Player para Sincronização Multimídia Utilizando Especificação Baseada em Redes de Petri - HTSPN

Álvaro Nunes Prestes, André Mesquita Rincon, Jânio Elias Teixeira Júnior, Carlos Eduardo de Lima, Fabiano Fagundes

Centro Universitário Luterano de Palmas (CEULP/ULBRA)

{alvaro,rincon,janiojunior,careli,fagundes}@ulbra-to.br

Abstract. *In this paper will be presented a proposal of a player for Multimedia synchronization using Petri Net specifications. The synchronization will be made using the PNSched language (XML) based in HTSPN, that was created to a syncronization system under development in which the player is inserted. Besides the proposal that will be presented, the paper also describes the PNSched language, details about OCPN, TSPN and HTSPN and more information on the player implementation.*

Resumo. *Este artigo apresenta uma proposta de um player para Sincronização Multimídia utilizando uma especificação baseada em Redes de Petri. A sincronização será feita a partir da linguagem PNSched (XML) baseada em HTSPN, que foi criada para o desenvolvimento de um sistema de sincronização no qual o player proposto está inserido. Além da proposta que será apresentada, o artigo também traz a linguagem PNSched, detalhes sobre OCPN, TSPN e HTSPN e maiores informações sobre a implementação do player.*

1. Introdução

Um sistema que utiliza duas ou mais mídias (áudio, imagem, animação, vídeo, texto) é, de forma geral, chamado de sistema multimídia. Conceituando, um sistema multimídia é um sistema ou aplicação que suporta o processamento integrado de vários tipos de mídia (contínuas e discretas), com ao menos uma mídia contínua (BLAKOWSKI, 1996).

Tendo em vista a definição de sistemas multimídia de BLAKOWSKI (1996) e analisando uma das principais características desse tipo de sistema que, segundo LITTLE (1990), consiste no trabalho em conjunto com vários tipos de mídias heterogêneas, chega-se a uma das motivações do trabalho proposto: desenvolver um *player* para trabalhar a sincronização de mídias com diferentes características.

O *Player* descrito neste artigo está inserido no contexto de um ambiente de sincronização multimídia que possui, além do *Player*, outros módulos tais como um Editor Gráfico, um *Parser*, um Simulador de Redes de Petri Temporizadas (que irá trabalhar em conjunto com o Editor) e um módulo que irá trabalhar a Qualidade de Serviço (QoS), auxiliando diretamente ao *Player*. O *Player* para sincronização multimídia irá utilizar a especificação multimídia baseada em Redes de Petri baseado no modelo HTSPN.

2. Sincronização Multimídia

A sincronização multimídia depende da especificação das relações temporais entre os objetos de mídias, que podem ser implícitas, como no caso da aquisição simultânea de voz e vídeo, ou podem ser formuladas explicitamente, como em documentos multimídia compostos de texto e voz (STEINMETZ, 1995). As relações temporais implícitas correspondem à sincronização intra-objeto (sincronização de mídias contínuas) e as explícitas correspondem à sincronização inter-objeto (sincronização de mídias com características diferentes, como texto e áudio).

BLAKOWSKI (1996) apresenta um modelo de referência de quatro camadas, como apresentado na Figura 1, útil para entender os requisitos e os mecanismos que devem oferecer suporte a execução da sincronização. Cada camada implementa mecanismos de sincronização que são dotados de interfaces apropriadas para especificar e reforçar relacionamentos temporais. Cada interface pode ser usada pela aplicação diretamente ou pela camada imediatamente superior. Camadas superiores apresentam abstrações de programação e de QoS mais elevadas.

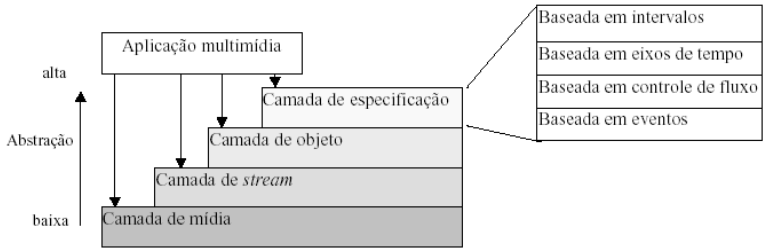

Figura 1: Modelo de referência para sincronização (BLAKOWSKI, 1996).

Nesse artigo, será dada ênfase à camada de Especificação, mais especificamente à Especificação Baseada em Controle de Fluxo com sincronização baseada em Redes de Petri utilizando o modelo HTSPN. A especificação baseada em controle de fluxo utilizando Redes de Petri foi escolhida para implementação do sistema por possuir algumas particularidades, como por exemplo:

- em algumas especificações existentes, como a baseada em intervalos, a sincronização multimídia pode ser trabalhada com relações binárias, ou seja, duas mídias podem ser sincronizadas entre elas através de várias formas existentes (*A before B*) (ALLEN, 1983); sendo possível, com Redes de Petri, trabalhar a sincronização com *n* objetos de mídias;
- na especificação baseada em eixos, os objetos de uma só mídia são ligados a um eixo de tempo que representa uma abstração do tempo real (STEINMETZ, 1995). Com esses eixos de tempo tem-se a impossibilidade de se definir o tempo destinado à interação com o usuário já que os eixos representam uma abstração real do tempo; além disso, caso a interação não ocorra, o sistema tem sua execução interrompida. Com Redes de Petri é possível estipular em tempo mínimo e máximo de interação do usuário, assim, caso essa interação não ocorra, o sistema terá sua execução continuada após se esgotar o tempo máximo.

A representação gráfica, juntamente com uma fácil modelagem de esquemas de sincronização e a possibilidade de analisar propriedades importantes fazem das Redes de Petri boas candidatas para modelagem de documentos multimídia e hipermídia (WILLRICH, 1997).

3. Especificação Multimídia Baseada em Redes de Petri

Nessa seção serão apresentadas algumas características dos modelos (OCPN, TSPN e HTSPN) de especificação multimídia baseada em Redes de Petri com o intuito de explicar a escolha do modelo HTSPN para implementação do *player*.

3.1. Object Compose Petri Net (OCPN)

O modelo OCPN tem como objetivo especificar formalmente os requisitos de sincronização de documentos multimídia, permitindo a representação da duração de recursos como sendo lugares da rede. As Redes de Petri temporizadas atribuem um tempo de duração de disparo para cada transição. No modelo OCPN, os lugares estão relacionados com a utilização de recursos e possuem um tempo de duração. Através da OCPN, é possível especificar as diversas relações temporais entre objetos multimídias (LITTLE, 1991), já que ela descreve efetivamente relações temporais do tipo "antes", "depois", "durante", "inicia junto", "termina junto" etc. Os recursos associados aos lugares da OCPN são relacionados aos tipos de objetos, tais como áudio, vídeo, texto etc (Figura 2).

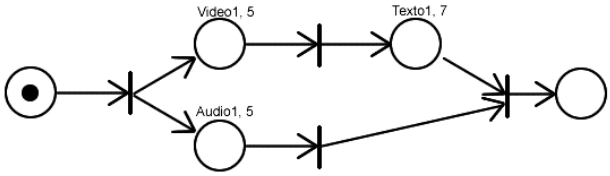

Figura 2: Exemplo de uma OCPN.

Na Figura 2, tem-se um exemplo de uma rede OCPN simples. Nesta ilustração, três mídias são executadas sendo que um vídeo e um áudio (ambos com duração de 5 tempos) são executados ao mesmo tempo (em paralelo); após a apresentação destes, inicia-se a apresentação de um texto que tem a duração de 7 tempos, tendo, então, uma duração total de 12 tempos

3.2 Time Stream Petri Net (TSPN)

O modelo OCPN tem como objetivo especificar formalmente os requisitos de sincronização de documentos multimídia, permitindo a representação da duração de recursos como sendo lugares da rede (LITTLE, 1990). Nesse mesmo sentido trabalha o TSPN (*Time Stream Petri Net*) acrescentando ao modelo uma tupla (x,n,y) em cada um dos arcos da rede e também um tipo de transição em cada uma das transições.

A tupla (x,n,y) corresponde ao tempo mínimo de apresentação, tempo ideal de apresentação e tempo mais atrasado de apresentação da mídia respectivamente. DIAZ (1983), define que a sincronização pode ser obtida através de nove tipos de transições, segundo o modelo: "*strong_or*", "*weak_and*", "*or*", "*and*", "*strong_master*", "*weak_master*", "*or_master*", "*and_master*". A estratégia *strong_or* especifica que a apresentação de uma das mídias é suficiente para que a apresentação seja considerada completa, desde que as demais mídias tenham sido apresentadas por um tempo mínimo especificado nos respectivos arcos. A estratégia *weak_and* só é considerada completa quando as apresentações de todas as cadeias são realizadas; enquanto a *master* necessita que somente a cadeia *master* esteja completamente apresentada para que o *token* seja

liberado. A estratégia *and* permite a liberação do *token* quando os arcos que chegam nessa transição tenham executado pelo menos o tempo mínimo estabelecido para cada um. A estratégia *or* passa o *token* quando uns dos dois arcos estejam finalizados.

3.3. Hierarchical Time Stream Petri Net (HTSPN)

As extensões do modelo TSPN levaram ao modelo HTSPN (*Hierarchical Time Stream Petri Net*) que permite modelar facilmente tanto a sincronização lógica quanto a temporal em sistemas hipermídia (SÉNAC, 1995). O modelo HTSPN permite a modelagem formal, com extensões temporais, dos três principais conceitos do modelo Dexter (HALASZ, 1994): os componentes atômicos; as ligações (*links*); e um componente composto. Estes conceitos são relevantes pela possibilidade de ser possível, através deles, especificar relações entre mídias, com a utilização ou não de *links* de interação com o usuário, em um hiperdocumento. Estes conceitos e sua utilização em Redes de Petri ficam claros quando (WILLRICH, 1997) descreve como o modelo HTSPN suporta cada um dos mesmos:

- um componente atômico é modelado por um arco com um intervalo de validade temporal (IVT) e um lugar do tipo atômico associado a um recurso. O IVT é uma tupla (α,η,β) em que α,η e β especificam, respectivamente, a duração mínima, ideal e máxima admissível do tratamento associado ao componente atômico para um lugar determinado;
- uma ligação (*link*) é modelada por um arco temporizado (Link1, T3), em que Link1, na Figura 4, é um lugar do tipo ligação (*link*);
- um componente composto é modelado por um lugar abstrato, chamado lugar composto, que representa uma sub-rede. O lugar C1, na Figura 4, é um exemplo de lugar composto.

A Figura 4 apresenta um exemplo de Rede de Petri utilizada para especificação de sincronização multimídia.

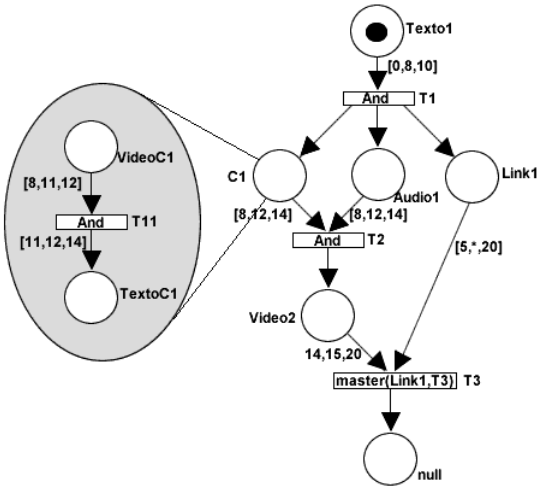

Figura 4: Modelagem dos componentes do modelo Dexter.

As apresentações de componentes atômicas são submetidas às durações nominais, intrínsecas ou dependentes de aplicação. Os últimos estão sujeitos a variações inesperadas, e conseqüentemente, uma técnica modelando com intervalos do tempo é recomendada (SÉNAC, 1995). Em HTSPN, um componente atômico é modelado usando um arco com um IVT, bem como um lugar do tipo atômico associado com um recurso.

Lugares do tipo composto são introduzidos no HTSPN para representar componentes compostos. Um arco de saída de um lugar composto especifica o IVT do componente composto. Um lugar composto é um lugar abstrato que especifica uma sub-rede chamadas *Structured TSPN* (STSPN). As regras de disparo do HTSPN trabalham da seguinte forma:

- quando um *token* entra num lugar composto, o lugar de entrada da sub-rede STSPN também é marcado;
- quando um *token* libera um lugar composto, todos os *tokens* relacionados à sub-rede são, recursivamente removidos

Um lugar composto é equivalente a uma STSPN relacionada a ele, tanto estrutural quanto temporalmente. Esta noção de equivalência temporal é introduzida com a ajuda do algoritmo de redução que possibilita reduzir qualquer STSPN a um arco temporizado. Qualquer arco de saída de um lugar do tipo composto é associado ao IVT obtido pela aplicação do algoritmo de redução no STSPN relacionado (SÉNAC, 1995).

Usando o modelo HTSPN, um link é modelado por um arco temporizado (L,t) e um lugar do tipo link. O IVT associado a um link introduz o conceito de link temporizado: links que são automaticamente disparados em função tanto da lógica quanto da condição temporal. Vale ressaltar que a duração normal de um evento assíncrono não pode ser conhecida antecipadamente. Então, a duração de um link é representada por "*".

O modelo HTSPN trouxe contribuições importantes para modelagem de sistemas hipermídia. Por exemplo:

- possibilita uma especificação fácil e formal dos conceitos fundamentais do modelo Dexter. Componentes (*i.e.*componentes *links*, compostos e atômicos) são uniformemente modelados como arcos temporizados e a definição recursiva de compostos tem um mapeamento imediato na hierarquia do modelo HTSPN;
- traz um melhor a entendimento de sincronização hipermídia por possibilitar que esquemas de sincronização que combinam a sincronização lógica e temporal sejam especificados. Além disso, o modelo HTSPN considera o não determinismo temporal em sistemas hipermídia distribuídos;
- com HTSPN é possível expressar como eventos assíncronos interrompem um cenário multimídia.

4. HTSPN com XML

A linguagem XML (*eXtensible Markup Language*) foi usada para a representação do modelo HTSPN. Através dela é possível estruturar o modelo de tal forma que uma especificação realizada em HTSPN seja facilmente intercambiável entre diferentes plataformas e aplicações. Com a linguagem XML, foi desenvolvido o DTD (*Document Type Definition*) *Petri-Net Scheduling Language* – PNSched (figura 7) que, para a modelagem de Redes de Petri, se utilizou dos elementos que permitem a especificação de lugares, arcos e transições, sendo que os arcos são representados através de *links* entre os lugares e as transições. A linguagem PNSched permite definir os elementos do modelo HTSPN e como os mesmos podem ser utilizados na especificação de uma aplicação multimídia com requisitos de sincronização.

```
<!ELEMENT net (place,(place+ | transition* | subnet*)?)+>
  <!ATTLIST net name ID #REQUIRED>
<!ELEMENT place (arc_place?,midia)>
  <!ATTLIST place name ID #REQUIRED
    type (atomic|compose|link) "atomic"
    mode (normal|final) "normal"
    target_subnet IDREF #IMPLIED >
<!ELEMENT arc_place (ivt)>
  <!ATTLIST arc_place target_transition IDREF #REQUIRED >
<!ELEMENT ivt EMPTY>
  <!ATTLIST ivt min CDATA #REQUIRED
    ideal CDATA #REQUIRED
    max CDATA #REQUIRED >
<!ELEMENT midia EMPTY>
  <!ATTLIST midia type (image | audio | video | text) "image"
            src CDATA #REQUIRED >
<!ELEMENT transition (arc_transition+, strategy) >
  <!ATTLIST transition name ID #REQUIRED >
<!ELEMENT  arc_transition EMPTY >
  <!ATTLIST arc_transition target_place IDREF #REQUIRED >
<!ELEMENT strategy EMPTY >
  <!ATTLIST strategy type
    (strong_or | weak_and | or | and | master | strong_master | weak_master |
     or_master | and_master) "and"
    target_place IDREF #IMPLIED>
<!ELEMENT subnet (subplace,(subplace+ | transition*)?)+>
  <!ATTLIST subnet name ID #REQUIRED>
<!ELEMENT subplace (arc_place?,midia)>
  <!ATTLIST subplace name ID #REQUIRED
    mode (normal|final) "normal">
```

Figura 7: DTD Petri-Net Scheduling Language (PNSched).

A linguagem PNSched (Figura 7), apresenta elementos para a representação do modelo HTSPN. Os elementos *place* e *transition* representam respectivamento os lugares e as transições. Cada um destes elementos possui arcos para a sua ligação, que são representados pelos elementos *arc_place* e *arc_transition*. A linguagem possui também um elemento para a representação das estratégias de sincronização. Esta é definida pelo elemento *strategy* que por sua vez possui um atributo *type* que armazena o tipo da estratégia que a transição possui.

A Figura 8 apresenta um exemplo de Rede de Petri utilizada para especificação de sincronização multimídia e a representação da mesma em XML utilizando a linguagem PNSched. O exemplo também é uma amostra de como o modelo HTSPN pode trabalhar com os conceitos do modelo Dexter. O exemplo apresenta um documento XML representando uma Rede de Petri, que foi construído através da linguagem PNSched. Cada elemento da linguagem é representado através de tags em um documento XML. A tag net, indica o início da modelagem de uma rede em um documento XML. A partir dela será representado todo o conteúdo de uma Rede de Petri. A tag place e trasition representam respectivamente os lugares e as transições. Cada uma delas possuem tags para representação dos seus arcos. Estes arcos são representados pelas tags arc_place e arc_transition. A diferença destes dois arcos é que o arc_place possui uma tag ivt, que armazena o tempo mínimo, ideal e máximo de execução de uma mídia. A tag subnet representa uma sub rede, ou seja, os lugares compostos contem uma outra rede e ela é representada pela tag subnet. Esta tag contém praticamente todos os elementos de uma rede comum, a diferença é que ela possui um novo elemento place que é definido pela tag subplace.

```
<?xml version='1.0'?>
<!DOCTYPE net SYSTEM "PNSched.dtd">
<net name = "n1">
    <place name = "Video" type = "atomic" >
    <arc_place target_transition = "t1">
      <ivt min = "3" ideal = "5" max = "7" />
    </arc_place>
    <midia type = "video" src = "c:\video1.mpeg" />
  </place>
  <transition name = "t1">
    <arc_transition target_place = "L" />
    <arc_transition target_place = "C" />
    <strategy type = "and" />
  </transition>
  <transition name = "t2">
    <arc_transition target_place = "p2" />
    <strategy type = "master" target_place = "L" />
  </transition>
  <place name="L" type="link">
   <arc_place target_transition = "t2">
      <ivt min = "5" ideal = "*" max = "16.5" />
    </arc_place>
    <midia type = "image" src = "c:\image1.gif" />
  </place>
  <place name = "C" type = "compose" target_subnet="sub1">
    <arc_place target_transition = "t2">
      <ivt min = "13" ideal = "14" max = "15" />
    </arc_place>
    <midia type = "video" src = "c:\videoC.avi" />
  </place>
  <subnet name="sub1">
       <subplace name="subp1" > ...
       <transition name = "subt1"> ...
       <subplace name="subp2" > ...
       <subplace name="subp3" > ...
       <transition name = "subt2"> ...
       <subplace name="subp4" mode = "final"> ...
  </subnet>
  <place name = "p2" type = "atomic" mode = "final">
    <midia type = "video" src = "c:\video2.avi" />
  </place>
  </net>
```

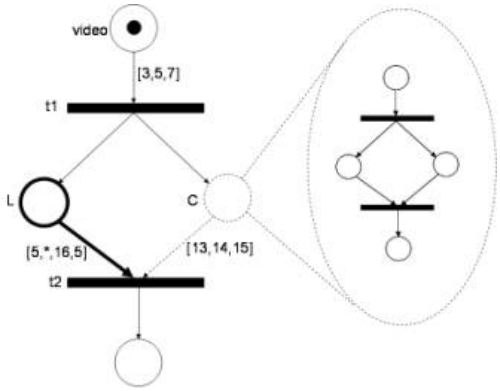

Figura 8: Representação de uma Rede de Petri em XML utilizando a linguagem PNSched.

5. Estrutura do *Player*

O *Player* que está sendo proposto neste artigo, como foi dito anteriormente, está inserido em um contexto maior de um sistema de sincronização multimídia. A função do *Player* (um dos módulos do sistema) consiste em traduzir o código XML gerado pelo Editor e que deverá ser validado pelo *Parser* (outros dois módulos do sistema).

O trabalho do Player consiste em traduzir o código PNSched (Petri-Net Scheduling Language) baseada em XML, gerado pelo Editor e validado pelo Parser em código JMF (Java Mídia Framework). JMF consiste em uma API (Aplication Program Interface) que possui um conjunto de classes e bibliotecas para trabalhar com aplicações multimídia (SUN, 2000). Para a análise dos arquivos gerados através da linguagem PNSched, será construído um Parser para verificar se o documento XML está bem formado, ou seja, será feita a

verificação do documento, para garantir que ele esteja de acordo com a estrutura definida. Na implementação do Parser será utilizada a linguagem Java e a API JAXB Java Architecture for XML Binding (SUN, 2003), que provê uma API e ferramentas que automatizam o mapeamento entre documentos XML e objetos Java. Com o uso da ferramenta JAXB, é possível gerar classes na linguagem Java e a documentação da mesma através da compilação de um documento XML Schema. Essas classes facilitam a implementação do Parser, pois possuem métodos para manipulação e validação de documentos XML. Na Figura 9 pode ser visto um esquema de como funciona o caminho percorrido pela Rede de Petri até chegar à construção da apresentação.

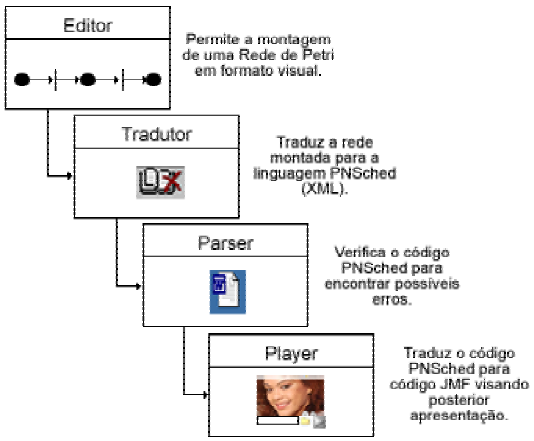

Figura 9: Caminho "percorrido" por uma Rede de Petri até a construção da apresentação pelo *Player.*

O caminho percorrido pela Rede de Petri apresentado na Figura 9 se dá da seguinte forma: o usuário irá montar uma rede no Editor gráfico, essa rede será traduzida para um código PNSched por um tradutor agregado ao Editor. Após isso, esse código PNSched deve ser validado pelo Parser antes de seguir para a apresentação no Player. O exemplo apresenta um código PNSched gerado a partir de uma parte de uma Rede de Petri e a forma como será o código JMF após a tradução feita pelo Player.

É apresentada uma rede 'net1' em que as mídias que estão nos lugares L2 e L3 deverão ter seu início no mesmo instante de tempo. Assim, tem-se o código PNSched gerado para a Rede de Petri utilizada no exemplo informando as características e o tempo de duração das mídias. Esse arquivo XML gerado com a linguagem PNSched será lido pelo *player* que irá traduzi-lo para código JMF. O código JMF correspondente apresenta os comandos *createPlayer()* para carregar as mídias a partir do caminho indicado e o comando *start()* para iniciar a execução das mídias. Para a geração dos códigos apresentados seguem-se dois passos: no primeiro (indicado na Figura 10 pelo número 1 ao lado da seta), o editor irá mapear Rede de Petri criada e a cada elemento encontrado irá ter seu código correspondente inserido no documento XML; no segundo passo (indicado na Figura 10 pelo número 2 ao lado da seta), o *player* irá fazer a leitura do documento XML gerado e deverá criar as classes em JMF para posterior execução como explicado na sub-seção anterior.

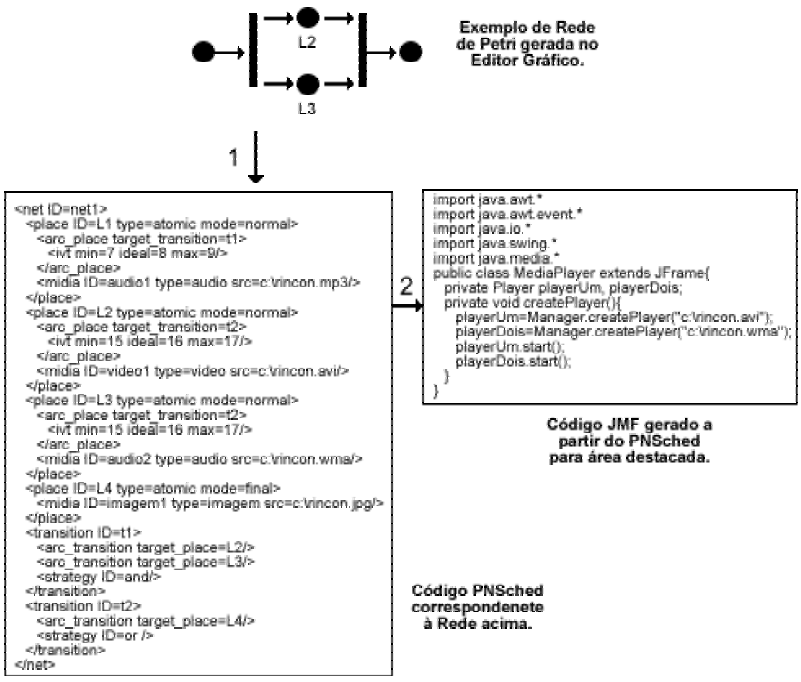

Figura 10: Exemplo de codificação de uma Rede de Petri em PSNched e JMF.

Assim, pode-se visualizar o Player como composto por dois módulos: o módulo de interpretação e tradução de código, descrito na subseção acima, e o módulo da apresentação. Enquanto que o primeiro módulo deverá prover a interpretação e tradução de um código PNSched (XML), o segundo módulo possui o trabalho de executar esse código. Tomando como referência o código gerado na Figura 11, o Player irá carregar as mídias a partir do local especificado passado por parâmetro através do comando createPlayer() e irá iniciar sua apresentação através do comando start(), neste caso, para as duas mídias.

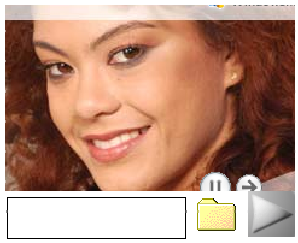

Figura 11: *Player* inicia a execução das duas mídias de forma paralela.

A Figura acima apresenta um exemplo de interface para o p*layer* em que o usuário poderá escolher o caminho da Rede de Petri que deverá ser executada e iniciar a execução através do botão *Play*. Vale salientar que essa interface consiste em um exemplo que possui a intenção apenas de mostrar a execução das mídias de áudio e vídeo, como mostrado no exemplo da Figura 11, e que o *Player* irá possuir outras funcionalidades que não se faz necessário apresentar no momento.

6. Considerações finais

A partir do estudo dos modelos de sincronização multimídia baseados em Rede de Petri, verificou-se que o modelo escolhido se apresentou adequado aos requisitos do sistema

no qual o *player* está inserido. Alguns requisitos propostos pelo sistema e que impulsionaram a escolha das Redes de Petri, entre outros, são: possuir uma forma de representação gráfica dos componentes que a compõem, permitir a sincronização de mais de dois objetos de mídia e, sobre outra vantagem que está presente no modelo HTSPN, a possibilidade de especificar a interação do usuário sem prejudicar a continuidade do sistema de forma corretamente sincronizada.

A implementação de um *player* permitirá verificar, na prática, como se dá a sincronização através das propriedades de HTSPN citadas acima. Além disso, este trabalho se apresenta como o início de um sistema que visa ter como resultados futuros: a verificação da possibilidade de se fazer a obtenção dos dados multimídia através do novo protocolo IPV6 e também verificar as possibilidades e técnicas de implementação para um ambiente WYSIWYG com sincronização temporal e espacial.

7. Referências Bibliográficas

ALLEN, J. F. Maintaining knowledge about temporal intervals. Communications of the ACM, 1983.

BLAKOWSKI, Gerold e STEINMETZ, Ralf. "A Media Synchronization Survey: Reference Model, Specification, and Case Studies," IEEE J. Select. Areas Commun., v. 14, 1996.

DIAZ, M; SCÉNAC, P. "Time stream Petri nets, a model for multimedia streams synchronization", proceedings of MultiMedia modeling'93, Singapura, novembro 1993.

HALASZ, F; SCHWARTZ, M. The Dexter Hypertext Reference Model. Communication of ACM 37(2), 1994.

Java Architecture for XML Binding (JAXB). Disponível em http://java.sun.com/xml/jaxb/. Acessado em 08/2003.

*Java Media Framework API***.** Disponível em http://www.java.sun.com/jmf. Acessado em 08/2003.

JENSEN, K. Coloured Petri Nets: A High Level Language for System Design and Analysis. Advances in Petri nets 1990. LNCS 483, Springer-Verlag, 1990.

LITTLE, T. D. C., GHAFOOR, A. Synchronization and storage models for multimedia objects. IEEE JSAC, v. 8, novembro 1990.

LITTLE, T.D.C. e GHAFOOR, A., "Multimedia Synchronization Protocols for Broadband Integrated Services", *IEEE J. Select. Areas in Commun*, vol. 9, dezembro de 1991.

SÉNAC, Patrick; SAQUI-SANNES, Pierre de; WILLRICH, Roberto. Hierarchical Time Stream Petri Net: a Model for Hypermedia Systems. 1995.

SOUZA, G. L. de F. "Sincronismo na Modelagem e Execução de Apresentações de Documentos Multimídia". Tese de doutorado. Departamento de Informática. Pontifícia Universidade Católica do Rio de Janeiro. Rio de Janeiro, 1997.

STEINMETZ, R., NAHRSTEDT, K. Resource Management in Networked Multimedia Systems. IEEE Computer, maio 1995.

WILLRICH, Roberto; SANNES, Pierre de Saqui. Concepção Formal de Aplicações Multimídia Java. In: XV SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES (SBRC'97), 1997, São Carlos. Anais do XV Simpósio Brasileiro de Redes de Computadores.