A Conceptual Methodology for Conducting Faster Than Real Time Experiments

D. Anagnostopoulos, M. Nikolaidou, P. Georgiadis

Dept. of Informatics, University of Athens
Panepistimiopolis, Athens, Greece
email: dimosthe@di.uoa.gr

Keywords: Real Time Simulation, Simulation Methodology

Abstract

Faster than real time simulation concepts are nowadays applied in multidisciplinary areas. Interaction between the real world and the simulation model is performed in real time, so that information obtained from each system is used to enhance our knowledge of the other. Even if such capabilities prove to be useful for real time applications, dealing with the requirements imposed is only accomplished through extending the traditional simulation methodology. Since model execution is performed faster than real world time, experimentation is considered as the most critical phase. A methodological approach towards an experimentation framework for interacting in real time with the actual system is introduced and analyzed in terms of its functionality, the activities it incorporates and the restrictions imposed. Realization of this methodological framework is also discussed through a simulation example.

1. Introduction

Real time simulation was introduced more than two decades ago and is widely adopted in many domains, as training and process control[1]. Simulation of real time systems requires a model that is accurate enough to accomplish the simulation objective and is computationally efficient[2]. In most cases, the term *real time*, as it relates to simulation, denotes that advancement of simulation time must occur in the real world time (i.e. not faster or slower). This issue is critical, since the model represents an actual dynamic process as it occurs. Real time simulation enables a more realistic representation of the system being studied, as opposed to both mathematical analysis and conventional simulation and also permits both quantitative and qualitative evaluation.

Based on the above, faster than real time simulation denotes that advancement of simulation time occurs faster than real world time. Making models run faster is the modeler's responsibility and certainly not a trivial task, since real time systems often have hard requirements for interacting with the human operator or other agents. A significant disadvantage is that timing problems are recognized during or even after testing. Researchers have thus pointed out that timing requirements should be addressed in the design phase[3]. Towards this direction, two activities of the architectural design are defined: the logical architectural design and the physical architectural design. The latter embraces the non-functional requirements and forms the basis for asserting that these will be met once the detailed design and implementation have taken place. Appropriate scheduling paradigms are often integrated to handle non-functional (e.g. timing) requirements[3].

Towards meeting this objective, OOPM/RT is a multimodeling methodology based on the idea of selecting the appropriate model among multiple model types through trading structural information for faster runtime, while minimizing the loss of behavioral information[2,4]. A set of multiple methods is generated through abstraction techniques and the optimal abstraction degree is selected to compose a model for the real time simulation. The selected optimal model guarantees delivering simulation results by the given amount of time. The Design-to-

Time method is an alternative method suggesting the use of a single model type and multiple methods generated through approximation techniques[5].

When successfully dealing with the real time requirements, simulation provides improved capabilities in the direction of reaching conclusions for the system future state. Nevertheless, even though faster than real time simulation is used extensively for high interactive applications[6], there is no generic framework for such experiments. Towards this direction, a methodological approach has been introduced by the authors[7].

In this approach, faster than real time simulation is used to denote the enhanced simulation process where model evolution occurs faster than the real world, thus enabling information, originating as output of the experimentation phase and, moreover, from the actual system itself, to be used to improve the effectiveness of the simulation experiment.

The bi-directional information exchange between the model and the actual system is depicted in figure 1. The left information flow is the conventional one, while the other is the one established to obtain information about the actual system evolution. This figure is often extended to include the human operator participating in real time experiments, as a closed man-in-the-loop system. This is not a necessity, however, since control tasks are often performed in an automated way, without the operator intervening.

Even though information exchange between the model and the actual system can only occur in discrete time points, this does not exclude systems evolving in continuous time. Either discrete or continuous systems can be studied through a common methodology.

Introducing faster than real time simulation in the current context was motivated by the existence of numerous systems that cannot be fully specified during an initial, single specification stage. Due to their dynamic nature, any initial knowledge about them is modified according to the conditions of their real evolution. The basic idea is making such systems serve as an information source, instead of adopting doubtful statistical distributions, when representing the conditions of their evolution.

The term *non-predetermined* is thus used to denote actual systems that may be unpredictably reformed during their evolution. In most cases, reformations occur due to interventions caused by external operators. Non-predetermined systems cannot be specified during the single modeling phase, since reformations have a strong impact on the key characteristics of the actual system, as its structure and input data. Two main types are thus considered: structure and input data reformations. Structure reformations involve the system components and the coupling relation between them, while input data reformations involve the critical system operation parameters subjected to modifications.

When attempting to predict the behavior of such systems, traditional simulation modeling cannot provide reliable results, since incorporating into the model any occurring reformations is not possible. The faster than real time approach provides a concise framework towards this direction. Important methodological aspects are discussed in the following, while emphasis is given to experimentation, as the simulation phase facing critical requirements imposed by the real time dimension.

2. Simulation Methodology

The perception we have for the traditional simulation process is depicted in figure 2.

In this process, when modeling is completed, the model is subjected to modification only depending on the output analysis results and this causes a feedback to the model construction phase. However, system specifications are never modified while experimenting with the model. In real time experiments, on the other hand, any reformations occurring during experimentation must be handled as modifications to the original system specifications, as indicated with the dotted line. System specification should therefore be an iterative process and output analysis must be performed in real time and also be iterative, since it is not only invoked when experimentation is terminated.

Real time interaction with the system being studied imposes an increased degree of complexity. The introduced faster than real time methodology extends the traditional simulation phases to incorporate additional tasks and includes two new phases. The methodology thus consists of the following individual phases:

- Modeling
- Experimentation
- Remodeling
- Plan Scheduling

The invocation mechanism for the four phases is depicted in figure 3.

2.1 Modeling Phase

As the initial simulation phase, modeling includes system specification and model construction and is considered to be time consuming. There are no main differences compared to the traditional approach, except from the requirement for automated model generation. Acquiring generation capabilities is not crucial for the initial model construction activity, since modeling precedes experimentation and is not performed in real time. However, the techniques used determine the efficiency of model modification tasks performed in real time.

The experimentation framework introduced is not associated with a specific modeling formalism. However, remodeling requirements impose the use of modular models to handle structural reformations. Modular models often have a hierarchical structure according to which components are coupled together to form larger models. The formalism in which models are expressed should therefore make recursive structuring possible[8]. Coupling concept combined with the object oriented paradigm enables late binding, an essential feature for accomplishing remodeling without recompilation. A hierarchical, modular formalism supporting object orientation, as DEVS-Scheme[9], can therefore serve as a theoretical basis for model specification, since it has already been extended to meet the requirements of real time simulation[10,11].

In this sense, remodeling requirements are handled through object-oriented modeling and use of preconstructed model components, which are organized in object hierarchies and reside in model libraries. Preconstruction of primitive and composite models is enabled for all higher level entities corresponding to the accepted primitive entity combinations. Preconstruction is expected to extend to the level where structural reformations may be encountered.

Component preconstruction also enables automated model generation. Whenever additional primitive models are constructed and inserted in object hierarchies, the corresponding composite models can be simultaneously derived so that their availability is ensured.

2.2 Experimentation Phase

Experimentation is the main phase in the proposed simulation framework. It implements key features of the extended functionality and, thus, acquires an increased degree of complexity. It also includes a part of output analysis phase. The dominant factor when performing experimentation is handling of real time and achieving faster than real time model evolution. Experimentation includes a number of activities discussed in section 3.

2.3 Remodeling Phase

Invocation of this phase depends on the outcome of experimentation. As discussed in the following, there are two cases in which remodeling is invoked. In the first, reformations occurred to the actual system must be incorporated into the model. In the second, deviations detected during output analysis between the evolution of the system and the model impose serious doubts for the reliability of the model, thus leading to remodeling. To conclude, remodeling is viewed as an iteration of the original modeling phase, aiming at restoring consistency between the actual system and the model.

Remodeling is executed in real time and includes fewer steps than the original phase since modeling decisions are already taken and model components are implemented. It is important to note that experimentation may resume its operation only after remodeling is completed and that a warm up period is needed whenever remodeling is performed.

2.4 Plan Scheduling Phase

Plan scheduling concepts are already introduced and used in diverse domains[12]. In this case, plan scheduling denotes the activities of the last simulation phase, which is invoked only when specific conditions of the real time experiment are satisfied. Plan scheduling implements specific aspects of the traditional output analysis and decision making process and does not coincide with the generic output analysis framework. Since it only takes advantage of simulation predictions under specific circumstances, if simulation predictions were to be used in an alternative way, plan scheduling could be as well substituted by an appropriate activity.

In this context, simulation results obtained are used to generate and choose the most efficient plan for the current situation. Plan scheduling is responsible for evaluating the current and predicted states and proposing an appropriate plan, aiming at leading the actual system to an acceptable state. Even though plans may be proposed and scheduled, their execution is performed externally to the simulation environment. In this way, any reformations caused can be handled appropriately, as external reformations to the actual system. Thus, a simulation environment built on the basis of the introduced guidelines only contributes to the control process of the actual system.

3. Experimentation

When experimenting in real time, the model and the actual system evolve concurrently. The state of the actual system always relates to the current real time point. In the general case, the state of the model, as denoted by simulation time, may correspond to a previous, future or the current state of the actual system. As expected, in faster than real time simulation, the model refers only to future time points.

These concepts are presented in figure 4. Evolution of both systems is depicted on the two horizontal axons. Real time points are noted as t_i, where i is an index denoting the sequence of time points (e.g. if $k < m$ then t_k precedes t_m). The state of the system in point t_i is noted as R_i. The state of the model is noted as S_i.

When at time point t_i the model refers to the system state at time point t_j (simulation time is equal to t_j) we use the notation $S_i \Rightarrow R_j$. As expected, the condition for performing real time simulation is $S_i \Rightarrow R_j$, $i = j$, as in case (b) in figure 4, and the condition for faster than real time simulation is $S_i \Rightarrow R_j$, $i < j$, as in case (a).

Experimentation includes the following main activities: monitoring, auditing and decision making.

3.1 Monitoring

This activity is necessary for implementing the bi-directional information exchange that enables the reliability of simulation results. Monitoring (i.e. observation of both systems evolution) is performed continuously in order to obtain model data and real observations. There are two key issues that need to be resolved: which data are collected and when these are obtained.

Dealing with these issues depends on the system under monitoring. When referring to the actual system, only data denoting its current state can be obtained at any point t_n. This includes data denoting that reformations have occurred to the actual system. Monitoring activity is responsible for indicating and expressing reformations through appropriate measures.

This does not apply to model monitoring, since simulation time refers to future points. Monitoring activity is responsible for collecting data describing the preceding and the current model states, as indicated by simulation time. These are used to perform the necessary comparisons between the corresponding states (i.e. referring to the same real time points) of the system and the model. Model data must thus be collected when the corresponding system states are predicted.

In figure 5, for instance, if actual system data are obtained at time point t_n, simulation data should have been obtained when this specific state was predicted, at time point t_x. Thus, when $S_x \Rightarrow R_n$, $x < n$, system are model data are collected in real time points t_n and t_x, respectively.

Monitoring data are expressed in the form of appropriate *monitoring variables*, which are commonly defined for both the model and the system. Comparisons between corresponding states are thus made in terms of monitoring variable values. The case where values of specific measures cannot be directly obtained from the system can be handled through the use of alternative variables expressing measures closely related to the required ones and also through appropriately forming the comparison algorithm between the corresponding states of the two systems.

Determining the time points where real system data are obtained must be accomplished before the experiment initiates, since model monitoring should be aware of them in advance. Monitoring decisions are therefore expressed as part of control parameters, which must be forwarded to the simulation environment prior to real time experimentation.

3.2 Auditing

Auditing is the main experimentation activity, which examines model data and real observations in order to determine:

a) Whether faster than real time experimentation conditions are satisfied

b) Whether the model represents efficiently the actual system

c) If any measures should be taken

As a term, auditing is introduced in order to emphasize on the comparison process between the corresponding states of the system and the model. Monitoring, which in this paper refers only to observation process, could as well be used to denote the concepts of auditing.

When performing faster than real time simulation, certain conditions must be satisfied: simulation time should ad hoc advance faster than the real world time and monitoring data must be available when auditing is initiated. In this case, consequent states of model and system evolution, as recorded during monitoring, are analyzed and compared. Monitoring data depict both systems evolution within specific time intervals. Due to the requirement for comparing data referring exactly to the same time points, these intervals should be predetermined and common for both systems. Since this interval is also common for monitoring and auditing activities, the term *auditing interval* is introduced to denote it.

This is not the only alternative, however, since it is also possible to obtain data describing the evolution of both systems for shorter periods of time, that is, to have a shorter monitoring interval than auditing interval. This could be the case when requesting more than one experimental data set for a single auditing interval. The opposite case (i.e. having a shorter auditing interval) is obviously not possible. Nevertheless, in the general case, these two intervals are considered to be identical.

In figure 5, the auditing interval is of the form $(t_{n+i,}\ t_{n+i+1}]$. In addition, we have that $S_x \Rightarrow R_n$, $x < n$ and $S_n \Rightarrow R_y$, $n < y$. Based on the above, auditing is performed right after an auditing interval has elapsed (e.g. at time points t_{n-1}, t_n and t_{n+1}). Each time auditing initiates, a single comparison is made between the corresponding states of both systems, as for time point t_n, where S_x is compared to R_n.

Analysis and comparison between the corresponding states result in either of three possible scenarios, as illustrated in figure 3.

1. The actual system has been reformed during the last time interval: All simulation results referring to future system states are discarded. Since indicated reformations must be incorporated into the simulation model, remodeling should be performed to restore

consistency between the system and the model. When remodeling is completed, experimentation resumes.

2. Deviations are detected between the evolution of the actual system and the simulation model: Deviations can emerge due to the stochastic nature of simulation when comparing corresponding states of the two systems, even when reformations have not occurred. When deviations are detected, it is obvious that the previous modeling attempt has failed and remodeling should be performed. When remodeling is completed, experimentation resumes.

3. Simulation results obtained are considered to be reliable: This can only be accomplished when the current and previous comparisons between data from the two systems prove that simulation predictions do not deviate from the actual system evolution. Predictions are therefore considered to be reliable and simulation results are to be further analyzed. This task strongly depends on the application domain. Our orientation is not limited to gaining knowledge for the future system states, but extends to controlling the system behavior through intervening to the conditions the system is subjected to in the near future. Evaluation is therefore performed to determine whether the predicted states are acceptable on the basis of predetermined criteria. The term *acceptable* is used to denote that future states do not deviate from what is expected under normal conditions. In this case, no intervenes are needed. If the predicted states are not acceptable, measures should be taken and alternative plans are thus examined. Plan scheduling activity is invoked, while experimentation resumes.

Indication of reformations and deviations is accomplished on the basis of appropriate monitoring variables. To correspond to the above scenarios, monitoring variables should denote the system components, the coupling relation between components, critical system parameters and statistical measures representative of the system behavior. It is also required to have pre-determine how close values of model variables should be to the corresponding ones of the system. For this task, comparison methods between real observations and model data can be used, as the confidence interval or the inspection approach[13].

Inspection of whether faster than real time experimentation conditions are satisfied precedes the above comparisons, since it takes place right after the auditing interval has elapsed, and ensures that the experiment advances as expected. In case that simulation time is less than real time, as in case (c) in figure 4, simulation results are discarded. If this is a single occurrence, the experiment can advance to the next auditing interval; otherwise, prediction cannot be achieved. Evidently, achieving faster than real time simulation in only a limited number of auditing intervals cannot ensure the reliability of predictions.

Having considered this possibility, it is useful to describe the terminating conditions for a simulation experiment conducted according to the methodology introduced. When remodeling or plan scheduling phase are invoked, the experiment is not terminated. In the first case, experimentation pauses until remodeling is completed; in the second, experimentation continues without considering the outcome of this phase. The experiment is terminated, however, when the actual system operation is terminated or when faster than real time simulation cannot be achieved. Common pre-specified stopping conditions, as the length of the simulation experiment which, in this case, is expressed in real or simulation time, can also be used.

3.3 Decision making

Decision making is the last experimentation activity, which is invoked when auditing is completed. Its purpose is to carry out the decisions taken during experimentation. Thus, if conditions of scenario (1) or scenario (2) are satisfied, remodeling phase is invoked. Otherwise, depending on the outcome of simulation results evaluation, decision making can either invoke plan scheduling, when the predicted states are not acceptable, or just terminate. In the latter case, monitoring of both systems evolution is re-initiated.

The flow of control for the proposed simulation methodology is depicted in figure 6.

4. Potential Offered

Faster than real time experimentation enhances considerably the potential of the traditional simulation process. It also enables new application areas to emerge, as in dynamic process control and training, where efficient solutions cannot always be given through traditional simulation. The potential offered through the proposed methodology is briefly summarized in the following:

1. Ensuring the consistency between the simulation model and the actual system: System identification can be accomplished when performing monitoring and remodeling, starting from an initial, probably inaccurate model.
2. Increased interactivity: Real time simulation is based on the continuous interaction between the real and simulation world where the human operator may also participate, as a closed man-in-the-loop system. Training, as well as other relevant applications, is thus supported.

3. Ensuring the validity of the model: An automated model validation process is implemented when monitoring and remodeling the initial system representation for a number of consecutive time intervals.

The ultimate purpose of faster than real time simulation is to ensure the reliability of predictions for the future system state. This is accomplished on the basis of the knowledge we obtain up to the current time point for the system behavior when simulation predictions are confirmed for a number of consecutive intervals, assuming that no further reformations will occur in the near future.

Nevertheless, if faster than real-time simulation fails (i.e. simulation time is less than world time), the proposed framework can still be used for system identification and model validation, since the model can be adapted to the current system state, but not towards extracting predictions for the near future.

5. Simulation Example

In this section, we present a prototype faster than real time application for computer networks[14]. This application domain is used since it enables dealing with both structure and input data reformations, according to the simulation approach introduced, and also data exchange between the system under study and the model. However, since it is questionable whether faster than real time simulation can be achieved for all network types, the actual network under study is a 10BaseT local network, which is relatively slow.

Since networks can be viewed as variable structure systems, dealing with structure modifications during the simulation experiment imposes that modular models are used[15]. Modular models have a hierarchical structure according to which components are coupled together to form larger models. Remodeling requirements (especially the one for dynamic binding) are handled through object-oriented modeling and use of preconstructed model components, which are organized in object hierarchies. Since preconstructed models correspond to all network entities, the overall set of acceptable entity combinations must be supported. Preconstruction of primitive and composite models is thus required for the higher level entities corresponding to the accepted combinations and extends to the level where structure modifications may be encountered (nodes and applications). Component preconstruction also enables automated model generation. In this way, if additional models of an entity were to be provided (e.g. protocol models) composite models making use of them would be directly formed.

Network models are composite and consist of primitive and composite models, as nodes, communication protocols and applications. An example composition scheme for a network with four processing nodes and their corresponding applications is abstractly depicted in figure 7. This figure includes two types of links: *composition* and *in_out*, indicating the network components and the coupling relation between them, respectively. Interconnection between applications is also depicted.

Common examples where remodeling is caused are when a significant application initiates or terminates, an active node crashes or a processing node is activated. In these cases, structure, input data or even both reformation types are likely to occur. Auditing is executed when the preceding interval is completed and exploits monitoring variables from both systems to conclude for potential reformations or deviations.

Assuming that $node_2$ in figure 7 is no longer active, monitoring data are cross-examined to indicate the specific node and applications affected. Auditing indicates that $node_2$ and applications $appl_{21}$, $appl_{22}$, $appl_{12}$, $appl_{41}$ are no longer active and concludes that structure reformations have occurred during the preceding interval. Remodeling is thus invoked, which removes and disposes the corresponding components from the model composition tree. When all required modifications are accomplished, the model is once more subjected to experimentation, starting from the current real time point.

In case that other reformations impose the integration of additional components, as when another node is activated, the appropriate models are directly imported from model libraries, initialized and coupled to the existing components. Remodeling may also be caused due to input data reformations when critical parameters, as communication protocol parameters and the data unit generation rate of an application, are modified. Finally, deviations between the model and the network, expressed through appropriate statistical measures, as the end-to-end delay, may also lead to remodeling, even when reformations have not occurred. In all above cases, the network model is modified without recompilation or the human operator intervening, in minimum time.

Even though remodeling tasks are not trivial, this time overhead must be minimized to ensure that faster than real time simulation is achieved in the next auditing interval. Attention should also be given to resetting the simulation clock back to the current real time point, since experimentation is re-initiated from this point. Except from resetting the clock and the event list, since previously scheduled events must be disposed, model entities should also be no longer associated with future time points.

Efficient integration of new models within the aggregate network model relies on ensuring their availability. Automated model generation contributes to this through enabling the creation of all composite models for the lately constructed primitive models. An example is presented in figure 8 for modeling the protocol stack of network nodes, where a new protocol model (*100BaseT*) is provided. Hierarchical layering is used. The model hierarchy on the right side of the figure is directly formed through applying a set of rules determining the higher layer models to be built, based on the corresponding protocol compatibility rules.

When simulation predictions for the near future are reliable, plan scheduling is responsible for evaluating the predicted network states and proposing an appropriate plan so that critical measures (e.g. delay and throughput) remain within acceptable limits. Execution of proposed actions is performed externally from the simulation environment so that any reformations encountered will be appropriately handled. Operating on the basis of these guidelines, the simulation environment contributes - but does not intervene - to network management and performance evaluation.

6. Conclusions

The proposed faster than real time methodology establishes directions for conducting experiments within a well-described framework. This methodology enables the enrollment of simulation experiment evolution in real time and provides the means for enhancing the effectiveness of the decision making process through reaching reliable conclusions for the near future and ensuring the validity of the model. The issue of making models run faster than real time was not considered in this paper; there are other research activities, however, oriented towards this objective.

To exercise faster than real time simulation, certain requirements must be met. These include making the model run faster than the real world, monitoring the appropriate system behavioral characteristics and minimizing the time required for auditing and remodeling. Parameterization of the simulation environment according to the specific application domain details is also significant for the experiment effectiveness due to the inherent complexity of auditing activity. The concepts of the proposed methodology are not yet expressed through a well-defined modeling formalism, which would be especially useful for remodeling, in order to provide a complete specification of how the various reformation types are handled. Current and future research of the authors is focused on these open issues.

References

1. Cleveland J. et al., *Real Time Simulation User's Guide: The Red Book*, Analysis and Simulation Branch, NASA Langley Research Center, 1997
2. Lee K., P.A. Fishwick, "Generation of multimodels and selection of the optimal abstraction level for real time simulation", in *Proceedings of AeroSense 98*, 1998
3. Burns A., A. Wellings, "Hrt-Hood: A structured design method for hard real time systems", *Real Time Systems*, vol. 6, pp. 73-114, 1994
4. Lee K., P.A. Fishwick, "OOPM: An object-oriented multimodeling and simulation application framework", *Simulation*, vol. 70, no. 6, 1998
5. Garvey A.J., V.R. Lesser, "Design-to-time real-time scheduling", *IEEE Transactions on Systems, Man and Cybernetics*, vol. 23, no. 6, pp. 1401-1502, 1993
6. Kramer B. et al., "Developing an expert system technology for industrial process control: An experience report", *Lecture Notes in Computer Science*, 1081, pp. 172-186, 1996

7. Anagnostopoulos D. et al., "Establishing the real time simulation concept through a four phase methodology", in *Proceedings of European Simulation Multiconference (ESM'95),* SCS, Prague, 1995

8. Zeigler B.P., *Object-oriented simulation with hierarchical, modular models*, copyright by Author, 1995 (originally published by Academic Press, 1990)

9. Zeigler B.P. et al, "The DEVS Environment for high-performance modeling and simulation", *IEEE Computational Science and Engineering*, vol. 4, no. 3, 1997

10. Hong S. Jun et al, "A Real-time Discrete Event System Specification Formalism for Seamless Real-time Software,"*Discrete Event Dynamic Systems*, vol. 7, pp. 355-375, Oct., 1997

11. Cho M. Sung, Tag G. Kim, "Real-time DEVS Simulation: Concurrent, Time-Selective Execution of Combined RT-DEVS Model and Interactive Environment", in *Proceedings of Summer Computer Simulation Conference* (SCSC '98), pp. 410-415, July, 1988, Reno, Nevada

12. Lee J., P. Fishwick, "Real time simulation-based planning for computer generated force simulation", *Simulation*, vol. 63, no. 5, 1994

13. Law A.M., W.D. Kelton, *Simulation Modeling and Analysis,* McGraw Hill, 1991

14. Anagnostopoulos D. et al, "Computer Network Performance Prediction for the near Future through Faster than Real Time Simulation", to appear in *Proceedings of Symposium on Performance Evaluation of Computer and Telecommunication Systems '99 (part of SCSC '99),* Chicago, 1999

15. Pawletta T. et al, "An Object Oriented Framework for Modeling and Simulation of Variable Structure Systems", in *Proceedings of Summer Computer Simulation Conference (SCSC '96)*, Portland, 1996

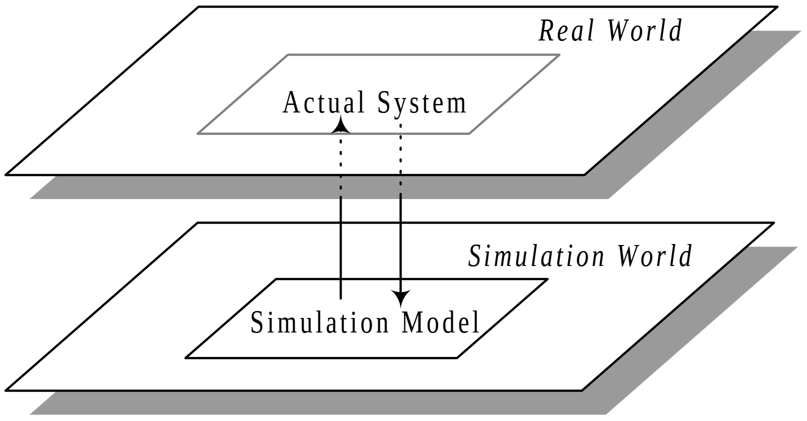

Figure 1: Information exchange between the simulation model and the real world

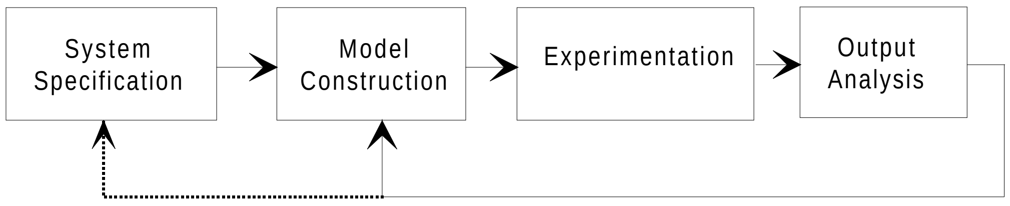

Figure 2: Feedback to system specification phase

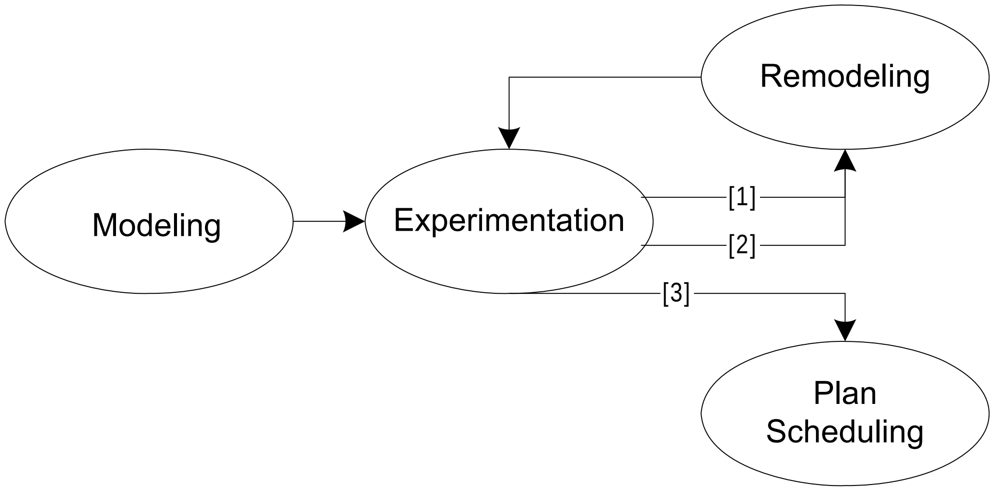

Figure 3: Scheduling of individual phase execution

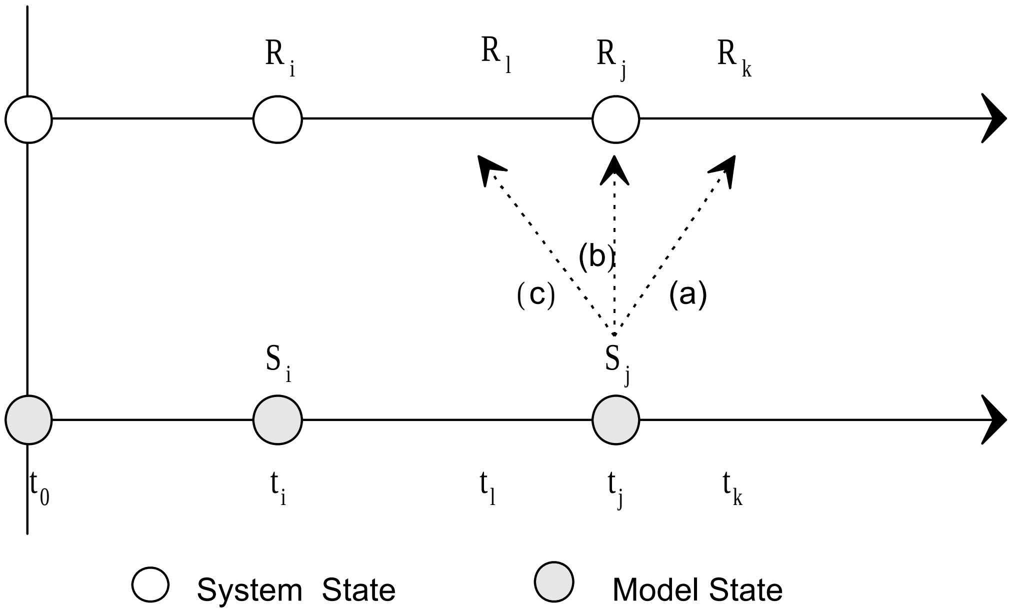

Figure 4: Actual system and simulation model concurrent evolution

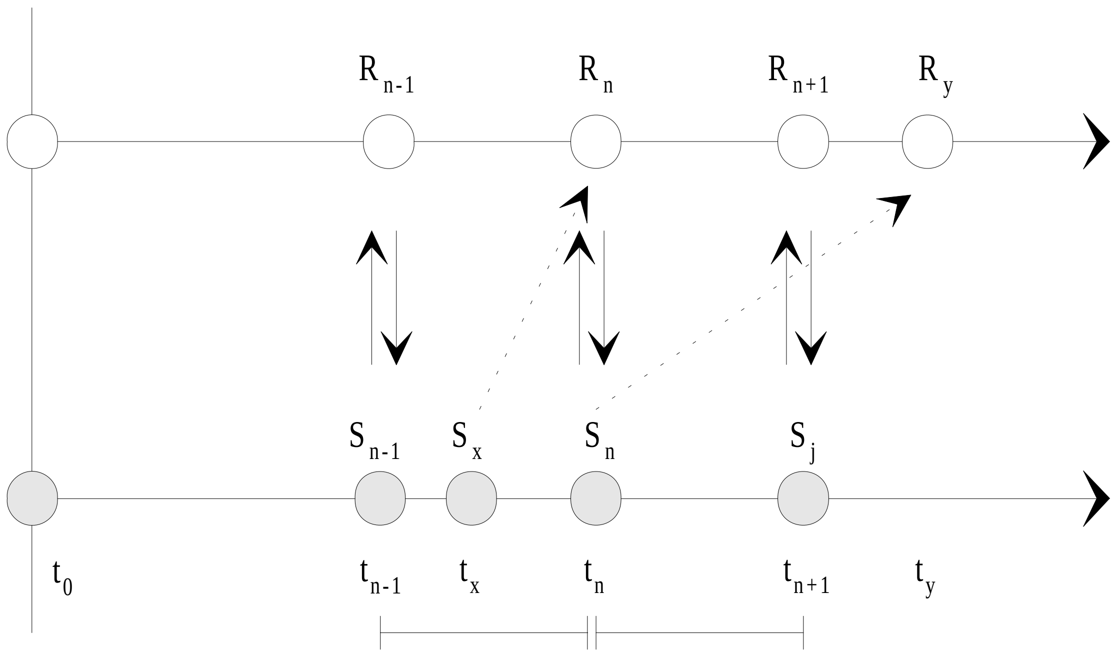

Figure 5: Experimenting with the actual system and the simulation model

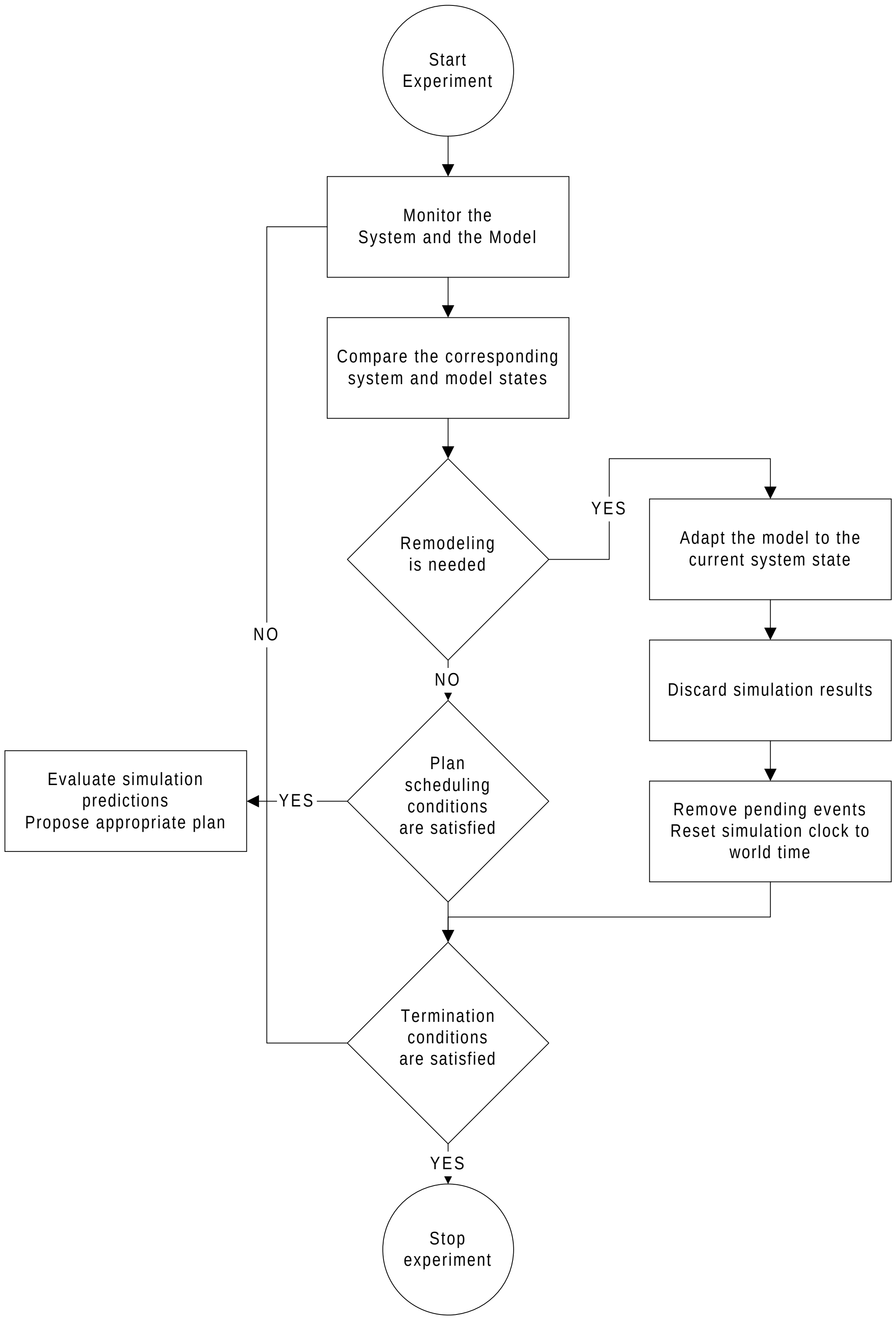
Start Experiment
Monitor the System and the Model
Compare the corresponding system and model states
Remodeling is needed
YES
Adapt the model to the current system state
Discard simulation results
Remove pending events Reset simulation clock to world time
NO
NO
Plan scheduling conditions are satisfied
YES
Evaluate simulation predictions Propose appropriate plan
Termination conditions are satisfied
YES
Stop experiment

Figure 6: Control flow for the proposed simulation methodology

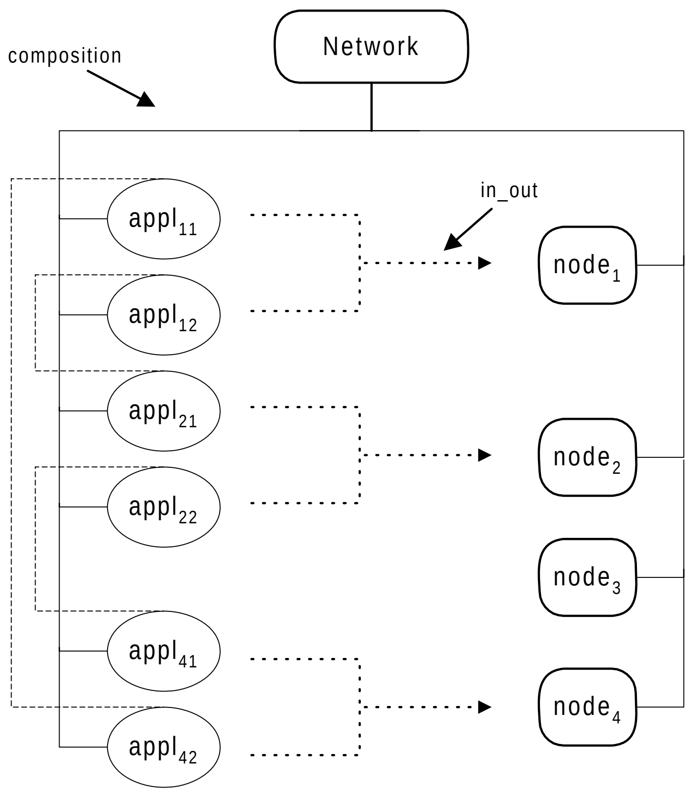

Figure 7: Abstract view of network model composition

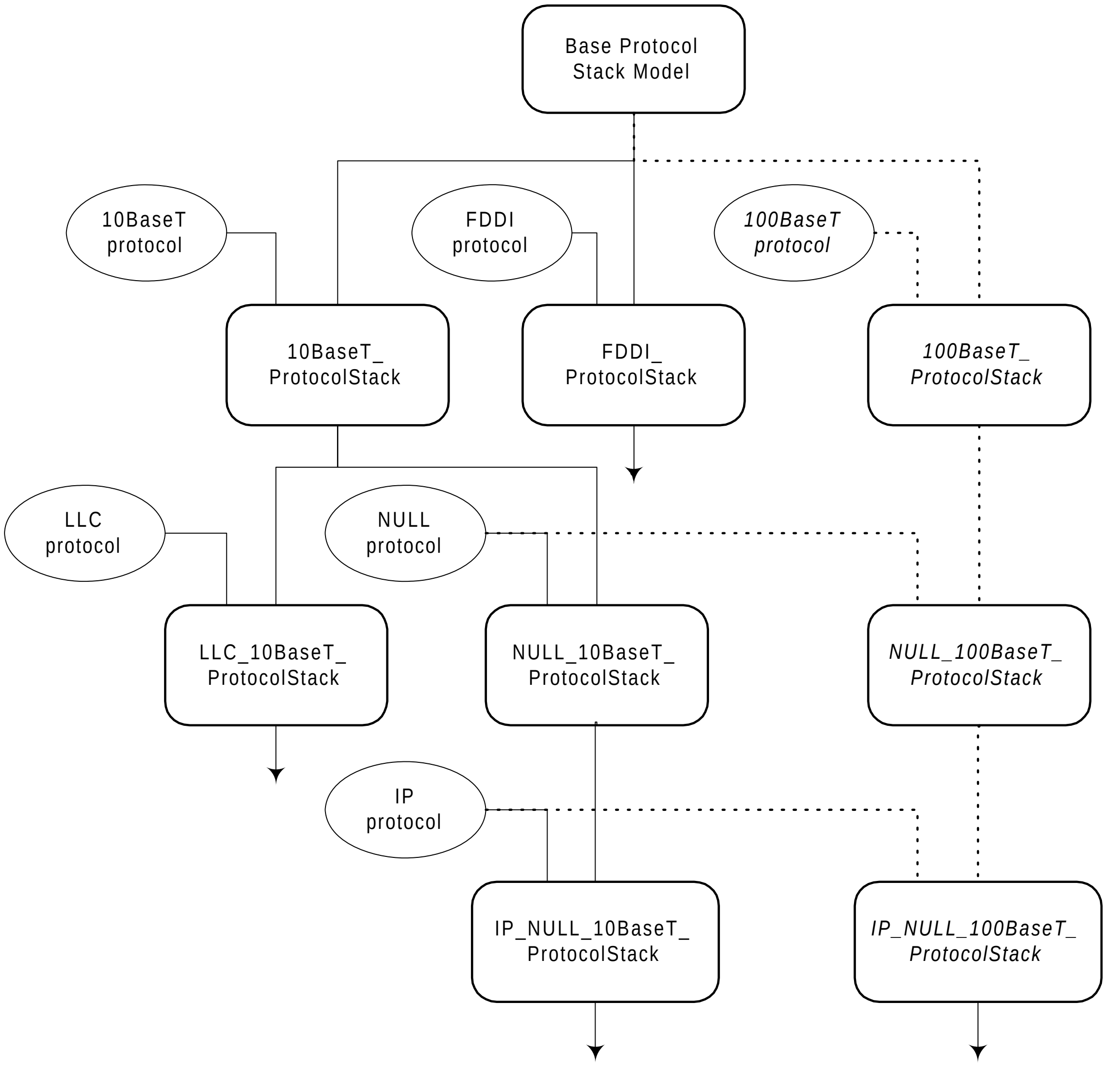
Base Protocol Stack Model
10BaseT protocol
FDDI protocol
100BaseT protocol
10BaseT_ ProtocolStack
FDDI_ ProtocolStack
100BaseT_ ProtocolStack
LLC protocol
NULL protocol
LLC_10BaseT_ ProtocolStack
NULL_10BaseT_ ProtocolStack
NULL_100BaseT_ ProtocolStack
IP protocol
IP_NULL_10BaseT_ ProtocolStack
IP_NULL_100BaseT_ ProtocolStack

Figure 8: Automated model generation for protocol stack modeling