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

Towards a Standard Modeling of Haptic Software System

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

Abstract – *Computer haptics refers to the discipline concerned with generating and rendering haptic stimuli to the human user. The last decade has witnessed a rapid progress in haptic applications software development. We envision a need for a standard for haptic application software modeling. This paper introduces the approach of the Unified Modeling Language based haptic software engineering. We present the rationale and a reference model for haptic software development, and propose the basic modeling technique that comprises modeling elements, notation, and methods for haptic software systems. A startup systematic engineering process that describes how a haptic software system could be developed is also presented. Finally, we summarize our findings and provide vision for future work.*

Keywords – *Haptics, software engineering, UML modeling.*

I. INTRODUCTION

Haptics, a term which was derived from the Greek verb "haptesthai" meaning "to touch", refers to the science of touch and force feedback in human-computer interaction. Currently research on haptics is categorized into human haptics, machine haptics, and computer haptics [1]. While human and machine haptic fields are beyond the scope of this paper, our focus is on computer or visualized haptics. Just as computer graphics deals with generating and rendering visual images, computer haptics is concerned with rendering haptic stimuli to human users. It covers all aspects related to the development of haptic applications.

Most haptic software systems follow the architecture shown in Figure 1 to incorporate visual, auditory, and haptics feedback. The synchronization engine is responsible for computing the virtual environment's behavior over time. Generally, haptic-based applications require the visual and/or auditory interaction to provide a complete or realistic sensory feedback to the users. The Audio/Visual rendering interface displays the rendered graphics and sound using the audio/video transducers. The audio/video transducers convert audio and visual signals from the computer into a form perceivable by the human operator. The importer basically loads the constructed virtual reality (VR) models into the VR environment. These models might have graphic and/or haptic parameters. The VR object constructor accepts the VR model specification from the correspondent interface and assembles the object. VRML interface, X3D interface, and JX3D interface are interface components that interpret the different VR models from their source files according to their modeling rules.

This work is motivated by the lack of an object oriented software engineering approach for haptic software systems. General object-oriented software engineering approaches, such as the Unified Process [2] and the Rational Unified Process [3], are not sufficient to model haptic applications as they do not incorporate characteristics unique to the haptic modality such as: haptic rendering, graphic rendering, and contact modeling. Currently, and to the best of our knowledge, there exists no software engineering process that allows the systematic development of haptic software systems. In addition, UML [4] was the choice for this work since it is well established and been standardized by OMG [5]. The use of UML for modeling purposes is essential, as there is a guarantee that UML is updated and improved. Moreover, it is supported by tools, conferences and books, and even most importantly UML improves the communication between people involved in a software development project as they speak the same language.

The proposed idea is for a software engineering approach that consists of an object-oriented, incremental, and iterative development process. The main focus of the process is the description of a systematic methodology for the analysis and design of haptic applications. An extension to the UML – or UML profile – is required to provide an adequate notation for the visual representation. This profile allows for an easy construction of domain, presentation, haptic rendering, graphic rendering, and contact models, integrated and standardized in the methodology. The proposed approach will be validated using several case studies.

1-4244-0761-3/06/$20.00 ©2006 IEEE

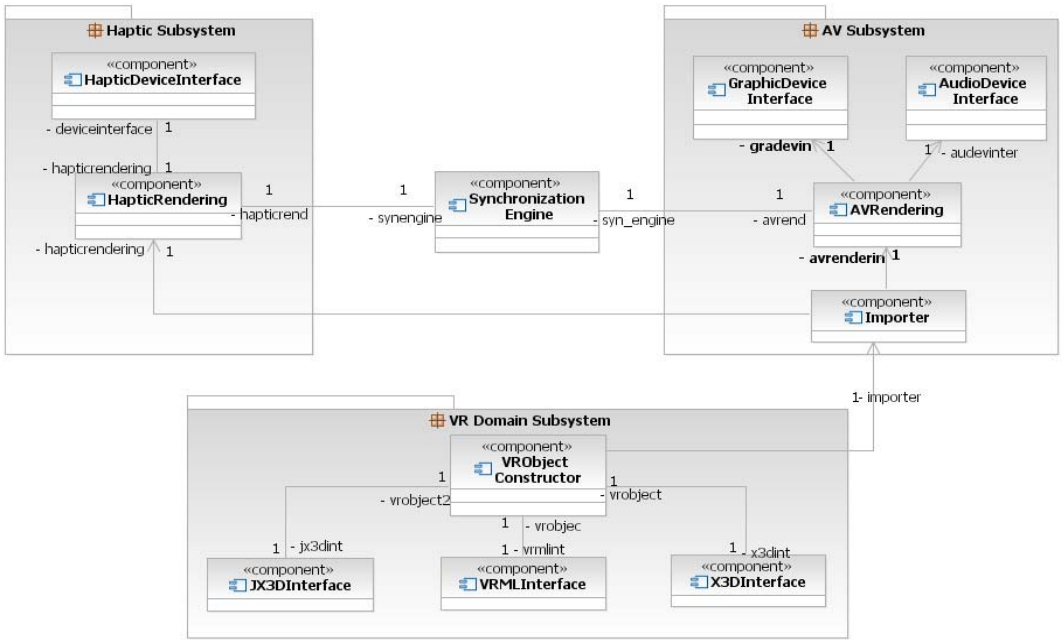

Figure 1. The structure of a VR application incorporating visual, auditory, and haptics.

This paper is organized as follow: section 2 introduces the concept of haptic software system. Section 3 discusses the rationale for a standard development process for haptic software systems. The following sections describe our brainstorming for the approach: Section 4 talks about the object-oriented reference model for haptic software systems and its importance, section 5 discusses the modeling technique and the preliminary distinguished analysis and design models for haptic software systems, section 6 briefly defines the workflows that are initially proposed for the development process. Finally, section 7 recapitulates what has been stated in this paper and presents our immediate future work.

II. WHAT IS HAPTIC SOFTWARE SYSTEM?

Haptic applications are complex software systems, whose development process demands an exhaustive feasibility study, adequate planning and experience in the construction of haptic rendering, device interface modeling, and collision detection techniques.

Any haptic system consists of software and hardware components. The hardware components mainly are the haptic interfaces and audio/visual transducers used in the system that could be application specific. The haptic software subsystem is almost the same in most of the haptic systems. In Figure 2, the basic components of haptic subsystem, namely haptic rendering component and haptic device interface. The haptic rendering is the core of any haptic-based software system. It manages the algorithms to detect and report when and where the geometric contact between the end effector point of the haptic device and the virtual environment objects has occurred. It also computes the correct interaction forces between the haptic interface and its virtual environment. Haptic rendering comprises three parts: collision detection, force response, and control algorithm. Collision detection is the task of determining over time whether any points of two given objects – which may have different representations – occupy the same location in space simultaneously. After a collision is detected, force-response algorithms are fired to compute the interaction forces between avatars and virtual objects. Control algorithms command the haptic device in such a way that minimizes the error between ideal and applicable forces. Finally, the device interface component provides the haptic subsystem with the required custom methods and tools to instruct the haptic device to do various interactions.

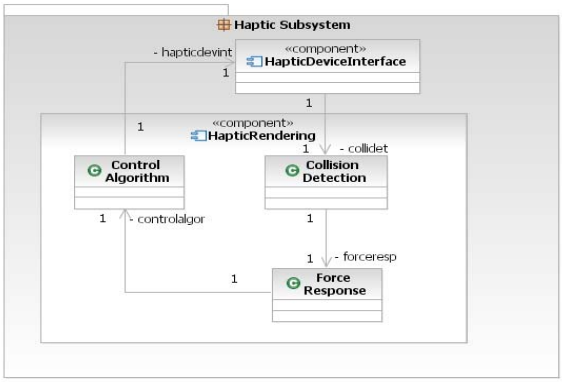

Figure 2. Basic architecture of a haptic subsystem.

III. MOTIVATION FOR STANDARD DEVELOPMENT OF HAPTIC SOFTWARE SYSTEMS

The development of haptic software systems differs from the development of any traditional software. Many of the existing haptic systems are built as prototypes. Their implementation is usually performed in an ad hoc fashion and is improved in successive steps.

Developers from both the industry and academy still consider haptic software development as an authoring activity rather than an application development to which well-known software engineering practices could apply. Moreover, traditional software engineering methodologies can not be applied as-is, and if applicable they are not precise enough to describe and fit haptic application. As far as we know, there is no currently systematic engineering process which describes how haptic software systems should be developed.

As we mentioned before, haptic systems are complex software systems. It has the distinguished feature of real-time bidirectional interaction with the human user. The division of "input" and "output" is usually very fine and hard to model. Therefore, they require an appropriate software engineering process. We believe that a standard development process for haptic software systems is welcomed by the haptic community because of the followings:

- *Documentation:* the lack of documentation for current haptic software systems. This lack is a logical result from the fact that many of the existing haptic systems are built in an ad hoc fashion.
- *Reusability:* the development process will provide predictable building blocks for others to use in the design of haptic software system so that the need for re-invention is minimized.
- *Cost reduction:* with the existence of a standard development process that includes best practices in the development of haptic software systems the cost of creation and innovation is significantly reduced.
- *Readability:* using a standardized development process for haptic software systems makes them more readable and thus minimizes the efforts spent to understand already developed haptic software systems.
- *Maintainability:* the huge number of events expected in the interaction between the haptic device and the user in addition to the 3D space arrangement increase the complexity of haptic software systems and the possibility of missing some important details of the software application increase. Standard development process makes the software more maintainable.
- *Synchronization:* Particular attention must be paid to the synchronization of the visual, auditory, and haptic displays as it might be problematic because each modality requires different types of approximations to simulate the same physical phenomenon.
- *Security:* sometimes haptic systems are critical applications that deal with important information and data and a degree of security is needed. In most of current approaches of development security is not considered.
- *Haptic rendering:* there is considerably high data rate that should be sampled and processed from the haptic device by the haptic software system. This requires the haptic software to use some comprehensive algorithm for collision detection and force feedback in order to detect and capture the interaction between the system and the user.
- *Diversity of skills:* people involved in the development of haptic software systems require different skills, such as haptic experts, graphic designers, programmers, and multimedia experts. And current available development processes do not fit these type roles or skills.
- *Personalization:* the structure of the system domain is becoming more user-specific. For example, if the haptic application to be developed is intended for the medical surgery domain, then surgeons are assumed to be the users of the system and they should be involved in the structuring of the system domain.
- *Complexity:* due to the fact that haptic interactions incorporate real-time bidirectional flow of data – from and toward the user – a lot of complexities will be associated with interfacing the haptic device to the software system. This will increase the overall complexity of the haptic software system.

For all the up-mentioned reasons we believe that there is a need to standardize the development process of haptic software systems.

IV. THE REFERENCE MODEL

The reference model for haptic software systems will be elaborated in order to identify the features that characterize this type of software as a prerequisite step to the definition of the best to-be-used modeling technique. This reference model is an object oriented meta-model that defines all modeling elements and relationships associated with haptic software, and will use a general terminology applicable to every application field in haptics. The model is formally represented by UML and specified by OCL [6]. UML provides the notation and the object-oriented modeling techniques for the visual representation of the reference model, whereas OCL is used for the formal specification of invariants on the model elements and attributes and pre-conditions and post-conditions for the functions as well.

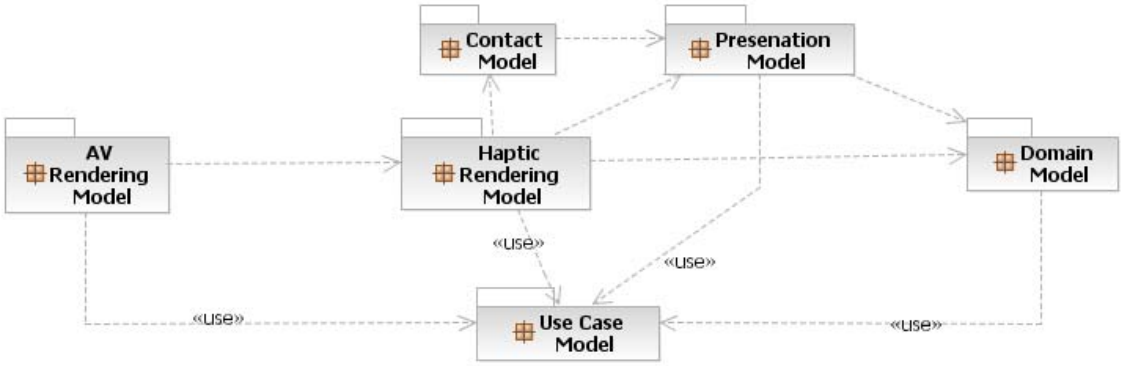

Figure 3. The basic analysis and design models of the modeling technique.

V. THE MODELING TECHNIQUE

The proposed modeling technique comprises the modeling elements, their notation, and a method to build set of models for haptic software systems. Therefore, the modeling technique will focus on the haptic software concepts like haptic rendering, graphic rendering, and haptic device.

The notation and semantics of the modeling elements will define a "lightweight" UML profile for haptic software systems development. It is defined as a set of stereotypes for haptic systems. These stereotypes are built using the UML extension mechanism, and thus, will support visual modeling. They are used to indicate the descriptive and restrictive properties that the modeling elements have in comparison with standard UML elements.

The method consists of the construction of some analysis and design models. Figure 3 shows the basic idea of these models represented as UML packages related by usage and dependency relationships. Basically, there will be six analysis and design models generated using this method. The use case model consists of a use case diagram that captures the system functionality and description scenarios. The domain model consists of a class diagram identifying the objects of the problem domain and their relations. The presentation model describes the structure of the software presentation elements using a class diagram, and captures the dynamic behavior of these elements using a sequence diagram. The presentation model is built before its dependant models: the contact model and the haptic rendering model. The contact model describes the abstract interface to haptic devices used in the software by a composite objects and its dynamic behavior by state machine diagrams. The haptic rendering model describes the collision detection algorithm used for the specific haptic application. The model comprises a class diagram that defines the detection and controlling algorithms and a sequence diagram showing how these elements cooperate together. To construct the haptic rendering model all the other models should be available except the graphic rendering model, which actually depend mainly on the haptic rendering model. Finally, the audio visual rendering model depicts the audio and visual behavior generated within the system and how they are rendered to back to the human operator. The structure of this model is depicted using a class diagram and the rendering mechanism is explained using a sequence diagram. A state machine diagram may be used to illustrate the life cycle of the rendering engine. Each model is built using the notations provided by the UML while applying the extension mechanism of the UML, when necessary.

VI. THE DEVELOPMENT PROCESS

From a software-engineering perspective, the proposed development process comprises the widely accepted workflows: study feasibility, requirement capturing, analysis, design, implementation, and test and maintenance [7]. The idea is to enrich the software development process with aspects from the haptic software prospective. The development process covers the entire life cycle of haptic software systems; it moves through a series of iterations and increments and uses UML notation and diagrams. In the following subsections, we briefly define the workflows that are initially proposed for the development process.

A. Study Feasibility

Feasibility study is required to define and evaluate the worthiness for haptic software that performs specific task. The global functional requirements of the application should be defined, and a first budget and schedule plan is to be worked out. It is recommended to develop a prototype with a minimized effort just in case the idea fails to live up, even though it is an expensive alternative.

B. Requirements Capturing

In case the feasibility study was positive, this workflow as well as the following ones will only be realized. The goal of this workflow is to describe what the haptic software system should do and allows the developers and the stakeholder to agree on that description. This workflow differs from other similar workflows in other development process in that it concentrate on capturing the different aspect of touch interaction that occur between the haptic software system and end users.

C. Analysis and Design

The objective of this phase is to analyze and specify the haptic software to be built. We do not include the design as a separate workflow as it is considered as a refinement of the analysis. The first iteration corresponds to a rough analysis, followed by successive refinements until an implementation orientation is reached during the last iteration. In this workflow, all the previously mentioned models in section 4 should be constructed and visualized using the modeling technique.

D. Implementation

The objective of this workflow is to define and implement classes and objects in terms of components (source files, binaries, executables, and others). Particular attention should be paid to integrate this workflow with the haptic API needed for the haptic software system. The decision of where and how the integration should occur will be made by the developer in this workflow.

E. Test

The objective of the workflow is to verify that the interactions between haptic software system objects are correct, the integration between haptic system components is accurate, and the predefined haptic requirements of the software system are well addressed. Special methods of testing may be needed to enforce the functional and non-functional haptic requirements of the software system.

VII. CONCLUSION AND FUTURE WORK

In this paper we have shown that haptic software systems require specific software engineering process. We also discussed the need for a reference model for haptic software systems. Also, we identified six different models that are constructed during the analysis and design of haptic software systems using an extension UML profile. Also, we briefly discussed our vision for five workflows that haptic application development should go through.

For the time being, the reference model is under development. It will provide the basis for the UML extension or profile that will be used in the modeling technique. We envision uncharted waters in the development of this meta-model. The instantiation of the different reference modeling elements will be the UML profile, and the modeling techniques will be developed accordingly. Also, we plan to integrate the developed modeling technique into a well-established open source UML CASE tool such as ArgoUML [8], which in turn enables developers to utilize the modeling technique and partially automate the development process.

As a proof of concept, we will investigate a case study of a haptic enabled UML CASE tool. Using this CASE tool, the user is enabled to feel different types of force feedback and sense the stiffness of the modeling elements. We are planning to finish this task before gaining a global view of the development process because the modeling technique is at the core of the development process.

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] Jacobson I., Booch G., & Rumbaugh J.. The Unified Software Development Process. Addison Wesley. 1999.

[3] Kruchten P. The Rational Unified Process: An In troduction. Addison Wesley. 1998.

[4] The Unified Modeling Language Resource Page, accessed on 08/11/2006, http://www.uml.org/

[5] The Object Management Group, accessed on 08/11/2006, http://www.omg.org.

[6] Warmer J. and Kleppe A. The Object Constraint Language: Precise Modeling with UML. Object Technology Series. Addison Wesley. 1999.

[7] Jacobson I., Booch G., & Rumbaugh J. The Unified Software Development Process. Addison Wesley. 1999.

[8] ArgoUML website: http://argouml.tigris.org.