INTERACTION CONTROL IN A COMBINED LOGISTICS AND CHEMICAL PROCESS SIMULATION

Alexander Lavrov, Dietmar Hietel, Stefan Nickel

Fraunhofer-Institute for Industrial Mathematics
D-67663 Kaiserslautern, Germany
E-mail: lavrov@itwm.fraunhofer.de

KEYWORDS

Parallel & distributed simulation, hybrid systems, logistics simulation, chemical process simulation

ABSTRACT

This paper describes a synchronisation component of a framework intended at a rapid and transparent integration of (sub)models of logistics components (usually discrete event based) and of chemical processes (continuous) into an overall simulation model of an industrial system. Such kind of model enables an analysis of those behavioral features of the complete system which result from the interplay of its components and are not directly observable under their separate investigation. Domain-specific properties of subsystem interaction allow to implement a flexible, powerful yet simple synchronisation scheme. It operates with standard communication interfaces and does not rely upon arranging a complex specialised platform for distributed simulation. A detailed description of the synchronisation algorithm is given, and some causality related issues are discussed. The prototype framework works with models implemented in eM-Plant (for logistics) and WinZPR (for chemical processes).

INTRODUCTION

Simulation is a standard technique in the design and analysis of complex industrial systems, in particular those involving interacting discrete and continuous components, e.g. chemical enterprises. Simulation of hybrid systems (Barton 2002) has for a long time attracted attention of researchers and practitioners. Usual practice however still demonstrates a strong separation between the process simulation (primarily continuous) (Turton 2002), and the logistics simulation (mostly discrete) (Banks 2001).

In the models used at one of these two sides, the influence of the other side is usually represented by the imitation of the latter via input/output flows. For the common purposes such a separated scheme usually does satisfy its needs. In addition, separate consideration is often justified by the imparity between both sides: The process side plays the superior one, and the aim of its simulation is to find the optimal operation modes of the equipment, ensuring high quality of the end product, safety and reliability. The logistics side, on the contrary, is required to fulfil the prerequisites imposed by the process side (e.g. transport service, personnel, etc.), and its simulation serves for determining the necessary capacities and developing corresponding operation modes.

Yet there are applications (e.g. control system testing) where an integrated view of the system, with an explicit interaction between its heterogeneous parts, is especially important. For example, complex causal chains may occur in the operation of the whole system, which can lead to blocking, failures, etc. Such chains often cannot be detected via analysing separate subsystems under restricting assumptions about their interaction.

The main barrier to use industry relevant hybrid simulation is the fact that each of the parties uses completely different simulation tools that are best appropriate for its needs. The most widespread tools are either of discrete (e.g. AutoMod, eM-Plant) or of continuous (e.g. ASPEN, ChemCAD) nature, with sometimes available restricted hybrid features. At the same time, existing hybrid simulation languages and tools do not provide enough application-oriented functionality and do not enjoy a wide popularity in industry. Possibilities to combine specialised simulation models in a distributed framework are rather restricted, since the corresponding tools rarely possess interfaces to special tools and environments of parallel and/or distributed simulation based, e.g. based on HLA (DMSO 2003). In addition, the use of the latter requires special knowledge, experience, essential amount of additional programming including, possibly, an intervention into the source code.

The goal of the work presented in this paper was to develop an easy-to-use open framework with the following properties. Firstly, it should be based on special domain features, allow a fast and transpar-

Proceedings 15th European Simulation Symposium
Alexander Verbraeck, Vlatka Hlupic (Eds.)

ISBN 3-936150-28-1 (book) / 3-936150-29-X (CD)

ent integration of different discrete event and continuous submodels into an overall combined (hybrid) simulation model, avoiding the time- and specific knowledge-consuming application of specialised frameworks. Secondly, it should support the model development process and the usage of optimisation components both for guiding the experiments and for representing the decision-making activities inside a model. The primary reason was to develop simple-to-use facilities for integrating available models, a higher parallelisation etc. being a secondary aspect.

This paper is focused on the development of a synchronisation unit which would, on one hand, follow the main causality-related principles of distributed simulation, and, on the other hand, avoid the technicalities and programming complexity associated with an implementation or usage of a universal distributed simulation environment.

First we give a general characteristic of the components. The main part describes the implemented synchronisation algorithm. After that, a weak conservative time advance scheme, used in the case of multiple models without rollback, is described. Finally, some implemenation issues related to causality and communication are discussed.

INTENDED FRAMEWORK AND CHARACTERISTICS OF SIMULATION COMPONENTS

The synchronisation scheme presented in this paper, is used in the control unit (CU) of a simulation-based decision-support framework (Fraunhofer 2003) consisting of discrete event simulation models, continuous models, an optimisation component, and a groupware component with an associated database (fig. 1). We focus on the part of the framework highlighted in fig. 1 by a dotted line.

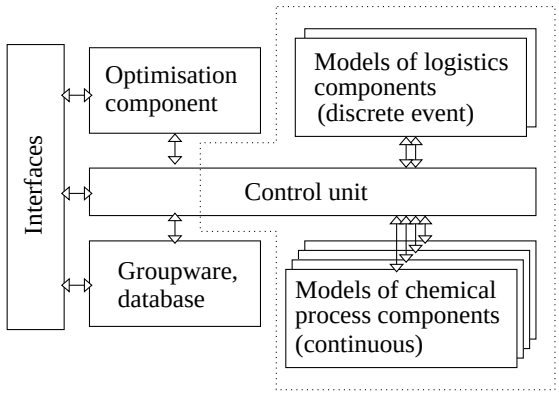

Figure 1: General structure of the combined simulation-based decision-support framework

The modeled system (chemical enterprise) has special properties which can be used while developing a framework for combined simulation:

P1 Interactions between the logistics components and the process components typically relate to transport operations (arrivals, departures, loading/unloading, pumping, etc.).

P2 The tolerance of time representation of such operations is usually higher than that of the duration values inside a submodel (especially chemical).

P3 Interactions take place rarely compared to the internal events of individual models.

From the view of distributed simulation, the to-be-combined models (tools) must be examined concerning the availability/implementability of the following features: (1) rollback, (2) lookahead, (3) monitoring and detection of certain (types of) events and conditions, (4) interruption/resumption of simulation in predetermined points of model (virtual) time, (5) communication and control interfaces, (6) event list observation and external control. Table 1 contains corresponding characteristics of logistics- and chemical process-oriented simulation tools.

Table 1: Characteristics of logistics- and chemical process-oriented simulation tools

	Chemical process simulation tools	Logistics simulation tools
(1)	Operation consists in solving differential equations and representing the results as system's evolution over time. Technically, the main functionality of the rollback can thus be implemented via recalculation (**rerun**) from a given point of time.	Operation consists in processing of event lists. Rollback features and the possibility to rerun the simulation from an intermediate point are usually not available.
(2)	For some modeled reactions the shortest duration may be available, thus giving a basis for a lookahead. Embedded programming features are usually enough for its implementation.	Some operations allow forecasting (e.g. the shortest transportation time) and hence a certain lookahead is possible. It is easily implementable via internal programming features (e.g. SimTalk language in eM-Plant).
(3)	Availability of continuous monitoring and detection of special events and/or conditions varies essentially depending on tools.	Continuous monitoring of conditions and events is implementable via **waituntil**-like features, while the scope of the functionality is determined by the restrictions on the triggering condition.
(4)	The possibility of recalculation (see feature (1) above) always allows to implement detection via a combination of "model run with trace logging + retrospective detection and evaluation + rerun up to the desired point".	Interruption at a given time point is usually realisable, however for implementation of possible additional requirements, such as "after processing all the events with the given timestamp", additional programming efforts are needed.

Table 1 (continued)

(5)	Communication capabilities of most of the available tools are sufficient for general purpose exchange (at least: file read/write features, data base interfaces, etc.).	Sufficient general-purpose communication features are usually available.
(6)	Event list is not available. Indirect observability is essentially associated with the feature (3) above.	The event list is often open for complete or partial observation. The possibilities to directly influence it are either not available or very restricted.

Analysis of features P1-P3 of the domain and features (1)-(6) (table 1) of simulation tools allows to determine the following main principles of the intended integration framework design:
– simulation proceeds via subsequent advance steps in the two groups of models: those with and without rollback,
– a limited set of interaction events ("I-events") is completely specified,
– simulation steps inside a group are synchronised using a simple barrier-based technique, the barriers being associated with the I-events,
– parallel model runs are possible under availability of the lookahead,
– two strictly distinct message types are used: control and domain-related,
– a wide spectrum of communication options is available: via file read/write, database, sockets, etc.

These principles are implemented in the algorithms and approaches presented in the following sections.

MAIN SYNCHRONISATION SCHEME

Information on individual models, or logical processes (LPs), and corresponding communication types is available to the CU as specified in table 2:

Table 2: General features of the participating submodels as represented in the CU

path	log. name	var. type	synchr. type	comm. medium
D:\trsp\t.exe	transport	discr.event	cons.	socket
C:\ref\ref.exe	refinery	contin.	optim.	file
...	...	...	...	...

For simplicity, we will call "optimistic" those LPs that are able to perform rollback, and "conservative" the other LPs. The sets of conservative and optimistic LPs will be denoted $\mathcal{CLP}$ and $\mathcal{OLP}$, respectively.

The implemented synchronisation scheme represents a combination of standard techniques of conservative and optimistic synchronisation, involving, in one or another form, the functionality of barrier algorithms, null messages, rollback (fig. 2).

Control unit actions

Set starting time t_0; start all LPs
while not termination-condition **do**
  determine minimal guarantees:
    $t^o_{guar} = \min\{t_{guar_i} \mid LP_i \in \mathcal{CLP}\}$;
    $t^c_{guar} = \min\{t_{guar_i} \mid LP_i \in \mathcal{OLP}\}$;
  if $t^o_{guar} > 0$ **and** $t^c_{guar} > 0$ **then**
    module **A**, fig. 3
  elseif $t^o_{guar} = 0$ **and** $t^c_{guar} = 0$ **then**
    module **B**, fig. 4
  elseif $t^o_{guar} = 0$ **and** $t^c_{guar} > 0$ **then**
    analogous to module **B** (fig. 4), with value t^c_{guar} instead of t^o_{lim}
  else /* i.e. if $t^o_{guar} > 0$ **and** $t^c_{guar} = 0$ */
    module **C-1**, fig. 6, or module **C-2**, fig. 7
  endif
  read the domain-related messages from LPs;
  run the supervisory decision component;
    (LP of the central control system);
  forward messages and commands;
  update $t_0 = t^*$ (see below)
endwhile

Figure 2: CU actions (synchronisation algorithm)

The algorithm consists of cyclic repetition of a model time advance step, starting in the current point t_0 (common for all submodels). Such a step consists of a sequence of actions (denoted as "module" in fig. 2) resulting in one of four possible cases depending on the relation between the lookahead values (guarantees) obtained from the different groups (optimistic and conservative) of LPs. The termination condition corresponds to reaching a given upper bound on the simulation time or completing the to-be-simulated production period.

The case with nonzero guarantees from both, conservative and optimistic, LP groups (fig. 3) is the most efficient: it allows parallel runs of all the models over a common predetermined model time interval and does not require a rollback.

Module **A**

Run all models (optimistic and conservative) from t_0 to $t^* = \min\{t^o_{guar}, t^c_{guar}\}$.

Figure 3: Module A (both guarantees are available)

In the case when both minimal guarantees are zero, parallel runs are only possible inside a group of LPs (conservative or optimistic), whereas between the

two groups a purely sequential form is used (fig. 4): first, the optimistic LPs are ordered to proceed to the given point t^o_{lim}.

Module **B***; actions illustrated in fig.5(a)*

run each $LP_i \in \mathcal{OLP}$ from t_0 to its first
  I-event at t^e_i, but not further than t^o_{lim};
let t^*_i be the stop time of LP_ith run;
set $t^o_{min} := \min\{t^*_i \mid LP_i \in \mathcal{OLP}\}$;
run "weak conservatively" all $LP_i \in \mathcal{CLP}$
  from t_0 to the first I-event at t^*,
  but not further than t^o_{min};
rerun each model $LP_i \in \mathcal{OLP}$
  such that $t^*_i > t^*$ from t_0 to t^*.

Figure 4: Module B (no lookahead guarantees available)

If an I-event e_i occurs in an LP, this LP stops just after processing all internal events with the timestamp equal to that of e_i (step 1 in fig. 5 (a), see next page).

The minimum stopping time t^o_{min} of all optimistic LPs yields the upper bound for the next advance of the conservative LPs (step 2 in fig. 5 (a)). If there are more than one conservative LPs, then a conservative synchronisation must be applied inside group $\mathcal{CLP}$ (symbolically represented in fig. 5 by small steps on the "conservative" side).

Taking into account property P2 of the domain, a special "weak conservative" approach is used here which is implemented using three elementary operations on event list (*check the next event time*, *scheduled method call*, and *cancel a scheduled method call*) and thus does not require complicated programming intervention. This approach is explained in a separate section below.

The conservative LPs (as a weak conservatively synchronised group) can either reach the point t^o_{min} (diagram ⓐ in step 2 in fig. 5 (a)) or stop earlier at t^c_{min} if an internal interaction event has occurred (diagram ⓑ in step 2 in fig. 5 (a)). Let $t^* = \{t^o_{min}, t^c_{min}\}$. The third, optional step, performs rollbacks (via rerun) of those optimistic LPs whose end time point at step 1 was greater than t^*: these are either all optimistic LPs (diagram ⓒ in step 3 in fig. 5 (a)) or all optimistic LPs except the one(s) stopped at t^o_{min} (diagram ⓓ in the same step).

When a nonzero minimal guarantee t^c_{guar} is available only from the group of conservative LPs, the action sequence is the same as in module B, with the only difference that t^c_{guar} is used as the upper bound for the optimistic runs at step 1.

If, on the contrary, only the optimistic LPs delivered a nonzero minimal guarantee, then possible actions can take on two main forms: sequential (module C-1 in fig. 6) or parallel (module C-2 in fig. 7).

Module **C-1***; actions illustrated in fig.5(b)*

run "weak conservatively" all $LP_i \in \mathcal{CLP}$
  from t_0 to the first I-event at t^e,
  but not further than t^o_{guar};
let t^* be the stop time of the conservative LPs;
rerun the optimistic models to t^*.

Figure 6: Module C-1 (optimistic guarantees available; alternative 1: sequential)

Module C-1 is rollback free, though this is reached via purely sequential runs of the two groups of LPs and thus may be time consuming. Module C-2 allows a partial parallelism and is appropriate when the probability of earlier interaction events during the next runs of the conservative LPs is low.

Module **C-2***; actions illustrated in fig. 5(c)*

run optimistic models from t_0 to t^o_{guar} and
  simultaneously run "weak conservatively"
  all $LP_i \in \mathcal{CLP}$ to the first I-event
  at t^e, but not further than t^o_{guar};
if stopping time of the conserv. group $t^* < t^o_{guar}$
then rerun the optimistic models from t_0 to t^*
endif

Figure 7: Module C-2 (optimistic guarantees available; alternative 2: parallel)

WEAK CONSERVATIVE ADVANCE

Weak conservative advance (fig. 8) consists of repetitive small parallel steps (equal to the tolerance interval) of all conservative LPs; the advance stops as soon as an interaction related event has been detected in any participating LP during the last step, or if the upper bound of the interval has been reached (fig. 9).

Weak conservative advance; actions illustrated in fig.9

$t_1 = t_0$; $t_2 = t_0$; *int.event* = FALSE;
while $t_2 < \hat{t}$ **and not** *int.event* **do**
  $t_2 = t_1 + \min\{\Delta t, t_{max} - t\}$;
  run conservative LPs over $]t_1, t_2]$;
  if interaction event(s) registered in some LPs
    then *int.event* = TRUE
  endif
  $t_1 = t_2$;
endwhile

Figure 8: Weak conservative advance

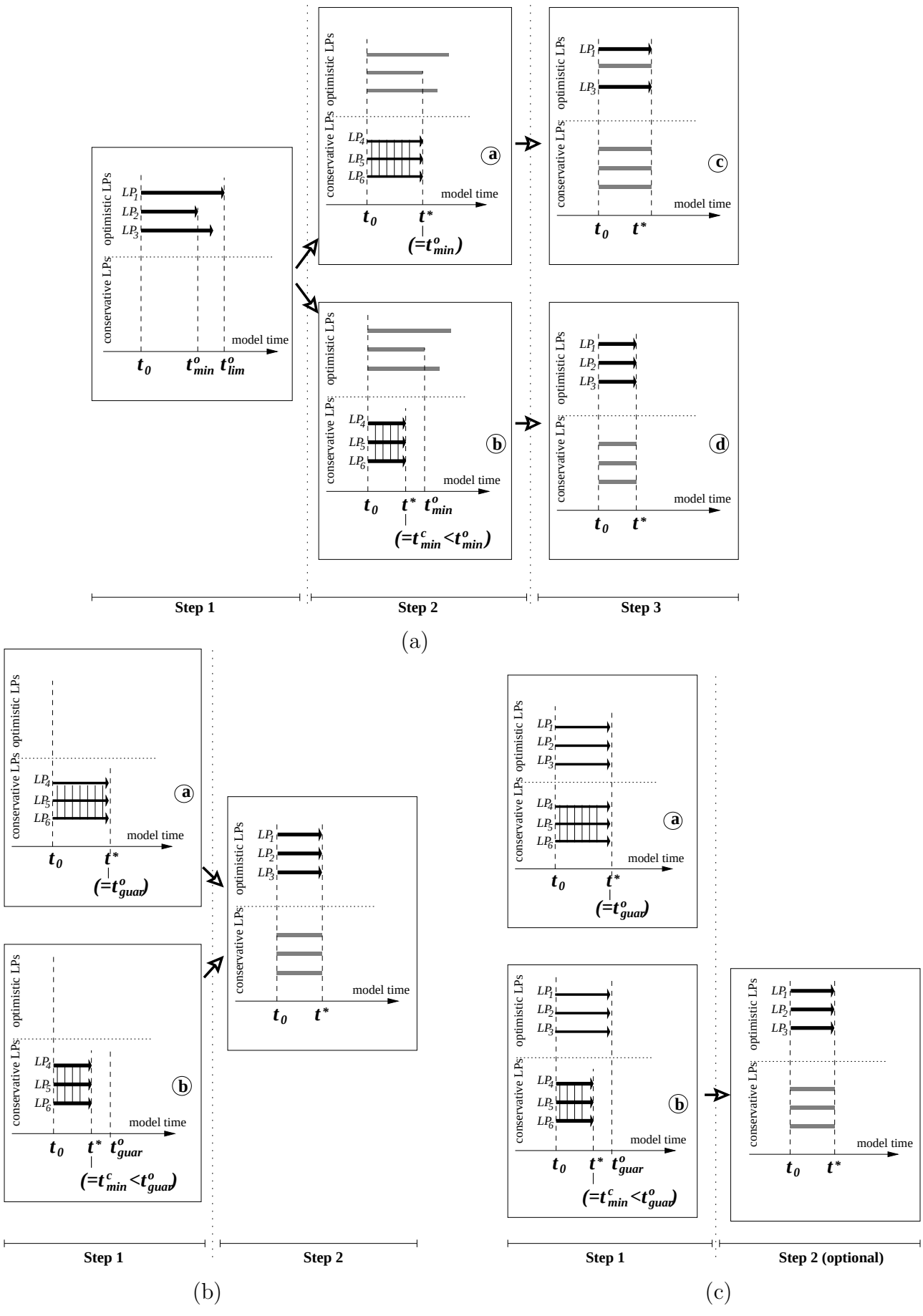

Figure 5: Illustrating module actions: (a) action sequence of module B from fig. 4, (b) action sequence of module C-1 from fig. 6, (c) action sequence of module C-2 from fig. 7

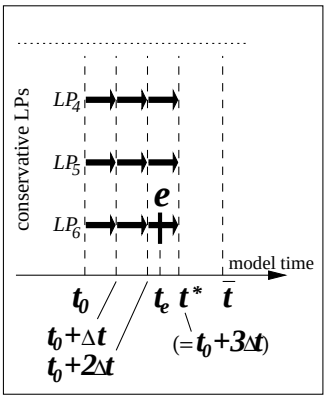

Figure 9: Weak conservative advance

The underlying idea is based on property P2 of the domain and on an assumption about the existence of some time tolerance interval which allows to extend certain duration values (and consequently the timestamps of corresponding events events). The I-events which occurred in an LP inside this interval will be considered by the other (recipient) LPs as having the timestamp of the end of the interval. This is a plausible assumption since the tolerance (precision) of several seconds (sometimes even dozens of second) in the logistics processes (transport arrivals and departures) is usually acceptable. On the other hand, this allows to avoid synchronisation on every time increment related only to internal events in every LP.

CAUSALITY AND COMMUNICATION: IMPLEMENTATION ISSUES

The CU communicates individually with each LP. Two types of messages are used: control messages (domain independent), and domain-related messages, each type having its own fixed format. LPs exchange control messages with the CU at the interaction (barrier) points. Part of the synchronisation-relevant information maintained by the control unit is shown in table 3.

A control message, issued by the CU, contains, among other items, its number (each pair "CU - LP" has an independent counter), timestamp (issuing time), command ("forward", "rerun" or "stop"), the start point and the end point of the time interval for the corresponding model run. For the replies, issued by an LP, additional positions are reserved for the real end point of the run and for the indicator of the interaction events occurred during the last run.

A domain-related message contains information on the event type (arrival, departure, etc.), order number, substance, aggregate state, volume, components and their mol percentage. Domain-related messages from other LPs are processed by an LP only at the beginning of the "forward" run. Domain-related message exchange can only be initiated by the CU, which issues a special "read impulse" in the control message.

A strict sequence of internal and external information processing is used (fig. 10): first, own events up to the barrier point are completely processed. Afterwards, the information on the (simultaneous) events in other LPs becomes available. Thus, it can impact only the future evolution of the process. This ensures that only unconditional external information is used. (Note that all the domain-related messages distributed after each iteration in algorithm as in fig. 2 have the same timestamp corresponding to the barrier point.) The messages generated by an optimistic LP during its run, which have become invalid due to a subsequent rerun, are destroyed without having been read by the recipients.

Distribution of the domain-related messages can be organised in two ways: either in a centralised manner, when the CU reads the output buffers of the LPs and then forwards the messages to their recipients, or directly by the LPs, which write their messages into the input buffers of the recipient LPs. For the prototype implementation, the first way was chosen, since the centralised distribution allows also to easily prepare an integrated data set on the complete system, which can be sent to the optimisation component making some strategic decisions.

The participating LPs are assumed to be deadlock-free. The obligatory control messages at the barrier points, together with the strict pre-defined sequencing of LP group advances depending of the guarantees availability, as well as the information processing order as in fig. 10 ensures the deadlock prevention and a causality respecting synchronisation.

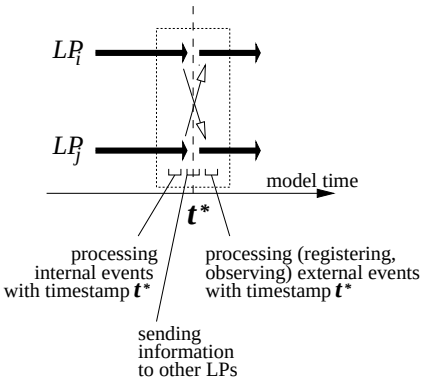

Figure 10: Information processing at the barrier point

CONCLUSIONS

A framework for integrating discrete-event and continuous simulation models implemented by different tools into a combined model is presented. The framework uses a simply-to-realize and transparent

Table 3: Fragment of a synchronisation management table

LP	request No.	request time	command	t_{start}	t_{end}	...	reply No.	reply time	int. events	...
transport	3	5.0	forward	7.0	10.0	...	4	9.0	true	...
refinery	7	9.0	rerun	7.0	9.0	...	6	10.0	false	...
...	...	...	...	...	...	...	...	...	...	...

synchronisation scheme. This scheme is based on accounting only the predefined sets of interaction-relevant events and on special rules for preventing causality violation. The synchronisation control performs the runs of groups of logical processes depending on their rollback capabilities. The framework can be used both for a rapid prototyping of complex models of hybrid systems, and for in-depth simulation analysis of a complete system whose individual components' models are available.

ACKNOWLEDGEMENTS

This research was supported by grant UMTS 126 ("SILVER") of German Federal Ministry for Education and Research.

REFERENCES

Alur, R.; T. Dang, ; J. Esposito; Y. Hur; F. Ivancic; V. Kumar; I. Lee; P. Mishra; G. Pappas; and O. Sokolsky. 2003. "Hierarchical modeling and analysis of embedded systems." *Proceedings of the IEEE* 91, No. 1, 11-28.

AspenTech. 2003. *Aspen Engineering Suite.* http://www.aspentech.com/products

AutoMod. User's Manual v 10.0. Vol. 1,2. Brooks Automation, Inc. - AutoSimulations Division. 2001.

Banks, J. (Ed.) 1998. *Handbook of Simulation.* John Wiley & Sons, Inc. Prentice-Hall, N.J.

Banks, J.; J.S. Carson, II; B.L. Nelson; and D.M. Nicol. 2001. *Discrete-Event System Simulation.* Prentice-Hall, N.J.

Barton, P.J. and C.K. Lee. 2002. "Modeling, Simulation, Sensitivity Analysis, and Optimization of Hybrid Systems." *ACM Transactions on Modeling and Computer Simulation* 12, No. 4, 256-289.

van Beek, D.A.; J.E. Rooda; and M. van den Muyzenberg. 1996. "Specification of Combined Continuous-Time/Discrete-Event Models." In *Proc. 1996 European Simulation Multiconference.* Budapest, June 1996, 219-224.

van Beek, D.A.; J.E. Rooda. 2000. *Multi-domain Modelling, Simulation, and Control.* In *Proceedings of 4th International Conference on Mixed Processes: Hybrid Dynamical Systems (ADPM2000)*, Dortmund, 139-146.

ChemCAD. 2003. *CHEMCAD Process Simulation Software.* http://www.chemcad.fr/en/index.html.

eM-Plant 6.0: *Objects Manual* and *Reference Manual.* Tecnomatix. 2001.

DMSO (Defence Modeling and Simulation Office of U.S. Department of Defense). 2003. *High Level Architecture.* https://www.dmso.mil/public/transition/hla

Esposito, J.M.; G. Pappas,; and V. Kumar. 2001. "Accurate event detection for hybrid systems." In *Proceedings of HSCC2001 (Hybrid Systems: Computation and Control)*, Rome, Italy, March 2001, 204-217.

Fraunhofer UMSICHT. 2003. *Anlagensimulation mit WinZPR.* http://www.umsicht.fhg.de/WWW/UMSICHT/Produkte/software/zpr/index.html

Fujimoto, R.M. 2000. *Parallel and Distributed Simulation Systems*, John Wiley & Sons.

Kim, Y.J. and T.G. Kim. 1998. "A Heterogeneous Simulation Framework Based on The DEVS Bus and the High Level Architecture". In *Proceedings of the 1998 Winter Simulation Conference.* Washington, D.C., 421-428.

Lamport, L. 1978. "Time, clocks, and the ordering of events in a distributed system." *Communications of the ACM* 21, No. 7, 558-565.

Fraunhofer (ITWM, IML, UMSICHT, FIT). 2003. *SILVER: Simulationsbasierte Systeme zur Integration logistischer und verfahrenstechnischer Entscheidungsprozesse.* Berichte zu Arbeitsbereichen 1 ("Leitszenario"), 2 ("Gesamtkonzepte"), 3 ("Gekoppelte Modelle zur hybriden Simulation").

Turton, R.; R.C. Bailie; W.B. Whiting; and J.A. Shaeiwitz. 2002. *Analysis, Synthesis, and Design of Chemical Processes.* Prentice Hall.

Zeigler, B.; H. Praehofer; and T.G. Kim. 2000. *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems.* Academic Press, London.

AUTHORS

Dr. Alexander Lavrov is a Research Scientist at the Optimisation Department of Fraunhofer-Institute for Industrial Mathematics (ITWM) in Kaiserslautern (Germany).

Dr. Dietmar Hietel is a Research Scientist at the Department of Transport Processes of Fraunhofer ITWM.

Dr. Stefan Nickel is the Head of the Optimisation Department of Fraunhofer ITWM and Professor at the University of Saarbrücken (Germany).