived structure of the system. *Behavioural validation* is the evaluation of the simulation model behaviour. An overview of verification and validation activities is shown in Figure 4. It is noted that the correspondence in generated behaviour between a system and a model will only hold within the limited *context* of the Experimental Frame. Consequently, when using models to exchange information, a model *must* always be matched with an Experimental Frame before use. Conversely, a model should never be developed without simultaneously developing its Experimental Frame. This requirement has its repercussions on the design of a model representation language.

2 The modelling and simulation process

To understand any enterprise, it is necessary to analyze the *process*: which activities are preformed, what entities are operated on, and what the causal relationships (determining activity order and concurrency) are. A *described* process gives insight, a *prescribed* process can be the basis for automation and implementation of a software tool [Hum89, HK89]. Note how a prescribed process is not necessarily deterministic as it may still leave a large number of decisions to the user. The importance of studying processes is exemplified by the SEI Capability Maturity Model (http://www.sei.cmu.edu/cmm/cmms/cmms.html) which assesses the quality of software companies by the level of knowledge, re-use, and optimization of their processes.
The simulation activity is part of the larger model-based systems analysis enterprise. A rudimentary process model for these activities is depicted in Figure 5. By means of a simple mass-spring experiment example (see Figure 6), the process will be explained. In this example, a mass sliding without friction over a horizontal surface is connected to a wall via a spring. The mass is pulled away from the rest position and let go.

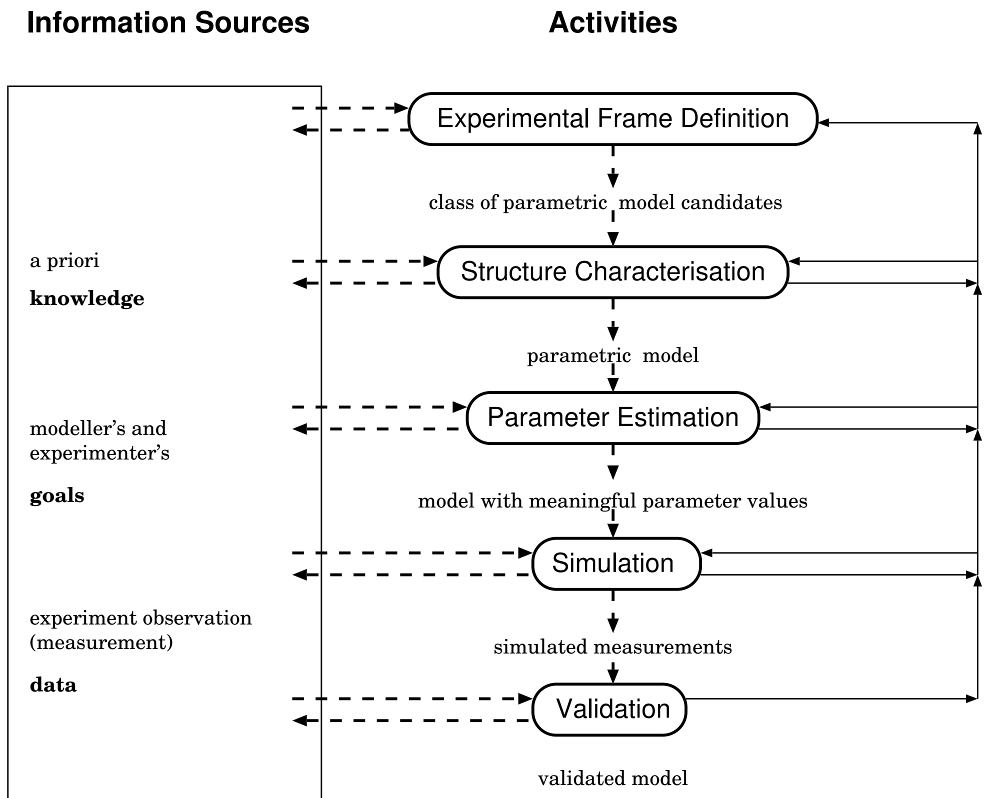
Information Sources
Activities
a priori
knowledge
modeller's and experimenter's
goals
experiment observation (measurement)
data
Experimental Frame Definition
class of parametric model candidates
Structure Characterisation
parametric model
Parameter Estimation
model with meaningful parameter values
Simulation
simulated measurements
Validation
validated model

Figure 5: Model-based systems analysis

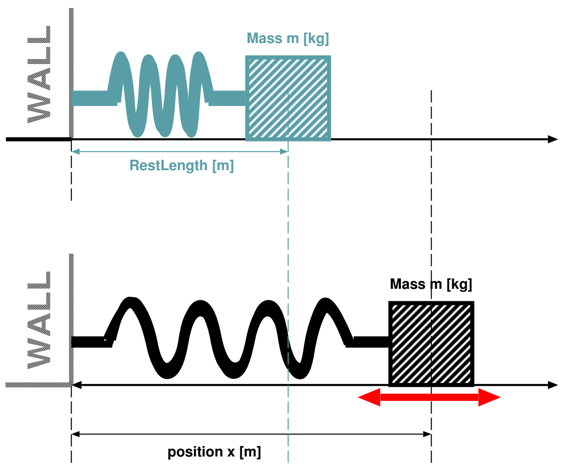

Figure 6: Mass-Spring example

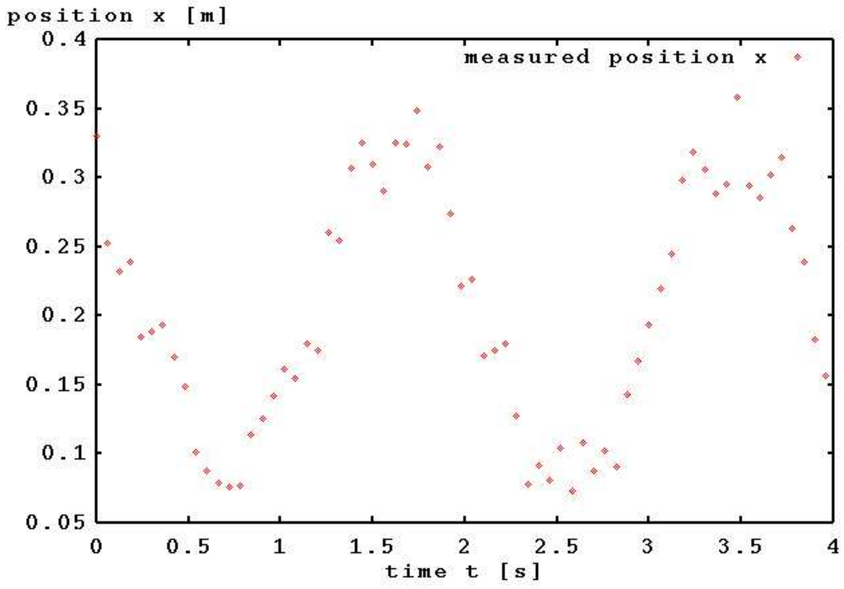

Figure 7: Measurement data

A number of *Information Sources* (either *explicit* in the form of data/model/knowledge bases or *implicit* in the user's mind) are used during the process:

1. A Priori Knowledge: in *deductive modelling*, one starts from general principles –such as mass, energy, momentum conservation laws and constraints– and deduces specific information. Deduction is predominantly used during system *design*. In the example, the a priori knowledge consists of Newton's second law of motion, as well as our knowledge about the behaviour of an ideal spring.

2. Goals and Intentions: the level of abstraction, formalisms used, methods employed, . . . are all determined by the type of *questions* we want to answer. In the example, possible questions are: "what is a suitable model for the behaviour of a spring for which we have position measurements ?", "what is the spring constant ?", "given a suitable model and initial conditions, predict the spring's behaviour", "how to build an optimal spring given performance criteria ?", . . .

3. Measurement data: in *inductive modelling*, we start from data and try to extract structure from it. This structure/model can subsequently be used in a deductive fashion. Such *iterative* progression is typical in systems *analysis*. Figure 7 plots the noisy measured position of the example's mass as a function of time.

The process starts by identifying an Experimental Frame. As mentioned above, the frame represents the experimental conditions under which the modeller wants to investigate the system. As such, it reflects the modeller's goals and questions. In its most general form, it consists of a *generator* describing possible inputs to the system, a *transducer* describing the output processing (*e.g.,* calculating performance measures integrating over the output), and an *acceptor* describing the conditions (logical expressions) under which the system (be it real or modelled) match. In the example, the experimental frame might specify that the position deviation of the mass from the rest position will/may never be larger than 10% of the rest length of the spring. Environment factors such as room temperature and humidity could also be specified, if relevant. Based on a frame, a class of matching models can be identified.

Through structure characterization, the appropriate model structure is selected based on a priori knowledge and measurement data. In the example, a *feature* of an ideal spring (connected to a frictionless mass) is that the position amplitude stays constant. In a non-ideal spring, or in the presence of friction, the amplitude descreases with time. Based on the measured data, we conclude this must be an ideal spring.

A suitable model as shown below can be built. Note how the model is non-causal (not specifying which variables are known and which need to be computed) and contains an assertion encoding the Experimental Frame acceptor.

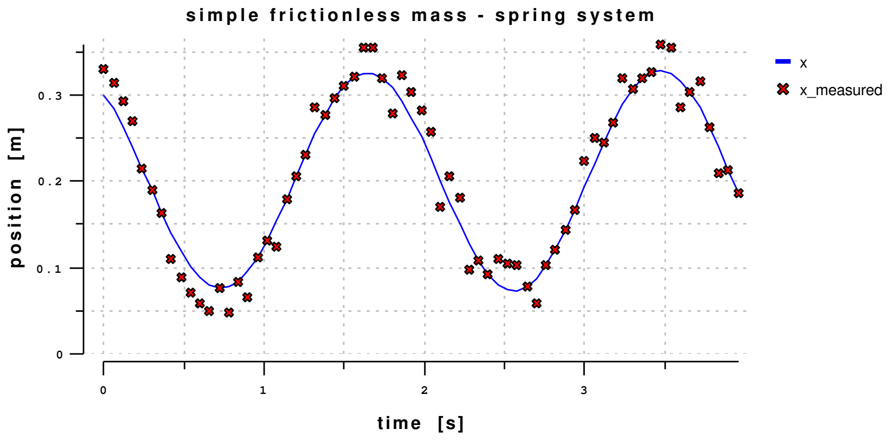

Figure 8: Fitted simulation results

```
CLASS Spring "Ideal Spring": DAEmodel ::=
{
 OBJ F_left: ForceTerminal,
 OBJ F_right: ForceTerminal,

 OBJ RestLength: LengthParameter,
 OBJ SpringConstant: SCParameter,

 OBJ x: LengthState,
 OBJ v: SpeedState,

 F_left - F_right = - SpringConstant *
                     (x - RestLength),
 DERIV([ x, [t,] ]) = v,

 EF_assert( x - RestLenght < RestLength/100),
},
```

Subsequently, during model *calibration*, parameter estimation yields optimal parameter values for reproducing a set of measurement data. From the model, a simulator is built. Due to the *contradicting aims* of modelling –meaningful model representation for *understanding and re-use*– and simulation –*accuracy and speed*–, a large number of steps may have to be traversed to bridge the gap. Using the identified model and parameters, simulation allows one to mimic the system behavior (virtual experimentation) as shown in Figure 8. The simulator thus obtained can be embedded in for example, an optimizer, a trainer, or a tutoring tool.

The question remains whether the model has predictive validity: is it capable not only of reproducing data which was used to choose the model and to identify parameters but also of predicting new behavior ? With every use of the simulator, this validity question must be asked. The user determines whether validation is included in the process. In a flight simulator, one expects the model to have been validated. In a tutor, validation by the user may be part of the education process.

In Figure 5, one notices how each step in the modelling process may introduce errors. As indicated by the feedback arrows, a model has to be corrected once falsified. A desirable feature of the validation process is the ability to provide hints as to the location of modelling errors [YVV98]. Unfortunately however, very few methods are designed to systematically provide such information. In practical use, the process is refined and embedded in more general (*e.g.,* tutoring, training, optimal experimental design, control) processes.

3 Verification and validation

The presentation of an experimental frame given above enables a rigourous definition of model *validity*. Let us first postulate the existence of a unique *Base Model*. This model is assumed to accurately represent the behavior of the Real System under *all* possible experimental conditions. This model is *universally valid* as the data $D_{RealSystem}$ obtainable from the Real System is always equal to the data $D_{BaseModel}$ obtainable from the model.

$$D_{BaseModel} \equiv D_{RealSystem}$$

A Base Model is distinguished from a *Lumped Model* by the limited experimental context within which the latter accurately represents Real System behavior.

A particular experimental frame E may be applicable to a real system or to a model. In the first case, the data potentially obtainable within the context of E are denoted by $D_{RealSystem}||E$. In the second case, obtainable data are denote by $D_{model}||E$. With this notation, a model is valid for a real system within Experimental Frame E if

$$D_{LumpedModel}||E \equiv D_{RealSystem}||E$$

The data equality $\equiv$ must be interpreted as “equal to a certain degree of accuracy”.
The above shows how the concept of validity is *not absolute*, but is related to the experimental *context* within which Model and Real System *behavior* are compared and to the *accuracy metric* used.

One typically distinguishes between the following types of model validity:

Replicative Validity concerns the ability of the Lumped Model to *replicate* the input/output data of the Real System. With the definition of a Base Model, a Lumped Model is replicatively valid in Experimental Frame E for a Real System if

$$D_{LumpedModel}||E \equiv D_{BaseModel}||E$$

Predictive Validity concerns the ability to identify the *state* a model should be set into to allow *prediction* of the response of the Real System to *any* (not only the ones used to identify the model) input segment. A Lumped Model is predictively valid in Experimental Frame E for a Real System if it is replicatively valid and

$$F_{LumpedModel}||E \subseteq F_{BaseModel}||E$$

where F_S is the set of I/O functions of system S within Experimental Frame E. An I/O function identifies a *functional relationship* between Input and Output, as opposed to a general non-functional *relation* in the case of replicative validity.

Structural Validity concerns the *structural relationship* between the Real System and the Lumped Model. A Lumped Model is structurally valid in Experimental Frame E for a Real System if it is predictively valid and there exists a morphism $\overset{\triangle}{=}$ from Base Model to Lumped Model within frame E.

$$LumpedModel||E \overset{\triangle}{=} BaseModel||E$$

When trying to assess model validity, one must bear in mind that one only observes, at any time t, $D_{RealSystem}(t)$, a subset of the potentially observable data $D_{RealSystem}$. This obviously does not simplify the model validation enterprise.
Whereas assessing model validity is intrinsically impossible, the *verification* of a *model implementation* can be done rigorously. A *simulator* implements a lumped model and is thus a source of obtainable data $D_{Simulator}$. If it is possible to prove (often by design) a structural realtionship (morphism) between Lumped model and Simulator, the following will hold unconditionally

$$D_{Simulator} \equiv D_{LumpedModel}$$

Before we go deeper into predictive validity, the relationship between different *refinements* of both Experimental Frames and models is elaborated. In Figure 9, the *derived from* relationship for Experimental Frames and the *homomorphism* re-

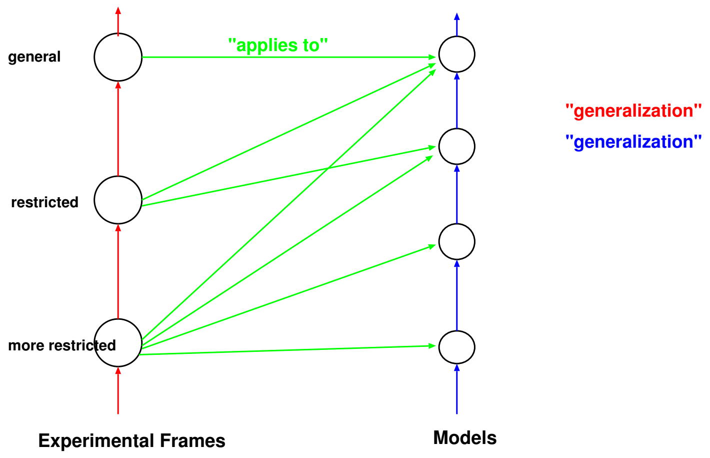

Figure 9: Experimental Frame – model relationship

lationship for Models are depicted. If we think of an Experimental Frame as a formal representation of the "context within which the model is a valid representation of the dynamics of the system", a more restricted Experimental Frame means a more specific behaviour. It is obvious that such a restricted Experimental Frame will "match" far more models than a more general Experimental Frame. Few models are elaborate enough to be valid in a very general input/parameter/performance range. Hence, the large number of "applies to" (*i.e.,* match) lines emanating from a restricted Experimental Frame. The homomorphism between models means that, when modifying/transforming a model (*e.g.,* adding some non-linear term to a model), the simulation results (*i.e.,* the behaviour) within the same experimental frame must remain the same.

Though it is meaningful to keep the above in mind during model development and use, the highly non-linear nature of many continuous models (as used in WEST++) makes it very difficult to "automate" the management of information depicted in Figure 9. Non-linear behaviour makes it almost impossible, based on a model or experimental frame symbolic representation, to make a statement about the area in state-space which will be covered (*i.e.,* behaviour). A pragmatic approach is to

1. let an "expert" indicate what the different relations are. This is based on some "insight" into the nonlinear dynamics. Such expert knowledge can be built from a large number of conducted experiments.

2. constantly –with each experiment– validate the expert information.

A crucial question is whether a model has predictive validity: is it capable not only of reproducing data which was used to choose the model and parameters but also of predicting new behavior? The predictive validity of a model is usually substantiated by comparing new experimental data sets to those produced by simulation, an activity known as model validation. Due to its special importance in the communication between model builders and users, model validation has received considerable attention in the past few decades (for a survey, see for example [Bal97]. Problems from general validation methodologies to concrete testing technologies have been extensively studied. The comparison of the experimental and simulation data are accomplished either subjectively, such as through graphical comparison, Turing test, or statistically, such as through analysis of the mean and variance of the residual signal employing the standard F statistics, Hotelling's T^2 tests, multivariate analysis of variance regression analysis, spectral analysis, autoregressive analysis, autocorrelation function testing, error analysis, and some non-parametric methods. An excellent presentation of the different issues as well as a classification of verification, validation, and testing techniques is given by Balci in [Bal97].

As indicated by the feedback arrows in Figure 5, a model has to be corrected once proven invalid. The above mentioned methods are designed to determine, through comparison of measured and simulated data, the validity of a model. As one might intuitively expect, different modelling errors usually cause the behavior of the model to deviate in different ways from that of the real system. Or, in other words, different modelling errors correspond to different "patterns" in the error signal, the difference between experimental data and simulated data. These "patterns", if extract-able, can obviously be used to identify the modelling errors. In the sequel, we present a simple biological process to introduce different

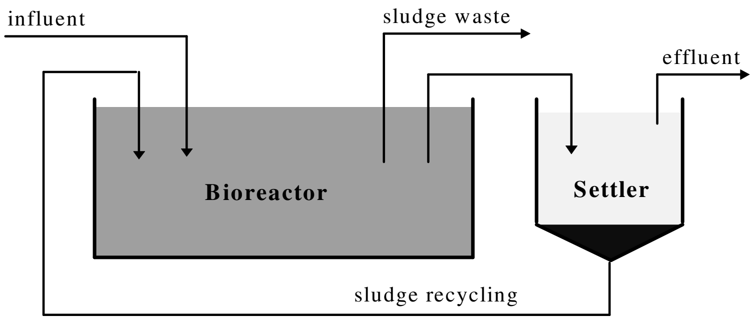

Figure 10: Biological Denitrification Process

modelling errors and their unified representation. This representation is the starting point for automated error detection [YVV98].

3.1 Modelling a biological process

Figure 10 shows a biological denitrification plant, which aims to remove the nitrate as well as the carbon organics contained in the influent water by means of biological reactions. It consists of two functional units, a bio-reactor and a settler. In the reactor, which is often completely mixed, heterotrophic biomass is present. It biodegrades the carbon organics with nitrate as the electron acceptor. The carbon organics and the nitrate are thus both removed. The 'overflow' of the reactor, containing the substrate residuals and the sludge flocks (where the biomass resides), flows into the settler. There, the sludge settles and thus separates itself from the treated water, and is subsequently recycled to the reactor through the recycling line. In order to prevent the sludge concentration in the reactor from becoming too high due to its continuous growth in the reactor, surplus sludge is removed from the waste flow (see Figure 10). Models of the denitrification process usually aim to predict the effluent quality (the amount of carbon organics and nitrate in the effluent) and the sludge production. This implies that the following three variables are crucial to the model: the carbon organics concentration, the nitrate concentration, and the biomass concentration. The main biological reaction occurring in the reactor is known to be,

$$S + NO_3^- + H^+ \overset{\longleftarrow}{\xrightarrow{r}} X + N_2 + CO_2 + H_2O$$

where $S, NO_3^-, H^+, X, N_2, CO_2$ and H_2O denote, respectively, the carbon organics, nitrate, proton, biomass, nitrogen gas, carbon dioxide gas and water. r denotes the reaction rate. The "feedback" arrow in the scheme expresses the auto-catalytic action of the biomass X. As clearly shown in the scheme, the reaction results in the removal of the nitrate and carbon organics and in the growth of the biomass. Another reactor process is the decay of the biomass which causes the decrease of the biomass on the one hand and the consumption of the nitrate on the other hand. In the context of modelling the effluent quality, the *a priori* knowledge allows one to model the process by making mass balances for the three materials,

$$\begin{aligned}
\dot{X}(t) &= \mu(t)X(t) - bX(t) - \frac{Q_w(t)}{V}X(t) \\
\dot{S}_S(t) &= -\frac{1}{Y_S}\mu(t)X(t) - \frac{Q_{in}(t)}{V}S_S(t) + \frac{Q_{in}(t)}{V}S_{S,in}(t) \qquad (1) \\
\dot{S}_{NO}(t) &= -\frac{1-Y_S}{2.86Y_S}\mu(t)X(t) - \frac{1-f_P}{2.86}bX(t) - \frac{Q_{in}(t)}{V}S_{NO}(t) + \frac{Q_{in}(t)}{V}S_{NO,in}(t)
\end{aligned}$$

where X, S_S, S_{NO} denote the biomass, the carbon organics and the nitrate concentrations in the bioreactor, respectively; $S_{S,in}$ and $S_{NO,in}$ denote the carbon organics and the nitrate concentrations in the influent, respectively; Q_{in} is the influent flow rate; Q_w is the waste flow rate; V is the volume of the bioreactor; Y_S is the yield coefficient; b is the biomass decay coefficient; f_P is the fraction of the inert materials in biomass; $\mu(t) = r(t)/X(t)$ is the specific biomass growth rate, which is still to be modelled.

Experiments show that μ is a nonlinear function of S_S and S_{NO}. It has been revealed that μ increases almost linearly with S_S and S_{NO} when they are low, but becomes independent of them when they are high. Several empirical laws have been proposed to model this relationship. The following double Monod law is commonly used [HGG+86],

$$\mu(t) = \mu_{max}(\frac{S_S(t)}{K_S + S_S(t)})(\frac{S_{NO}(t)}{K_{NO} + S_{NO}(t)}) \tag{2}$$

where μ_{max} is the maximum specific growth rate, K_S and K_{NO} are the so-called half saturation coefficients for the carbon organics and the nitrate, respectively.

Equation (1), together with equation (2), gives a parametric model of the denitrification process. All the parameters involved are plant dependent and hence have to be specifically estimated for each individual case based on the data obtained either from on-site measurements or from laboratory analyses (of on-site samples).

3.2 Different types of modelling errors and their unified representation

Assume that the paramerized model to be validated takes the form,

$$\dot{x}_m(t) = f_m(x_m(t), \theta_m, u(t), t) \tag{3}$$

where $x_m(t) \in \mathbb{R}^n$ is the state variable vector of the model, $u(t) \in \mathbb{R}^p$ is the input vector, and θ_m is the model parameter vector, which is known. On the basis of this model, the real behavior of the system can generally be represented as,

$$\dot{x}_r(t) = f_m(x_r(t), \theta_m, u(t), t) + e_m(t) \tag{4}$$

where $x_r(t) \in \mathbb{R}^n$ is the state vector of the system, $e_m(t) \in \mathbb{R}^n$ is the modelling error vector. It is assumed in equation (4) that the real system has the same number of state variables as the model. This representation does not limit the generality of the representation since the errors introduced by erroneous state aggregations in deriving model (3) can also be represented by the error term $e_m(t)$.

In order to make the modelling error identification possible, an appropriate representation of the error term $e_m(t)$ in equation (4) is required. This representation should be obtained by making use of the *a priori* knowledge about the possible modelling errors. Basically, modelling errors may be introduced in each stage of the modelling process as depicted in Figure 5. In this section, it will be shown, taking the biological model developed in the previous section as an example, how the *a priori* knowledge concerning the modelling errors can be obtained through the analysis of the modelling process and the model itself. The mathematical representation of the modelling errors will also be discussed. As will be shown in the next section, such a representation allows the identification of the modelling errors based on the comparison of the observed data with data produced by simulation of the erroneous model.

3.2.1 Modelling Errors due to an improperly defined Experimental Frame

In defining the boundaries of the process or system to be modelled, some important components may be missed, some significant disturbances to the system may be improperly neglected and so on. All of these introduce errors into the model. The Experimental Frame is the formalisation of the experimental conditions (inputs applied to the system, outputs observed, criteria of acceptance, ...) and as such the above mentioned modelling errors can be formally expressed as Experimental Frame errors. For a rigorous treatment, see [Tak96].

For instance, an assumption underlying model (1) is that no other reactions occur in the process which affect the mass balance of the concerned materials. One knows, however, that this assumption is not valid when dissolved oxygen is present in the influent. In fact, when dissolved oxygen is fed to the bioreactor, the following reaction, which is called the aerobic oxidation, will also occur, accompanying the denitrification reaction described in the previous section,

$$S + O_2 \overset{\longleftarrow}{\xrightarrow{r_o}} X + CO_2 + H_2O$$

where r_o denotes the oxidation reaction rate. The reaction scheme clearly shows how the aerobic oxidation affects the mass balance of the carbon organics and the biomass. This will inevitably introduce errors in the prediction of these two variables. Since, as shown in model (1), both $S_S(t)$ and X appear in the equation concerning the dynamics of S_{NO}, the prediction of the nitrate concentration will be affected indirectly.

A characteristic of the modelling error described above is that it does not directly affect the third equation in model (1). The above aerobic oxidation introduces an r_o term into the first equation of (1) and an $\frac{1}{Y_S}r_o$ term into the second equation. The modelling error term in equation (4) takes the following form,

$$e_{m,o}(t) = [1 \ -\frac{1}{Y_S} \ 0]^T r_o(t) \tag{5}$$

While $[1 \ -\frac{1}{Y_S} \ 0]^T$ is apparently a known vector, $r_o(t)$ is an unknown, time-variant scalar.

3.2.2 Modelling Errors due to an improperly characterized Model Structure

Due to for instance lack of knowledge of the mechanism of the process to be modelled, or due to an oversimplification of the model, one may assume a wrong model structure. Typical errors include choosing an incorrect number of state variables or incorrectly assuming non-linear behavior. Structural errors may accidentally be produced through incorrect choice of parameters (usually, 0), whereby some part of the model structure vanishes, thereby altering the model structure.

For instance, in model (1), there does not exist a fundamental law that precisely characterizes the dependence of the denitrification reaction rate on the concentrations of the materials. The "laws" which have hitherto been proposed are all quite empirical. A problem of this type of laws is that they have a limited applicability range. An inappropriate choice of the "laws" may introduce errors. For example, when the model of the denitrification rate given in equation (2) is not a good description of the real reaction rate: $\mu_r(t) = \mu(t) + \delta\mu_s(t)$, where $\mu_r(t)$ is the real specific reaction rate and $\delta\mu_s(t)$ is the modelling error, the following error term is found by substitution in equation (4),

$$e_{m,\mu}(t) = [1 \ -\frac{1}{Y_S} \ -\frac{1-Y_S}{2.86Y_S}]^T \delta\mu_s(t)X(t) \tag{6}$$

3.2.3 Modelling Errors due to inaccurate estimates of the Model Parameters

Either by improper or inadequate data used for parameter estimation or by ill designed estimation algorithms, one may use incorrect parameter values. The error terms in equation (4) due to the estimate errors of the parameters in model (1) are as follows,

modelling error of b

Assuming $b_r = b + \delta b$, where b_r is the real decay coefficient and δb is the modelling error, one obtains,

$$e_{m,b}(t) = [-1 \ 0 \ -\frac{1-f_P}{2.86}]^T \delta b X(t) \tag{7}$$

modelling error of f_P

Assuming $f_{P,r} = f_P + \delta f_P$, where $f_{P,r}$ is the real inert fraction in a biomass cell and δf_P is the modelling error, one obtains,

$$e_{m,f_P}(t) = [0 \ 0 \ 1]^T \frac{\delta f_P}{2.86} bX(t) \tag{8}$$

modelling error of Y_S

Assuming $\frac{1}{Y_{S,r}} = \frac{1}{Y_S} + \delta(\frac{1}{Y_S})$, where $Y_{S,r}$ is the real yield coefficient and $\delta(\frac{1}{Y_S})$ is the modelling error, one obtains,

$$e_{m,Y_S}(t) = [0 \ -1 \ -\frac{1}{2.86}]^T \delta(\frac{1}{Y_S})\mu(t)X(t) \tag{9}$$

modelling errors of μ_{max}, K_{NO} and K_S

Assuming $\mu_r(t) = \mu(t) + \delta\mu_p(t)$, where $\mu_r(t)$ is the real specific reaction rate and $\delta\mu_p(t)$ is the error caused by the modelling error of μ_{max}, K_{NO} or K_S, one obtains,

$$e_{m,\mu_{max},K_{NO},K_S}(t) = [1 \ -\frac{1}{Y_S} \ -\frac{1-Y_S}{2.86Y_S}]^T \delta\mu_p(t)X(t) \tag{10}$$

One finds that every single modelling error shown above takes the form of a product of a known constant vector and an unknown time-variant variable. This is not an artifact of this particular example, but is in fact a general property. Usually, each modelling error affects only a subspace of the n-dimensional state space, and can hence be represented in equation (4) with a term $F_i d_i(t)$, where $F_i \in R^{n \times s_i}$, $d_i(t) \in R^{s_i}$. The vectors of F_i span the subspace affected by the concerned modelling error. F_i is called the feature vector or feature matrix of the modelling error. $d_i(t)$ represents the magnitude of the modelling error, and is generally unknown and time-varying. Thus, equation (4) can be rewritten as,

$$\dot{x}_r(t) = f_m(x_r(t), \theta_m, u(t), t) + \sum_{i=0}^{l} F_i d_i(t) \tag{11}$$

Since it is usually not possible to predict all possible modelling errors, it is necessary to include a special feature matrix, say F_0, in equation (11) to represent modelling errors which were not explicitly modelled. Obviously, the n-dimensional identity matrix is suitable for that purpose.

To allow for meaningful error identification, some assumptions are made with respect to equation (11):

- The individual errors are written in “additive” form:

  $$v_r = v + \delta v$$

  Such a “choice” of individual error terms is always possible without loss of generality. One may be required to “lump” non-linear errors as in $\delta(Y_S)$ or $\delta\mu_p$ above.
- Simultaneously occurring errors are assumed to be either additive, or sufficiently small to allow for an linear approximation:

  $$f(A + \delta A, B + \delta B) \cong f(A,B) + \frac{\partial f}{\partial A}(A,B)\,\delta A + \frac{\partial f}{\partial B}(A,B)\,\delta B$$

  Though such an assumption is not necessary *per se*, as non-linear effects can always be lumped into an extra error term (using the above mentioned F_0), this would defeat our purpose of isolating individual error contributions.

4 Abstraction levels and formalisms

There are several reasons why abstract models of systems are used. First of all, an abstract model description of a system *captures knowledge* about that system. This knowledge can be stored, shared, and re-used. Furthermore, if models are represented in a standard way, the *investment* made in developing and validating models is paid off as the model will be understood by modelling and simulation environments of different vendors for a long time to come.
Secondly, an abstract model allows one to formulate and answer *questions* about the structure and behaviour of a system. Often, a model is used to obtain values for quantities which are non-observable in the real system. Also, it might not be financially, ethically or politically feasible to perform a real experiment (as opposed to a simulation or virtual experiment). Answering of *structure related* questions is usually done by means of symbolic analysis of the model. One might for example wish to know whether an electrical circuit contains a loop. Answering of questions about the *dynamic behaviour* of the system is done (by definition) through *simulation*. Simulation may be symbolic or numerical. Whereas the aim of modelling is to provide insight and to allow for re-use of knowledge, the aims of simulation are accuracy and execution speed (often real-time, with hardware-in-the-loop).

One possible way to construct systems models (particularly in systems design) is by copying the structure of the system. This is not a strict requirement. A neural network which simulates the behaviour of an aeration tank in an activated sludge waste water treatment plant is considered a “model” of the tank. It may accurately replicate the behaviour of the tank, though the physical structure of the tank and its contents is no longer apparent. For purposes of control, we are often satisfied with a performant (real-time) model of a system which accurately predicts its behaviour under specific circumstances, but bears no structural resemblance with the real system.

Abstract models of system behaviour can be described at different levels of abstraction or detail as well as by means of different formalisms. The particular formalism and level of abstraction used depend on the background and goals of the modeller as much as on the system modelled. As an example, a temperature and level controlled liquid in a pot is considered as shown in Figure 11. This is a simplified version of the system described in [BZF98], where structural

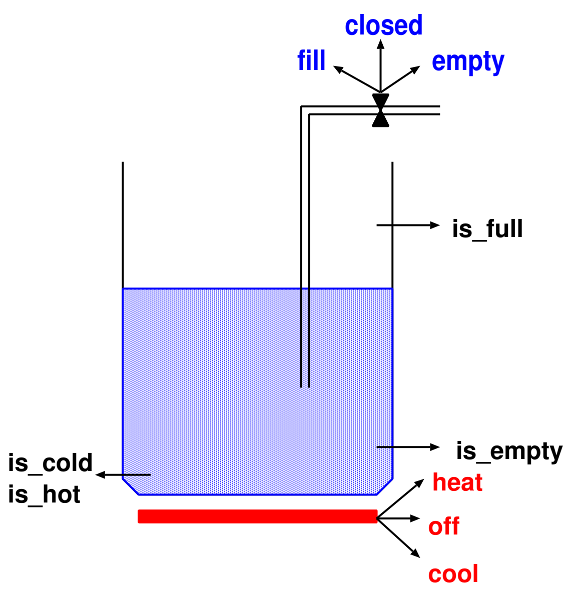

Figure 11: T,l controlled liquid

change is the main issue. On the one hand, the liquid can be heated or cooled. On the other hand, liquid can be added or removed. In this simple example phase changes are not considered. The system behaviour is completely described by the following (hybrid) Ordinary Differential Equation (ODE) model:

Inputs (discontinuous $\rightarrow$ hybrid model):

- Emptying, filling flow rate $\phi(t)$
- Rate of adding/removing heat $W(t)$

Parameters:

- Cross-section surface of vessel A
- Specific heat of liquid c
- Density of liquid ρ
- Temperature of incoming liquid T_{in}

State variables:

- Temperature T
- Level of liquid l

Outputs (sensors):

- $is\_low, is\_high, is\_cold, is\_hot$

$$\left\{\begin{array}{rcl}
\frac{d\mathsf{T}}{dt} & = & \frac{1}{l}[\frac{W}{c\rho A} - \phi(\mathsf{T}-\mathsf{T}_{in})] \\
\frac{dl}{dt} & = & \phi \\
is\_low & = & (l < l_{low}) \\
is\_high & = & (l > l_{high}) \\
is\_cold & = & (\mathsf{T} < \mathsf{T}_{cold}) \\
is\_hot & = & (\mathsf{T} > \mathsf{T}_{hot})
\end{array}\right.$$

The inputs are the filling (or emptying if negative) flow rate ϕ, and the rate W at which heat is added (or removed if negative). This system is parametrized by A, the cross-section surface of the vessel, H, its height, c, the specific heat of the liquid, and ρ, the density of the liquid. The state of the system is characterized by variables T, the temperature and l, the level of the liquid. The system is observed through threshold output sensors $is\_low, is\_high, is\_cold, is\_hot$. Given input signals, parameters, and a physically meaningful initial condition (T_0, l_0), simulation of the behaviour yields a continuous state trajectory as depicted in Figure 12. By means of the binary (on/off) level and temperature sensors introduced in the differential equation model, the state-space may be discretized. The inputs can be abstracted to heater heat/cool/off and pump fill/empty/closed. At this level of abstraction, a Finite State Automaton (with 9 possible states) representation of the dynamics of the system as depicted in Figure 13 is most appropriate. Though at a much higher level of abstraction, this model is still able to capture the essence of the system's behaviour. In particular, there is a *behaviour morphism* between both models: model discretization (from ODE to FSA) followed by simulation yields the same result as simulation of the ODE followed by discretization. This morphism is shown as a commuting diagram in Figure 14.

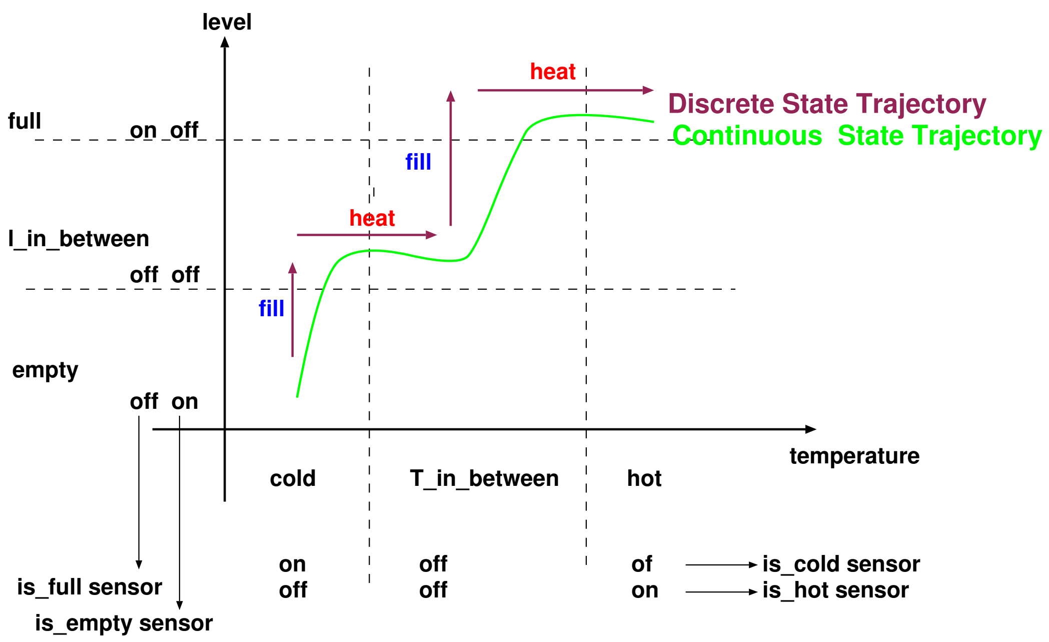

Figure 12: Trajectories

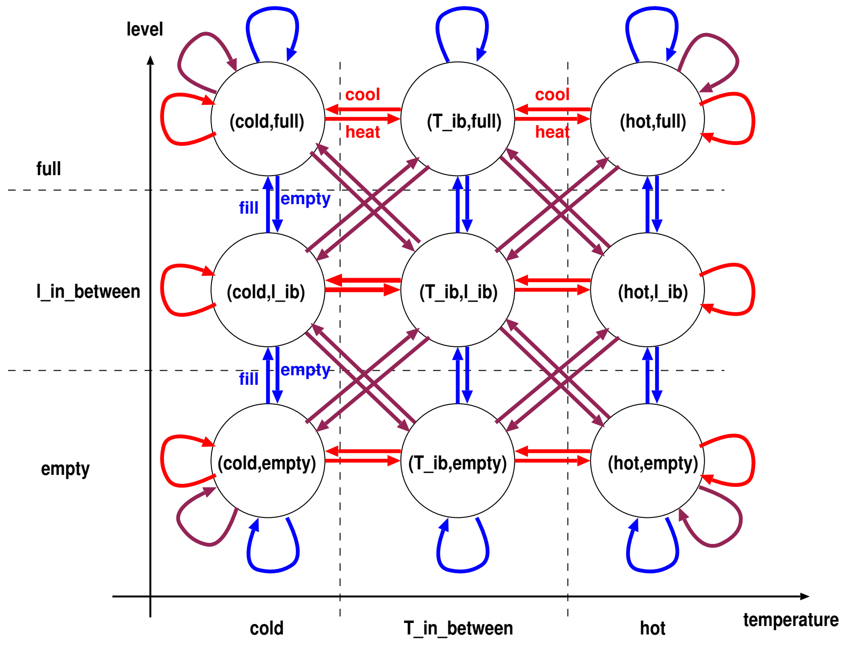

Figure 13: FSA formalism

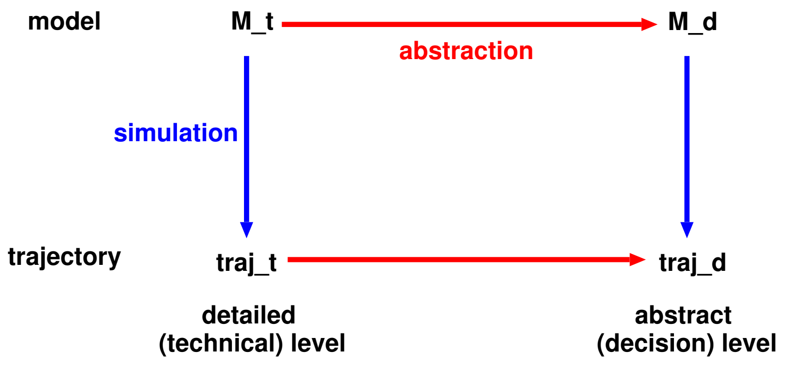

Figure 14: Behaviour morphism

References

[Bal97] Osman Balci. Principles of simulation model validation, verification, and testing. *Transactions of the Society for Computer Simulation International*, 14(1):3–12, March 1997. Special Issue: Principles of Simulation.

[BO96] Louis G. Birta and F. Nur Özmizrak. A knowledge-based approach for the validation of simulation models: The foundation. *ACM Transactions on Modeling and Computer Simulation*, 6(1):76–98, January 1996.

[BZF98] Fernando J. Barros, Bernard P. Zeigler, and Paul A. Fishwick. Multimodels and dynamic structure models: an integration of DSDE/DEVS and OOPM. In D.J. Medeiros, E.F. Watson, and Manivannan M.S., editors, *Proceedings of the 1998 Winter Simulation Conference*, pages 413–419. Society for Computer Simulation International (SCS), 1998.

[Cel91] François E. Cellier. *Continuous System Modeling*. Springer-Verlag, New York, 1991.

[HGG+86] M. Henze, C.P.L. Grady, W. Gujer, G.v.R. Marais, and T. Matsuo. Activated sludge model no. 1. Scientific and Technical Report 1, International Association on Water Pollution Research and Control (IAWPRC, now IAWQ), London, England, July 1986. IAWPRC Task Group on Mathematical Modelling for Design and Operation of Biological Wastewater Treatment.

[HK89] Watts S. Humphrey and Marc I. Kellner. Software process modeling: Principles of entity process models. In *Proceedings of the 11th International Conference on Software Engineering*, pages 331–342, Pittsburg, PA, May 15-18 1989. ACM.

[Hum89] Watts S. Humphrey. *Managing the Software Process*. SEI Series in Software Engineering. Addison-Wesley, October 1989.

[Mag85] Brian Magee. *Popper*. Fontana Press (An Imprint of HarperCollins Publishers), London, 1985.

[RJB99] James Rumbaugh, Ivar Jacobson, and Grady Booch. *The Unified Modeling Language Reference Manual*. Object Technology Series. Addison-Wesley, 1999.

[Tak96] Shingo Takahashi. General morphism for modeling relations in multimodeling. *Transactions of the Society for Computer Simulation International*, 13(4):169–178, December 1996.

[YVV98] Zhiguo Yuan, Hans Vangheluwe, and Ghislain C. Vansteenkiste. An observer-based approach to modeling error identification. *Transactions of the Society for Computer Simulation International*, 15(1):20–33, 1998.

[Zei84] Bernard P. Zeigler. *Theory of Modelling and Simulation*. Robert E. Krieger, Malabar, Florida, 1984.

[ZPK00] Bernard P. Zeigler, Herbert Praehofer, and Tag Gon Kim. *Theory of Modelling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems*. Academic Press, second edition, 2000.