HAVE'2006 - IEEE International Workshop on
Haptic Audio Visual Environments and their Applications
Ottawa, Canada 4-5 November 2006

MPEG-7 Description of Haptic Applications Using HAML

Mohamad Eid, Atif Alamri, and Abdulmotaleb El Saddik
Multimedia Communications Research Laboratory - MCRLab
School of Information Technology and Engineering - University of Ottawa
Ottawa, Ontario, K1N 6N5, Canada
{eid, atif, abed} @ mcrlab.uottawa.ca

Abstract – *The continuous evolution of computer haptics, as well as the emergence of a wide range of haptic interfaces has recently boosted the haptics domain. Even though efficient tools that support the developer's work exist, little attention is paid to the reuse and compatibility of haptic application constituents. In response to these issues, we propose an XML-based description language, namely Haptic Application Meta Language - HAML. HAML is designed to provide a technology-neutral description of haptic models. It contains ergonomic requirements and specifications for haptic hardware and software interactions. The envisioned goal is to allow for the creation of plug-and-play environments in which a wide array of supported haptic devices can be used in a multitude of virtual environments, with the compatibility issues being handled by automated engines instead of programmatically by the user. As per implementation, MPEG-7 standard has been used to instantiate HAML schema through the use of description Schemes (DS). Our preliminary experimentation demonstrates the suitability of HAML for solving the compatibility issue.*

Keywords – *Haptics, HAML, virtual environments.*

I. INTRODUCTION

Haptics, meaning "of or relating to the sense of touch", refers to the science of perceiving the ambient environment through touch, via the human body [1]. Since the emergence of PHANToM series haptic interfaces from SensAble Inc. in the early nineties [2], haptic devices are getting more diversified and cheaper in the market now. For instance, a desktop haptic device, called Novint Falcon from Novint Technologies Inc., is going to be released in the coming spring, which will cost only about $100 [3]. Therefore, we anticipate that a haptic device will be deployed with PCs for personal customers in the near future. Currently each haptic device uses a different API and thus making application development API- and device-specific. Nowadays we are able to download audio or video clips, and play them back with some standard commercial players, such as RealOne Player or Windows Media Player. Since these videos are coded following certain standard formats, they are highly independent from the graphics cards or displays we are using. Similarly, HAML is meant to define a standard technology-neutral format by which haptic application components such as haptic devices, haptic APIs, or graphic models make themselves and their capabilities known.

Furthermore, the process of incorporating support for multiple haptic devices in a single virtual environment is non-trivial. This is because each haptic device requires a development API that includes specific haptic rendering algorithms, compatible collision detection algorithms, and – in some cases – limited support for general graphical tools. Consequently, HAML is envisioned to be comprehensive, scalable, and extensible to provide the models and XML schema encodings for the representation of haptic applications. The description is divided into seven categories: application general information, haptic device and its capabilities and limitations, the haptic and visual rendering, the haptic API, Quality of Experience (QoE), and haptic data.

There have been several modeling languages, such as SensorML [4] and Transducer Markup Language (TML) [12] that can partially describe a haptic application. For instance, SensorML models a sensor or actuator as a process that has input(s) and produces output(s) based on predefined methods. SensorML could not be efficiently used to describe haptic applications for at least two reasons: first the haptic interface is characterized by bi-directional flow of data/energy where the division between "input" and "output" is often very fine and difficult to define, and second SensorML does not provide description for the mechanical design and behavior of the device – such as applied forces and workspace dimensions. Furthermore, virtual environment modeling languages such as VRML [5] and Web3D Consortium's X3D [6] fall short too in describing the haptic interface hardware and consequently, tailoring the virtual environment to fit a particular haptic device is tiresome, low-level, and programmatic endeavor. The goal of HAML is to intuitively solve this problem by providing a highly descriptive document that enables an interpretive backend engine to discern and solve compatibility issues.

Many researchers realized the need for a formal standard description language for haptic models. For instance, the work in [7] proposed a novel XML-based approach to represent generic haptic application. The model included: application general information, haptic interface, haptic and visual rendering, and the system behavior, among others. Many key features related to tactile and haptic interaction were not covered. Meanwhile, an ongoing effort to introduce a work plan for the development of a new set of ISO standards for tactile and haptic interactions, based on the GOTHI model [8], has been described in [9]. These standards will provide ergonomic requirements and recommendations for haptic and tactile hardware and software, and guidance related to the design and evaluation of hardware, software, and combinations of hardware and software interactions. Even though the proposed draft for the standard provides comprehensive guidelines for haptic interactions, it lacks the description of the application-specific features, the virtual environment, and the quality of experience parameters. Conversely, this paper presents a continuation of our previous work on HAML [15] to describe all aspects related to a haptic application including application level requirements, quality of experience descriptions, and all specifications related to graphic and haptic rendering.

The remainder of the paper is organized as follows: in section 2, we present the scope statement of HAML and highlight the rationale and applications of the proposed Meta language, the HAML framework, and the structural model of the HAML schema. Section 3 describes two examples instances of the general application and device DSs. Finally, in section 4 we pinpoint related issues and our immediate future work.

II. HAML META-LANGUAGE

A. Scope of HAML

HAML is designed to provide a technology-neutral description of haptic models. It contains ergonomic requirements and specifications for haptic hardware and software interactions. In other words, HAML is the standard by which haptic application components such as haptic devices, haptic APIs, or graphic models become self-described. Thus the purpose of HAML is to:

- Provide a standard technology-neutral description language for haptic application components.
- Provide general haptic information in support of device/component discovery.
- Support a standard for haptic data representation.
- Support the processing and analysis of haptic data.
- Solve the incompatibility issues between different haptic devices and APIs.
- Capture application-specific requirements and specifications that might help in programming or configuration of specific applications.

There have been at least three (3) foreseeable approaches to implementing and utilizing HAML instance documents: (1) Application Description: Define a haptic system description that might be used in the future to build similar applications – given equivalent requirements/specifications, (2) Feature Description: The HAML description is obtained from the device/API/model via a manual, semi-automatic or automatic extraction and saved in a storage system for later use, and (3) Haptic Application Authoring/Composition: In this approach, the system receives a query and finds out a set of descriptions matching the user's query. Then an intelligent agent filters the descriptions to compose the haptic application by performing some programmed actions (such as wrapping the API, adapting the haptic rendering algorithms, building the virtual environment, etc.).

B. HAML Framework

The HAML framework is designed to prove that the HAML description could be utilized to make devices, APIs, and their corresponding rendering algorithms almost irrelevant. In order to accomplish this goal, the haptic component is separated from the virtual environment so that the haptic API could be changed without affecting the environment. The basic components of the HAML framework are shown in Figure 1.

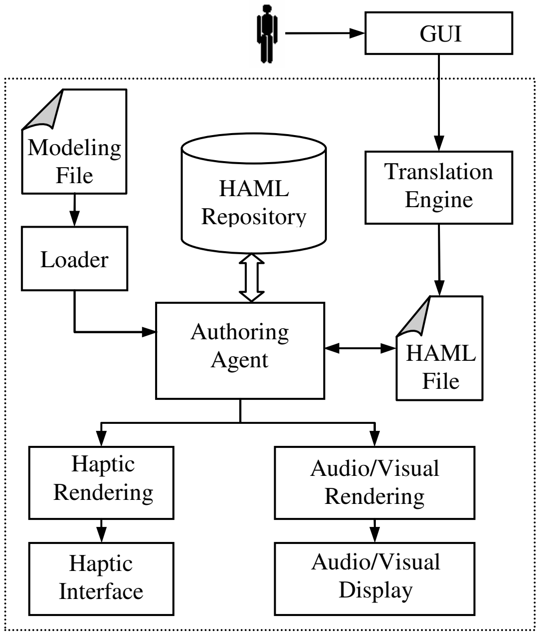

Fig. 1. HAML framework.

The user interacts with the whole framework via the GUI component that captures the basic user requirements (such as the interaction type/device, the virtual environment components, data recording, etc.). These requirements are then passed through the translation engine, which relies on the HAML schema to "pump-out" a HAML-formatted document. This document holds a startup/default configuration of the haptic application required for the framework to work - a discussion of the structuring of HAML will be held later in section 2.3. The Authoring Agent (AA) parses the HAML file to dynamically create the haptic application by selecting and composing components – haptic device, rendering engines, collision detection engines, graphic components, and APIs – that meet the specifications defined in the HAML file and are compatible as well. Notice that the HAML repository stores HAML-formatted description for all available devices, haptic and graphic APIs, and all related information. At this stage, the HAML document is not yet complete, as we are missing the information regarding the newly authored virtual environment application. After the environment has been created and finalized, a "commit" function is performed to update the HAML document accordingly, which means that the HAML document is no longer static.

C. HAML Structure Overview

As mentioned earlier, HAML is haptic-related information, XML based schema meant to describe the haptic device, API, rendering engines (haptic/graphic), general application specifications, quality of experience parameters, and haptic data. As it stands, the structure of HAML consists of seven (7) main categories. It is important to note that this is not the final structure of HAML, and is open to modification as new needs arise.

C.1 Application Description

This section organizes high-level and general considerations of a haptic application, including interaction models and techniques, system requirements, and the Meta meta-data:

a. General application: application name, field of usage, and application type (local versus networked and stand-alone versus collaborative).
b. Interaction task [8]: Tasks may require multiple forms of interaction. There are three main types of interaction tasks: navigation, selection, and manipulation. Navigation tasks include browsing, targeting, searching, zooming, and/or re-orienting the environment. Selection tasks can be for object, group, space, or system properties. Manipulation is described based on: manipulation level, function manipulation, and information retrieval. Manipulation level can be touch, dynamic enabled, and topology changeable. Functional manipulation includes activation, creation, deletion, modification, and management of alternatives, individualization, or personalization. Finally, information retrieval can be either subjective (feeling or motivation) or objective (factual).
c. Interaction techniques: Deal with physical actions required to accomplish various interaction tasks. There are five main interaction techniques: moving relative to object (tracking, tracing, entering, or pointing at an object), moving the object (dragging, pushing, pulling, displaying, and directing the object motion), possessing the object (grabbing, grasping, holding, and releasing), touching the object (tapping, hitting, pressing, squeezing, stretching, and rubbing the object), and gesturing.
d. System requirements: Computer specifications (such as processor, operating system, RAM, video requirements, network card, fire-wire port, PCI slot, etc.), device/SDK/API/Driver support.
e. Meta metadata: Information in this category are meant to describe the HAML document instance being created, such as author information (name, address, company, etc.), dates of creation, modification, and release, intellectual property (copyright, patents, and usage restrictions), and document information (such as HAML version and file location).

C.2 Haptic Device Description

This category attempts to fully describe the haptic device itself. Information about the haptic device includes, but is not exclusive to:

a. Observation Characteristics: Including physical properties (such as inertia, mass/weight, stiffness, hardness/softness, temperature, friction, resonate-frequency, and backend inertia and friction), quality characteristics (accuracy, workspace dimensions, haptic refresh rate, duration, bandwidth, range of sensory and force reflection, Degree-of-Freedom), and response characteristics (minimum and maximum forces, torques, vibrations, and motor DAC).
b. Spatial/Temporal characteristics: Such as device geometry (size, shape, texture, location, and motion) and geometric and temporal characteristics of haptic interface (spatial position, orientation, velocity, control button status, and deformation).
c. Description and documentation: This section includes: (1) identification information such as device name, type, model, serial number, and manufacturer, (2) overall information about the device such as driver(s), control type, haptic and graphic API compatibility, owner, and operator, (3) device reference that might be ground-based, body-based, or un-based, and (4) history and system requirements.

C.3 Haptic API Description

This category describes the haptic development API and includes details of:

a. API general information: Such as API name, version, programming language, support information, supported platform(s) and devices, graphic modeling capabilities, and calibration (automatic versus manual).

b. API generic functions: This describes the basic functions that are common among haptic APIs such as device calibration, initialization, object creation and deletion, property setting, scene graph loading, force interaction (create/get force/torque and graphic and event callbacks), and device release.
c. Error reporting and handling: Such as function error, force error, device error, rendering error, and scheduling error.
d. Haptic and graphic scene synchronization: That might be either synchronous or asynchronous callbacks.

C.4 Haptic Rendering Description

This includes the basic concepts related to haptic rendering, namely collision detection, force generation, and control algorithms:

a. Haptic rendering system (haptic rendering API and servo loop rate).
b. Collision detection: Including position and contact information, indentation, collision detection approach (such as linear programming, kinetic data structures, etc.), collision detection techniques (such as intersection tests, proximity queries, bounding box), types of queries (Boolean, disjoint separation distance, penetration separation distance, discrete or continuous intersection), and response schemes (no response, geometric or mechanical responses).
c. Force response: This comprises the force computation technique and response technique (geometry or surface based).
d. Control algorithms: This includes force generation method, interaction models such as impedance and admittance (open-loop and compensation).

C.5 Graphic Rendering Description

All general and programmatic (non-scene describing) graphics-related information is grouped in this category:

a. Graphic rendering system: Such as Graphic rendering API name, release version, graphic rendering refresh rate, display frequency, depth buffer bits, graphic modeling language, and graphic programming language.
b. Contact model: Contact models use one of two possible techniques: historic or non-historic. Historic methods use previous position along with current position to check for collision whereas non-historic methods use only the current position.
c. Avatar representation: The device avatar can be point-based, multi-point based, or ray-based.
d. Object modeling methodology: The most popular methodologies for object modeling are: polygonal, bi-cubic parametric patches, Constructive Solid Geometry (CSG), spatial subdivision technique, and implicit representation.
e. Model type: There exist many classification types for object modeling in a virtual environment. One possible way to classify object modeling is: rigid, deformable, or dynamic. Rigid objects can be either constraint-based (penalty-based or analytic-based) or impulse-based, whereas deformable objects can be geometry-based (vertex-based versus spline-based) or physics-based (continuum-based, approximate continuum-based, particle-based, or finite element based).

C.6 Quality of Experience Description

The characteristics of the sensations, perceptions, and opinions of people as they interact with their environments are grouped in this category:

a. Haptic perception: The study of haptic perception is usually divided into haptic exploration and manipulation. Haptic exploration is usually evaluated against well-established metrics (such as roughness, hardness, and stickiness or blurriness, distortion, and aberration).
b. Haptic stability: To maintain mechanical stability, many parameters must be kept within thresholds. Examples of these parameters are force threshold, force duration, torque ripple, damping factor, stiffness factor, and sampling rate.
c. Quality of Service (QoS) parameters: Some generic quality of service parameters for the application are listed and defined in this section (such as cost, execution time, latency, throughput, security, trustworthiness, and availability).
d. Haptic Interface quality: This section measures the device quality in terms of Symmetricity (as per inertia friction, stiffness, and resonance frequency), balance (range, resolution, sensing and actuation bandwidth), and back drive (inertia and friction).

C.7 Haptic Data Description

This section describes the technical data that can be measured within the haptic application (such as data types, acquisition and encoding techniques, and compression algorithms).

a. Data unit: A data unit is characterized by format, range, resolution, type (timing, trajectory, force feedback, or material property), and data type (such as simple, aggregate, and constraint data types).
b. Data acquisition: Data acquisition incorporates techniques and methodologies related to sampling and encoding the haptic data. The encoding technique can be based on the object properties, spatial attributes, temporal attributes, perceptual attributes, or content specific. It can be hybrid by combining techniques as well. Encoding format can be either text-based or graphic-based (examples of graphic formats include maps, pictures, figures, charts, textures, or

animations). The encoding rhythm can be either subjective or objective. As per sampling, there are many methodologies such as fixed sampling, grouped sampling, adaptive sampling, or adaptive delta pulse code modulation. Sampling quality parameters might be included in this category (such as sampling rate and resolution).

c. Data compression: This section groups compression algorithms descriptions and parameters such as compression class, approach, and settings. Compression class is classified as lossy or lossless compression, whereas the compression approach can be either off-line or run-time. Finally compression settings contain – but not limited to: compression method, compression factor, degradation factor, and Just Noticeable Difference (in terms of force, velocity, position, and torque).

III. IMPLEMENTATION

A. MPEG-7 Overview

MPEG-7 is a standard that allows interoperable search and access to multimedia data by attaching metadata to multimedia contents [13]. To create descriptions, MPEG-7 offers a comprehensive set of audio-visual metadata elements, and their structure and relationships. These elements are defined in the form of Descriptors (D) and Description Schemes (DS).

B. DS Examples

In this section, we present two DS examples to proof how intuitive and simple it is to instantiate XML-based HAML descriptions. We consider the virtual maze application [14], developed at the University of Ottawa DISCOVER Lab, as an example of general application DS, and the Omni PHANToM interface as a device DS, both are shown in Figures 2 (device) and Figure 3 (application).

IV. CONCLUSION AND FUTURE WORK

We have proposed seven (7) description schemes for HAML that could be implemented using MPEG-7 standard. We have paid particular attention to the structuring contents of the seven schemes and presented two examples of how such description schemes looks like. We admit that the cost and effort associated with generating DSs are critical indeed, but the good thing is that once the content is generated, it can easily be reused by people.

An integral part of our future work is finalizing the HAML schema, and extending it considerably. As with any standard, HAML is subjective to further extension and evolvement, and suggestions and novel ideas are welcomed. As mentioned earlier, the diversity of devices and their methodology of implementation should be catered to as much as possible. As per implementation, we will be building the authoring agent that will utilize the DS contents to compose an application – with minimum intervention from the end user.

```
<?xml version="1.0" ?>
<DeviceDS xmlns="http://www.mcrlab.uottawa.ca/HAML"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www. mcrlab.uottawa.ca deviceds.xsd">
<ObservationCharacteristic>
<PhysicalProperties>
  <Inertia>45</Inertia>
  <Footprint>
          <Width>168</Width>
          <Depth>203</Depth>
  </Footprint>
  <Weight>
          <Pounds>3</Pounds>
          <Ounces>15</Ounces>
  </Weight>
   <MinimumStiffness> x-axis (1.26 N/mm), y-axis (2.31 N/mm), z-axis
   (1.02 N/mm) </MinimumStiffness>
   <BackendFriction>0.26</BackendFriction>
</PhysicalProperties>
<QualityCharacteristics>
 <PositionResolution>0.055</PositionResolution>
 <WorkspaceDimensions>
          <Width>160</Width>
          <Height>120</Height>
          <Depth>70</Depth>
 </WorkspaceDimensions>
  <HapticRefreshRate>1000</HapticRefreshRate>
  <RangeOfSensoryReflection>x|y|z</RangeOfSensoryReflection>
 <RangeOfForceReflection>x|y|z</RangeOfForceReflection>
 <DOF>6</DOF>
</QualityCharacteristics>
<ResponseCharacteristics>
 <MaximumForce>3.3</MaximumForce>
 <ContinuousExecutableForce>0.88</ContinuousExecutableForce>
</ResponseCharacteristics>
<DescriptionDocumentation>
  <Identification>
          <DeviceName>Omni</DeviceName>
          <DeviceType>PHANToM</DeviceType>
          <Model>Omni</Model>
          <SerialNumber>123456789</SerialNumber>
          <Manufacturer>SensAble Inc. </Manufacturer>
          </Identification>
  <OverallInformation>
          <Driver>Phantom Device Driver V 4.2.26</Driver>
          <Platform>Intel-based PCs</Platform>
          <CompatibleAPI>OpenHaptics | CHAI 3D|Reachin
          </CompatibleAPI>
          <Owner>University of Ottawa</Owner>
          <Operator>DISCOVER Lab</Operator>
  </OverallInformation>
  <DeviceReference>Ground-Based</DeviceReference>
</DescriptionDocumentation>
</ObservationCharacteristic>
</DeviceDS>
```

Fig. 2. DS for the Omni PHANToM device.

```
<?xml version="1.0" ?>
<ApplicationDS xmlns="http://www.mcrlab.uottawa.ca/HAML"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www. mcrlab.uottawa.ca
applicationds.xsd">
<ApplicationName>Virtual Maze</ApplicationName>
<FieldOfUsage>Gaming</FieldOfUsage>
<ApplicationType>
	<HostBased>
		<StandAlone>yes</StandAlone>
	</HostBased>
</ApplicationType>

<InteractionTask>
<Navigation>
	<Browsing>
		<ExploringEnvironment>yes</ExploringEnvironment>
	</Browsing>
</Navigation>
<Manipulation>
	<ManipulationLevel>
		<Touch>yes</Touch>
	</ManipulationLevel>
</Manipulation>
</InteractionTask>

<InteractionTechnique>
	<MovingRelativeObject>
		<EnteringObject>yes</EnteringObject>
	</MovingRelativeObject>
</InteractionTechnique>

<SystemRequirements>
	<ComputerSpecifications>
		<Processor>Pentium II | AMD</Processor>
		<OperatingSystem>Win XP | Win 2000 | Red Hat / Mac
		X</OperatingSystem>
		<Ram>32 MB</Ram>
		<FirewirePort>IEEE 1394</FirewirePort>
		</ComputerSpecifications>
		<SupportedDevice>Phantom Omni</SupportedDevice>
		<SupportedSDK>Reachin</SupportedSDK>
		<SupportedAPI>Phantom Device Driver V
		4.2.26</SupportedAPI>
	</ComputerSpecifications>
</SystemRequirements>

<MetaData>
	<AuthorInformation>
		<Name>A. El Saddik</Name>
		<Address>Ottawa</Address>
		<ContactInformation>abed@mcrlab.uottawa.ca</ContactIn
		formation>
		<Company>University of Ottawa</Company>
	</AuthorInformation>
	<Dates>
		<CreationDate>01/09/2005</CreationDate>
		<ModificationDate>15/07/2006</ModificationDate>
	</Dates>
	<DocumentInformation>
		<HAMLVersion>1.0</HAMLVersion>
		<DocumentLocation>Local</DocumentLocation>
	</DocumentInformation>
</MetaData>
</ApplicationDS>
```

Fig. 3. DS for the virtual maze application.

REFERENCES

[1] J. K. Salisbury, and M. A. Srinivasan, "Sections on Haptics, In Virtual Environment Technology for Training (BBN Report No. 7661)", Cambridge, USA: The Virtual Environment and Teleoperator Research Consortium (VETREC) affiliated with MIT.

[2] SensAble Technologies: P roducts P age, OMNI P hantom (2006). Available online at: http://www.sensable.com/ products/phantom_ghost/phantom-omni.asp.

Novint Technologies, Inc., http://www.novint.com/.

[3] SensorML developer website (2006), Available online at http://vast.uah.edu/SensorML/.

[4] Web3D Consortium, "The Virtual Reality Modeling Language", accessed on 05/05/2006, http://www.w3.org/MarkUp/VRML/

[5] Web3D Consortium, "X3D", accessed on 05/31/2006, http://www.web3d.org/.

[6] Zhou J., Shen X., Shakra I., El Saddik A., and Georganas N. D. XML-based Representation of Haptic Information. In proceedings of the IEEE International Workshop on Haptic Audio Visual Environments and their Applications, Ottawa, Canada, October 2005.

[7] Carter,J., van Erp, J.,, Fourney,D., Fukuzumi, S., Gardner,J., Horiuchi, Y., Jansson, G., J_rgensen, H., Kadefors, R., Kobayashi, T., Kwok, M.G., Miyagi, M., Nesbitt, K.V. The GOTHI Model of Tactile and Haptic Interaction. In Proceedings of GOTHI'05 Guidelines on Tactile and Haptic Interactions, October 24-26, 2005.

[8] Van Erp, J. B.F., Andrew, I., and Carter, J. ISO's Work on Tactile and Haptic Interaction Guidelines. In proceedings of the EuroHaptics 2006, paris, France 2006.

[9] Immersion Technologies, 3D Interactions P roduct P age (2006). Available online at: http://www.immersion.com/3d/products/cyber_grasp.php

[10] SensAble Technologies, "Open Haptics Toolkit", accessed on 05/31/2006, http://www.sensable.com/products/phantom_ghost/OpenHapticsToolkit-intro.asp.

[11] CHAI 3D, "The Open Source Haptics P roject", accessed on 05/28/2006, http://www.chai3d.org/.

[12] Transducer Markup Language official website: http://www.transducerml.org/standards.htm

[13] P . Salembier and J. R. Smith. MP EG-7 multimedia description schemes. 1EEE Transactions on Circuits and Systems for Video Technology, Vol. 11, NO. 6, June 2001.

[14] M. Orozco, Y. Asfaw, A. Adler, S. Shirmohammadi, and A. El Saddik. Automatic Identification of P articipants in Haptic Systems. Instrumentation and Measurement Technology Conference Ottawa, Canada, 17-19 May 2005.

[15] F. R. El-Far, M. Eid, M. Orozco, and A. El Saddik. Haptic Application Meta-Language. Accepted for publication in the 10-th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications, Torremolinos, Malaga, Spain, October 2 - 4, 2006.