MODELLING AND SIMULATION OF AN AUTOMATIC DEBITING SYSTEM

L.O. Hertzberger, A.G. Hoekstra, J. Lagerberg
Department of Computer Science
Faculty of Mathematics, Computer Science, Physics, and Astronomy
University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam

Abstract
This paper describes the modelling and simulation of an Automatic Debiting System (ADS). A simulation environment (ADSSIM) is discussed and an example of a fictitious ADS is given which is implemented within the framework of the simulation environment.

1 INTRODUCTION

We have developed a modelling approach for Automatic Debiting Systems (ADS) on Motor Highways and realized a generic software environment to implement these models. The tool, called ADSSIM, is currently used in a project of the Dutch government to assess the technical feasibility of ADS's proposed by industry. An ADS is modelled by a top-down, hierarchical decomposition, resulting in a number of (virtual) sub-components. It is assumed that the sub-components have a well-defined state which changes on pre-computable timestamps. The ADS can then be represented as a discrete event system. ADSSIM is a discrete event simulation environment specifically tailored to implement high-level ADS models. ADSSIM is a versatile tool which not only is used to assess the technical feasibility of an ADS, but can also be applied as a computer aided design tool.

The modelling approach and the simulation environment (ADSSIM) will be presented. As a case study, an example of modelling and simulation of a fictitious ADS will be shown.

2 DESCRIPTION OF AN AUTOMATIC DEBITING SYSTEM

Due to heavy road congestion in the Netherlands, especially in the western part around the major cities (Amsterdam, Rotterdam, Den Haag) the Dutch Government intends to implement a system of Electronic Toll Collection with the goal to reduce these congestion. It is foreseen that these systems will have to become operational in the year 2001. The toll collection should not interfere with normal traffic flow. Therefore, an Automatic Debiting System (ADS) is required.

An ADS will be constructed as follows (see Fig. 1). Vehicles will have an On Board Unit (OBU) which contains an electronic purse (a smart card) and μ-wave equipment to communicate with the Road Side System (RSS). The RSS consists of a communication system, a sensor system, a registration system and a coordination system.

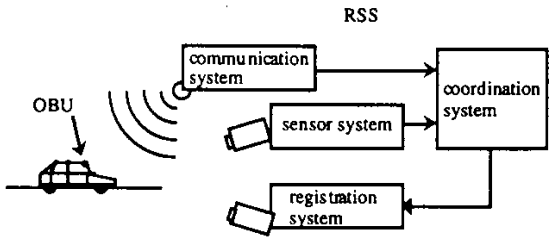

Fig. 1 Schematic drawing of an Automatic Debiting System.

When a vehicle enters the tolling zone the communication system will start an exchange of messages with the OBU, resulting, under normal operation, in a debiting of the smart card in the OBU. The communication system keeps record of the debiting of each vehicle. Furthermore, the communication system will also assess the position of the OBU with which it is communicating. All this data is sent to the coordination system. At the same time the sensor system will be measuring the positions of vehicles in the tolling zone. This data is also transferred to the coordination system. The coordination system will correlate all incoming data, and in that way is able to verify if all passing vehicles have actually paid the tolling fee. Normally this is the case and the coordination system is able to correlate all vehicles seen by the sensor system with the data coming from the communication system.

However, the ADS must also be able to handle violators. If a vehicle does not have an OBU, if it is turned off or does not contain a smart-card, or if the smart card does not contain enough money, the vehicle is considered to be a violator. In the first two cases the vehicle will not communicate with the communication system. However, the sensor system will still detect the car, and the coordination system is in that case not able to match the sensor data with communication data. In the other two cases the communication system will communicate with the OBU, but is not able to charge (enough) money from the smart card. The communication system will send this information to the coordination system, where it is correlated with data coming from the sensor system. In all four cases the coordination system concludes that the vehicle is a violator, and triggers the registration system, which takes a photograph of the license plate of the vehicle. This photograph is sent to a back office system for further processing.

So far the desired, correct operation of an ADS is described. However, in many situations errors can occur. For instance, if a vehicle is not a violator, but for some reason an error occurs in the communication system. In that situation the coordination system can only conclude that the vehicle is a violator and trigger the registration system. In the most severe case the driver will have no proof that he or she was willing to pay and the driver is unjustly forced to pay a fine. These so-called Incorrect Enforcement and Non Charging Event have to be reduced to an absolute minimum.

Another case occurs when the debiting has worked correctly, but through an error in the sensor system, or through a wrong correlation of data in the coordination system, the coordination system decides to trigger the registration system. Again, the driver is forced to pay a fine, but now a proof of toll fee payment is available by presenting the log file which is maintained on the smart card. This mistake is less severe than the previous example, but should still be very rare to avoid administrative overhead.

A last example is that of a free ride, where a violator, through an error in the sensor system or coordination system, is not identified as such and therefore is not registered. Many other possible errors in the operation of an ADS can be identified.

The requirements of an ADS, as laid down by the Ministry, result in a set of System Quality Factors (SQF) which describe the chance that a particular error in the system occurs. An example is the already mentioned Non Charging Event which, in the Dutch setting, is required to be smaller than one in a million. The key issue is to proof that an ADS can operate correctly under such strict requirements. To model and simulate the behaviour of an ADS is a method to study the cause of faulty operation. In such a way the technical feasibility of every ADS system can be evaluated.

We have developed a modelling approach to set up discrete event models of an ADS, and we have realized a simulation environment, called ADSSIM, to implement the models and run the simulations. ADSSIM is used by the Dutch government to assess the technical feasibility of ADS designs proposed by industry.

In the next chapter the approach will be discussed in more detail.

3 MODELLING AND SIMULATION OF AN ADS

The technical requirements for an ADS are of such a stringent nature that highly sophisticated methods are needed to assess to what extend the proposed solutions can be compliant with those requirements.

For this we need to model a complete ADS, apply realistic traffic input to the model, and use as much as possible knowledge which is available on ADS sub-systems. The model should be able to deal with stochastic parameters such as the probability of a non functioning OBU or dirty license plates. As the response of an ADS depends on detailed microscopic traffic configurations it is natural to use a discrete traffic model as input to the ADS simulator. Finally, the ADS model should be able to take environmental parameters into account, such as the influence of weather conditions on sensor behaviour and on traffic, or e.g. rare specular reflections of sunlight directly into a registration camera.

We model an ADS in a top-down approach. This means that we hierarchically decompose an ADS into a small number of sub-systems. The sub-systems themselves are modelled using (known) parametrizations. We assume that the state of a sub-system changes on pre-definable time-stamps, due to external inputs and/or due to changes in other sub-modules. This assumption results in a discrete event model of an ADS.

The ADS model is set up as follows. First, we apply a small number of hierarchical decompositions of the ADS and use (known) parametrizations of the remaining modules to describe their behaviour. The connections between the modules and the response of the modules to the presence of vehicles are specified, resulting in a discrete event model of an ADS.

As an example we will discuss the development of a model for a fictitious ADS (fADS). The first step in building the model of an ADS is to specify the sensitive volumes of all sensors. A sensitive volume is defined as the volume where a specific sensor (be it an antenna, or a camera) is able to detect a vehicle.

Next we perform the hierarchical decomposition. The rear and front registration system are modelled as one abstract module. We assume that the OBU wake up always succeeds, and therefore the wake up system is not included in the model. The communication system is also modelled as one abstract module, which performs the debiting and measures the position of an OBU. Finally, we assume that the detection system cannot be modelled as one single module. In real models this depends to what extend it is possible to define validated models of e.g. a track sensor. In the fADS case the sensor system is decomposed further into three sensors that measure the front position of the vehicle, which are used to build a track containing three front measurements.

It should be pointed out here that, especially in modelling of the detection system, it is possible that after the decomposition of the ADS the resulting sub-modules need no longer be physical sub-systems, but might as well be logical units which perform a certain measurement. We could say that the detection system of fADS is modelled as three virtual sensors which measure a position. The concept of logical sensors was first introduced by Henderson and Silcrat [1] and later Weller, Groen and Hertzberger refined it to virtual sensors [2]. A discussion of the virtual sensor modelling that we have developed for ADS simulations is beyond the scope of this document.

Finally we need to find a model for the remaining elements. In the fADS example the models are very simple, but they do reveal the general ideas. The virtual position sensors are characterized through a parametrization of the final measurement errors made by the virtual sensors, which are Δx and Δy respectively. The errors will normally depend on a number of other parameters, but in the example we assume that the errors are constant. In the simulation we use these errors to generate a measurement, by:

$$x_m = x + RAN,$$

where x is the real x-position of the vehicle, x_m is the measured position and *RAN* is a random number drawn from a normal distribution with zero mean and with a standard deviation Δx.

The communication module is modelled by a probability p that the communication fails, by a transaction time t needed for the communication, and by the errors Δx and Δy for the OBU localization.

Modelling a real ADS will result in more elements in the model and more complicated parametrizations of the elements. However the models will be comparable to the highly simplified fADS example.

4 ADSSIM

ADSSIM is a simulation environment for discrete event simulation. This tool is developed using Modsim, which is a generic discrete event system. ADSSIM executes ADS models. The dynamics of vehicles in the ADS is governed by the statistics of traffic flow on highways for a number of selected scenarios. The vehicles, i.e. cars, motor drivers or trucks, are traced through the ADS geometry, and events are scheduled which represent all activities of the ADS. These events represent e.g. a start-up of communication, OBU activity, a sensor activity or a registration activity. For each vehicle all ADS activity is logged. Next, after the vehicle leaves the ADS, it enters an analysis module, which generates estimates of the desired System Quality Factors. The analysis module carries out the statistical analysis of the generated data (e.g. variance estimation), and decides upon termination of the simulation.

ADSSIM, which is a joint development of CMG and the University of Amsterdam, consists of three main modules (see Fig. 2). A traffic generator, developed by the RWTH Aachen, Germany, simulates traffic moving over a segment of the road. The traffic is validated against real (Dutch) traffic. Currently the traffic generator writes trajectories of each vehicle (i.e. position of vehicles as function of time) to a file, which is read by the Framework. The Framework is based on the evaluator tool developed by CMG [3]. It contains the man-machine interface of ADSSIM, and takes care of analysis and logging of the simulation results. The Kernel is used to implement the actual ADS model and to run the simulation. The Kernel contains a number of predefined events which are used to map the discrete event model of an ADS to ADSSIM.

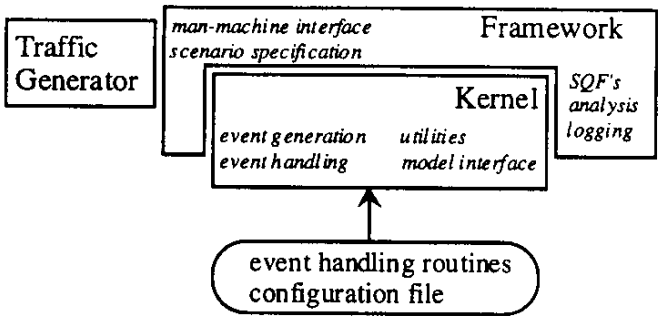

Fig. 2 ADSSIM design.

In the evaluation project it is necessary to be able to evaluate different ADS designs and consequently to implement several, different ADS models. We provided a mechanism for this. The user of ADSSIM provides a configuration file, containing the geometry of the ADS (number of lanes, positions and sensitive volumes of all the sensor systems). When a vehicle enters one of the sensitive volumes, a corresponding `in_zone` event is generated automatically by ADSSIM. (`InDetectionZone` for entering the detection zone, `InCommunicationZone` for entering the communication zone and `InRegistrationZone` for entering the registration zone). When it leaves a zone a corresponding `out_zone` event is generated by ADSSIM (`OutDetectionZone`,

OutCommunicationZone and OutRegistrationZone resp.). The user has to specify in C-code what activities should follow up as result of a vehicle entering or leaving a sensitive zone, i.e. the user has to write the event handling routines. These routines are linked with the Kernel through the model interface. They contain the parametrizations of the sub-systems and all the logic to connect sub-systems, mapped to events, with each other.

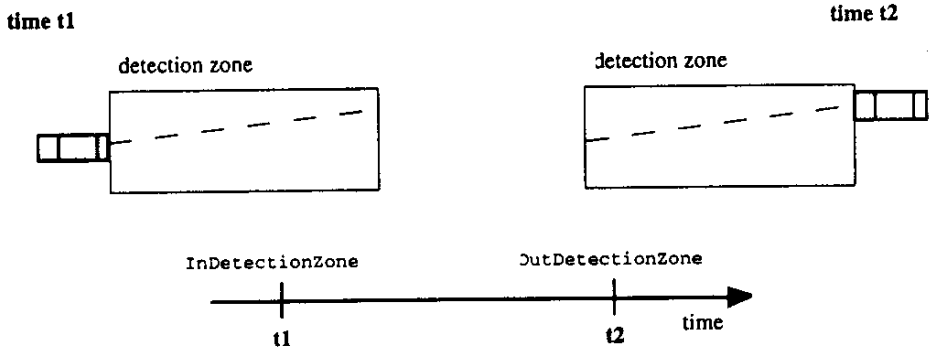

Fig. 3 Vehicle events generated when a vehicle enters and leaves the detection zone.

The simulation proceeds as follows. The framework reads trajectories of each vehicle and passes them to the kernel. At the correct time the kernel launches the vehicles in the simulator. Next, the kernel calculates at which time the vehicle enters and leaves all sensitive zones which are defined in the configuration file, and on each of these times schedules an in_zone or out_zone event (see Fig. 3). These events, which are automatically scheduled by ADSSIM, form the basis to implement the ADS model. The event handling routines for the in_ and out_zone events, which have to be provided by the user of ADSSIM, will typically schedule other events, available in ADSSIM, which are used to implement the ADS model. This could for example be a scheduling of a VehicleMeasured event when the front of the vehicle is measured at a curtain in the middle of the detection zone at time *t'*. In the example of Fig. 3, if an InDetectionZone event is scheduled on time *t1*, and the user has scheduled a VehicleMeasured event at time *t'*, ADSSIM will execute the event handling function of VehicleMeasured at the right time.

5 DETAILED DESCRIPTION OF FICTITIOUS ADS

Here we go further to illustrate the concept based on the case of a fictitious ADS (fADS) and show how this hypothetical system can be modelled. The first step in building the model of fADS is to specify the geometry and the sensitive volumes of all subsystems. Then we give the relation between the subsystems in a diagram. Finally we discuss the scheduling of the events.

5.1 Geometry of fADS

The geometry of fADS is given in Fig. 4 and specified in the configuration file. It consists of a 3-lane road with each lane 3.5 m. It contains 4 gantries at which the sensor subsystems are mounted and has a length of 15.5m. The rear registration cameras are mounted at the first gantry, the three detection curtains at the second, the communication antennas at the third and the front registration cameras at the fourth. The communication system is based on the European standard, but for the time being only a number of OBU localizations is generated during the passage of the vehicle through the communication zone. The OBU localizations are stored in an OBU track. The detection system measures at three positions (2m apart) the lateral position of the middle front of the vehicle when it passes one of the detection curtains. The front positions are used to build a detection track of the vehicle. The front registration camera is triggered when the front of the vehicle is detected by the fist curtain, the rear registration camera when the rear of the vehicle is detected by the third curtain. For all vehicles a front and a rear picture are stored. The coordination decision is taken when the vehicle leaves the detection zone, where the detection track of the vehicle is matched to an OBU track. If no

OBU track is found for the vehicle, it is registered and the pictures are stored permanently for further usage. Otherwise the pictures are removed.

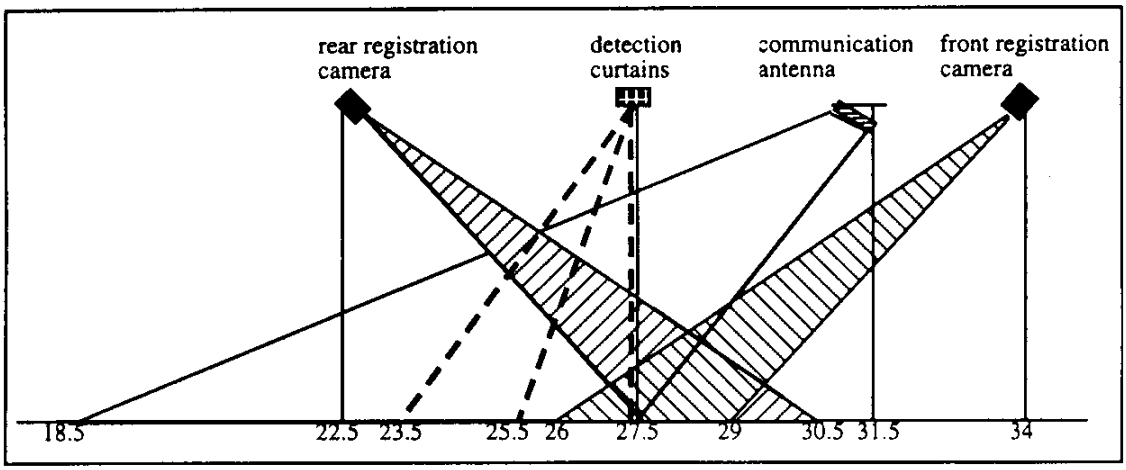

Fig. 4 Side view of fictitious ADS.

5.2 Relation of subsystems

In Fig. 5 an example of the relationship between the events of fADS is given in which the basic events `InDetectionZone` and `OutDetectionZone` are caused by the presence of vehicles. This relationship of the events doesn't give an indication of the time at which the events take place, but only by which events they are caused. When a vehicle enters the detection zone three type of events are scheduled, the events `VehicleMeasured(TRACK)` and `VehicleMeasured(FRONT)` immediately at the next clock tick of the detection system. The event `VehicleMeasured(REAR)` is scheduled when the rear of the vehicle is detected at the third curtain. In `VehicleMeasured(TRACK)` a detection track is built consisting of three front measurements at the three detection curtains. In `VehicleMeasured(FRONT)` and `VehicleMeasured(REAR)` the right registration camera is selected and the event `TakePicture` is scheduled. The pictures are temporarily stored in the detection track.

For the moment the communication is modeled by generating `OBUMeasured` events every 0.2 seconds during which the vehicle is in the communication zone (when it has an OBU). These measurements are used to build an OBU track. The event `TransactionCompleted` is scheduled in the beginning of the communication zone (when the vehicle has an OBU). This means that we assume for the moment that communication always succeeds when the vehicle has an OBU.

In the `OutDetectionZone` event handler the `CoordinationDecision` event is scheduled, which means that all the information of the vehicle is available: i.e. a detection track, front and rear picture, and an OBU track. In the `CoordinationDecision` the detection track of the vehicle is fitted to an OBU track present in the OBU track pool. If no OBU track is found, the vehicle is registered: i.e. the registration pictures of the vehicle are stored permanently. If an OBU track is found, the vehicle can be forgotten.

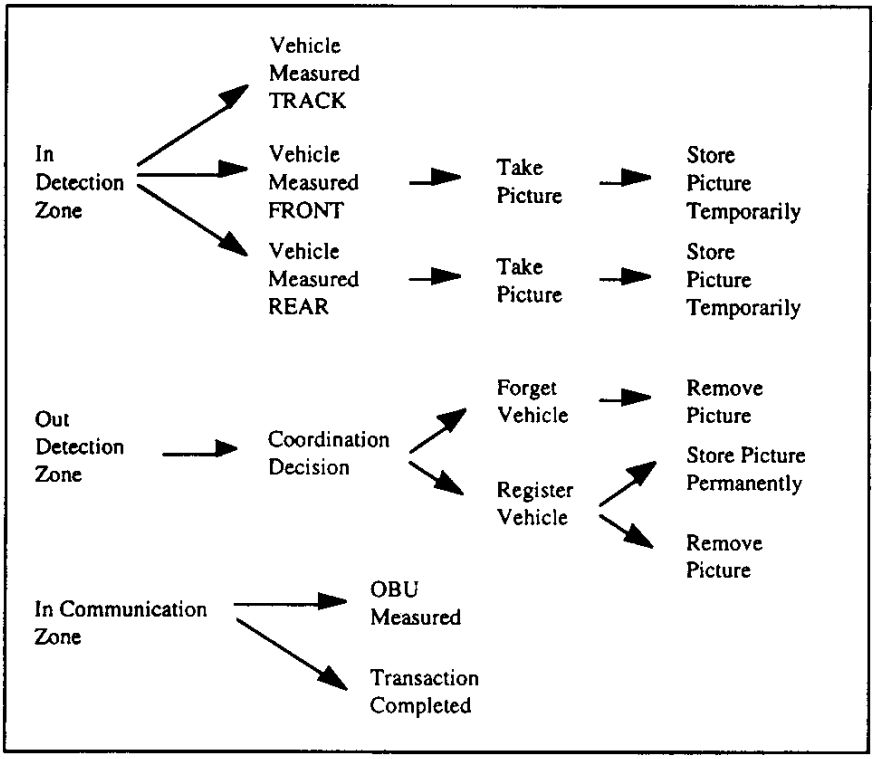

Fig. 5 Relation between subsystems mapped to events.

If a vehicle (with an OBU) enters the ADS, the following sequence of events is scheduled (see Fig. 6). First, the vehicle enters the communication zone (see Fig. 4). This means that an `InCommunicationZone` event is scheduled. Within the event handling code of the `InCommunicationZone` event a number of communication events is scheduled. The first one is the `InitRequest` event which initializes the transactions for the vehicle. The `DebitConfirm` event is scheduled when the debit cart is received by the road side system (RSS). The transaction is at that moment ended for the RSS, which is reported to the coordination with a `TransactionCompleted` event. This could be at the beginning of the communication zone when a vehicle is driving not to fast. If a vehicle is driving faster or if a dedicated smart card is exchanged by the much slower chipknip, this event is scheduled closer to the end of the communication zone. During the passage of the vehicle through the communication zone a number of `OBUMeasured` events is scheduled at irregular time intervals. Here the localizations of the OBU are measured and stored in an OBU track. Somewhere in the middle of the communication zone the `InDetectionZone` event is scheduled, in which at the three curtains a `VehicleMeasured(TRACK)` event is scheduled where the measurement of the front of the vehicle is added to a track. The `VehicleMeasured(FRONT)` event is scheduled at the first curtain which is used to trigger the front registration camera. The `VehicleMeasured(REAR)` event is scheduled when the rear of the vehicle is measured at the third curtain, which will result in a trigger of the rear registration camera. The `OutCommunicationZone` and `OutDetectionZone` event are scheduled at the end of the ADS zone. Within the event handler of the `OutDetectionZone` event a `CoordinationDecision` event is scheduled with a delay of 0.3 seconds. Within the event handler of the `CoordinationDecision` the information of the vehicle obtained during its passage through the ADS is evaluated. First of all the track of the vehicle is used to find a matching OBU track. If a track is found and the right fee is deducted from the smart card, all information concerning this vehicle (also the registration pictures) is deleted. When no OBU track is found or when no fee is deducted, the vehicle is registered. I.e. the front and the rear picture of the vehicle are stored permanently for further usage.

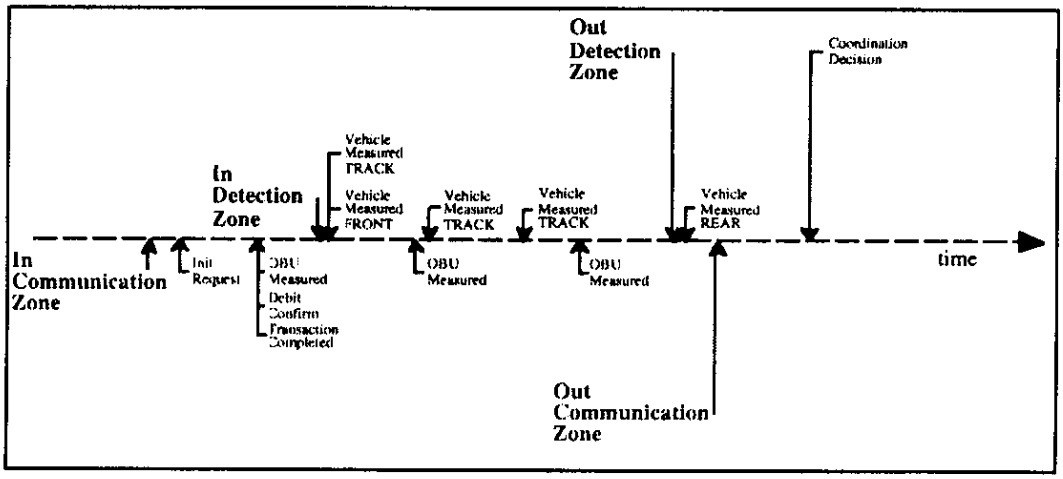

Fig. 6 Scheduling of events in course of time for one vehicle with an OBU.

6 CONCLUSION

We have developed a discrete event simulation environment for ADS simulations. This tool, called ADSSIM, is developed using Modsim, which is a generic discrete event system. ADSSIM executes ADS models. It is a generic discrete event simulation tool specifically tailored to handle ADS models. We modelled and implemented a fictitious ADS to show how an ADS model can be linked to ADSSIM. As long as fADS does not contain validated models, the simulation is not a representation of the real system. If the ADS simulation has to resemble the real ADS, validated models have to be included in the simulation and the range of validity of the models has to be known exactly [4]. Another important aspect of ADSSIM is that it can be used to design and optimize an ADS, which makes ADSSIM a computer aided design tool.

References

[1] T.C. Henderson, E.Silcrat, Logical sensor systems, *Journal of Robotics Systems*, 1, 1984, 169-193.

[2] G.A. Weller, F.C.A. Groen, L.O. Hertzberger, A sensor processing model Incorporating error detection and recovery, in *Traditional and Non-traditional Robotic Sensors*, T.C. Henderson ed., NATO ASI series Vol.63, 1990, Springer Verlag, 351-364.

[3] G-J. van Dijk, R. Poestkoke, G.R. Meijer, A methodology for performance evaluation of electronic toll collection systems, in *Proceedings of the Third Annual World Congress on Intelligent Transport Systems*, Orlando, USA, 1996, 14-18.

[4] P.L. Knepell and D.C. Arangno, *Simulation Validation, A Confidence Assessment Methodology* (IEEE Computer Society Press, 1993).