# Accuracy of Simulation

Rainer Gerlich
BSSE System and Software Engineering

Auf dem Ruhbuehl 181
D-88090 Immenstaad, Germany

Phone +49/7545/91.12.58 Mobile: +49/171/80.20.659 Fax +49/7545/91.12.40
e-mail: gerlich@t-online.de www: http://home.t-online.de/home/gerlich/

**Keywords**: SDL, validity of simulation results, performance impacts, EaSySim II

## 1. INTRODUCTION

This paper discusses the correctness of simulation results in view of system validation, addressing functional, behavioural and performance aspects. It raises the question "how much can we rely on the results of simulation?" and tries to give an answer on "what we need to do to get representative results by simulation".

During the ESA/ESTEC study HRDMS (Highly Reliable Data Management System and Simulation) [1] which concentrated on performance validation several faults were identified which were directly related to timing conditions, i.e. under different timing conditions the faults would not have been occured. These results doubted that purely functional or behavioural validation of a system without consideration of performance impacts will allow to conclude on a system's correctness.

Errors which are related to performance impacts may force a re-design of a system and may be very expensive if the errors are detected too late. Usually performance aspects are considered rather late in the lifecycle, so that concentration on functional or behavioural aspects may lead to serious problems during and especially at the end of system development. However, most of the known methods and tools ignore performance aspects. This makes it difficult to identify risks early enough.

Based on this experience the question was raised if and how we can detect such errors with current methods and tools. One case out of the set of faults identified during HRDMS was selected (as described in [2]) and investigated in more detail with SDL extended by performance capabilities in the EaSySim II environment [3,4]. The general problem raised by the identified fault is that the sequence of state transitions usually looks different if delays caused by the physical environment are considered.

During the exercises executed for above example [2] it was recognised that the risk to miss a fault is the higher the more the system is optimised for performance. However, without consideration of performance impacts such an optimisation is not possible. Hence, currently - without having capabilities for performance simulation - a system is validated for functions and behaviour, but later modified for performance reasons: so the proof of validation is lost, the system may behave erroneous. In worst case an error occurs rarely only in some situations which may not happen during the tests.

The weak points of purely functional and behavioural simulation and validation are discussed now for SDL and it is shown why and how performance extensions will help to succeed.

## 2. ACCURACY IMPACTS

The exercises which were executed in the EaSySim II environment showed that

1. competition for a shared resource such as a CPU or a bus impacts significantly the validation result,
2. introduction of resource consumption helps to reduce the number of system states, and to avoid state explosion.

According to (1) nearly every (wrong) result can be obtained when the simulation model and environment are not sufficiently representative (as described in [2]), which is especially true if resource consumption is not considered.

On the other side state explosion may prevent that we will get any valid, representative result on a system's correctness at all. By the exercises with peformance extensions it was recognised that the number of system states is reduced when performance impacts are considered: a physical system behaves much simpler than an ideal mathematical system. As SDL executes a model more mathematically-oriented, performance extensions will help to get (valid) results where otherwise no representative result is obtained due to state explosion.

### 2.1 The Impact by Resource Consumption

A resource is usually a hardware component of a system such as a processor or a bus. When a process is consuming such a resource it is usually delayed: because either the use of the resource takes some time or a process has to wait because the resource is consumed by another process which shares the same resource. This will impact the sequence of state transitions, of course.

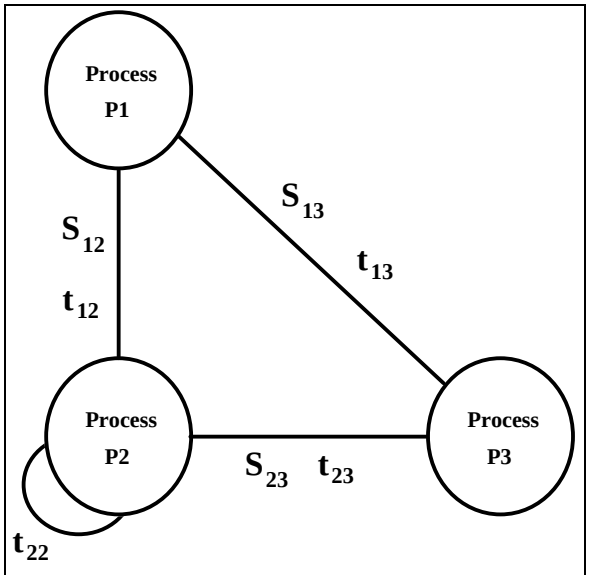


*Fig. 1: Transitions vs. Propagation Time*

The question was raised whether the set of state transitions as obtained during exhaustive simulation includes all possible combinations, i.e. also the sequences occuring when (shared) resources are consumed. However, this is not true in general as is explained by the example given by Fig. 1.

Here we have three processes P1, P2 and P3 which communicate via signals, e.g. $S_{12}$, $S_{13}$ and $S_{23}$ where $S_{ij}$ indicates the flow of data from process i to process j. We further assume that $S_{23}$ is not sent before P2 has executed an internal state transition. The transmission times of the signals are $t_{12}$, $t_{13}$, $t_{23}$ and the execution time of the state transition at P2 is $t_{22}$.

We now consider following sequence without applying resource (time) consumption: $S_{12}$ and $S_{13}$ are issued at the same time by P1. On reception of $S_{12}$ P2 executes a state transition during which $S_{23}$ is sent to P3. This causes that $S_{13}$ arrives earlier at P3 than $S_{23}$.

By a second iteration we introduce time and assume that $t_{12}+t_{22}+t_{23} < t_{13}$. Hence, under this condition $S_{23}$ arrives before $S_{13}$ at P3 and this sequence will never occur in the set of

sequences for $t_{12}=t_{22}=t_{23}=t_{13}=0$, because $S_{12}$ and $S_{13}$ are sent during the same state transition.

Different conclusions are now possible:

1. no time consumption:
   a. $S_{13}$ shall really arrive before $S_{23}$, then the simulation gives a correct result; however, its correctness still depends on performance properties of the real system which may invalidate this result, so this result is worthless.
   b. $S_{13}$ shall arrive after $S_{23}$, then P1 needs to be changed, $S_{13}$ needs to be issued by a later state transition, possibly triggered by a timer; however, if the channel between P1 and P3 would later cause the needed delay, then this change would not be needed and leads to erroneous or less performant system behaviour later on. In any case, the result is useless because the real properties are not considered.
2. with time consumption
   a. $S_{13}$ shall really arrive before $S_{23}$, then the simulation indicates a problem, if the delay of $S_{13}$ by channel P1 - P3 is too high: the system architecture needs to be changed; if delay $t_{13}<t_{12}+t_{22}+t_{23}$ then simulation gives a correct and representative result.
   b. $S_{23}$ shall arrive after $S_{23}$, then in case $t_{13}>t_{12}+t_{22}+t_{23}$ the simulation confirms correctness of the system, otherwise a problem is identified.

In case 2 always representative results are obtained and the right conclusions are derived, while in case 1 the simulation result "ok" or "not ok" still may be changed when the performance properties of the system are introduced.

This example demonstrates the impact of performance on a simulation result. Time consumption in a part of a model is independent of the logic of a state transition in another part, and vice versa.

In the example given by [2] (which is a little bit more complicated) such missing performance impact caused that the final checks which are sent at the end of a protocol to arrive before the last data.

In consequence, without consideration of performance a representative validation result may never be achieved by simulation. Vice versa, the performance properties of a physical system will help us to succeed with validation if we condition the validation environment in the right manner. The results of such exercises were considered when EaSySim II was built.

## 2.2 Verification with SDL

SDL allows to model a real physical system as an abstract mathematical system only: it assumes zero time consumption for a state transition and zero propagation time of signals. This may lead to different and wrong results compared with the behaviour of the physical system. As discussed above, if time is consumed during a state transition the sequence of issued (SDL) signals may look quite different compared with the case when no (processor and bus) time is consumed. Of course, this impacts the overall behaviour of the system because the queue inputs will also change over time. Consequently, a MSC generated for the physical system will look quite different.

The need for modelling of performance aspects becomes more urgent for a distributed system for which propagation of information through a network is significant. When time is not considered a processing sequence may occur for the SDL model which will never be observed in the real environment or vice versa, a sequence of the real world will never be generated by the SDL model.

E.g. information may propagate through a system with different speed. This will cause that the sequence of data may be changed and that the data (or the results of data procvessing) will arrive in another order at a certain destination. Hence, illegal racing conditions may never be observed, if time is not considered during simulation, and they will cause erroneous, non-expected system behaviour later on.

Also, the processing of the queue inputs as it is usually performed in SDL differs from what happens really in the physical system. SDL considers all possible combinations of queue inputs which causes an explosion of system states already when a rather small number of elements is stored in a queue.

For a physical system usually a certain strategy is applied for processing of a number of inputs, which is done e.g. based on priority-driven pre-emptive scheduling. This causes that only one of the large number of combinations will be executed in practice: for n elements of a queue only n possible queue states occur compared to n! queue states which are currently considered during exhaustive simulation.

Moreover, having passed a resource, signals will arrive at a process queue one after the other. Consequently, in most cases only one element will be in the queue, apart from such cases were the processor is not fast enough to finish its work before the next signal arrives. So we have the following two principal cases:

1. a queue has one element only (due to "dispersion" by resources), or
2. if more members are in the queue we apply a strategy which allows us to select uniquely a certain queue element for processing (which is not necessarily the first one in the queue).

So a combinatorial explosion does not happen. The full (discrete and finite, but huge) spectrum of combinations is significantly reduced when taking into account properties of the physical system.

Now, it could be argued that the combinatorial treatment of the queue contents reflects the potential delays of signals by the resources (channels): due to different transmission or processing time signals will be mixed such that they arrive randomly at a process queue. Hence, the random selection of queue elements just simulates such delays and random arrival times.

However, as we have seen by the example of section 2.1 the intersection between the spectrum of system states for a physical and a mathematical model may be empty: the combinatorial processing of queues may not be representative for the real world. With other words: if we try to model performance impacts by combinatorial execution we may not use the real set of event sequences. We can only be sure to meet the real behaviour if we consider all properties of a system, especially performance properties, where they impact system behaviour. We surely will fail, when we try to model such impacts at parts of the systems where the performance constraints are not occuring in practice.

Consequently, a performance extension of SDL should allow for strategy-driven modelling and simulation, not only to support representative modeling, but also to reduce the number of system states to the number which is related to the physical system.

### 2.3 A Formal Notation on Performance is still Needed

As was discussed by [2] and in section 2.1 nearly arbitrary results can be achieved by simulation which may be either correct or wrong: the simulation is saying "ok", but it is not, or, it is saying "not ok", but the system is "ok". However, in any case when considerung performance impacts such representaive results are still obtained by execution of the model, not by formal analysis like for the Finite State Machines. This is dissatisfying. The reason is that still a formal notation is missing which allows to express performance impacts in a representative manner and which is useful in practice.

The question is now can we add to SDL a formal notation concerning consumption of and competition for resources such that we can identify in a formal manner that there is conflict or not. For the time being this question cannot be answered, but we can think about a solution, at least.

A major advantage of exhaustive simulation is that it provides results and identifies bugs without being asked specifically for the violated property or the property in question. This makes it superior to other approaches which will give correct answers, but need to be asked for it. And here we have again a problem: a human being is not capable to take care of all properties and potential bugs in a system. If we wouuld be aware of them, we already now would be able to build perfect systems. Hence, the best approach is useless if it needs a perfect operator. So the question remains which notation can help us to improve the situation.

Also, there is another problem which is related to interference of signal sequences: when only one sequence of signals, e.g. of a protocol, is considered, everything may seem to be ok, although it is not under real conditions later on when several sequences may compete for the same resource.

In case of the example which was shown in [2] a signal had to be delayed by several bus cycles in order to prevent that it arrives to early at the destination. Now, for performance optimisation another signal could try to use the empty slots. Then it may happen that a signal with the same destination as the delayed signal (issued by another process) will take such slots: hence, data might arrive at the destination which will disturb the still on-going protocol and cause an error.

Unfortunately, such an erroneous situation does not occur rather often, it is more likely to occur very rarely. Missing such events during simulation may also lead to wrong conclusions about a system's correctness. Again, currently such errors related to performance matters can only be identified by test. It would be extremely helpful if they could be detected by analysis.

## 3. CONCLUSIONS

Although SDL provides good support for system analysis and simulation, it still is not perfect: it lacks (at least the language) of support for performance modelling, and it makes things more complex as they really are. First steps towards extension of SDL have been done, but more steps are still needed to allow for representative modelling, and to get results which can be trusted.

What is needed is

1. an extension which allows
   a. to specify consumption of resources,
   b. to define a strategy how competing processes can consume a shared resource
2. a formal notation by which the temporal propagation of information can be specified.

To add such features to SDL would mean a big step forward to allow for representative modelling and to get accurate results by simulation.

## 4. REFERENCES


[1a] HRDMS (Highly Reliable DMS and Simulation), ESTEC contract no. 9882/92/NL/JG(SC), Final Report, 1994, Noordwijk, The Netherlands

[1b] R.Gerlich, N.Schäfer, A.Schäferhoff: Early Validation of DMS Design by a Reusable Environment, EUROSPACE On-Board Data Management Symposium on "Technology and Applications for Space Data Management Systems", January 25-27, 1994, Rome, Italy

[2] R.Gerlich: Tuning Development of Distributed Real-Time Systems with SDL and MSC:Current Experience and Future Issues", SDL'97 Forum, September 22-26, 1997, Evry, France

[3] EaSySim II: The Enhanced Environment for System Validation, R. Gerlich BSSE, Auf dem Ruhbuehl 181, D-88090 Immenstaad, Germany

[4] EaSySim II User's Manual, R. Gerlich BSSE, Auf dem Ruhbuehl 181, D-88090 Immenstaad, Germany