HANDLING SYSTEM VARIABILITY IN REAL TIME, DISCRETE EVENT EXPERIMENTATION

D. Anagnostopoulos
Dept. of Informatics, University of Athens
Panepistimiopolis, Athens, Greece, 15771
email: dimosthe@di.uoa.gr

KEYWORDS
Faster than Real Time Simulation, Simulation Methodology, Structure Variable Systems

ABSTRACT
Contemporary systems, as communication networks and transportation systems, are characterized by variable structure, resulting in the change of structure and rules of behavior. When attempting to reach conclusions for the behavior of these systems in the near future, traditional simulation cannot provide reliable results, since incorporating into the model any occurring reformations is not possible. The faster than real time approach discussed in this paper provides a concise framework towards this direction. Relevant methodological aspects are discussed in the following, using the DEVS modeling formalism. Emphasis is given to the experimentation phase. Realization of this methodological framework is also discussed through a network simulation example.

1 INTRODUCTION
Discrete event systems have an important role in managing contemporary systems, as communication networks, manufacturing and transportation systems. An operation range of a discrete event system can be specified by a state trajectory which is piece wise constant in time function.

Variable structure models are models that entail in their description the possibility to change their own structure, i.e. their constitutive components as well as the relations that exist amongst them (Urhmacher and Zeigler 1996). In most applications, the change of structure refers to the addition or deletion of single components (Cellier and Elmqvist 1993), the change of interactions between entities or the change of rules of behavior (Lorenz et al. 1989).

In most approaches, structural changing is embedded in an object-oriented modeling environment. The object-oriented paradigm, i.e. definition of classes and their structure in taxonomic hierarchies, provides a nearly natural mechanism for the structuring of knowledge. Problems of redundant and inconsistent models are sidestepped through the concise mechanisms of inheritance. Problems of model variation have been discussed within the context of object-oriented modeling systems, as DEVS (Zeigler 1990) and EMSY (Urhmacher 1992), in most cases for control purposes.

Real Time Simulation (RTS) and Faster than Real Time Simulation (FRTS) are increasingly applied in contemporary systems. 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). Simulation of real time systems requires a model that is accurate enough to accomplish the simulation objective and is computationally efficient (Lee and Fishwick 1998). There are many reasons for the employment of real time simulation: the currently available computing power, enabling conducting on-line, real time experiments, the advances in parallel and distributed simulation and the recent research performed in facing the real time requirements at the methodological level. There is also a strong motivation: 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. Requirements, however, are drastically increased to achieving real time model execution and dealing with the dynamic system behavior, often in the form of structure variability.

Faster than real time simulation (FRTS) denotes that advancement of simulation time occurs faster than real world time. Concerning this first requirement, which is not amongst the key objectives of this paper, making
models run faster is a task of increased complexity and the modeler's responsibility, since real time systems often have hard requirements for interacting with the human operator or other agents. It is reasonable to classify systems in three main categories: fast systems, where simulating in faster than real time is practically impossible, slow systems, where it is rather trivial, and near-computing-time systems. It is for this late category that research is done on how FRTS may be achieved. Even though computing power is intensely growing, most systems (e.g. reactions completed within ms) classification will remain the same. A significant disadvantage is that timing problems are recognized during or even after testing. It is thus pointed out that timing requirements should be addressed in the design phase (Burns and Wellings 1994).

When successfully dealing with the real time requirements, simulation provides an improved potential towards reaching conclusions for the system future state. Since model execution is performed faster than real world time, experimentation is considered as the most critical phase. The authors have introduced a methodological approach towards an experimentation framework for interacting in real time with the actual system (Anagnostopoulos et al. 1999), in a context that it is possible to deal with the dynamic system behavior.

The basic idea is making such systems serve as an information source, instead of adopting a doubtful stochastic behavior. We use the term non-predetermined to denote actual systems that may be reformed during their evolution (the term time dynamic systems has also been used (Pawletta et al. 1996)). In most cases, reformations occur due to interventions caused by external operators. Reformation events have a strong impact on the key characteristics of the actual system, as its structure and operation parameters. Structure reformations involve not only the system components, but also the coupling relationship between them. We thus distinguish the following main reformation types: structure, coupling and operation parameter reformations.

Structure variable systems and multimodel systems (Pawletat et al. 1999) have also been used as terms to denote systems with a variable structure. Reformations cover a wider range than structure variability, since they refer to all potential variations of a system's key characteristics that may dynamically occur.

When attempting to reach conclusions for the behavior of these systems in the near future, traditional simulation 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. Relevant methodological aspects are discussed in the following, using the DEVS modeling formalism. Emphasis is given to the experimentation phase. Realization of this methodological framework is also discussed through a simulation example.

2 METHODOLOGY
The real time dimension imposes the following main requirements for the simulation process compared to the traditional approach:

1. The model has to run faster than the system. Faster model execution is a prerequisite in order to conclude for the system state in the near future.
2. Traffic data must be collected and processed in real time to monitor the current state of the system.
3. To handle structure variability, the model must be adapted to depict the current state of the system. Dynamic model modification must therefore be accomplished during experimentation.
4. The validity of the model must always be ensured.

The FRTS methodology consists of four phases: modeling, experimentation, remodeling and plan scheduling (Anagnostopoulos et al. 1999).

2.1 Modeling and Remodeling
Modeling precedes experimentation and is not performed in real time. Modeling techniques, however, determine the efficiency of model modification tasks performed during remodeling. Remodeling is executed in real time, aiming at restoring consistency between the system under study and the model. 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. Formalistic issues are already discussed on the basis of the DEVS formalism, the most widely used formalism, introduced by Zeigler (Zeigler 1990). Coupling concept combined with the object oriented paradigm enables late binding, an essential feature for accomplishing remodeling without recompilation. The DEVS modeling formalism specifies discrete event models in a hierarchical, modular form (Zeigler et al. 1997). An atomic model M is specified by a 7tuple:

\[
M = \langle X, S, Y, δ_{int}, δ_{ext}, λ, τ, α \rangle
\]

where
- \(X\): input events set,
- \(S\): states set,
- \(Y\): output events set,
- \(δ_{ext}: Q \times X \rightarrow S\): external transition function,
SELECT: 2M - IC relation, EIC M: DEVS components set, Y: output events set, X: input events set, DN = < X, Y, M, EIC, EOC, IC, SELECT > where represented by a coupled model DN:


The three elements EIC, EOC and IC represent the connections between the set of models and the input and output ports X, Y. SELECT function acts as a tie-breaking selector, i.e. it selects the component that produces the next event.

For the definition of the coupled model, DEVS makes use of SES, the system entity structure (Zeigler 1990). SES is a knowledge representation scheme that combines taxonomy, coupling and decomposition. The SES plays an important role in modeling structural change, as it restricts the possible structure that the overall model can assume. Structural changes in DEVS are realized through controllers, called endomorphic agents, which are embedded into the original DEVS layer. Structural changes include cases as the creation, deletion and exchange of models, each case acquiring its own degree of complexity.

Invocation of remodeling depends on the outcome of experimentation. There are two cases in which remodeling is invoked: when reformations must also be depicted in the model and when deviations detected during output analysis between the evolution of the system and the model impose serious doubts for the reliability of the model. Remodeling is viewed as an iteration of the original modeling phase, aiming at restoring consistency between the actual system and the model.

Remodeling requirements at the implementation level are supported through the use of preconstructed model components. Components are organized in object hierarchies and reside in model libraries (or model bases). Preconstruction of primitive and composite models is enabled for all higher level entities corresponding to the accepted primitive entity combinations and extends to the level where structural changes are encountered, to ensure component availability.

Component preconstruction should also be used in combination with automated model generation features. When constructing new primitive models and inserting them in object hierarchies, the corresponding composite models must be simultaneously derived, also to ensure their availability.

3 EXPERIMENTATION

Experimentation phase encompasses monitoring and auditing (the activity of examining whether both systems are evolving towards the same direction). Time restrictions are imposed and concern a) the model execution, since it must be faster than the actual system b) auditing, which must be completed with minimum time overhead, since all other activities are paused prior to concluding for the model evolution and c) remodeling, when restoring consistency is needed, for the same reasons.

Performance evaluation can be efficiently accomplished when handling all frequent scenarios where the system under study is subjected to modifications, as the addition of new components or component exchange. The model must then be customized to the new conditions and still be in position to ensure the reliability of results. To achieve this, the simulation environment must be capable of collecting and processing real network data as well as extracting conclusions in real time.

Structure variability may be examined in either at the theoretical or the implementation level. In the first case, where orientation is towards the modeling methodology, numerous researchers have discussed structure variability, mostly based on the DEVS methodology (Zeigler 1989) (Zeigler 1997). If, however, changes occur during experimentation, as in the case of real-time experiments, a domain-oriented approach is required, emphasizing on these specific changes. The domain-oriented type of approach is discussed in this paper.

To conclude about any changes, specific measures of both systems are put under monitoring. The variables used to obtain the corresponding values are referred as monitoring variables. Appropriate measures of performance are used as monitoring variables. Auditing examines variable values corresponding to the same time points (i.e. the current network state and simulation predictions for this
time point) and concludes for the evolution of the system and the model.

Assuming that the system structure is changed, monitoring data are cross-examined to detect the components affected. Auditing detects structural reformations that have occurred. Remodeling is thus invoked, which rearranges the model composition tree. When modifications are accomplished, the resulting model is once more subjected to experimentation, starting from the current real time point. Auditing is initiated after monitoring is performed for the duration of predetermined monitoring intervals.

Setting up the real time experiment imposes the following steps to be performed prior to experimentation:

1. Determining conditions leading to remodeling
2. Determining the monitoring variables
3. Expressing conditions through monitoring variables
4. Determining the degree in which each condition contributes to causing remodeling
5. Forming the auditing algorithm
6. Constructing the auditing tree

Step 1 involves determining the conditions that cause remodeling, since it is not imperative that all structural changes and all operation parameter modifications are handled through remodeling. The selection of monitoring variables supports the transformation of remodeling conditions into well-formed expressions, which is performed in step 3. The algorithm, determining how the corresponding variable values are compared to decide whether remodeling should be invoked, is referred as auditing algorithm. This algorithm also has to consider whether fulfilling each condition is essential for causing remodeling. Finally, the auditing tree is built when comparing monitoring variable values.

This structure is formed and accessed to decide whether remodeling should be performed, each time monitoring is completed. After auditing, end nodes are removed and then are disposed.

The auditing tree is depicted in figure 1 and is structured according to the following:

1. The auditing tree implements the auditing algorithm
2. The tree is divided into two subtrees and includes two respective types of end nodes, AND and OR.
3. Each end node corresponds to a single condition that may cause remodeling and is expressed through the appropriate monitoring variables.
4. End nodes are created and inserted in the appropriate subtree when expressions are evaluated for the system and the model.
5. All end nodes are positioned in the same layer and have the following structure:
   (condition, expression_sim, expression_real, deviation_params, [weight])

The weight field is used only for nodes of type OR to denote how much this condition contributes to the decision for remodeling.

The specification of the FRTS experiment includes the complete description of the conditions under which remodeling is caused (steps 1-5), which are ultimately embedded in the auditing algorithm. We describe the realization of the above steps in the following example for the network domain.

4 SIMULATION EXAMPLE

A simplified network $N$ consists of $n_1, n_2, ..., n_k$ processing nodes. The communication element is formed as sequence of protocols $pr_1, pr_2, ..., pr_n$, starting from the lowest (MAC) layer. Since nodes are identical in communication aspects, network $N = \{n_i, 1 \leq i \leq k\}$ and $n_i = (pr_1, pr_2, ..., pr_n)$. The widely used TCP/IP stack along with an Ethernet protocol could thus be represented as (10BaseT, IP, TCP), if the LLC layer (802.2) is not supported. It is thus possible to build stacks representing the acceptable protocol combinations up to the highest supported layer. Network nodes transmit and receive data originating from applications. Applications can be viewed as sockets operating above the transport layer. Each node $n_i$ is thus linked to a set of applications $A_i = \{a_{i1}, a_{i2}, ..., a_{ix}\}$, where $x_i$ is the number of applications of $n_i$. Composition of a simple network is given in figure 2, which includes two types of links: composition and in_out, indicating the network components and the coupling relation between them, respectively.

Let $Snode = \{n_i, 1 \leq i \leq s\}$ be the set of nodes in the simulation model, $|Snode|$ the cardinality of $Snode$ and $Rnode = \{n_i, 1 \leq i \leq r\}$ the corresponding set in a real network. We define the set of critical applications operating on all model nodes as $Sappl$, where an

![Figure 1: Auditing tree](image-url)
application is considered as critical when the number of data units (packets) or bits transmitted exceeds a predetermined threshold. Then \( \text{Sappl} = \{a_{ij}: \text{criticalS}(a_{ij}), 1 \leq i \leq s, j \geq 1\} \) and \( \text{Rappl} = \{a_{ij}: \text{criticalR}(a_{ij}), 1 \leq i \leq r, j \geq 1\} \). The notation \( a_{ij}(S|R).\text{property\_name} \) is used as a reference of any property of application \( a_{ij} \) in the model or the network, respectively, e.g. \( a_{ij}(S).\text{bits} \).

A simplified approach to the conditions leading to remodeling is presented in table 1, along with the reformation type caused. Whether a condition is of type AND/OR is also depicted. The network monitoring variables used to describe these conditions are presented in table 2. Remodeling conditions are thus expressed as depicted in the last column of table 1. The auditing algorithm is formed on the basis of columns Type and Description.

A simplified view of the auditing tree that is formed when monitoring is completed is provided in figure 3. In this experiment, there is only one common application in the network and the model. Condition 5 is actually distinguished into two separate conditions.

### Table 1: Remodeling conditions

<table>
<thead>
<tr>
<th>Condition</th>
<th>Classification</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Node activation</td>
<td>Structure reformation</td>
<td>AND</td>
<td>( \exists n \in \text{Rappl}: n \notin \text{Sappl} )</td>
</tr>
<tr>
<td>2 Node deactivation</td>
<td>Structure reformation</td>
<td>AND</td>
<td>( \exists n \in \text{Sappl}: n \notin \text{Rappl} )</td>
</tr>
<tr>
<td>3 Application initiation</td>
<td>Structure reformation</td>
<td>OR</td>
<td>( \exists a \in \text{Rappl}: a \notin \text{Sappl} )</td>
</tr>
<tr>
<td>4 Application termination</td>
<td>Structure reformation</td>
<td>OR</td>
<td>( \exists a \in \text{Sappl}: a \notin \text{Rappl} )</td>
</tr>
<tr>
<td>5 Critical modification of application load</td>
<td>Operation parameter reformation</td>
<td>AND</td>
<td>( \exists a \in \text{Sappl}: a \in \text{Rappl} ) AND (deviates(a(S).bits, a(R).bits) OR deviates(a(S).packets, a(R).packets))</td>
</tr>
</tbody>
</table>

### Table 2: Monitoring variables

<table>
<thead>
<tr>
<th>Representation</th>
<th>Monitoring Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>(S</td>
<td>R)node</td>
</tr>
<tr>
<td>(S</td>
<td>R)appl</td>
</tr>
<tr>
<td>(S</td>
<td>R).packets</td>
</tr>
<tr>
<td>(S</td>
<td>R).bits</td>
</tr>
<tr>
<td>(S</td>
<td>R).delay</td>
</tr>
<tr>
<td>thrput(S</td>
<td>R).packets</td>
</tr>
<tr>
<td>thrput(S</td>
<td>R).bits</td>
</tr>
</tbody>
</table>

Following the above six steps for setting up the FRTS experiment and also using the auditing tree concept, we ensure that all conditions causing remodeling are evaluated in a systematic way prior to the initiation of remodeling. It is thus possible to detect the potential deviations between the model and the system and to suggest the appropriate actions for remodeling. Even not discussed in this paper, indicating and executing the appropriate action for each reformation that may have occurred is also an issue of increased significance.

**Figure 2:** Network composition in terms of processing node and application components

**Table 1: Remodeling conditions**
REFERENCES


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


