EMPLOYING A REAL-TIME SYSTEM SPECIFICATION FOR THE DEVELOPMENT OF FRTS SYSTEMS

Dimosthenis Anagnostopoulos
Dept. of Geography
Harokopion University of Athens
70, El. Venizelou Str
17671 Athens, Greece
dimosthe@hua.gr

Mara Nikolaidou
Dept of Informatics and Telecommunications
University of Athens
Panepistimiopolis
15771, Athens, Greece
mara@di.uoa.gr

Vassilis Dalakas
Dept of Informatics and Telecommunications
University of Athens
Panepistimiopolis
15771, Athens, Greece
grad0284@di.uoa.gr

Vassilios Vescoukis
Software Engineering Lab
NTUA
15771, Athens, Greece
v.vescoukis@cs.ntua.gr

KEYWORDS
Simulation Methodology, Real-Time Systems, Faster-than-Real-Time Simulation, Structured Analysis

ABSTRACT

Even though numerous applications use real-time and faster-than-real-time simulation (FRTS), there is no concrete formalism that focuses on planning and execution of the experiment. Having previously introduced a conceptual FRTS methodology, we adopt a widely used real-time system specification method and propose a consistent specification for FRTS systems. This involves the activities (processes) that have to be carried out, data and control flows transferred between processes and simulation experiment specifications. Due to the hard real-time constraints, it is also required to analytically examine timing issues concerning the execution of simulation activities. Based on the specification provided, development of FRTS systems can be realized for diverse domains, by both real-time system and simulation researchers.

1. INTRODUCTION

Real time simulation is widely used for the performance evaluation of systems behaviour in real time. Accomplishing real-time experimentation depends on the system speed and nature, which determines the allowed degree of human interaction with the system. When attempting to reach conclusions for the near future, faster-than-real-time simulation (FRTS) is used (Cleveland J. et al. 1997). In this type of simulation, advancement of simulation time occurs faster than real world time. Making models run faster is the modeller's responsibility and rather demanding, since real time systems often have hard requirements for interacting with the human operator or other agents. Relevant methodological issues and solutions are discussed by (Fishwick 1999). Despite the broad use of FRTS, there is no formal methodology describing how FRTS experiments can be planned and methodological issues are partially addressed in domain-oriented approaches.

Gaaf discusses a practical method for maintaining model validity during the simulation run (Gaafar 1999). Middelham addresses the issue of identifying what can be reasonably predicted using a customised traffic modelling application as an example (Middelham 2001). The provision of a monitoring mechanism that allows the computation of results during real-time simulation is discussed in (Valderruten et al. 1999) for a multithreaded system. Current FRTS research directions overall involve the distribution of the experiment over a network of workstations, intelligent control (Cai et al. 1996) and fault diagnosis (Norvilas et al. 2000), interactive dynamic simulation (i.e. manipulation of the simulator by the user in real-time) (Tyreus 1997) and modelling formalisms (Zeigler et al. 2000).

A conceptual faster-than-real-time simulation methodology has been introduced in (Anagnostopoulos et al. 1999), providing a framework for conducting experiments dealing with the complexity and the hard real-time requirements. The following simulation phases have been identified: modelling, experimentation and remodelling. During experimentation, both the system and the model evolve concurrently and are put under monitoring. Data depicting their consequent states are obtained and stored after predetermined, real-time intervals of equal length, called auditing intervals. In the case where the model state deviates from the corresponding system state, remodelling is invoked. This may occur due to system modifications, involve its input data, operation parameters and structure (Anagnostopoulos et al. 1999). Modelling issues and formalisms for structure modifications have been thoroughly studied either at the methodological level,
mostly by (Barros 1997) and (Zeigler and Praehofer 1997), or for domain-oriented approaches, such as computer networks (Anagnostopoulos and Nikolaidou 2001). To deal with system modifications, remodelling adapts the model to the current system state. This should be accomplished without terminating the real time experiment, that is, without performing recompilation. When model modifications are completed, experimentation resumes.

Remodelling can also be invoked when deviations (expressed through appropriate statistical measures) are indicated between the system and the model due to the stochastic nature of simulation, even when system parameters/components have not been modified. Finally, in case simulation results (predictions for the near future) are considered to be valid, an additional phase, called plan scheduling, is invoked to take advantage of them (Anagnostopoulos et al. 1999). Experimentation phase thus comprises monitoring, that is, obtaining and storing system and the model data during the auditing interval, and auditing, that is, examining a) if the system has been modified during the last auditing interval (system reformations), b) if the model no longer provides a valid representation of the system (deviations) and, c) if predictions should be used in plan scheduling. Evidently, if conditions (a) or (b) are fulfilled, remodelling is invoked without examining condition (c).

Due to process and data orientation in FRTS description, we use a structured-analysis specification method and diagramming techniques, which are widely adopted in real-time system description (Goldsmith 1993). We do not focus on application-oriented approaches in order to provide a generic specification. In section 2, we present the real-time systems specification method employed. In section 3, we discuss the specification of FRTS, following the above specification method, which extends to the critical (in terms of real-time constraints) simulation activities, that is, monitoring, auditing and remodelling. We also emphasise the description of the experiment states and state transition conditions along with timing issues.

2. REAL-TIME SYSTEM SPECIFICATION

We provide an overview of the major concepts and features of the real-time system, structured analysis specification method and then describe the corresponding FRTS environment as an aggregate system, consisting of processes and data. In System analysis, an essential model is an implementation-free documentation of the system consisting of the following (Goldsmith 1993):

- Statement of purpose
- Environmental model
  - Context diagram

![Diagram of System States and Model States](image-url)
The environmental model shows the entire system from the outside point of view. It consists of a context diagram, which documents the boundaries among a system and its environment and an event list (there is no relationship to discrete simulation events), which is a table holding all the external stimuli and the system reactions (Hatley and Pirbahai 1987) (Ward and Mellon 1985). Environmental model functionality is given in detail through diagramming techniques. Structured-analysis diagramming techniques have been extended to support the real-time functionality.

| experiment specifications = | monitoring variables specification | + | auditing interval + state interval |
| + | prediction interval + model initialisation parameters |
| monitoring variables specification = | monitoring variable specification = |
| + | state monitoring indication |
| monitoring variable specification = | name + deviation range |
| state monitoring indication = | elemental |
| deviation range = | elemental |
| /* determines if used in state monitoring */ |
| state monitoring indication = | [on | off] |
| auditing interval = | elemental |
| state interval = | elemental |
| prediction interval = | elemental |
| model initialisation parameters = | model initialisation parameter = |
| model initialisation parameter = | model class + model parameters |
| model class = | elemental |
| model parameters = | (model parameters) |
| model parameter = | model parameter = |
| model parameter = | [model initialisation parameter | value ] |
| value = | elemental |
| raw model data = | elemental |
| /* model output data set */ |
| raw system data = | elemental |
| /* system output data set */ |
| new model = | elemental |
| /* new composite model */ |

Table 1: External data flow specifications

As introduced by Yourdon (Yourdon 1989) and DeMarco (DeMarco 1978), data flow diagrams (DFD) model the basic system functionality in terms of the widely used constructs: data transformations (processes), data flows and event flows, data stores and terminators. Control processes were introduced for controlling the invocation of data transformations, as discussed by Ward and Mellon (Hatley and Pirbahai 1987) (Ward and Mellon 1985). This is accomplished by event flows, which are depicted as dotted lines. There are two ways for processes invocation: E/D denotes that a process is enabled and then disabled by the controlling process, while T denotes that a process is triggered and, when completed, it may return a corresponding signal. In the latter case, process execution duration is not determined by the controlling process. State transition diagrams (STD) show the dynamics of a system, i.e. how a system behaves over time and what causes the system to change its behaviour, in terms of states, transitions, conditions and actions. Finally, the supporting project dictionary (DeMarco 1978) (Yourdon 1989) holds the specification of all diagram components. In a widely-used notation for the specification of data flows (Goldsmith 1993), data composition is denoted by ‘+', multiple data elements by ‘{ }', choice of data elements by [' | '] and optional data elements by ‘(  )'. The term ‘elemental' denotes that data cannot be broken down any further. We do not discuss E-R specification, as it is depended on each specific application domain.

3. FRTS SYSTEM SPECIFICATION

3.1. Environmental Model

Context diagram

We provide a consistent specification of FRTS using the above structured-analysis method and diagramming techniques. In the context diagram depicted in figure 2, the FRTS system is represented as a data transformation process (i.e. simulation activity). System and model terminators represent the actual system and the model and send raw system data and raw model data, respectively, to the FRTS system. When the model deviates from the current system state, remodelling is invoked and a new mode is constructed to replace the old one. User terminator has the ability to start/stop the experiment. Experiment specifications consist of monitoring variable, auditing interval, state interval, prediction interval and model initialisation specifications, which are presented in table 1.

Event list

The event list includes the start/stop experiment and change parameter event flows (table 2).
### Event Response

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>change parameters</td>
</tr>
<tr>
<td>2.</td>
<td>start experiment</td>
</tr>
<tr>
<td>3.</td>
<td>stop experiment</td>
</tr>
</tbody>
</table>

**Table 2: Event list**

#### 3.2. Behavioural model

**Data flow diagrams**

At the main FRTS DFD depicted in figure 3, *Control Process* is responsible for controlling the experiment by examining event flows, which are set to either true or false, and invoking the corresponding activities. *Process Control* enables and disables *Get User Parameters, Initialise Model, Monitor Model, Monitor System, Audit and Remodel*. *Model* is initialised and executed according to experiment specifications. Monitoring of the model and the system is performed concurrently, and both types of raw data are collected. Data is then processed and respectively stored into *Model Data* and *System Data* stores. Monitoring is executed for a time period equal to auditing interval, such as \([t_{n-1}, t_n]\) in figure 1. Model execution is then paused and *Audit* is invoked. *Monitor Model* is disabled during *Audit* and *Remodel*. *System Monitoring*, though, is never terminated, so that system changes can always be perceived. System and model monitoring variables are calculated and then stored. *Audit* determines if the model still provides a valid representation of the system. If invalid, *Remodel* is invoked. The corresponding data flow specifications (DFD 0) are presented in table 3.

**Figure 3: FRTS activities, data and event flows (DFD 0)**

Assume that auditing is invoked at \(t_{n-1}\) and that we want to reach predictions for \(t_y\) within auditing interval \([t_{n-1}, t_y]\) (figure 1). Then, prediction interval = \(t_y - t_{n-1}\). Evidently, prediction interval > auditing interval. As discussed in [10], we often choose a value \(p\) so that:

**Table 3: FRTS data flow specifications**

\[
\text{prediction interval} = p \cdot \text{auditing interval}, \quad p \in \mathbb{N}^*
\]

Model validation is performed through comparing the corresponding system and model states, described via monitoring variables, which are commonly defined for the model and the system.
We describe monitoring variable specifications, which is part of experiment specifications forwarded to the FRTS system, when a comparison is performed with a single sample from the system and the model. Monitoring variable specifications, described in table 2, includes its name, a deviation range, which is determined for each specific experiment, and a state monitoring indication. State monitoring is described in the remainder of the paper. Deviation range (dr) supports a basic-inspection comparison between system and model data, that is, no deviation is encountered for $MV_i$ when:

$$MV_i.s \in [MV_i.r \cdot (1-dr), MV_i.r \cdot (1+dr)]$$

The structure of monitoring variables - enabling realizing the comparison - is constructed combining $MV_i.r$ (system value) and $MV_i.s$ (model value) with the respective $MV$ specification (table 2), where name, state monitoring indication (smi) and deviation range (dr) are user-specified. The structure of $MV_i$ is the following:

$$MV_i = (\text{name}, r, s, dr, smi)$$

To give an example, suppose we implement a FRTS experiment for a traffic light system. We consider two monitoring variables for testing model validity against real observation from the traffic light system: the number of cars in the queue (i.e. waiting for the green light), which also serves as state monitoring variable, and the number of cars crossing the junction within a traffic light cycle. We use the traffic light cycle as auditing interval and impose a deviation range equal to 30%. The structure of monitoring variables formed when executing auditing is the following:

$$MV_1: (\text{‘incar_no’}, \{\text{system value}\}, \{\text{model value}\}, 0.3, \text{on})$$

$$MV_2: (\text{‘outcar_no’}, \{\text{system value}\}, \{\text{model value}\}, 0.3, \text{off})$$

In the following, we discuss the main FRTS activities, that is, Monitor Model, Monitor System, Audit and Remodel.

![Figure 4: Monitor System](image)

System monitoring (figure 4) collects raw system data and calculates system monitoring variables. When Process Control invokes Monitor System, monitoring of the real-time clock is also triggered. System monitoring never terminates, but is re-initiated whenever auditing is invoked (e.g. at $t_{n-1}$, $t_n$). Assume that function realtime() returns the current real-time point and that monitoring is re-initiated at $t_n$. Monitor Real Time Clock continues while:

$$\text{realtime}() - t_n < \text{auditing interval} - T_{Proc}$$

Then, Control System Monitoring activates Determine Model State, which processes raw system data and calculates the values of system monitoring variables consuming $T_{Proc}$, which is considered as a constant throughout the experiment.

In this way, handling deviations between the model and the system is performed after the auditing interval. This cannot be effective when critical, such as structural, modifications have occurred, where remodelling must be performed to restore consistency between the model and the system. In state monitoring, a limited set of system data is collected within a smaller interval (state interval) than the auditing interval, that is, while:

$$\text{realtime}() - t_{\text{state\_init}} < \text{state interval}$$

where $t_{\text{state\_init}}$ is initially set to $t_n$. Then, state monitoring completed signal is emitted, $t_{\text{state\_init}}$ is set to realtime() and state monitoring is re-initiated. State monitoring is efficient, as less computationally intensive, due to the amount of state monitoring data and the rather trivial comparisons it requires, so that $T_{Proc} = 0$ for calculating state monitoring variables. State monitoring
variables are examined within Audit (state auditing activity) with no time overhead, as model execution is not paused. State monitoring interval (or state interval) is often chosen so that:

\[
\text{auditing interval} = g \cdot \text{state interval}, \ g \in \mathbb{N}^*
\]

Model monitoring (figure 5) is executed while the model is running, that is, while predictions are reached for the predetermined prediction interval within the given time frame (i.e. auditing interval). The predicted time point at any real time point is denoted by the simulation clock (figure 1). Assume that auditing is invoked at point \( t_n \). Monitoring is initiated at \( t_x \), after auditing and remodelling are completed, and simulation clock is set to the starting point of the current auditing interval, thus:

\[
t_x - t_n = T_{\text{Audit}} + T_{\text{Remodel}}
\]

Monitor Simulation Clock duration is equal to the model execution time \( T_{\text{Exec}} \):

\[
T_{\text{Exec}} = m \cdot (\text{auditing interval} - T_{\text{Audit}} - T_{\text{Remodel}} + T_{\text{Proc}})
\]

Then, Control Model Monitoring activates Determine Model State, which sends request data to the model (i.e. invokes the corresponding method of the model object) and obtains raw model data. Data is then processed to calculate monitoring variables, consuming \( T_{\text{Proc}} \). Then, model monitoring terminates.

Audit (figure 6) is the key experimentation activity determining model validity through comparing the corresponding system and model monitoring variables. Auditing is activated either after a state interval or an auditing interval. Two distinct cases are thus considered:

1. State auditing: Check System inspects the current system state to determine if remodelling has occurred. In this case, the model no longer provides a valid representation and the relevant remodelling indication is produced. Auditing Control then notifies Process Control, so that remodelling is invoked. Only variables designated as state monitoring variables (table 2) are used in this process. As each variable may potentially cause remodelling, a state auditing algorithm invokes
remodelling without exhaustively examining all remodelling conditions, so that model modification is done with minimum time overhead.

2. Standard auditing: System modifications, involving its input data, operation parameters and structure, as well as deviations between the system and the model occurring due to the stochastic nature of simulation, are examined in order to determine model validity. If remodelling is required, remodelling indication is produced. Compare States examines the corresponding monitoring variables, and informs Auditing Control. All monitoring variables are used in this process. Considering that all monitoring variable comparisons may potentially cause remodelling, an algorithm determining whether remodelling should be invoked examines all potential conditions before invoking remodelling, so that a complete indication is formed.

Remodel (figure 7) is invoked to handle system reformations and deviations between the system and the model. Especially for structural changes, accomplishing remodelling in real time is possible when model components are preconstructed and reside in model libraries, so that recompilation can be avoided (Anagnostopoulos and Nikolaidou 2001).

Determine Updated Model is the activity generating new model specification after examining current state (indicated by the system monitoring variables), remodelling indication and object specifications, which describe all available model classes residing in the library. The generated specification includes the new model class and model parameters, which involve its properties and submodels. Build Model activity constructs and initialises the new model based on these parameters. When this is accomplished, remodelling completed signal is emitted back to FRTS Control Process. Remodel data flow specifications are presented in table 4.

```
new model specification = new model class +
  new model parameters
new model class = elemental
/* object class, member of object hierarchy */
new model parameter = (new model specification | value)
  value = elemental
object specifications = (object specification)
object specification = model class +
  model description
model class = elemental
model description = elemental
/* related to the model library organisation */
model objects = (model1)
model = elemental
/* object class, member of object hierarchy */
```

Table 4: Remodel data flow specifications

State transition diagrams
States and state-transition conditions are a substantial part of a discrete system description and are depicted in the STD of figure 8. There is a direct correspondence between an STD and a control process – whenever a control process exists on a DFD there is corresponding STD, and vice versa (Goldsmith 1993). Conditions in the STD correspond to incoming event flows to the process and actions correspond to outgoing event flows, as depicted in figure 3 and figure 8. FRTS states involve both the activities that must be accomplished in real time, that is, Model Initialising, Model Monitoring, System Monitoring, Audit and Remodel as well as others not executed in real time (figure 8). As all activities consume time, it is apparent that there is a one-to-one correspondence between STD states and FRTS processes, i.e. each simulation activity is a separate state.

3.3. Project dictionary
Textual specifications of data flows were included in the description of the corresponding simulation activities for readability and clarification purposes.
4. CONCLUSIONS

We expressed the functionality of FRTS using structured-analysis, real-time system specification techniques to provide an analytical, formal description of this specific type of simulation. A conceptual FRTS methodology was used as a basis, as there is no formal method focusing on planning and execution such experiments. An analytical specification of simulation activities executed in real time and of data flows transferred among system components was provided and respective timing issues were addressed. As activities and data flows may be common in diverse FRTS implementations, we supported establishing a common basis for FRTS system development.

REFERENCES


Yourdon E., Modern Structured Analysis, Prentice Hall (Yourdon Press), Hemel Hempstead, 1989
