Verifying Trace Inclusion between an Experimental Frame and a Model

V. Albert[1], A. Nketsa[1] and C. Seguin[2]
[1]CNRS; LAAS; 7 avenue du colonel Roche, F-31077 Toulouse, France
Université de Toulouse; UPS, INSA, INP, ISAE; LAAS; F-31077 Toulouse, France
[2]DCSD / CD; ONERA - Centre Midi-Pyrénées; 2, av. Edouard Belin, BP 4025 - 31055 Toulouse Cedex 4, France
valbert@laas.fr, alex@laas.fr, christel.seguin@onera.fr

Keywords: Experimental Frame, Abstraction, Applicability, Component, Formal Matching

Abstract

The concept of experimental frame is employed to define circumstances under which a model is simulated and observed. Verifying the applicability of an experimental frame to a model is a key part to ensure that the intended purpose of a simulation can be reach. We assume that the specification of an experimental frame and a model are two software components in the formal sense. This paper suggests a component-based approach to build the right simulation used for system models verification and validation. We use Input/Output automata for components' behaviour specification. Behavioural signature is employed to specify the experimental frame and model capabilities. Then we formally define applicability conditions with matching rules to verify the trace inclusion between an experimental frame and a model. This approach is illustrated with an application case, i.e. autonomous intelligent cruise controller.

1. INTRODUCTION

The concept of model abstraction is described by F.K. Frantz [1] as a method used to reduce the complexity of a real system while maintaining the validity of the simulation results **with respect to the questions the simulation is supposed to answer**. An abstraction is valid if it maintains the validity of the results of the simulation, which depend on the questions raised, i.e. the intended purpose. Thereby, capturing the dual relationships between a model and its intended purpose is a key part of the M&S process [2].

In that context, challenging issues in Modelling and Simulation (M&S) are mostly related to the specification/documentation of both, models capabilities and properties expected from these models to reach the intended purpose. There are many works which suggests processes and good practices for such documentation [3] [4] [5] [6]. Those proposals, and in a broader scope, the M&S process, cruelly lack of formal methods. Some works have been done in integrating formal method in M&S [7] [8]. The idea is to construct a formal specification of models and to perform formal analysis on this formal specification. Furthermore, if a formal specification of the intended purpose is also provided, model validation become feasible by formal analysis too.

This paper describes a formal approach to verify that the system behaviours which are included in the model allows to reach the expected behaviours of the intended purpose. We first describe such a property within the well-established M&S framework suggested by B.P. Zeigler [9]. Then, we redefine this property on a component-based approach to incorporating formal methods. We suggest using behavioural typing [10] [11] [12] to specify the data of interest for formal analysis. Finally, we establish matching rules for mapping between model's behaviours and simulation's expected behaviours. We illustrate our approach with an autonomous intelligent cruise controller system.

2. TRACE INCLUSION PROPERTY WITHIN THE M&S FRAMEWORK

2.1. M&S framework

Figure 1 illustrates the framework for Modelling & Simulation suggested by B.P. Zeigler [9] and its entities. The system is the real or virtual element used as a source of observable data and subject to modelling. Through a modelling activity we obtain a model which is typically a set of instructions, controls, equations or constraints to generate its behaviour. Modelling is the process of abstraction of some aspects of the structure or the behaviour of the real system regarding some questions. Those questions can be rigorously described with the concept of experimental frame. The experimental frame is a specification of the conditions in which a system is observed or experimented on. As suggested by the author, it can be seen as a system that interacts with the system to obtain the data of interest in given conditions.

Since a model is only a partial/simplified/abstract representation of the real system we used to call it the substitute of the real system. The *validity* of a model is defined as the degree to which a model faithfully represents a system in an experimental frame of interest.

The simulation is a method allowing to approximate the behaviour of the real system by executing the model over time. The simulator is a computer system used to execute the model and generate its behaviour based on model instructions and injected inputs.

A simulator is correct if it faithfully generates the model output values given the model state and the input values. The

correctness of a simulator refers to the principle of separating preoccupations between model design and its implementation.

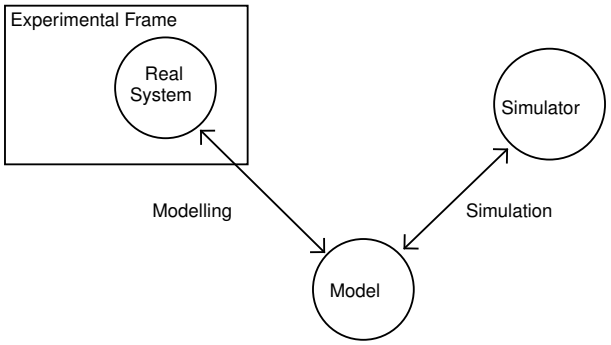

Figure 1. M&S process and its entities

2.2. Zoom on the concept of experimental frame

An experimental frame has three components as illustrated figure 2: a *generator* which generates a set of input segments for the system; an *acceptor* which selects the data of interest of the system while monitoring whether the desired experimental conditions are complied with and a *transducer* which observes and analyses the output segments of the system.

An experimental frame is given in [13] as a structure $\langle T, X, Y, \Omega_X, \Omega_Y, SU \rangle$ where T is a time base, X is the set of simulation observation points, Y is the set of simulation control points, $\Omega_Y \subseteq (Y, T)$ is the set of segments injected onto the model inputs, $\Omega_X \subseteq (X, T)$ is the set of segments observed onto the model outputs and SU is a set of conditions, also called summary mappings, which establish relationships between inputs and outputs within the frame. A summary mapping can be seen as a set of pre- and post-conditions mapping the stimulation variables to the observation variables and so on monitoring the experimentation. If, for example, we consider the Boolean observation point *alarm* in the experimental frame: "If the value of *alarm* is true in one state of the execution, then there is a previous state in the execution where $altitude \leq 10$" is a summary mapping.

2.3. M&S relations

Besides validity and correctness concepts, B.P. Zeigler defined three fundamental relationships: *morphism*, *applicability* and *derivability*.

2.3.1. Morphism

A morphism relation establishes a correspondence between a "concrete" model and an "abstract" or simplified model, the abstract model being the substitute of the concrete model.

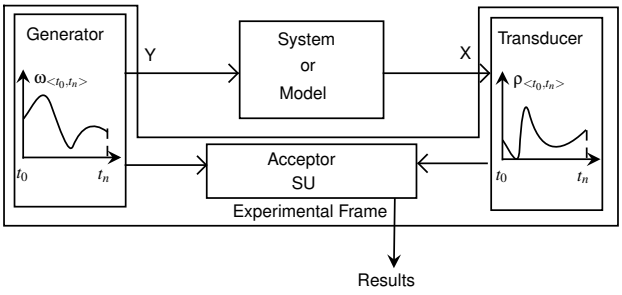

Figure 2. The experimental frame and its components

Then we can consider a set of models of a same system hierarchized by a *morphism* relation. The concrete model is a model with more capabilities, meaning that it can be used for a greater number of experimental frames. However, for a given experimental frame, the abstract model can be as capable as the concrete model. What must be remembered is that, for a given experimental frame, an abstract model must be as valid as the concrete model.

If we consider, as in Figure 3 [9], two systems S and S' such that S is greater than S' in terms of the number of states. There has been abstraction between S and S' by state aggregation. For each state of system S', a correspondence is established, illustrated by bold print arrows, with one or a set of system S states. This matching is called a *homomorphism* if, when S' performs a transition, for example $a \xrightarrow{\alpha'} b$ then S performs the transition concerning the corresponding states, i.e. $A \xrightarrow{\alpha} B$. We will say that the transition function has been preserved. Similarly, a homomorphism implies the preservation of the output function. This means that the output produced by system S' in status q, for example at state b, must be the same as the output produced by system S in the corresponding state, in this case B. If two systems are homomorphic, they have equivalent behaviour.

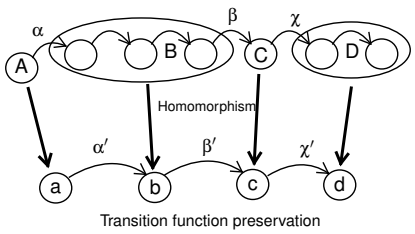

Figure 3. Morphism by state aggregation

2.3.2. Applicability and derivability

The applicability relation determines whether an experimental frame can be applied to a model. This relation is very important as it serves to state whether a use objective can

be reached with a specific M&S application. Only few models can implement experimentation conditions required by an experimental frame used to reach objectives and may possibly supply valid simulation results. B.P. Zeigler also defines a derivability relation between experimental frames. This relation refers to the degree to which an experimental frame defines more restrictive conditions (which allow fewer observations) than another. Figure 4 [9] below illustrates the morphism, applicability and derivability relations. Experimental frame EF4, not very restrictive, applies to models M1, M2 and M3. Model M4 is too abstract to accommodate EF4 as well as experimental frames EF1, EF2 and EF3 which are more restrictive than EF4. EF2 is applicable to Ml. EF3, which is less restrictive than EF2 is therefore also applicable. In this case, we can say that M1 is too concrete or complex for the given experimental frame but not less valid. No models can accommodate EF1.

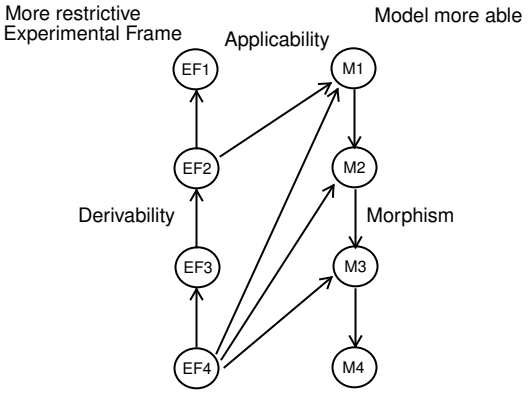

Figure 4. Morphism, applicability and derivability relations

2.4. EF-Model applicability according to their behaviours

A system is a structure

$$S = \langle T, X_S, \Omega_S, Y_S, Q, \Delta, \Lambda \rangle \text{ where}$$

- T is the time base,
- X is the set of input values,
- $\Omega \subseteq (X, T)$ is the set of acceptable input segments,
- Y is the set of output values,
- Q is the set of states,
- $\Delta : Q \times \Omega \to Q$ is the transition function,
- $\Lambda : Q \times X \to Y$ is the output function.

A system is an entity characterised by a set of states Q ordered on a time base T which reacts to a set of input segments ω on X, originating from its environment. The future of the system is determined according to the current state and an input segment, by the transition function Δ. The output function Λ defines a set Ω_{Y_M} of output trajectory ρ on Y given a current state and an input segment.

Consider an experimental frame $EF = \langle T, X_{EF}, Y_{EF}, \Omega_{X_{EF}}, \Omega_{Y_{EF}}, SU \rangle$ and a model $M = \langle T, X_M, \Omega_{X_M}, Y_M, Q, \Delta, \Lambda \rangle$. EF specifies the executions required to correctly perform a specific experimentation (observation of a system of interest). Then $\Omega_{Y_{EF}}$ are the simulation results that the experimenter wishes to observe, whereas $\Omega_{X_{EF}}$ are the stimuli injected in the simulation. Similarly, Ω_{X_M} are model acceptable inputs segments and all $\rho \in \Omega_{Y_M}$ are the possible simulation results.

We say that an experimental frame is applicable to a model according to their behaviours if:

$$\Omega_{Y_{EF}} \subseteq \Omega_{X_M}$$

$$\Omega_{Y_M} \subseteq \Omega_{X_{EF}}$$

3. COMPONENT-BASED APPROACH FOR TRACE INCLUSION VERIFICATION

We suggest a component-based approach to describe a M&S product. We say that the experimental frame and the model of the system of interest are two components as defined by [14]. A component has a set of ports and is coupled to another through connectors via their respective ports. A component can be either coupled, i.e. it results from a components' composition or atomic, i.e. it cannot be decomposed any more.

Behavioural type systems have been defined in recent years with the aim to be able to check the compatibility of communicating concurrent objects, not only regarding data exchanged, but also regarding the matching of their respective behaviour. A behavioural type provides a notion of compliance of components to their interfaces, namely a signature. By signature analysis we can verify such compliance, in our case the applicability of the experimental frame to the model.

3.1. Component's behaviour specification

We use IOA (Input/Output Automata) [15] to describe the dynamic behaviour of components.

An IOA is an n-uplet

$$A = \langle D, I, O, dom, S, \sigma, \Sigma, \delta, S_0 \rangle \text{ where}$$

- D is a set of data types (integer, real, string, etc.),
- I is a finite set of names representing the set of input variables,

- O is a finite set of names representing the set of output variables,
- $dom : I \cup O \rightarrow D$ is the typing function which allocates a type of data to each variable,
- S is the set of states,
- $\sigma : S \rightarrow ((I \cup O) \rightarrow dom(I \cup O))$ is the configuration function, i.e. mapping of values with the typed input/output variables for each state.
- Σ is the set of event names. These names are the state transition labels. ε is the set of non-observable events.
- $\delta \subset S \times \Sigma \times S$ is the transition relation, a transition identifies a change to the inputs, the outputs or the states.
- $S_0 \subset S$ is the set of initial states possible, the states in which the automaton can be if no inputs have yet been transformed.

$I \cup O \cup \Sigma$ is the vocabulary used to define the automaton. For the sake of simplicity, we will assume that the names used in the vocabulary are unique.

Example If, for example, we consider a communication system with a shared memory, the behaviour of which is described by the IOA *buffer* in Figure 5. In this example, a variable queue is defined to describe a sequence of messages. This variable is declared as a table of length `len` that can contain m messages of type id_m with an initial value `empty`. This automaton is composed of an initial state s_0, where the sequence is empty, and a state s_1 where the sequence is not empty. These states are connected by transition relations. These transitions are triggered by two types of events:

- `send(m)` is an input event where each message is added to the head of the sequence `queue` using the configuration function `add(m)`,
- `recv(m)` is an output event, where each message is deleted from the sequence `queue` using the configuration function `remove()`.

The configuration functions modify the value of the variable `queue` for each state. For example, if `len=2`, `queue` can have the values $(0,0),(id_{m1},0),(id_{m1},id_{m2}),(id_{m2},0)$.

In the IOA theory, we can synchronise an IOA output event with the input event of another IOA if these events have the same name. For example, if output event `send(m)` appears in an IOA then this event can be synchronised with input event `send(m)` of another IOA. Figure 5, for example, illustrates two other IOAs *client* and *serveur* which exchange messages via *buffer*.

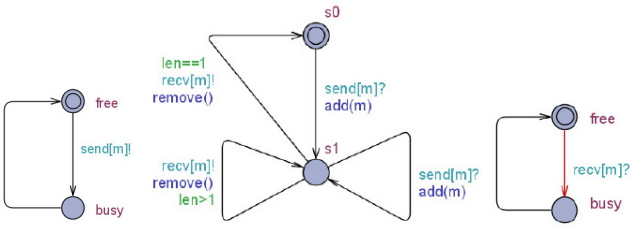

Figure 5. Automaton of a shared memory communication system

3.2. Behavioural signature specification

Definition 3..1 (Run (or trace))**.** *An IOA run is a sequence (finite or not)* $\sigma(s_0)l_0 \ldots \sigma(s_i)l_i \ldots$ *where* s_i *is a state (*s_0 *is an initial state) and* l_i *is an event name such that any* $(s_i\ l_i\ s_{i+1})$ *of a run is a transition of the automaton. A run can also be defined by a path taken from the computation tree of an automaton.*

We call a temporal interfacing constraint an execution property defined with respect to a given vocabulary (names of events and input/output variables).

The properties of invariants are constraints on the configuration function. For example, if we consider *altitude* to be an input variable of the automaton's vocabulary. The constraint "variable *altitude* is positive or zero" is an example of an invariant. The automaton that satisfies this invariant includes an integer input variable called *altitude* and a configuration function σ such that, for any state s, $\sigma(s)(altitude) \leq 0$.

The temporal properties are constraints which link the configuration and the transitions. There are two separate types of time-dependent properties:

- linear versus computation tree: refers to the constraints on all executions of the automaton versus those on the computation tree taken from the automaton,
- past versus future.

If, for example, we consider the Boolean output variable *alarm* in the vocabulary of an automaton: "If the value of *alarm* is true in one state of the execution, then there is a previous state in the execution where $altitude \leq 10$"" is a linear time and past property. "If $altitude \leq 10$ in one node of the computation tree of an automaton, then we should be able to find a path and a subsequent node in the path which satisfies $alarm = true$"" is an example of a computation tree and future property.

Temporal properties with a vocabulary which refers to clock variables are timed properties. "If $altitude \leq 10$ was true for at least ten units of time then $alarm = true$" is an example of a timed property.

Definition 3..2 (Execution set satisfied by a TIC)**.** *If we have an automaton A and TIC a constraint defined on the vocabulary included in the vocabulary of A. We note $\|TIC\|_A$ the set of executions of A which satisfies the TIC.*

3.3. Matching rules for signature analysis

We propose to formalise an EF by a TIC on the vocabulary $I_{EF} \cup O_{EF} \cup \Sigma_{EF}$. The EF inputs therefore are the simulation results that the experimenter wishes to observe, whereas the EF outputs are the stimuli injected in the simulation. We suggest to formalise a model of a system of interest by a TIC on a vocabulary $I_M \cup O_M \cup \Sigma_M$. The model inputs must be supplied by the outputs of the experimental frame whereas the model outputs are all the possible simulation results.

Model and experimental frame can be connected if they have compatible vocabularies:

- $I_{EF} \subseteq O_M$: all the results of interest of the experimental frame are supplied by the simulation and the simulation may supply more results than necessary.
- $O_{EF} = I_M$: all the stimulations planned by the experimentation can be performed and all the inputs necessary to perform the simulation are defined by the experimentation (the exact matching is required here since non-assigned model inputs may lead to bias in the simulation results).
- $\Sigma_{EF} \subseteq \Sigma_M$: all the events of interest in the experimentation can be observed during the simulation, but the simulation allows more events to be observed than necessary.

Compatibility of the vocabularies is the first condition necessary for using simulation with respect to an experimentation frame. We will now clarify how to check the applicability of an experimental frame to a model by comparing traces fulfilled by the corresponding TICs.

Definition 3..3 (Restriction of an automaton)**.** *The restriction of an automaton $\langle D, I, O, dom, S, \sigma, \Sigma, \delta, S_0 \rangle$ to a subset $I' \cup O' \cup \Sigma'$ of its vocabulary is an automaton $\langle D, I', O', dom, S, \sigma/I' \cup O', \Sigma', \delta/\Sigma', S_0 \rangle$ in which the configurations and restricted transitions are defined below.*

Definition 3..4 (Restriction of a configuration)**.** *The restriction $\sigma/I' \cup O'$ of a configuration $\sigma : S \to ((I \cup O) \to dom(I \cup O))$ restricted to a vocabulary $I' \cup O'$ such that $I' \cup O' \subset I \cup O$ is the configuration $\sigma'/I' \cup O' : S \to ((I' \cup O') \to dom(I' \cup O'))$ such that $\sigma/I' \cup O'(s)(x) = \sigma(s)(x)$ for all $x \in I' \cup O'$.*

Intuitively, the restriction of a configuration keeps for each state the value of a subset of variables of interest and puts aside the values of all other variables.

Definition 3..5 (Restriction of a transition)**.** *The restriction δ/Σ' of a transition relation $\delta \subset S \times \Sigma \times S$ restricted to a vocabulary $\Sigma' \subset \Sigma$ is the transition relation $\delta/\Sigma' \subset S \times \Sigma' \cup \{\varepsilon\} \times S$ such that :*

- *for all $e \in \Sigma'$, for all $(s, s') \in S \times S$, if $(s, e, s') \in \delta$ then $(s, e, s') \in \delta/\Sigma'$,*
- *for all $e \in \Sigma - \Sigma'$, for all $(s, s') \in S \times S$, if $(s, e, s') \in \delta$ then $(s, \varepsilon, s') \in \delta/\Sigma'$.*

Intuitively, the restrictions of a transition relation keep the structure of a transition relation unchanged, but identify the names of a subset of events of interest and masks the other names with non-observable events ε.

Definition 3..6 (Restriction of a run)**.** *A run restricted to a vocabulary is the initial run in which the initial configuration function and the initial function relation are replaced by their respective restrictions.*

If we have EF and M, two TICs for the respective vocabularies $V_{EF} = I_{EF} \cup O_{EF} \cup \Sigma_{EF}$ and $V_M = I_M \cup O_M \cup \Sigma_M$ such that these vocabularies are compatible. EF is fully applicable to M according to their behaviours if, and only if, all traces of M restricted to the vocabulary of EF are traces of EF: $\|M\|_{V_M/V_{EF}} = \|EF\|_{V_{EF}}$.

The exact matching being not required, intuitively, this means that:

- $\|M\|_{V_M/V_{EF}} \subset \|EF\|_{V_{EF}}$: the behaviours of the model are in the envelope of significant behaviours with respect to the EF.
- $\|EF\|_{V_M} \subset \|M\|_{V_M/V_{EF}}$: all the experimentations planned by the EF can be performed on the model.

When the first condition is false, there are model executions which are not EF executions. We identify two cases:

1. There is a test coverage risk. This may be the case if the model considers input variables with other values than those planned by the EF. The EF does not therefore explore all executions of the simulation. This means that either the unexplored executions are not relevant for the experimentation, or that the EF is not comprehensive enough.
2. There is a bias in the simulation or in the reference definition. This may be the case as the model considers output variables with other values than those expected by the EF. This means either that the simulation results are incorrect or that the EF assumptions are false.

When the second condition is false, there are executions envisaged by the EF which are not model executions. Here again, we identify two cases:

3. Some experiments are outside the usage domain of the simulation model. This may be the case if the EF considers output variables with more values than those accepted by the model. The EF therefore plans to explore simulation executions outside the scope recommended by the model and the simulation results can no longer be guaranteed. The model or EF must be modified.

4. There is a risk concerning the completeness of the model. This may be the case if the EF considers the input variables with more values than those supplied by the simulation results. This means that either there is something to be learnt from the simulation if we reduce the uncertainties of the EF, or that the simulation overlooks the implementation of some cases.

Figure 6 illustrates these four cases.

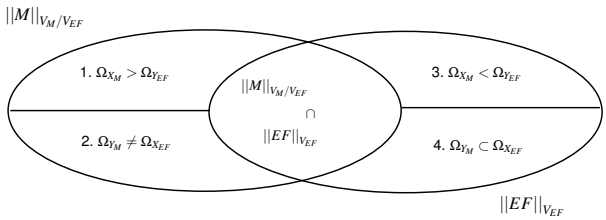

Figure 6. Level of EF/Model behavioural applicability

4. APPLICATION

We illustrate our approach on the design of an embedded system, the control unit of an autonomous, intelligent cruise controller (AICC) [16] [17]. The AICC is more than a regular cruise controller which maintains a speed required by a driver, it adapts the speed vehicle according to the distance and speed of the vehicle ahead. The safe speed is the maximum speed that keeps the car within a safe distance of the vehicle ahead. The wanted speed is the speed the driver requests. The control unit must keep the speed of the vehicle within $\pm 2km.h^{-1}$ of the safe speed or the wanted speed, whichever is lower.

4.1. Modelling

4.1.1. Structural definition

The structural definition of the model of the system of interest is given figure 7 below. It is an important step since it allows defining the boundaries between the model and the experimental frame [18] [19]. In that way, we can clearly distinguish among what drives the model, what is observed as its output and the model itself. It must be avoided to incorporate data-gathering facilities into the model, which would make the model not only more complex but also unsuitable for reuse or for association with different experimental frame. Defining the boundaries between the model and the experimental frame is equivalent to change the scope of the model [20]. So, selecting the scope of the model implicitly includes selecting the exogenous parameters. All other components of the global system (e.g. AICC vehicle) now belong to the experimental frame to represent the environment of the cruise controller. Some of these elements can be abstracted or simplified if their contributions are not relevant to the experimentation.

The system of interest consists of the cruise controller calculation module which computes a throttle setting according to data from the driver and the sensors. Those data include a brake and gas commands, coast and acc controls to respectively decrease and increase the required speed. The safe speed and the speed are retrieved from speed sensors.

In that context, the experimental frame consists on scenarios of brake, gas and required speed and a function to compute the safe speed according to the distance and speed of lead vehicle. This is done by the generator. The transducer observes the throttle setting and update the vehicle speed accordingly. The acceptor continually tests the run-control segments to satisfy a set of constraints, e.g. the speed of the vehicle must remain within $\pm 2km.h^{-1}$ of the safe speed or the wanted speed.

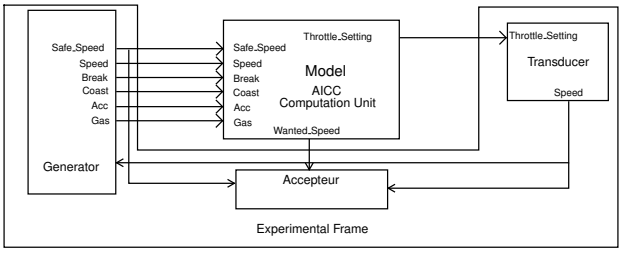

Figure 7. Structural definition of the AICC simulation

4.1.2. Behavioural definition

The behaviour of the AICC calculation module is given by automaton figure 8. When the cruise controller is OFF, and the COAST button is pressed, the cruise controller goes to state ON and the desired speed is maintained. In this state, pressing the COAST button leads to decreasing the driver required speed, while pressing the ACC button leads to increasing the driver required speed. A brake or a gas command turns OFF the cruise controller. The second automaton figure 9 is used to continually check which speed must be used for cruise control, i.e. safe speed or wanted speed.

The behaviour of the experimental frame is given by automata figure 10. The transducer (right side of the figure) updates the vehicle speed according to the throttle setting. Initially, the required speed is greater than the safe speed. The generator (left side of the figure) initializes the speed and

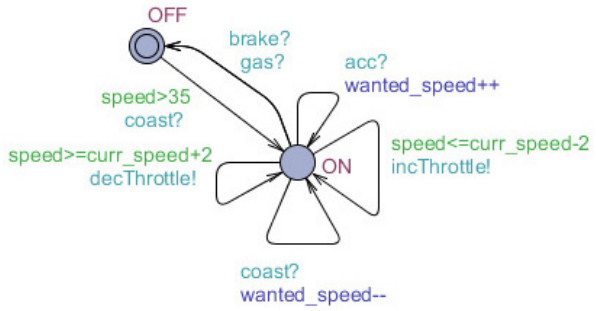

Figure 8. IOA 1 of the model: calculation module

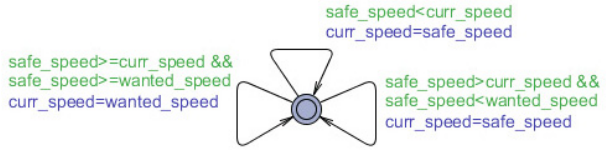

Figure 9. IOA 2 of the model: update speed

the safe speed then it turns on the cruise controller with the COAST button. The cruise controller adapts the speed with the safe speed. Then a break command turns off the cruise controller.

The acceptor is specified with temporal logics. If properties `A◊ speed>=curr_speed-2` and `A◊ speed<=curr_speed+2` are true, this means the speed of the vehicle always remains within $\pm 2 km.h^{-1}$ of the current speed.

4.2. Trace inclusion verification

In fact, the proposed scenario consists in validating the speed selection process: if the safe speed is lower than the required speed, the cruise controller must maintain the speed to the safe speed. Otherwise, it must maintain the speed to the required speed. This requirement can be expressed with the following formula: if the property `E◊ curr_speed==wanted_speed` is true, this means that there is a sequence of alternating delay transitions and action transitions where the current speed was equals to the required

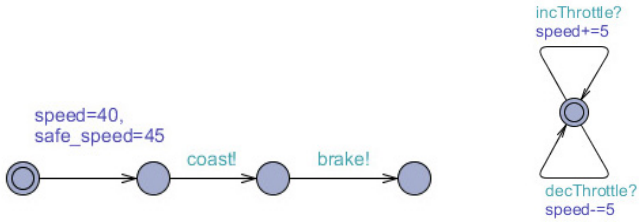

Figure 10. IOA of the experimental frame

speed. With the generator defined above, the property is false. The safe speed is always lower than the required speed, the experimental frame does not therefore explore all executions of the simulation. We are in the first case where the model considers input variables with other values than those planned by the experimental frame.

5. CONCLUSION

This paper introduced an applicability verification approach based on formal matching rules between a model and its intended purpose. We used the concept of experimental frame, proposed in the M&S theory, to address the problem within a well-founded methodological framework. It also allowed us to treat a recurring problem in that the simulation intended purpose is often not well-defined. We have shown that the model is not the only cause of any bias to a simulation but that the experimentation performed with this model may also have been poorly defined.

The issue of matching is founded on the principle that an intended purpose and a model are two components, in the formal meaning of the term, i.e. interacting through their interfaces and only through their interfaces. We turned to component-based engineering techniques to iteratively enhance the concept of an experimental frame of ”symbolic concepts” and determining the system behaviours that must be included in the model.

The generator defines the logical order of the stimuli injected in the simulation. The transducer defines the logical order of the observations. We specified these two components using automata. The acceptor, specified using temporal logic, consists in verifying whether a requirement has been verified based on stimulus/observation pairs. As for the properties related to the verification of applicability between the experimental frame and the model, we also defined temporal logic properties for EF states to determine which case of applicability we found ourselves in. While this approach is consistent, an effort remains to be made to find more or less systematic applicability properties.

With such an approach, the applicability verification can be performed either on the development ab initio of a simulation necessary and sufficient to satisfy an intended purpose, or on the reuse of an already existing simulation model to satisfy an intended purpose. Furthermore, developers are free to associate a model with different experimental frames, each corresponding to a particular simulation objective of use. If we consider that morphisms between models of a same system are well documented, juggling between abstractions and previous simulation results allows verifying the applicability of a ”new” simulation without executing the simulation. In the same way, one can put another model of the system of interest in the experimental frame and verify by matching if this model can accommodate the experimental frame.

REFERENCES

[1] F. K. Frantz, A taxonomy of model abstraction techniques, in: WSC '95: Proceedings of the 27th conference on Winter simulation, IEEE Computer Society, Washington, DC, USA, 1995, pp. 1413–1420.

[2] M. K. Traoré, A. Muzy, Capturing the dual relationship between simulation models and their context., Simulation Modelling Practice and Theory 14 (2) (2006) 126–142.

[3] O. Balci, W. F. Ormsby, J. T. Carr, III, S. D. Saadi, Planning for verification, validation, and accreditation of modeling and simulation applications, in: WSC '00: Proceedings of the 32nd conference on Winter simulation, Society for Computer Simulation International, San Diego, CA, USA, 2000, pp. 829–839.

[4] R. G. Sargent, Verification and validation of simulation models, in: WSC '05: Proceedings of the 37th conference on Winter simulation, Winter Simulation Conference, 2005, pp. 130–143.

[5] D. Brade, Vv&a ii: enhancing modeling and simulation accreditation by structuring verification and validation results, in: WSC '00: Proceedings of the 32nd conference on Winter simulation, Society for Computer Simulation International, San Diego, CA, USA, 2000, pp. 840–848.

[6] W. L. Oberkampf, T. G. Trucano, C. Hirsch, Verification, validation, and predictive capability in computational engineering and physics, Appl. Mech. Rev. 57 (5) (2004) 345–385.

[7] M. K. Traoré, Analyzing static and temporal properties of simulation models, in: WSC '06: Proceedings of the 38th conference on Winter simulation, Winter Simulation Conference, 2006, pp. 897–904.

[8] C. J. Jacques, G. A. Wainer, Using the cd++ devs tookit to develop petri nets, in: 2002 Summer Computer Simulation Conference, 2002.

[9] B. P. Zeigler, H. Praehofer, T. G. Kim, Theory of Modelling and Simulation, Academic Press, San Diego, California, USA, 2000.

[10] C. Carrez, A. Fantechi, E. Najm, Behavioural contracts for a sound assembly of components, in: In FORTE, Springer, 2003, pp. 36–39.

[11] F. Arbab, Abstract behavior types: a foundation model for components and their composition, in: Sci. Comput. Program., Vol. 55, Elsevier North-Holland, Inc., Amsterdam, The Netherlands, 2005, pp. 3–52.

[12] E. Lee, Y. Xiong, Behavioural types for component-based design, Technical Memorandum UCB/ERL M02/29.

[13] B. P. Zeigler, Theory of Modelling and Simulation, Krieger Publishing Co., Inc., Melbourne, FL, USA, 1984.

[14] C. Szyperski, Component Software: Beyond Object-Oriented Programming, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[15] L. Alfaro, T. A. Henzinger, Interface automata, in: Proceedings of the Ninth Annual Symposium on Foundations of Software Engineering (FSE), ACM, Press, 2001, pp. 109–120.

[16] U. Palmquist, Intelligent cruise control and roadside information, IEEE Micro 13 (1) (1993) 20–28.

[17] S. Schulz, J. W. Rozenblit, K. Buchenrieder, Towards an application of model-based codesign: An autonomous, intelligent cruise controller, in: Proceedings of the 1997 IEEE Conference and Workshop on Engineering of Computer Based Systems, 1997, pp. 73–80.

[18] V. Albert, A. Nketsa, Signature matching applied to simulation/frame duality, The Fourth International Conference on Systems (ICONS 2009) (2009) 190–196.

[19] V. Albert, A. Nketsa, M. Paludetto, C. Seguin, R. Jacquart, J. Casteres, Simulation validity assessment: simulation objectives of use description guidelines, Tech. rep., LAAS-CNRS and ONERA-CERT (2009).

[20] D. S. Weld, Reasoning about model accuracy, Artif. Intell. 56 (2-3) (1992) 255–300.