

## Specification of Faster Than Real Time Simulation (FRTS) using the Unified Modeling Language UML

Sarantos Psycharis<sup>1</sup>, Stratis Konakas<sup>2</sup>, Kostas Aggelakos<sup>3</sup>, Tina Galousi<sup>2</sup> and Georgios Kapogiannis<sup>4</sup>

<sup>1</sup> University of the Aegean, Department of Primary Education, Leoforos Dimokratias 1, 85100, Rhodes, Greece.

<sup>2</sup> System Engineer

<sup>3</sup> University of the Ionian, Department of History

<sup>4</sup> Msc Student, University of Athens

In this paper we present a general framework for the implementation of Faster-Than-Real-time Simulation (FRTS) using the Unified Modelling Language (UML). In FRTS type of simulation the simulation time advances faster than the real time and the model attempts to make conclusions of the real system in the near future. In the paper we describe the different agents and their actions as well as the different subsystems participating in the processes using the corresponding UML diagrams. We also present the modeling, experimentation and remodelling phases as well as the monitoring of the system and the model.

**Keywords** Faster-Than-Real-time Simulation; Unified Modelling Language; Artificial Intelligence; Semantics

### 1. Introduction

Faster-Than-Real-time Simulation is used when we want to reach predictions for the time development of a system in the near future (Anagnostopoulos et.al 1999 & Cleveland 1997).

In this type of simulation, the advancement of simulation time occurs faster than real world time and the system interacts with agents (Fishwick, P., & Lee, K. 1999). To achieve concrete specifications of the model, UML is used to specify the different issues of FRTS. UML diagrams are used to describe distinct entities and their roles, overall logic of FRTS system, synchronized communication, remodeling, experimentation, monitoring, auditing and data specification. Faster-than-real-time simulation can be used for performance evaluation of systems behaviour in real time, providing significant capabilities for studying systems with a time-varying behaviour. FRTS enables model validation through comparing simulation results with the corresponding system observations. Using models in the FRTS type presents difficulties for their implementation since times runs faster than the real time. In this work we present the framework for a general methodology to describe the design of FRTS systems using the UML language. For this purpose we analyze the different aspects of a system and its interactions, while for the description of the internal operation we use the static and the dynamic structure of the different subsystems. Modelling the system using UML diagrams (case, state, class, activity, sequence diagrams) enables the annotation of UML models with properties that are related to modelling of time. Each UML diagram is designed to let developers and customers view a software system from a different perspective and in varying degrees of abstraction. UML diagrams commonly created in visual modelling tools include diagrams and have modular structure. In addition, certain stereotypes with tags may be used to UML models in order to extend its semantics (e.g. the duration of an event).

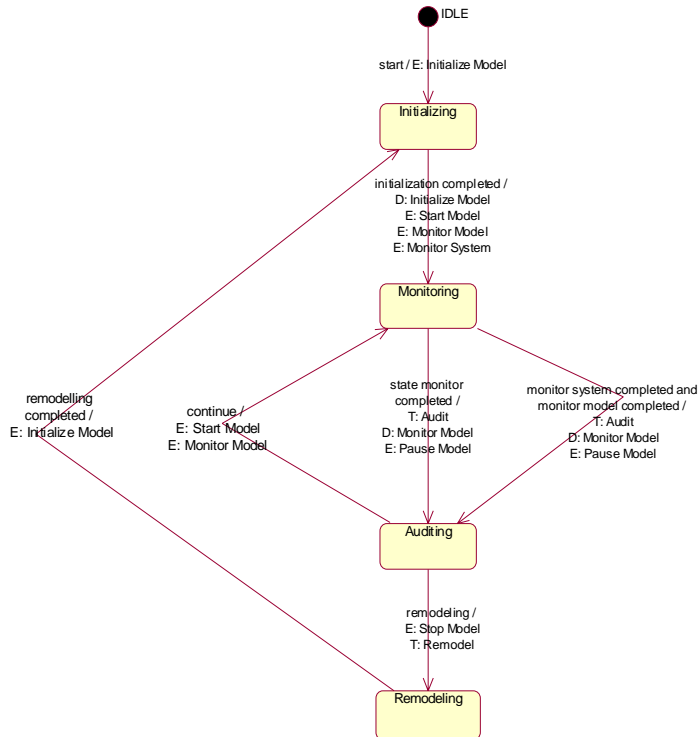
### 2. FRTS Specification with UML

The specification described below determines the structure and the operation of a FRTS system. Initially we present the interaction of the system with the external factors and then its internal structure. For

the description of the internal operation we need two kinds of information: the static one for the system and its subsystems as well as the dynamic behaviour for its life cycle. Generally, a FRTS system interacts with three entities: the user, the system under consideration and the model which simulates the system. Every entity has the following attitudes.

- User
  - Start/Stop: It starts and stops the operation of the system
  - Set Experiment Specifications: It defines the specifications of the experiment
- System
  - Send Raw System Data: It sends data for the system
- Model
  - Start/Stop/Pause: FRTS initiates and stops the model
  - New Model: FRTS creates a new model
  - Set Initialization Parameters: FRTS sets the initial conditions for the model
  - Send Raw Model Data: It sends data for the model

The internal structure of the FRTS has been modeled with the use of the logical view. The main diagram is the view class diagram where the different parts are depicted as classes or packages. The main subsystem is the ProcessControlClass which coordinates the rest of the subsystems. The input comes from the class UserClass and it coordinates the following subsystems: MonitorSystemPackage, MonitorModelPackage, AuditPackage, RemodelPackage. In Figure 1 we present the state diagram which analyses the different states of the system as well as the different events which cause transitions between the different states. After the initialization of the system, monitoring of the system and the model is executed. After monitoring auditing is executed to verify if the model really represents the system. In that case we repeat the monitoring, otherwise remodeling is executed to debug the faults of the model.



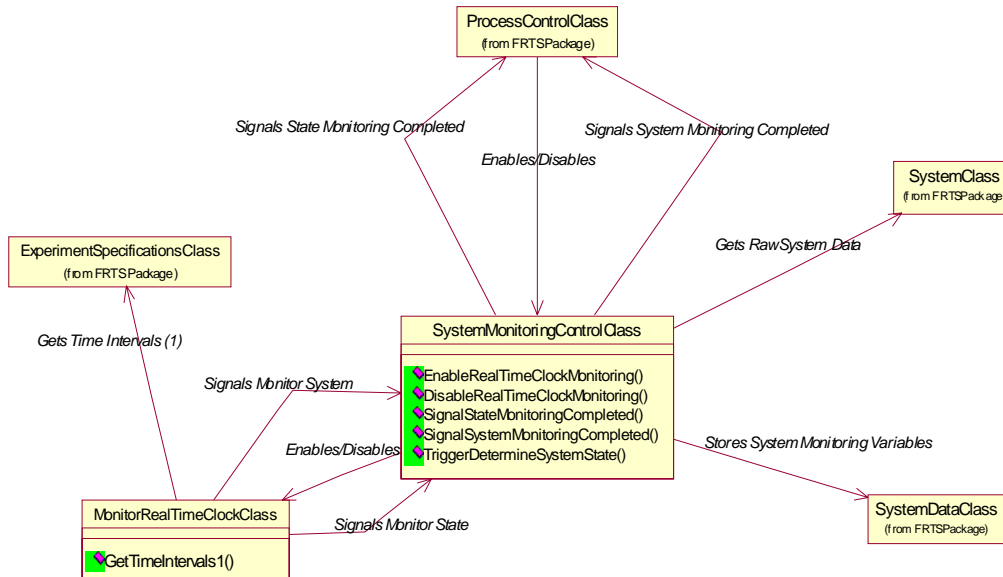
**Fig. 1:** FRTS State Diagram

We now describe in detail the different subsystems.

**System Monitoring**

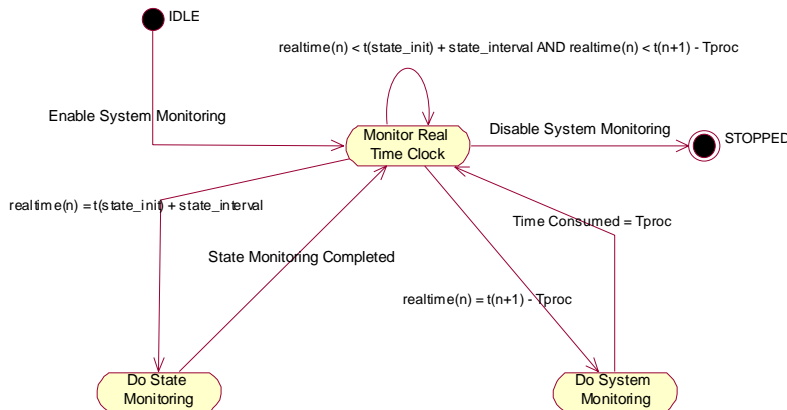
In Figure 2 the subsystem for the system monitoring is presented.

The basic class of that is the *SystemMonitoringControlClass*, which coordinates the observation of the system.



**Fig.2:** System Monitoring Package Class Diagram with tags

The flow chart of the system is depicted with the aid of the activity diagram. After starting it checks the continuously the real time until the moment when state or system monitoring is needed.



**Fig. 3:** System Monitoring Package Activity Diagram

**Model Monitoring**

Control is realized by the class *SystemMonitoringControlClass* (Fig.4)

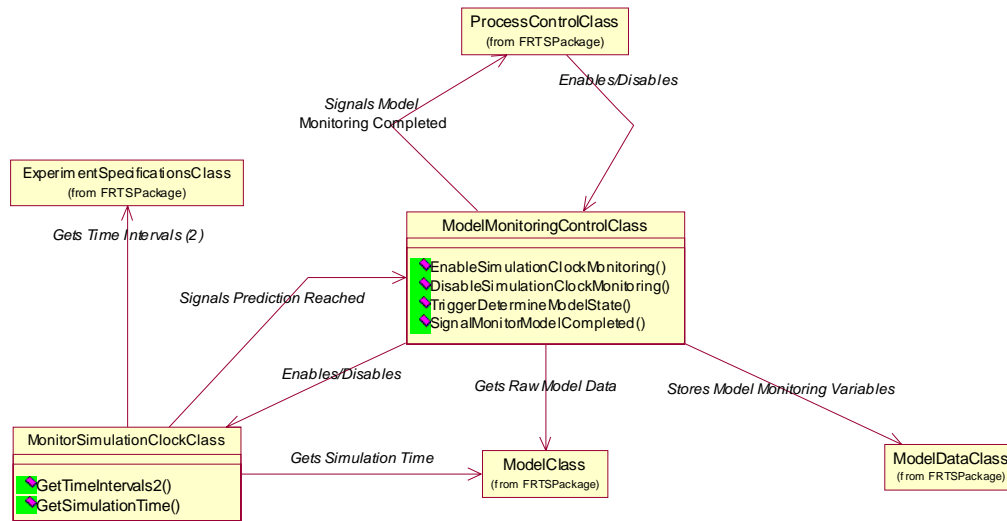


Fig. 4: Model Monitoring Package Class Diagram

### Auditing

This subsystem controls if the model has big discrepancies with the real system. The main part is the the *AuditingControlClass* which decides if state  $\eta$  system auditing will be executed which consequently calls the classes *CompareStatesClass* or *CheckSystemClass* respectively.

### Remodeling

In case auditing demonstrates big differences between the system and the model, *ProcessControlClass* activates remodeling.

## 3. Conclusions

Simulation combined with artificial intelligence could allow the computer to discover optimal solutions and actions faster than real time eliminating the human test and guess (Ho and Li N., 2000). FRTS combined with artificial intelligence techniques it is possible to replace the process of external management with internal discovery moving the learning process into a virtual environment avoiding costs and risks associated with performing experiments in engineering (e.g. robotics) manually. The computer may be faster than the real environment, thus saving time, especially if many tests can be run in parallel. FRTS can be implemented with the simulation phases: modeling, experimentation and remodeling. During experimentation, both the system and the model evolve concurrently and data representing the states are obtained and stored in auditing intervals. If the state of the model deviates from the corresponding system state we apply the remodeling to the current system state.

FRTS can be used for virtual learning environments to remove the programmer from the process of developing code for mechanical or electrical responses and this could allow the computer to discover optimization techniques and actions for machines in faster than real-time. This has as effect to eliminate the need human guess-and-test movement. Artificial intelligence combined FRTS simulation would allow the creation of such systems and can be also used in the learning process at Secondary education for Vocational Training, in Tertiary Education for the study of Autonomous robotics (an inherently multidisciplinary domain) which draws from the fields of mechanical, electrical, and computer engineering as well as computer science. In engineering the difficulty in modeling the robot is to know which measurements are crucial, and which have some leeway. For example, the upper leg of the robot, which doesn't move or change shape, could easily be modeled by a rectangular block the same size and mass,

as opposed to precisely detailing every component. Making an estimate like this could be based on FRTS and can be modelled with UML.

It can also be used in genetics to predict growth of population, for exploring successively complex military scenarios and generally for cases where predictions should be used in plan scheduling.

### References

- [1] D. Anagnostopoulos, M. Nikolaidou, & P. Georgiadis, *Transactions of the Society for Computer Simulation International*, 16/2, 70–77 (1999).
- [2] J. Cleveland, *Real Time Simulation User's Guide*. NASA, Langley Research Center: Central Scientific Computing Complex. (1997).
- [3] P. Fishwick & K. Lee, *OOPM/RT: A Multimodelling Methodology for Real-Time Simulation*. *ACM Transactions on Modelling and Computer Simulation*, 9/2, 141–170. (1999).
- [4] J. Ho, N. Li and Y. F. Li, "Development of a Physically Behaved Robot Work Cell in Virtual Reality for Task Teaching," *Robotics and Computer Integrate Manufacturing*, vol. 16, (2000), pp 91-101.