Roberta Lima Gomes

Autoria e Apresentação de Documentos Multimídia Adaptativos em Redes

Instituto de Matemática – Mestrado em Informática

Luiz Fernando Rust da Costa Carmo
Docteur, Université Paul Sabatier/LAAS - França - 1995

Luci Pirmez
D.Sc. - COPPE/UFRJ - Brasil –1996

Rio de Janeiro
2001

Autoria e Apresentação de Documentos Multimídia Adaptativos em Redes

Roberta Lima Gomes

Dissertação (Tese) submetida ao corpo docente do Instituto de Matemática da Universidade Federal do Rio de Janeiro - UFRJ, como parte dos requisitos necessários à obtenção do grau de Mestre.

Aprovada por:

_____

Prof. Luiz Fernando Rust da Costa Carmo,
Dr. UPS, France

_____

Profª. Luci Pirmez,
D.Sc. COPPE/UFRJ

_____

Prof. Oswaldo Vernet de Souza Pires,
D.Sc. COPPE/UFRJ

_____

Prof. José Gonçalves Pereira Filho,
D.Sc. USP

Rio de Janeiro - RJ
Agosto de 2001

FICHA CATALOGRÁFICA

GOMES, ROBERTA LIMA.

Autoria e Apresentação de Documentos Multimídia Adaptativos em Redes [Rio de Janeiro] 2001

xi, 85 p., 29,7 cm (IM/NCE/UFRJ, MSc., Informática, 2001)

Dissertação (Mestrado) - Universidade Federal do Rio de Janeiro, IM/NCE

1. Sistemas Multimídia 2. Redes 3. QoS

I. IM/NCE/UFRJ II. Título (série)

A meu noivo Magnos e minha família,

e

em memória do Prof. Júlio Salek Aude

e

do meu avô Aristides.

AGRADECIMENTOS

A Deus, pela minha vida.

Aos meus pais, que sempre me amaram e se empenharam em me educar, permitindo que eu chegasse até aqui ... e desde que parti, mesmo de longe, estiveram mais perto do que nunca.

A meus avós que, mesmo preocupados, apoiaram minha decisão e me ajudaram me dando muito amor e carinho.

A meu noivo Magnos que em todo momento esteve ao meu lado, me oferecendo seu amor, seu ombro e seu carinho, me dando forças para que eu não desistisse nos momentos de desânimo e cansaço, me fazendo rir por várias vezes em que chorei, e que sempre me fez acreditar que eu conseguiria vencer este desafio.

Aos amigos que fiz durante o curso de mestrado, Cesar, Eduardo, Paulo André, Renata, Reinaldo, Leonardo, Leila, Noel, Ana Paula, Queiroz, Sidney, Iuri, Cláudio, e às grandes amigas Andromeda, Patrícia, Adriana, Cynthia, Clarisse, Aninha, Jorgeana, Renatinha, dentre muitos, muitos outros, com os quais dividi vários momentos e que sempre estiveram por perto nas horas em que deles precisei.

Aos meus orientadores, Rust e Luci, pela determinação, cuidado e carinho com que orientaram o desenvolvimento deste trabalho, bem como o meu crescimento como pessoa, e aos professores do Mestrado em Informática do IM/NCE, pelos milhares de conhecimentos passados durante todo o curso.

Ao Núcleo de Computação Eletrônica da Universidade Federal do Rio de Janeiro, por toda infra-estrutura disponibilizada durante o curso para o desenvolvimento deste trabalho.

Resumo da Tese apresentada ao IM/NCE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

Autoria e Apresentação de Documentos Multimídia Adaptativos em Redes

Roberta Lima Gomes

Agosto/2001

Orientadores: Prof. Luiz Fernando Rust da Costa Carmo

Prof[a]. Luci Pirmez

Departamento: Informática

Os sistemas multimídia distribuídos empregam uma tecnologia aplicável em vários domínios como, em particular, os ambientes de ensino a distância. Considerando que os documentos multimídia são normalmente compostos de mídias contínuas, a apresentação destes documentos pode demandar uma grande quantidade de recursos devido às dimensões dos dados e às restrições de QoS das mídias. A disponibilidade de recursos em ambientes distribuídos no instante da apresentação é variável e imprevisível. Desta forma, os recursos disponíveis para a apresentação do documento multimídia podem ser insuficientes, resultando em uma degradação arbitrária da qualidade da apresentação. Uma forma de se tratar a escassez de recursos é tornar os documentos multimídia adaptativos. Este trabalho de pesquisa apresenta um formato flexível para a especificação de documentos multimídia adaptativos. É definida, também, uma arquitetura para a implementação de dois mecanismos de adaptação propostos. Sobre essa arquitetura são realizados alguns testes de desempenho para verificar a viabilidade da proposta de adaptação.

Abstract of Thesis presented to IM/NCE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

Authoring and Presentation of Adaptive Multimedia Documents over Networks

Roberta Lima Gomes

August/2001

Advisors: Luiz Fernando Rust da Costa Carmo, Dr, UPS
Luci Pirmez, D.Sc.

Department: Informatics

Multimedia presentations are applicable in various domains such as distance learning. Since multimedia documents may comprise continuous medias, the presentation of those documents may require vast system resources due to the huge amount of data and to their QoS restraints. The availability of resources in a distributed environment at presentation time is variable and potentially unpredictable. Hence, it can happen that there are not sufficient resources to render a multimedia document according to its specification, resulting in an arbitrary decrease of presentation's quality. To handle resource scarcity multimedia documents can be made adaptable to different resource availability. This research project presents a flexible format to the specification of adaptable multimedia documents. An architecture over which are implemented two proposed adaptation mechanisms is defined. Some performance tests are also performed in order to check the assessment of the adaptation proposal.

SUMÁRIO

Capítulo 1 Introdução 1

1.1 Sistemas Distribuídos Multimídia 1
1.1.1 Qualidade de Serviço 2
1.1.2 Adaptação em Sistemas Distribuídos 4

1.2 Objetivos 5

Capítulo 2 Documentos Multimídia 8

2.1 Modelos e Meta-Linguagens para Arquitetura de Documentos Multimídia 8
2.1.1 ODA 8
2.1.2 SGML 9
2.1.3 XML 10

2.2 Especificação de Documentos Multimídia 12
2.2.1 HyTime 12
2.2.2 MHEG 12
2.2.3 SMIL 14

2.3 Análise dos Requisitos do ServiMídia 15

2.4 Resumo 16

Capítulo 3 Documentos Multimídia Adaptativos 17

3.1 Adaptação em Sistemas Multimídia Distribuídos 17

3.2 Estratégia de Autoria de Documentos Multimídia Adaptativos 19

3.3 Arquivos de Controle 20
3.3.1 Especificação de Dependências Condicionais 21
3.3.2 A linguagem SCL 23

3.4 Comparação com Outras Abordagens Adaptativas 25

3.5 Resumo 26

Capítulo 4 Implementação e Execução da Estratégia de Adaptação 27

4.1 Metodologia de Adaptação 27
4.1.1 Execução dos Mecanismos de Adaptação 30
4.1.2 Reinício da Apresentação Adaptada 33

4.2 Arquitetura do Sistema 35
4.2.1 SPlayer 37
4.2.2 QoS Manager 37
4.2.3 QoS Agent 38
4.2.4 Adapter 39

4.3 Resumo 42

Capítulo 5 Avaliação da Estratégia de Adaptação 43

5.1 Análise do Atraso no Nível de Chaveamento de Fluxos 44
5.1.1 Caso 1: Adaptação Suave (mesma sessão RTP) 45
5.1.2 Caso 2: Adaptação Forte (sessões RTP diferentes) 46

5.2 Análise do Atraso no Nível de Documento (Adaptação forte) 47
5.2.1 Metodologia 48
5.2.2 Resultados 50

5.3 Análise dos Resultados 51

5.4 Resumo 52

Capítulo 6 Verificação Semântica das Informações de Adaptação 54

6.1 Consistência das Informações de Adaptação 54

6.2 Ferramenta de Verificação de Consistência de Documentos Adaptativos 57
6.2.1 Análise de Requisitos 57
6.2.2 Implementação da Ferramenta *SCL Semantics* 58
6.2.3 Interface da Ferramenta *SCL Semantics* 63

6.3 Resumo 66

Conclusões 67

Apêndice A DTD da Linguagem SCL 71

Apêndice B Módulo *Adapter* 72

Apêndice C Ferramenta *SCL Semantics* 76

Referências 81

Publicações da Autora 85

LISTA DE FIGURAS

Figura 3.1 – Arquitetura de restituição da informação 20
Figura 3.2 – Dependências condicionais entre objetos de mídia 22
Figure 3.3 – Representação de um objeto *link* 22
Figura 3.4 – Exemplo de documento SCL 23
Figura 3.5 – Documento SCL referente ao documento adaptativo da figura 3.2 25
Figura 4.1 – *Benefit function* de um vídeo 28
Figura 4.2 – Documento adaptativo (a e b) e documento adaptado (c) (1º exemplo) 31
Figura 4.3 – Dependências condicionais (1º exemplo) 31
Figura 4.4 – Documento adaptativo (2º exemplo) 32
Figura 4.5 – Dependências condicionais (2º exemplo) 32
Figura 4.6 – Arquivos SMIL adaptados (2º exemplo) 33
Figura 4.7 – Possíveis reinícios de uma apresentação após sua adaptação 34
Figura 4.8 – Mídia "B" substitui "A" ($d_B \neq d_A$) 35
Figura 4.9 – Principais módulos do sistema 36
Figura 4.10 – Conjunto de parâmetros de QoS 39
Figura 4.11 – Execução do processo de busca por *links* condicionais disparados 41
Figura 5.1 – Tempo de chaveamento para 5 rodadas distintas (adaptação suave) 45
Figura 5.2 – Tempo de chaveamento para 5 rodadas distintas (adaptação forte) 46
Figura 5.3 – Documentos adaptativos com diferentes números de *links* condicionais .. 48
Figura 5.4 – Atraso introduzido na adaptação forte pelo número de *links* condicionais 51
Figura 6.1 – Caminhos de *liks* distintos chegando a uma mesma mídia 55
Figura 6.2 – *Link* condicional nunca disparado 55
Figura 6.3 – *Link* condicional desativando mídia já apresentada, para $t_B < t_f \leq t_A$ 56
Figura 6.4 – Diagrama de estados para o escalonamento de uma apresentação 59
Figura 6.5 – Apresentação de um documento adaptativo 61
Figura 6.6 – Apresentação adaptada 61
Figura 6.7 – Agregação dos diagramas de estados 61
Figura 6.8 – Ferramenta *SCLSemantics*: diagrama de estados inicial 64
Figura 6.9 – Ferramenta *SCLSemantics*: caminhos representando adaptações 65
Figura 6.10 – Ferramenta *SCLSemantics*: caminhos representando adaptações 65

Figura 6.11 – Módulo complementar (diagrama de *links* condicionais) 66

LISTA DE TABELAS

Tabela 3.1 – Diferentes abordagens adaptativas e suas principais características. 26
Tabela 5.1 – Atrasos medidos durante a adaptação de três documentos com números distintos de *links* condicionais 50

Capítulo 1

Introdução

As redes de computadores, inicialmente, eram utilizadas para transmitir dados do tipo textual, independentes do tempo. Com o crescimento das redes de alta velocidade e das tecnologias multimídia (captura, armazenamento e apresentação), os sistemas multimídia distribuídos tornaram-se fundamentais, permitindo a execução de aplicações como vídeo-conferência e telefonia IP. No futuro, a tendência é que produtos para ensino a distância, simulação distribuída e grupos de trabalho distribuídos tornem-se bastante comuns.

1.1 Sistemas Distribuídos Multimídia

Atualmente, a multimídia é uma tecnologia indispensável na infraestrutura de comunicação de dados devido à sua grande fidelidade em representar a informação. Os documentos multimídia são compostos por objetos de mídia discreta, como imagens e textos, e contínua, como vídeos e áudios. A utilização de mídias contínuas torna o processo de apresentação de documentos multimídia em sistemas distribuídos uma tarefa complexa, exigindo uma disponibilidade bastante considerável de capacidade de transmissão e processamento.

De uma forma geral, as aplicações multimídia distribuídas apresentam três grandes dificuldades [24]: (i) se comparadas às aplicações textuais tradicionais, as aplicações multimídia exigem muito mais banda de transmissão; (ii) aplicações multimídia apresentam tráfegos com característica de tempo real, onde os dados (e.g. áudio e vídeo) devem ser apresentados continuamente na mesma taxa em que foram produzidos, limitando o atraso de transmissão, bem como a variação deste atraso; e (iii) os fluxos de dados multimídia, normalmente, apresentam rajadas, o que pode causar a

perda de dados devido à falta de espaço em memória (*buffers*) no sistema local ou nos pontos intermediários da rede.

Contrastando com as exigências por altas bandas de transmissão, tráfego de tempo real e tráfego em rajadas, o cenário atual apresenta as redes de computadores sendo compartilhadas por milhões de usuários, possuindo uma banda limitada com atrasos médios e disponibilidade de acesso imprevisíveis. Consequentemente, pode ocorrer escassez de recursos durante a apresentação de documentos multimídia distribuídos, ocasionando a danificação ou até mesmo a interrupção da apresentação.

A maneira de se resolver esta situação conflitante é um desafio que deve ser enfrentado pelos sistemas multimídia em redes. Portanto, uma questão muito importante a ser considerada para a transmissão de tráfego multimídia é a que diz respeito à qualidade de serviço (QoS), tornando-se uma exigência cada vez mais freqüente nas redes de computadores que suportam este tipo de tráfego.

1.1.1 Qualidade de Serviço

O termo qualidade de serviço (QoS) surgiu no campo das comunicações para que se tornasse possível a descrição de determinadas características técnicas de transmissões de dados. O modelo de referência OSI (*Open System Integration*) apresenta um certo número de parâmetros os quais descrevem a taxa e a confiabilidade da transmissão (desempenho, atraso médio, taxa de erro e probabilidade de falha, etc.). A utilização desses parâmetros ocorre nas camadas mais baixas da pilha de protocolos, não sendo acessados no nível de aplicação, o que leva a qualidade de serviço OSI a ser considerada incompleta.

Considerando que, atualmente, os dados com restrições temporais prevalecem nas aplicações multimídia, é importante que todo o sistema distribuído interaja para que os níveis de desempenho requisitados sejam garantidos. Durante a inicialização de uma aplicação, os requisitos de serviço definidos devem ser enviados para os outros componentes (nós) do sistema. Geralmente, o processo de negociação entre os nós

determina se os mesmos podem, como um todo, satisfazer o nível de qualidade de serviço requisitado.

Na literatura são encontradas várias definições de qualidade de serviço, como: "uma descrição quantitativa de quais serviços oferecidos pelo sistema, expressos como um conjunto de pares de parâmetros, satisfazem as necessidades da aplicação" [23]; ou "um conjunto de parâmetros que define as propriedades de fluxos de mídia" [40].

Apesar de não existir uma padronização, a qualidade de serviço oferecida por um recurso de um determinado sistema pode ser caracterizada de acordo com os seguintes critérios [29]:

(i) *Throughput*, ou largura de banda, é definido como um produto entre o tamanho ou número de unidades de dado e a taxa na qual as unidades são processadas.

(ii) *Delay*, ou atraso fim-a-fim, é o tempo gasto desde a geração de um dado até sua exibição.

(iii) *Jitter* é a variação média do atraso fim-a-fim entre unidades de dados de um mesmo fluxo de mídia.

(iv) *Reliability*, ou grau de confiabilidade, é caracterizada pela razão de pacotes perdidos e a razão de bits errados.

Em [32], é apresentada uma noção mais generalizada de qualidade de serviço: "qualidade de serviço representa o conjunto de características quantitativas e qualitativas de um sistema multimídia distribuído necessário para atingir a funcionalidade exigida por uma aplicação". Neste caso, "funcionalidade" pode ser entendida como a apresentação apropriada do dado multimídia ao usuário, ocasionando a satisfação geral do mesmo. No entanto, a maioria dos sistemas multimídia distribuídos trata a questão da qualidade de serviço do ponto de vista do sistema computacional, empregando políticas dinâmicas de gerência de recursos baseadas em mecanismos de monitoramento dos parâmetros técnicos de QoS [2].

Embora os parâmetros técnicos sejam úteis, eles possuem pouco significado para o usuário final. Considerando a perspectiva do usuário, a qualidade de serviço oferecida durante uma apresentação está associada à possibilidade de assimilar os conteúdos informacionais da mesma. Muitas vezes, a quantidade de recursos alocada a uma aplicação multimídia pode não interferir na percepção por parte do usuário [30].

Em [11], George Ghinea define Qualidade de Percepção (*Quality of Perception* – QoP) para representar a QoS do ponto de vista do usuário. Segundo Ghinea, QoP é um termo que abrange tanto a satisfação com relação à qualidade da apresentação, quanto a habilidade de analisar, sintetizar e assimilar o conteúdo informacional das mídias exibidas.

1.1.2 Adaptação em Sistemas Distribuídos

Considerando a variabilidade e a imprevisibilidade caracterizadas nos sistemas de rede, as apresentações multimídia executadas de forma distribuída podem não ter seus requisitos de QoP e de QoS respeitados. A reserva de recursos e as técnicas de admissão convencionais não são capazes de garantir uma qualidade de serviço sem um considerável desperdício de recursos e uma ineficiente utilização dos mesmos [9]. Há, portanto, uma necessidade latente de que os sistemas distribuídos apresentem uma natureza adaptativa, de forma a suavizar o impacto da escassez de recursos sobre o usuário final. Ao adotar mecanismos de adaptação, as aplicações passam a tolerar flutuações na disponibilidade de recursos, ou a infra-estrutura de rede torna-se capaz de moldar-se às mudanças dinâmicas nos requisitos das aplicações.

Várias propostas e implementações de sistemas levantam a questão da adaptabilidade para a transmissão de mídias em ambientes distribuídos. Cristina Aurrecoechea [2] descreve uma variedade de mecanismos de adaptação da qualidade de serviço. Por exemplo, filtros de QoS localizados em pontos específicos na rede que transcodificam os fluxos que passam por eles de acordo com a capacidade do *link* e os recursos do cliente (*Heidelberg QoS model*); mecanismos de adaptação da taxa de envio do emissor (*Closed-loop flow control*); e camadas incrementais de QoS enviadas por *multicast* (*Layered Multicast*). Já em [37], "Gerenciamento de QoS" (*QoS Management*)

é apresentado como um mecanismo adaptativo para a configuração e o controle dos recursos fim-a-fim. Esta adaptação é implementada através de um balanço e uma redistribuição dos recursos locais e de rede, a partir de notificações emitidas por mecanismos de monitoramento de QoS. Os elementos de rede devem, portanto, fornecer mecanismos como monitoramento de *buffers*. Entretanto, todos esses mecanismos se preocupam apenas com parâmetros de tráfego específicos e com a percepção direta do usuário com relação a mídias isoladas. A apresentação de um documento multimídia, considerando todos os seus fluxos de mídias distintas e suas relações semânticas, não é tratada como um todo.

O CMIF (*CWI Multimedia Interchange Format*) [3,4] define um modelo para as apresentações multimídia, permitindo a especificação de formatos alternativos de codificação para um mesmo objeto de mídia ou a definição de uma mídia alternativa. Uma vez efetuada a seleção do formato a ser apresentado, não será mais possível a interrupção da mídia devido à violação de seus parâmetros de qualidade de serviço (QoS). Neste caso, apenas o instante em que a transmissão da mídia é solicitada é considerado. Nenhuma adaptação é oferecida uma vez que o fluxo de mídia já tenha sido estabelecido.

A adaptabilidade a nível de apresentação também é considerada em [39], onde é apresentada uma proposta de extensão ao modelo temporal TIEMPO [38]. Nesse modelo é permitido especificar documentos multimídia flexíveis com alternativas de grupos de mídia (trechos da apresentação) a serem apresentados (informações redundantes). Um algoritmo de escalonamento adaptativo é executado em intervalos predeterminados de tempo, podendo haver uma sobrecarga de processamento.

1.2 Objetivos

O trabalho descrito nesta dissertação foi desenvolvido como parte do projeto ServiMídia [5], um projeto de pesquisa na área de redes de computadores e de ensino à distância. A autoria, o armazenamento e a recuperação de documentos multimídia fazem parte do escopo deste projeto. O objetivo básico está na necessidade de se acomodar em um mesmo ambiente multimídia de ensino (baseado no paradigma

cliente/servidor) usuários com diferentes perfis de sistema e recursos de comunicação. Dentro deste cenário, a qualidade da apresentação multimídia pode ser afetada de acordo com a quantidade de recursos disponíveis na rede, o que poderá interferir na qualidade de percepção verificada pelo usuário. Um mesmo documento, quando recuperado por diferentes usuários, pode ser interpretado de maneira completamente distinta.

Esse trabalho tem como proposta tornar as aplicações multimídia adaptativas, de forma a contornar, dinamicamente, o problema da escassez de recursos. Dentro dessa proposta, é definida uma estratégia de autoria e apresentação de documentos adaptativos baseada na especificação de arquivos de controle associados a documentos multimídia convencionais. Um arquivo de controle contém, portanto, as informações necessárias para a realização da adaptação de um documento multimídia. Para a especificação desse arquivo, foi criado o SCL (*SMIL Control Language*), uma linguagem descritiva baseada em XML. Os elementos do SCL permitem a especificação de parâmetros de QoS, assim como a definição de mídias alternativas e das condições para que as mesmas sejam ativadas durante uma adaptação.

Nesta estratégia descrita acima também são definidos os mecanismos de adaptação que, com base em informações de monitoramento, são executados durante a apresentação dos documentos. As condições do sistema no decorrer da apresentação multimídia e as informações de controle definem quando e como os mecanismos de adaptação devem ser executados. Caso a adaptação seja executada, a estrutura lógica da apresentação é alterada de forma a permitir que os usuários absorvam corretamente o conjunto de conceitos idealizados pelo autor durante a etapa de autoria, independentemente das condições do ambiente de comunicação.

Para a execução dos mecanismos de adaptação foi desenvolvida a arquitetura de um sistema de restituição seguindo o paradigma cliente-servidor. O *Adapter*, módulo desta arquitetura responsável pela reestruturação da apresentação multimídia em tempo real, foi implementado. Com objetivo de verificar a viabilidade da estratégia de adaptação foram, então, realizadas medidas de desempenho sobre esse módulo.

Durante o desenvolvimento da estratégia de adaptação, foi verificado que a autoria de documentos adaptativos poderia resultar em informações de adaptação inconsistentes. Para auxiliar o autor a identificar essas inconsistências foi desenvolvida uma ferramenta gráfica (*SCL Semantics*) que, com base em simulações, permite a verificação da consistência semântica dos documentos adaptativos.

Este trabalho está estruturado em sete capítulos. O capítulo 2 apresenta uma visão geral de alguns modelos para a padronização da arquitetura de documentos multimídia. Também são descritas algumas linguagens que utilizam conceitos e características apresentados por esses modelos. O capítulo 3 introduz alguns conceitos de adaptação em apresentações multimídia distribuídas. Além disso, o capítulo apresenta a estratégia de autoria de documentos adaptativos desenvolvida neste trabalho. O capítulo 4 apresenta a metodologia de adaptação e alguns exemplos ilustrando o processo de adaptação. Em seguida, o capítulo descreve a arquitetura do sistema de restituição desenvolvido. O capítulo 5 apresenta os testes executados com o objetivo de se avaliar a viabilidade da proposta deste trabalho. Neste capítulo são apresentados os resultados experimentais obtidos através de algumas medidas de desempenho realizadas no sistema implementado. O capítulo 6 apresenta questões relativas à semântica das informações de adaptação. Neste capítulo também é descrita uma ferramenta desenvolvida durante o trabalho para auxiliar a identificação de inconsistências semânticas em documentos adaptativos. Por fim, no capítulo 7 são apresentadas as conclusões finais do trabalho e são apontadas algumas sugestões de trabalhos futuros com o objetivo de dar continuidade ao Projeto ServiMídia.

Capítulo 2

Documentos Multimídia

Um documento multimídia é constituído por objetos de mídia de diferentes tipos apresentados em instantes de tempo distintos. A integração temporal das diferentes mídias é atingida através de relações bem definidas entre suas unidades de informação, denominadas relações de sincronização. Durante a etapa de autoria de um documento multimídia, além da especificação de seu comportamento temporal também são realizadas a estruturação lógica da apresentação (descrevendo abstrações para os objetos de mídia, assim como uma organização dessas abstrações) e a organização espacial entre os objetos de mídia (os objetos visuais são devidamente posicionados).

2.1 Modelos e Meta-Linguagens para Arquitetura de Documentos Multimídia

Nos últimos anos, pôde-se observar um grande esforço no sentido de padronizar a arquitetura de documentos multimídia. Foram criados padrões como ODA (*Office/Open Document Architecture*) e SGML (*Standard Generalized Markup Language*), o que disponibilizaria formatos neutros para o armazenamento de documentos.

2.1.1 ODA

ODA é um padrão internacional (ISO 8613) [15,25] para a representação da estrutura e do conteúdo de um documento. Ele define uma arquitetura abstrata que facilita o intercâmbio desses documentos, tornado-os independentes de plataforma. O padrão ODA organiza, de forma hierárquica (estrutura em árvore), a informação de um documento em três partes: (i) estrutura lógica; (ii) estrutura de *layout* – o posicionamento do conteúdo nas páginas; e (iii) conteúdo – as mídias efetivamente. A

estrutura lógica associa o conteúdo do documento a objetos lógicos, como capítulos, parágrafos e figuras, enquanto a estrutura de *layout* associa esse conteúdo a objetos físicos, como páginas e áreas. Para a apresentação do documento, é definido um mapeamento entre essas duas estruturas.

Grande parte dos modelos de estrutura utilizada atualmente baseia-se no modelo de estrutura em árvore de ODA. A estrutura em árvore é bastante indicada para a representação de documentos convencionais ou multimídia. No entanto, este método dificulta a representação de informações não-lineares entre esses objetos (e.g. objetos que podem ser acessados de várias formas, através de interatividade). Uma segunda fraqueza de ODA é a falta de um mecanismo para descrever relacionamentos temporais entre objetos.

2.1.2 SGML

O SGML [16] é uma meta-linguagem criada em 1986 como padrão internacional (ISO 8879) para a definição de linguagens de *markup*. As linguagens de *markup* permitem ao autor especificar um documento representando as informações de estrutura, apresentação e semântica juntamente ao conteúdo (HTML, ou *Hypertext Markup Language*, é um exemplo de linguagem de *markup*). O SGML descreve somente as propriedades formais e as inter-relações dos componentes de um documento, definindo o seu conteúdo de forma independente de seu processamento.

Um documento SGML é dividido em três partes: (i) a declaração SGML; (ii) o DTD (*Document Type Definition*); e (iii) a instância do documento. A declaração SGML define quais caracteres são utilizados no DTD e no texto do documento. A estrutura lógica do documento é definida em um DTD. O DTD especifica a gramática a qual indica os tipos de *markup*s e como elas se distinguem do texto convencional. A instância do documento contém o texto do documento, incluindo uma referência ao respectivo DTD.

Uma das desvantagens do SGML é não tratar o *layout* do documento, embora isso possa ser resolvido através de especificações DSSSL [13]. Um outro problema do

SGML (talvez um de seus maiores problemas) é sua complexidade. Por consequência, as ferramentas disponíveis para o tratamento de documentos SGML geralmente são caras e difíceis de serem utilizadas.

2.1.3 XML

O XML (*Extensible Markup Language*) [33] é uma *W3C Recommendation* resultante do trabalho de um grupo de especialistas estabelecido em 1996 pelo W3C com o objetivo de propor uma simplificação do SGML que fosse voltada às necessidades específicas da *Web*. Como o SGML era considerado muito complexo e o HTML, a aplicação mais popular do SGML, muito específico para representar informações de forma genérica, a linguagem XML ganhou destaque na *Web*.

XML é uma meta-linguagem de marcação (*meta-markup language*) que provê um formato para descrever dados estruturados, facilitando descrições mais precisas do conteúdo. Por ser uma meta-linguagem, o XML permite a definição de um número infinito de *tags* (marcas), provendo um sistema para criação dessas *tags* para dados estruturados. Desta forma ele provê uma representação estruturada dos dados que mostra-se amplamente implementável e fácil de ser desenvolvida.

A mais importante característica do XML é separar os dados estruturados de sua apresentação. Ele define o conteúdo do documento. No documento XML, as *tags* são utilizadas para descrever a semântica de um determinado elemento, e não a forma como ele deve ser apresentado. Isso assegura que os dados estruturados sejam uniformes e independentes de aplicações e fornecedores. Além disso, por ser um padrão flexível, aberto e independente de dispositivo, ele pode prover a capacidade de interoperabilidade entre sistemas heterogêneos. Portanto, o XML provê um padrão que pode codificar o conteúdo, as semânticas e as esquematizações para uma grande variedade de aplicações, garantido, por exemplo, resultados mais significativos de busca através de múltiplas plataformas.

O XML ainda conta com recursos como folhas de estilo definidas pelo *Extensible Style Language* (XSL) e *Cascading Style Sheets* (CSS) para a apresentação

dos dados. O XML separa os dados da apresentação, o que permite visualizar e processar o dado de várias formas, utilizando diferentes folhas de estilo e aplicações.

No XML, a definição de elementos e atributos bem como as regras para utilizá-los é realizada através de DTDs (*Document Type Definitions*). As regras definidas em um DTD ajudam a validar os dados, caso estes não estejam descritos internamente no próprio documento (DTDs podem estar incluídos dentro de documentos XML ou podem estar externos a eles). Portanto, um DTD XML permite que a aplicação verifique, por exemplo, que o usuário inseriu um subtítulo de terceiro nível sem antes ter inserido um de segundo nível. Ao contrário do SGML, o XML não exige que seja definido um DTD para um documento. Mas o DTD é necessário para que os dados de um documento sejam verificados e designados "dados XML válidos" (*valid*). Um analisador de documentos pode checar os dados analisando as regras contidas no DTD para ter certeza de que foram estruturados corretamente. Os dados enviados sem DTD, mas que seguem as regras do XML, são designados "dados XML bem formados" (*well formed*), mas não válidos. Com os dados XML válidos e bem formados, o documento XML se torna auto-descritivo pois as *tags* dão idéia do conteúdo e encontram-se em meio aos dados.

Devido ao formato do documento ser aberto e flexível, ele pode ser usado em qualquer lugar onde a troca ou transferência de informação é necessária. Um exemplo de utilização do XML é na descrição de informações sobre páginas HTML. O XML pode ser inserido dentro de documentos HTML, o que foi definido pelo W3C como *data-islands*. Esse recurso permite que um documento HTML possa ter múltiplas formas de visualização quando se faz uso da informação semântica contida no XML.

Apesar de ter sido inicialmente planejado para a especificação de documentos distribuídos pela *Web*, o XML tem sido amplamente adotado para permitir o intercâmbio de informações entre vários tipos de aplicações. Em particular, ele é visto como a melhor solução para o intercâmbio de meta-dados (e.g. *Open Software Description* [34]) e de informações comerciais (e.g. *Open Financial Exchange* [26]).

2.2 Especificação de Documentos Multimídia

Várias linguagens de descrição de apresentações multimídia têm sido criadas, algumas delas utilizando os conceitos e características apresentados pelos modelos descritos na sessão anterior. No entanto, não existe ainda um consenso indicando qual linguagem poderia se tornar um padrão. Algumas tentativas como o HyTime e o MHEG chegaram a tornar-se padrões internacionais (ISO), mas falharam em não considerarem a Internet como ambiente de armazenamento e recuperação dos documentos multimídia. No âmbito de sistemas distribuídos, o SMIL aparece como resultado do empenho do *W3Consortium* em definir recomendações e padrões para a área de multimídia e Internet.

2.2.1 HyTime

O HyTime (*Hypermedia/Time-based Structuring Language*) [14] é um padrão internacional para a representação de *links* de hipertexto e sincronização de informações estáticas ou baseadas no tempo. Ele foi definido como uma extensão do SGML, e resolve suas limitações fornecendo um modelo geral de *hyperlinks* (e sua sintaxe de representação) utilizando elementos SGML.

O HyTime aceita todos os tipos de tecnologias multimídia e hipertexto, uma vez que ele focaliza o intercâmbio da informação, e não sua apresentação (comportamento similar ao do SGML). Os objetos em um documento HyTime podem ser compostos por segmentos de áudio e vídeo, imagens, animações, gráficos, e até mesmo outros documentos HyTime.

Apesar de possuir características como interoperabilidade e fornecer um modelo bem elaborado de *hyperlinks* e sincronização, o HyTime não conseguiu atingir um público satisfatório, tornando-se pouco familiar para a maioria dos usuários.

2.2.2 MHEG

O modelo MHEG (*Multimedia and Hypermedia information coding Expert Group*) foi proposto por um comitê da ISO (ISO/IEC 13522-5) com o propósito de

descrever os relacionamentos (*links* condicionais e relações espaço-temporais) entre as diferentes partes de uma apresentação multimídia. Ele baseia-se no conceito de orientação a objeto e define uma variedade de classes a partir das quais são criados os objetos MHEG (*MH-objects*). Essas classes são utilizadas para descrever a maneira como um vídeo deve ser apresentado, um áudio é reproduzido, ou como o usuário pode interagir com a aplicação apresentada. As relações criadas entre as instâncias dessas classes constituem a estrutura da apresentação.

O MHEG é caracterizado pelos seguintes aspectos: (i) permite a especificação de apresentações de tempo real utilizando recursos de sincronização entre mídias baseado em objetos *Links*; (ii) permite interatividade através de objetos do tipo *Interactible*; (iii) pode ser implementado em ambientes distribuídos com uma exigência mínima de recursos de *buffering* e comunicação.

No MHEG-5 [17], uma das partes que compõem o padrão, foi desenvolvido um interpretador MHEG (*MHEG Engine*) que exige poucos recursos para a apresentação de aplicações multimídia. Desta forma, este interpretador pode ser executado em sistemas do tipo *set-top-boxes* (mecanismos utilizados em sistemas de TV interativa).

Uma das principais características do MHEG-5 é permitir a distribuição de aplicações multimídia em uma arquitetura do tipo cliente/servidor de diferentes plataformas. As aplicações são inicialmente armazenadas em servidores. À medida que as mesmas são apresentadas aos usuários, os *MH-objects* são enviados sob-demanda para seus respectivos terminais. Nos sistemas clientes são executadas as *engines* MHEG, responsáveis por interpretar e apresentar esses objetos, além de tratar as interações dos usuários.

Apesar do MHEG ter sido idealizado para ser executado em ambientes distribuídos, ele não se tornou um padrão *de facto* dentro do contexto da Internet. Atualmente, o grupo MHEG se preocupa em fornecer uma codificação de formato alternativa em XML para o MHEG-5, com o objetivo de promover o uso do MHEG pela comunidade *Web*. Esta nova parte do MHEG (MHEG-8 [20]) foi aprovada em 1999, e atualmente seu DTD XML está sendo desenvolvido.

2.2.3 SMIL

Atualmente, a *World Wide Web* enfrenta o problema de não conseguir "manejar" satisfatoriamente documentos contendo mídias contínuas como áudio e vídeo. Documentos HTML não conseguem expressar as primitivas de sincronização necessárias para a coordenação de partes independentes de dados contendo informações temporais. Em 1997, o W3C (*World Wide Web Consortium*) *Working Group* estabeleceu o SYMM (*Working Group on SYnchronized MultiMedia*) com o objetivo de estudar a definição de um formato multimídia declarativo para a *Web*. Neste formato, as interações de controle exigidas em aplicações multimídia deveriam estar codificadas em um arquivo texto sob a forma de um conjunto estruturado de relações de objetos.

O SMIL (*Synchronized Multimedia Integration Language*) [36] é uma linguagem baseada em XML que em 1998 foi submetida sob a forma de *W3C Recommendation* em sua versão 1.0 pelo SYMM. O objetivo do SMIL é fornecer um formato declarativo independente de plataforma para especificação de apresentações multimídia em ambientes distribuídos. Trata-se de uma linguagem de *markup* simples, porém poderosa. Do ponto de vista de arquitetura, o SMIL constitui um formato de integração. Ele não descreve o conteúdo de nenhuma das partes de um documento multimídia, mas ao contrário, define como vários componentes serão combinados considerando a dimensão espaço-temporal para a criação de uma apresentação. Esta linguagem não foi proposta para ser uma substituição de outros formatos individuais (como o HTML, AIF ou MPEG). SMIL utiliza os objetos de informação codificados nestes formatos e os combina em uma apresentação multimídia. Utilizando SMIL, um autor pode: (i) descrever o comportamento temporal da apresentação; (ii) descrever a disposição da apresentação em um *display*; e (iii) associar *hyperlinks* a objetos multimídia.

Em 1999, o W3C apresentou através de um *Working Draft* uma nova versão do SMIL, o SMIL Boston. Através do SMIL Boston seria possível o reuso da sintaxe e da semântica definida pelo SMIL 1.0 em outras linguagens baseadas em XML. Por exemplo, os componentes do SMIL Boston poderiam ser utilizados para integrar características de temporização ao XHTML [12]. Em setembro de 2000, o SMIL Boston

foi renomeado SMIL 2.0 [35] e sua especificação encontra-se atualmente como uma *Proposed Recommendation*. O SMIL 2.0 alterou uma pequena parte da sintaxe original do SMIL 1.0, mas por motivos de compatibilidade, o W3C estipulou que *players* SMIL (ferramentas que suportam o *playback* de documentos "application/smil") devem suportar tanto SMIL 1.0 quanto SMIL 2.0.

2.3 Análise dos Requisitos do ServiMídia

Como foi dito anteriormente, o Projeto ServiMídia propõe o desenvolvimento de aplicações multimídia adaptativas dentro do contexto de ensino à distância. Como esse tipo de aplicação geralmente é executado em ambientes *Web* (internet ou intranet), é interessante que a especificação dos documentos multimídia criados no ambiente ServiMídia seja realizada utilizando-se uma linguagem já difundida na comunidade WWW. Essa linguagem deve permitir o armazenamento e a apresentação desses documentos de forma distribuída. Um outro requisito para esta linguagem é que ela seja aberta, simples e flexível, permitindo a criação de extensões ou anotações contendo as informações de adaptabilidade.

Tanto o HyTime quanto o MHEG são padrões poderosos para a especificação de documentos multimídia. No entanto, eles apresentam alguns conceitos complexos e sintaxes de especificação que tornam a criação de documentos multimídia uma tarefa nada trivial para o autor. Por esse mesmo motivo eles não foram adotados dentro do domínio WWW. Além disso, a extensão desses padrões é pouco viável por serem padrões fechados.

A linguagem SMIL, por ser baseada em XML, é aberta e flexível. E da forma como foi definido o seu DTD, sua sintaxe ficou simples e intuitiva. Esses foram os principais motivos que levaram a comunidade WWW a adotá-la para a especificação de documentos multimídia de forma distribuída. Para a especificação de documentos multimídia dentro do ambiente ServiMídia foi adotado, portanto, o SMIL 1.0 (o SMIL 2.0 não foi escolhido por estar ainda em desenvolvimento).

O SMIL 1.0, originalmente, fornece algumas características de adaptabilidade, não exploradas por outras linguagens de especificação. Através da *tag* "<switch>", o autor do documento pode especificar um conjunto de elementos alternativos dentre os quais apenas um deve ser escolhido pelo *player*. Cada elemento dentro de um "<switch>" possui um conjunto de atributos de teste, como "system-bitrate", "system-language" e "system-screen-size". Um elemento é selecionado da seguinte forma: no instante da apresentação do documento, o *player* avalia os elementos (de acordo com seus atributos) na ordem em que aparecem no "<switch>"; o primeiro elemento aceitável é selecionado. Desta forma, os autores devem ordenar as alternativas da mais desejável para a menos desejável. É aconselhável que o último elemento do "<switch>" seja o mais livre de falhas, para que pelo menos um dos itens seja escolhido. A escolha dentro do elemento "<switch>" é realizada com base em variáveis estáticas que são configuradas nas ferramentas de apresentação de cada cliente.

Para os objetivos deste projeto, as características de adaptabilidade do SMIL apresentam as seguintes limitações: (i) escolha de um elemento é feita com base em variáveis estáticas configuradas pelo usuário; (ii) a escolha em si é estática, ocorrendo somente antes do início da apresentação; (iii) a escolha é realizada de forma independente em cada elemento "<switch>". Portanto, a especificação de uma adaptabilidade dinâmica para os documentos deve ser implementada através de outros recursos mais elaborados (através de extensões ao SMIL ou anotações).

2.4 Resumo

Este capítulo apresenta uma visão geral de modelos para a padronização da arquitetura de documentos multimídia. Também são descritas algumas linguagens que utilizam conceitos e características apresentados por esses modelos e abordam três aspectos fundamentais: estruturação lógica da apresentação, relações temporais e relações espaciais entre os objetos multimídia que a compõem. Dentre as linguagens, o SMIL foi escolhido para a especificação de documentos criados no ambiente ServiMídia. Os principais motivos para esta escolha foram a flexibilidade e a simplicidade do SMIL, além de ter sido desenvolvido para ser utilizado na *Web*.

Capítulo 3

Documentos Multimídia Adaptativos

À medida que os sistemas multimídia distribuídos (SMDs) tornam-se maiores e mais rápidos, o gerenciamento de recursos para o suporte adequado de tráfego multimídia torna-se mais complexo. A proposta deste trabalho é tentar resolver o problema da disponibilidade de recursos de comunicação durante uma apresentação multimídia tornando essa apresentação adaptativa. Como parte da proposta, é definido um formato flexível para especificação de documentos multimídia adaptativos, permitindo que o autor defina, através desta especificação, o comportamento da adaptação.

Neste capítulo é discutida a questão da adaptação em sistemas distribuídos. Em seguida é detalhada a estratégia de autoria adaptativa desenvolvida neste trabalho. Esta estratégia tem o objetivo principal de descrever formas alternativas para a apresentação do documento e os critérios de seleção, baseados em relacionamentos semânticos, que definem os momentos de ativá-las.

3.1 Adaptação em Sistemas Multimídia Distribuídos

Tipicamente, o gerenciamento de aplicações em SMDs exige que, antes da execução, uma aplicação determine seus requisitos de QoS para parâmetros como qualidade de imagem e som, tempo de resposta e grau de sincronização. Esses requisitos são, então, traduzidos em um conjunto de recursos de sistemas (e.g. banda de transmissão, memória, etc.) alocados para a aplicação. Um processo de negociação de QoS e uma infraestrutura de gerenciamento de QoS garantem que todos os componentes de sistema forneçam suas parcelas dos recursos requisitados durante o tempo de vida da sessão. Esse método funciona consideravelmente bem em ambientes homogêneos, onde

a alocação de recursos é realizada de forma direta e as aplicações não são muito complexas.

Para redes e aplicações mais complexas, o método descrito anteriormente passa a ser insuficiente. Os SMDs podem não atingir a negociação da QoS, ou quando conseguem, podem não ser capazes de manter os níveis de QoS negociados através de toda a rede por grandes períodos de tempo. Algumas razões para este fato são:

(i) muitos sistemas de redes, como a própria Internet, são do tipo *best-effort* (melhor esforço), tornando a reserva de recurso uma tarefa muito difícil de ser atingida;

(ii) SMDs são heterogêneos e envolvem vários tipos de componentes de comunicação e processamento;

(iii) a grande variação na carga das redes geralmente causa congestionamentos temporários, aumentando o atraso e a taxa de erro detectados; falhas transientes nos nós e reconfigurações da rede apresentam efeitos similares; o monitoramento dinâmico da QoS e a renegociação tornam-se necessários, resultando em um aumento na complexidade do sistema.

Para viabilizar o gerenciamento de recursos em sistemas complexos como SMDs, dois caminhos podem ser tomados [10]:

(i) utilizar técnicas de alocação de recursos, apesar do custo em termos de complexidade e *overhead*;

(ii) tornar as aplicações (e outros componentes de sistema) mais tolerantes às inevitáveis flutuações no desempenho do ambiente de suporte; neste caso é utilizada a noção de aplicações adaptativas, projetadas para lidar com variações de QoS de forma a amenizar seus efeitos sobre o usuário final.

Grande parte das aplicações atuais não é adaptativa ou apresenta uma adaptabilidade limitada. Neste último caso, as aplicações reagem a flutuações repentinas na QoS executando transições de estado “pesadas”, podendo apresentar um

comportamento imprevisível. Até hoje, nenhuma metodologia para projetar aplicações adaptativas foi estabelecida. No entanto, algumas soluções e várias implementações *ad hoc* já existem [2,3,4,38,39].

A abordagem de adaptação apresenta algumas vantagens. A primeira delas é que os processos de negociação de QoS e gerenciamento de recursos podem ser simplificados ou totalmente omitidos. Além disso, adaptabilidade é uma pré-condição necessária dentro de alguns contextos (e.g. ambientes móveis). Por último, sistemas adaptativos são resistentes às mudanças evolutivas dos ambientes de rede, possuindo maiores tempos de vida e, portanto, apresentando uma maior viabilidade econômica.

Em 1994, as conclusões resultantes do *Dagstuhl Seminar on the Fundamentals and Prespectives of Multimedia Systems* [8] demonstraram a significância deste problema. Seus participantes identificaram o seguinte problema como sendo um dos três grandes problemas na pesquisa de sistemas multimídia: "como adaptar as aplicações multimídia dinamicamente e continuamente aos seus ambientes de execução, fazendo com que as mesmas ofereçam os melhores serviços possíveis sob quaisquer conjuntos de condições".

3.2 Estratégia de Autoria de Documentos Multimídia Adaptativos

Em ambientes de ensino a distância, o autor deseja que um determinado conjunto de conceitos seja transferido para o aluno através de um conjunto de documentos multimídia. Este conjunto de conceitos é representado, no modelo mental, sob a forma de uma rede semântica. O processo de mapeamento, ilustrado na figura 3.1, ou a realização desta rede semântica em um conjunto de documentos multimídia faz parte da etapa de autoria destes documentos. Uma especificação flexível dos documentos permite que usuários com diferentes perfis de sistema possam absorver corretamente o conjunto de conceitos originais. As condições do sistema no momento da restituição dos documentos e de suas respectivas mídias determinam a estrutura lógica a ser apresentada ao usuário.

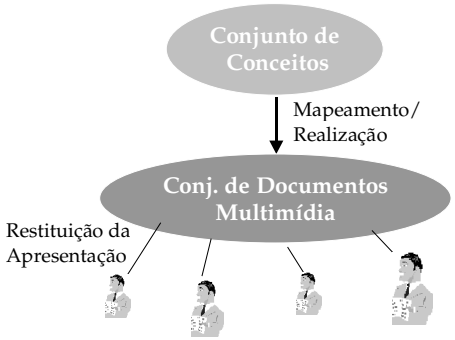

Figura 3.1 – Arquitetura de restituição da informação

Dentro da proposta apresentada neste trabalho, a especificação flexível de um documento multimídia é obtida através da criação de dois arquivos distintos. No primeiro arquivo, é especificada uma apresentação multimídia convencional utilizando-se como linguagem de especificação o SMIL 1.0 [36]. Este documento contém, então, a estrutura lógica ideal ou original da apresentação multimídia que seria executada em um sistema com todos os recursos necessários disponíveis.

O segundo arquivo (arquivo de controle ou anotação) é composto por informações de controle necessárias à realização da adaptação da apresentação. Para a especificação do arquivo de controle, foi definida, dentro deste trabalho, a linguagem *Smil Control Language* (SCL), baseada no XML [33].

3.3 Arquivos de Controle

No arquivo de controle são descritos os requisitos de QoS dos elementos do documento original e as alternativas para estes elementos. Também são especificadas as dependências condicionais entre esses objetos, o que permitirá a realização de uma reestruturação coerente no formato do documento durante sua adaptação (preservando a qualidade de percepção do documento).

A abordagem adotada (informações de controle externas ao arquivo SMIL) garante a interoperabilidade do ambiente com sistemas multimídia já existentes, além de simplificar o desenvolvimento de um *player* para apresentar os documentos adaptativos. Esta abordagem baseou-se no conceito de acesso universal (*Universal Multimedia*

Access – UMA) adotado pelo grupo de desenvolvimento do MPEG-7 [18,19]. O objetivo dos sistemas de acesso universal é criar diferentes apresentações da mesma informação para atender diferentes formatos, equipamentos e redes, a partir de uma mesma especificação (base de informação).

3.3.1 Especificação de Dependências Condicionais

A maioria das propostas de adaptabilidade trata os objetos multimídia de uma mesma apresentação de forma independente. Essas adaptações no nível de objeto, ou no nível de codificação, são limitadas à estrutura lógica estática do documento permitindo apenas a descrição de alternativas para os objetos de mídia. Além disso, da maneira como estas alternativas têm sido especificadas, a adaptação é executada simplesmente através de uma escolha entre o conjunto de alternativas de cada objeto antes do início da apresentação. Não é possível interromper outros objetos como parte deste processo de adaptação.

Para realizar uma adaptação no nível da aplicação, realizando uma reestruturação do documento como um todo, este trabalho propõe a especificação de dependências condicionais entre os diferentes objetos.

As dependências condicionais (ou relacionamentos semânticos) [6] foram propostas para que se pudesse explorar o conhecimento de relações semânticas entre os diversos objetos de mídia. As dependências condicionais expressam relações de causalidade entre os objetos que estão relacionados semanticamente, ajudando a descrever os requisitos de seleção de um objeto com relação a outros objetos. Desta forma, a apresentação de uma determinada mídia pode ser dependente da apresentação de outras mídias com as quais ela possui relações de causalidade. A especificação de dependências condicionais permite que durante a adaptação de um documento multimídia seja realizada uma reestruturação coerente no formato do documento.

Por exemplo, um objeto de áudio “A” descrevendo um vídeo “V” deve ser apresentado apenas enquanto este vídeo também for apresentado. Através da especificação de uma dependência condicional entre as duas mídias (figura 3.2), o autor

pode garantir que, no instante da apresentação, caso o vídeo seja interrompido devido a uma escassez de recursos no sistema (local ou de comunicação), o áudio também seja interrompido. Além disso, o autor também pode descrever mídias alternativas para "V" e "A". Como a figura 3.2 mostra, isto pode ser feito através da especificação de duas outras dependências condicionais: a primeira entre "V" e "An1" ("An1" deve ser apresentado somente se "V" for interrompido), e a segunda entre "An1" e "T2" ("T2" deve ser apresentado somente se "An1" também for apresentado). Portanto, a descrição dessas dependências condicionais permite que o processo de adaptação execute uma reorganização na estrutura do documento mantendo a semântica desejada pelo seu autor.

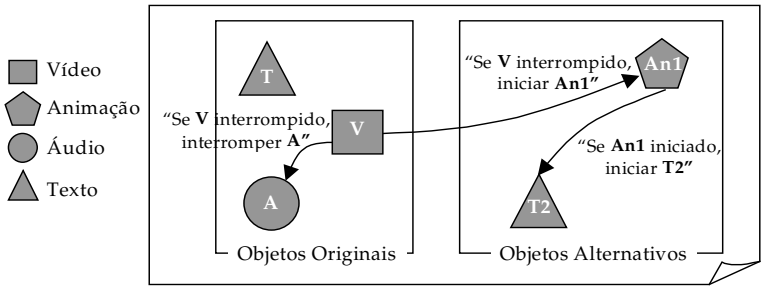

Figura 3.2 – Dependências condicionais entre objetos de mídia

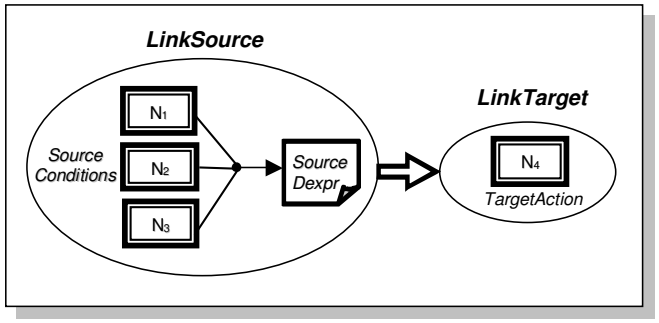

Figure 3.3 – Representação de um objeto *link*

Para a especificação de dependências condicionais, é utilizado o conceito de objeto *link* (*link* condicional). Como a figura 3.3 mostra, um *link* consiste de um *LinkSource* (origem do *link*) e um *LinkTarget* ("alvo" do *link*). O *LinkSource*, por sua vez, consiste de uma lista de *SourceConditions* (condições associadas a diferentes objetos de mídia "N_i"), as quais são combinadas através de uma expressão lógica definida no *SourceDexpr* (definindo a condição de disparo ou ativação do *link*). O

LinkTarget é formado por uma *TargetAction*, a qual é associada a apenas um objeto e é executada quando o *link* é disparado. Dentro da linguagem SCL são definidos dois tipos de *links*: *startlink* e *stoplink*, para apresentar e interromper um objeto, respectivamente.

Os *links* condicionais, da forma como estão definidos, são unidirecionais. Quando é detectada uma escassez de recursos no sistema é executada uma adaptação, realizando-se a reestruturação do documento (seguindo-se os *links* condicionais definidos pelo autor). Se essa escassez for temporária e o sistema retornar a seu estado inicial, voltando a haver recursos suficientes para a execução da apresentação original, não será possível realizar a adaptação no sentido inverso (i.e. reestruturar o documento seguindo os *links* condicionais no sentido contrário).

3.3.2 A linguagem SCL

SCL é uma linguagem de marcação baseada em XML que permite a especificação de dados estruturados para descrever as informações de controle de um documento adaptativo. Para a definição do SCL foi criado um DTD (apêndice A) onde são especificados todos os elementos da linguagem (*tags*¸ou marcas) e as regras de utilização destes elementos.

```
<scl>
  <controls>
    <description about="a2" restart="media">
      <minRequirements bw="8kbps">
      <maxRequirements bw="16kbps">
      <stoplink expr="(v2:stopped)">
    </description>
    ...
  </controls>
  <alternatives>
    <replace target="v2">
      <startlink expr="(v2:stopped)">
      <regionToAdd>
        <region id="rp3" title="rp3" ... />
      </regionToAdd>
      <resourceToSubstitute>
        <text id="t3" region="t3" ... />
      </resourceToSubstitute>
    </replace>
    ...
  </alternatives>
</scl>
```

Figura 3.4 – Exemplo de documento SCL

A figura 3.4 apresenta um exemplo de um arquivo de controle com o vocabulário da linguagem SCL. Basicamente, o arquivo SCL é dividido em dois grupos de elementos: “<controls>” e “<alternatives>”. O grupo “<controls>” é formado pelas informações de controle (e.g. parâmetros de QoS e dependências condicionais) relativas aos objetos de mídia presentes no arquivo SMIL original. Cada elemento “<description>” presente no grupo “<controls>” contém as informações de controle referentes ao respectivo objeto de mídia (indicado pelo atributo “about”). Um segundo atributo do elemento “<description>” é o “restart”, utilizado para indicar em que instante uma apresentação, após ser adaptada devido a uma falha de QoS na mídia indicada pelo “about”, deverá ser reiniciada. O atributo “restart” pode possuir os seguintes valores: (i) “document”, indicando o reinício a partir do início do documento; (ii) “media”, indicando o reinício a partir do início do elemento que substituiu a mídia que sofreu a falha de QoS (i.e. início da mídia indicada por “about”); ou (iii) “failure”, indicando o reinício do mesmo ponto em que ocorreu a falha de QoS.

Cada elemento “<description>” pode conter os seguintes elementos: (i) “<maxRequirements>” e “<minRequirements>”, especificando os requisitos máximos e mínimos para os parâmetros de QoS; e (ii) “<stoplink>” especifica um *link* condicional descrevendo sob quais condições o respectivo objeto de mídia deve ser removido da apresentação (o atributo “expr” possui a expressão lógica que deve ser satisfeita).

O grupo “<alternatives>” é formado pelas informações referentes às mídias alternativas que poderão substituir os objetos de mídia originais na ocorrência de uma adaptação. Cada elemento “<replace>” presente no grupo “<alternatives>” representa um objeto de mídia alternativo (ou um conjunto de objetos). O “<replace>” pode conter os seguintes elementos: (i) “<startlink>”, especifica um *link* condicional descrevendo sob quais condições a alternativa deve ser ativada (o atributo “expr” possui a expressão lógica que deve ser satisfeita); (ii) “<resourceToSubstitute>”, especifica o recurso (mídias ou um conjunto de mídia descritas em SMIL) que substituirá o elemento apontado pelo atributo “target” do respectivo “<replace>”; (iii) “<regionToAdd>”, especifica novas regiões no *layout* da apresentação adaptada.

Na figura 3.5 é ilustrada a utilização da linguagem SCL. Ela apresenta o documento SCL referente ao documento adaptativo da figura 3.2.

```
<scl>
  <controls>
    <description about="V" restart="media">
      <minRequirements bw="16kbps">
      <maxRequirements bw="64kbps">
    </description>
    <description about="A" restart="media">
      <minRequirements bw="8kbps">
      <maxRequirements bw="16kbps">
      <stoplink expr="(V:stopped)">
    </description>
  </controls>
  <alternatives>
    <replace target="V">
      <startlink expr="(V:stopped)">
      <regionToAdd>
        <region id="r-An1" title="rp3" ... />
      </regionToAdd>
      <resourceToSubstitute>
        <text id="An1" region="r-An1" ... />
      </resourceToSubstitute>
    </replace>
    <replace target="A">
      <startlink expr="(An1:started)">
      <resourceToSubstitute>
        <text id="T2" ... />
      </resourceToSubstitute>
    </replace>
  </alternatives>
</scl>
```

Figura 3.5 – Documento SCL referente ao documento adaptativo da figura 3.2

3.4 Comparação com Outras Abordagens Adaptativas

Como foi descrito na sessão 1.1.2, várias propostas e implementações de sistemas apresentam funcionalidades adaptativas para garantir a QoS de apresentações multimídia distribuídas. Na tabela 3.1 é possível observar as principais características de alguns sistemas e modelos adaptativos em comparação com a estratégia de adaptação desenvolvida neste trabalho.

	Descrição	Nível de Adaptação	Considera a QoP da Apresentação	Instante de Execução da Adaptação
***Heidelberg QoS model* [2]**	Filtros de QoS que transcodificam os fluxos em função da capacidade do *link* e os recursos do cliente.	Objeto	Não	No decorrer da transmissão.
***Closed-loop flow control* [2]**	Adaptação da taxa de envio do emissor.	Objeto	Não	No decorrer da transmissão.
***Layered Multicast* [2]**	Camadas incrementais de QoS enviadas por *multicast.*	Objeto	Não	No decorrer da transmissão.
***QoS Management* [37]**	Balanço e redistribuição dos recursos locais e de rede, a partir de notificações emitidas de monitoramento .	Objeto	Não	No decorrer da transmissão.
CMIF [3,4]	Modelo para apresentações multimídia, permitindo a especificação de formatos alternativos para as mídias.	Aplicação	Sim	Quando a transmissão da mídia é solicitada.
TIEMPO [38,39]	Documentos multimídia flexíveis com alternativas de grupos de mídia.	Aplicação	Sim	Execução em intervalos predeterminados de tempo (sobrecarga).
Projeto ServiMídia	**Descreve formas alternativas para a apresentação do documento e os critérios de seleção, baseados em relacionamentos semânticos.**	**Aplicação**	**Sim**	**No decorrer da apresentação, sinalizada por monitoramento.**

Tabela 3.1 – Diferentes abordagens adaptativas e suas principais características.

3.5 Resumo

Neste capítulo foram discutidas algumas questões referentes à adaptação de apresentações multimídia em ambientes distribuídos. Os conceitos básicos envolvidos com a adaptação de conteúdo também são apresentados. Também foi apresentada a estratégia de autoria de documentos adaptativos desenvolvida no Projeto ServiMídia, a qual define a especificação das informações de adaptação em um arquivo de controle, mantendo-as externas ao documento multimídia original.

Foi apresentada a linguagem SCL, definida para se implementar a estratégia de autoria. Através do SCL pode-se especificar o arquivo de controle que descreve as alternativas e os relacionamentos entre elas de maneira a realizar uma adaptação coerente da apresentação.

Capítulo 4

Implementação e Execução da Estratégia de Adaptação

Com o objetivo de suavizar o impacto das adaptações sobre os usuários e, ao mesmo tempo, garantir que a grande variedade de alternativas disponíveis para cada objeto possa ser utilizada, a metodologia de adaptação foi dividida em dois níveis. No primeiro nível a adaptação é executada em cada objeto. No segundo nível a adaptação é executada na aplicação como um todo. Os dois níveis de adaptação são implementados através de dois mecanismos que, por sua vez, são aplicados de acordo com a especificação do autor e com os níveis de qualidade verificados durante a apresentação de um documento.

Neste capítulo a metodologia de adaptação é descrita e exemplos ilustrando o processo de adaptação são apresentados. Em seguida é apresentada a arquitetura do sistema de recuperação desenvolvido no Projeto ServiMídia. Os módulos desse sistema são responsáveis por executar os dois níveis de adaptação descritos.

4.1 Metodologia de Adaptação

Para executar os dois níveis de adaptabilidade citados, dois tipos de mecanismos são implementados. A execução coordenada desses mecanismos visa garantir a consistência semântica desejada pelo autor do documento.

O primeiro mecanismo utilizado pelo sistema é denominado mecanismo suave (*soft*) e é aplicado sobre as mídias individualmente. Para a execução deste mecanismo, variantes de qualidade de serviço para uma determinada mídia devem ser definidas pelo autor de forma que a apresentação da mesma possa ser adaptada com base nestas variantes. Este mecanismo realiza uma adaptação no nível de codificação e,

efetivamente, executa uma seleção entre as diferentes codificações existentes para uma mesma mídia.

No instante de autoria, após definir as mídias que compõem o documento multimídia, as mesmas devem ser codificadas em formatos com resoluções ou qualidades distintas. A degradação do nível da qualidade de uma mídia deve ser tal que a permutação entre as variantes, durante a apresentação, ofereça um desconforto mínimo ao usuário final. No arquivo de controle, o autor especifica a faixa de variantes de QoS dentro da qual deve ser executada a adaptação suave. Para isso ele determina os valores máximos e mínimos para parâmetros técnicos como "bw" (*bandwith*, ou banda de transmissão), "lossrt" (*loss rate*, ou taxa de perda), "delay" (atraso médio) e "jitter" (variação do atraso) dentro dos elementos "<minRequirements>" e "<maxRequirements>" do SCL.

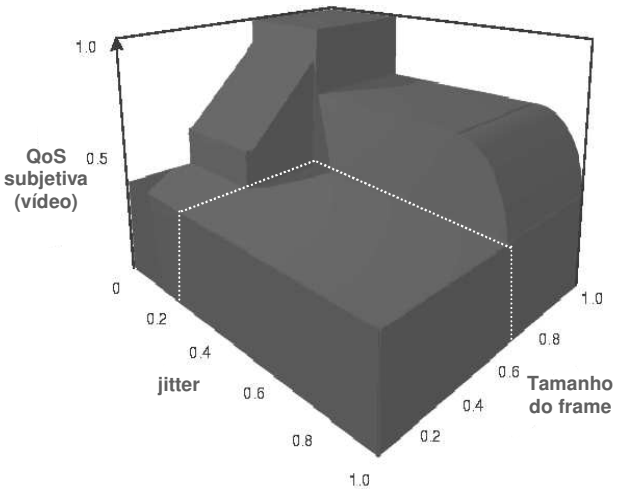

Figura 4.1 – *Benefit function* de um vídeo

Seria mais natural para o autor trabalhar com requisitos de QoS subjetivos da mídia [31] do que com requisitos de QoS objetivos (parâmetros técnicos de utilização de recursos). Uma solução é o estabelecimento de uma relação entre os dois tipos de requisitos de QoS. Esta relação pode ser qualificada através de uma expressão funcional, conhecida como *benefit function*, originalmente proposta em [7]. Um exemplo de uso da *benefit function* é a descrição do relacionamento entre *jitter* e tamanho de *frame* e a QoS subjetiva para um vídeo. Como a figura 4.1 mostra [7], uma

vez que os limites de *jitter* (0.2) e de tamanho de *frame* (0.7) são atingidos, o benefício resultante do uso de mais recursos (diminuindo o *jitter* ou aumentando o tamanho do *frame*) para aumentar a QoS subjetiva resultante é insignificante.

O mecanismo suave de adaptação baseia-se em uma política de monitoramento. Por definição, monitoramento é o processo de observar a utilização de recursos ou de características de QoS no sistema [37]. O processo de monitoramento é responsável por gerar mensagens que indiquem a ocorrência de violações de contrato de QoS. A adaptação realizada no primeiro nível ocorre sempre que o mecanismo de monitoramento detecta que os requisitos de QoS de uma variante sendo transmitida não estão sendo garantidos. Como resultado, o mecanismo de adaptação suave substitui a transmissão da variante de QoS corrente por uma variante que exija menos recursos de transmissão (e.g. um áudio codificado a 64kbps é substituído por uma versão de 32kbps).

A vantagem aparente deste tipo de mecanismo é a transparência com que uma adaptação é executada. O usuário percebe, de forma suave, a alteração do nível de qualidade de serviço. Além disso, trata-se de um mecanismo reversível. O processo de monitoramento, ao detectar uma sub-utilização dos recursos do sistema, pode gerar mensagens para que o mecanismo de adaptação modifique o fluxo de mídia para uma variante com melhor QoS. No entanto, essa reversibilidade pode causar um efeito bastante desagradável ao usuário caso as condições do sistema oscilem constantemente em um curto espaço de tempo. Neste caso, o mecanismo de adaptação executa chaveamentos frequentes entre variantes de uma mesma mídia. Para evitar esse efeito oscilatório, deve ser definido um tempo mínimo o qual o mecanismo de adaptação deve aguardar após executar um chaveamento entre duas variantes e então executar um novo chaveamento .

Se o mecanismo suave de adaptação, na ocorrência de escassez de recursos do sistema, ocasiona a apresentação da variante de menor QoS e, mesmo assim, a apresentação da respectiva mídia atinge um nível de qualidade abaixo do especificado pelo autor, esta mídia sofre uma falha de QoS (*QoS failure*) ativando o segundo mecanismo de adaptação.

O segundo mecanismo, denominado mecanismo forte (*hard*), é executado com base nas informações de adaptação (dependências condicionais) especificadas no arquivo de controle. De acordo com as dependências condicionais, esse mecanismo manipula a estrutura lógica do documento, gerando uma nova apresentação utilizando mídias com exigências de QoS menos rigorosas. As dependências condicionais exercem, portanto, um papel fundamental no processo de adaptação forte. Através das mesmas o autor pode garantir que a apresentação adaptada seja semanticamente equivalente à original. A apresentação adaptada é então convertida em um novo documento SMIL, e finalmente este documento é submetido e apresentado em substituição ao documento original.

A adaptação no nível de aplicação, executada pelo mecanismo forte, é uma técnica pouco referenciada na literatura. A maioria das propostas de adaptabilidade trata os objetos de uma mesma apresentação de forma independente. Essas adaptações no nível de objeto, ou nível de codificação, são limitadas à estrutura lógica estática do documento e proporcionam uma funcionalidade restrita.

4.1.1 Execução dos Mecanismos de Adaptação

Para ilustrar a execução dos dois mecanismos de adaptação, considere o documento multimídia descrito na figura 4.2a. No documento SMIL deste primeiro exemplo o autor define a apresentação de um vídeo (“V1”). No respectivo arquivo SCL (figura 4.2b), o autor descreve as seguintes informações necessárias para a realização dos dois níveis de adaptação: (i) a faixa definindo as variantes de QoS para a apresentação do vídeo “V1” (adaptação suave); (ii) as mídias alternativas (“anim1” e “text1”) e as condições para que as mesmas tornem-se ativas (adaptação forte). A figura 4.3 ilustra as dependências condicionais especificadas no arquivo SCL da figura 4.2b.

a) Arquivo SMIL original

```
<smil>
...
 <body>
   <video id="V1" region="r_v"...
 </body>
</smil>
```

b) Arquivo SCL

```
<scl>
  <controls>
    <description about="V1" restart="media">
      <minRequirements bw="64kbps" .../>
      <maxRequirements bw="128kbps" .../>
    </description>
  </controls>
  <alternatives>
    <replace target="V1">
      <startlink expr="(V1:stopped)">
      <regionToAdd>   ...
      </regionToAdd>
      <resourceToSubstitute>
        <par>
          <animation id="An1" ... />
          <text id="T1" ... />
        </par>
      </resourceToSubstitute>
    </replace>
  </alternatives>
</scl>
```

c) Arquivo SMIL adaptado

```
<smil>
...
 <body>
   <par>
     <animation id="An1" ... />
     <text id="T1" ... />
   </par>
 </body>
</smil>
```

Figura 4.2 – Documento adaptativo (a e b) e documento adaptado (c) (1º exemplo)

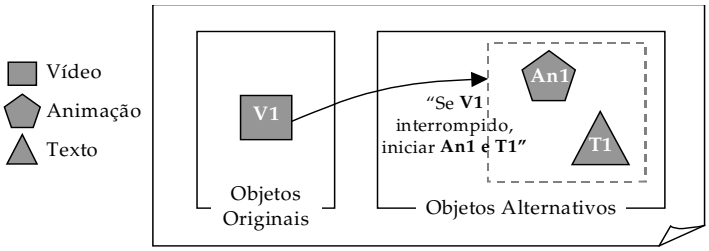

Figura 4.3 – Dependências condicionais (1º exemplo)

Sob congestionamento, a apresentação deste documento é, inicialmente, submetida à adaptação suave, implicando a diminuição progressiva da taxa de apresentação do vídeo "V1" até um mínimo especificado pelo autor. Se o mecanismo de monitoramento do cliente continuar detectando o congestionamento no fluxo de "V1" será ativada a adaptação forte, implicando (i) a interrupção da apresentação do documento, (ii) criação (em tempo real) de um novo documento SMIL (figura 4.2c) com base nas informações do arquivo SCL, e (iii) apresentação deste novo documento.

A figura 4.4a ilustra o documento SMIL referente a um segundo exemplo de aplicação. Neste documento o autor define a apresentação de um texto "T1" em paralelo a uma sequência de mídias (é apresentado o vídeo "V1" e em seguida são apresentadas duas mídias em paralelo, o vídeo "V2" e o áudio "A1"). No arquivo SCL da figura 4.4b foram especificadas as dependências condicionais, ilustradas na figura 4.5.

a) Arquivo SMIL original

```
<smil>
...
 <body>
   <par>
     <text id="T1" region=... />
     <seq>
       <video id="V1" region=... />
       <par>
         <video id="V2" region=... />
         <audio id="A1" region=... />
       </par>
     </seq>
   <par>
</body>
</smil>
```

b) Arquivo SCL

```
<scl>
  <controls>
    <description about="V1">
      <minRequirements bw="64kbps".../>
      <maxRequirements bw="128kbps" .../>
    </description>
    <description about="V2">
      <minRequirements bw="64kbps" .../>
      <stoplink expr="(V1:stopped)">
    </description>
    <description about="A1">
      <minRequirements bw="8kbps" .../>
    </description>
  </controls>
  <alternatives>
    <replace target="V1">
      <startlink expr="(V1:stopped)"/>
      <resourceToSubstitute>
        <animation id="An1" ... />
      </resourceToSubstitute>
    </replace>
    <replace target="V2">
      <startlink expr="(V2:stopped)"/>
      <resourceToSubstitute>
        <animation id="An2" ... />
      </resourceToSubstitute>
    </replace>
    <replace target="A1">
      <startlink expr="((A1:stopped)or(An2:started))"/>
      <resourceToSubstitute>
        <text id="T2" ... />
      </resourceToSubstitute>
    </replace>
  </alternatives>
</scl>
```

Figura 4.4 – Documento adaptativo (2º exemplo)

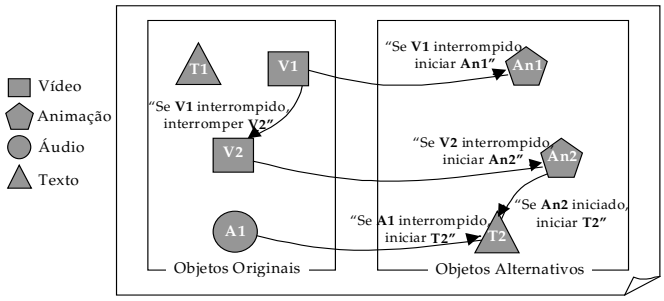

Figura 4.5 – Dependências condicionais (2º exemplo)

Inicialmente, a apresentação do documento é submetida à adaptação suave caso seja detectado algum congestionamento. Este processo ocorre de forma independente sobre cada fluxo de mídia. Se o mecanismo de monitoramento do cliente continuar detectando o congestionamento, ocorre a falha de QoS em uma das mídias sendo recebida. Portanto, pode-se observar que para esta aplicação existem três cenários de adaptação distintos, dependendo de que mídia sofre a falha de QoS:

(i) a primeira possibilidade é a adaptação disparada pela falha de QoS na mídia "V1", gerando uma nova apresentação especificada pelo documento SMIL da figura 4.6a;

(ii) a segunda possibilidade é a adaptação disparada pela falha de QoS na mídia "V2", gerando uma nova apresentação especificada pelo documento SMIL da figura 4.6b;

(iii) a terceira e última possibilidade é a adaptação disparada pela falha de QoS na mídia "A1", gerando uma nova apresentação especificada pelo documento SMIL da figura 4.6c;

Figura 4.6 – Arquivos SMIL adaptados (2º exemplo)

4.1.2 Reinício da Apresentação Adaptada

Durante o desenvolvimento da estratégia de adaptação, em particular, do mecanismo de adaptação forte, a seguinte questão foi levantada: após a criação do documento adaptado, em que instante deveria ser iniciada a apresentação do mesmo? Após algumas argumentações, três possibilidades foram levantadas. Como é mostrado a seguir, cada possibilidade pode ser adequada, ou não, dependendo da apresentação. Portando, foi decidido que o autor deve definir, através do atributo "restart" do SCL, o instante de reinício da apresentação adaptada. A figura 4.7 ilustra as três possibilidades (a mídia "B2" sofre falha de QoS e "D2" a substitui): (a) iniciar a apresentação a partir do início efetivo do documento; (b) iniciar a apresentação a partir do início da mídia que, no documento adaptado, substitui a mídia que sofreu a falha de QoS; ou (c) iniciar a apresentação a partir do mesmo instante em que ocorreu a falha de QoS.

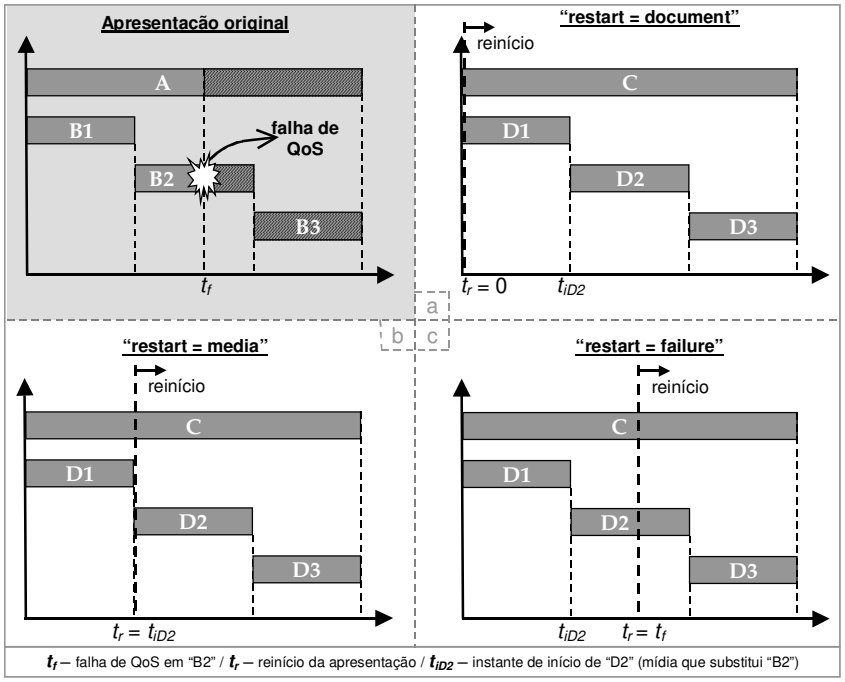

Figura 4.7 – Possíveis reinícios de uma apresentação após sua adaptação

O primeiro caso (figura 4.7a), onde a apresentação adaptada é executada desde o seu início, é o mais simples de ser implementado. No entanto este procedimento ocasionará a repetição de todo o conteúdo já apresentado ao usuário antes da ocorrência da falha de QoS. Por outro lado, este comportamento pode ser desejável na seguinte situação: a apresentação original é monolítica (apresenta um único "cenário") e o processo de adaptação altera todo o seu conteúdo.

No segundo caso (figura 4.7b), onde a apresentação é reiniciada a partir do início do elemento que substitui a mídia que sofreu a falha de QoS, também ocorrerá repetição de conteúdo já apresentado. Dependendo da duração d_{D2} da mídia "D2" e da parcela de d_{D2} já apresentada até a ocorrência da falha (e.g. a falha ocorre em 4min decorridos de uma mídia de 5min de duração) este comportamento pode ser pouco desejável. No entanto, a quantidade de informação reapresentada pode ser bem menor do que no primeiro caso. Além disso, o autor pode achar necessária a apresentação da mídia substituta desde seu início, caso as alterações nos tipos das mídias realizadas pelo processo de adaptação possam descontextualizar o usuário. Por exemplo, se a falha de QoS ocorre durante a apresentação de um áudio "A", e o mesmo é substituído por uma

seqüência de textos “seqT”, o autor pode preferir que “seqT” seja apresentado desde seu início.

No terceiro e último caso (figura 4.7c), o reinício da apresentação ocorre no mesmo instante t_f em que ocorreu a falha de QoS. Este comportamento pode ser visto como ideal uma vez que, após a adaptação, nenhuma informação é reapresentada ao usuário. No entanto, sua implementação apresenta algumas implicações referentes, principalmente, à sincronização desse reinício. Suponha que ocorra uma falha de QoS na mídia “A” (figura 4.8a) e que o autor tenha definido “B” sua mídia alternativa. Seja d_A a duração de “A” e d_B a duração de “B”. Se d_B=d_A, aparentemente, não haverá nenhum problema em reiniciar a apresentação em t_f. No entanto, mesmo que as duas mídias possuam a mesma duração, pode ser que um instante t qualquer de “A” não represente o mesmo instante semântico que o instante t de “B”. Isto pode ocorrer, principalmente, quando “A” e “B” representarem mídias de naturezas distintas (e.g. áudio e texto, ou vídeo e animação). O problema da sincronização torna-se mais aparente quando as durações de “A” e “B” são distintas. Se d_B<d_A, dependendo do valor de t_f, o fim de “B” (t_{fB}) pode ocorrer antes de t_f, como ilustrado na figura 4.8b. Neste caso, a ferramenta de apresentação pode sinalizar um erro no início da apresentação do documento adaptado. Na figura 4.8c, apesar de não ocorrer erro, é fácil perceber que se d_B>d_A, dificilmente t_f representa instantes semânticos equivalentes nas duas mídias.

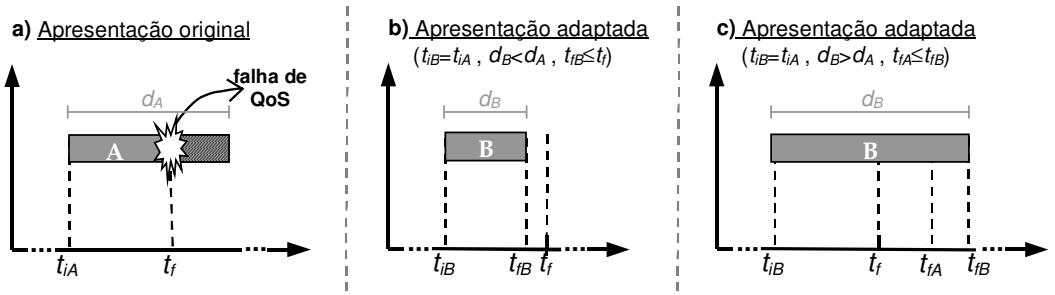

Figura 4.8 – Mídia “B” substitui “A” (d_B≠d_A)

4.2 Arquitetura do Sistema

Para a verificação da proposta apresentada neste trabalho, foi desenvolvida a arquitetura de um sistema para apresentação distribuída de documentos adaptativos

baseada no paradigma cliente-servidor. Os elementos dessa arquitetura são: (i) os servidores de mídia (*Media Servers*), responsáveis pelo armazenamento das mídias e dos documentos que fazem referência às mesmas; e (ii) os clientes (*Media Client*), responsáveis pela apresentação do documento multimídia, podendo apresentar diferentes tipos de características de sistema (terminal, acesso à rede, etc.).

Nesta arquitetura, os elementos de rede intermediários, presentes no caminho de transmissão entre o cliente e os servidores, são desconsiderados. Todo o controle dos parâmetros de QoS é efetuado nas pontas (*edges*) do sistema. Esta política foi adotada devido à sua simplicidade, se comparada às arquiteturas que dependem dos nós intermediários do sistema distribuído.

Na figura 4.9 são ilustrados os principais módulos do sistema, presentes no *Media Client* e no *Media Server*. Também são descritos os tipos de mensagens geradas pela comunicação entre esses módulos. O sistema foi desenvolvido utilizando o ambiente de desenvolvimento de aplicações JBuilder 3.0 em conjunto com a plataforma Java 2, o JMF 2.0 (*Java Media Framework 2.0* [22]) e o JAXP 1.0 (*Java API for XML Parsing* [21]).

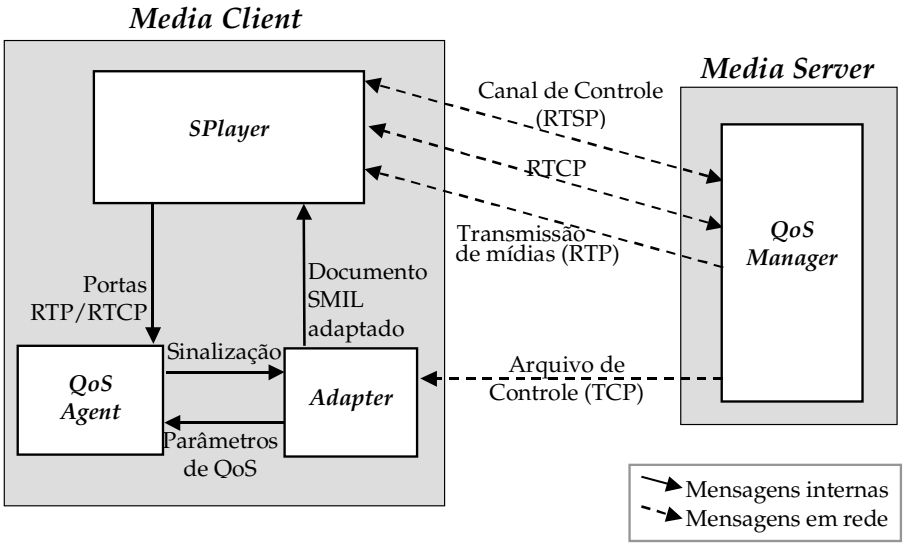

Figura 4.9 – Principais módulos do sistema

4.2.1 *SPlayer*

O *SPlayer* realiza a apresentação de documentos SMIL 1.0. Em um instante inicial, o usuário solicita através do *SPlayer* a apresentação de um documento multimídia localizado em um dos servidores. Após receber o documento SMIL, o *SPlayer* faz a solicitação aos servidores (via RTSP [28]) da transmissão (via RTP [27]) das mídias referenciadas pelo documento SMIL. Estas solicitações são realizadas sob-demanda, de acordo com o escalonamento da apresentação.

O *SPlayer* é uma ferramenta que comunica-se tanto com o *QoS Manager* quanto com qualquer servidor *RealServer* (ou outros servidores de mídia que implementem os protocolos RTSP e RTP/RTCP). Desta forma, no instante da autoria do documento, o autor não fica restrito às mídias armazenadas nos servidores do sistema.

4.2.2 *QoS Manager*

O primeiro papel deste módulo é transmitir para o *Adapter* o arquivo de controle associado ao documento SMIL. Quando um documento SMIL é solicitado por algum *player* (qualquer ferramenta de apresentação multimídia), o *QoS Manager* verifica se existe algum documento SCL associado ao mesmo (arquivo como mesmo nome, porém com a extensão ".scl"). Em caso positivo, este arquivo de controle é enviado (utilizando o protocolo TCP) através de uma porta conhecida. Caso o cliente seja um *SPlayer*, o *Adapter* recebe o arquivo, caso contrário este último é descartado.

O *QoS Manager* também é responsável por todas as sessões RTP/RTCP através das quais são transmitidas as mídias aos clientes. Ele realiza um monitoramento sobre esses fluxos de forma a controlar o chaveamento entre as variantes de QoS para um mesmo fluxo, executando a adaptação suave. Supondo, por exemplo, que existam três variantes de QoS para um mesmo vídeo (30 frames/s, 20 frames/s e 10 frames/s), o *QoS Manager* transmite, inicialmente, a variante de melhor qualidade. Se for detectado que os requisitos de QoS para a respectiva variante estão sendo violados, a variante de qualidade média passa a ser transmitida.

O controle do chaveamento entre as variantes de QoS é realizado com base nas informações intercambiadas nas sessões RTCP de cada fluxo de mídia. Em uma sessão RTP/RTCP, diferentes parâmetros de QoS relevantes são informados ao sistema. Nesta sessão, o fluxo de dados RTP é recebido por uma porta *n* e as informações concernentes ao controle da sessão são recebidas pela porta *n+1*. A RFC 1889 [27] define cinco tipos de pacotes RTCP: Receiver Report (RR); Sender Report (SR); Source DEScription; BYE; e um pacote especial APP, previsto para outras aplicações. Uma política de gerência de QoS é implementada com base nos parâmetros observáveis (definidos nos campos do protocolo RTCP) que permitem o monitoramento do *jitter*, da taxa de perda e do atraso ponto a ponto.

O *QoS Manager* apresenta uma base de informações composta pelas variantes de QoS pertencentes a cada mídia. Nesta base de informações, estão presentes os limites aceitáveis de *delay*, *jitter* e *loss rate* para cada variante. Durante a transmissão de um determinado fluxo RTP, há um controle dos parâmetros informados pelo receptor deste fluxo através dos pacotes RTCP tipo RR. Este controle é feito com base nos limites definidos para a variante de QoS que está sendo transmitida. A violação de algum desses limites implica na substituição da variante corrente por uma variante com requisitos de QoS menos exigentes, caso esta outra exista. Deve-se observar que se a variante transmitida for a de menor qualidade e, mesmo assim, houver uma violação de algum de seus parâmetros, o *QoS Manager* não realiza mais nenhuma ação.

4.2.3 *QoS Agent*

O *QoS Agent* tem como propósito o monitoramento (no sistema cliente) dos fluxos de mídia pertencentes a uma apresentação, mantendo para cada fluxo um estado sobre o qual é efetuado este monitoramento. Este módulo é responsável por controlar os estados de cada mídia ativa de uma apresentação. Para isso utiliza-se uma tabela com uma entrada para cada fluxo recebido pelo *SPlayer*. Em cada entrada gera-se um par de triplas ($jitter_{lim}$,$delay_{lim}$,$lossrt_{lim}$) ($jitter_s$,$delay_s$,$lossrt_s$): a primeira tripla indica os limites superiores (menor qualidade aceitável) para os parâmetros de QoS determinados pelo autor do documento multimídia (dados informados pelo *Adapter*); a segunda tripla informa o estado corrente dos parâmetros do respectivo fluxo de mídia.

Os dois conjuntos de parâmetros mantidos pelo *QoS Agent* podem ser representados em um espaço multidimensional (figura 4.10), onde cada eixo corresponde a um parâmetro. O conjunto que indica os limites para os parâmetros define um paralelepípedo neste espaço, dentro do qual o vetor de estado (o conjunto de parâmetros que informa o estado corrente do fluxo) pode mover-se. A extrapolação dos parâmetros de QoS é indicada pela presença desse vetor fora dos limites estabelecidos pela superfície do paralelepípedo, ocasionando a sinalização de uma falha de QoS.

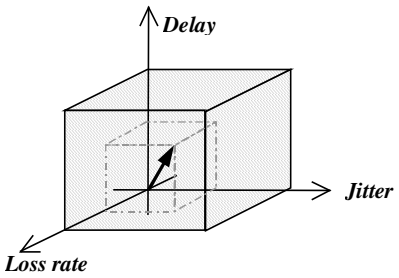

Figura 4.10 – Conjunto de parâmetros de QoS

A atualização do estado do fluxo de cada mídia, mantido pelo *QoS Agent*, é realizada sempre que um pacote RR é gerado. O *SPlayer* calcula os valores dos parâmetros de QoS e, ao mesmo tempo em que os transmite através das sessões RTCP, informa-os ao *QoS Agent*, de forma que o mesmo possa atualizar o estado do respectivo fluxo de mídia.

4.2.4 *Adapter*

Este módulo apresenta duas funcionalidades importantes dentro da arquitetura do sistema, descritas a seguir.

(i) Receber e interpretar o arquivo de controle associado ao respectivo documento multimídia, no instante inicial de uma apresentação. As informações de QoS obtidas deste arquivo são repassadas ao *QoS Agent*, que irá utilizá-las para o monitoramento dos fluxos de mídias estabelecidos pelo *SPlayer*.

(ii) Reestruturar um novo documento SMIL, ao ser sinalizado pelo *QoS Agent,* efetivando o segundo mecanismo de adaptação (adaptação forte). No instante em que o *QoS Agent* detecta a violação dos requisitos de QoS definidos para uma determinada mídia, ele informa ao *Adapter* o "id" da mídia e o instante de tempo em que ocorreu essa violação (valor relativo ao início da apresentação desta mídia). Baseado nas informações obtidas da especificação SCL, a estrutura lógica do documento SMIL é reformulada, gerando-se "*on-the-fly*" um novo documento SMIL referente a uma nova apresentação multimídia.

A construção do novo documento SMIL parte de alterações realizadas sobre o documento multimídia original. Essas alterações são definidas de acordo com os dados sinalizados pelo *QoS Agent* e com os relacionamentos semânticos especificados no arquivo SCL.

Na implementação do *Adapter*, foi utilizado o JAXP [21], uma API que oferece ferramentas de interpretação (*parsers*) e geração de saídas XML. A estrutura *Adapter* é dividida, basicamente, em dois sub-módulos: um tratador XML e um gerenciador. O tratador XML é responsável pela interpretação das *tags* descritas no arquivo de controle, bem como pela geração dos documentos SMIL adaptados. O sub-módulo gerenciador mantém os dados de controle extraídos destas *tags*. Este gerenciamento consiste na identificação das informações (associação dessas informações de controle com as mídias originais e com as mídias alternativas), e formatação das informações em parâmetros que são transferidos ao *QoS Agent*.

Inicialmente, o *Adapter* interpreta os dois arquivos XML (SMIL e SCL) e constrói, em memória, os modelos estruturados em árvore destes documentos (apêndice B). Quando o *Adapter* é sinalizado (por exemplo, a mídia "A" sofreu uma falha de QoS), ele inicia uma busca nos dois modelos estruturados com o objetivo de descobrir quais *links* condicionais devem ser disparados, gerando uma lista de eventos. O primeiro evento desta lista é, portanto, "*A:stopped*" (figura 4.11). Para cada iteração, ele verifica o *LinkSource* de cada *link* (*startlink*s e *stoplink*s) testando suas expressões lógicas com relação à lista de eventos (a expressão lógica define a condição de disparo de um *link*). Este processo é interrompido quando não há mais novos *link*s disparados, ou quando

todos os *links* especificados já foram disparados. A figura 4.11 ilustra um exemplo de execução do processo de busca por *links* condicionais disparados quando a mídia "A" sofre uma falha de QoS.

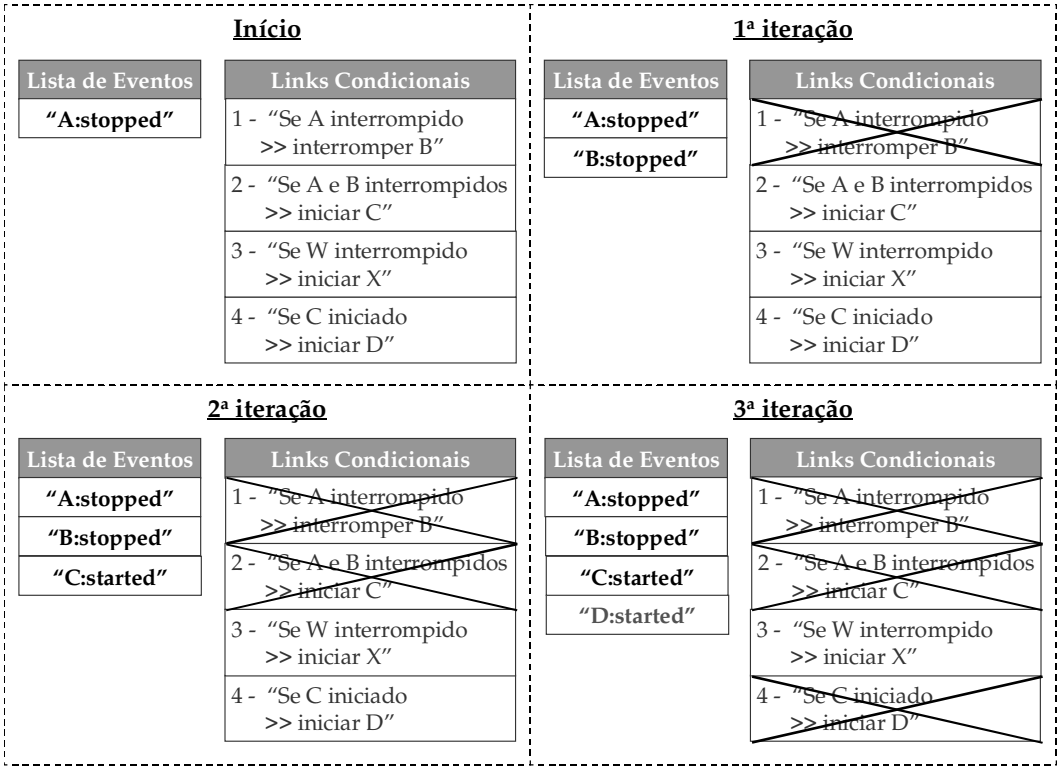

Figura 4.11 – Execução do processo de busca por *links* condicionais disparados

A lista final de eventos define quais mídias devem ser interrompidas (eventos do tipo "*X:stopped*") e quais mídias alternativas devem ser iniciadas (eventos do tipo "*Y:started*"). Baseado nestas informações, o *Adapter* constrói uma nova estrutura XML contendo o modelo de objetos da apresentação multimídia adaptada. A estrutura XML é então transcrita em um novo arquivo SMIL.

O novo documento SMIL é repassado ao *SPlayer*, que substitui a apresentação corrente pela nova apresentação. Neste momento, o *SPlayer* é informado do instante em que a apresentação deve ser reiniciada. Esse instante é calculado com base no atributo "restart" definido para a mídia que sofreu a falha de QoS. O usuário, já informado de que uma falha de QoS ocorreu, é avisado de que uma nova apresentação multimídia está

sendo preparada. Enquanto isso, os fluxos pertencentes à "antiga" apresentação são cancelados e os novos fluxos são solicitados aos servidores. A implementação atual do sistema suporta os três valores possíveis para o "restart" definidos no SCL. Mas para "restart=failure" existe a seguinte restrição: considerando *dm* a duração da mídia que sofreu a falha de QoS, e *dn* a duração da nova mídia que substituiu esta última, o parâmetro "restart=failure" só é válido se $dn \geq dm$, caso contrário é considerado o seu valor *default* definido para este atributo ("restart=media").

4.3 Resumo

Neste capítulo foi descrita a metodologia de adaptação utilizada durante a recuperação dos documentos multimídia. Dentro desta metodologia, são executados dois níveis de adaptação: adaptação no nível de objeto e adaptação no nível de aplicação. A execução coordenada dos dois níveis de adaptação garante a consistência semântica desejada pelo autor.

Também foi apresentada a arquitetura do sistema de recuperação desenvolvido durante o trabalho, onde foram descritos os módulos desse sistema que, de forma integrada, são responsáveis por executar os dois níveis de adaptação. Esta arquitetura segue o paradigma cliente-servidor, o que simplificou a implementação da mesma.

Capítulo 5

Avaliação da Estratégia de Adaptação

O tempo de resposta do sistema durante o processamento de adaptações em uma apresentação multimídia é uma característica importante para a verificação da viabilidade da estratégia de adaptação. Dentro do Projeto ServiMídia, como parte do trabalho apresentado em [41], foram realizados testes em um protótipo cliente-servidor para observar o desempenho dos mecanismos de adaptação no nível de chaveamento de fluxos. Os resultados destes testes mostraram que o chaveamento de fluxos (executado tanto durante a adaptação suave quanto durante a adaptação forte) introduzia um atraso bastante aceitável. No entanto, o chaveamento de fluxos ocorre somente após a substituição da apresentação original pela apresentação adaptada, representando apenas uma das etapas da adaptação forte. Portanto, para verificar o desempenho total do mecanismo de adaptação forte, também deve ser considerado o atraso introduzido durante o processamento e a geração do documento adaptado.

Neste capítulo são descritos os resultados experimentais apresentados em [41], que representam uma análise do atraso gerado no nível de chaveamento de fluxos. Este atraso é introduzido pelos mecanismos de adaptação quando o chaveamento entre *streams* de mídia é realizado para executar a adaptação suave (chaveamento entre variantes de QoS de uma mesma mídia) e a adaptação forte (chaveamento entre diferentes tipos de mídia resultante da substituição da apresentação original pela apresentação adaptada). Em seguida, é apresentada uma análise do atraso gerado no nível de documento, realizada através de medidas de desempenho do *Adapter* que, dentro da arquitetura descrita na sessão 4.2, é o módulo responsável pela criação em tempo de execução de documentos multimídia adaptados.

5.1 Análise do Atraso no Nível de Chaveamento de Fluxos

Como foi apresentado no capítulo 4, entre os módulos *SPlayer* e *QoS Manager* são estabelecidos os fluxos RTP/RTCP por onde são transmitidas as mídias de uma apresentação. Durante esta apresentação, a execução de adaptação suave ou forte resulta no chaveamento entre *streams* de mídia. No caso da adaptação suave, este chaveamento ocorre quando o *QoS Manager* substitui a variante de QoS sendo transmitida por uma outra variante (com menor ou maior QoS) de uma mesma mídia. No caso da adaptação forte, o chaveamento ocorre entre diferentes tipos de mídia no momento em que o *SPlayer* recebe o documento adaptado e prepara sua apresentação (este chaveamento reflete o cancelamento dos antigos fluxos de mídia e estabelecimento de novos fluxos).

Os testes apresentados em [41] se concentram na avaliação do tempo de resposta do sistema, mais especificamente do *SPlayer* e do *QoS Manager*, a uma solicitação de chaveamento de fluxos. Para a execução dos testes foi implementado um protótipo cliente-servidor representando esses dois módulos. O ambiente de testes consiste de duas estações (Pentium-II 250MHz, 128MB de memória, Windows NT Workstation 4.0) conectadas através de um barramento *Fast-Ethernet*, compartilhado por mais três outros computadores (um *gateway* e duas estações utilizadas para gerar tráfego através de aplicações *Web*).

Enquanto a aplicação servidora é responsável pela transmissão dos fluxos de vídeo, a aplicação cliente é responsável pela apresentação desses vídeos e pela solicitação da adaptação das mídias. Esta solicitação pode corresponder a um pedido de adaptação suave ou de adaptação forte. Para simular o comportamento do sistema no primeiro nível de adaptação, o cliente envia uma mensagem *SWITCH* ao servidor requisitando que este troque o vídeo corrente por outro, como se o novo vídeo representasse uma variante de QoS do primeiro. O servidor faz o chaveamento dos fluxos na mesma sessão RTP. Para simular o comportamento do sistema no segundo nível de adaptação, o cliente envia uma mensagem *SWITCH* para o servidor requisitando que ele inicie a transmissão de um outro vídeo no lugar do primeiro, como se o novo vídeo fosse uma mídia alternativa ao primeiro. Neste caso, o servidor cria uma nova sessão RTP para transmitir o novo fluxo.

Durante as simulações, a avaliação do tempo de resposta do sistema foi baseada na percepção do chaveamento dos fluxos por parte do usuário. Ou seja, considerou-se como tempo de resposta do sistema o intervalo de tempo entre a paralisação do primeiro vídeo na interface gráfica e o início da apresentação do segundo vídeo (eventos percebidos pelo usuário).

5.1.1 Caso 1: Adaptação Suave (mesma sessão RTP)

No primeiro caso, quando o chaveamento é solicitado pelo cliente (mensagem *SWITCH*), o servidor processa e transmite o novo vídeo na mesma sessão RTP. Para o cliente, tudo se passa como se o fluxo que ele está recebendo fosse o mesmo. Desta forma, o atraso de chaveamento foi medido através da análise do *log* gerado apenas no servidor. Neste arquivo, três eventos principais foram observados: "*Stopping processor 1*" (interrupção do objeto responsável pelo processamento do primeiro vídeo), "*Starting processor 2*" (criação e inicialização do objeto responsável pelo processamento do segundo vídeo) e "*NewSendStreamEvent*" (evento que indica o início da transmissão do segundo vídeo).

O gráfico apresentado na figura 5.1 mostra as curvas do tempo de chaveamento para 5 rodadas de simulação do primeiro nível de adaptação. O atraso médio calculado entre o primeiro e último evento é inferior a 1500ms.

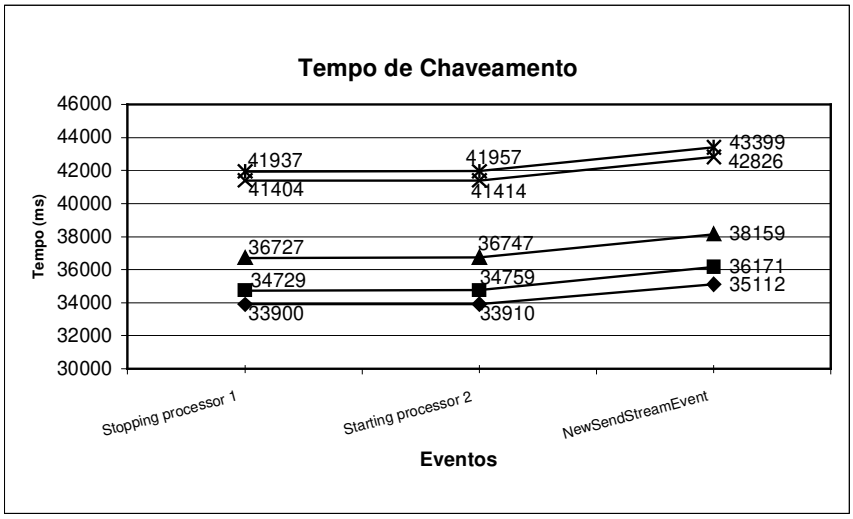

Figura 5.1 – Tempo de chaveamento para 5 rodadas distintas (adaptação suave)

5.1.2 Caso 2: Adaptação Forte (sessões RTP diferentes)

No segundo caso, quando o chaveamento é solicitado pelo cliente, o servidor estabelece uma nova conexão RTP para transmitir o novo vídeo. Ao detectar o novo fluxo na nova sessão RTP, o cliente cria um novo objeto (*Player*) para receber e apresentar os dados do segundo vídeo. Além disso, ele interrompe o objeto que recebia e apresentava o vídeo inicial. Portanto, o atraso inserido pelo cliente também deve ser considerado.

Os eventos registrados pelo cliente são: "*NewReceiveStreamEvent*" (detecção do novo fluxo de vídeo na nova sessão RTP), "*Creating a new player*" (criação de um novo objeto *Player*), "*KillOldPlayer*" (interrupção do objeto *Player* inicial) e "*StartEvent*" (início da apresentação do novo vídeo). A figura 5.2 ilustra o tempo de ocorrência dos eventos descritos (registrados no cliente e no servidor) para cinco rodadas de simulação do segundo nível de adaptação. Até o terceiro evento, os dois gráficos apresentam as mesmas características. No segundo gráfico, o tempo gasto até o terceiro evento responde por 2/3 do tempo total de chaveamento, representando o atraso introduzido pelo servidor. A outra parcela (1/3) do tempo total representa os eventos registrados pelo cliente. O atraso médio calculado neste teste foi de 2044ms.

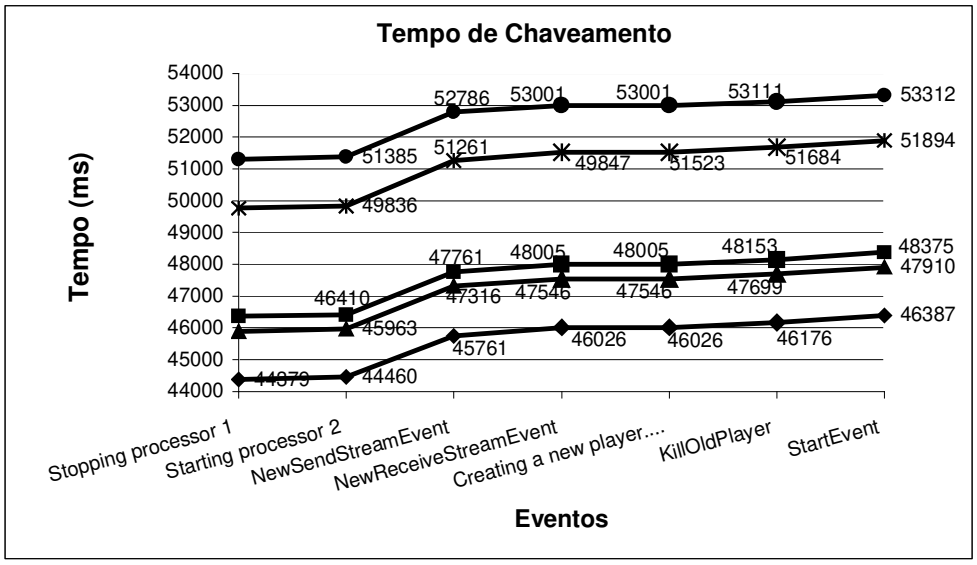

Figura 5.2 – Tempo de chaveamento para 5 rodadas distintas (adaptação forte)

5.2 Análise do Atraso no Nível de Documento (Adaptação forte)

Como descrito no capítulo 4, o módulo *Adapter* é o responsável por reestruturar a apresentação corrente e gerar um novo documento SMIL, executando a primeira etapa da adaptação forte. Quando o *QoS Agent* detecta alguma violação dos requisitos de QoS definidos para um determinado fluxo de mídia, ele sinaliza o *Adapter*. Baseado na informação contida no arquivo de controle, a estrutura lógica do documento é redefinida e um novo documento contendo a apresentação adaptada é construído em tempo real (primeira etapa). Em seguida, o documento adaptado é repassado ao *SPlayer* para que o mesmo possa substituir a apresentação (executando a segunda etapa da adaptação forte – chaveamento de fluxos).

Considerando que o *Adapter* deve (i) processar os arquivos SMIL e SCL (para extrair as informações de adaptação) (ii) realizar uma busca nas informações de controle, (iii) reestruturar a apresentação (iv) e construir um novo documento multimídia em tempo de execução, o atraso introduzido por este módulo pode ser bastante crítico. Para otimizar o processo de adaptação forte, o *Adapter* faz o *parsing* (interpretação) dos arquivos SMIL e SCL no início da apresentação, mantendo em memória as estruturas de dados referentes a estes arquivos até que alguma falha de QoS seja sinalizada. Apesar de haver uma utilização maior de memória, ganha-se em processamento durante a execução da adaptação.

A partir do momento em que é sinalizado, o *Adapter* realiza a devida busca nas estruturas em memória (como descrito na sessão 4.2.4) para descobrir quais mídias devem ser interrompidas e quais alternativas devem ser iniciadas. Em seguida, ele altera a estrutura da apresentação original, criando a apresentação adaptada. Para verificar o tempo de resposta durante a execução desse processo, foram realizadas medidas de desempenho no módulo *Adapter*.

O *Adapter* foi implementado em Java (mais detalhes no apêndice B), e os testes foram rodados em uma estação Pentium-II 250MHz com 128MB de memória e sistema operacional Windows NT Workstation 4.0. Basicamente, durante os testes foram executadas chamadas simulando falhas de QoS durante a apresentação de aplicações

testes. Com isso, foram medidos os intervalos entre cada chamada e o instante em que o *Adapter* concluía a criação de cada documento adaptado.

5.2.1 Metodologia

O desempenho do processo de adaptação realizado pelo *Adapter* está diretamente relacionado ao tamanho das estruturas de dados, obtidas a partir dos arquivos SMIL e SCL: inicialmente, é realizada uma busca pelos *links* condicionais a serem disparados (quanto maior o número de *links* definidos no documento SCL, maior o número de passos realizados durante a busca); com base no resultado dessa busca, a estrutura do documento original é alterada (o custo computacional depende do número de elementos especificados no documento SMIL original).

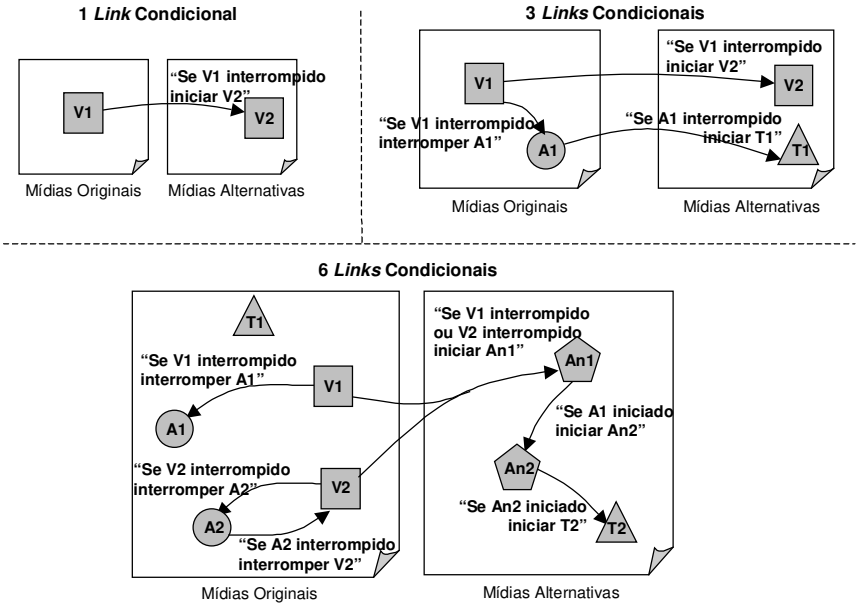

Figura 5.3 – Documentos adaptativos com diferentes números de *links* condicionais

Como a figura 5.3 mostra, à medida que o número de *links* condicionais aumenta, o documento adaptativo torna-se mais complexo. Portanto, para medir o desempenho do processo de adaptação executado pelo *Adapter*, foi criado um conjunto de documentos adaptativos (arquivos SMIL e SCL). Para cada documento adaptativo foi definido um número diferente de *links* condicionais, entre 1 e 28 *links*, representando as dependências condicionais entre os objetos de mídia. A apresentação de documentos

com número de *links* variados permite, portanto, a observação do desempenho do processo de adaptação em função da complexidade do documento adaptativo.

Após a autoria de vários documentos adaptativos, verificou-se que, quanto maior o documento multimídia, maior o número de *links* condicionais especificados. Desta forma, os documentos que possuem mais de 20 *links* condicionais tornam-se muito grandes e complexos. A criação de documentos com essa dimensão pode torna-se uma tarefa complicada para o autor, mesmo que seja utilizada alguma ferramenta de autoria. Nesse caso, apesar de não ser uma limitação da estratégia de adaptação, é aconselhável que o autor subdivida a apresentação em documentos que contenham até 20 *links* condicionais, para que o mesmo possa ter um maior controle e organização da apresentação. O limite superior no número de *links* condicionais para os testes foi adotado, então, com base nesse fato.

Durante a apresentação de cada documento adaptativo, foi simulada a falha de QoS de um de seus objetos de mídia com o objetivo de disparar o processo de adaptação forte. Para simular situações de pior caso, o objeto de mídia escolhido para sofrer a falha de QoS seria o que disparasse o maior número de *links* condicionais. Uma vez que o algoritmo de busca do *Adapter* é reiniciado sempre que um novo *link* condicional é disparado, a quantidade de *links* disparados durante a busca piora seu desempenho.

Para realizar as medidas de desempenho desejadas, é necessário medir o intervalo de tempo entre o evento de disparo da falha de QoS (evento "*QoS failure"*) e o evento representando o início da apresentação adaptada (evento "*Replay-Execution"*). Este último evento é sinalizado pelo *Adapter* quando o mesmo finaliza a criação do documento adaptado e envia uma mensagem ao *SPlayer* submetendo o novo arquivo SMIL. No entanto, durante o início dos testes, verificou-se que a duração da adaptação para um mesmo documento adaptativo variava bastante. Na verdade, isso ocorria principalmente devido ao ambiente de implementação e execução dos testes. A JVM (*Java Virtual Machine*) utilizada para executar os testes implementa um mecanismo de otimização baseado em *cache* para os códigos das classes: após as rodadas iniciais de um mesmo teste, a JVM passa a manter em memória o código das classes (e até de

objetos) utilizados na aplicação. Consequentemente, as últimas execuções de um teste apresentavam um desempenho melhor que as primeiras.

	2 *links* condicionais			**10 *links* condicionais**			**20 *links* condicionais**		
Nº de rodadas (i)	Atraso (A_i) (ms)	Média ($\bar{A}_i$)	DP (σ_i)	Atraso (A_i) (ms)	Média ($\bar{A}_i$)	DP (σ_i)	Atraso (A_i) (ms)	Média ($\bar{A}_i$)	DP (σ_i)
1	70	70.000	- - -	110	110.000	- - -	161	161.000	- - -
2	50	60.000	14.142	120	115.000	7.071	170	165.500	6.364
3	61	60.333	10.017	130	120.000	10.000	171	167.333	5.508
4	60	60.250	8.180	131	122.750	9.845	200	175.500	16.941
5	120	72.200	27.644	130	124.200	9.121	201	180.600	18.582
6	70	71.833	24.742	100	120.167	12.813	201	184.000	18.590
7	40	67.286	25.591	60	111.571	25.572	210	187.714	19.610
8	60	66.375	23.832	71	106.500	27.682	211	190.625	19.935
9	20	61.222	27.128	90	104.667	26.472	230	195.000	22.804
10	70	62.100	25.727	90	103.200	25.385	220	197.500	22.907
11	60	61.909	24.415	60	99.273	27.379	231	200.545	23.964
12	60	61.750	23.285	70	96.833	27.439	240	203.833	25.530
13	40	60.077	23.096	100	97.077	26.285	241	206.692	26.528
14	40	58.643	22.829	80	95.857	25.663	270	211.214	30.592
15	50	58.067	22.112	60	93.467	26.406	300	217.133	37.344
16	80	59.438	22.054	100	93.875	25.563	231	218.000	36.085
17	40	58.294	21.868	90	93.647	24.769	190	216.353	35.561
18	70	58.944	21.394	120	95.111	24.819	200	215.444	34.664
19	30	57.421	21.241	80	94.316	25.429	230	216.211	34.552
20	80	58.550	21.254	90	94.100	24.797	220	216.400	35.320

Tabela 5.1 – Atrasos medidos durante a adaptação de três documentos com números distintos de *links* condicionais

Para a obtenção de um valor médio confiável, foram medidos os atrasos durante diferentes execuções do processo de adaptação de três dos documentos criados (possuindo 2, 10 e 20 *links* condicionais). Para cada rodada i ($i=[1..20]$) foi obtida a medida do atraso A_i. Para cada conjunto $c_i = [A_1..A_i]$ foram calculados a média $\bar{A}_i$ e o desvio padrão σ_i. Na tabela 5.1 são mostrados os valores de A_i, $\bar{A}_i$ e σ_i obtidos. Pode-se notar que para $i \geq 15$ os valores de $\bar{A}_i$ e σ_i se estabilizam. Com base nesses resultados, as medidas de desempenho do processo de adaptação foram repetidas 15 vezes para cada documento adaptativo.

5.2.2 Resultados

A figura 5.4 mostra os valores médios e os desvios padrões do atraso introduzido pelo processo de adaptação de documentos possuindo diferentes números de

links condicionais. Como era previsto, o gráfico mostra que a complexidade do documento exerce uma influência considerável no desempenho do processo.

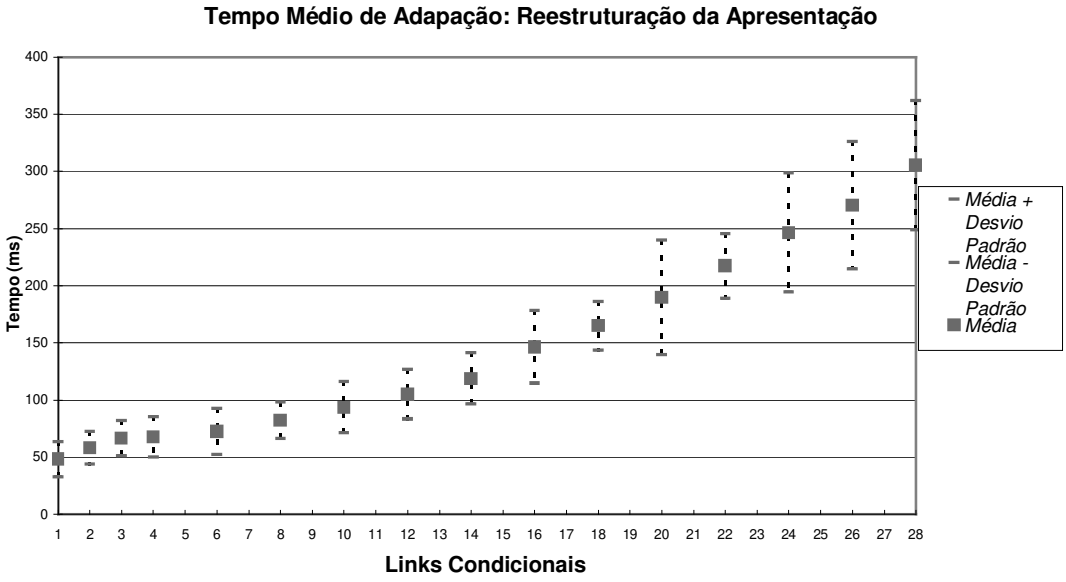

Figura 5.4 – Atraso introduzido na adaptação forte pelo número de *links* condicionais

Por consequência da grande variação dos valores medidos durante os testes, os desvios padrões calculados não são ideais, em um pior caso chegando a atingir 35% do valor da respectiva média calculada. Apesar dessa grande variação, todos os atrasos medidos foram inferiores a 400 ms. Se for considerado que existe uma expectativa de que o autor defina em torno de 20 *links* condicionais por documento, o desempenho constatado é ainda melhor. Neste caso os atrasos introduzidos permanecem inferiores a 250 ms. Estes resultados permitem a conclusão de que a geração, em tempo de execução, de novos documentos multimídia adaptados é viável.

5.3 Análise dos Resultados

Através destes resultados, pode-se notar que o atraso introduzido durante a adaptação suave é menor do que o atraso introduzido na adaptação forte. No entanto, uma vez que na adaptação suave é executado apenas um chaveamento de fluxos e, além disso, a aplicação cliente (*SPlayer*) não introduz nenhum atraso (este chaveamento, por

ser realizado dentro da mesma sessão RTP, é mais rápido do que o da adaptação forte), esperava-se que esta diferença fosse maior.

Durante a adaptação forte, o atraso total é gerado em duas etapas: (i) enquanto as informações de adaptação são processadas e um novo documento multimídia (contendo uma apresentação reestruturada) é gerado, e (ii) enquanto a apresentação original é substituída pela nova apresentação, o que implica no estabelecimento de novas conexões e no cancelamento da conexões inicialmente estabelecidas (chaveamento de fluxos utilizando sessões RTP distintas). Portanto, além do tempo gasto pelo servidor, também há o atraso introduzido pelo cliente para criar o documento e preparar o escalonamento da nova apresentação.

Apesar do atraso gerado na adaptação forte ser superior ao atraso gerado na adaptação suave, em um ambiente de ensino a distância, este atraso torna-se aceitável quando são consideradas as durações das mídias neste tipo de aplicação. Além disso, espera-se que a maioria das aplicações realize adaptações suaves com mais freqüência e que, somente quando estiverem esgotadas as possibilidades de adaptação no primeiro nível, uma adaptação mais severa seja realizada no documento (adaptação forte).

Um usuário habituado a assistir apresentações multimídia em rede geralmente se defronta com paralisações constantes e transmissões com baixa qualidade, o que muitas vezes inviabiliza o objetivo da apresentação. Portanto, quando se trata de aplicações de ensino, pode ser melhor experimentar atrasos de adaptação do que haver uma probabilidade de se prejudicar a interpretação do conteúdo informacional dos documentos. Nos testes realizados neste trabalho, os usuários concordaram que um intervalo de 2000 a 2500ms entre a paralisação e a retomada da apresentação é aceitável, principalmente quando a continuidade da apresentação é garantida e a semântica inicial do documento é preservada.

5.4 Resumo

Neste capítulo foram apresentados os testes realizados como o objetivo de avaliar a viabilidade da implementação dos componentes responsáveis pelos

mecanismos de adaptação. Na primeira parte dos testes, realizados e apresentados em [41], foi realizada uma análise do atraso gerado no nível de chaveamento de fluxos. Este atraso é verificado tanto na adaptação suave (chaveamento entre variantes de QoS de uma mesma mídia) quanto na adaptação forte (chaveamento entre diferentes tipos de mídia resultante da substituição da apresentação original pela apresentação adaptada).

Na segunda parte dos testes foi realizada uma análise do atraso gerado no nível de documento. O objetivo foi verificar a viabilidade da criação, em tempo real, de documentos multimídia adaptados. As medidas de desempenho foram realizadas no módulo *Adapter*, responsável pela reestruturação da apresentação e criação do documento adaptado. Documentos multimídia com números diferentes de *links* condicionais foram criados e, durante a apresentação de cada documento, foram simuladas falhas de QoS para disparar o processo de adaptação.

Por último os resultados dos testes foram analisados em conjunto, onde foi verificada a viabilidade da estratégia de adaptação.

Capítulo 6

Verificação Semântica das Informações de Adaptação

A criação de um arquivo de controle, separando as informações de adaptação do documento multimídia, permite que a autoria de documentos adaptativos seja dividida em duas etapas. Na primeira etapa o autor cria uma apresentação SMIL convencional (apresentação original) utilizando, ou não, uma ferramenta gráfica de autoria de apresentações multimídia. Na segunda etapa o autor define as mídias alternativas e os *links* condicionais entre elas e entre as mídias originais. A partir dessas informações, é criado o arquivo de controle utilizando a linguagem SCL.

A etapa de especificação do arquivo de controle é crítica, uma vez que nela será definido todo o comportamento adaptativo do documento. Deve ser realizada, portanto, uma verificação sintática e semântica do arquivo de controle resultante desta etapa para garantir que o processo de adaptação ocorra como desejado. A verificação sintática pode ser executada através de um *parser* XML convencional. No entanto, isso não garante a correção semântica das informações de adaptação.

Neste capítulo são discutidas algumas questões concernentes à semântica das informações de adaptação e são apontadas as principais inconsistências semânticas que podem ser definidas durante a autoria de documentos adaptativos. Também é descrita uma ferramenta de verificação de documentos adaptativos que, com base em simulações, auxilia o autor a identificar essas inconsistências.

6.1 Consistência das Informações de Adaptação

Após a especificação do SCL, verificou-se que a flexibilidade apresentada pela linguagem permite ao autor especificar informações inconsistentes no arquivo de

controle. O esquema adotado de definição de *links* condicionais entre pares (ou grupos) de mídias permite que ocorra o encadeamento desses *links*, o que, em alguns casos, pode representar algum tipo de inconsistência semântica nas informações de adaptação. Essas inconsistências podem causar um comportamento indesejável dos mecanismos de adaptação (mais especificamente, do mecanismo de adaptação forte). As seguintes inconsistências podem ser apontadas e detectadas através de uma ferramenta de verificação.

(i) O encadeamento dos *links* condicionais definidos entre as mídias originais e as mídias alternativas poderá gerar caminhos de *liks* distintos chegando a uma mesma mídia, como ilustrado na figura 6.1. Isto pode apresentar duas consequências: a ativação e desativação de uma mesma mídia alternativa (figura 6.1a) ou interrupção/ativação de uma mídia já interrompida/iniciada (figura 6.1b).

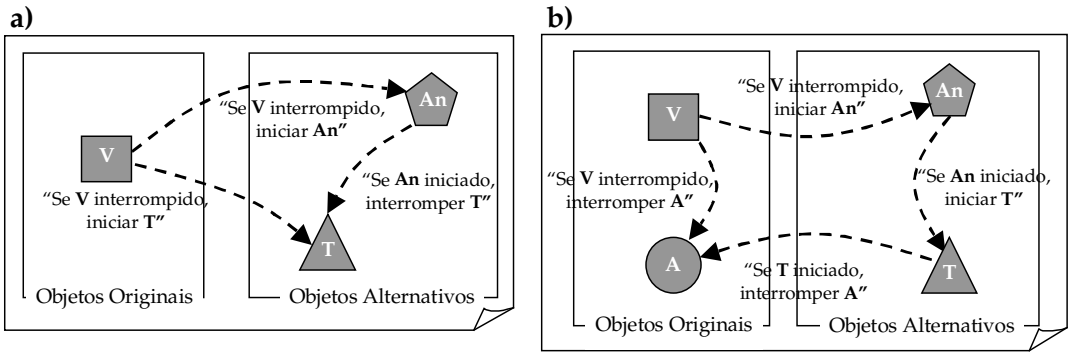

Figura 6.1 – Caminhos de *liks* distintos chegando a uma mesma mídia

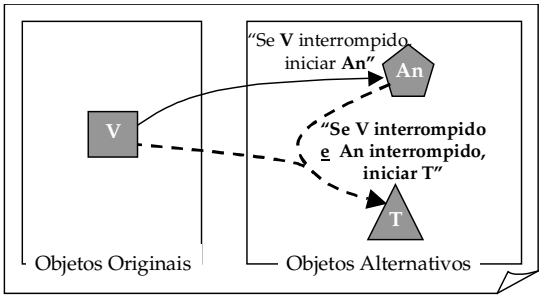

Figura 6.2 – *Link* condicional nunca disparado

(ii) A expressão lógica de um *link* condicional a qual condiciona ativação do mesmo pode nunca se tornar verdadeira. Neste caso, o *link* condicional não será disparado em nenhuma situação, representando, por exemplo, a especificação de mídias alternativas jamais ativadas (figura 6.2).

(iii) A definição de *links* condicionais entre mídias apresentadas em diferentes instantes da apresentação, ou iniciadas em conjunto mas possuindo durações distintas, pode causar a interrupção de mídias já apresentadas. A figura 6.3a mostra o *timeline* da apresentação de um documento multimídia. Na figura 6.3b são ilustrados os *links* condicionais para o respectivo documento. De acordo com as informações de adaptação, se ocorrer uma falha de QoS em "A", ela será substituída por "C", assim como a mídia "B" será interrompida e substituída por "D". Seja t_f o instante da apresentação em que ocorre uma falha de QoS na mídia "A" e sejam t_A e t_B os instantes de finalização das mídias "A" e "B", respectivamente. Para $0 \leq t_f \leq t_B$, o processo de adaptação é executado normalmente. No entanto, quando $t_B < t_f \leq t_A$, a mídia "B" não está mais ativa. Portanto, o processo de adaptação interrompe uma mídia já apresentada. Se o atributo "restart" definido para a mídia "A" possuir os valores "document" ou "media", a adaptação ocorre da forma desejada, uma vez que a mídia "D" (alternativa de "B") será apresentada ao usuário. No entanto, para "restart=failure", o processo de adaptação apresenta uma inconsistência, uma vez que ele interrompe uma mídia já apresentada ("B") e ativa uma mídia alternativa que não será exibida ("D") durante a apresentação do documento adaptado.

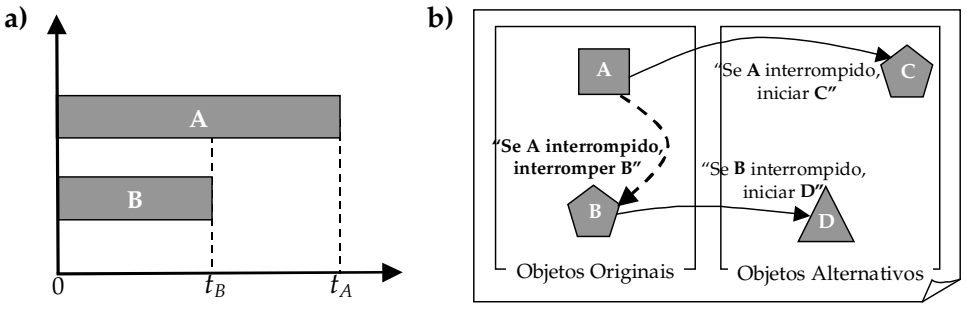

Figura 6.3 – *Link* condicional desativando mídia já apresentada, para $t_B < t_f \leq t_A$

6.2 Ferramenta de Verificação de Consistência de Documentos Adaptativos

Como foi apresentado na sessão anterior, documentos multimídia adaptativos podem apresentar inconsistências referentes às informações de adaptação. Para auxiliar o autor durante a criação dos documentos, mais especificamente, durante a especificação das dependências condicionais, foi desenvolvida uma ferramenta gráfica ("*SCL Semantics*") que permite a visualização dessas inconsistências semânticas. Após a criação dos arquivos SMIL e SCL, o autor executa a ferramenta que, com base na simulação de falhas de QoS, executa todas as adaptações possíveis do documento e aponta as inconsistências identificadas.

6.2.1 Análise de Requisitos

O desenvolvimento da ferramenta "*SCL Semantics*" teve como principal objetivo automatizar a identificação de inconsistências semânticas definidas no documento adaptativo. Durante a análise de requisitos foram levados em consideração os três tipos de inconsistência semântica descritos anteriormente.

Como foi descrito na sessão 4.2.4, como resultado intermediário do processo de adaptação de um documento é gerada uma lista de eventos representando todas as mídias que devem ser interrompidas e ativadas. Analisando esta lista é possível, portanto, identificar o primeiro tipo de inconsistência descrito (caminhos de *liks* distintos chegando a uma mesma mídia): basta verificar se existe mais de um evento referente a uma mesma mídia. Por exemplo, se existem dois eventos "A:stopped" ou se existe um evento "A:stopped" e um evento "A:started". Desta forma, a execução de todas as adaptações possíveis e a análise das listas de eventos geradas durante essas adaptações permitem a identificação deste tipo de inconsistência. Para executar todas as adaptações possíveis de uma apresentação deve-se, portanto, simular uma falha de QoS para cada mídia contínua referenciada no documento multimídia.

A simulação de falhas de QoS para executar todas as adaptações possíveis do documento também é uma solução que permite a identificação do segundo tipo de

inconsistência semântica (especificação de mídias alternativas jamais ativadas). Após a execução de todas as simulações é possível verificar se algum dos *links* condicionais especificados não foi disparado em nenhum momento.

A identificação do terceiro tipo de inconsistência (*links* condicionais desativando mídias já apresentadas) é uma tarefa um pouco mais complexa, uma vez que essa inconsistência depende do instante em que ocorre a falha de QoS em uma mídia e do instante em que a apresentação deve ser reiniciada (definido pelo parâmetro "restart" desta mídia). Neste caso, a simulação de falhas de QoS para uma mesma mídia em instantes distintos da apresentação permite que seja verificado se o processo de adaptação está alterando mídias que já foram apresentadas, o que caracteriza este tipo de inconsistência. Uma solução para determinar em que instantes as simulações devem ser realizadas é a divisão da apresentação em estados, onde cada transição entre os estados é caracterizada pelo início da apresentação de novas mídias e/ou pela finalização de mídias que estão sendo apresentadas. Desta forma, em cada estado tem-se o conhecimento das mídias que já foram apresentadas e das mídias que estão sendo apresentadas.

6.2.2 Implementação da Ferramenta *SCL Semantics*

Basicamente, a ferramenta de verificação semântica realiza os seguintes passos para identificar as inconsistências de um documento adaptativo: (i) a apresentação original (definida no documento SMIL) é dividida em estados; (ii) para cada estado é simulada a falha de QoS de cada mídia contínua e ativa (i.e. que está sendo apresentada) do respectivo estado, resultando na execução de todas as adaptações do documento; (iii) utilizando os resultados das adaptações são identificadas as inconsistências semânticas e as mesmas são apontadas através de uma interface gráfica. No apêndice C são descritas as classes definidas durante a implementação da ferramenta.

A ferramenta inicia o processo de verificação analisando o arquivo SMIL com o objetivo de construir um diagrama de estados representando o escalonamento do documento (*scheduling graph*). Para isso o documento deve estar consistente, ou seja, as restrições temporais de todas as suas mídias devem estar bem definidas. Partindo

deste princípio é possível identificar o instante de todos os eventos internos determinísticos da apresentação. Esses eventos podem ser do tipo "*s*" (*start*) indicando o início de uma determinada mídia "X" (e.g. "*s_X*"), ou do tipo "*e*" (*end*) indicando o término de uma determinada mídia "X" (e.g. "*e_X*"). Eventos externos não-determinísticos, como interações do usuário ou atrasos de rede, não são considerados.

Para a construção do diagrama, os eventos são ordenados cronologicamente e separados em grupos de eventos simultâneos. Cada transição t_i no diagrama de estados é caracterizada por um grupo de eventos simultâneos e possui sua origem em um estado e seu destino em um segundo estado. Por outro lado, cada estado s_i é caracterizado por um conjunto de mídias já apresentadas PM_i (*presented media*) e um conjunto de mídias ativas AM_i (*active media*). Se a transição t_i tem origem em s_i e destino em s_{i+1}, sendo ela formada pelo conjunto de eventos simultâneos $[e_X_1 \ldots e_X_j, s_Y_1 \ldots s_Y_k]_{j\geq 0,k\geq 0}$, e seja $X=\{X_1..X_j\}$ e $Y=\{Y_1..Y_k\}$, então $PM_{i+1}=PM_i \cup X$ e $AM_{i+1}=\{AM_i-X\}\cup Y$. Para o primeiro e último estado (s_0 e s_n,) não há nenhuma mídia ativa ($AM_0=AM_n=\varnothing$). A figura 6.4 mostra um exemplo de diagrama de estado representando o escalonamento do *timeline* ilustrado na mesma figura.

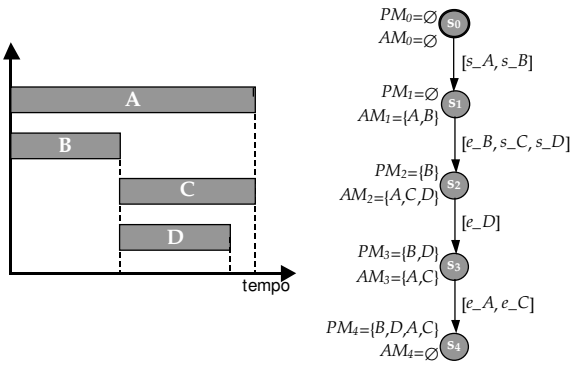

Figura 6.4 – Diagrama de estados para o escalonamento de uma apresentação

A definição de um diagrama de estados da apresentação permite que a ferramenta analise o comportamento da adaptação em função dos estados em que são simuladas as falhas de QoS. Como foi descrito anteriormente, algumas inconsistências nas informações de adaptação só são identificadas em determinados estados da apresentação. Por exemplo, a substituição de uma mídia *A* por sua alternativa somente é identificada como uma inconsistência se a adaptação é disparada em um estado s_i onde *A* não está mais ativa (i.e. $A \notin AM_i$ e $A \in PM_i$).

Definido o diagrama de estados, a ferramenta percorre cada estado s_i (com exceção do primeiro e último estado) e simula uma falha de QoS para cada mídia contínua *A* (áudio, vídeo ou animação) que esteja ativa no estado ($A \in AM_i$). Para cada falha de QoS simulada, um processo de adaptação forte é disparado. Neste ponto é realizada uma chamada ao módulo *Adapter*, descrito na sessão 4.2.4. O *Adapter* executa o processo de adaptação, retornando como resultado a lista de todos eventos disparados (indicando as mídias que são interrompidas e as alternativas que são ativadas) e o documento SMIL adaptado, caso nenhum erro tenha ocorrido durante a adaptação. Esses erros podem ser causados, por exemplo, por problemas na sintaxe do documento SCL ou por referências a mídias inexistentes no documento SMIL.

Para cada documento SMIL adaptado retornado pelo *Adapter* é construído um novo diagrama de estados representando seu escalonamento. Em seguida, este novo diagrama é agregado ao diagrama da apresentação original. Para ilustrar o processo de agregação dos diagramas de estados, considere o exemplo da figura 6.5. Nessa figura é ilustrado o *timeline* de uma apresentação original (figura 6.5a) e o seu respectivo diagrama de estados (figura 6.5b). Neste exemplo é simulada uma falha de QoS da mídia "C" dentro do estado "s_{A2}". De acordo com os *links* condicionais definidos para este documento (figura 6.5c), a apresentação é adaptada ocorrendo a substituição da mídia "C" pela mídia "F". A figura 6.6 ilustra o *timeline* da apresentação adaptada (figura 6.6a) e o diagrama de estados gerado para esta apresentação (figura 6.6b).

Supondo que o parâmetro "restart" da mídia "C" tenha o valor "media", a apresentação adaptada deve ser iniciada a partir do início da mídia "F". Como essa mídia só inicia sua apresentação em "s_{B2}", os dois primeiros estados do diagrama da apresentação adaptada podem ser desconsiderados. Portanto, a apresentação adaptada é iniciada, efetivamente, em "s_{B2}" (figura 6.6). A agregação entre os dois diagramas deve ser realizada criando-se uma transição para conectar o estado em que é simulada a falha de QoS ("s_{A2}") e o estado em que a apresentação adaptada é iniciada ("s_{B2}"). A lista de eventos que caracteriza essa transição é definida com base nos conjuntos de mídias ativas desses dois estados. No exemplo, como *"M_{A2}={A,C}"* e *"M_{B2}={A,F}"*, a lista de eventos da transição entre "s_{A2}" e "s_{B2}" deve ser "*[e_C,s_F]*" (figura 6.7a).

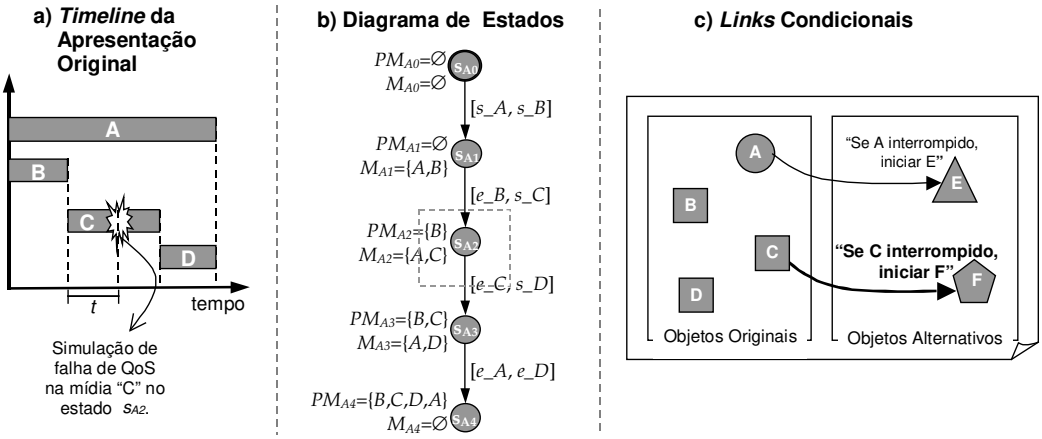

Figura 6.5 – Apresentação de um documento adaptativo

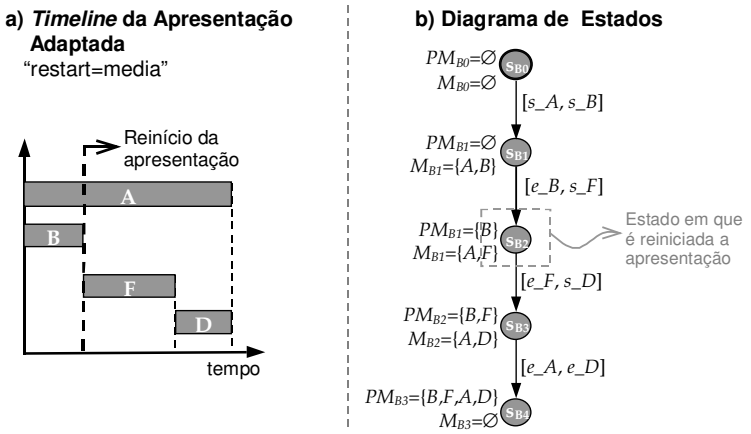

Figura 6.6 – Apresentação adaptada

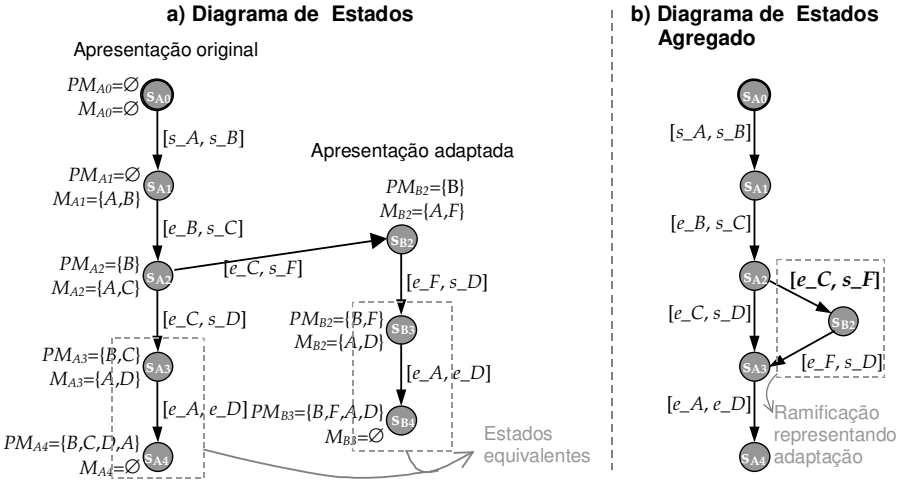

Figura 6.7 – Agregação dos diagramas de estados

Existem ainda outras duas situações em que o processo de agregação pode ser executado: quando “restart=failure” ou “restart=document”. Suponha que seja simulada a falha de QoS de “X”, e que a mídia “Y” substitui “X” durante a adaptação. No caso em que o parâmetro “restart” de “X” possui o valor “failure”, para executar o processo de agregação entre os diagramas da apresentação adaptada e da apresentação original é necessário saber quanto tempo de “X” já foi apresentado até o instante em que é simulada a falha de QoS (seja “t_X” este tempo). Da mesma forma, dentro do diagrama de estados também devem ser informados os instantes de ocorrência de cada transição (indicando quando ocorre cada evento). Com isso, é possível saber a partir de qual dos estados em que a mídia “Y” encontra-se ativa deve-se iniciar a apresentação adaptada. Neste caso, se “s_Y” ocorre no instante “t_{sY}”, basta identificar qual estado equivale ao instante “t_{sY}+t_X”. Se “Y” só está ativa em um estado, a agregação ocorre de forma equivalente ao caso em que “restart=media” (esta situação ocorreria no exemplo anterior se o parâmetro “restart” de “C” fosse igual a “failure”). É importante lembrar que os problemas descritos na sessão 4.1.2, referentes ao reinício da apresentação, também devem ser tratados durante a implementação deste processo de agregação.

No caso em que o parâmetro “restart” tem valor “document”, a agregação entre os dois diagramas é realizada entre o estado da apresentação original em que é simulada a falha de QoS e o segundo estado da apresentação adaptada (o primeiro estado representa a apresentação ainda não iniciada). Se o processo de adaptação não alterar nenhuma mídia na parte da apresentação anterior ao instante em que é simulada a falha de QoS, os estados iniciais dos dois diagramas serão idênticos. Neste caso, pode-se criar uma transição ligando o estado em que ocorre a falha e o segundo estado da própria apresentação original. No entanto, isso pode comprometer a ilustração do comportamento adaptativo da apresentação.

Os valores “document” e “failure” do parâmetro “restart” ainda não são suportados na versão atual da ferramenta “*SCL Semantics*”. Somente o valor *default* (“media”) está sendo considerado.

Como resultado do processo de agregação, a apresentação adaptada passa a ser representada por uma ramificação partindo do estado em que foi simulada a falha de

QoS. Dependendo da apresentação gerada pelo processo de adaptação, esta ramificação pode retornar ao caminho principal (caminho definido pelo diagrama de estados da apresentação original). Isso ocorre quando o processo de adaptação manipula (substitui ou cancela) somente as mídias de um determinado trecho da apresentação. Desta forma, os estados finais da apresentação adaptada são equivalentes ao da apresentação original. Ainda considerando o exemplo anterior, como é apontado na figura 6.7a, os estados finais "s_{B3}" e "s_{B4}" da apresentação adaptada são equivalentes aos estados "s_{A3}" e "s_{A4}" da apresentação original. Esta equivalência permite, portanto, que os estados "s_{B3}" e "s_{B4}" sejam desconsiderados e que o estado subsequente a "s_{B2}" seja "s_{A3}". O resultado final da agregação dos diagramas de estados das duas apresentações é ilustrado pela figura 6.7b. É importante observar que o último estado do diagrama de uma apresentação adaptada é sempre equivalente ao último estado do diagrama da apresentação original.

Finalmente, após a realização de todas as adaptações possíveis, a ferramenta gera um diagrama de estados onde estão presentes todas as ramificações representando essas adaptações. Este diagrama permite a visualização de todo o comportamento adaptativo da apresentação. Para cada ramificação de adaptação, a ferramenta apresenta a lista de *stoplinks* e *startlinks* disparados, apontando, quando for o caso, dois dos três tipos de inconsistências semânticas descritas anteriormente: (i) ativação e desativação de uma mesma mídia e (ii) substituição de mídias já apresentadas. O terceiro tipo de inconsistência nas informações de adaptação é identificado caso seja verificado que algum dos *links* condicionais especificados não é disparado em nenhum momento.

6.2.3 Interface da Ferramenta *SCL Semantics*

A figura 6.8 apresenta a janela principal da ferramenta. Após ter criado os documentos SMIL e SCL, o autor entra com o nome do arquivo SMIL (o arquivo SCL é homônimo, possuindo a extensão ".scl"). Inicialmente a ferramenta exibe apenas o diagrama de estados representando o escalonamento da apresentação especificada pelo documento SMIL.

Quando um dos estados é clicado, o diagrama é "explodido" exibindo novos caminhos ou ramificações (figura 6.9, figura 6.10). Como foi dito anteriormente, essas ramificações representam as alterações no comportamento da apresentação devido às possíveis adaptações no documento original. Cada nova transição que parte do estado clicado é rotulada com o "id" da mídia cuja falha de QoS foi simulada (e.g. "*[a1]*") e a lista de eventos representando o re-escalonamento da apresentação em direção ao primeiro estado da ramificação de adaptação (e.g. "*(e_a1,s_t1)*").

A ferramenta *SCLSemantics* possui um módulo complementar para a visualização de todos os *links* condicionais especificados no documento SCL. Quando o botão "*Show Links Diagram*" é clicado, é exibida uma janela (figura 6.11) contendo um diagrama ilustrando as mídias originais e alternativas e os *links* entre elas (*startlinks* em azul e *stoplinks* em vermelho). Os *links* associados às inconsistências identificadas pela ferramenta são destacados.

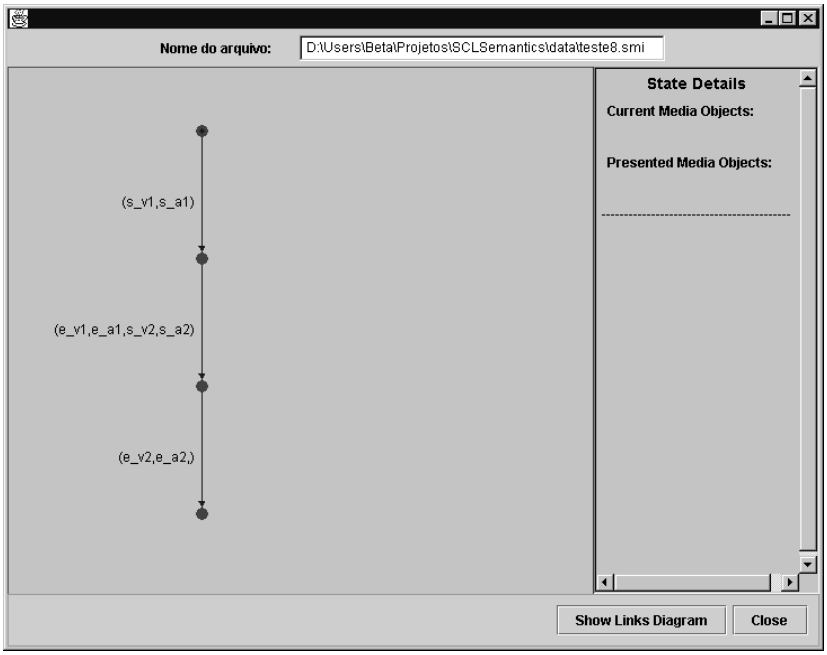

Figura 6.8 – Ferramenta *SCLSemantics*: diagrama de estados inicial

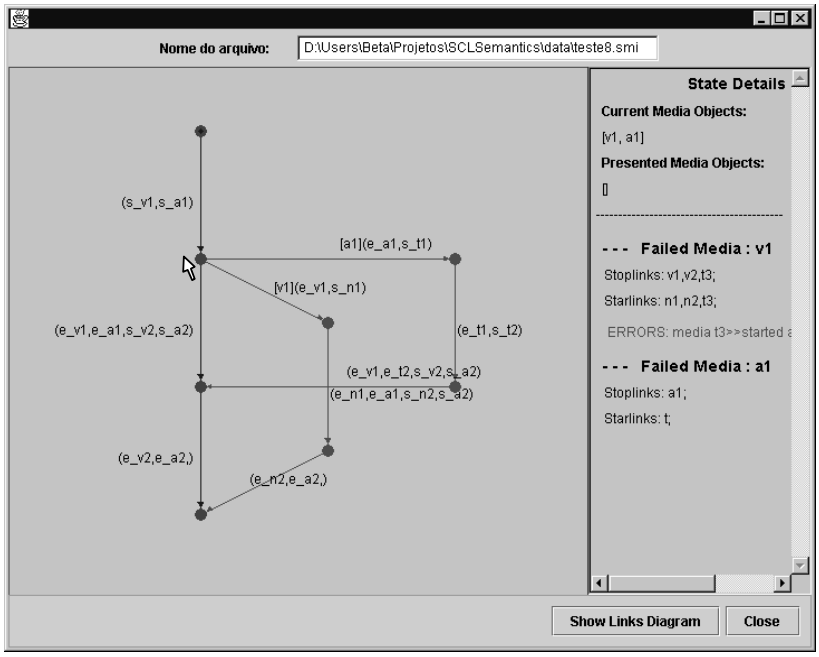

Figura 6.9 – Ferramenta *SCLSemantics*: caminhos representando adaptações

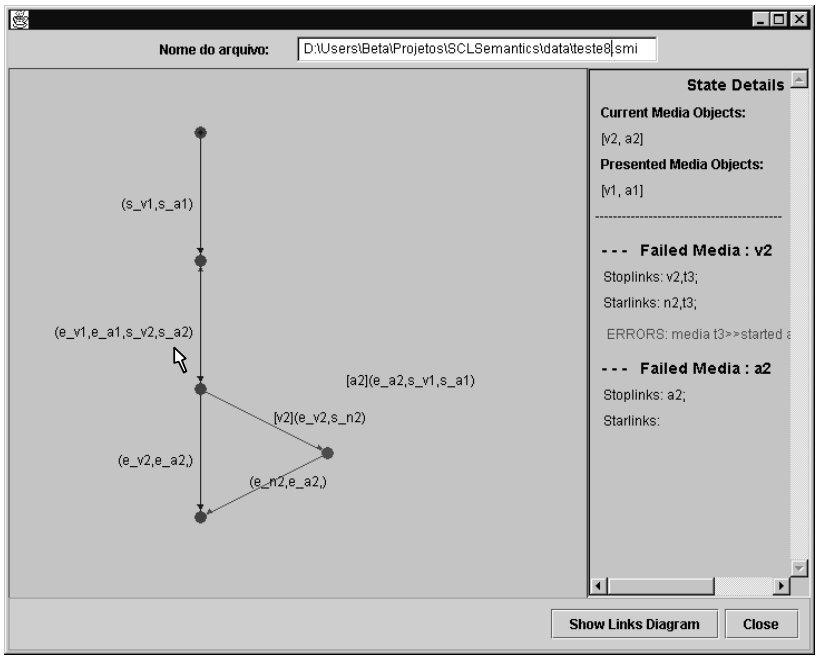

Figura 6.10 – Ferramenta *SCLSemantics*: caminhos representando adaptações

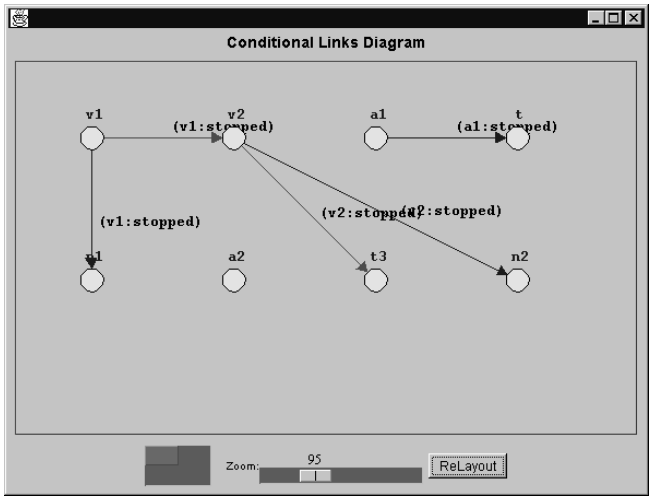

Figura 6.11 – Módulo complementar (diagrama de *links* condicionais)

6.3 Resumo

Neste capítulo foram apresentadas algumas questões relativas à semântica das informações de adaptação contidas no arquivo de controle. Foi mostrado que a flexibilidade da linguagem SCL permite que o autor especifique inconsistências semânticas junto a essas informações, o que pode afetar o comportamento adaptativo da apresentação. Três tipos de inconsistências foram descritos e exemplificados.

Este capítulo também apresentou uma ferramenta de verificação de documentos adaptativos desenvolvida para auxiliar a identificação dos três tipos de inconsistências descritos. Esta ferramenta realiza uma verificação do comportamento adaptativo através de simulações gerando todas as adaptações possíveis do documento multimídia. Com base nos resultados das adaptações, a ferramenta constrói e apresenta um diagrama que permite a visualização de todos os estados pelos quais a apresentação pode passar. Além disso, a ferramenta aponta todas as inconsistências encontradas. Esta ferramenta também possui um módulo complementar para a visualização de um diagrama representando todos os *links* condicionais definidos pelo autor no arquivo de controle.

Conclusões

Neste trabalho foi proposta uma estratégia de autoria e apresentação que visa tornar adaptativas as aplicações multimídia. Esta estratégia baseia-se no estabelecimento de uma malha de relacionamentos condicionais que permite que, durante uma apresentação, seja verificada a necessidade de adaptar alguma informação e que define a forma como esta adaptação deve ser realizada.

A especificação dos documentos adaptativos é realizada através de dois arquivos distintos: um arquivo contendo uma apresentação multimídia convencional e um segundo arquivo contendo as informações de adaptação. Desta forma pode-se garantir a interoperabilidade com sistemas multimídia já existentes. A especificação do documento multimídia original é realizada através de uma linguagem padronizada (SMIL 1.0). As informações de controle de adaptabilidade (compostas dos descritores de QoS e dos *links* condicionais) são mantidas externas ao documento multimídia original, em um arquivo de controle. Durante o trabalho foi verificado que este tipo de abordagem simplificava o desenvolvimento de uma ferramenta de apresentação (*player)* dos documentos adaptativos. Basicamente, a utilização desta abordagem implicou as seguintes vantagens: (i) a apresentação do documento multimídia pode ser efetuada por um *player* SMIL, mesmo que este não seja capaz de considerar as informações de controle; (ii) não existe a necessidade do desenvolvimento integral de uma nova ferramenta de apresentação, uma vez que outros módulos já desenvolvidos para processar XML (*parsers, geradores, etc.*) podem ser aproveitados no desenvolvimento da ferramenta de apresentação; e (iii) torna o sistema independente de novas versões do SMIL.

Dentro da estratégia são definidos dois mecanismos combinados de adaptação para a apresentação de documentos multimídia em rede: (i) um mecanismo suave (nível 1), que efetua uma adaptação no nível de codificação de mídias independentes, de forma transparente ao usuário; e (ii) um mecanismo forte (nível 2), que leva o conceito de

adaptabilidade ao nível do documento, gerando uma nova estrutura lógica e temporal, em tempo de apresentação, com base na avaliação dos relacionamentos condicionais entre as mídias. O mecanismo de adaptação forte só deve ser ativado nas situações em que o primeiro mecanismo não conseguir manter os requisitos de QoS especificados. Uma vez esgotadas as possibilidades de adaptação no primeiro nível, uma adaptação mais severa é realizada no documento e, somente neste caso, a apresentação do mesmo deverá ser interrompida. A execução da adaptação em dois níveis foi adotada com o objetivo de atenuar seu efeito sobre o usuário final.

Durante o trabalho, foi implementado o módulo *Adapter* com o objetivo de verificar a viabilidade da estratégia de adaptação proposta com respeito à criação de documentos adaptados em tempo real. Sobre essa implementação, foram realizados alguns testes de desempenho de forma a examinar o comportamento do tempo de resposta do sistema em função do número de *links* condicionais do documento adaptativo. Com isso, foi verificado que o número de *links* condicionais exerce uma influência direta no desempenho do mecanismo. Durante a implementação foi utilizada a plataforma *Java API for XML Parsing 1.0*, desenvolvida pela SUN. O fato de a implementação ser baseada na linguagem Java implicou um considerável *overhead* ao desempenho. Mas, por outro lado, a interoperabilidade do sistema foi garantida.

Durante o desenvolvimento da estratégia de adaptação, esperava-se que a adaptação suave apresentasse um tempo de resposta bem melhor que o da adaptação forte. Isto porque o atraso introduzido pela adaptação suave é decorrente apenas do processo de chaveamento de fluxos, enquanto o atraso total introduzido pela adaptação forte é composto pelo (i) tempo de criação do documento adaptado mais o (ii) tempo de chaveamento de fluxos. No entanto, de acordo com os testes apresentados em [41], os atrasos introduzidos no nível de chaveamento de fluxos para os dois mecanismos de adaptação são da mesma ordem (em média, 1500ms para a adaptação suave e 2000ms para a adaptação forte). Além disso, os testes realizados com o módulo *Adapter* mostraram que o tempo de criação do documento adaptado é bem menor (em média, de cinco a dez vezes menor) do que o tempo de chaveamento de fluxos. Desta forma, o atraso total introduzido pela adaptação forte, ao contrário do que se esperava inicialmente, é da mesma ordem que o atraso introduzido pela adaptação suave.

Em um ambiente de ensino à distância o atraso de adaptação tende a ser aceitável, pois as dimensões temporais das mídias acessadas neste tipo de aplicação são da ordem de alguns minutos. Além disso, quando se trata de uma aplicação voltada ao ensino, é melhor a convivência com um pequeno atraso de chaveamento do que a possibilidade de uma interpretação equivocada das informações. Vale ressaltar novamente que a adaptação forte deve ser ativada apenas nas situações em que o mecanismo de adaptação suave não conseguir manter os requisitos de QoS especificados.

Também como parte do trabalho, foi desenvolvida uma ferramenta de auxílio à autoria de documentos adaptativos. Esta ferramenta permite a identificação de possíveis inconsistências semânticas definidas nas informações de controle. A ferramenta, com base em simulações de falha de QoS, executa todas as possíveis adaptações do documento. Através de uma interface gráfica, são apresentados todos os estados que a apresentação pode percorrer, além das inconsistências identificadas. Durante o desenvolvimento deste trabalho, verificou-se que a especificação de todos os *links* condicionais de uma apresentação para determinar seu comportamento adaptativo pode ser uma tarefa bastante complexa. Desta forma, a ferramenta implementada mostrou-se bastante funcional no sentido de simplificar esta tarefa e garantir a consistência dos dados de adaptação especificados.

Como trabalhos futuros, podem ser realizados os estudos a seguir.

Pode-se implementar um mecanismo para o monitoramento dos recursos locais, como taxa de utilização de processador e memória. Desta forma, novas opções para a determinação dos parâmetros de QoS podem ser definidas.

No que diz respeito aos mecanismos de adaptação, um refinamento na sincronização do reinício do documento adaptado pode ser realizado. A definição de "pontos de sincronização" entre mídias originais e mídias alternativas permitiriam uma maior garantia de equivalência semântica entre o instante de ocorrência da falha de QoS e o instante de reinício da apresentação adaptada. Em conjunto, deve ser feito um estudo

sobre o momento de solicitação da adaptação, com base no momento da violação da QoS, buscando eliminar possíveis oscilações de adaptação.

Também pode ser analisada a viabilidade de se adaptar o documento seguindo os *links* condicionais no sentido inverso, quando a QoS verificada na rede sofrer uma melhoria. Nesta análise, deve ser considerada a relação custo-benefício de se executar uma adaptação forte no sentido de retornar à condição original da apresentação.

Com relação ao SCL e à ferramenta de verificação desenvolvida, pode ser realizada uma análise semântica da linguagem mais precisa, com o objetivo de identificar outros tipos de inconsistências contidas em sua definição. Da mesma forma, a ferramenta poderia ser complementada no sentido de identificar essas novas inconsistências.

Um trabalho que representa uma consequência direta deste é o desenvolvimento dos servidores de mídias que implementem as estratégias de monitoramento e adaptação dos fluxos (adaptação suave). Desta forma, é possível verificar a escalabilidade desses servidores durante o gerenciamento de várias adaptações simultâneas. Também, a posterior incorporação na ferramenta de autoria desenvolvida no projeto ServiMídia [41] de toda a metodologia de criação de documentos adaptativos que está sendo gradualmente amadurecida. Da mesma forma, integrar a esta ferramenta o módulo de verificação da consistência semântica das informações de adaptação. O objetivo final é tornar a difícil tarefa de criação de documentos multimídia adaptativos o mais transparente possível para o autor.

Até então, no Projeto ServiMídia, têm sido considerados ambientes de ensino com escopo local (LANs). No caso aplicações executadas em uma WAN, a estratégia de adaptação pode ter seu desempenho deteriorado em função dos atrasos de rede. Levar esse processo de adaptação para nós intermediários utilizando-se do paradigma de redes ativas pode tornar os efeitos indesejáveis desse atraso menos perceptíveis para o usuário. A utilização de redes ativas também pode permitir que apresentações adaptativas sejam executadas em grupos *multicast*. Neste caso, os nós intermediários da árvore *multicast* poderiam ser os responsáveis pela solicitação das adaptações.

Apêndice A

DTD da Linguagem SCL

A seguir, encontra-se listado o arquivo “scl.dtd”, onde é definido o DTD da linguagem *Smil Control Language* (SCL).

```
<!ENTITY % subsmil SYSTEM "smil.dtd">
<!ELEMENT scl (controls,alternatives)>
<!ELEMENT controls (description)+>
<!ELEMENT description (minRequirements?,maxRequirements?,stoplink?)>
<!ATTLIST description
 about CDATA #IMPLIED
 restart (failure|media|document) "media" >
<!ELEMENT minRequirements EMPTY>
<!ATTLIST minRequirements
 width CDATA #IMPLIED
 height CDATA #IMPLIED
 delay CDATA #IMPLIED
 jitter CDATA #IMPLIED
 bpp CDATA #IMPLIED
 lossrt CDATA #IMPLIED
 bw CDATA #IMPLIED>
<!ELEMENT maxRequirements EMPTY>
<!ATTLIST maxRequirements
 width CDATA #IMPLIED
 height CDATA #IMPLIED
 delay CDATA #IMPLIED
 jitter CDATA #IMPLIED
 bpp CDATA #IMPLIED
 lossrt CDATA #IMPLIED
 bw CDATA #IMPLIED>
<!ELEMENT stoplink EMPTY>
<!ATTLIST stoplink
 expr CDATA #IMPLIED>
%subsmil;
<!ELEMENT alternatives (replace)+>
<!ELEMENT replace (startlink,regionToAdd?,resourceToSubstitute?)>
<!ATTLIST replace
 target CDATA #IMPLIED
 target-type CDATA #IMPLIED>
<!ELEMENT startlink EMPTY>
<!ATTLIST startlink
 expr CDATA #IMPLIED>
<!ELEMENT regionToAdd (region)+>
<!ELEMENT resourceToSubstitute (%body-content;)>
```

Apêndice B

Módulo *Adapter*

Na implementação do *Adapter*, foi utilizado o JAXP [21], uma API que oferece ferramentas de interpretação (*parsers*) e geração de saídas XML.

B.1 Algumas Características do *Java API for XML Parsing*

O JAXP é um conjunto de APIs Java desenvolvido pela SUN *Microsystems* que suporta o processamento de documentos XML utilizando os modelos DOM, SAX e XSLT. Ele permite que aplicações possam executar o *parsing* de informações XML, permitindo também a geração de documentos XML.

As principais classes do JAXP estão definidas no pacote `javax.xml.parsers`. Dentro desse pacote estão as classes do tipo *Factory* que retornam objetos de classes que representam os *parsers* XML. Existem dois níveis básicos de *parsers*, os *parsers* da SAX API e da DOM API.

A *Simple API for XML* (SAX) oferece um mecanismo orientado a eventos que fornece um acesso serial aos dados, realizando o processamento elemento-por-elemento. A API para este nível realiza leitura e escrita em um repositório de dados (ou na *Web*). Este mecanismo é bastante indicado para aplicações servidoras que devem apresentar altos desempenhos.

A API do *Document Object Model* (DOM) é, geralmente, considerada mais simples de ser utilizada. Ela fornece uma estrutura de objetos em árvore relativamente familiar. A DOM API pode ser utilizada para manipular a hierarquia de objetos encapsulada por uma aplicação. Ela é ideal para aplicações interativas uma vez que o

modelo de objetos é mantido integralmente em memória, de forma que ele possa ser acessado e manipulado pelo usuário.

Por outro lado, a construção de um DOM exige que toda a estrutura XML seja lida e que toda a estrutura de objetos em árvore seja mantida em memória, o que faz com que a aplicação exija mais recursos de CPU e memória. Por esta razão, a SAX API é mais indicada para aplicações servidoras e para filtros de dados que não exigem uma representação em memória dos dados.

B.2 Classes Implementadas

O módulo *Adapter* é constituído de seis classes principais, como mostrado no diagrama de classes da figura B.1.

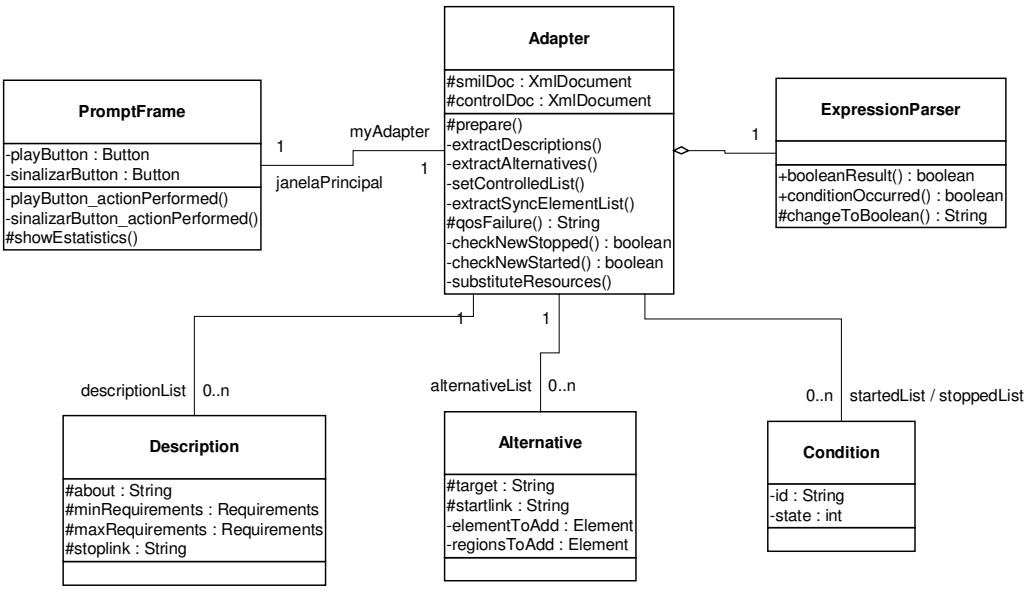

Figura B.1 – Diagrama de classes simplificado do módulo *Adapter*

(i) *PromptFrame* – Esta classe representa uma janela de interface através da qual são feitas chamadas à classe *Adapter*. Nesta interface o usuário entra com o *path* de um arquivo SMIL, acionando sua apresentação através de um botão "*Play*" (*PromptFrame.playButton_actionPerformed()*). Neste momento, o *PromptFrame*

faz uma chamada a um *player* SMIL (e.g. *RealPlayer*). Além do botão "*Play*", esta janela possui um botão "Sinalizar" que, quando clicado (*PromptFrame. sinalizarButton_actionPerformed()*), faz uma chamada ao *Adapter* informando uma falha de QoS em uma mídia (o usuário indica a mídia através da mesma interface).

(ii) *Adapter* – Esta é a classe principal dentro do módulo. Ela realiza todo o processamento relativo ao *parsing* dos arquivos SMIL e SCL, além de executar todo o processo de adaptação até a geração de um arquivo SMIL adaptado. Quando o usuário clica o botão "*Play*" do *PromptFrame*, o *Adapter* já se prepara (*Adapter.prepare()*) realizando o *parsing* dos dados XML gerando em memória duas estruturas DOMs (uma para o documento SMIL e uma para o documento SCL). Para a construção dessas estruturas, são utilizadas as classes da DOM API fornecida pelo JAXP. Com isso, ele realiza uma busca na estrutura relativa aos dados SCL para gerar duas listas (*Adpter.extractDescriptions()* e *Adapter.extractAlternatives()*): (i) a primeira contendo objetos da classe *Description* (representando os dados contidos em cada elemento "description" do SCL) e (ii) a segunda contendo objetos da classe *Alternatives* (representando os dados contidos em cada elemento "replace" do SCL). A manutenção desta lista permite otimizar a posterior busca pelos *links* condicionais disparados na ocorrência de uma falha de QoS. Quando o *PromptFrame* faz uma chamada ao *Adapter* solicitando a execução da adaptação (*Adapter.qosFailure()*), este último realiza um processo de busca por *links* condicioniais disparados, fazendo chamadas consecutivas aos métodos *Adapter.checkNewStopped()* e *Adapter.checkNewStarted()*, até que nenhum *link* seja mais disparado. Por último ele cria uma nova estrutura DOM (cópia da estrutura referente ao arquivo SMIL) sobre a qual ele realiza a reestruturação do documento (*Adapter.substituteResources()*).

(iii) *Description* – Esta classe é utilizada como uma estrutura de dados para representar o elemento "description" do SCL.

(iv) *Alternative* – Esta classe é utilizada como uma estrutura de dados para representar o elemento "replace" do SCL.

(v) *Condition* – Esta classe define a representação lógica de uma expressão do tipo "*A:started*" ou "*A:stopped*". A combinação booleana de objetos dessa classe compõe um *startlink* ou um *stoplink*.

(vi) *ExpressionParser* – Esta é uma classe utilitária usada para a manipulação de *Strings* e de objetos da classe *Vector*. Ela possui funções, por exemplo, para transformar uma *String* representando um *startLink* em uma expressão lógica.

Apêndice C

Ferramenta *SCL Semantics*

A ferramenta *SCL Semantics* foi implementada em Java, sendo definidos dois grupos de classes: (i) no primeiro grupo estão as classes utilizadas para a criação de um diagrama de *links* condicionais (definidas no *package SCLSemantics*); (ii) no segundo grupo estão as classes utilizadas na geração do diagrama de estados de uma apresentação adaptativa (definidas no *package SCLSemantics.AdaptationMachine*). Na figura C.1 encontra-se ilustrado um diagrama de classes simplificado da ferramenta.

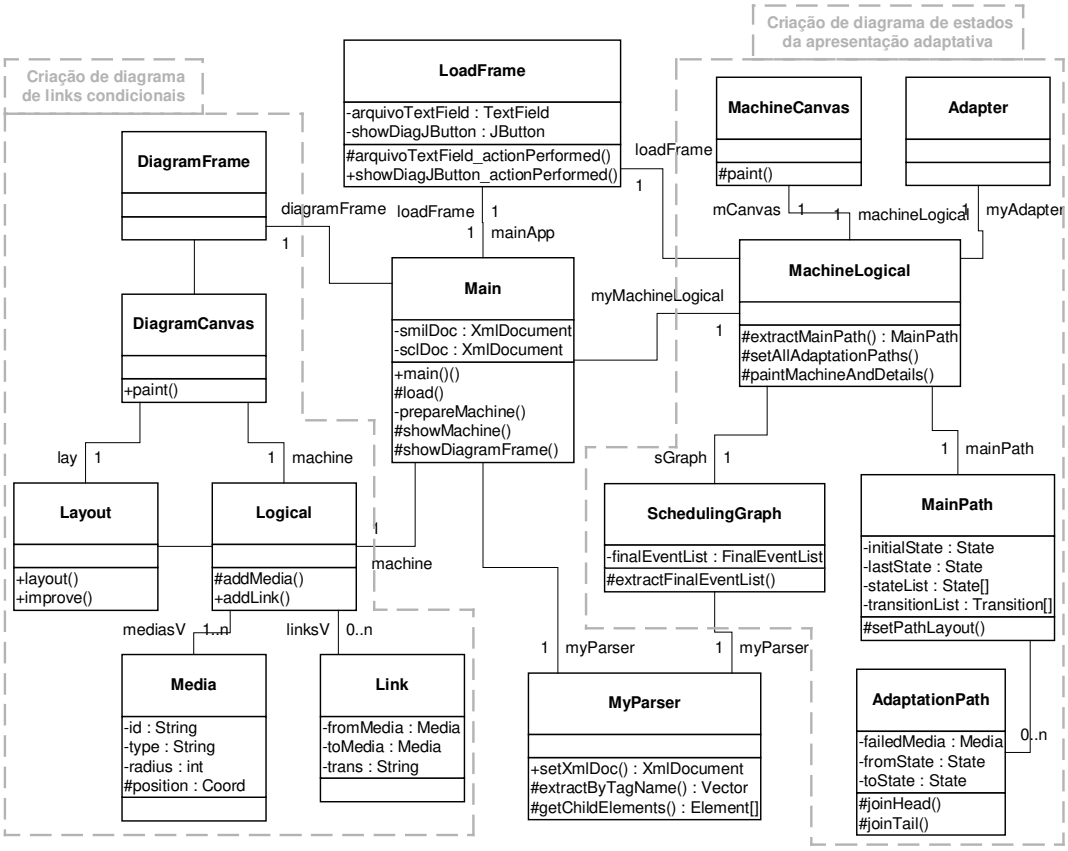

Figura C.1 – Diagrama de classes simplificado da ferramenta *SCL Semantics*

São descritas, a seguir, as classes *Main* e *LoadFrame*, que representam um "elo" entre os dois grupos de classes descritos, e a classe *MyParser*, que é utilizada nos dois grupos.

(i) *Main* – Esta classe é uma aplicação Java (possui o método *main()*) que define a linha de execução da ferramenta. Ela aciona os métodos de leitura dos documentos XML e de criação e preparação dos objetos das demais classes. Esta classe representa, portanto, um elo entra as demais classes do sistema. O método *Main.load()* executa a leitura dos arquivos SMIL e SCL e cria os objetos das classes *MyParser*, *Logical*, *Layout*, e *DiagramCanvas*. Estas classes são utilizadas para a criação do diagrama de *links* condicionais definidos no documento adaptativo. O método *Main.prepareMachine()* cria um objeto da classe *SchedulingGraph* e da classe *MachineLogical*. Esses objetos são os responsáveis por criar o diagrama de escalonamento da apresentação e realizar as possíveis adaptações do documento. Por último, é acionado o método que apresenta o diagrama de estados (*Main.showMachine()*).

(ii) *LoadFrame* – Esta classe representa a janela de interface da ferramenta. Inicialmente o usuário entra com o *path* de um arquivo SMIL (*LoadFrame.arquivoTextField_actionPerformed()*), sendo chamado o método *Main.load()*. Após a execução de toda a linha de execução definida na classe *Main*, esta última retorna ao *LoadFrame* o diagrama de estados (objeto da classe *MachineCanvas*) da apresentação definida no documento adaptativo. Nesta interface há um botão (*LoadFrame.showDiagJButton*) que, quando clicado, chama o método *LoadFrame.showDiagJButton_actionPerformed()* (dentro deste método é feita uma chamada ao método *Main.showDiagramFrame()*) para apresentar uma segunda janela (objeto da classe *DiagramFrame*) contendo um diagrama com os *links* condicionais do documento.

(iii) *MyParser* – Esta classe é responsável por todo tratamento de documentos XML (SCL e SMIL), como a geração de uma estrutura DOM *(MyParser.setXmlDoc()*) e a busca de elementos XML em uma estrutura DOM (*MyParser.extractByTagName()* e *MyParser.getChildElements*). Esta classe é

utilizada por *Main*, para iniciar o processo de criação do diagrama de *links* condicionais, e pela classe *SchedulingGraph*, para gerar as estruturas DOMs de cada documento adaptado.

A seguir, são descritas as classes pertencentes ao primeiro grupo de classes citado (classes para a criação do diagrama de *links* condicionais).

(i) *Logical* – Esta classe representa a estrutura lógica do diagrama de *links* condicionais extraído dos arquivos SCL e SMIL. Esse diagrama é formado por um vetor de objetos da classe *Media* e um vetor de objetos da classe *Link*. *Logical* possui a funcionalidade similar ao de uma máquina de estados, no entanto os estados são mídias e as transições são *links*.

(ii) *Media* – Esta classe é utilizada como uma estrutura de dados para representar uma mídia no diagrama de *links* condicionais.

(iii) *Link* – Esta classe é utilizada como uma estrutura de dados para representar um *link* no diagrama de *links* condicionais.

(iv) *Layout* – Esta classe tem a funcionalidade de gerar o *layout* para um objeto da classe *Logical*. Seu algoritmo de posicionamento dos elementos de mídia funciona fixando-se uma mídia e posicionando as demais mídias no melhor lugar em torno da primeira (de acordo com sua função de avaliação utilizada dentro do método *Layout.layout()*, é chamado o método *Layout.improve()* para melhorar o posicionamento.).

(v) *DiagramCanvas* – Esta classe representa um objeto Java que contém uma imagem. No método *DiagramCanvas.paint()* é desenhado o diagrama de *links* condicionais definidos logicamente pelo objeto da classe *Logical*. Quando um objeto *DiagramCanvas* é gerado, *Main* cria um objeto *DiagramFrame* no qual é inserido o primeiro objeto. *DiagramFrame* é então repassado ao objeto de interface da classe *LoadFrame*.

(vi) *DiagramFrame* – Esta classe representa uma janela de interface que contém um objeto da classe *DiagramCanvas*.

Por último, são descritas as classes pertencentes ao segundo grupo de classes citado (classes para a criação do diagrama de estados de uma apresentação adaptativa).

(i) *Adapter* – Descrita no Apêndice A.

(ii) *MachineLogical* – Esta classe representa a estrutura lógica do diagrama de estados de uma apresentação adaptativa. O método *MachineLogical.extractMainPath()* executa a geração do diagrama de estados da apresentação original (representado por um objeto da classe *MainPath*). No método *MachineLogical.setAllAdaptationPaths()* são disparadas as simulações das falhas de QoS em cada estado do diagrama original, e são realizadas chamadas ao método *MachineLogical.extractAdaptationPath()* para gerar um diagrama de estados para cada adaptação executada (representado por um objeto da classe *Adaptation.Path*). Sempre que é gerado um diagrama de estados para uma adaptação, é executada a agregação deste último ao diagrama da apresentação original. O método *MachineLogical.paintMachineAndDetails()* gera um objeto da classe *MachineCanvas* contendo a figura do diagrama de estados, e repassa esta figura para a janela *LoadFrame*.

(iii) *MainPath* – Esta classe é utilizada como uma estrutura de dados para representar um diagrama de estados de uma apresentação.

(iv) *AdaptationPath* – Esta classe é utilizada como uma estrutura de dados para representar uma ramificação de adaptação de um diagrama de estados de uma apresentação. Esta classe possui os métodos que realizam a agregação com um objeto da classe *MainPath* (*AdaptationPath.joinHead()* e *AdaptationPath.joinTail()*).

(v) *SchedulingGraph* – Esta classe é responsável por realizar a análise das restrições temporais de um documento SMIL e gerar, através do método

SchedulingGraph.extractFinalEventList(), uma lista ordenada dos eventos de escalonamento da apresentação (*SchedulingGraph.finalEventList*).

(vi) *MachineCanvas* – Esta classe define um objeto Java que contém uma imagem. No método *MachineCanvas.paint()* é desenhado o diagrama de estados representado logicamente pelo objeto da classe *MachineLogical*.

Referências

[1] **Analyzer: a public domain protocol analyzer**. Disponível na INTERNET via www.url: http://netgroup-serv.polito.it/analyzer/. Arquivo consultado em 2000.

[2] AURRECOECHEA, C., CAMPBELL, A.T., HAUW, L. "A survey of QoS architectures". **ACM Multimedia Systems**, n.6, p.138-151, 1998.

[3] BULTERMAN, D.C.A "Synchronization of Multi-Sourced Multimedia Data for Heterogeneous Target Systems". In: NETWORK AND OPERATING SYSTEMS SUPPORT FOR DIGITAL AUDIO AND VIDEO, 1993, p.119-129.

[4] BULTERMAN, D.C.A., VAN LIERE, R., VAN ROSSUM, G. "A Structure for Transportable Dynamic Multimedia Documents". In: USENIX SPRING CONFERENCE ON MULTIMEDIA SYSTEMS, 1991, p. 137-155.

[5] CARMO, L.F.R.C., PIRMEZ, L. "ServiMídia: An Integrated System for Multimedia Document Creation and Retrieval with Adaptive QoS Control". In: FRANCE-BRAZIL SYMPOSIUM ON DISTRIBUTED COMPUTER SYSTEMS, 2, 1997, Recife, Brazil.

[6] COURTIAT, J.P., CARMO, L.F.R.C, OLIVEIRA, R.C. "A General-purpose Multimedia Synchronization Mechanism Based on Causal Relations". **IEEE Journal on Selected Areas in Communications**, v.14, n.1, p.185-195, January 1996.

[7] DAVIS, M., DOWNING, A. "Adaptable System Resource Management for Soft Real-Time Systems". In: SYMPOSIUM ON COMMAND AND CONTROL RESEARCH AND DECISION AIDS, 1994, Monterey, California.

[8] ENCARNAÇÁO, J., FOLEY, J., HERRTWICH, R.G. "Fundamentals and Perspectives of Multimedia Systems". In: DAGSTUHL SEMINAR, 1994, Saarbrucken, Germany. **Report…** Seminar Report 92, Univ des Saarlandes.

[9] FERGUSON, P., HUSTON, G. "Quality of Service on the Internet: Fact, Fiction or Compromise?". In: INTERNET CONFERENCE, 1998, Genebra, Suiça.

[10] GECSEI, J. "Adaptation in Distributed Multimedia Systems". **IEEE Multimedia**, v.4, n.2, p.58-66, 1997.

[11] GHINEA, G., THOMAS, J.P., FISH, R.S. "Quality of Perception to Quality of Service Mapping Using a Dynamically Reconfigurable Communication

System", In: IEEE GLOBAL TELECOMMUNICATIONS CONFERENCE, 1999, Rio de Janeiro, Brazil.

[12] **HyperText Markup Language Home Page**. Disponível na INTERNET via www.url:http://www.w3.org/MarkUp/. Arquivo consultado em 2001.

[13] ISO. **Document Style Semantics and Specification Language.** ISO/IEC JTC1/SC34/N10179. January 1996.

[14] ISO. Hypermedia/Time-based Structuring Language (HyTime), 2nd Edition. JTC1/SC18/WG8/N1920rev.1997.

[15] ISO. Information Processing – Text and Office Systems – Office Document Architecture (ODA), Part 1-8. ISO/IEC 8613, 1988.

[16] ISO. Information processing – Text and Office systems – Standard Generalized Markup Language (SGML). ISO/IEC 8879, 1986.

[17] ISO. Information Technology Coding of Multimedia and Hypermedia Information, Part 5: Support for Base-Level Interactive Applications, MHEG-5 IS Document Pre-release 5. ISO/IEC DIS 13522-5, 1996.

[18] ISO. **MPEG 7 Applications document V.7**. ISO/IEC JTC1/SC29/WG11/N2462. October 98, Atlantic City, USA.

[19] ISO. **MPEG 7 Requirements document V.7**. ISO/IEC JTC1/SC29/WG11/N2461. October 98, Atlantic City, USA.

[20] **ISO/IEC/JTC1SC 29 Working Group web page**. Disponível na INTERNET via www.url:http://www.km.giti.waseda.ac.jp/WG12/. Arquivo consultado em 2001.

[21] **Java API for XML Parsing, Release 1.0**. Disponível na INTERNET via www.url:http://java.sun.com/xml/index.html. Arquivo consultado em 2000.

[22] **Java Media Framework API**. Disponível na INTERNET via www.url:http://java.sun.com/marketing/collateral/jmf_ds.html. Arquivo consultado em 2000.

[23] LI, V. O. K., LIAO, W. "Distributed Multimedia Systems". **Proceedings of IEEE**, p. 1061-1108, July 1997.

[24] LIU, C. **Multimedia Over IP: RSVP, RTP, RTCP, RTSP.** Disponível na INTERNET via www.url:http://www.netlab.ohio-state.edu/~jain/cis788-97/ip_multimedia/index.htm. Arquivo consultado em 2000.

[25] MABROUK, M. *et al.* "A Hyperdocument Model Based upon the ODA Standard". In: CONFERENCE ON INTELLIGENT TEXT AND IMAGE HANDLING, 1991, Barcelona, Spain. **Proceedings…** Barcelona, p.245-263.

[26] **Open Financial Exchange (OFX)** Disponível na INTERNET via www.url:http://www.ofx.net/. Arquivo consultado em 2001.

[27] SCHULZRINNE, H. *et al*. "RTP: A Transport Protocol for Realtime Applications". IETF RFC 1889, jan. 1996.

[28] SCHULZRINNE, H., RAO, A., LANPHIER, R. "Real Time Streaming Protocol (RTSP)". IETF RFC 2326, april 1998.

[29] STEINMETZ, R. "Analyzing the Multimedia Operating System". **IEEE Multimedia**, v.2, p.68-84, 1995.

[30] STEINMETZ, R. "Human Perception of Jitter and Media Synchronization". **IEEE Journal on Selected Areas in Communications**, v.14, n.1, p.61-72, 1996.

[31] VERSCHEURE, O., JUBAUX, J.-P. "Perceptual Video Quality and Activity Metrics: Optimization of Video Service Based on MPEG-2 Encoding". In: INTERNATIONAL COST237 WORKSHOP, 3, 1996, Spain. **Springer Series LNCS 1185.**

[32] VOGEL, L. *et al.* "Distributed Multimedia and QoS: a Survey". **IEEE Multimedia**, p.10-19, Summer, 1995.

[33] W3C. **Extensible Markup Language (XML) 1.0 Specification**. W3C Recommendation, February 1998. Disponível na INTERNET via www.url:http://www.w3.org/TR/REC-xml. Arquivo consultado em 2000.

[34] W3C. HOFF, A. van, PARTOVI, H., THAI, T. **The Open Software Description Format (OSD)**. OSD Submission, August 1997. Disponível na INTERNET via www.url:http://www.w3.org/TR/NOTE-OSD.html. Arquivo consultado em 2001.

[35] W3C. **Synchronized Multimedia Integration Language (SMIL 2.0) Specification**. W3C Proposed Recommendation, June 2001. Disponível na INTERNET via www.url:http://www.w3.org/TR/smil20/. Arquivo consultado em 2001.

[36] W3C. **Synchronized Multimedia Integration Language (SMIL) 1.0 Specification**. W3C Recommendation, June 1998. Disponível na INTERNET via www.url:http://www.w3.org/TR/REC-smil. Arquivo consultado em 2000.

[37] WADDINGTON, D.G., HUTCHISON, D. "End-to-end QoS Provisioning through Resource Adaptation". In: IFIP CONFERENCE ON HIGH PERFORMANCE NETWORKING, 1998.

[38] WAHL, T., WIRAG, S., ROTHERNMEL, K. "TIEMPO: Temporal Modeling and Authoring of Interactive Multimedia". In: IEEE INTERNATIONAL CONFERENCE ON MULTIMEDIA COMPUTING AND SYSTEMS, 2, 1995. **Proceedings…** 1995. p.274-277.

[39] WIRAG, S. “Modeling of Adaptable Multimedia Documents”. In: INTERACTIVE DISTRIBUTED SYSTEMS AND TELECOMMUNICATION SERVICES, 4, 1997, Darmstadt, Germany. **Proceedings…** Darmstadt: Ralf; Wolf, Lars C., 1997. p.420-429.

[40] WOLF, L. C., GRIODZ, C., STEINMETZ, R. "Multimedia Communications”. **Proceedings of IEEE**, v.85, n.12, p.1915-1933, December 1997.

[41] CUNHA, E.C. **Uma Estratégia de Criação e Apresentação de Documentos Multimídia Adaptativos em Rede**. Orientador: Luiz Fernando Rust da Costa Carmo. Rio de Janeiro: UFRJ/IM/NCE, 2000, 84p. Dissertação. (Mestrado em Informática)

Publicações da Autora

GOMES, R. L., CUNHA, E. C., CARMO, L. F. R. C., PIRMEZ, L. "Assessment of a Distributed System for Processing Adaptive Multimedia Documents"" In: International Conference on Distributed Multimedia Systems, 2001, Taiwan. Proceedings of DMS'01 , 2001.

GOMES, R. L., CUNHA, E. C., CARMO, L. F. R. C., PIRMEZ, L. "An Adaptive Presentation System for Multimedia Documents" In: Seventh International Conference on Intelligent Multimedia and Distance Education, 2001, Fargo. Proceedings of ICIMADE'01 , 2001.

GOMES, R. L., CUNHA, E. C., CARMO, L. F. R. C., PIRMEZ, L. Avaliação de um Sistema Distribuído de Criação e Apresentação de Documentos Multimídia In: XIX Simpósio Brasileiro de Redes de Computadores, 2001, Florianópolis - SC. Anais do XIX Simpósio Brasileiro de Redes de Computadores, 2001.

GOMES, R. L., MENEZES, H. J., CARMO, L. F. R. C., PIRMEZ, L. Suporte à Apresentação Adaptativa de Aplicações Multimídia em Sistemas Distribuídos In: XVIII Simpósio Brasileiro de Redes de Computadores, 2000, Belo Horizonte. Anais do XVIII Simpósio Brasileiro de Redes de Computadores, 2000. v.1.