2010 13th International IEEE
Annual Conference on Intelligent Transportation Systems
Madeira Island, Portugal, September 19-22, 2010

TraSMAPI: An API Oriented Towards Multi-Agent Systems Real-Time Interaction with Multiple Traffic Simulators

Ivo J.P.M. Timóteo, Miguel R. Araújo, Rosaldo J.F. Rossetti, *Member*, *IEEE*, Eugenio C. Oliveira
Department of Informatics Engineering
Artificial Intelligence and Computer Science Lab
Faculty of Engineering, University of Porto
Rua Dr Roberto Frias S/N, 4200–465 Porto, Portugal
{ivo.timoteo, miguel.araujo, rossetti, eco}@fe.up.pt

Abstract—**TraSMAPI (Traffic Simulation Manager Application Programming Interface) is designed to provide real-time interaction with Traffic Simulators, collect relevant metrics and statistics, and offer an integrated framework to develop Multi-Agent Systems. It is presented as a tool for the simulation of dynamic control systems in road networks with special focus on Multi-Agent Systems. The abstraction over the simulator opens up the possibility of running different traffic simulators using the same API (application programming interface) allowing the comparison of results of the same application in different simulators. The proposed approach is, therefore, expected to be a key asset in supporting and enhancing engineers and practitioners to make more effective control decisions and implement more efficient management policies while analyzing and addressing traffic related problems in urban areas.**

I. Introduction

Traffic management is widely accepted as an important problem in modern society as it is responsible for keeping a steady flow of people, goods and services within cities. The quality of this flow directly affects the city's economy and general well-being of the population. We feel, however, that this problem has not been satisfactorily solved and, therefore, our vision is to create tools which allow us to study this complex problem and create solutions which would maximize the use and efficiency of a city's road network.

This paper focuses on the implementation of three such tools: an Application Programming Interface (API) which allows real-time interaction between traffic-management agents and the environment created by the simulator; a traffic-oriented Statistics module to assess the performance of the solution and to provide historical statistics which will be essential for learning agents; and a Multi-Agent System framework specifically designed to use the API efficiently which allows the exchange of messages between the agents and infinite (though a threshold can be defined) negotiation between agents before reaching a stable state. These tools provide a solid basis on which to develop new solutions aiming at Efficient Traffic Management.

Our API is designed to provide real-time interaction with Traffic Simulators because one of the major flaws we found in traffic control systems was that they were very static in their approach, taking little to none information from their domain. Real-time interaction enables the simulation of dynamic control systems which can adapt according to their environments and offer the most appropriate, if not the best, solution to the specific traffic situation that is being addressed. We built our API in an abstraction level higher than most common Microscopic Traffic Simulators so that it can be independent from the simulator being used allowing the comparison of results from different simulators using exactly the same solution. The great diversity of the information available and possible actions over the simulation environment renders possible the simulation of a large range of scenarios and real-world applications.

We are interested in Efficient Traffic Management of the road network rather than addressing specific local problems. In fact, we believe that searching for integrated solutions having the whole network as basis will lead to better results. However, it is eventually reduced to finding local solutions taking other local solutions into consideration and expecting some kind of swarm behaviour. Another interesting point is that traffic agents in real world tend to behave very selfishly. Drivers never think of the quality of the flow of the road network they are travelling in. Instead, they just aim at their own efficiency, attempting to arrive to their destination as soon as possible. Having this in mind, we opted for a Multi-Agent System approach since we believe that a set of autonomous agents is a good metaphor for such distributed and complex domains as traffic management. In fact, it has been widely used by the community working on traffic analysis as suggested by Schleiffer [1].

With these premises, it is possible to simulate a wide range of real-world applications, such as: a GPS/Satellite system which receives local information from a vehicle and spreads that information to the others; a vehicle re-routing system; a vehicle-to-vehicle wireless advice system; a group of independent autonomous agents controlling traffic lights and sharing information about their local environment to neighbours, etc. These simulation applications are all possible because we can rely on the agents' direct access to an object of the simulation, i.e. agents can be assigned to traffic lights, vehicles, lanes, etc., by having a reference to the object in the simulation. For instance, with one agent

978-1-4244-7659-6/10/$26.00 ©2010 IEEE

responsible for each vehicle, a vehicle-to-vehicle wireless advice system is reduced to the decision algorithm since everything else is already available in TraSMAPI.

There are many other application possibilities and we aimed at the creation of a tool generic enough to be able to respond to the maximum possible situations with the minimum effort by the application designer.

II. Basic Components of Traffic and Transport Analysis

Traffic and transportation systems are heterogeneous and stochastic domains in nature whose processes are rather dependent on stochastic phenomena. This means their entities can play different roles and pursue different goals while interacting with each other in a very unpredictable environment. Urban travel demand and traffic flow modelling have progressively evolved over the last forty or fifty years into an established methodology, commonly referred to as the traditional, or classical, approach called the "four-step" model [2]. This model has been devised to address travel demand, both spatial and temporal, through four basic procedures named trip generation, trip distribution, modal split and trip assignment. These procedures try to encompass every aspect of an individual's decision-making process in order to produce a representation as realistic as possible of the whole system. Traffic management systems need to integrate the supply side that represents infrastructure elements, as well as control decisions and management policies.

The first phase in this classical approach, trip generation, intends to reproduce the numbers of trips originating and ending in certain zones of the transport network. Trip distribution is the next step which consists in distributing each of the trips obtained in the first phase across various destinations. When trip distribution finishes, modal split takes place. Each of the origin-destination volumes are now "split" or distributed into various alternative modes (typically, deciding which mode of transport will be used). The order in which trip distribution and modal split happen is sometimes disregarded. Finally, in the trip assignment step, modal trips from a given origin to a given destination on a given mode, as obtained in the preceding phases, are assigned to the network's links, more precisely, routes between origins and destinations.

Ever since this approach was first proposed, integration has been an issue many systems have been trying to address. However, the complexity inherent to the application domain is considered a development bottleneck hard to overcome. Thus, the way towards integrated transport analysis tools is sometimes accomplished through a collection of independent and difficult-to-integrate models addressing specific issues of the original model in which simulation is very often an important instrument.

The relationships identified in the transportation domain brought into evidence the social aspects involved in the interaction among various autonomous entities that inhabit a common environment. In fact, contemporary transportation solutions, especially those related to traffic control and management, are achieving a high degree of autonomy and starting to interact closer with the users. Vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communications, autonomous vehicles, ad-hoc sensor networks, mobile computing and many other contemporary ubiquitous technologies are now transforming the way traffic and transport in urban areas interacts with users. The benefits promise to be mutual, as shown by some encouraging recent results. Whether such a novel interaction between demand and supply is to last on an efficient and sustainable basis is still to be confirmed. This will be possible only through the use of appropriate and advanced analysis tools, capable of covering and supporting a wide range and variety of parameters and dimensions that current simulation tools do not support.

In this context, the agent metaphor seems to represent a very propitious way to model such a domain. It eases the representation of the social interactions now playing a decisive role in the overall system performance as users explicitly interact with the system and it starts to behave rather intelligently. Such a scenario gives rise to the concept of Artificial Transportation Systems (ATS), allowing these characteristics to be modelled and tested in controlled environments where different kinds of interactions can be verified. In such systems, two main sorts of interactions are especially worth being mentioned: on the one hand, supply entities, such as control elements, act in a rather collaborative way trying to find an optimum performance set-up for the system, whereas on the other hand users usually act in a competitive way, seeking self improvement and the maximisation of their own benefits disregarding the effects their actions can have upon others' performance. In the next section we shall discuss on the concepts of agents that underlie the implementation of ATS.

III. Autonomous Agents and their Applications to Traffic and Transport

Before going deeper into the specification of our approach, some discussion on key concepts related to autonomous agents, their relationships and how applicable they are to the field of traffic and transport is necessary. Multi agent-based models become a natural metaphor to represent domains where such a large number of intelligent and autonomous entities interact with each other and with the environment. These models are being increasingly used within analysis frameworks as an effective tool to aid the understanding of complex and stochastic phenomena. Traffic and transportation systems have profited from and stimulated much research on the development of agent-based technologies. As we shall see in the following discussion, most applications of Multi-Agent Systems (MAS) to traffic

and transport end up with traffic control and management, being other forms basically relegated to a secondary spot.

The main premise in multi-agent systems is to interpret real world in terms of agents that exhibit intelligence, autonomy, and some degree of interaction with other agents and with its environment. Other characteristics of agents include reactivity, adaptability, pro-activity, and the ability to communicate and to behave socially. The basic structure of an agent features sensors through which it can gather information from the environment, and effectors through which it can act and behave according to its objectives [3]. This structure can feature both reactive and cognitive abilities, and a mixture of both, to mimic human behaviour in a wide range of applications. Steels [4] suggests that each single agent, albeit possibly having a very simple structure, can contribute to a more complex and efficient behaviour of the system as a whole. If the behaviour of such single agent can be backtracked, then this can be used to aid the understanding of the more complex behaviours at aggregate level, such as the social phenomena for instance.

To the best of our knowledge, some former attempts to apply agent-based techniques to address transportation issues date back to the 90's. For instance, Haugeneder and Steiner [5] proposed a co-operative agent-based architecture as a means to improve traffic management and control [6]. If in the beginning people from AI community benefited from the intrinsic complex and dynamic nature of transportation systems to devise and support agent theory, transportation engineers and practitioners have now started to recognise the natural ability of the multi-agent metaphor to model traffic phenomena. Owing to their characteristics and concepts, multi-agent systems have a natural aptitude to cope with a wide range of issues in contemporary traffic and transportation scenarios [1].

In this work we focus on the cognitive abilities of autonomous agents to devise a multi-agent architecture to underlie the implementation of an environment in order to analyse and test with different traffic control strategies and management policies effectively. The MAS-based approach herein proposed is conceived in a way agents can be easily instantiated and can interact both autonomously and cooperatively through a specific interaction protocol that fosters short-term strategy as well as long-term tactic control and management decisions. The potentials of such an analysis tool are enormous, which shall be discussed later on in the paper.

IV. Goals and General Architecture

As seen in previous sections, such a large domain requires several systems to interact in order to achieve a comprehensive model of reality. Researchers would profit from an easy-to-use development environment, independent of the simulator and independent of the other building blocks of the simulation (as defined earlier, namely supply and demand).

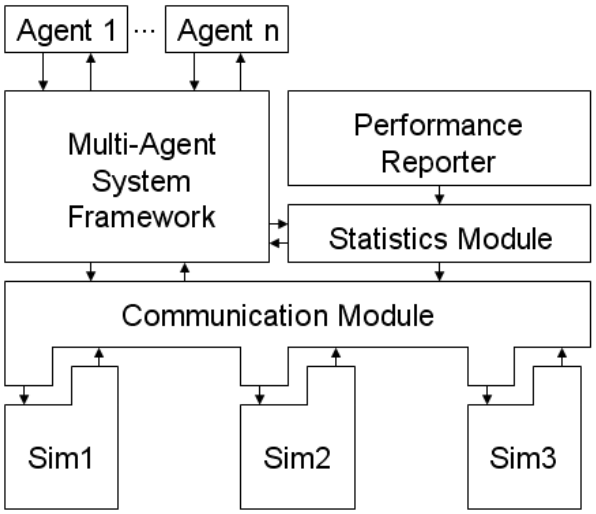

Fig. 1. General architecture of TraSMAPI showing the interactions among different modules

Researchers typically want to develop a specific component, but still retain the ability to interact with others. For instance, an application for V2V communication would require the implementation of vehicle agents (possibly leaving to the simulator the control of some vehicles), and still use the simulator's traffic light states. Another example could be a traffic light controller, which would implement a traffic light agent. This agent certainly needs to interact with vehicles, though its creator might not be interested in building a specific vehicle agent.

All this interactions are accomplished by defining agents and object controllers directly responsible for an element of the simulator environment. To the agent's creator, the simulator being used and the other agents (unless he wishes to communicate with them) are completely transparent.

Given the simulator independence, solutions can be easily tested in several simulators providing richer results. This multi-simulator testing increases the validity of the built solution, as solutions are not simulator-crafted. It is especially important when implementing intelligent systems, as they will not use simulator-specific characteristics, or simulation flaws, during their learning phase.

TraSMAPI is made of different independent modules which interact with each other to form the whole solution, as seen in Fig. 1. The decision to make the solution modular was motivated so as to make the solution prone to collaborative development and to allow the development and integration of new modules without major difficulties. This is important as we aim to provide a basic framework for the simulation of Traffic Management Systems. This way, the developers of specific applications using this solution will, most certainly, want to create new abstractions over the existing modules to further adapt to their specific problem. These new abstractions can be easily implemented by adding new modules which will then communicate with the existing ones creating a more complex and specific solution (e.g. a new module representing a GPS System working in real time

over the simulation can be easily built. It would use the Communication and Statistics modules in order to obtain the same information it would need to retrieve, as a GPS System would in the real world).

A. Architecture Overview

The general architecture of TraSMAPI is based on modules each with a well-defined function in the whole system (Fig. 1. may be useful for the reader to have a correct understanding of the current section).

The researcher is responsible for selecting the simulator to be used (given it is supported by the API) and for implementing the desired agent. TraSMAPI uses Java objects in order to hide several layers used by the agent. Ideally, the agent should be independent of the simulator chosen. This is guaranteed as far as the simulators chosen allow it, and provided that their communication interface differs and they do not implement the same set of features.

During the rest of this section, TraSMAPI hidden layers will be described.

B. Modules Specification

The application is organized in three main Modules: the Communication Module, the Statistics Module, and the Multi-Agent System Framework. These modules will now be analysed in further detail.

1) Communication Module

The communication module provides the basic API for the interaction with the simulator. The interaction is based on queries, to gain knowledge on the environment state, and orders, to change the state of the environment. The exchange of information is usually done by remote communication technologies in order to keep the modularity and the simulator running completely independent from the communication module (the exact communication technology used is dependent on the simulator over which the abstraction is being built).

In its lowest abstraction level, the Communication Module is based in a set of methods responsible for simple message exchanging. These methods are reused if distinct simulators use the same remote communication technology. Immediately above this level there are the methods responsible for the implementation of the communication protocol (i.e. message order, content and timing so as to query the simulator) which is dependent on the simulator. With this set of methods it is possible to build all the higher level queries and orders.

The most commonly used and widely implemented queries allow us to know information about the vehicle position, the vehicle speed, the vehicle route, a lane's maximum allowed speed, a specific traffic light state, the lanes leading to a specific traffic light, the amount of vehicles passing over an induction loop, and their speed, and so forth.

The orders allow us to change things such as the state of the traffic light, the maximum speed of a lane, the route of a vehicle, and the speed of a vehicle.

Naturally, more complex queries can be built from these simple requests (e.g. the number of vehicles waiting in a traffic light lane – combining the position and speed of a vehicle and comparing the lane to one of the lanes leading to the traffic light.)

2) Statistics Module

The Statistics Module is a passive module that does not interfere directly with the simulation. However, it is very important to develop real-world applications. The Statistics Module records all the information regarding the simulation environment of the different simulation steps using the Communication Module to access the Traffic Simulator for the data.

The Statistics Module stores the information of every simulation step and implements a simple query interface thus acting as a historical archive. However, the Statistics Module was created and is mainly used to assess performance at the end of the simulation and to provide past-dependent statistics useful to intelligent agents. The actual statistics that a developer would like to retrieve are dependent on the simulation he intends to run and are hard to predict. With that in mind we assumed that the developer of the application would implement its own methods for statistics retrieval.

Taking into consideration performance issues and assessment requirements of the traffic light agent, these metrics were considered useful and were implemented: traffic light throughput and vehicles average waiting time in the traffic light queue. These values can be retrieved from the present, specific points in simulation or sums or averages of the last n simulation steps.

3) The Multi-Agent System Framework

The Multi-Agent System Framework is a module that is meant to serve as a starting point for the creation of Multi-Agent Systems. Its Framework allows the creation of new agents by following a common interface. There is a moderator between the agents and the API which controls the flow of information between the agents and between the agents and the simulation environment.

The agents themselves are created with a reference to an object with direct access to the simulation environment so that they can query and change the environment autonomously. The Multi-Agent System Framework however ensures that two agents cannot be accessing or altering information from the simulation environment simultaneously to guarantee the integrity of the data. It will then wait for every agent to declare themselves as "ready" and then advance one simulation step.

The communication in the Multi-Agent System Framework is based on an asynchronous message system, and is depicted in Fig. 2. The agents have their own message queue in which other agents, and the moderator, might leave information. This information is variable and highly dependent on the implementation and goals of the application (e.g. the messages can be information requests, answers to requests previously made, advice, general messages sent by the moderator to control the flow of the

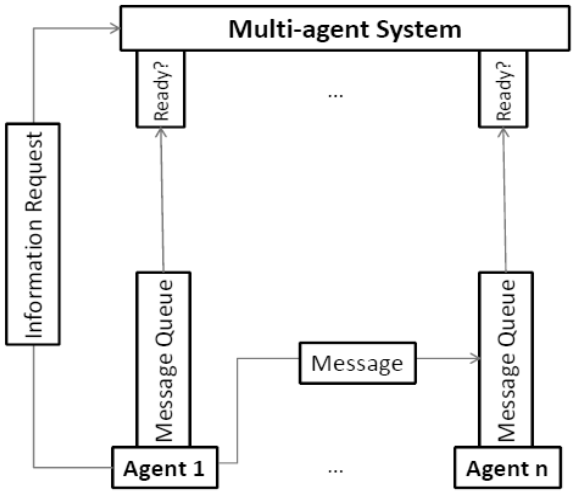

Fig. 2. TraSMAPI message mechanism

negotiation, and more).

Each agent executes on its own thread, in order to be able to go to hibernation when it has declared himself "ready" (so that it would not waste processor time) and wait until the simulation advances one simulation step or there is a new message in his message queue. This opens up the possibility of an agent to declare himself "ready" and then remove himself from that state if a new message arrives at his message queue opening the possibility of an infinite exchange of messages between agents (a maximum value of negotiation rounds should be defined in case the Multi-Agent System cannot reach a stable state, though).

V. A Simple Example

This section was included in order to demonstrate how a simple application can be written using TraSMAPI.

We decided to implement a simple Traffic Light Controller Agent on a road intersection between a general use road and a road used solely by priority vehicles. The Traffic Light Controller Agent should change the state of the traffic light when a priority vehicle approaches, detecting it with an induction loop, to allow it to pass the intersection without stopping as shown in Fig. 3. The simulator used in this simple application was the SUMO [7] simulator.

There are two main objects responsible for the simulator connection and agent interaction which need to be created by our application. The *TraSMAPI* object is used for general simulation control: launching the simulator, connecting and time control. The *MAS* object is responsible for the interaction among the different agents, including message exchange, and is responsible for communicating to the agent changes in the status of the simulation.

The next step consists in creating an Agent and implementing the *Agent* interface. It can be easily done following a template though writing one from scratch is possible and allows greater customization. The agent might be as complex as one wishes, from a static traffic light to a complex, predictive, social controller (in more complex solutions, it is natural to make more complex queries such as the number of vehicles stopped on a given lane or even use the Statistics Module to infer traffic density on different lanes based on the recent past activity).

Our traffic light agent uses a *TrafficLight* object representing the traffic light in the simulation environment. This *TrafficLight* object can be used both to query the simulator and to change the traffic light's state. This is done by calling its methods as if we were acting over the simulation object (i.e. the object in the simulator), hiding all the communication tasks underneath.

The other agent we created was responsible for the induction loop. Even though this agent is not strictly needed since the traffic light agent could query the information relative to the induction loop without violating our multi-agent metaphor, we opted to create it for the sake of demonstration – in this case, of the MAS communication system. The agents were then added to the *MAS* object to be managed.

The induction loop agent is then programmed to retrieve the instant velocity of the priority vehicle when passing over the induction loop and to transmit that information, via the Multi-Agent System Framework message system, to the traffic light agent. The traffic light agent, using that information, schedules the change of the traffic light state. Despite simple, this example uses most features implemented in the system architecture, as well as the collaboration protocol that was devised.

Finally, we assessed our solution using the Statistical Module where we gave special emphasis to the waiting time of the vehicles at the intersection, being the priority vehicles heavily penalized. Naturally, our solution had better results

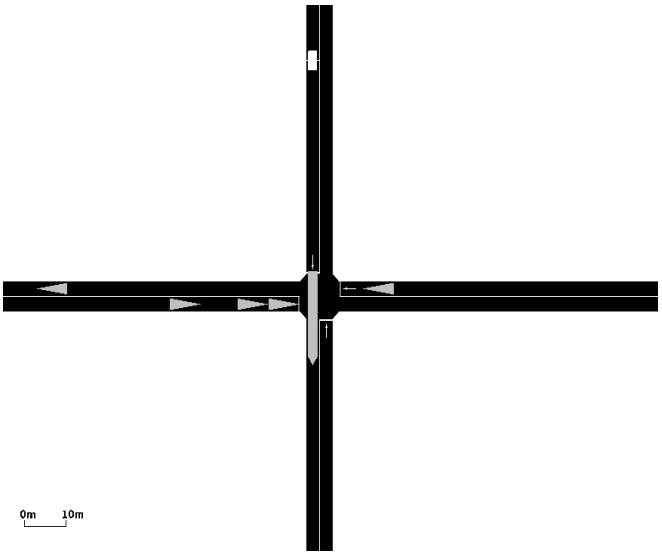

Fig. 3. An intersection between a road with normal traffic (horizontal) and a road reserved to priority vehicles (vertical), simulated using SUMO [7]. The induction loop is visible on the vertical road (white square)

than the static solution.

With this very simple example we were able to use the main features of TraSMAPI such as the Multi-Agent System Framework, its agent-to-agent message system, the basic queries of the Communication Module and the Statistics Module.

VI. Conclusion and Future Work

We are confident that with the creation of generic tools that can serve as basis for the development of traffic management systems we are giving the community the possibility of focusing more on the solutions and less on the implementation details. Having the tool previously described as a starting point allows the developers to skip the set up for the implementation of their applications spending more time of their research looking for actual solutions.

Therefore, future work with TraSMAPI will not be primarily on the tool design and features but rather on the applications that can be developed using it. In fact, the architecture and abstraction made with TraSMAPI enables the development of a broad spectrum of solutions concerning Efficient Traffic Management.

With the GPS becoming a common gadget and its massive use by civil vehicles it is plausible to think of an Advice System based on the information retrieved from the GPS. The vehicles would inform a central system of their position and the system would return advice on the best route available. This kind of system is easily simulated using TraSMAPI given that it is possible to retrieve each vehicle's position using the Communication Module and the exchange of advice is also simplified by the message communication implemented among agents.

It is arguable that a centralized system like the one explained before is impossible to implement efficiently given that it is highly dynamic and the domain is very large. However, it is also possible to create a wireless vehicle-to-vehicle advice system in which the vehicles would "shout" messages about the flow of the traffic to all the neighboring vehicles. If we consider that important messages would be propagated to more than one node, it is not hard to believe that every vehicle in the network would benefit from updated information about the network state. This kind of information together with the information of location and routing provided by the trivial GPS could give birth to a re-routing system that, looking at local solutions, would improve the flow of the entire network.

In a more passive fashion from the point of view of the driver, we can think of intelligent traffic control agents such as traffic lights, variable speed signs, and others. These could adapt to the traffic conditions locally and communicate to improve the flow of the network as a whole.

Acknowledgment

This work has been partially supported by FCT, the Portuguese Agency for R&D, within the BII Framework (Integration into Research Grants).

References

[1] Schleiffer, R. (2002). Intelligent agents in traffic and transportation. *Transportation Research*, **10C**, 325-329.

[2] Oppenheim, N. (1995) Urban travel demand modeling: from individual choices to general equilibrium. Wiley, New York.

[3] Russell, S. J. and P. Norvig (1995). *Artificial intelligence: a modern approach*. Prentice-Hall, Englewood Cliffs.

[4] Steels, L. (1990). Cooperating between distributed agents through self-organisation. In: *Decentralized A.I.* (Y. Demazeau and J. P. Muller, eds.), pp.175-196. North-Holland, Amsterdam.

[5] Haugeneder, H. and D. Steiner (1994). A multi-agent approach to cooperation in urban traffic. In: *Proc. of the Workshop of the Special Interest Group on Cooperating Knowledge Based Systems, CKBS'93*. pp.83-99. DAKE Centre, Keele.

[6] Steiner, D., A. D. Burt, M. Kolb, C. Leri (1995). The conceptual framework of MAI^{2}L. In: *Proc. of the 5th European Workshop on Modelling Autonomous Agents in a Multi-agent World, MAAMAW'93*. LNAI 957, pp.217-230. Springer, Berlin.

[7] Krajzewicz, D., Hertkorn, G., Rössel, C., Wagner, P. (2002) SUMO: Simulation of Urban Mobility. In *Proc. of the 4th Middle East Symposium on Simulation and Modelling*, MESM2002, SCS European Publishing House, pp.183-187.