IDEA GROUP PUBLISHING

701 E. Chocolate Avenue, Hershey PA 17033-1117, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

ITB8047

Chapter XIX

HTSPN: An Experience in Formal Modeling of Multimedia Applications Coded in MHEG or Java

Roberto Willrich
INE-UFSC, Brazil

Pierre de Saqui-Sannes and Patrick Sénac
ENSICA, France
LAAS-CNRS, France

Michel Diaz
LAAS-CNRS, France

Multimedia authoring involves people not familiar with formal design techniques, but eager to check documents against synchronization, scheduling and resource allocation errors. Visual modeling with Petri Nets answers that need with a graphic syntax and a formal semantics that makes models executable. A Petri Net defines a bipartite graph where summits are divided into places associated with data processing functions and transitions which represent synchronization points. The chapter discusses an experiment with Hierarchical Time Stream Petri Nets, a timed extension of Petri Nets which was designed with distributed multimedia systems in mind. HTSPNs define a unified framework to structure complex and interactive documents. Dynamic synchronization strategies based on temporal intervals composition make it possible to take the asynchronous nature of distributed systems into account. Last but not least, a HTSPN remains an open model, which can be extended with implementation details regarding, e.g., the URLs and the presentation characteristics. The interpreted model or I-HTSPN for short, has been instantiated for MHEG and Java, respectively. The MHEG I-HTSPN editor, document checker and code generator were prototyped in C++ for Solaris. The Java I-HTSPN counterpart has been developed in Java. Their use is exemplified on a guided tour of a university and a distance learning application.

INTRODUCTION

The advent of e-commerce, distance learning and home entertainment systems has stimulated research work on document authoring in multimedia information systems. Clearly, the objective is to supersede Web page composers, and to overcome problems inherent to the Internet: lack of guarantee on transmission delays, variable delays characterized by a jitter and a Quality of Service limited to a "best-effort" policy, just to mention a few.

Given the complexity reached by multimedia information systems, formal design techniques have received particular attention for the possibility given to carry out proofs on a model and to derive an implementation from a model checked against design errors. An FDT essentially relies on a modeling language that differs from an implementation one by its capacity to abstract from low level implementation details and to focus on critical mechanisms that make the core of the application, system or document.

Multimedia authoring techniques have used visual modeling languages, and timelines in particular. Text, sound, images and video data units are combined on a temporal line, so as to build up a multimedia scenario that can be understood by authors not necessarily familiar with task scheduling techniques. Because they assign a fixed duration to each data unit, timelines convey the impression that multimedia documents could be authored independently of network constraints, thus ignoring the asynchronous nature of distributed systems. The opposite is true. A document model must include QoS parameters, and describe recovery procedures that apply in case of QoS violation or degradation. For instance, in case of bandwidth reduction, a video will be replaced by a still image, so as to keep the user's attention alive.

Commercial tools, such as ToolBook, IconAuthor, Director and Authorware, have evolved accordingly, but still miss analysis techniques to predict and detect design errors before actual implementation. By contrast, this chapter shares an experiment in hypermedia authoring with timed extension of Petri Nets, a Formal Design Technique that enables visual modeling and relies on a formal semantics that makes models executable. First, execution can take place in a verification context where the purpose is to check documents against design errors, including incorrect schedules, synchronization failures and resource allocation problems. Second, the debugged model can be augmented with implementation details, such as the URLs.

In this chapter, we first discuss multimedia document design in general and survey related work. Afterward, we introduce HTSPN, the formal model which serves as reference throughout the chapter. We present two interpretations of the HTSPN model for the production of MHEG (ISO, 1997a) and Java code. Finally, we conclude the chapter and outline future work.

DESIGN OF INTERACTIVE MULTIMEDIA DOCUMENTS

The design of interactive multimedia documents demands computer skills and artistic contributions in team projects that involve software developers and book authors. The former include programmers, system and network architects and specialists in image, audio and video processing. The latter hesitate to invest in technical issues and base authoring on an 'attempt and error' strategy. The challenge on the human side is to help computer people and artists work together.