Promovendo Aplicações Multimídia Interativas com Autoria por Incorporação de Informações de Contexto

Erick Lazaro Melo

Promovendo Aplicações Multimídia Interativas com Autoria por Incorporação de Informações de Contexto[1]

Erick Lazaro Melo

Orientador: *Prof. Dr. Cesar Augusto Camillo Teixeira*

Tese apresentada ao Programa de Pós Graduação em Ciência da Computação da Universidade Federal de São Carlos (PPG-CC) como parte dos requisitos para obtenção do título de Doutor em Ciência da Computação.

São Carlos - SP
Outubro/2014

[1]Trabalho realizado com auxílio financeiro da CAPES.

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária/UFSCar

M528pa

Melo, Erick Lazaro.
Promovendo aplicações multimídia interativas com autoria por incorporação de informações de contexto / Erick Lazaro Melo. -- São Carlos : UFSCar, 2015.
225 f.

Tese (Doutorado) -- Universidade Federal de São Carlos, 2014.

1. Sistemas multimídia. 2. Computação ubíqua. 3. Sincronização. 4. Autoria multimídia. 5. Interação homem-máquina. I. Título.

CDD: 006.7 (20^a)

Universidade Federal de São Carlos
Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

"Promovendo Aplicações Multimídia Interativas com Autoria por Incorporação de Informações de Contexto"

Erick Lazaro Melo

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Doutor em Ciência da Computação

Membros da Banca:

[illegible]

Prof. Dr. Cesar Augusto Camillo Teixeira
(Orientador - DC/UFSCar)

[illegible]

Prof. Dr. Luis Carlos Trevelin
(DC/UFSCar)

[illegible]

Prof. Dr. Antonio Francisco do Prado
(DC/UFSCar)

André Santanchè.

Prof. Dr. André Santanchè
(UNICAMP)

[illegible]

Prof. Dr. Rudinei Goularte
(ICMC/USP)

São Carlos
Outubro/2014

Dedico este trabalho à Yara Kerber, minha companheira de todos os momentos!

Agradecimentos

- A Deus acima de tudo. Sou grato pelas bençãos derramadas em minha vida, pelo cuidado e por propiciar vida e saúde para que este trabalho pudesse ser desenvolvido.

- À minha família pela paciência e compreensão nos momentos de *stress*, pela ausência nas confraternizações familiares, pela falta de tempo para me dedicar mais a vocês. Agradeço imensamente aos meus pais pelos investimentos dados à minha educação. Eles foram fundamentais para que eu chegasse até aqui.

- À Yara, minha esposa e companheira, pela compreensão, apoio, paciência e palavras de motivação nos momentos difíceis.

- Ao Caio Viel, meu braço direito no desenvolvimento dos projetos. Não tenho nem palavras para expressar a minha gratidão a você.

- Aos meus companheiros do *Lince* pelo companheirismo. Foi fundamental o apoio de vocês! Agradeço especialmente aos alunos de iniciação científica que contribuiram diretamente para a realização deste trabalho.

- Ao meu orientador Cesar Teixeira pela paciência, companheirismo e sábios conselhos. O tempo que passamos juntos tem sido uma grande escola.

- À professora Maria da Graça Pimentel pela parceria na elaboração de artigos e escrita de projetos. As discussões e sugestões contribuíram muito neste trabalho e foram fontes de aprendizado e incentivo à pesquisa.

- Ao professor Luis Carlos Trevelin, que me proporcionou a oportunidade de atuar no projeto XPTA.Lab.

- Ao professor Celso Saibel, pelas ricas e calorosas discussões e contribuições.

- Aos meus colegas da Secretaria Geral de Informática, pelo apoio, por segurarem as pontas nos momentos em que precisei de dedicação integral à tese. Agradeço à minha equipe de trabalho: Bruno Lopes, Tadeu Borges e à minha chefe Cláudia Mello, que esteve sempre com uma palavra de motivação, mesmo nos momentos mais difíceis. Agradeço especialmente o professor Antônio Francisco do Prado, pelo incentivo a ingressar na carreira acadêmica ainda na graduação.

- Ao meu psiquiatra, que me forneceu o apoio necessário para superar os momentos mais difíceis da escrita da tese.

- Às instituições parceiras do *Lince* que contribuem com os nossos trabalhos. Agradeço especialmente às agências de financiamento: CAPES, CNPq, FINEP, RNP, FAPESP e Ministério da Cultura, que proveram recursos financeiros para que o trabalho pudesse ser desenvolvido.

E a todos que de alguma forma contribuíram para o sucesso deste trabalho. Muito obrigado!!!

Resumo

Aplicações multimídia interativas podem tornar conteúdos muito mais interessantes para os usuários, promover transmissão de informação mais efetiva, prender a atenção do usuário e eventualmente promover maior satisfação. Entretanto, observa-se que a disseminação desse tipo de aplicação é muito tímida. Conteúdo multimídia, de maneira geral, tem se resumido a vídeo e documentos hipermídia baseados em imagem e texto. São poucas as iniciativas de interatividade efetiva com o conteúdo que se beneficiam da sincronização de mídias. O ambiente da TV Digital Interativa apresenta bom potencial para exploração do sincronismo de mídia, entretanto observam-se inúmeros obstáculos decorrentes do conservadorismo do setor. As poucas opções disponíveis aos telespectadores acabam sendo muitas vezes ignoradas. Algumas razões podem explicar a baixa adoção de aplicações multimídia interativas:(i) a produção desse tipo de conteúdo é complexa - as ferramentas de autoria pouco contribuem para simplificar esse processo, em geral afastando *designers* e desenvolvedores não especializados, que optam por utilizar conteúdos capturados de maneira simples (geralmente vídeo); (ii) a apresentação de conteúdo multimídia interativo pode requerer ambientes ou aplicações específicas não muito disseminadas ou que necessitam desenvolvimento especializado; (iii) é desejável que o consumo de conteúdo ocorra em múltiplas plataformas, tornando custoso o desenvolvimento específico para cada uma; (iv) não se conhecem aplicações que sejam inovadoras e que realmente estimulam a interatividade. Acredita-se que esse cenário pode ser mudado e que aplicações multimídia interativas possam contribuir com a satisfação do usuário. Tem-se como hipótese que aplicações multimídia interativas e potencialmente atrativas podem ser construídas com a incorporação automática ou semiautomática, a uma mídia principal, de outras mídias a ela relacionadas. As mídias relacionadas podem ser resultado de autoria humana, com apoios facilitadores, da captura de informações de contexto, de mineração da interação coletiva em sessões de apresentação com multiespectadores ou combinações dessas fontes. Fundamentando-se em artigos publicados, com autoria ou coautoria do autor desta tese e relacionados a projetos por ele gerenciados, apresenta-se neste texto os esforços realizados e os resultados que contribuem para a validação da hipótese. São trabalhos que abordam aspectos de autoria multimídia, da difusão de aplicações multimídia em múltiplas plataformas e da sincronização de mídias, todos inerentes a aplicações inovadoras com potencial de cativar usuários e servirem como catalisadores de novas aplicações.

Palavras-chave: Computação Ubíqua, Multimídia, Sincronização, Autoria, Ad-Avoidance, Interação

Abstract

Interactive multimedia applications can turn contents into a much more interesting presentation for users, promote more effective transmission of information, hold the user's attention and eventually promote more satisfaction. However, it is observed that the spread of such applications is very timid. Multimedia content, in general, has been summarized to video and hypermedia documents based on images and text. There are few effective initiatives interactivity with content that benefit from media synchronization. The environment of the Interactive Digital TV offers good potential for exploitation of sync media, however we observe numerous obstacles arising from the conservatism of the industry. The few options available to viewers often end up being ignored. Some reasons may explain the low adoption of interactive multimedia applications: (i) the production of such content is complex - existing authoring tools contribute little to simplify this process, keeping away unskilled designers and developers who choose to use content that can be captured in a simple way (usually video); (ii) the presentation of interactive multimedia content may require specific environments or applications not very widespread or require specialized development; (iii) the content consumption should occur on multiple platforms, making it expensive to develop specific applications for each; (iv) there is a lack of applications widely known to the general public that are really attractive and innovative and that stimulate interactivity. It is believed that this scenario can be changed and interactive multimedia applications can contribute to user satisfaction. It has been hypothesized that attractive interactive multimedia applications can be built with automatic or semiautomatic incorporation, to a leading media, of other media related to it. Related media may be the result of human authorship, with facilitators support, capturing context information, mining the collective interaction in presentation sessions with multiple viewers or combinations of these sources. Based on several published articles, authored or co-authored by the author of this thesis and related to projects managed by him, this text presents efforts and results that contribute to the validation of the hypothesis. These are works that address aspects of multimedia authoring, dissemination of multimedia applications on multiple platforms and media synchronization, all inherent to innovative applications with strong potential to captivate users and serve as catalysts for new applications.

Keywords: Ubiquitous Computing, Multimedia, Synchronization, Multimedia Authoring, Ad-Avoidance, Interaction

Lista de Figuras

Lista de Tabelas

Sumário

CAPÍTULO 1

Introdução

Aplicações multimídia interativas podem tornar conteúdos muito mais interessantes para os usuários, promover transmissão de informação mais efetiva, prender a atenção do usuário e eventualmente promover maior satisfação. Entretanto, observa-se que a disseminação desse tipo de aplicações é muito tímida.

Conteúdo multimídia, de maneira geral, tem se resumido a vídeo e documentos hipermídia baseados em imagens e texto. São poucas as iniciativas de interatividade efetiva com o conteúdo que se beneficiam da sincronização de mídias.

Algumas razões podem explicar a baixa adoção de aplicações multimídia interativa:

- a produção desse tipo de conteúdo é complexa - as ferramentas de autoria existentes pouco contribuem para simplificar esse processo, em geral afastando *designers* e desenvolvedores não especializados, que optam por utilizar conteúdos capturados de maneira simples (geralmente vídeo), os quais dispõem de amplas opções de ferramentas de apoio;

- a apresentação de conteúdo multimídia interativo pode requerer ambientes ou aplicações específicas não muito disseminadas ou que demandem desenvolvimento especializado;

- o consumo de conteúdo deveria ocorrer em múltiplas plataformas; tornando-se restrito pelos custos do desenvolvimento específico para cada uma;

- não se conhecem aplicações que sejam inovadoras e que realmente estimulam a interatividade.

Neste capítulo são apresentadas as motivações desta tese, os projetos de P&D desenvolvidos e que serviram de justificativas para os estudos, os objetivos da tese e a organização deste texto.

1.1 Motivação

A atuação do autor desta tese na área de Sistemas Multimídia teve início com o projeto *Avaliação do Middleware Ginga*, em que foram conduzidos estudos para avaliar a proposta de *middleware* para TV Digital Interativa adotada pelo Sistema Brasileiro de TV Digital (SBTVD). As investigações tiveram como panorama a utilização do *middleware* para a execução de programas educacionais e o seu suporte à construção de aplicações com demandas de personalização de conteúdo.

Como resultado desse trabalho observou-se a carência de elementos no *middleware* que suportassem a construção de aplicações mais sofisticadas, além daquelas relacionadas à sincronização de mídias. Observou-se também, que a autoria de aplicações multimídia ainda se tratava de processo complexo e que poderia ser facilitada com a utilização de ferramentas mais apropriadas que pudessem promover a autoria de maneira automática ou semiautomática. Isso levou o grupo de pesquisa envolvido no projeto a se associar a outros grupos e propor ao CTIC/RNP[1] um projeto que contemplasse o desenvolvimento de extensões ao *middleware* Ginga. Os componentes propostos na extensão visavam facilitar a construção de aplicações para autoria e anotação colaborativa de conteúdo no lado do cliente, além de fornecerem recursos para a construção de aplicações com demanda por personalização (sistemas de recomendação) e integração com outros dispositivos de redes domésticas. Além disso foi proposto, no projeto, um conjunto de boas práticas para avaliação de acessibilidade de conteúdo e da interação para TV digital. Esse projeto foi aprovado e se fundiu a outros 2 projetos submetidos por outros grupos, resultando no projeto *GingaFrEvo & GingaRAP* [2].

O desenvolvimento do projeto obteve êxito, sendo entregues diversos módulos integrados à implementação de referência do *middleware* Ginga. Esses resultados chegaram a ser apresentados a representantes da indústria em um evento promovido pela RNP e no maior evento/feira nacional de equipamentos

[1]Centro de Pesquisa e Desenvolvimento em Tecnologias Digitais para Informação e Comunicação - https://www.ctic.rnp.br/
[2]http://www.ctic.rnp.br/web/ctic/gingarap-gingafrevo

e serviços da área de TV, o *SET/BroadCast & Cable*. Entretanto observou-se pouco interesse da indústria nacional em inovar na área de TV Interativa.

O fato é que a interatividade na TV tem progredido de forma muito lenta. Os fabricantes de TVs e *set-top-boxes* (conversores de sinal digital para TVs analógicas) têm resistência em incorporar o *middleware* Ginga aos seus equipamentos. Por outro lado, as emissoras de TV não têm demonstrado interesse na produção de conteúdo interativo, talvez com receio de introduzir ruídos em seu modelo de negócio bastante clássico e sob controle. Nesse mercado o número de *players* é bastante reduzido e os fabricantes não oferecem facilidades para que desenvolvedores independentes disponibilizem aplicações interativas para execução em seus equipamentos.

Pode-se constatar também que os recursos trazidos pelo *middleware* Ginga, especificamente a linguagem **NCL**(*Nested Context Language*) e sua máquina de apresentação, permitem a construção de aplicações multimídia bastante sofisticadas, que podem ser aplicadas a outros domínios além do ambiente de TV. A linguagem permite que a especificação de apresentações multimídia com relações de sincronismo complexas possa ser feita de maneira simples.

Essa situação gerou uma certa frustração nos pesquisadores associados ao grupo de pesquisa em que os projetos foram desenvolvidos (LINCE[3]), mas motivou a busca por alternativas para alavancar a interatividade na TV e também aproveitar os recursos propostos no *middleware* Ginga em outros domínios além da TV e do SBTVD.

Esse interesse pela busca de alternativas para melhor promover a interação do usuário com aplicações multimídia potencialmente atrativas e em diferentes plataformas motivou o desenvolvimento desta tese. A percepção de que: **aplicações multimídia interativas atrativas podem contribuir com a satisfação do usuário e que; a autoria dessas aplicações pode-se dar com base em um processo automático ou semi-automático, a partir de uma mídia principal e a ela se incorporando novas mídias com informações contextuais** motivaram pesquisas visando:

1. O desenvolvimento de solução viável para a disponibilização de apresentações multimídia em ambientes multi-plataforma, que possam ser utilizadas tanto embarcadas em *middlewares* de TV Digital, como em ambientes totalmente dissociados da TVDi (web, dispositivos móveis);

2. O desenvolvimento de novos mecanismos de sincronização multimídia, que possam ser aplicados a ambientes distribuídos e com redes de dis-

[3]Laboratório para Inovação em Computação e Engenharia, da Universidade Federal de São Carlos

tribuição heterogêneas;

3. O desenvolvimento de ferramentas de autoria que permitam a criação de aplicações multimídia interativas com a incorporação de conteúdo complementar àquele apresentado na TV. O objetivo é que o conteúdo complementar possa apoiar tanto portadores de necessidades especiais (com o fornecimento de conteúdo direcionado à cada necessidade específica), quanto enriquecer o conteúdo de massa apresentado (considerando a incipiência de conteúdo interativo disponível);

4. O desenvolvimento de aplicação interativa para TV que tenha potencial apelo para incentivar a sua adoção e com isso também alavancar outras aplicações interativas.

Os desafios propostos demandaram um grande volume do trabalho, especialmente nas atividades de codificação de *software*. Foram buscadas fontes de financiamento para as pesquisas, que se estruturaram em projetos de pesquisa. O autor desta tese colaborou diretamente na escrita e submissão desses projetos de pesquisa.

Os projetos submetidos se desdobraram em pesquisas mais específicas, objeto de outros trabalhos de titulação e iniciações científicas. O autor desta tese colaborou com esses trabalhos, exercendo um papel de arquiteto de *software*, bem como atuando como co-orientador dos trabalhos. Esses trabalhos resultaram em produção científica e tecnológica. Essa colaboração se deu em função do papel exercido pelo autor de coordenador técnico do Lince desde 2008.

Nesta tese são apresentados os resultados específicos relacionados à contribuição com a satisfação do usuário através de aplicações multimídia interativas, tanto no ambiente de TV quanto em outros, como na *Web*. São tratados aspectos que permitem os processos de autoria de aplicações multimídia através da incorporação automática ou semiautomática de mídias, com informações de contexto, a uma mídia principal.

Ciência de contexto é uma área bastante investigada na Ciência da Computação (Abowd et al., 1999; Harter et al., 2002; Dey, 2001) e com definições bastante amplas. Uma definição bem aceita, proposta por Dey descreve contexto como:

> "Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an ap-

plication, including the user and the application themselves."(Dey, 2001)

No escopo desta tese entende-se contexto como o conteúdo associado a uma mídia principal (*what*) no espaço de tempo com valor semântico (*when*) e no dispositivo adequado para o usuário (*how*).

Alguns resultados desses trabalhos já encontram-se publicados em veículos de divulgação científica. As publicações mais relevantes obtidas encontram-se transcritas ao longo desta tese e são partes integrantes da mesma. São também apresentados comentários sobre essas publicações, destacando-se a relação entre elas e como os resultados contribuem com os objetivos da tese. Tópicos não abordados nos artigos mas relevantes à tese também são apresentados. Optou-se por preservar os textos já publicados na sua íntegra, respeitando o idioma de publicação. Entende-se ser essa, forma bastante apropriada e fiel para transmitir o trabalho.

O desenvolvimento do trabalho contou com uma equipe que apoiou as atividades e dedicou atenção a atividades de codificação. O autor desta tese atuou coordenando todas essas atividades, bem como executando algumas pessoalmente.

1.2 Projetos

Os projetos desenvolvidos tiveram como objetivo comum a busca do enriquecimento de conteúdo multimídia e por consequência uma melhor experiência para usuários finais. Os projetos foram desenvolvidos em cascata, em que cada projeto ofereceu oportunidades e desafios para o desenvolvimento de novas pesquisas. A Figura 1.1 apresenta uma visão do desencadeamento dos projetos.

1.2.1 Avaliação do Middleware Ginga - Aprendizado e Personalização

As pesquisas realizadas no projeto resultaram em: novos módulos que podem ser incorporados ao *middleware* Ginga ou mais apropriadamente comporem camada de mais alto nível; desenvolvimento de ferramental de apoio à construção de aplicações de aprendizado eletrônico; concepção e implementação de uma aplicação que explora os novos módulos e o ferramental desenvolvidos. O autor atuou como arquiteto de software e gerente do projeto, coordenando equipe de desenvolvimento. O projeto foi financiado pela FINEP.

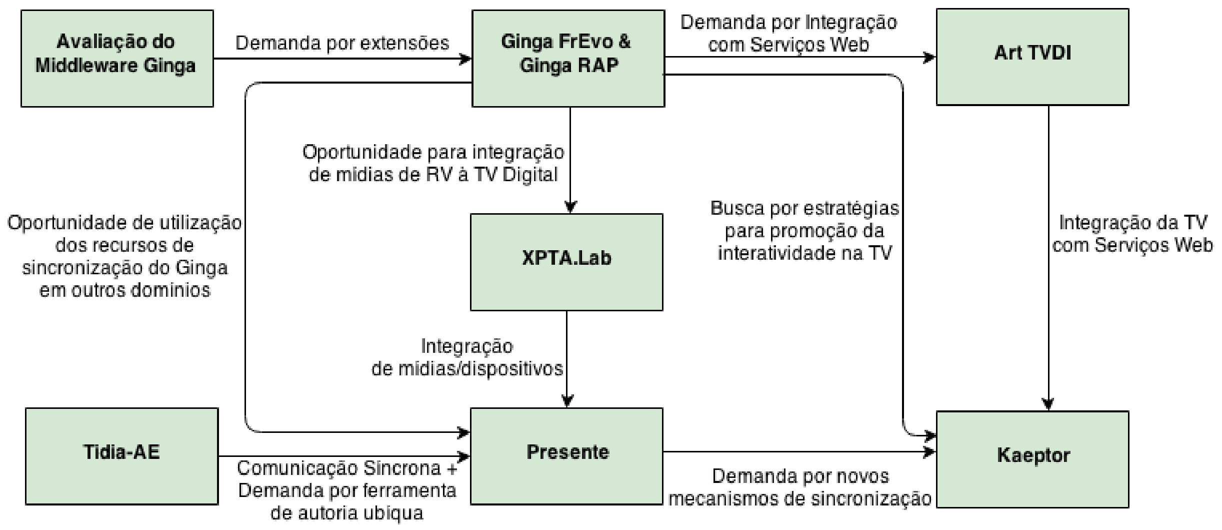

Avaliação do Middleware Ginga
Demanda por extensões
Ginga FrEvo & Ginga RAP
Demanda por Integração com Serviços Web
Art TVDI
Oportunidade para integração de mídias de RV à TV Digital
Oportunidade de utilização dos recursos de sincronização do Ginga em outros domínios
XPTA.Lab
Busca por estratégias para promoção da interatividade na TV
Integração da TV com Serviços Web
Integração de mídias/dispositivos
Tidia-AE
Comunicação Síncrona + Demanda por ferramenta de autoria ubíqua
Presente
Demanda por novos mecanismos de sincronização
Kaeptor

Figura 1.1: Fluxo dos Projetos Desenvolvidos

1.2.2 Tidia-Ae - Comunicação Síncrona e Integração de Laboratórios Remotos

A participação do Lince/UFSCar nesse projeto colaborativo foi dirigida a quatro tipos de atividades: Pesquisa explorativa e desenvolvimento em ciência da computação nos assuntos pertinentes ao projeto; Pesquisa e desenvolvimento de ações de aprendizagem; Colaboração com laboratórios associados; Contribuição no desenvolvimento da infraestrutura do TIDIA-Ae dando continuidade ao desenvolvimento de ferramentas modeladas na fase 1. O autor atuou no projeto apoiando o desenvolvimento de ferramentas para Web, principalmente as focadas em comunicação síncrona de baixa latência baseada em vídeo e áudio. O projeto foi financiado pela FAPESP.

1.2.3 XPTA.lab - Virtualidade Imersiva e interativa baseada em cloud computing

O projeto se propôs à experimentação do desenvolvimento de tecnologias para suporte a conteúdos distribuídos multiusuários de realidade virtual e aumentada para TV digital, dispositivos móveis e PCs. O autor atuou no projeto como arquiteto das soluções empregadas no projeto relacionadas a TV Digital e dispositivos móveis, bem como coordenou o desenvolvimento de artefatos de *software*. Este projeto foi financiado pelo Ministério da Cultura.

1.2.4 GingaFrEvo & GingaRAP

Foi um projeto colaborativo desenvolvido em uma rede de pesquisa envolvendo 20 instituições de ensino. O autor atuou no projeto gerenciando o desenvolvimento das atividades técnicas da sub-rede envolvendo ICMC-USP, UFJF, UFU, UNIFEI e FUCAPI. Nessa sub-rede realizou-se: desenvolvido um conjunto de módulos para autoria e anotação colaborativa de conteúdo no lado do cliente, integrado a outras ferramentas; desenvolvimento de um conjunto de boas práticas para avaliação de acessibilidade de conteúdo e da interação para TV digital e; a especialização do Ginga-CC para aplicações não convencionais focadas na integração com dispositivos de redes domésticas (como *smartphones, tablets*). O projeto foi financiado pela RNP (CTIC).

1.2.5 Art TVDI – Arcabouço Tecnológico para Desenvolvimento Ágil e Reutilização de Aplicações para TVDI com suporte a Serviços Web

O projeto apresenta um Arcabouço Tecnológico para Desenvolvimento Ágil e Reutilização de Aplicações para TVDI com suporte a Serviços Web. O subprojeto InserTV, de responsabilidade do núcleo em que a UFSCar participa, propôs-se a apresentar soluções para permitir a integração de serviços Web aos conteúdos transmitidos aos receptores do sinal de TV digital. O autor atuou na especificação dos artefatos de software e na implementação de protótipos. O projeto foi financiado pelo CTIC/RNP.

1.2.6 PRESENTE – Produção e Apresentação de Objetos Educacionais Orientadas pelo Paradigma da Aula Presencial e pela Computação Ubíqua

O objetivo deste projeto foi desenvolver, avaliar e refinar um sistema composto por ferramentas computacionais para facilitar os processos de produção e apresentação de objetos educacionais multimídia. Os objetos são baseados no paradigma da aula presencial e o processo de produção orientado pela computação ubíqua. Câmeras de alta definição, microfones, sensores de telas e softwares de reconhecimento capturam a experiência da aula de forma ubíqua sem necessidade de técnicos para auxílio. O resultado da captura são diversas "streams" de dados de diferentes mídias enriquecidas com metadados propiciados pelos sensores e softwares de reconhecimento. Uma ferramenta integra os dados em um objeto multimídia. Uma interface gráfica permite que alunos possam “navegar” pela apresentação multimídia, procurando reproduzir a experiência da aula de acordo com seus interesses no momento. O autor atuou nas etapas de proposição do projeto, desenvolvimento de software e testes. Este projeto foi financiado pela SEaD/UFSCar (Secretaria Geral de Educação a Distância) em parceria com a CAPES.

1.2.7 *Kaeptor*

Observa-se um fenômeno de crescimento em larga escala da utilização de dispositivos móveis (*smartphones* e *tablets*). Observa-se, ainda, que é cada vez mais comum as pessoas assistirem TV acompanhadas dos seus dispositivos móveis e produzindo conteúdo atrelado aos programas de TV em redes sociais. Isso pode ser observado através dos *trending topics* to Twitter, nos quais é frequente a presença de comentários sobre programas de TV (Digital, 2012).

Diante da resistência dos fabricantes de TVs e *set-top-boxes* em abrir espaço para o ingresso de novos *players* (desenvolvedores independentes que ofereçam aplicações interativas) foi proposto neste projeto um novo modelo de difusão de aplicações interativas. Esse modelo baseia-se na utilização do dispositivo móvel para a difusão de aplicações interativas sincronizadas com o conteúdo da TV, integradas a redes sociais.

O projeto demandou novos mecanismos de sincronização, já que não existe um forte acoplamento entre as mídias (como no caso de transmissões terrestres de TV), e as mesmas encontram-se em diferentes redes de distribuição, sujeitas a latências variadas.

O projeto introduz uma aplicação do tipo *ad-avoidance* como meio de promoção da interatividade. Baseia-se na ideia de entrega de um produto com valor para o usuário que estimule a interatividade com o conteúdo da TV. Esse projeto agrega o resultado de diversas pesquisas conduzidas pelo autor e descritas ao longo da tese. Essa proposta é detalhada no Capítulo 5.

O projeto demandou uma atenção direta do autor da tese, que foi o responsável pelas fases de concepção, com base nas experiências e conhecimentos observados nos demais projetos. As atividades realizadas neste projeto incluíram desde a discussão de modelos de negócio e aplicações que pudessem ser potencialmente atrativa aos usuários, passando pela modelagem dos artefatos de *software*, sua implementação e testes. Foi utilizado um conjunto de diferentes tecnologias, desde programação de baixo nível para comunicação direta com *hardware*, passando pelo desenvolvimento de aplicações *desktop*, web, TV Digital e dispositivos móveis.

1.3 Objetivos

O objetivo principal do trabalho é apresentar indícios, fortemente fundamentados em projetos desenvolvidos, que aplicações multimídia interativas e potencialmente atrativas podem ser construídas com a incorporação automática ou semiautomática, a uma mídia principal, de informações de contexto representadas em diferentes mídias. As mídias incorporadas podem ser resultado de autoria humana, com apoio de facilitadores, da captura automática de informações de contexto, de mineração da interação coletiva em sessões de apresentação com multiespectadores, de mineração em base de dados, ou combinações dessas fontes.

Destacam-se também alguns objetivos secundários, que caracterizam as diversas etapas que foram necessárias no trabalho. Essas etapas referem-se a aspectos de áreas como autoria multimídia, difusão de aplicações multimídia em múltiplas plataformas e sincronização de mídias:

1. Propor e explorar ferramentas para facilitar a autoria de conteúdo multimídia, tanto de maneira automática quanto de maneira semi-automatizada;

2. Propor e explorar novos mecanismos de sincronização multimídia, que não estejam atrelados a linhas temporais;

3. Propor e explorar uma máquina de apresentações multimídia multiplataforma, capaz de ser executada em qualquer *browser* com suporte a HTML5 (*smartphones, tablets, smartvs, PCs*);

4. Propor e explorar uma ferramenta da classe das "ad-avoidance"que visa promover a interatividade na TV, atuando como um catalisador de outras aplicações interativas.

1.4 Organização da Tese

A tese está organizada em capítulos que detalham os problemas encontrados em cada uma das áreas apontadas, que precisaram ser exploradas para se alcançar o objetivo principal. O levantamento bibliográfico correspondente a cada etapa encontra-se descrito em seu respectivo capítulo. No modelo de tese adotado são utilizados artigos científicos, de coautoria do autor da tese, para apresentar algumas discussões acerca dos temas propostos e apresentar os resultados obtidos. Ao fim de cada capítulo são tecidas considerações sobre os artigos e como se relacionam com os objetivos da tese e demais capítulos.

No **Capítulo 2** abordam-se as questões relativas à autoria de conteúdo multimídia. Nele é apresentada uma abordagem para a discriminação de momentos e segmentos de mídia, parte do processo de autoria multimídia. Ainda, nesse capítulo são apresentados uma proposta de captura e autoria automática de apresentações multimídia no ambiente educacional, a partir da captura de informações contextuais e; um estudo que propõe o enriquecimento de linguagens de sincronização multimídia com base em componentes de software que podem tornar o processo de autoria desses documentos mais simplificado. Essas pesquisas tiverem como resultado inovações que tornam o processo de autoria de apresentações multimídia interativas mais acessível aos usuários/autores.

O **Capítulo 3** explora os aspectos da pesquisa com relação a promoção de maior difusão e portabilidade de aplicações multimídia. Apresenta-se a proposta e os desafios de desenvolvimento de uma plataforma capaz de orquestrar a apresentação de documentos multimídia em diversas plataformas e sistemas operacionais.

No **Capítulo 4** apresentam-se as propostas e os desafios de desenvolvimento de novos mecanismos de sincronização multimídia capazes de promover a sincronização em ambientes distribuídos e que não dispõem de uma linha temporal comum entre as máquinas de apresentação.

No **Capítulo 5** é proposto e apresentado um modelo de aplicação baseado em *Ad-Avoidance*, que incorpora conteúdo complementar sincronizado àquele apresentado na TV, inclusive aos comerciais. Espera-se que com uma aplicação de grande potencial de atratividade, por trazer benefícios evidentes para telespectadores e anunciantes, a interatividade possa ser estimulada tanto para essa como outras aplicações.

Por fim, no **Capítulo 6** são apresentadas as conclusões, enumeradas as principais contribuições decorrentes do trabalho desenvolvido no escopo desta tese e indicados alguns trabalhos futuros, relacionados a esta tese, com potencial de desenvolvimento de novas pesquisas.

CAPÍTULO
2

Autoria de Conteúdo Multimídia

Pesquisadores da área de computação ubíqua investigam técnicas para prover serviços para usuários de forma transparente, sem que os mesmos tenham que se preocupar com procedimentos técnicos para a execução desses serviços.

Com base nos conceitos de captura e acesso (Abowd et al., 2002), propõem-se, em Pimentel et al. (2007b), que a experiência do usuário ao assistir um vídeo possa ser capturada. A experiência do usuário, através da sua interação com um vídeo, pode ser aproveitada na criação de novos documentos multimídia, que podem ser de interesse do próprio usuário ou de outras pessoas. O princípio é que interações do usuário possam promover anotações não intrusivas sobre o vídeo original, resultando em outro documento (vídeo anotado).

Os estudos nessa área foram aprofundados, culminando com o estabelecimento do paradigma *watch-and-comment* (Cattelan et al., 2008). No paradigma *watch-and-comment*, ou WaC, são estabelecidos diversos princípios dos quais destacamos os seguintes: inexistência de restrições quanto à fonte do vídeo (o vídeo pode ser capturado ao vivo através de uma câmera, estar armazenado em um arquivo local ou acessível remotamente); inexistência de restrições quanto à linguagem do documento resultante; o documento declarativo resultante deve manter separadas da mídia original as anotações realizadas; a sessão pode ser colaborativa, síncrona e distribuída; a forma de captura da interação e a semântica não são restritas.

Abordando o que se percebe no cotidiano das pessoas pode-se observar que realizar anotações e fazer comentários sobre conteúdos são atividades bastante frequentes. Observa-se também que as facilidades tecnológicas têm viabilizado o oferecimento de serviços cada vez mais personalizados, aderentes aos perfis, gostos e preferências das pessoas, na busca de promover maior satisfação aos clientes.

A viabilização da autoria de conteúdo personalizado de forma ubíqua, aproveitando-se das anotações, comentários e interações que pessoas fazem naturalmente sobre um conteúdo base, pode contribuir com o aumento da satisfação que essas pessoas, ou terceiros, terão ao rever/interagir com o documento resultante.

Quando se fala aqui em autoria, não se trata de processos tradicionais de autoria de conteúdo que, geralmente, requerem o conhecimento de ferramentas dominadas apenas por especialistas (principalmente se o conteúdo tratar-se de vídeo; ou ainda de vídeo interativo). Reitera-se que a autoria a que se refere este trabalho baseia-se na captura de interações realizadas, em grande parte, de maneira natural por usuários, telespectadores.

A flexibilização do que se refere por interação natural, ou seja, a inclusão de interações simples, porém realizadas de forma intencional para se obter uma anotação mais refinada e maior especificidade no documento que se cria, pode estender a aplicabilidade do paradigma WaC também a profissionais de mídia. Uma ferramenta simples de autoria, fundamentada em simples anotações e comentários, pode ser utilizada por diretores de cinema, vídeo ou TV para promover edições de alto nível em material básico, a serem realizadas eventualmente *a posteriori* por técnicos, com o uso de ferramentas especializadas.

Os trabalhos desenvolvidos relacionados a autoria de aplicações multimídia com base em processos automático e semi-automático, a partir de uma mídia principal e a ela incorporando-se novas mídias com informações contextuais, estão relatados neste capítulo.

A Sessão 2.1 refere-se à transcrissão do artigo **End-user live editing of iTV programs**, publicado no *International Journal of Advanced Media and Communication* (2010), de autoria de Maria da Graça Campos Pimentel, Renan G. Cattelan, **Erick Lazaro Melo**, Antônio Francisco do Prado e César Augusto Camillo Teixeira. Nesse artigo são discutidos os problemas relacionados à engenharia de documentos multimídia interativos, incluindo aqueles relativos à autoria de documentos multimídia em fase de exibição e interação (edição ao vivo). São propostas soluções relacionadas à engenharia de documentos, interação e casos de uso de autoria ubíqua, baseados na interação do usuário. É proposto um *framework* para apoiar a construção de aplicações com foco na autoria de documentos multimídia voltados à área de TV Digital Interativa, baseado na implementação de referência do *middeware* Ginga. Esse trabalho contribui com o objetivo da tese ao explorar mecanismos para a autoria automática de aplicações multimídia, permitindo que essa autoria se dê de maneira ubíqua e esteja acessível ao usuário final (telespectador), através de comentários e anotações.

A Seção 2.2 refere-se à transcrição do artigo **Discrimination of media moments and media intervals: sticker-based watch-and-comment annotation**, publicado no periódico *Multimedia Tools and Applications* (2012). Esse artigo é de autoria de César Augusto Camillo Teixeira, **Erick Lazaro Melo**, Giliard Brito Freitas, Celso Alberto Saibel Santos e Maria da Graça Campos Pimentel. Nele é apresentada a proposta de um mecanismo para discriminação de momentos e intervalos em mídia. É proposta uma extensão ao paradigma *watch-and-comment*, permitindo a usuários de TV Digital Interativa a discriminação de momentos e intervalos nas mídias através de dispositivos móveis pessoais e a autoria ubíqua de documentos baseado em anotações através de *stickers*. Esse trabalho contribui com a tese ao experimentar a autoria de conteúdo complementar baseado nas interações do usuário através de dispositivos móveis e com a definição de operadores para discriminação de momentos, úteis para a identificação do contexto da interação do usuário. O trabalho apontou indícios para a necessidade de mecanismos de sincronização não convencionais para a incorporação de conteúdo complementar (personalizado ou não) a apresentações multimídia, objeto das pesquisas relatadas no Capítulo 4.

No decorrer do trabalho foi observado que a autoria ubíqua de documentos multimídia poderia ser aplicada a outros domínios além da TV Digital Interativa, em ambientes com uma riqueza ainda maior de informações contextuais que podem agregar valor a uma apresentação multimídia. Isso motivou a proposta e desenvolvimento de uma ferramenta de autoria capaz de gerar um documento multimídia, de maneira automática, a partir da captura de informações contextuais. Essa ferramenta foi desenvolvida no escopo do projeto Presente e avaliada no cenário educacional. Os resultados obtidos nesse trabalho foram positivos ao demonstrar a viabilidade da autoria de aplicações de maneira totalmente automatizada através da captura de informações contextuais. Esse processo de autoria é descrito na Sessão 2.3 através do artigo **Multimedia multi-device educational presentations preserved as interactive multi-video objects**, de autoria de Caio Cesar Viel, **Erick Lazaro Melo**, Maria da Graça Pimentel e Cesar Augusto Camillo Teixeira, publicado nos anais do *Simpósio Brasileiro de Sistemas Multimídia e Web - Webmedia* (2013).

Na Seção 2.4 são apresentados aspectos técnicos relacionados à autoria de aplicações multimídia que demandam a manipulação e sincronização de mídias, como aquelas que demandam uma *timeline* (possibilidade do usuário avançar e retroceder a apresentação através de uma navegação temporal). É apresentada uma proposta que busca facilitar o processo de construção de aplicações multimídia por desenvolvedores de *software*. O conteúdo desta seção refere-se ao artigo **Go beyond boundaries of iTV applications**, publicado no *ACM Symposium on Document Engineering* (2013), em que autores são Caio César Viel, **Erick Lazaro Melo**, Maria da Graça Campos Pimentel e César Augusto Camillo Teixeira. Embora esse trabalho não esteja relacionado diretamente com o objetivo da tese, a proposta contribui ao facilitar a autoria de aplicações multimídia (incluindo aquelas de conteúdo complementar à mídia principal da TV).

Finalmente, na Seção 2.5, são apresentadas considerações relacionadas às principais contribuições trazidas na área de autoria de conteúdo multimídia e como estas corroboram com o objetivo da tese.

2.1 End-user live editing of iTV programs

Watching TV is a practice many people enjoy and feel comfortable with. While watching a TV program, users can be offered the opportunity to, while making annotations, create their own edited versions of the program. In this scenario, it is a challenge to allow the user to add comments in a ubiquitous, transparent way. In this paper we exploit the concept of end-user live editing of interactive video programs by detailing an environment where users are able to live edit a video using the iTV remote control. We contextualize our approach in the context of the Brazilian Interactive Digital TV platform.

2.1.1 Introduction

The area of ubiquitous computing investigates alternatives for providing services to users in a transparent way, without taking their attention from their main task (Weiser, 1991). Considering the three recurring and complementary themes in ubiquitous computing research — natural interfaces, context awareness, and capture and access applications — the composite term *capture and access* refers to *the task of preserving a record of some live experience that is then reviewed at some point in the future* (Abowd et al., 2002).

After exploiting the *capture and access* concepts of ubiquitous computing, we have proposed the automatic *capture* of the user interaction while watching a video stream in a device such as a laptop computer (Pimentel et al., 2007b). The aim is that the captured interaction be used to automatically construct an interactive video corresponding to the user experience – and the resulting interactive video can be *accessed* for review. As an alternative to tradicional approaches in which the author employs editing tools to create interactive multimedia documents (e.g. (Guimarães et al., 2008)), this approach results in empowering the viewer (César et al., 2006b) (César et al., 2008).

We have previously explored such ideas by defining (Pimentel et al., 2007b) and demonstrating (Maria da Graça et al., 2008) the *Watch-and-Comment* (WaC) paradigm in which a user-watching TV session is transparently captured so that natural annotations (e.g. voice or electronic ink comments) and pen-based annotations are integrated to automatically generate a corresponding interactive video document. The interactive video must combine the original (video) media with the comments in a properly synchronized interactive multimedia document. As a result, when the interactive video is later watched, the edits and annotations can be made available along with the video.

We have extended those ideas focusing on collaboration and distribution issues: we employ annotations as simple containers for context information by using them as tags in order to organize, store and distribute information in a P2P-based multimedia capture platform (Cattelan et al., 2008).

We built a prototype application WACTOOL which, when running on a computer based-platform, supported the capture of digital ink and voice comments over individual frames and segments of the video, producing a structured XML-based document that synchronizes the different media streams. Such an approach takes advantage of the fact that capture devices are available in most computer-based platforms: microphones and keyboards may be used to capture audio and text in most computers and smart phones, for instance. Moreover, pen-based devices such as PDAs and tablet PCs are becoming popular. If the user chooses to watch the video using a pen-based device, a tap with the digital pen on the video window may cause the video to be paused so that the user can, for instance, make notes on the image using digital ink at the same time that a voice comments is captured.

Our work (Pimentel et al., 2007b) (Maria da Graça et al., 2008) allowed the generation of documents in SMIL (Bulterman et al., 2005) and NCL (Silva et al., 2004) — the latter adopted in the Brazilian Digital TV System (SBTVD) as the official standard for declarative[1] interactive programs. From the digital rights perspective, the approach demands that edits and annotations be kept separate from the original media, which means that they can be distributed

[1]Ginga-NCL: http://www.ncl.org.br, visited on March 13th, 2009.

independently from the video stream.

In a complementary approach, we have also proposed capturing the user-remote control interaction and treating the captured information as contents pervasively generated by the user. Such content contains detailed information relative to the user experience that is most valuable to many applications and services (Baladrón et al., 2008).

Leveraging the ubiquitous existence of remote controls for viewer-TV interaction, in earlier work we have briefly presented (Pimentel et al., 2008b) our results relative to extending and integrating the aforementioned efforts to allow the end-user transparent live editing of interactive TV programs at the time of broadcast, including a scenario where the user is able to live edit an ongoing soccer game (Pimentel et al., 2008a). We detail in this paper the design of an application that supports users editing a structured interactive multimedia document by means of the concept of *end-user changing the watching experience* — which means that we allow end-user edits to affect not only the parts of the program a viewer has already watched, but also contents not yet presented.

It is important to observe that we support changes *not* planned by the author and already coded in the original (broadcast) program. Allowing the end-user to modify the multimedia document in "real-time" means that the viewer edit commands can affect not only parts of the program already viewed but also contents not yet presented, which should allow changes not specified by the original author. This relates to CSS style sheets in the sense that authors (or publishers) and readers (or viewers) are able to specify their preferences or needs.

We propose that end-user live editing functionality be provided by a user editing module implemented as an extension to the interactive TV set-top box middleware. The idea is to take advantage of the communication capabilities of digital interactive TV remote controls and of the computational capabilities of typical interactive TV set-top boxes to allow the viewer (that is, the end-user) to edit a live interactive program. In this paper we detail our design and current implementation in the context of the Brazilian interactive TV scenario, in which such live editing is possible given the underlying document-based mechanism used to support user-program interaction via NCL-documents.

In the remaining of this manuscript we present a scenario where we define the concept of *end-user changing the watching experience* (Section 2.1.2); discuss document issues related to supporting this concept in the context of interactive TV (Section 2.1.3); detail our proposal by discussing its specification via a use case diagram along with a presenation of our current prototype 2.1.4;

illustrate in Section 2.1.7 documents generated by the end-user edits as presented in Section 2.1.4; discuss related work in Section 2.1.9, and present our final remarks (Section 2.1.10).

2.1.2 End-user live editing of ITV documents: a scenario

The scenario presented in this section illustrates the concept of *end-user changing the watching experience* supported by the Watch and Comment paradigm. The scenario assumes the user watches a movie and may add marks as follows:

- Bookmark scene: the viewer selects one specific instant in the video as of interest to be reviewed later. A copy of the corresponding frame is extracted and added to a list of bookmarked instants that can later be used to revisit the video.
- Skip interval: the viewer selects two instants in the video which specify a time interval to be skipped when wacthed later – the two corresponding frames are extracted and added to the list of intervals that can be skipped.
- Loop interval: the viewer selects two instants in the video which specify a time interval to be reviewed in loop – the two corresponding frames are extracted and added to the list of intervals to be reviewed in loop.

In a real interactive TV scenario, such as the one of the Brazilian Digital TV, the movie would be broadcast together with its underlying *application document*, represented as an NCL declarative document – to enjoy the interactive experience the client should use a set-top box with recording capabilities.

Besides watching and listening to the movie, the viewer may also interact with the application through the facilities provided by the author and included in the *application document*, such as consulting reviews about the movie, answering quizzes, etc. These alternatives should even be included in the document by the provider of the content.

The discussion that follows assumes the viewer watches an old (copyright free) movie. However, similar services apply to other types of video — we have discussed elsewhere (Pimentel et al., 2008a) the scenario where the user is able to edit a live soccer game featuring edit options similar to those discussed in this paper, given that the game is broadcast live with commands issued through DSM-CC (ISO/IEC International Organization for Standardization, 1998) stream events and files encoded within the object carousel (Costa et al., 2006).

In the *watching an old movie scenario*, a resident software application — our current prototype is implemented in Java — could offer extra editing facilities, since the software application has capabilities to interact very closely with the middleware and the underlying infrastructure of the set-top box. Our proposal is to incorporate, in a traditional middleware, an *End-user Editing Software Application* which has such privileges of interaction with the infrastructure and may offer *APIs* that allow extra editing facilities, as exemplified next.

Given the availability of a middleware with a *End-user Editing Module* and an application for replaying the scenes selected by the viewer, the viewer may select moments of the video to watch again later, whenever and how many times he wishes.

As implemented in our current prototype, the simple user interface for this software application consists of the activation of a special key on the remote control that enters a *Watch and Comment mode* which allows the viewer to add marks to the video — the exact key is previously defined by the *End-user Editing Module*, depending on the options offered by the particular model of the remote control, and is known by the user.

Once the user has decided to add the very first mark in the movie (that is, the first time the user enters the *Watch-and-Comment mode* while watching that particular movie), a new document is created and associated with the original document so that it includes the annotations added by the user on the original media. From this moment on, all marks made by the user while watching the same video are stored in that annotation document. Moreover, the annotation document is used by the *End-user Editing Software Application* to allow the user to review the annotations while the movie is still being broadcast by the TV channel.

To each mark added by the user, appropriate information is added to the annotation document – which means that the original document is kept intact whereas a new document includes all marks as associated editing information. For each mark made by the user (bookmark, start & end loop or start &end skip), a copy of corresponding frame is extracted from the video so that it can be shown to the user (as thumbnails) as part of the corresponding reviewing menu.

For each editing operation possible – in the current implementation: bookmark, skip interval and loop interval – a reviewing menu is managed by the *End-user Editing Software Application*. When activated, the reviewing menu first presents the list of reviewing options available (e.g., Bookmark, Loop and Skip) and, when the user selects one of the options, it presents a list of thumbnail images corresponding to all marks of the selected type. If the viewer then chooses to review one the marked scenes from that list, the video is presented from the corresponding time offset. It must be observed that the original *application document* has changed and allows the viewer, at any moment, to easily interrupt the original *program* presentation (as delivered by the TV channel) in order to see one of the moments he has marked himself.

The example shows a situation in which the authors' original document is extended. End-user edits may also include changes and cuts in the original document. A viewer could decide, for example, to remove every attempt of the original document application when establishing some interaction with the user. It may even be the case that the viewer does not want to be bored with any kind of query, and does not want to see anything in the screen besides the video of his TV program. We have to put more attention in this question of changing the original document by the user in order to avoid situations which may lead to presentations that do not make sense. To avoid something like that happening, it must always be possible to undo some action or reset to the original document.

The edited program, stored in the set-top box, would be available for the user, and for distribution, even after the end of its broadcasting. To avoid proprietary rights problems, the distribution may be restricted to a NCL document that registers only the end-user edit commands. Someone who owns, even temporally, the right to watch the program, may combine its original underlying NCL application with the distributed editing and have a different watching experience. The NCL language has support for such merge of application documents (ABNT Associação Brasileira de Normas Técnicas, 2008).

2.1.3 Interactive multimedia document engineering issues

In order to allow the viewer to edit a program which is being broadcast and to see the consequences of such editing still during its presentation, it is necessary to provide some kind of live editing facility. The concept is analogous to the editing on the fly promoted by the service provider over a live broadcast program in which the new structured application document is created as the stream events and the files encoded withing the object carousel are transmitted (Costa et al., 2006). The middleware Ginga, adopted in the Brazilian Digital Interactive TV platform, offers such facility. The content provider sends editing commands to the set-top box. The commands are then interpreted and the necessary changes in the document, which is being considered for the presentation/interaction, are made.

One of the possibilities considered to allow live editing is to use the NCL live editing commands to support the edits promoted by the end-user as well. An underlying procedural application of a TV program, coming from the content provider together with the NCL application document, would establish a set of editing possibilities the user could promote over the original NCL code. The user edits would be translated to the appropriate live editing command which would be sent to the Ginga engine for completing the alterations. However, the current standards for the SBTVD impose formal restrictions[2] with respect to the use of live edit commands: they cannot be embedded in applications[3] because the provider is not allowed to construct an application that promotes self modifications. As a consequence, the user TV watching experience is limited to what the provider can offer to him. A richer interaction, with the user introducing particular editing elements, according to his experience and preferences, is restrained.

The alternative to overcome these restrictions is to work more closely to the set-top box middleware and its underlying infrastructure. Our proposal is to incorporate in the traditional middleware a *User editing Component* (UEC) which has privileges of interaction with the infrastructure and may offer APIs that allow extra editing facilities as aforementioned. Resident applications may then use these APIs to offer different types of editing to viewers.

The core of the middleware Ginga is composed by a component called *formatter* (Rodrigues & Soares, 2003), which deals with issues related to presentation, and a private base manager, which is in charge of managing the active NCL documents and of receiving editing commands coming from the provider. The proposed UEC has a direct interface with the private base manager for manipulating the presented documents. If by any chance there is no NCL being executed, the UEC is responsible to create a new NCL document to keep the viewer edits.

Saving information from the original underlying NCL document corresponding to the program being presented and the state of its presentation, controlled by the formatter module, are required actions for the proper synchronization among the viewer edits and the original application. The syntax of the end-user editing commands may be the same one adopted for the editing commands of NCL.

It is also necessary to analyse the mapping among the keys of the remote control already associated with the underlying NCL document of the broadcast program to avoid conflicts with the mapping required by an end-user editing

[2]Restrictions formalized in the specification document for the SBTVD (ABNT Associação Brasileira de Normas Técnicas, 2008).

[3]Using neither Java nor Lua programs, although there is language support in NCL for this.

resident application. A new private base is created to manipulate a copy of the document (the original underlying NCL application) which is, in fact, used to manage the program presentation and interaction with the viewer.

The original document is preserved and may be used by the viewer to resume the original presentation should the end-user editing produce any unexpected result. The synchronization between the two bases is required in order to keep consistency with live edits came from the provider, which must promote changes in both documents.

In this paper, we refer to the structured document that specifies the synchronization and interactivity of an (interactive) TV program as the *application document*; the word *program* is used to refer to a TV program as it is the usual jargon in TV.

2.1.4 End-user live editing of ITV documents: application design

The scenario discussed in the previous session leads to the specification of the use case diagram illustrated in Figure 2.1, which highlights eight use cases supporting transparent end-user live editing. This diagram has lead to the design of our current prototype, which has been incorporated into the Ginga middleware. In this section, we detail the use cases introduced in the figure along with the corresponding prototype we have implemented.

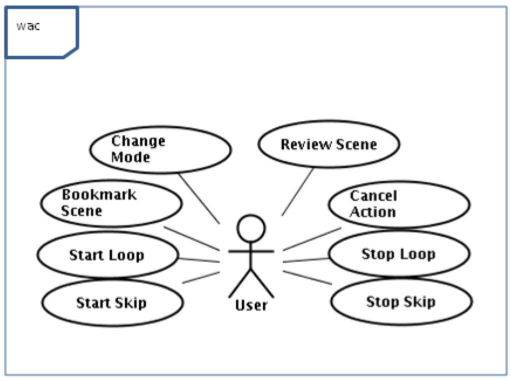


Figura 2.1: Use case Diagram for WaC TV: using the remote control, a viewer can create and review annotations

.

2.1.5 The viewer

The actor illustrated in Figure 2.1 represents the viewer who watches TV and interacts with the system we have modeled. By means of a typical input device associated with the TV, the remote control (as detailed next), the viewer will trigger the actions indicated by the eight use cases: Change Mode, Bookmark

Scene, Start Loop, Start Skip, Stop Loop, Stop Skip, Review Scene and Cancel Action.

2.1.6 The input

As far as the digital TV is concerned, the main viewer-TV interaction device is still the remote control. In other words, when interactive TV is available, its associated remote control contains buttons whose input is treated by software applications that provide interactive TV services.

In the Brazilian iTV scenario, at least four buttons for the user-interaction with applications must be available in all devices – they are of different shapes and colors (red circle, yellow triangle, blue square and green diamond) (SBTVD, 2006). Figure 2.2 (left) shows a minimal iTV remote control. The device in Figure 2.2 also feature extra buttons for access to applications or special functions. In the case of the remote pictured in the figure, for instance, four green buttons related to *guide*, *help*, *TV* and *info* are located above the navigation arrows (blue arrows ←↑→↓ circling the blue OK button).

Our current design allows users to watch-and-edit a TV program via a minimal remote control for iTV. However, it is relevant to observe that novel multimodal devices supporting gesture, touch and voice, for instance, are under active research (e.g. (Cattelan et al., 2008), (César et al., 2007)).

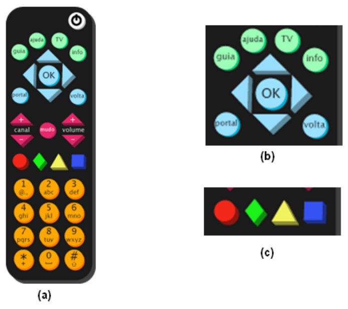


Figura 2.2: Remote control for the Brazilian Interactive TV. (a) minimal iTV remote control. (b)detail of navigation buttons on the bottom (blue) and buttons with special functions on the top (light green). (c) detail of four special colored buttons placed above the numeric keypad.

Use case: Change Mode

Our proposal is available on two modes: the *Watch mode* and the *Edit mode*. The *Watch mode* is the normal one in which the user watches TV, while the *Edit mode* allows the user to add editing marks on the video being watched. Considering our current implementation, from now on in this paper we will use the terms *Watch and Comment mode* and *Edit mode* interchangeably.

Our design assumes the use of one of the special programmable keys to activate the *Edit mode*. Our implementation has configured one of the special buttons on the remote control for this change of modes: the green *TV* button located above the navigation menu shown on the remote on left of Figure 2.2. By pressing the *Change mode* button while watching TV, the user enters the *Edit mode*.

Figure 2.3 shows (a) the feedback the viewer receives when the button associated with the *Change mode* is pressed: to indicate that the viewer has entered the *Edit* mode, the icon WATCH AND COMMENT MODE is shown on the top left portion of the TV, while the edit options available are indicated on the bottom: (BOOKMARK, SKIP and LOOP). Figure 2.3(b) presents in detail the *Watch and Comment mode* icon.

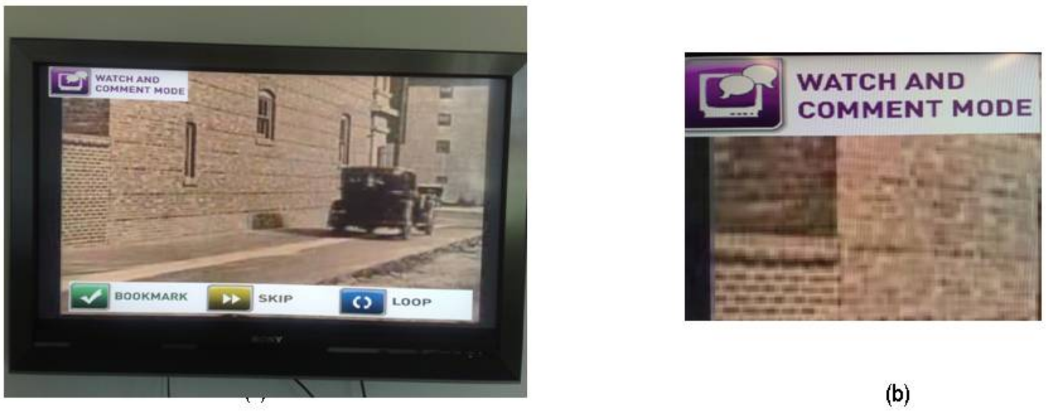


Figura 2.3: (a) Viewer has entered EDIT mode (indicated by icon *Watch and Comment mode* on top left); options available are indicated on the bottom (BOOKMARK, (start) SKIP and (start) LOOP). (b) The *Watch and Comment mode* icon is shown in detail

Once the user has changed modes, the usual semantics associated with remote control colored buttons (yellow, blue, green and red) is modified by our application. For instance: in the normal *Watch mode*, the green diamond button could be programmed by the TV producer to present a shopping form associated with the scene being presented; in the *Watch and Comment mode*, the same green diamond button can be programmed to bookmark the current scene (to be inserted in the user's bookmark list). This and the other options available in the *Watch and Comment mode* are detailed next.

Use case: Bookmark Scene

In the *Watch and Comment mode*, the *Bookmark Scene* option is chosen by clicking the green diamond button on the remote control. Figure 2.3 shows the BOOKMARK option as the first in the list of options presented on the bottom of the page, with a green icon shown next to the word BOOKMARK.

By activating this option, the user selects the current instant to be included in a “Bookmark list” associated with the current movie (or interactive program). Our current prototype extracts a copy of the corresponding video frame to be used as an icon in the bookmark list.

Use cases: Start Skip and Stop Skip

Our design specifies that the user may mark two instants to be the start and end of a video segment to be skipped when the video is played back in another opportunity.

Our implementation works as follows: when the user has entered the *Watch and Comment mode*, the *Start Skip* command is activated via the yellow triangle button on the remote control. In Figure 2.4(a), the SKIP option is listed as the second on the bottom of the page, with a yellow icon next to the word SKIP.

If the user selects one scene with the Skip option, our prototype then shows WAITING STOP SKIP as the second option on the bottom of the page as illustrated in Figure 2.4(b). At this point two alternatives are available: one is the selection of the corresponding stop skip frame; the other is to cancel the selection of the skip sequence altogether.

To select the frame associated with the end of the sequence, the user issues the *Stop Skip* command by pressing the same yellow triangle button which activated the Start Skip option. If the viewer decides to abandon the whole operation, he can press any button on the remote control other than the yellow triangle button.

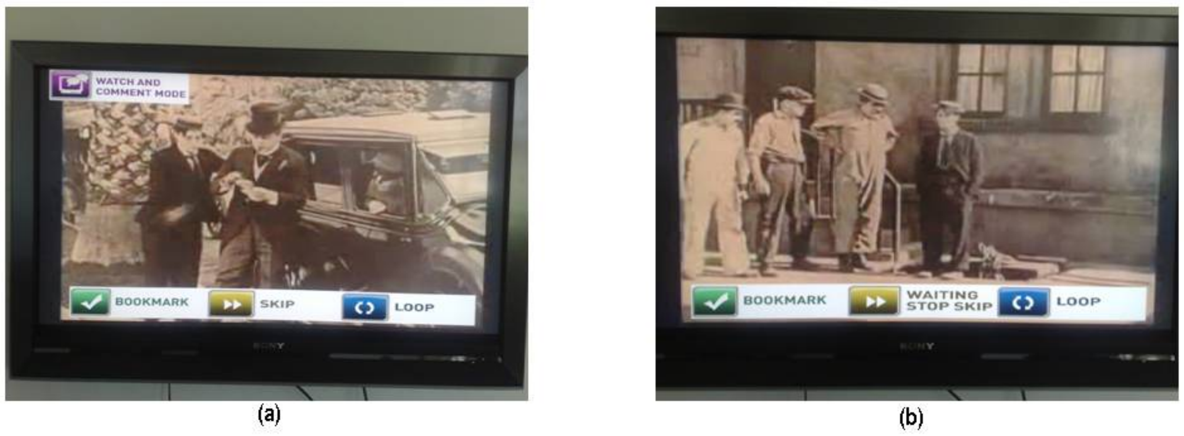


Figura 2.4: (a) Viewer has entered EDIT mode and presses the yellow triangle bottom associated with the start of a SKIP sequence. (b) The prototype replaces SKIP by WAITING STOP SKIP to indicate that the end of the skip sequence is expected

The video frame corresponding to both the start and end marks are copied by our prototype. If the user abandons the operation, the captured frame (the starting one) is discarded; if the operation is completed, both starting and ending frames are saved. In our current implementation, the start frame is used as an icon presented to the user when reviewing the "Skip list" corresponding to the program.

Use cases: Start Loop and Stop Loop

Among other options that we have designed to be offered to the viewer, the last edit option we have implemented in our prototype is the selection of a Loop sequence: the user may mark an instant to be the start of a segment to be played back in loop, and another instant to mark the end of the segment.

When the user is in the *Watch and Comment mode*, the *Start Loop* command is activated via the blue square button on the remote control. In Figure 2.4(a), the LOOP option is the last on list at the bottom of the page, with a blue icon next to the word LOOP.

If the user selects the start *Loop* option, the corresponding video frame is captured by our prototype, which then shows the WAITING STOP LOOP icon as the last option on the bottom of the screen. Similarly to the behavior of the Skip operation, the user may press the same (blue square) button on the remote control to select the corresponding stop frame, or cancel the selection by pressing any other button. Also, if the operation is concluded both start and end frames are saved.

Use case: Review Scene

The *Review Scene* use case specifies that the user may access a menu listing all reviewing options available. Our design concluded that one special button, programmable via a local application, activates the menu – similarly to the situation of changing mode, which also demands a special button.

Our current implementation works as follows: while in *Watch mode*, the user may activate the *Review menu* selecting the special green button named *Info*. This causes the video size to be reduced and a list of reviewing options to be presented on the right portion of the screen as illustrated in Figure 2.5: BOOKMARK, LOOP, SKIP, VOLTAR (which means BACK in Portuguese). The playback of the program is maintained in reduced video size. The bottom portion of the figure shows that the viewer may use the arrow keys (up and down) on the remote control to selected among the reviewing options, and use the OK button on the remote control to select one of the options. The figure also indicates that the user can abandon the selection of a bookmark by pressing the green diamond button of the remote control.

If one of the reviewing options is selected by the viewer, a list with video

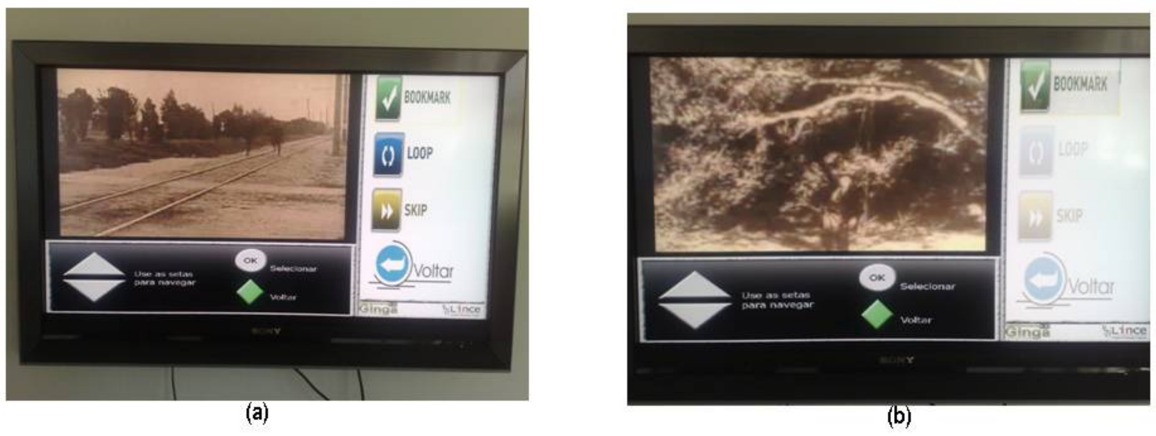


Figura 2.5: (a) The viewer has activated the *Review annotation button*: the video size is reduced to allow the review menu to be presented on right portion of the screen, and navigation options to be shown on the bottom. (b) The viewer has selected to review the BOOKMARKS option.

frames is presented as thumbnails on the right portion of the screen: the bookmark list presents the frame corresponding to the bookmarked scenes, the loop and skip lists present the frame corresponding to the starting instant of the marked interval. The playback of the program is maintained in reduced video size while the user can then navigate among the items using the arrow buttons of the remote control and select one of marked instants, or abandon the selection by pressing the diamond button.

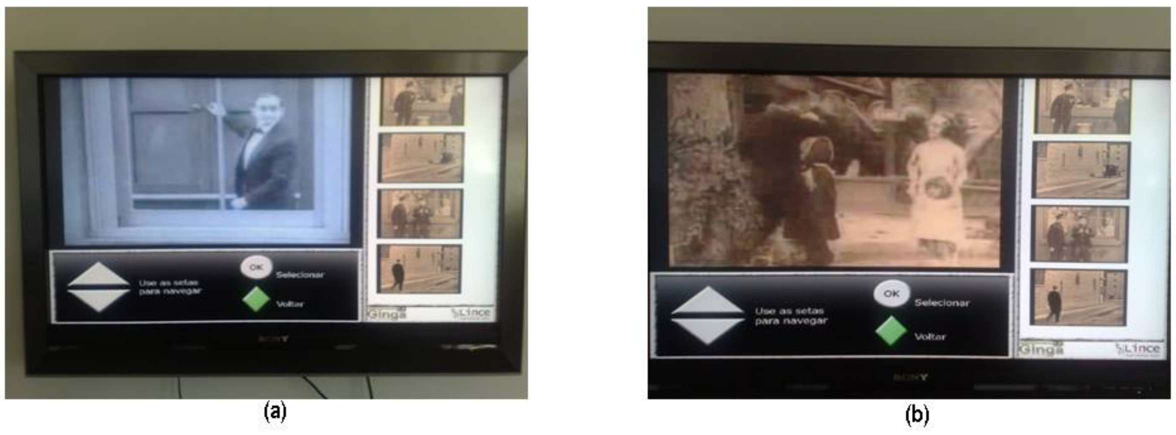


Figura 2.6: The movie continues to playback in reduced size while a list of frames is shown on the right portion of each screen. (a) and (b) illustrates the presentation of the bookmark list in two different moments in the video.

The *Review Scene* use case also shows the need for allowing the viewer to be made aware of previous editing in a program. When a program that has been edited is watched later, at the time of playback (during the *Watch mode*), our current implementation indicates when there is a segment that has been marked to be played back in loop (a) or to be skipped (b), as illustrated in Figure 2.7.

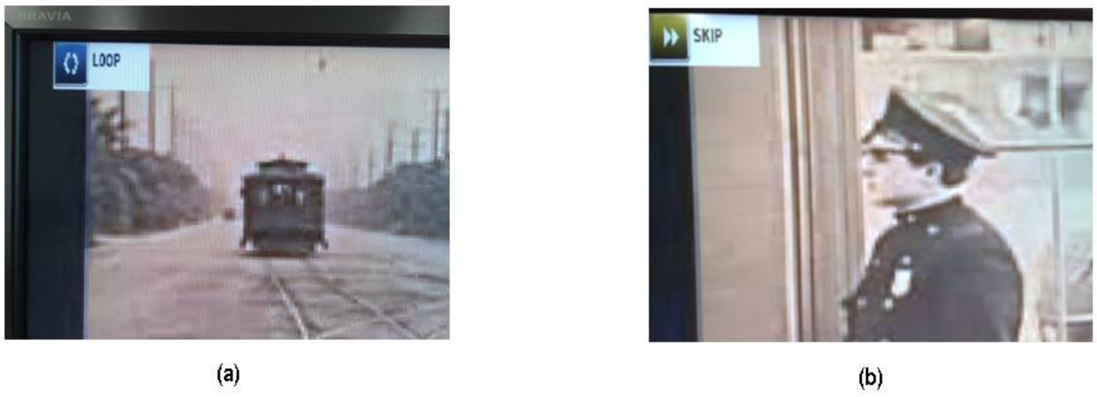


Figura 2.7: When a program that has been edited is watched later, at the time of playback (during the *Watch mode*), our application indicates when there is a segment that has been marked to be played back in loop (a) or to be skipped (b).

Use case: Cancel Action
The use case *Cancel Action* specifies that the user may cancel the edit operations when in Edit mode or when reviewing the edits. In our current implementation: to cancel the editing when in Edit mode, the user presses the same button that activated the Edit mode: the green *TV* button. If one of editing operations has already been selected, the user may abandon the operation by pressing the green diamond button, as detailed in the previous section.

2.1.7 Application structured document: encodings

In this section we illustrate two structured multimedia documents corresponding to an antique film encoded in NCL, as detailed in the previous section. The first document, illustrated in Figure 2.8, is the original NCL document. The second document registers the edits made by one end-user who made some comments while watching the original film – excerpts of the second document are illustrated in Figures 2.9 to 2.12.

The underlying scenario is one in which the user watches the antique film using an interactive TV remote control. In Figure 2.8, the `regionBase` element contains only one `region` element, since our original document was a video presented in the whole screen (100% of the `region` in line 08). Any interactive options in the original document would be specificied in the `connectorBase.ncl` document specified by the `importBase` element in line 19.

```
01 <?xml version="1.0" encoding="ISO-8859-1"?>
02
04 <ncl id="live"
05 xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile">
06  <head>
07   <regionBase>
08     <region id="rgVideo" left="0" top="0" width="100%" height="100%" zIndex="1"/>
09   </regionBase>
10
11   <descriptorBase>
12     <descriptor id="dVideo" region="rgVideo">
13       <descriptorParam name="soundLevel" value="1" />
14     </descriptor>
15   </descriptorBase>
16
17   <connectorBase>
18     <importBase alias="connectors" documentURI="connectorBase.ncl"/>
19   </connectorBase>
20  </head>
21
22  <body>
23    <port id="pStart" component="video" />
24      <media id="video" src="media/video.mpg" descriptor="dVideo">
25        <property name="bounds" />
26      </media>
27  </body>
28 </ncl>
```

Figura 2.8: NCL sample program (original)

The excerpts presented in the Figures 2.9 to 2.12 illustrate how the annotations made in the original program by the end-user causes a whole new document to be edited live, that is, while the original film is being broadcast.

For didactic reasons, the new document we present in this paper integrates the end-user edits with the original document. However, it is important to observe that the original document can preserved by making a reference to it (using the NCL `import` element, as detailed elsewhere (Pimentel et al., 2008a)): once the editing starts, the new document, which makes the reference to the original one, is the one played back. Besides preserving the original interaction, the new document presents new interaction points used for controlling the exhibition of the menu with the scenes selected by the user.

The documents illustrate the approach of creating a new document as a result of live end-user editing: this new document is owned by the user and, because the new document contains references to the original one, it can be reused respecting the original's authorship. However, anyone interested in the annotated contents must have permission to access the original contents (a likely scenario in education-related documents, for instance).

Figure 2.9 corresponds to the initial part of the document created as a result of the interaction detailed in the previous section. The `regionBase` element (lines 07 to 19) now specifies the several regions used by our applications (e.g, for the menu); the `descriptorBase` element (lines 21 to 42) associates identifiers to those regions. The reference to the original `connectorBase.ncl` document is kept intact (line 45).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ncl id="live"
 xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile">

 <!-- HEAD -->
 <head>
   <regionBase>
     <region id="rgVideo" left="0" top="0" width="100%" height="100%" zIndex="1"/>

   <!-- REGIONS FOR MENU COMPOSITION AND DISPLAY ANNOTATIONS-->
    <region id="rgMenu" left="70%" top="0" width="30%" height="100%" zIndex="1">
      <region id="rgSample1" left ="10%" top="2%" width="63%" height="20%"/>
  <region id="rgSample2" left ="10%" top="24%" width="63%" height="20%"/>
  <region id="rgSample3" left ="10%" top="46%" width="63%" height="20%"/>
  <region id="rgSample4" left ="10%" top="68%" width="63%" height="20%"/>
 </region>
 <region id="rgBottom" left="0" top="70%" width="70%" height="30%" />
 <region id="rgIcon" left="2%" top="2%" width="8%" height="8%" zIndex="10"/>
   </regionBase>

   <descriptorBase>
     <descriptor id="dVideo" region="rgVideo">
        <descriptorParam name="soundLevel" value="1" />
     </descriptor>

   <!-- REGIONS FOR MENU COMPOSITION AND DISPLAY ANNOTATIONS-->
   <descriptor id="dIcon" region="rgIcon"/>
     <descriptor id="dMenu" region="rgMenu"/>
   <descriptor id="dBottom" region="rgBottom" />
   <descriptor id="dSample1" region="rgSample1"
                 focusIndex="1" moveDown="2"
                 focusBorderWidth="-2" focusBorderColor="yellow" />
   <descriptor id="dSample2" region="rgSample2"
                 focusIndex="2" moveDown="3" moveUp="1"
                 focusBorderWidth="-2" focusBorderColor="yellow" />
   <descriptor id="dSample3" region="rgSample3"
                 focusIndex="3" moveDown="4" moveUp="2"
                 focusBorderWidth="-2" focusBorderColor="yellow" />
   <descriptor id="dSample4" region="rgSample4"
                 focusIndex="4" moveUp="3"
                 focusBorderWidth="-2" focusBorderColor="yellow" />
   </descriptorBase>

   <connectorBase>
   <importBase alias="connectors" documentURI="connectorBase.ncl"/>
   </connectorBase>
 </head>

```

Figura 2.9: NCL declaration and `head` element containing regions where menus are shown

```

 <!-- BODY -->
 <body>
   <port id="pStart" component="video" />

   <!-- ADDED ANCHORS TO THE ANNOTATED FRAMES-->
   <media id="video" src="media/video.mpg" descriptor="dVideo">
     <property name="bounds" />
   <area id="areaScene1" begin="36s" />
   <area id="areaScene2" begin="75s" />
   <area id="areaLoop1Begin" begin="125s" />
   <area id="areaLoop1End" begin="143s" />
   <area id="areaSkip1Begin" begin="89s" />
   <area id="areaSkip1End" begin="103s" />
   </media>

   <media id="recordedVideo" src="media/video.mpg" descriptor="dVideo">
     <property name="bounds" />
   </media>

```

Figura 2.10: NCL code: `body` element and list of 6 anchors added by the user: 2 bookmarked regions, 1 start/end loop and 1 start/end skip

Figure 2.10 shows the start of the `body` section of the new document. The marks created by the user are associated with the `media` element (lines 07 to 15) as new `area` elements, which specify the start time of the mark (`begin`). The reference to the original `media/video.mpg` media now refers to the recorded one (id="recordedVideo") (line 17).

```

   <!--   SKIP ANNOTATION LINKS (show icon, hide icon, skip the video segment) -->
   <link id="linkSkip1Begin" xconnector="onBegin1StartN">
   <bind component="video" interface="areaSkip1Begin" role="onBegin"/>
   <bind component="skipIcone" role="start"/>
   </link>

   <link id="linkSkip1End" xconnector="onBegin1StopN">
   <bind component="video" interface="areaSkip1End" role="onBegin"/>
   <bind component="skipIcone" role="stop"/>
   </link>

   <link id="Skip1" xconnector="onKeySelection1StopNAbortNStartN">
   <bind component="skipIcon" role="onSelection">
   <bindParam name="keyCode" value="RED" />
   </bind>
   <bind component="skipIcon" role="stop"/>
   <bind component="video" role="abort"/>
   <bind component="video" interface="areaSkip1End" role="start"/>
   </link>

```

Figura 2.11: NCL sample annotation - declaration and `head` element

Many other elements are automatically created in the new document as a result of the viewer-interaction. Figure 2.11 shows the links created to relate the instant marked as Skip for the first time with the starting and ending times of the region to be skipped.

```

   <!--   GO TO A SELECTED SCENE -->
   <link id="linkSelectCena1" xconnector="onSelection1StopNAbortNPauseNStartN">
   <bind component="Scene1" role="onSelection"/>
   <bind component="Scene1" role="stop"/>
   <bind component="Scene2" role="stop"/>
   <bind component="Scene3" role="stop"/>
   <bind component="Scene4" role="stop"/>
   <bind component="bottom" role="stop"/>
   <bind component="menu" role="stop"/>
   <bind component="video" role="abort"/>
   <bind component="mainStream" role="pause"/>
   <bind component="video" interface="areaScene1" role="start"/>
   </link>

 </body>
</ncl>
```

Figura 2.12: NCL sample annotation

The last excerpt, presented in Figure 2.12, is also the last portion of the document generated and illustrates how the list of scenes shown to the user is encoded as a multidimensional link (one with several bindings) in the NCL document.

2.1.8 Architecture

In this section we detail the architecture underlying our current implementation, which extends the original Ginga-NCL architecture supporting the SBDTV.

The Ginga Architecture illustrated in Figure 2.13 contains five main layers: Resident Applications, Ginga-Implementation Execution Environment, Ginga-NCL Presentation Environment, Ginga Common Core and Protocol Stack.

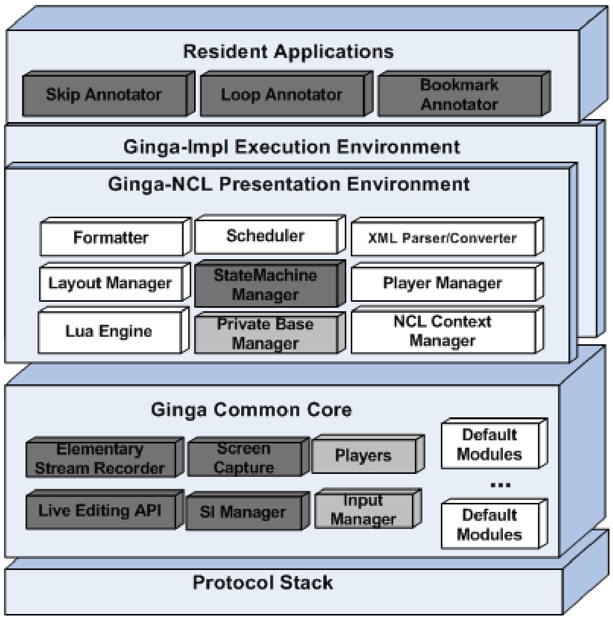


Figura 2.13: Architecture used in WaCTV: Original Ginga modules are indicated in white boxes, modified Ginga modules are indicated in light gray boxes, new modules are indicated in dark gray

.

To implement our proposal we have implemented eight new blocks which we have integrated into three layers of the original Ginga Architecture. The eight new modules, indicated in dark gray in Figure 2.13, are the following: three new modules in the Resident Applications layer (Skip Annotator, Loop Annotator and Bookmark Annotator), one new module in the Ginga-NCL Presentation Environment layer (StateMachine Manager), and four new modules in the Ginga Common Core layer (Elementary Stream Recorder, Screen Capture, Live Editing API and SI Manager).

We have also made modifications to three modules from the original architecture, as detailed later in this section: the *Private Base Manager* of the Ginga-NCL Presentation Environment layer, and in the modules *Players* and *Input Manager* of the Ginga Common Core Layer.

Specialized modules in the Resident Applications layer

The three new modules in the Resident Applications layer, indicated in dark gray in Figure 2.13, are the *Skip Annotator*, *Loop Annotator* and *Bookmark Annotator* modules. They are responsible for implementing the semantic of the annotations as discussed in the previous sections. The implementation of the annotation features demanded modifications in two other layers, as detailed in the next section.

The algorithm for each application is presented in Figure 2.14: the algorithms make calls to two other modules in other layers (ScreenCapture and InputManager), and manage their own lists of annotations.

```
// Specialized Modules in Resident Applications layer
// called when in WaC Mode

// Wac mode + Bookmark key
//
Bookmark(t){
  ScreenCapture(t,frame)
  AddBookmarkList(t,frame)
}

// Wac mode + StartSkip key
//
StartSkip(t){
  ScreenCapture(t,frame)
  key = InputManager()
  if key == end_skip
    then AddSkipList(t,frame)
  else release(t,frame)
}

// Wac mode + StartLoop key
//
StartLoop(t){
  ScreenCapture(t,frame)
  key = InputManager()
  if key == end_loop
    then AddLoopList(t,frame)
  else release(t,frame)
}
```

Figura 2.14: Algorithms for Bookmark, Skip and Loop Annotator modules

Support to multiple documents in the Ginga-NCL Presentation Environment layer
The Ginga middleware processes NCL interactive multimedia documents that specify the composition of medias, their spacial and temporal behavior, as well as any user interaction. The playback of the document is controlled by the modules of the Presentation Environment layer shown in white in Figure 2.13 (e.g. the Scheduler, the Formatter, the Private Base Manager and the NCL Context Manager modules).

The original Ginga middleware has been designed to allow one document to be played at a time: the document being broadcast. In order to allow live end-user editing, the document being broadcast is annotated by the user *on the fly*, which means that a new document must be created that modifies the original one to accommodate the user annotations (e.g., bookmarks) as anchors to proper time offsets in the original document. This allows the new annotations to affect the current playback if the user decides to review the annotation (e.g. playback from the time offset indicated by one of the bookmarks). This also means that, once the user makes some WaC annotation, a new document is created and must be the one played back once the user decides to review one of the marks – that is, the playback of the original document is replaced by the edited one.

New module: StateMachine Manager
To control the playback of the edited document, we added a new *StateMachine Manager* module in the Ginga-NCL Presentation Environment layer. This module manages the playback of the new document – which contains the new anchors created according to the user's marks and references to the original document.

In this opportunity, it is relevant to observe that the original document has not been modified (so as not to violate its authorship) and the newly created document cannot be played back if the original one is not captured as well – what implies that users must have the right to capture the document (and contained media) in their own set-top-box.

Extended module: Private Base module
Another modification we made in this layer is an extension to the *Private Base Manager* module. This was necessary because the original module managed (received and stored) only one document at a time – the original broadcast document.

We designed and implemented an extension so that the module now also manages the new document generated by the WaC edit commands entered by the viewer.

Capture & Access in the Ginga Common Core layer
Supporting the WaC mode demanded new modules to be included in the original Ginga Common Core layer: modules *Live Ediding API*, *Elementary Stream Recorder*, *Screen Capture* and *SI Manager*. We also made extensions to the original *Players* and *Input Manager* modules.

New module: Screen Capture
The ubiquitous approach adopted in our design demanded the transparent capture of the user interaction with the interactive video. Considering what the viewer watches at the time of the annotation, we chose to capture the corresponding video frame so that it can be presented later when the user decides to access, or review, the marks. These video frames can be used in several reviewing options — the important point is that the frame was the information available to the user at the time the annotation was made and, as such, should help the user in choosing one when several reviewing options are available (e.g, choosing one among the several bookmarks made in the program).

It is important to observe that the document context of the captured video frame must be captured as well. In other words, the overall WaC application must capture the exact portion of the structured document in which the interactive video is presented – since the review will demand the video frame not only to be presented, but also to be played back in its original interactive video context, as specified by the NCL document.

New module: Elementary Stream Recorder
The WaC approach requires that the viewer has an infrastructure that allows the recording of selected interactive documents. The original Ginga middleware includes services that controls the playback of all elementary streams of an interactive multimedia program. We designed a new module that allows the recording of all elementary streams in such a way that they can be later referenced and played back, along with other information captured by the new module Information System Manager, as detailed next.

New module: Information System Manager (SI Manager)
The Ginga middleware follows the official standard specification with respect to the identification of which multiplexed stream must be played back for a given interactive TV application. However, additional information relative to temporal synchronization is also demanded to allow the playback a posteriori as implied by the live editing approach.

We designed an additional module in the Ginga Common Core layer: the *Information System Manager (SI Manager)*. This module registers information relative to the synchronization of all elementary streams of the original program, and controls the playback of that information in concordance with the additional user edits.

New module: Live Editing API
As far as the support to live editing in the Ginga infrastructure is concerned, the WaC edits demand the original document to be modified on the fly: the new edited document must replace the original one. A new module was designed to allow the communication with the (extended) Private Base Manager, using DSM-CC, in a way that is analogous to that proposed by Costa et al. to support the live editing controlled by the TV station (Costa et al., 2006)).

We provide an API that is responsible for both the generation of the new NCL code and the communication with the Private Base. As a result, applications demanding live-editing behavior can make calls to this API to request the generation of the NCL documents and the communication the Private Base.

Extended module: Players
A NCL document is composed of several media, each one associated with information that describes its state (occurring, stopped, paused, anchors values,

timeline). The *Players* module in the Ginga-CC contains several components, each responsible for managing the playback of one particular type of media (e.g. video, text and HTML).

In the context of linear video, annotations are made by anchoring information to points in the media. In the context of interactive video, anchors must be associated with the structured NCL document: the condition of activating a link demands it be defined by the composition of the state machine controlling each media.

Extended module: Input Manager

The WaC approach implies that a viewer is able to add marks to a video being played back, and use the marks to index moments of the video that are to be reviewed or even skipped. This demands that some type of user input be recognized by the annotation infrastructure.

Considering that the interactive programs broadcast may already have functions for the special colored buttons which are expected to be present in all remote controls for the SBTVD (illustrated in Figure 2.2), we designed the viewer-TV interaction to provide input using two distinct modes: Watch mode and WaC mode. This means that one particular key of the remote control is programmed to change modes from the default watching mode to the new WaC mode when the user decides to edit the video by making an annotation. One particular button (usually there is one available for "system" functions) is programmed to toggle between modes so that, when the WaC mode is activated, the colored buttons can then be associated with the WaC functions.

To support the two distinct modes, the Input Manager module was extended so that user input can be directed not only to the middleware declarative and procedural environments, but also to other modules and native applications.

2.1.9 Related Work

Ursu et al. (Ursu et al., 2008) discuss how the concept of "Interactive narrativity, that is, the ability to interact with (and influence) stories whilst they are being told, represents one clear development path for interactive television". Their work discusses how the user-interaction with the program can be processed so as to influence the course of the presentation. The work we present in this paper, on the other hand, focuses on how the user-interaction with the program can change the user's own TV watching experience. However, we are investigating alternative ways to capture and process the viewer-interaction experience from a large population (Baladrón et al., 2008).

The support to facilitate the generation of personal marks by users has LONG been investigated by many researchers, from early hypertext systems (e.g. (Marshall, 1998)) to recent Web-based multimedia tools processing annotation data (e.g. (Elliott & Özsoyoglu, 2008)). In the Web environment, the opportunity to provide support to bookmark generation by capturing the user-interaction and navigation history has been recently demonstrated by Hupp et al. (Hupp & Miller, 2007). Similarly to the proposal by Hupp et al, the work detailed in this paper leverages the fact that the capture of the user-interaction may be exploited in the automatic generation of annotations – but their work is capable of exploiting further interaction data, such as a long term history provided by the Web environment.

The problems users face while interacting with the remote-control in home environments have led researchers to propose some rather intelligent devices (Omojokun et al., 2006) as well as simpler devices to be used by older (Rice & Alm, 2008) or less-technically-inclined people (Darnell, 2008). One of the problems identified is the confusion caused by the use of modes. Given that our current implementation depends on modes, we need to provide alternative ways to implement our design.

The need to investigate alternative ways to provide user-centered control *within* multimedia presentations has been discussed in depth by Bulterman (Bulterman, 2007). Starting from an analysis of the limited interaction allowed in most systems, Bulterman outlines various alternatives for providing user-centered control of multimedia when in the context of a structured multimedia presentation, which include (a) allowing the user to control the interaction upon alternatives provided in advance by the authors (e.g. selecting the audio channel to choose the languages for audio and/or captions), and (b) providing to the user a layered model for augmentation and personalization. His discussions exploit a document-oriented approach based on experiences inspired in the interactive TV scenarios, as in other reports from the same group (e.g. (César et al., 2006b) (César et al., 2007)). The opportunity to exploit a secondary screen to allow the user to control, enrich, share, and transfer contents has also been discussed by the group (César et al., 2008). As far as interactive TV is concerned, the work we present in this paper is complementary to Bulterman's views: we provide opportunity for the user to control the presentation in moments in which the author had not provisioned structured control points. This opportunity is important in the context of live broadcast programs in which part of the media, in spite of being a component of a higher level structured multimedia presentation, has contents with portions of interest at a lower level of granularity than that embedded in the structure at broadcast time, from the end-user perspective.

With respect to live editing of interactive documents for digital TV, our work is mostly related to the work reported by Costa et al. (Costa et al., 2006). The authors report their solution to the problem of allowing edits on a document to be possible at the same time of its presentation. This is necessary in situations where information such as temporal and spacial relationships among program media objects are missing when the document relative to the program is produced. Such a demand is real in situations where portions of a program is not available before the presentation time: their motivating example is *An unexpected game pause for a player medical assistance may be used to add an advertisement temporally related with the medical product in use at that exact moment* (Costa et al., 2006). Their solution involves adding edit commands and chunks of NCL code which are sent from the content provider (broadcasting company) to the set-top box by means of a DSM-CC (ISO/IEC International Organization for Standardization, 1998) multiplexer: NCL files are encoded withing the object carousel and editing commands transmitted as DSM-CC stream events. At the client side, there is a module responsible for receiving files from the carousel and storing them at the local file system. Another component, which works as a listener of the *edit command events*, interprets the command semantics and take the necessary actions demanded by the author to change the original document.

The focus of the work by Costa et al. (Costa et al., 2006) is the support to the live editing demanded by the fact that not all information has been defined at authoring time, and their strategy applies to live programs at the broadcast (server) side. Our work, on the other hand, target the editing at the client side but works both for live and recorded interactive programs. The opportunity for use in recorded programs is relevant because, when a user is allowed to capture an interactive program (that is, the license permits the recording for later review), the user does not have access to the recording underlying structured document.

Our work is also related the Bulterman's work with respect to the use of an structured document language to code annotations (Bulterman, 2003). In both cases, there is a need to consider compositions of objects within their original context. One difference is that, in our case, we assume that the document can be annotated (indicated as such at the time of the broadcast) with other previous preparation. Another difference is that we provide annotation templates (such as menus for the user to allocate the best moments he elects while watching the live program). With respect to copyright, both works are similar considering that a separate document is generated with the annotations for interchange purposes; the difference is that, at the time of playback, we present the edited document directly in the user's set-top box.

One difference to other existing research is the fact that we support live editing by the viewer: there is no need for a new document to be presented because we make changes directly to the playback of the program within the client machine (made possible by access given to the set-top box middleware). Because we support this live editing, we are also able to synchronize the document being presented with other online services: for instance to synchronize with the user's calendar and, say, suggest (an edit) a pause because of an event indicated in the calendar. This is related to cross-media linking approaches in which links are supported among digital artifacts and services (e.g. (Weibel et al., 2007)).

2.1.10 Final Remarks

In this paper we evolve our previous work by extending and integrating the efforts to allow the end-user to make live editing of interactive video programs at the time of the broadcast.

Moreover, we also introduce and explore the concept of *end-user changing the watching experience* to allow end-user edits to affect not only the parts of the program already watched but also contents not yet presented. We support changes *not* planned by the author and already coded in the original (broadcast) program. We propose that this functionality be provided by a user editing module implemented as an extension of the set-top box middleware and we illustrate its use in a typical scenario involving a soccer game.

We believe that the more interactivity the is between the user and the TV program, the more interesting the user experience will be and the more involved the user will get. In fact, our approach allows scenarios in which the user may interact in many novel ways with the interactive video contents. Our proposal, illustrated in the context of NCL and the Brazilian Digital TV environment, can be generalized to other environments and middlewares built upon document-based specifications for the interactive multimedia contents.

In the short term, we plan to carry out evaluations with real users using the prototype detailed in this paper. Next, we plan to integrate the live editing mechanism to other services available to the user via the set-top-box.

Acknowledgments

We thank Rudinei Goularte and Felipe Silva Santos, who worked in previous versions of our infrastructure. We thank the following organizations for their support: FINEP, FAPESP, CAPES and CNPq.

2.2 Discrimination of media moments and media intervals: from watch-and-comment to sticker-based annotation

The *Watch-and-Comment* authoring paradigm has been proposed as the seamless capture of comments made by users when appreciating a video and the association of the comments with the original media so as to generate interactive videos automatically — the resulting interactive video corresponding to the original video annotated with the captured comments. In previous work we have defined Watch-and-Comment operators for the discrimination of media moments and of media intervals within continuous media, and we have shown how these operators can be used by applications processing annotations generated collectively by several users, over distinct instances of media and at different times. Considering the opportunity of editing an interactive video on the fly, as reported in the literature in the context of interactive television, in this paper we show how our operators may be used, at the time of broadcast, to annotate interactive videos with previously created media — which we call media *stickers*. We discuss how our operators allow the annotation of interval-based stickers in a proof-of-concept prototype handling stickers on a Digital Television platform: our examples involve the use of the stickers by the final user and the broadcast service.

2.2.1 Introduction

In comparison with previous technologies, interactive digital TV provides quality audio and video and has the potential to offer many types of user-video interactions and viewer-service interactions. Interactivity may occur, for instance, by means of access to external services, made possible by the computing power and communication capabilities that can be incorporated into TV sets. These features open new opportunities for research, since new types of services, applications and interaction paradigms can be exploited. Examples include the investigation of alternatives for supporting social interaction via TV-related asynchronous interaction among viewers by means of implicit recommendations of TV shows (Nathan et al., 2008), as well as the use of synchronous explicit recommendation (Coppens et al., 2004). Another example is the study of communication choices and practices in TV-based text and voice chat (Huang et al., 2009). Yet another example is the investigation of how passive users can interact more directly with the content they watch in TV-environments (César et al., 2009). As a final example, the Watch-and-Comment authoring paradigm proposes the capture of the comments made by viewers while watching TV programs, so that the comments are treated as multimodal annotations used to automatically create user-annotated interactive videos (Cattelan et al., 2008) (Pimentel et al., 2008b).

We observed that it is possible to extend the application of the Watch-and-Comment paradigm (Cattelan et al., 2008) (Pimentel et al., 2008b) to support the transparent editing of continuous media, in the context of interactive digital TV, by multiple distributed users who use their own personal devices (Pimentel et al., 2009). The editing can occur both for broadcast content, by capturing and recording the content as well as the annotations, and for content stored locally. The editing, in this case, is related to the discrimination of segments of media, based on temporal data and association with the user who performed the action.

Inspired by earlier results on the Watch-and-Comment (WaC) authoring paradigm, in previous work we have exploited its approach in the definition of operators for the collective discrimination of moments and of media intervals within continuous media (Teixeira et al., 2010a) — the aim is to make it possible, for instance, the merging of annotations by individual users. It is worth observing that the meaning of the word discrimination, in the context of our work, is *the recognition of the difference between one thing and another*. In other words, users discriminate one scene when they recognize some particular value for the scene with respect to another scene, the value being associated with the users' own context and watching-TV experience. The objective of work reported in that previous work (Teixeira et al., 2010a) was to provide mechanisms — operators for the discrimination of instant and time intervals within continuous media — that allow multiple users to edit multimedia documents using their own personal devices, in a transparent way, using a protocol for the automatic discovery of services (Teixeira et al., 2010a).

There are several opportunities for users to personalize their watching experience when live editing operations are available — examples have been reported in the literature in the context of interactive television both on the server side (Costa et al., 2006) and on the client side (Coppens et al., 2004) (Nathan et al., 2008) (Pimentel et al., 2008a) (Pimentel et al., 2010). The need for offering personalization alternatives has led to efforts that include offering powerful search mechanisms (Patel et al., 2008) and analyzing metadata to extract the emotional atmosphere in the media (Petersen & Butkus, 2008).

In this paper we extend our previous work relative to the collective discrimination of moments and of media intervals (Teixeira et al., 2010a) to demonstrate an alternative way of how the operators for the discrimination of instant and time intervals may be used by the service provider or by the users themselves to enhance the live experience. Investigating personalization alternatives, we show how our discriminating operators may be deployed by users who watch an interactive video to enrich the watching of the video experience with previously created media — which we call media stickers. In other words, the idea is to offer users the opportunity to use *stickers* to enrich, and personalize, the user experience of watching the original media.

We use the term *sticker* as a reference to the fact that some existing piece of content may be attached (or glued) to the original media. In one sense, this is similar to a resource, commonly offered by webcams and digital cameras, by means of which a user may associate some predefined effects to a picture or video captured by the device. Is also worth observing that stickers may be *instant-based*, such as logom or a picture, or *interval-based*, such as an animation or a sound effect. Interval-based stickers have an intrinsic duration. Moreover, stickers can be combined: a logo, an animation and a sound effect can be used at the same time, for instance.

The adding of extra contents to video is a common practice among broadcast companies: animations and audio effects are often added both to live programs such as sports matches, and to recorded programs such as music video clips. However, the offering of similar operations to viewers as a way of personalizing their experience, to the best of our knowledge, has not been reported. The association of additional contents to TV programs has been reported in the literature, as in the cases of the AmigoTV (Coppens et al., 2004) and CollaboraTV (Nathan et al., 2008) systems, but in theses cases the research focus has been on television-based communication. In this paper we detail how our operators allow the use of both moment and interval-based stickers in a proof-of-concept prototype handling stickers on a Digital Television platform.

To experiment with our approach, we extended the Ginga middleware (the Brazilian Interactive TV Middleware (Soares et al., 2007) (Soares & Souza Filho, 2007)) to support the discrimination of segments of continuous media, possibly via multiple devices, and support the UPnP protocol. To demonstrate the utility of the extended version, we built applications which use the discrimination operators. This work leverages research results relative to supporting collaborative annotations of points of interest presented in our previous work (Pimentel et al., 2009) – novel contributions include the support to media intervals, the formalization of the discrimination operators, and the implementation of new software components which could be experimented by means of original interactive applications. In particular with respect to our previous reporting on the collective discrimination of moments and of media intervals (Teixeira et al., 2010a), the main novelty in this paper is the presentation of how the discrimination operators allow extra contents, in the form of stickers, to be added to the original media.

In this paper, Section 2.2.2 presents scenarios exploiting the discrimination of intervals of continuous media; Section 2.2.3 details the operators we have defined to the discrimination of moments and of media intervals within continuous media; Section 2.2.4 discusses the prototype implementation we designed to support the development of applications such as those presented in Section 2.2.2; Section 2.2.5 discusses illustrative applications we have built; Section 2.2.6 reviews related works; and Section 2.2.7 presents our final remarks.

2.2.2 Motivating Scenarios

Using examples in which the TV is a main interaction point, this section presents application scenarios of our concept of ubiquitous collaborative continuous media instant and time interval discrimination. We include scenarios in which stickers may associated with video content on the client side, by viewers, or on the server side. A more comprehensive list of scenarios is presented elsewhere (Pimentel et al., 2009).

Discrimination for later review

In the scenarios which use discrimination for later review, we assume that the video is previously recorded or, in the case of broadcast video from interactive TV, we assume that the video is recorded by at least one of the viewers during its presentation. We also assume that viewers have their own mobile devices with wireless communication capability. The scenarios illustrate situations in which a client application, running on the viewers' mobile devices, makes use of wireless communication to exchange data with applications running on a TV set-top box.

Educational Scenario

An instructor of karate can ask his students to watch a competition presented in the TV during the Olympic Games. While watching the competition, students should classify, using the numerical keyboard of their mobile phones, the name of a few karate strikes used by the contenders. Later, in the training area, students can send their annotations to an application running on the instructor's interactive TV set in which he recorded the competition while it was broadcast. The instructor, along with the students, can review the annotations as a group or individually using an application that synchronizes all annotations with respect to the start of the original broadcast. The collective vision can supply rich feedback to the group's skills.

In another educational scenario, students of medicine may be asked by one Professor to watch the video of a surgery procedure and discriminate (using their smartphones) portions in which a particular instrument was used, for instance. When gathered in the classroom, their selections can be sent from their mobile devices to a TV set which contains not only the original video but also an application designed to associate each selection with its corresponding moment in the video. A similar model may be applied to many other educational contexts.

Watching Sports Scenario

In a football match, each viewer could make marks on the scenes that they like best while they watch a match. In a future meeting, for instance when they gather at a friend's house, an interactive multimedia document would be generated automatically: this would be achieved by an application running on the TV set-top-box which would collect the marks made by all users from their mobile devices, and show the most marked moments of the match in accordance with the opinions collected. In the same example, the markings could also be used for the election of the most marked scene: the chosen scene would be the one which received the higher number of marks.

Discrimination on the fly with stickers

We present scenarios in which stickers may be used in everyday situations. In the scenarios, a group of friends gather to watch together a football match. All friends are fans of the same team— the Bahia Football Club — and the match they are going to watch is one in which Bahia plays against its most important rival — the Vitoria Football Club. In the first scenario, the fans are those who decide when a sticker should be used, and which sticker is to be used. In the second scenario, the same match is broadcast but it is the broadcast service who decides both which stickers are added and when they are added: viewers have only to select, if they want, which team they support.

Watching Sports Scenario: Stickers added by the Users

Dr. Prazeres, a lifetime fan of the Bahia Football Club, invites his friends to watch the championship final match against enemy Vitória Football Club in his home — he has just bought a brand new 52-inch TV because, from now on, all football matches are going to be broadcast using high definition digital TV quality with customizable interaction services.

To prepare for the event, Dr. Prazeres buys food and drinks; he also buys cups and napkins with the Bahia logo and hangs his huge Bahia F.C. flag on the wall above the TV. At one moment, his cell phone plays the Bahia F.C. hymn: this special ringtone announces that one of his friends is calling — his friend asks to borrow Dr. Prazeres' old Bahia t-shirt to watch the game.

As Dr. Prazeres' many friends arrive, he concludes the preparation of this TV set-top-box. He transfers to the set-top-box the latest pictures, animations and chants the club has made available in the website: these are the *stickers* designed to be added to the screen during this particular match. The stickers Dr. Prazeres has selected include (a) the chant from last year's championship, which he plans to play before the match; (b) the latest recording of the official hymn, which he plans to play when his team enters the field; (c) special animations and chants to be played when goals are scored during this match; (d) pictures of this game's players along with data relative to position played, number of goals scored in the championship, etc.

When Dr. Prazeres finishes explaining to his friends how he intends to use the remote control to add the stickers to the video during the match, the match is about to start. Since several friends have smartphones, Dr. Prazeres explains that they can add stickers themselves by connecting to the TV set-top-box using their mobile devices.

The next couple of hours go as happily as planned. Before the match, while the TV presents information about the Vitoria team, Dr. Prazeres uses the remote control to present the stickers he had selected. When his team finally enters the field, Dr. Prazeres proudly adds the sticker with the official hymn. During the match, each of the goals scored by Bahia is celebrated not only with stickers with an animation involving the number of the goal in the match but also with stickers with details of the player who scored the goal. Moreover, when Vitoria scored their only goal, many friends add stickers with animations corresponding to their sadness.

During the half-time interval the friends agree that at some moments the stickers were intrusive: Dr. Prazeres brings a 26-inch TV set from his bedroom and the friends configure their devices to present their stickers on the smaller

TV.

At the end of the match, the happy Bahia supporters learn that they can use the information stored in their devices — corresponding to records of the moments and intervals in which they added stickers to the image presented on the TV — to annotate the video when they watch the same match again.

Watching Sports Scenario: Stickers added by the Broadcast Operator

Because the broadcast provider has learned, from their market research reports, that many of their customers are not as active while watching TV as Dr. Prazeres and his friends, they have designed an application that allows users to take advantage of the interactive TV facilities in an alternative way; its use is as follows:

- During a whole week, at the times when the match between Bahia and Vitoria is advertised, the broadcast provider allows viewers to select one of the two teams as the one they support. This option is also offered when the match is about to start.
- When visiting his grandparents, Dr. Prazeres selects Bahia F.C as the team his dear grandparent supports.
- Just before the match the broadcast provider sends, along with the introductory talk from the match commentator, one of three different sets of stickers: a set customized to Bahia fans, another set customized to Vitoria fans, and a third set customized to viewers who have not declared their preferences.
- During the match, the video is broadcast within an interactive application which presents stickers according to the viewers preferences, i.e., fans from Bahia watch the game with stickers supporting their team, and Victoria fans watch the game with stickers supporting their team — the stickers being selected and added by the broadcast producers.

This alternative means that fans from each team may have a customized watching experience: the broadcast is adjusted to include stickers supporting the team selected, using their set-top-box, by the viewers. The impartial broadcast, in which a neutral position is assumed, is presented only to viewers who have not chosen a particular team to support.

It may also be the case that, as soon as the broadcast provider starts offering this customized alternative, the official sponsors of the Championship inform their interest in customizing their propaganda to take advantage of the preferences declared by viewers.

Discussion

The discrimination of segments of continuous media can be used by numerous applications, ranging from simple selections performed by users to more robust applications in educational or training environments, for example. As suggested by the scenarios described above, properly designed applications can be built that, making use of the scenes discriminated, are able to offer a variety of services of interest to viewers in particular or communities in general.

The scenarios presented above illustrate particular situations in which the Watch-and-Comment paradigm (Cattelan et al., 2008) can be exploited: all scenarios assume the selection of scenes or of programmed options, and all scenarios exploit the fact that the users can interact with the video they watch using a remote control or their own mobile device. It is worth noting that, in this case, the *comment* (from the watch-and-comment point-of-view) corresponds to the discrimination operations associated with the scenes, performed by, for instance, typing keys on a mobile device according to the semantic established by the application.

It is important to observe that the discrimination information can be used at the time of the capture, as in example allowing the sticker-based annotation of a football match, or in future opportunities, as in the examples involving students of karate and medicine. In all cases, the fact that the selected moments may be recorded in the viewers' mobile devices imply that applications can process that data at later opportunities: these applications process the discriminated media moments and media intervals captured in a distributed, synchronous or a asynchronous way.

The adoption of personal portable devices, such as mobile smartphones, demands the use of a protocol that allows the discovery and the negotiation of services automatically – as it is the case with the UPnP protocol. It also demands the provision of accessible and intuitive user interfaces – always a challenge when involving viewer-TV interaction or user-mobile device interaction. Moreover, the overall communication and information capture and retrieval must be supported by the underlying middleware.

We argue that the value of the specialization of Watch-and-comment paradigm by means of the collective discrimination operations lays on its simplicity and generality. We propose integrating the facilities provided by an interactive TV middleware with those which can be offered by personal mobile devices to permit the implementation of the applications discussed in this section.

2.2.3 Discrimination of Media Moments and Media Intervals

Allen and Hayed (Allen & Hayes, 1990) have discussed interpretations of moments, points and intervals related to time. They remark that, for a *time point*, it makes no sense to think about time. This is because a point in time has no quantifiable amount of time associated, as in the transition at the beginning of a race, for example, from the race not being in progress to the race being in progress. A *time moment* however, despite having the common-sense of the brevity corresponding to instantaneous events, has a short time associated, whose amount depends very much on the context. In a movie, for example, a *time moment* may be a frame, which duration is a fraction of second defined by the frame rate used. In a car race, for instance, a *time moment* may be associated, from the point of view of the viewer, with an overtaking with a duration of few seconds – while one car overtakes the other.

Allen and Hayed also interpret *time interval* as being the time corresponding to duration of events. The time a car used to complete a lap may be an example of time interval. It may be delimited by points or by moments. Although a frame in a movie has an associated duration, it makes no sense to define one time interval for a movie during the time moment of a frame.

The interpretations in this work for *continuous media moment* and for *continuous media interval* borrows concepts from Allen and Hayed (Allen & Hayes, 1990). A *continuous media moment* is an *instantaneous* segment of the media which takes a point m as a media offset, and a brief tolerance Δ before and after the offset m. Similarly to Allen and Hayed's definition for *time moment*, a tolerance Δ defines the meaning that the word *instantaneous* has in a given the context – in our case, the context is continuous media.

Operator: Discrimination of Media Moments

Considering M_i an instance i of a media M, a media moment is given by a point m as the media offset and a tolerance Δ relative to before and after that offset, i.e., a *media moment*

$$m^i_{a,\Delta} = M_i[m_a - \Delta, m_a + \Delta]$$

is discriminated by the *media moment* operator $\overset{\Delta}{\leftarrow} \overset{\uparrow}{m_a} \overset{\Delta}{\rightarrow}$ as shown in Figure 2.15(a).

This *media moment* operator can be implemented by applications which demand viewers to select moments in the video they watch: an example is an application which allows karate students who watch a match to use their smartphones to discriminate moments in which a particular strike is used.

Operators: Discrimination of Media Intervals

For the discrimination of a *continuous media interval*, we define two operators:

- A *media interval* is a segment of media that has a point m as media offset and a duration of $\alpha + \beta$. A *continuous media interval*

  $$mi^i_{b,\alpha,\beta} = M^i[m_b - \alpha, m_b + \beta]$$

  is discriminated by the operator ${}^{\upharpoonleft\alpha}\ m_b\ {}^{\beta\upharpoonright}$ as shown Figure 2.15(b).

- Alternatively, a *media interval* may be defined as a segment of media delimited by two time moments. A *continuous media interval*

  $$mi^i_{c,d} = M^i[m_c, m_d]$$

  is discriminated by the operator $\hookrightarrow m_c\ m_b \hookleftarrow$ as shown in Figure 2.15(c).

The *media interval* operators can be implemented by applications demanding viewers to discriminate segments of the video programs they watch. An example is an application which allows medicine students who watch a video from a surgery procedure to use their smartphones to discriminate segments in which a particular technique is used, or moments they want to discuss in a future class.

Matching of media moments and media intervals

The parameters α, β and Δ should be adjusted depending of the context and of the kind of the applications. With this flexibility we define:

- *Matching of media moments*: the matching of two media moments $m^i_{a,\Delta}$ and $m^k_{a,\Delta}$, discriminated over two different instances i and k of media, with the same reference, is *true* when

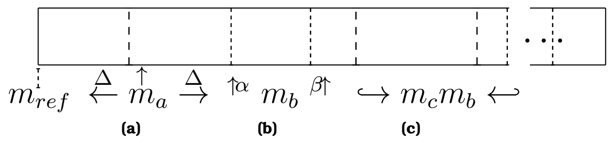
  $$mm^i_{a,\Delta} \cap mm^k_{a,\Delta} \neq \emptyset$$

- *Matching index of media intervals*: is given by an index that represents how coincident two media intervals discriminated over two different instances i and k of media are, based on the same reference, and is calculated by the ratio

$$(mi^{i}_{b,\alpha,\beta} \cap mi^{k}_{b,\alpha,\beta})/(mi^{i}_{b,\alpha,\beta} \cup mi^{k}_{b,\alpha,\beta})$$

The *matching of media moments* can be implemented by applications which, when processing the input from several users, should consider two media moments discriminated by different users as, in fact, the *same* media moment. On example is considering the moments karate students discriminate as *correct* if match the instructor's discrimination within a given tolerance.

The *matching of media intervals* should be implemented by applications that need to identify when video intervals discriminated by users have common segments, for instance. On example is an application which collects the media intervals discriminated by medicine students and computes how many of those intervals coincide with the segment discriminated by their Professor.



m_{ref}
$\overset{\Delta}{\leftarrow}$
m_a
$\overset{\Delta}{\rightarrow}$
$\uparrow\alpha$
m_b
$\beta\uparrow$
$\hookrightarrow m_c m_b \hookleftarrow$
(a)
(b)
(c)

Figura 2.15: (a) Continuous media moment given by media offset and tolerance Δ. (b) Continuous media interval with one media offset and duration of $\alpha+\beta$. (c) Continuous media interval with two media moments (each with its own defining media offset and tolerance)

Defining Stickers with Media Moments and Media Intervals

In our scenarios, users can use media *stickers* to enrich their viewing experience. Stickers may be atemporal (instant-based stickers), or may have an intrinsic time duration. The latter are interval-based stickers such as animations, chants and hymns. Examples of instant-based stickers which do not have a time dimension are graphics and photos.

To be perceived by viewers, the presentation of a sticker must be associated with a time interval. Therefore, instant-based stickers must be specified by a media moment, as discriminated by users, and a duration: the overall result in a continuous media interval. For interval-based stickers, their presentation may be specified by a media moment, as discriminated by the user, and their intrinsic time duration. In both cases, the use of the operators for the discrimination of media intervals is straightforward.

When stickers are combined, their presentation should take into account both the instant of time they are activated and their duration: the value of both parameters may vary as a result of the application provided. For instance, if the combination results from discrimination requests made by different users involving different stickers, each request specifies its continuous media interval moment. The resulting discrimination operator may be specified by the earliest media moment and largest duration, which results in all stickers starting and ending at the same time. On the other hand, if the combination results from discrimination requests made by the same user, the resulting discrimination operator may be specified by the latest media moment and its associated duration, which results in all stickers ending at the time of the latest discriminated moment.

Discussion

In order to discriminate media moments and media intervals, one can add physical marks to the content or, as a better approach, use time references. A point in the continuous media – its beginning, for example – may be used as a reference when specifying moments. Also, a reference of presentation speed may be considered – usually, as default, the same speed used when the content was produced. Moreover, if the content was edited and is not the original one anymore, when mapping previous time markups (related to the original content), the new version of the content must be adjusted. Any edits over the original content, that change the size of the content (delete, insert), must be registered and kept as a content metadata. Figure 2.16 illustrates a portion of media which had part of its content removed (Fig. 2.16-top) and another part that has been duplicated (Fig. 2.16-bottom): it is important to observe that original time line is maintained.

In this work we assume that the discrimination of media moments and intervals is performed specifying time relative to some media presentation. Any version of the media, edited or not, should preserve its original time line, as illustrated in Figure 2.16.

Broadcast content usually does not have a precise starting point which could be taken as reference for discriminations. For interactive digital televi-

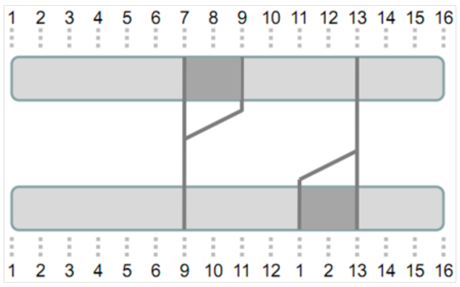


Figura 2.16: (top) Continuous media interval *(7-9)* removed. (bottom) Continuous media interval *(1-3)* copied before original time moment *13*.

sion, however, one can take advantage of the absolute time information the broadcasters have to send periodically, and use the absolute time information for discriminations purposes. This information is part of the transport stream and is usually updated in a table periodically (in the form of date, time, time zones of the country, etc.).[4]

The concept of distributed discrimination applies both when the discrimination is performed by different users and when performed over different presentations of some media. Two or more users, at their home, may perform discrimination operations over some content during its broadcast at different times. Moreover, discriminations may be performed on different recorded versions of the same media.

Applications may use several instances of the defined operators. As an example, one operator of type $\overset{\Delta}{\leftarrow} \overset{\uparrow}{m}_a \overset{\Delta}{\rightarrow}$ with code Ψ may be used to discriminate moments of type Ψ, while another operator of the same type but with code Ω may be used to discriminate moments of type Ω.

2.2.4 Prototype implementation

To provide the facilities detailed in the previous section – operators for the discrimination of moments and media intervals – for various applications with different functionalities, we implemented: (a) an extension to an interactive digital TV middleware which runs on the set-top-box, and (b) a service which runs on a user's mobile device.

Discrimination Service in the STB

The server portion of our implementation corresponds to the extension of a TV middleware. In our work we extend the Ginga-NCL middleware architecture (Soares et al., 2007) (Soares & Souza Filho, 2007), as shown in Figure 2.17: the Ginga-NCL original modules are presented in white (e.g. the *Formatter* in the Ginga-NCL Presentation Environment Layer), our modules are shown in dark gray, and one module which we extended from the original Ginga-NCL middleware is shown in light gray.

We have designed the extension to the original Ginga-NCL middleware architecture (Soares et al., 2007) (Soares & Souza Filho, 2007) so it can be used by any application that may want to take advantage of: (a) the UPnP protocol for communication; (b) the WaC paradigm for the capture of the interaction between user and the interactive TV applications; and (c) the presentation of

[4]In the European DVB and the Brazilian SBTVD digital television systems, the tables are called Time Offset Table; the American system ATSC calls it System Time Table.

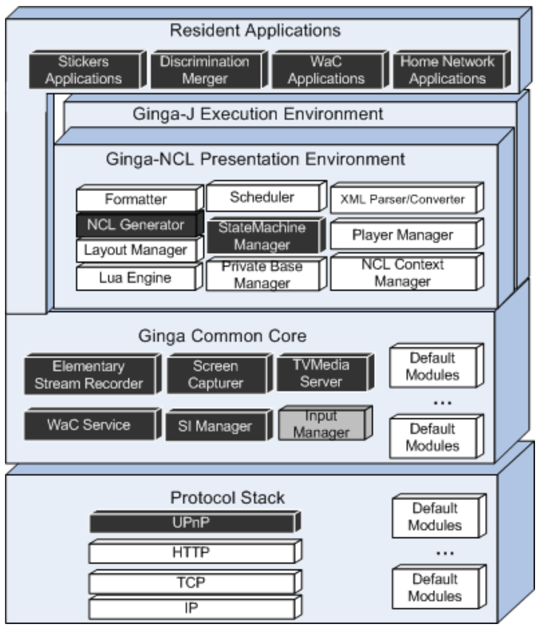


Figura 2.17: Ginga-NCL extension: original Ginga modules are indicated in white, modified Ginga modules are indicated in light gray, our modules are indicated in dark gray.

associated interactive programs generated on the fly by the server portion of application.

Our current extension includes modules from previous work (Teixeira et al., 2010a), with a new organization in the Resident Applications layer. More specifically:

- the new *UPnP module*, added to Protocol Stack layer, which implements code relative to the UPnP protocols so as to provide its functions to other devices throughout the network;
- the new implementation of the *Input Manager module*, which extends the original module to allow user input to be processed not only by the middleware declarative and procedural environments but also by other modules and native applications;
- new modules added to the Ginga Common Core layer:
  - *Elementary Stream Recorder module*: provides for the recording of all elementary streams in a way that it can be later referenced for presentation;
  - *Screen Capture module*: captures the selected video frame so that it can be presented later;
  - *TV Media Server module*: provides audiovisual content transparently in the network;

  - *System Information (SI) Manager module*: registers information relative to the synchronization of all elementary streams of the original program, and controls the later play back of that information in concordance with the additional user selections;
  - *WaC Service module*: offers the service of discrimination of moments and intervals in the network through the UPnP protocol, and allows residents applications to register to be notified about user interactions.

- a new *StateMachine Manager module* which, added to the Ginga-NCL Presentation Environment layer, informs the current state of a media presentation;

- the new *NCL Generator module* added to the Ginga-NCL Presentation Environment layer to produce an interactive multimedia document (in the NCL declarative language) aggregating the information captured by the other modules;

- a new *Discrimination merger module* which, added to the Resident Applications layer, merges the information stored in the users' mobile devices (device ID, channel ID, TV program ID, keys and timestamp) and make them available for applications;

- the *Matching Reporter module* existing in the previous work (Teixeira et al., 2010a) has been replaced by the *WaC Applications module*: this new module represents many applications that can be build by combining the timestamp and user identification information computed by the *Discrimination merger module* with the original media and the frames captured by the *Screen Capture module* to produce a matching report such as the one presented in Section 2.2.5.

The *WaC Service module* is the core module as far as the discrimination operators are concerned. The module treats independently each device associated with the viewers. Using this service, each device is responsible for transferring to the set-top box two parameters: the value of key pressed by the user and the identifier of who made the selection. These parameters are the basis for subsequent use in any applications demanding the discrimination of time intervals of continuous media.

The *Screen Capture module* is responsible for the capture of the image (video frame) displayed on the TV when a request for the discrimination of a scene is made. In the case of our scenarios, at the time of reviewing the annotations made by the viewers.

The *Elementary Stream Recorder module* captures a time interval of the media presented on the TV when a request for reviewing the discrimination of a segment is made. The captured frame or media interval is stored and associated with the discrimination performed by the viewer. The multimedia content generated from these modules is available in a network via the TV Media Server module, using the UPnP protocols.

The *Information System (SI) Manager module* is used in combination with Elementary Stream Recorder to obtain information relative to the beginning and the end of a TV program being transmitted (or played back) to permit, later, the synchronization of the documents.

The *StateMachine Manager module* provides information about the current state of the media (managed by the Ginga-NCL middleware). Information such as if the playback is in pause mode and what is the current time of the presentation are used by the NCL Generator module.

The *NCL Generator module* uses information from the SI Manager module and media generated by the Screen Capture and the Elementary Stream Recorder modules to create a NCL document which corresponds to the annotated media.

It is important to observe that our current extension reuses components implemented to support the collaborative annotations of points of interest presented in our previous work (Pimentel et al., 2009). In particular, our current implementation demanded the design of a new *WaC Service module*, as well as important modifications in the *NCL Generator module* (which was moved from the Ginga Common Core layer to the Ginga-NCL Presentation Environment layer).

Finally, we observe that an essential feature of the extended architecture is the support to a broad range of personal devices such as smartphones, handhelds and tablet PCs. As a result, our prototype implementation supports multiuser interaction with the interactive TV platform because the UPnP communication is not limited to a particular device type, operating system or programming language.

Discrimination Service in mobile devices

Running on the user's mobile device, the client portion of our prototype implementation is a service that implements a *UPnP control point* responsible for locating and using the services provided by TV set-top-box. The main advantage of exploring UPnP in this scenario is the use of the plug-and-play alternative to the communication network: this means that interactive applications can use the service without demanding the users to perform complex settings.

The overall architecture supports several types of devices, and the discrimination service running on the client device can be implemented in many programming languages. The proof-of-concept service we built was developed using the *Java Micro Edition* for mobile devices.

The main component in the discrimination service running in the client device is the UPnP *control point*. It is responsible for locating a UPnP device on the network, and for allowing the viewer to use the services provided by those devices. When the discrimination service is initiated on the client device, the search for UPnP devices is automatically performed. If a device with a service of the type *Watch and Comment* is found, the application automatically enables the viewer to use this service via a graphical interface on the device.

Once the Watch and Comment discrimination service is activated, the sequence indicated in the top portion of Figure 2.18 is initiated: when the viewer presses any numeric key in the mobile device keyboard, the device sends to the WaC module two parameters: the identifier of the device itself, and the identification of the key which was pressed. The module in the set-top-box replies back: the channel identifier, the TV program identifier, and the current timestamp.

The device identifier is generated automatically with information gathered from the device's operating system. However, our client module allows the user to change the device's identifier to other information, such as the user's name.

Discussion

The availability of the new and extended modules and services as presented in this section makes it possible the discrimination of moments and time intervals relative to some specific continuous media. The distributed aspect of the architecture is evident when the client sends a discrimination command and receives from the service provider, with the confirmation, the channel and the TV program identifiers as well as the timestamp. The timestamp identifies a moment in the TV program or in a stored media. Given that the TV station broadcasts a time reference (Time Offset Table) as part of the transport stream to make sure that all of the receivers have the correct time, the timestamp is an appropriate parameter for time-based annotations.

To further illustrate the sequence of events supporting the discrimination operators, Figure 2.18 shows the steps performed by viewers for the discrimi-

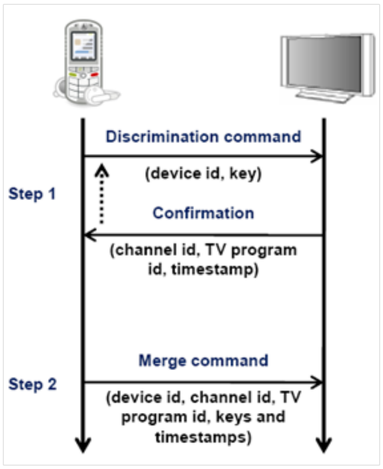

Discrimination command
(device id, key)
Step 1
Confirmation
(channel id, TV program id, timestamp)
Merge command
Step 2
(device id, channel id, TV program id, keys and timestamps)

Figura 2.18: Steps for discrimination (Step 1) and merging (Step 2) of moments and time intervals.

nation of moments and time intervals:

- In Step 1, the viewers make their annotations independently, possibly in different places — the data received from the set-top-box is stored on their mobile devices.
- In Step 2, viewers gather in one place. Each viewer may choose annotations based on the channel, TV program and date, and send to a receiver available in the room (discoverable automatically via the UPnP protocol). This receiver must contain the main (original) media stored, and should also contain the *Discrimination merger module* which merges the annotations received from all devices of users. The merge is based on the timestamp of each annotation. The information of annotations is made available via a Lua-based API, allowing NCL applications to built interactive programs using the corresponding information.

It is of paramount importance the fact that, once the discrimination operations are available, viewers can make discriminations on any interactive program presented in their TV set — the programs they watch do not have not to be designed to permit discriminations. The discrimination information generated by the viewers can be used by any new applications designed to take advantage of the availability of the moments and intervals that have been selected by the viewers — we present examples in the next section.

2.2.5 Proof of Concept Applications

To illustrate the use of the operators in the context of the infrastructure detailed in the previous section, we have build sample applications.

Later review using the discrimination of media moments and media intervals

Later review: Discrimination of media moments

Research efforts involving video and sports investigate many issues which include automatic personalized video abstraction (Nitta et al., 2009), methods for for event detection and retrieval (Dao & Babaguchi, 2010), and the automatic generation of structured documents describing motion trajectory (Yi et al., 2005), to name a few. We have built one application aimed at allowing the watch-and-commenting of sports videos, its use is outlined next.

A Judo instructor asks his pupils to watch a competition to be presented in the TV and to discriminate the moments in which they identify a specific grappling techniques being used by the competitors. Because the students have to use the numerical keyboard of their mobile phones, the number key corresponding to the technique to be observed is defined previously by instructor.

The instructor also watches the competition: while the competition is recorded in his receiver, the instructor registers the discrimination moments using the same number keys he asked the students use. Later, students and instructor meet in a room where a TV set is available. The students use their mobile to search for some available application in the TV set: the UPnP service in their mobile locates the *Matching Reporter* application running on the TV set. Instructed by their teacher, the students send the information stored in their mobiles relative to the Judo competition (the selection is made by the channel id, TV program id and timestamp as stored while the the students were watching the program). The *Matching Reporter* application activates the *Discrimination merger module* to compute the merging of the all annotations, and use the result to generate an interactive video which presents the *matching score* of all students in comparison with the discriminations made by the instructor — as in the example presented in Figure 2.19.

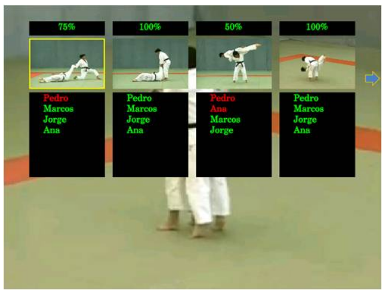


Figura 2.19: Comparison of discriminations made by judo students and their instructor.

Later review: Discrimination of media intervals

The processing of video in the medical domain has many applications reported in the literature. An example is the automatic summarization, for archival purposes, of video captured during a whole surgery (Lux et al., 2010). Another example is the automatic classification of video to allow summarization and skimming in online medical education scenarios (Luo et al., 2008). In a different domain, the processing of videos has been exploited to aid the psycholinguistic analysis of gesture and speech (Quek et al., 2002). We have built one application which can be use for the watch-and-commenting of videos from surgery procedures, as outlined below.

A Professor of medicine asks her students to watch the video of a surgical procedure to be broadcast in the TV university channel, and to use their mobile phones to discriminate media intervals corresponding to some points they want to discuss in a future class. In this case, the students just have to use two keys to mark the beginning and the end of a media interval. Later, in the classroom, the students send their annotations to a TV set (as in the example above). An application in the TV set receives the information, which is merged and processed to generate a new interactive video (Figure 2.20) which allows the Professor, along with her students, to review the annotations made by each student and to discuss the corresponding content relative to the surgical procedure.

Live editing using the discrimination of media moments and media intervals

Live editing: stickers added by users

Considering the scenario in which Dr. Prazeres and his friends watch the Bahia *vs.* Vitoria match, an interactive application allows them to add stickers as follows. When Bahia scores its first goal, Dr. Prazeres uses the remote control to add stickers as shown in Figure 2.21(a): an image containing the team logo is added first, and the Bahia hymn is added next. Elsewhere there

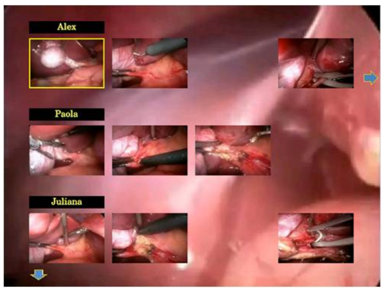


Figura 2.20: Scenes marked by students of medicine.

is an upset Victoria fan that adds a sticker with an ashamed version of his team mascot, the lion (Figure 2.21(b)).

As the match continues, Vitoria best player scores a goal. At this moment Dr. Prazeres becomes so frustrated that he sticks to the video an upset version of his team mascot during all the long moments in which the player celebrates his goal (Figure 2.21(c)). Elsewhere there is a Victoria fan that celebrates the goal by adding a sticker combining his team mascot and the team logo, followed by a sticker which plays back the Vitoria hymn (Figure 2.21(d)).

In our implementation, when the stickers are added to by users the original video is reduced to provide space to the stickers to be presented: they cannot be placed on top of the image because the current Ginga specification does not allow the broadcast video to be modified at the viewer location.

Live editing: stickers added by the broadcast server

When added at the server side, stickers can placed on top of the video during broadcast. We consider the scenario in which Dr. Prazeres' grandparent, called Dr. Serafim, has learned that he can press some of the remote control buttons when some alternative options are shown on his TV screen. When the Bahia *vs.* Vitoria match is about to start (Figure 2.22(a)), Dr. Serafim receives an option to select which team he supports (Figure 2.22(b)): he presses the green button firmly to indicate his devotion to Bahia, and then rests the remote control for the whole match.

When Bahia scores its first goal, Dr. Serafim is pleased to see that the image on TV shows his team mascot and the team logo (Figure 2.22(c)). Elsewhere there is a Victoria fan that observes that, when Bahia scoring that first goal, the TV presents an ashamed version of his team mascot (Figure 2.22(d)).

In this example, it is important to observe that video sent to all households is the same: in each TV set, a local application receives, along with the video stream, a command to add the local stickers to the image (the stickers were downloaded when the viewer selected which team he supports).

2.2.6 Related Work

In one of the earliest efforts in television-based communication, Coppens at al. (Coppens et al., 2004) allowed remote viewers using their AmigoTV system to add animations to their video — and the animation was shared among collaborating users. The presentation of extra contents associated with a program was also allowed in the CollaboraTV application (Nathan et al., 2008): the system allowed users to collaborate synchronous or asynchronously by selecting

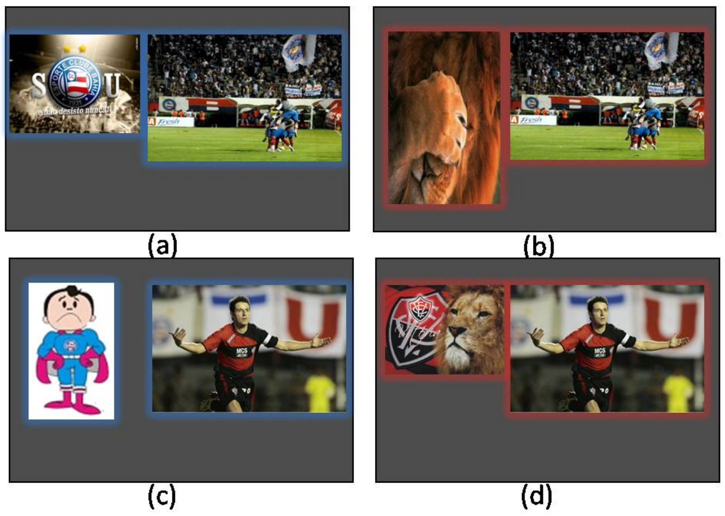


Figura 2.21: *Stickers* selected by the users during the match: when Bahia scores a goal, Bahia fans select a sticker supporting their team (a) while Vitoria fans select a sticker which shows their sadness; when Vitoria scores a goal, Bahia fans select stickers corresponding to their frustration (c) and Vitoria fans select stickers indicating their happiness (d).

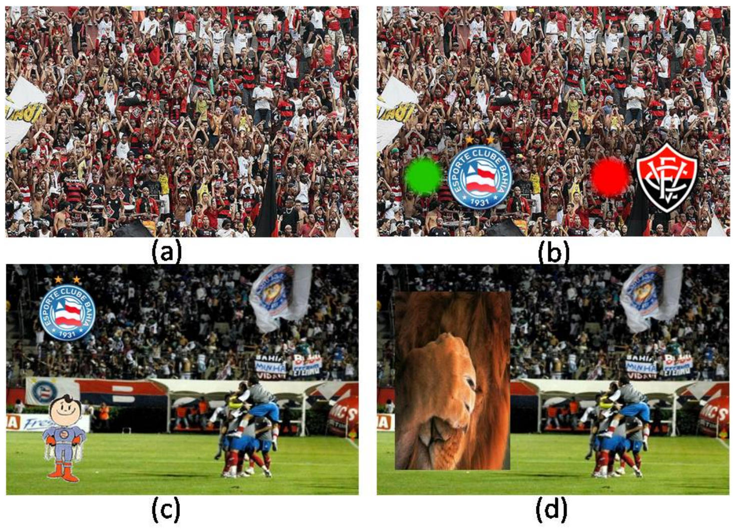


Figura 2.22: *Stickers* presented by the provider: At the beginning of game (a) the provider allows users to select (b) the team they support; during the match, when a goal is scored, a different animation is shown if the viewer supports the team that scored the goal (c) or the team that conceded the goal (d).

avatars to report the users' opinion by text or expressions associated with a particular time offset in the video. In both cases, the focus of the authors is on allowing the communication among remote users.

The idea of integrating the TV to other devices has been exploited with different goals in various research projects (Cabrer et al., 2006) (César et al., 2009) (Lucena et al., 2009) (Sakamoto et al., 2005) (Tkachenko et al., 2004). Our proposal involves integrating the TV to a networked environment to provide services that allows multiple users – possibly using different devices types – to interact with the TV to select, vote, delete, sort, evaluate, respond or take any other action that can be characterized by discrimination of moments or time intervals of continuous media. As far as the use of UPnP as the underlying communication protocol is concerned, our work is most related to that reported by Holbling et al. (Hölbling et al., 2008), who reported multiuser access to interactive TV platforms using the UPnP architecture.

In our current work, the discrimination of time intervals of continuous media is based on capturing (a) temporal information, (b) the type of discrimination, and (c) the identification of the user who performed the action: this information is registered as a structured hypermedia document, and allows applications to edit the multimedia documents presented to users in the interactive TV platform. As a result, our current approach complements our previous results relative to supporting the editing at the end-user side in the context of the Watch-and-Comment paradigm (Cattelan et al., 2008) (Pimentel et al., 2008b) using a p2p-collaboration infrastructure that registers comments made via voice, typed text, and pen-based ink marks, which are associated with individual frames and or video segments.

As far as the annotation on continuous media is concerned, the literature reports works with different purposes. Goularte et al. (Goularte et al., 2004) investigate the use of annotation on video frames by pen-based electronic ink or voice recognition – the annotations are, in both cases, converted to text so that an XML document can be generated to reference to all media corresponding to each annotation session. A comprehensive approach to enrich multimedia interactive documents has been proposed by César et al. (César et al., 2008), with focus on secondary screens. In another research effort, Costa et al. (Costa et al., 2002) propose a model in which annotations are made based on temporal data and multi-tracks, and each track has a perspective of the same video content. In a multi-user environment, the collaborative aspect can be exploited through the tracks: users define tracks and, then, are able to make annotations. In a complementary approach to our work, Ramos and Balakrishnan (Ramos & Balakrishnan, 2003) exploit various techniques of visualization and interaction for fluid navigation, segmentation, linking and annotating digital video by means of digitizer tablets supporting pen-based input.

As it has been illustrated in this section, the theme of annotation on continuous media is important and has been exploited for various purposes. Further research topics include natural interaction (Goularte et al., 2004), models (Costa et al., 2002), and techniques of fluid interaction (Ramos & Balakrishnan, 2003).

The research we present in this paper is related to several of the aspects reported by the work cited in this section – we argue that the simplicity, ubiquity and generic aspects of our proposed model permits designing collaborative and distributed applications which use interactive TV platforms.

2.2.7 Final Remarks

In this paper, we have proposed an extension of the Watch-and-comment paradigm by allowing interactive TV viewers to discriminate media moments and media intervals using their own personal mobile devices. We have shown that the discrimination information can be used at the time of the capture, as in example allowing the sticker-based annotation of a football match, or in future opportunities, as in the examples involving students of karate and medicine. In all cases, the fact that the selected moments may be recorded in the viewers' mobile devices imply that applications can process that data at later opportunities: these applications may use of the discriminating media moments and media intervals captured in a distributed and asynchronous way.

Our work involved implementing extensions on the Ginga-NCL middleware to demonstrate the feasibility of the implementation of compelling applications that make use of the captured discrimination information. Our extension uses a protocol for service discovery so as to allow handheld devices to be used by users to discriminate media moments and media intervals. The applications we implemented illustrate that the discrimination operations have potential to offer a good degree of ubiquity to applications.

Our current efforts involve supporting the Zeroconf protocol[5], which will allow the use iphone and similar devices for discriminating media moments and intervals. As next steps, we plan first to extend the prototype applications introduce in this paper, so they can be used in usability evaluations. Next, we plan to evaluate the applications in real settings involving karate training sessions, a medicine class and football matches. We also plan to formalize

[5]http://www.zeroconf.org

an API corresponding to our current implementation to facilitate the building of novel applications. In particular, we plan to extend our API to permit its deployment in an OSGi-based implementation such as the one presented by De Lucena et al. (Lucena et al., 2009).

2.3 Multimedia Multi-device Educational Presentations Preserved as Interactive Multi-video Objects

The capture of lectures or similar presentations is of interest for several reasons. From the attendee's perspective, students may use the recordings when working on homework assignments or preparing for exams, or to watch the contents of a missed class. From the instructor's perspective, a captured lecture may be evaluated, recaptured for improvements, or reused as complementary learning material. Moreover, captured lectures may be a valuable resource for e-learning and distance education courses. In this paper we detail the design rationale associated with the development of a prototype platform for the ubiquitous capture of live presentations and their transformation into a corresponding interactive multi-video object. Our approach includes capturing important context information which, when incorporated into the multimedia object, enables one to interact with the recorded lecture in novel dimensions. We tested our prototype by using case studies involving instructors and students, which allowed us to identify important features and novel uses for the platform.

2.3.1 Introduction

The literature reports several efforts that focus on the preservation of lecture presentations (e.g. (Brotherton & Abowd, 2004), (Canessa et al., 2009), (Pimentel et al., 2007a), (Dickson et al., 2012)). In our work, the term *educational presentation* has a broad meaning that includes not only traditional lectures but also related activities, such as problem solving sessions.

We are aware that there are strong divergences among educators as to the efficiency of the lecture format as a method of instruction. For instance, while Bauerlein (Bauerlein, 2011) remarks that "one of the axioms of progressivism education is that the lecture format is an inefficient and, potentially, alienating method of instruction", Schwerdt and Wuppermann (Schwerdt & Wuppermann, 2011) remark that "contrary to contemporary pedagogical thinking, we find students score higher on standardized tests in the subject in which their teachers spent more time on lecture-style presentations than in the subject in which the teacher devoted more time to problem-solving activities". Although the model of interactive multi-video lectures such as the one described in this work may impact on the teaching or the learning experiences, such discussion is out of the scope of this paper. We present the rationale applied in the development of a prototype environment for the ubiquitous capture of live presentations and transformation of the presentations into equivalent interactive multi-video learning objects.

In our work, capturing a (multimedia) presentation means (using multiple devices toward) recording one or more video and audio streams of the speaker, images presented on the screen or projector, writings and drawings made on whiteboards, and also capturing contextual information that is relevant to the corresponding playback. The captured information is then transformed into synchronized video streams and static media that compose the resulting multimedia object. We refer to the result as an interactive multi-video object because it is mainly composed of multiple synchronized videos. Therefore, we call an *interactive multi-video object* the composition of several videos, audio and static media, properly synchronized and with facilities for flexible interaction and browsing.

The problem. Although recording lectures is a common practice (in universities), producing quality video lectures demands high operational costs (cameraman, video director, editors and other audiovisual professionals). To reduce the operational costs, in the past decades many tools for the automatic capture of lectures were developed (e.g, (Brotherton & Abowd, 2004), (Canessa et al., 2009) (Chou et al., 2010), (Dickson et al., 2010), (Halawa et al., 2011), and (Nagai, 2009)). However, most capture tools for the educational domain record video only and slide streams and generate, as a result, a *single* video/audio stream as a lecture video or podcast. The value of multi-video has been recognized in scenarios that include newscasting and real-time videoconferencing.

Our proposal. We propose a capture and access-based model targeted at capturing much of the content presented in a classroom. The capture process, pervasive and without human mediation, triggers the automatic generation of an interactive multi-video object associated with the lecture. We built a prototype platform for the ubiquitous capture of live presentations and their transformation into a corresponding interactive multi-video object. Our approach includes capturing important context information which, when incorporated into the multi-video object, allows one to interact with the recorded lecture in novel dimensions. The student may be able, for example, to use a multi-window panel to review multiple synchronized audiovisual content that includes the slide presentation, the lecturer's web browsing, the whiteboard content, video streams with focus on the lecturer's face or the lecturer's full body, among others. The student has the option to select, at any time, which video object is more relevant to be exhibited in the main (larger) screen. The student is also able to execute semantic browsing operations using points of interest like slide transitions, spoken keywords, lecturer's interactions, etc. Moreover, facilities can be provided for users to annotate the captured lecture while watching it, as suggested by the watch-and-comment paradigm (Cattelan et al., 2008).

Case studies. We evaluated our proposal by means of case studies in which we invited instructors and students to use the system in different settings. We report data relative to multi-video objects generated automatically by the prototype, and also summarize data relative to the interactions students had with the interactive multimedia objects generated. For instance, the main operation performed by the students was the selection of which component of the multi-video object was to be presented in the main (larger) window. Also, an important feature of the prototype is the logging of all interactions performed by the users while watching the lecture: this allows a detailed analysis of which portions of the presentations students review the most, for instance, which is useful (Brooks et al., 2013) for instructors to evaluate their own performance during the lecture (Viel et al., 2013a).

This paper is organized as follows: in Section 2.3.2 we discuss related works; in Section 2.3.3 we describe our proposed model to preserve educational presentations, discussing decisions that guided our design; in Section 2.3.4 we present our current prototype and describe a multi-video object generated by a live presentation; in Section 2.3.5 we discuss case studies in which instructors used the prototype to capture lectures; and in Section 2.3.6 we present the conclusions and future works.

2.3.2 Related Work

Most of the systems designed to record lectures use, instead of multi-video objects, a single video stream as the resulting product of a captured session (e.g. (Bianchi, 2004), (Canessa et al., 2013), (Chou et al., 2010), (Lampi et al., 2008), (Nagai, 2009)).

In the work of Liu et al. (Liu & Kender, 2004), lectures are captured in a similar process to the aforementioned ones, resulting a single video stream. The difference is that the set of slides used in the presentation is added to the video stream. However, the slides are not synchronized with the video.

Given that the result is single-video-stream, students do not have autonomy to choose the camera that gives them the best view of the lecture for each situation, or to focus on a particular point of interest .

ClassX is a tool designed for online lecture delivery (Halawa et al., 2011) (Pang et al., 2011). A live lecture is captured by means of a high definition camera (AVCHD) stream split into several virtual standard resolution cameras. By using tracking techniques, the most appropriated virtual camera for a given moment is chosen and streamed to the remote students. The students also have the opportunity to choose a different stream from another virtual camera or even watch the original AVCHD stream. A synchronized slide presentation is also offered, but it does not offer navigation facilities.

Schulte et al. (Schulte et al., 2008) propose the REPLAY system for producing, manipulating and sharing lecture videos, which is similar to the above-mentioned ones. Two differences must be pointed out: the use of computational vision to recognize written words, and of MPEG-7 to index the videos. REPLAY allows semantic navigation even though it does not produce a multi-video object.

Dickson et al. (Dickson et al., 2010) (Dickson et al., 2012), Brotherton and Abowd (Brotherton & Abowd, 2004) and Cattelan et al. (Cattelan et al., 2003) present more advanced features for capturing context information based on image processing and audio transcription. In these works, hypermedia documents are generated with interfaces that provide several ways of indexing the recorded information.

The model for capturing and recovering lectures presented in this paper allows more flexibility than the ones above mentioned. The flexibility comes from the ability to specify which context information is captured, how the context information is combined to generate alternative navigation interactions in the resulting multi-video object, and to promote live interventions in the classroom during the capture process — for example, when there is a change in the illumination of the room.

2.3.3 Capture & Access Model

In order to produce quality lecture videos, the traditional lecture recording process usually requires the presence of audiovisual professionals, such as cameramen to operate the cameras, a video director to select the best frame or camera, and an editor to generate the final version of the lecture video. These professionals increase the operational cost of producing a lecture video and may prevent the wide adoption of lecture capturing. Furthermore, their influence in the lecture may add noise to the communication process between the instructor and the students, making the resulting learning object less effective. Our proposal works toward a self-service approach, allowing instructors to record lectures themselves.

Some existing solutions rely on computational vision, tracking techniques and sensors to perform camera orchestrations in a attempt to produce a single video or audio stream output. The model we propose goes a step further, as depicted in Figure 2.23. It aims at capturing the lecture in its entirety by recording much of the content presented in the classroom. The capture process is pervasive, does not rely on human mediation and automatically generates an interactive multi-video object designed to preserve as much of the lecture content and context as possible.

Our model demands an environment (usually a classroom) which is instrumented with physical devices (Figure 2.23(1)) such as video cameras, microphones, (interactive) whiteboards and projectors. The instrumented classroom may also contain sensors, such as temperature and luminosity sensors, and secondary screens, such as notebooks, TVs, tablets, etc. The video cameras should be positioned where it is possible to frame important views of the environment (instructor, students, whiteboard, slides, etc.).

Computer devices capture the content produced by the physical devices used in the classroom (e.g. whiteboards and slides) and represent them as video, audio and data streams (Figure 2.23(2)). Cameras produce video (and audio) streams, microphones produce audio streams, and sensors produce data streams. By capturing the screen output from the secondary screens or by intercepting the signal sent to the slide projector, we can also produce video streams. The electronic whiteboard can produce both data and video streams: by capturing ink strokes, we can generate a data stream, and by intercepting the output signal sent to the projector, we can generate a video stream.

All captured streams are stored (Figure 2.23(3)) for further use in the multi-video object generation. The streams are also sent to the *capture controller* (Figure 2.23(4)), a component responsible for managing the capture process. The *capture controller* uses signal processing to analyze the captured streams and to send commands (Figure 2.23(5)) back to the physical devices and actuators (Figure 2.23(6)) located in the classroom.

The instructions in the *capture controller* are defined in a customizable action table. The action table is used to define actions for events that may occur during the capture process, such as zooming into the image of a specific camera, when the lecturer starts talking, or activating an actuator in order to reduce the light intensity, when the lecturer starts a slide presentation.

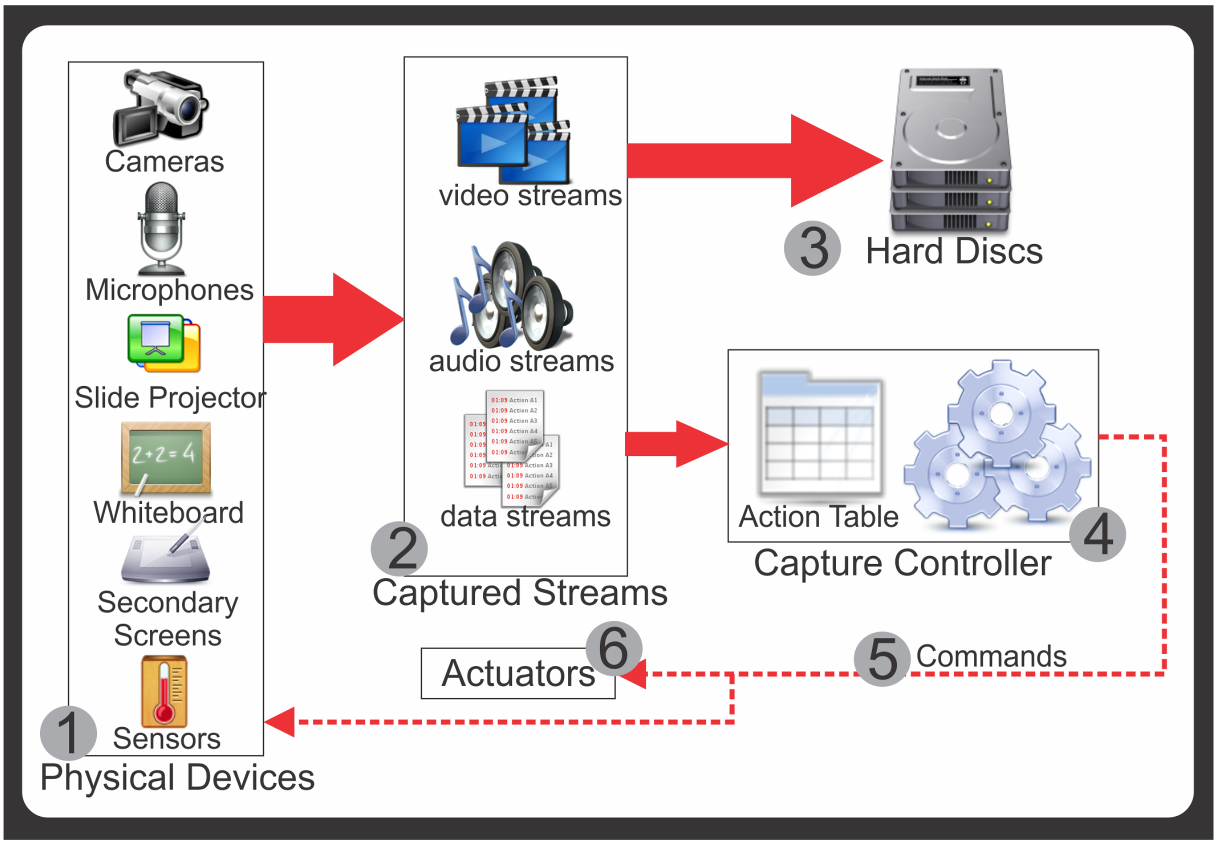


Figura 2.23: Capture Workflow

Our model allows the instructor to record a full lecture using several short-duration modules, which means that a multi-video presentation can be composed of one or more modules. This does not only allow the lecturer to take breaks during the recording process, but also to associate different modules to specific tasks. The lecturer may, for instance, prepare a problem solving presentation with one exercise per module. It is important to observe that students are able to navigate among the modules of the multi-video presentation (e.g., by skipping from one exercise to the other). Reusing modules to compose a new presentation is another advantage of splitting the recording process into modules — reuse is in fact one of the main aims of learning objects (McGreal, 2004).

All the analysis and conversion process involved in the captured streams can demand a lot of computational power and time. Once the capture of the streams is concluded, all the corresponding information is transferred to a server. The video, audio and data streams (Figure 2.24(1)) are sent to the *Recognition Module* (Figure 2.24(2)). The *Recognition Module* is composed of several recognizer components and each of them uses one or more captured streams to detect points of interest. Points of interest are moments in the lecture that, because they carry context information, may act as useful anchors

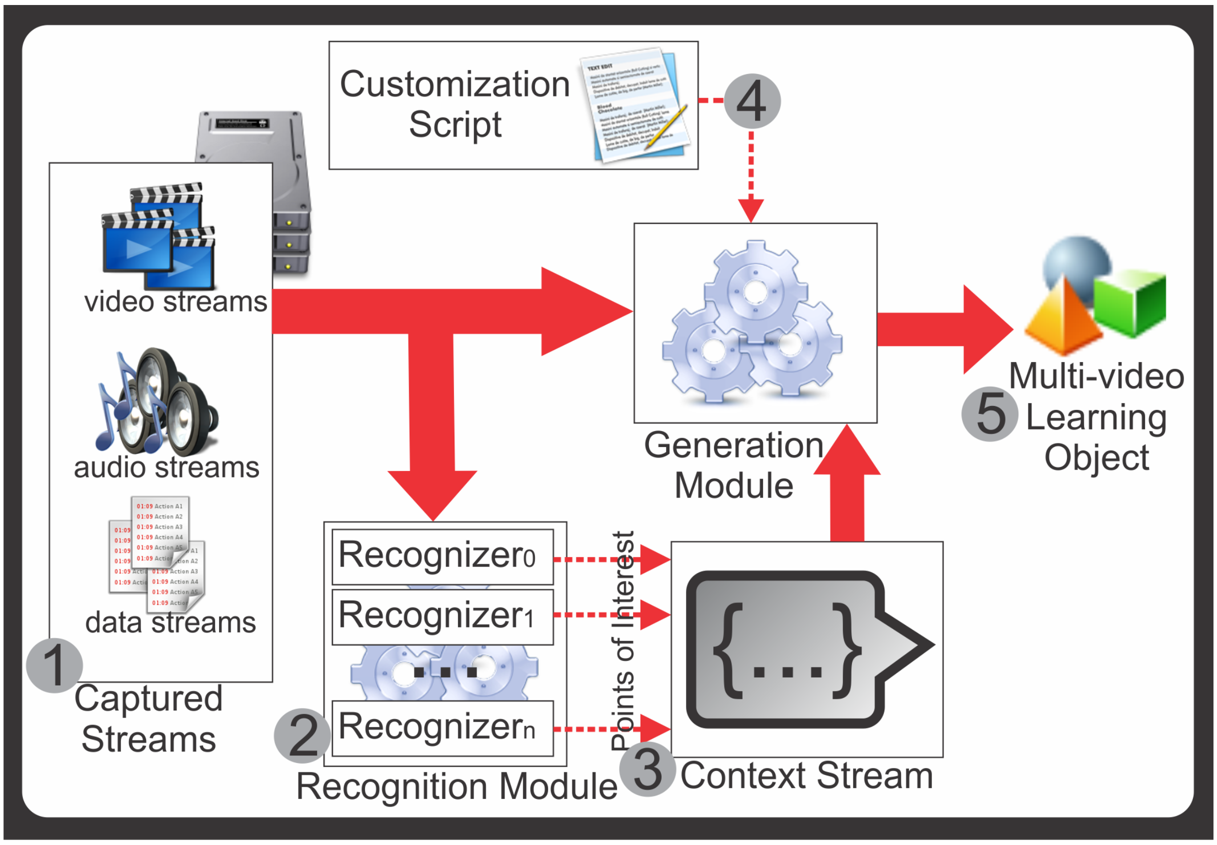


Figura 2.24: Multi-video object generation

for students and instructors. They may represent changes in the context of the lecture (e.g., the instructor moves from writing on the whiteboard to executing a video or program-based demonstration in his notebook), or moments of particular importance for the course (e.g., the instructor makes comments related to assignments, homework or exams). The points of interest can be used to provide semantic-based navigation on the multi-video object, allowing students to seek for the next slide transition, for instance.

Since many types of points of interest can be conceived by using different sensors, devices and other computer-based analysis techniques, the *Recognition Module* must publish an interface to allow developers to add new recognizer components as needed. By combining all the identified points of interest, the *Recognition Module* generates the *Context Stream* (Figure 2.24(3)). As a result, the *context stream* contains information about what and when something happened during the lecture.

The *Generation Module* in Figure 2.24 consumes all streams (context, audio, video, etc.) to generate a multi-video object. Content and format customizations of the multi-video object may be necessary in order to best suit (i) a course profile (e.g. a course which only uses a whiteboard does not need a slide presentation stream), (ii) the lecturer's style (e.g. if the instructor says "for instance" too often, it may not be a selective enough keyword), and (iii) the students' particular needs (e.g. some adaptation on the images for students with visual impairments may be necessary). The customization can be defined by a *Customization Script* (Figure 2.24(4)), which is used as input by the *Generation Module*. In the current version the customization is defined by an *ad hoc* set of rules, but in future versions they can be described in a declarative language similar to Business Modeling Language (Corallo et al., 2007).

The multi-video object (Figure 2.24(5)) is composed of videos and other captured media — with the association of the appropriate metadata, they should become multi-video learning objects[6]. Although the multi-video object is not able to reproduce the live lecture experience in its totality (which would involve live interactions, odors, temperature, etc.), it offers several interaction alternatives for the students while they are watching the lecture. These may diminish the loss of not having the students present in the classroom, when direct interaction between instructor and students usually bring benefits to the learning process.

2.3.4 Prototype

As a proof-of-concept of the model presented in the previous section, we developed a prototype tool for capturing lectures and generating multi-video learning objects. This prototype was mainly developed in Python.

Figure 2.25 depicts an overview of the prototype tools. The prototype is composed of three main parts: the *Capturing tool* used to capture streams; the *Processing tool* in charge of stream analysis and the generation of the multi-video object; and the *Presentation tool*, which allows the user to playback the multi-video object.

The capture tool prototype was deployed in an instrumented multi-purpose classroom (Figure 2.26). In the front of the classroom (Figure 2.26(a)) there is a conventional whiteboard, an electronic whiteboard and a notebook in which the presenter can browse the web or use any other software. The interactive whiteboard can be used to present slides (there is a Bluetooth presenter to control the presentation) and it allows drawing and writing over the screen. At the back side (Figure 2.26(b)) we placed two AVCHD, each one framing the interactive or the conventional whiteboard. We also placed a webcam as a wide-shot cam, framing the whole front of the room. As the room is a shared space in the university, the equipment is stored in lockers.

[6]http://www.ieeeltsc.org/

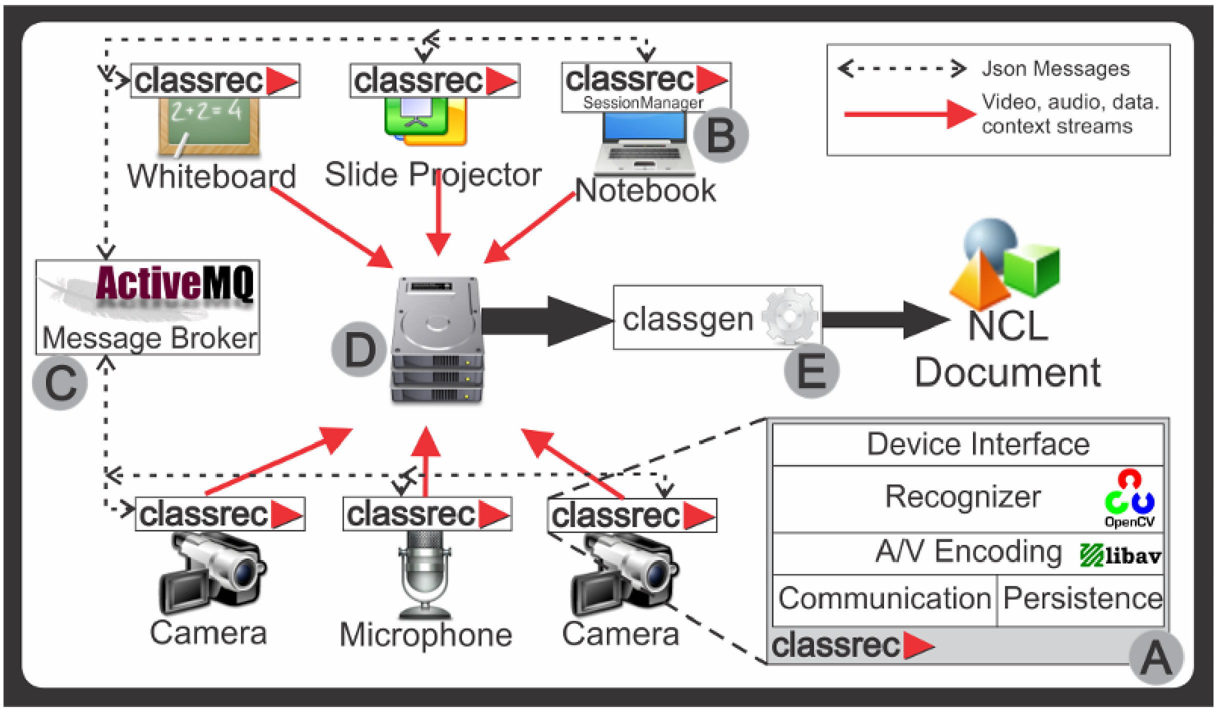
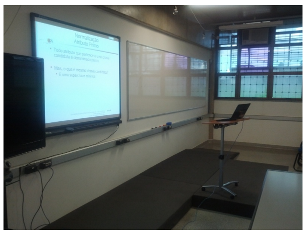


Figura 2.25: Prototype Overview


(a) Front

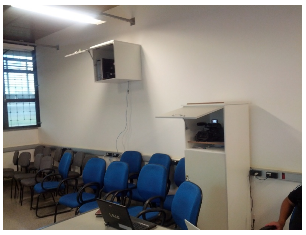

(b) Read

Figura 2.26: Instrumented Room

Capturing tool

The *Capturing tool*, named *Classrec* (Figure 2.25(A)), carries out the capturing process. Each computer used in the capturing process runs an instance of *Classrec*, and one of these instances is selected to be the session manager (Figure 2.25(B)). It corresponds to the *Capture Controller* of the workflow (Figure 2.23). The session manager is responsible for handling the lecturer's stimulus and for controlling the other *Classrec* instances, keeping them synchronized.

The capturing process is based on video streams. *Classrec* captures content (video and audio streams) produced by AVCHD and outputs produced by computers (such as computer screens, slide presentations, etc.).

We opted to capture the electronic whiteboard output as a video stream instead of its strokes. This was done because a video stream is more portable than strokes and, given modern video encodings, such as H.264/MPEG-4, and the static nature of whiteboard outputs, the bit rate of the video stream is low. We could record a stroke stream, but it would require a specialized engine to play it back (as it happens with other systems).

Some streams, such as slides, whiteboards, computer screens, can have segments with a lot of static content, but they are still captured as video streams. A possible improvement would be to replace the video for a combination of non-static content videos and a single image to represent a static segment (video with no changes during a period of time).

Classrec records audio/video streams using the *libav* library[7]. It also records metadata about the lecture, such as module structure, available streams and authoring information in an XML file named *XML lecture*.

The communication among the different applications is carried out using the Apache ActiveMQ message broker (Figure 2.25(C)).

In the instrumented classroom, all that lecturers who wish to record their educations presentations need to do is to press a button to start the capturing process and then they can deliver lectures as usual, whether to an audience of students or to no one. When they have finished delivering the lecture, all they need to do is to press another button. The *Classrec*'s instances carry out all the complex process of synchronously recording the different streams from the different devices and sent then to the Processing Tool.

[7]Libav - http://libav.org/

Processing tool

The *Processing tool*, named *Classgen* (Figure 2.25(E)), performs the multi-video generation process. *Classgen* uses information relative to the video streams and other metadata recorded by *Capturing tool* as input that is registered into the *XML lecture* file. It also supports an XML configuration description language, which allows the specification of which recognizers (and its inputs) and which codecs should be used to encode audio and video.

We not only considered points of interest suggested in the literature (e.g. (Brotherton & Abowd, 2004), (Cattelan et al., 2003), (Dickson et al., 2012)), but also designed novel ones. Points of interest implemented are:

- Module transition: usually denotes a change in the explanation context.
- Slide transition: similarly to Module transition, it also usually denotes a change in context.
- Whiteboard interaction: the starting instant of a sequence of annotations made on the whiteboard, which usually denotes an explanation.
- Lecturer close-up: the instructor may explicitly look at the camera placed in the front the the classroom, which allows capturing a face close-up, to create a corresponding point he wishes to emphasize.

It is important to observe that, in the current version of our tool, instructors and students are already offered options for highlighting their own points of interest in the form of comments and annotations, while watching the multi-video document.

We are working on to offer other points of interest, e.g.:

- Secondary device interaction: to mark points in which the instructor uses a secondary device (e.g. to show a video or to run a program-based demonstration).
- Spoken keywords: words that denote some context relative to the lecture. They may be generic as “important” or “exercise”, or lesson-dependent as “Lake ecosystem” or “Shakespeare”.
- Change in voice intonation and speech speed: can denote that some lessons are important or complicated.
- Students’ intervention: can denote a meaningful question or addition to an explanation.
- Lecturer’s gestures: can denote that something is important or indicate the connection between the different views of the contents (such as whiteboard and slide). The processing will focus on gestures made during the lecture.

Classgen uses the OpenCV library(Bradski, 2000) to perform pattern recognitions in order to identify points of interest for composing the context stream. The media manipulation during the orchestration process and the audio/video conversion is handled by the libav library. We have implemented recognizers capable of detecting (i) the presence of a lecturer in a video stream; (ii) whether the lecturer is facing a camera; (iii) slide transitions; and (iv) interactions with electronic whiteboard or PC.

The detections (i) and (ii) are performed by using *Haar* training. We check 3 frames every second in order to look for objects such as a face or a full body. A frame that contains the object is a success, and a frame that does not contain the object is a fail. We consider the beginning of a point of interest when there are at least three successes in a row; the point of interest finishes when there are three fails in a roll. We also established a minimum duration and a time distance for the point of interest. For instance, for a close-up of the instructor to be considered a point of interest, it should last at least 5 seconds and be 5 seconds apart from another close. If two closes are detected in a period of time shorter than this time, they are combined as a single point of interest. These values are configurable so as to take into account false positive matches.

For points of interest of types (iii) and (iv), we compare subsequent frames in order to detected variations. For instance, when there is a significant difference between two subsequent frames from the video stream generated from the slide projection capture, we identify a slide transition point of interest. We also group near points of interest of the same type as a single point of interest.

It is also possible to specify an orchestration of video streams in order to produce a new video stream. This is useful in environments with multiple cameras recording different angles of the lecturer. Through the XML configuration description language, it is possible to select which stream will be used in the orchestration and how to orchestrate them. For instance, it is possible to specify that when a recognizer detects the lecturer's face in video segments, the camera orchestration stream should include that segment.

After performing the recognition, orchestration and video conversion, the information generated by these processes (the points of interest, the orchestration stream, the converted streams) are stored in the *XML lecture*. The XML is then passed to a component of the *Processing tool* responsible for generating the final multi-video object (Figure 2.25(5)). Our prototype generates NCL[8] (ABNT Associação Brasileira de Normas Técnicas, 2007) documents, but the *Classgen* can be extended to generate other types of multi-video objects, such as HTML5 pages or stand-alone desktop, tablet or smartphone applications.

The XML configuration description language can also describe the video

[8]Nested Context Language - http://ncl.org.br/en

streams (including the orchestration, if any) and points of interest will be used in the final multi-video object. It is also possible to generate different multi-video objects using the same recorded lecture (for instance, by using the orchestration stream or not).

Presentation tool

It is desirable to offer students a platform-independent way to access the captured lectures. We would like to avoid students having to install specific software to playback the lectures. To fulfill this requirement, two possibilities were considered: (i) generating an HTML5+JavaScript multi-video object and (ii) generating a multi-video object based on a synchronization language like NCL or SMIL.

The multi-video object generated from the capture imposed some challenges. In the scenario where we have considered the generation of the object directly in HTML5 + JavaScript, a large development effort to implement the synchronization capabilities was demanded. We also noticed that most obstacles identified in the HTML5-based implementation would be easily overcome with the use of a declarative language specialized in media synchronization. However, there were no solutions to support it that did not demand external plug-ins.

As a result of these needs, we were motivated to propose and develop a multimedia presentation engine based on standard Web technologies. We conducted an implementation based on HTML5 + JavaScript that enables the presentation of multi-video NCL documents, named WebNCL [9] (Melo et al., 2012). Thanks to WebNCL, any device that has an HTML5-compatible browser (PC, Smart TV, Tablet, Smart Phone, etc.) can present NCL documents natively.

The choice to implement support to the NCL language was made because it is a powerful language for media synchronization that has been adopted as the standards for iDTV (ABNT Associação Brasileira de Normas Técnicas, 2007) and IPTV (H.761, 2009; Moreno et al., 2010). A good side effect of this choice was the possibility to reuse the content generated in different platforms. It is important to observe that the generation of the NCL documents demanded solving interesting document engineering problems, as detailed elsewhere (Viel et al., 2013d).

Figure 2.27 illustrates the playback of three NCL learning objects generated by the prototype. The NCL document offers some facilities for students. One of these facilities is the synchronization of the captured audio/video. The multi-

[9]WebNCL is an open-source software, available at http://webncl.org

video object synchronizes the multiple audio/video streams, so students can see what was written in the whiteboard when the lecturer points to the slide presentation. This synchronization is essential to recover the whole audiovisual context of the captured lecture at a given moment. It is also possible to insert non-synchronized complementary media to the multi-video object like, for instance, an image from a textbook.

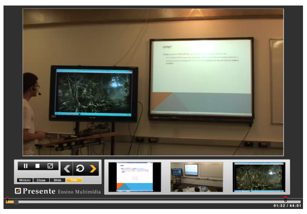
The multi-video object offers a more semantic and easier way to navigate in the captured lecture than timeline navigation. For instance, a student can move forward to the next slide transition or backwards to the previous one. When the lecturer begins to write something in the whiteboard, the student can skip all the writing process and see the final result. In a future implementation, students will also be able to search for a keyword to command navigation (e.g., to go to be next point in which the lecturer sais "for instance"). It is important to observe that a timeline-based navigation is always available: even though timelines are traditionally not available in declarative documents (i.e., NCL and SMIL documents), they are such a usual tool for navigation in video that our users explicitly demanded them to be available.

Similar to in-classroom lecture, wherein the student can pay attention to different spots (the lecturer, whiteboard, slide presentation, the textbook, or another screen), the multi-video object allows the student to choose which video stream he wishes to see and he can even see more than one at the same time.

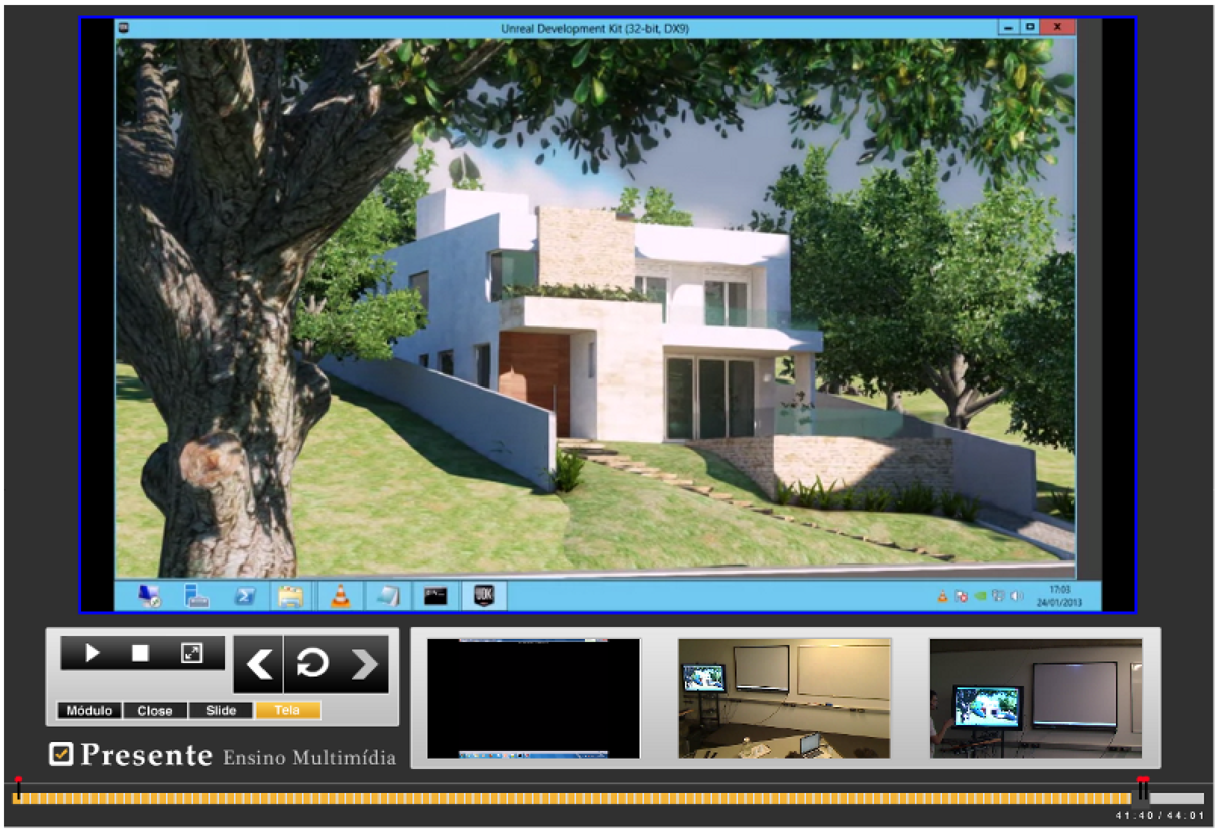
Finally, the student has the facility to make annotations in the multimedia object by means of the watch-and-comment paradigm. For instance, he can mark some part of the lecture as important or irrelevant, or he can delimit a snippet of the lecture which he did not understand for further research or to ask the professor or tutor. He can also make comments on the lecture via audio or text when in-classroom students would make notes with paper and pencil.

2.3.5 Case Studies

We invited seven professors to use the prototype and record their presentations. Four of them recorded a lecture simulation (without students), one professor a conventional lecture, with real students, one recorded a problem solving class and one professor recorded a lecture simulation and a problem solving class. Information about the generated multimedia learning objects are summarized in Table 2.1. The table shows that, while the duration of the sessions and type of resources used varied among the instructors, the size of


(a) Multiple Videos

(b) Full-screen


Unreal Development Kit (32-bit, DX9)
Módulo
Close
Slide
Tela
Presente Ensino Multimídia
41:40 / 44:01

(c) Timeline

Figura 2.27: Multi-video learning objects

Tabela 2.1: Multimedia Learning Objects Statistics

	Dur.	Modules	Resources Used	Size (MB)	LOC
Lecture (i)	2299s	1	Cams, Slide	1	17672
Lecture (ii)	2682s	3	Cams, Slide, WB	1.2	21238
Lecture (iii)	1894s	4	Cams, Slide	0.6	11257
Lecture (iv)	2002s	1	Cams, Slide	0.7	12881
Lecture (v)	3559s	1	Cams, Slide	1.3	23261
Lecture (vi)	2295s	12	Cams, Slide, PC	1.6	29933
Lecture (vii)	5879s	12	Cams, Slide, WB	1.4	25993
Lecture (viii)	4695s	12	Cams, Slide, WB, PC	1.8	33394

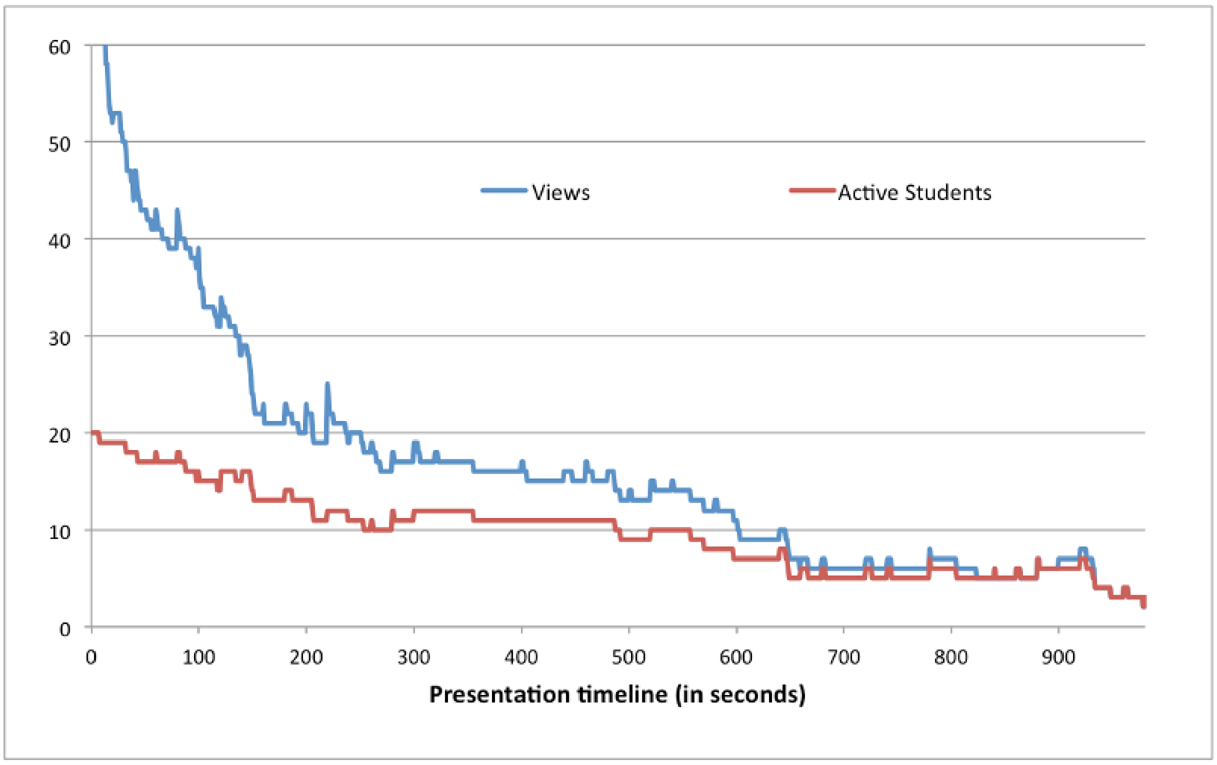


Figura 2.28: Students' Attendance

the resulting interactive multi-video object was always of an order not feasibly achievable without the use of an automatic authoring approach.

The prototype was also used to record a term paper presentation, two M.Sc. qualification exams, one M.Sc. defense, and several demos for workshops.

After a short explanation of how to use the capture tool, all lecturers could perform the recording alone, meeting the proposed self-service approach. Since the room is a real classroom, the lecturers could perform their presentations naturally, despite the cameras. Some of them did not modularize the presentation and record a single long module. We also observed that most lecturers did not use all the resources available in the environment.

After interacting with the multimedia learning objects generated from the capture of their lectures, we asked to the professors to, anonymously, answer a survey. The survey was organized in five parts: the first had questions about the purpose of capture lectures, the second had questions about the instrumented classroom infrastructure, the third had questions about the experience of capturing a lecture using our system, the fourth had questions about the user interface available for controlling the capture process and the fifth had questions about the multimedia learning object interface.

We have also carried out three case studies in real scenarios with traditional modality students of Computer Science and Computer Engineering courses and with distance education students of Information Systems. In each case study we invited a professor to capture an educational presentation. One presentation was for a Computer Organization course, another for a Design and Analysis of Algorithms course, and the third was for a Database course.

The resulting multimedia learning objects were made available for students on the Web via WebNCL and, by using the WebNCL's log API, we logged the interactions performed by the students. We also asked the students to anonymously answer a survey. The survey was organized in three parts: the first had questions about the purpose of capture lectures, the second about their experience in interacting with the multimedia learning object, and the third about its interface.

When asked about the purpose of capture lectures in order to generated multimedia presentations, both students and professors agreed that it is relevant. In addition, among the adjectives available to classify the feeling of recording a lecture by themselves, most professors selected "satisfied", "independent" and "motivated". Among the adjectives available to classify the experience of interacting with the multimedia learning object, most students chose "satisfied" and "independent". Students also pointed out that the multimedia learning object has helped them to understand the subject.

Figure 2.29 summarizes the interactions performed by the students. Note that 60% of the interactions were performed in order to select the main video. When asked about the facilities offered by the multimedia object, most students pointed out that the possibility of choosing which videos to see and the browsing facilities offered by the multimedia presentations are useful. A student declared "I really liked the multimedia lecture, especially the fact that I was able to rewind and listen again to an explanation. The different videos are cool."

When asked about the division in modules, both professors and students pointed out that this feature was relevant. Professors also declared that it

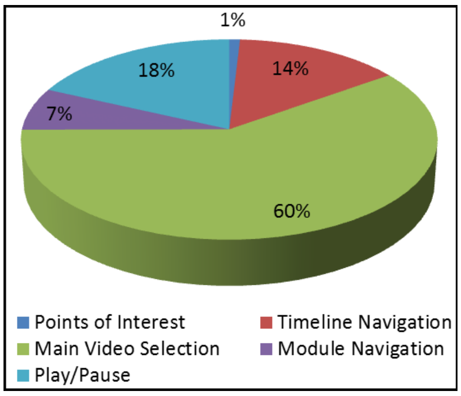


Figura 2.29: Interactions Categories

helped them to organize their presentations. We also observed that students tend to be more active when interacting with a multimedia presentation than when watching a video lecture, since they have more control in presentation reproduction, such as selecting the main video or semantic browsing.

Figure 2.28 summarizes the watching attendance of module 1 of one of the presentations . The horizontal axis is the number of seconds of the module 1 (which is 974 seconds). The blue line represents the number of times the instant was watched by students, and the red line the number of different students watched that instant.

Since a module always starts from 0 seconds, it is natural that the attendance of the first seconds is bigger. The points where the blue line is above the red line mean that the moment was watched more than once by the same students. This graphic can be useful for lecturers to find out which parts of a lecture are more useful or important for the students. For instance, around the second 213 there is a local maximum point, which is the same moment a slide transition occurs in the multi-video object. A more detailed analysis of interactions performed by the students is reported elsewhere (Viel et al., 2013a).

2.3.6 *Final Remarks*

In this paper we present a model and corresponding system for the preservation of multi-device educational presentations. Capturing both the content presented on multiple devices and important context information from different angles allow the generation of an interactive multi-video document that tries to preserve the level of reality of the presentation as much as possible. From the multi-video it is possible to reconstitute and explore the lecture in dimensions not achievable in the classroom.

The construction of an infrastructure to capture different views of an educational presentation in order to generate an equivalent multi-video object and to present it allowed testing concepts with instructor and students so as to identify improvements.

Our plans for future works include designing analysis tools to facilitate the extraction of useful information from the large amount of data generated by students navigating over the captured presentation, as demanded by specialists from the educational domain (Ronchetti, 2013). Moreover, we plan to offer alternative capture infrastructures using low end and mobile equipment such as smartphones, which is a growing demand (Ronchetti, 2013).

Acknowledgments

We thank the Brazilian agencies CAPES, CNPq, FAPESP and FINEP for financial support to our research. We thank FAPESP for financial support for the presentation of this paper at Webmedia 2013.

2.4 Go beyond boundaries of iTV applications

The development of multimedia applications that require the manipulation and the synchronization of multiple media and the handling of different types of user interactions usually requires specialized knowledge in imperative languages. Declarative languages have been proposed in order to make this task easier, especially when applications are restricted to certain classes, as it is the case of Interactive TV applications in which user interactions are restricted to a few simple models. However, those simple models may be too simple when documents are reused in other platforms: for instance, when watching a video most web users expect an interactive timeline to be available — which is not the case in interactive TV videos. This paper presents a component-based approach to the enrichment of declarative languages for multimedia so that desirable user-media interactions are made possible at the same time that the original ease of authoring is maintained. We detail the components and present a corresponding proof-of-concept prototype. We also discuss design decisions associated with the development of the components, which should be useful in further extensions.

2.4.1 Introduction

The design of multimedia presentations or applications — for the web, desktop, mobile devices or interactive TV (iTV) — is an interdisciplinary activity which demands professionals from different fields such as communication, design, marketing and art. The development of multimedia applications, however, requires the work of professionals with good programming skills, usually in imperative languages.

Therefore, good communication and understanding among computer professionals and others involved in the creation of multimedia presentations is essential to produce applications faithful to the wishes of their authors.

The possibility of the same professionals that design multimedia presentations be also able to undertake part of its implementation, even in draft or prototype versions, can be productive, for instance to improve the communication among programmers in terms of improvements. Moreover, if the final implementation can be carried out by the creators of the presentation, economic gains can also be obtained. Furthermore, the easy authoring of multimedia presentations may result in an increase in the amount of presentations created.

Declarative languages for the specification of multimedia applications such as SMIL (Bulterman & Rutledge, 2008), for web applications, and NCL[10] (ABNT Associação Brasileira de Normas Técnicas, 2007; Soares et al., 2010a), for iTV applications, were proposed to facilitate the authoring of multimedia applications — when compared with authoring with imperative languages. However, given that these languages provide good options for synchronizing media but fewer options for user interaction, a limited class of multimedia applications is supported — at least as far as the ease of authoring is concerned. It is not trivial, for example, to create declarative applications that allow media annotations or, in the case of NCL, use a time-slider to allow usert to interact with continuous media — even though the latter is an operation which many Web users are used to.

The opportunity for authoring interactive multimedia documents automatically by combining information captured from live experiences has been extensively exploited — in particular considering the lecture environment. In this case, information captured in a classroom (video, audio, slides, electronic ink, etc.) is used in the composition of interactive web-based multimedia documents (Abowd et al., 1996; Hürst, 2003; Lanir et al., 2008; Lienhard & Lauer, 2002).

Several platforms exist that allow, more or less automatically, combining captured information into web-based videos so that the result is a multimedia application usually encoded in HTML5 or Flash. Examples in the educational domain include Coursera[11], EDx[12] and EyA (Canessa et al., 2013). However, real world demands require that the generated multimedia documents be made available not only for the web and mobile platforms, in which HTML5 would suffice, but also for the iTV environments. This lead us to investigate how to automatically generate interactive multimedia applications encoded in the NCL, since NCL interactive multimedia documents can be used both in compliant native iTV environments and in the web (using the WebNCL engine (Melo et al., 2012)).

A challenging requirement, given the many platforms available, is the production of multimedia documents that offer interaction facilities expected by

[10]Declarative language adopted by the ISDB-Tb (International System for Digital Broadcast, Terrestrial, Brazilian version) and by the ITU-T for IPTV.

[11]www.coursera.org

[12]www.edx.org

users of all platforms — web users, for instance, are likely to be frustrated when facing a video without a time slider. Such requirement challenged us to build components that enable the necessary interactions at the same time that maintain the original declarative authoring approach. In this paper we present a component-based approach to the enrichment of declarative languages for multimedia so that desirable user-media interactions are made possible at the same time that the original ease of authoring is maintained. We also present a corresponding proof-of-concept prototype and discuss design decisions should be useful in future extensions.

This paper is organized as follows: in the next section we discuss related work, especially research related to simplicity in authoring of multimedia applications; in Section 2.4.3 we discuss both some interesting features for multimedia applications which have proven to be difficult to be implemented with declarative languages, and how these difficulties can be overcome; in Section 2.4.4 we present components that generate NCL + Lua code that implements these features; in Section 2.4.5 we illustrate the use of the components by means of the generation of a highly-interactive NCL Document. In Section 2.4.7 we present our final remarks.

2.4.2 Related Work

Declarative languages for authoring hypermedia documents present some limitations that can usually be overcome with aid of imperative script languages. Efforts for reducing the dependence of script languages are reported by King et al. (King et al., 2004): the authors add to SMIL some capabilities of functional languages, such as expression evaluation and value-based reactive events. Using a similar approach, Soares et al. (Soares et al., 2010b) add to NCL the capability for store and retrieve values in variables. The authors also extended the NCL's causality relations to support variable-based conditions. Both works reduce the necessity for imperative code, but they achieve this by incorporating in declarative languages low-level features of non-declarative languages.

The development of modern multimedia application for the Web can be carried out using HTML5, CSS and JavaScript. Although the content, structure and style of applications can be defined using declarative languages, time (and complex spatial) synchronization must be done with aid of JavaScript. The work of Cazenave et al. (Cazenave et al., 2011) combine the flexibility of HTML5 and CSS with the SMIL Timing and Synchronization module in order to avoid using JavaScript code to perform synchronization in Web applications. The work represents an enhancement in the development of HTML-based applications, but does not offer an easier way for developing multimedia applications than the one offered by SMIL.

The work of Azevedo and Soares (Azevedo & Soares, 2012) extends NCL in order to add support for 3D scenes described in the declarative language X3D. The authors propose the use of X3D documents as NCL media nodes, mapping NCL anchors into equivalent elements of X3D documents. They also extended the NCL's causality relations to add some conditions related to 3D scenes, such as collisions. Although the combination of X3D and NCL allows the creation of complex and rich multimedia applications, composed by image, videos and 3D scene, it does not enhance the ease of authoring of multimedia applications.

One issue present in some declarative languages, especially in NCL, is the verbosity and recurrence of constructions. This lead to template-based generators such as XTemplate (Santos & Muchaluat-Saade, 2012) and Template Authoring Language (TAL) (Soares Neto et al., 2012). These works specify languages for writing templates that can be imported into documents for further generation of multimedia applications in languages such as NCL or HTML. Although these works enhance the expressivity of the declarative languages by offering templates as high-level construction blocks, they lack in providing high-level interaction models such as timeline navigation.

Another common approach to ease the development of multimedia applications is the use of graphical tools such as SMIL builder (Bouyakoub & Belkhir, 2011), EDITEC (Damasceno et al., 2011) or FIND (Rodrigues et al., 2012). SMIL builder creates and validates SMIL documents using a visual representation of the SMIL timeline expressed in a specialized Petri Net. EDITEC allows the authoring of NCL documents using predefined constructions which are written in the XTemplate language. FIND proposes that authors add complementary content to video by means of annotations, and was inspired in the watch-and-comment approach (Teixeira et al., 2012) proposed to allow viewers to add annotations at the time of playback. FIND offers clues about moments in which annotations can be added, such as silence moments. Overall, these tools speed up the development of multimdia applications, but they may lead to a limited use of the expressivity of the declarative language.

Meixner and Kosch (Meixner & Kosch, 2012) propose a specific XML language to represent interactive non-linear video. The language maps the video reproduction into a flowchart at the same time that supports the synchronization of additional media with parts of the video. The SIVA Suite (Meixner et al., 2010) is the authoring tool and player for non-linear video resulting provided by the authors, who also provide a player for android smartphones (Meixner et al., 2011).

Tondorf et al. (Tonndorf et al., 2012) propose a system for authoring and delivering multimedia problem-solving content. The authors noted the lack of some features in declarative language, such as indexed search for content, and implemented a specialized player with such features which is associated with a XML-based declarative language.

The work of Meixner and Kosch (Meixner & Kosch, 2012) and Tondorf et al. (Tonndorf et al., 2012) are examples of results that ease the development of declarative language-based complex multimedia applications using for specific classes of applications.

The use of layered frameworks such as Django[13] or Grails[14] for speeding up the development of web applications is a common practice. Some researchers applied the same concept for the development of multimedia applications (e.g. (Boll, 2003) and (Lin et al., 2012)). However, while the first is more concerned with content adaptation among different devices than with easing the authoring task, the latter is based on an oriented-object approach rather than declarative languages.

2.4.3 Challenges in the Development of Declarative Presentations

For interactive multimedia presentations specified in declarative languages such as NCL and SMIL, corresponding presentation machines machines exist which run on top of standard web technologies (Gaggi & Danese, 2011; Melo et al., 2012). As a result, the presentations can be deployed in any platform with a compatible browser including tablets, smartphone and Smart TVs.

However, declarative languages for hypermedia synchronization are usually designed to meet some common requirements and constrains of the platforms they are originally intended for. For instance, NCL 3.0 is design for iDTV and IPTV systems and the limitations imposed by these platforms may impact in the development of applications that might be reused in other platforms. Moreover, some applications that stand beyond the boundaries of the iDTV systems may be arduous to be implemented in NCL. One example of limitation is that in NCL the user's interaction is based on TV remote control, which is not appropriate for highly interactive applications (Pedrosa et al., 2011).

In our experience with declarative languages for media synchronization (Pi-

[13]https://www.djangoproject.com/
[14]http://grails.org/

mentel et al., 2010; Teixeira et al., 2010b; Melo et al., 2012), we identified some features common to multimedia presentations that are not trivial to be implemented in NCL because of its limitations. Other features are easy to be implemented, but demand a great effort in defining language entities (Santos & Muchaluat-Saade, 2012).

By exploring NCL's expressivity, its integration with imperative Lua scripts and a bit of creativity, we were able to implement these non-trivial features. However, the complexity of the implementations is far from the expected for a declarative language. Next we detail the features we identified as hard or verbose to be implemented in NCL, along with the corresponding implementation solutions we offer.

Time-related Navigation

This functionality is common in multimedia presentations generated from the capture of human activities, such as recorded lectures or video conferences ((Vega-Oliveros et al., 2010), (Dickson et al., 2012)). The presentations usually contain events such as *slide transition* or a *keyword spoken* and the users can navigate to the moment in which these events occur. For instance, one may navigate until the beginning of the previous slide transition or just after the next keyword.

The main difficulty in implementing this functionality using a declarative language lies in the necessity of keep tracking of which are the next and previous events for the current time in the presentation. This requires a complex control in which a variable registers what the current event is as the time advances during playback or the user interacts with the presentation. Since we are working with declarative languages, it is also necessary to consider all the possible values that the tracking variable can assume in order to perform the correct navigation for the next or previous event.

Time-related navigation can be implemented in NCL by mapping each of those events to temporal anchors in a media node. If it is necessary to synchronize different content by these events — such as different video camera streams that frame a classroom — all the related media nodes should also have temporal anchors.

In order to keep track of the events that correspond to the current time, the NCL variable capabilities can be used (Soares et al., 2010b). A virtual media node is (declared and) used to store the variable which tracks the current state. Moreover, a link for each temporal anchor is also created to set the value of the tracking variable when the anchor begins (e.g. the associated event begins).

The actions that users can perform, such as *advance to the next slide transition*, can then be mapped into a button in the application's interface. When the user clicks or taps on this button, a NCL link, similar to the one presented in Listing 2.1, is triggered for each possible value that the tracking variable can assume. Upon checking the value of the tracking variable, only one of the links has its conditions met: this link then starts the media in the corresponding temporal anchor.

Listing 2.1: A "next slide" link

```
<link xconnector="...">
    <bind role="onSelection"
            component="btnNext"/>
    <bind role="propertyType"
            component="event_index"
            interface="last_slide">
        <bindParam name="testValue"
                value="slide_1"/>
    </bind>
    <bind role="abort" component="video"/>
    <bind role="start" component="video"
            interface="slide_2"/>
</link>
```

Multiple Navigation Indexes

Applications may allow users to navigate through different types of events. A user may opt at one time to navigate using type (or set) of events (e.g. *slide transition*), in another time a different set of events may be used (e.g. *annotations*), an in another time the union or intersection of the sets may be possible. In this work we refer to each set of events as a *Navigation Index*.

The control of navigation using different indexes is a complex task to be specified using a declarative language. Many possible situations should be considered, including the selection of the index an user may choose for navigation. This can be solved by mimicking a radio button. An image can represent the index for navigation. When clicking on the image, a variable is set to store the current active index. Moreover, the navigation links, such as the depicted in Listing 2.1, also need to check the active index variable.

Timeline Navigation

Timelines are common in applications presenting continuous media, as it is the case with YouTube and QuickTime. As a result, users expect to have a timeline in applications which manipulate video. In fact, when more than one video is availabe, users would also expect that a timeline synchronize the multiple continuous media streams as a single synchronous stream. For instance, it would be important to be able to control, with the same timeline, two video streams taken from different cameras that recorded the same scene but from different views.

The implementation of a timeline in NCL, a language that does not have native support for it, is complicated since: (i) some presentation machines do not allow to set the current time of a continuous media directly; (ii) representing the timeline interface with a composition of media would require a complex control; (iii) users are used to interact with the timeline using a mouse or touchscreen in Web and mobile environment. Such facilities are not common in interactive TV environments.

In order to solve (i), we opted to implement a discrete timeline, the same approach used by Vega-Oliveros, Martins and Pimentel (Vega-Oliveros et al., 2010). Instead of allowing access to each instant of the continuous media, the media is split into several short segments (e.g. 1, 5, 10 or 20 seconds). A temporal anchor for each of these segments is added to the media node.

To solve (ii) we used a Lua script to draw the timeline (NCLua Canvas API). The Lua script must know the current time of the media, which can be implemented by using the events.time() function to count time. When the playback of continuous media starts, the Lua script starts to count time. If the playback of the continuous media is paused, the Lua script should stop counting time until the playback is resumed. The Lua script also has a property anchor named *current_time* which must be set when navigating in the media, in order to update the time information kept by the script. For instance, if the user navigates to the second 85, the *current_time* property is set to 85: the Lua scripts then starts counting time from the second 85.

Regarding (iii), we observed that the basic interaction with the timeline can be implemented using buttons such as *next* and *previous*, similar to the used for Time-related Navigation. Each segment of the discrete timeline can be handled as an event from the timeline index. The problem of this approach is that every time the next button is pressed, a number of links equal to the number of segments of the timeline would be triggered. For instance, in a presentation lasting 1 hour and split in 1-second segments, each time the *next* button is pressed, 3600 links would be triggered and checked for the value of the tracking variable so that one single link would be activated.

We avoided checking all possible (thousands of) conditions by using the Lua script as a proxy. The Lua node has “command” anchors for the next and previous buttons and one virtual temporal anchor for each segment. There is a link for each button that performs a start action over the related “command” anchor. Given that the Lua script knows the current time, it can easily compute which is the next/previous segment and, by using the NCLua events API, trigger a start event in its anchor related to that segment. Back to the NCL, there is a link which has the beginning of the Lua virtual temporal anchor as condition to be fired. It does the job of seeking the continuous media (Listing 2.2).

Listing 2.2: Timeline Link

```
<link xconnector="...">
    <bind role="onBegin"
            component="lua_timeline"
            interface="t_segment_0"/>
    <bind role="stop"
            component="video"/>
    <bind role="start"
            component="video"
            interface="v_segment_0"/>
</link>
```

Although this approach allows the user interaction with the timeline, it still does not solve (iii). Since we aim to make the multimedia presentations available in platforms such as the Web or mobile devices, the approach of using buttons to interact with the timeline is unfamiliar to users who are habituated with mouses or touchscreens. WebNCL (Melo et al., 2012) maps mouse and touchscreen interactions into common remote control events. When the mouse is over a media, that media receives the focus. If the left or right button is pressed, an OK button event is triggered. In touchscreen devices, tapping in a media without the focus is equivalent to change the focus to that media. A second tap (or double tap) on that media triggers an event equivalent to pressing the OK button.

In order to offer Web-like mouse interaction with the timeline, we added an invisible virtual media node above each segment of the timeline interface. When the user clicks on these virtual media nodes, a link performs a start action over the related Lua node’s virtual temporal anchor. This action triggers the link of Listing 2.2 in charge of positioning the media in the desired segment.

Plotting indexes in the Timeline

It is possible to plot points on the timeline corresponding to events from a time index. Users may navigate to these events by clicking over the related points. In order to implement this feature we added a media for each event over the correct position on the timeline interface. It is also possible to show visual feedback about an event when the user moves the mouse over its media node on the timeline. For instance, in slide transition this visual feedback could be the new slide.

User Query

As an example of user query, an interactive video application that alerts the user about a scene which may contain inappropriate content offers user the option of watching or skipping the scene. As another example, at the end of a music video clip the user may choose which video clip is next.

In fact, asking the user to decide which media to play is a common requirement in many multimedia applications. It is not a complex functionality to implement, but it requires many different links to be implemented: (i) one to show the query, (ii) one to hide the query after a certain time, (iii) one for each possible alternative available.

Spatial Composition Changes

Spatial Composition Change is a common requirement of multi-video applications. For instance, in a camera system several streams may be presented in small windows which share the same screen. It may be possible to maximize one of the videos in order to verify something suspicious, choosing one of the video streams to be presented in a large (main) window while the other videos remain in small windows. Then, the user may select one video of the small windows to be swapped with the video presented in the main window. In a scenario in the educational domain, a multimedia presentation generated from a lecture may be composed of several video streams: one which captured the slides, another which captured the whiteboard or the instructor's computer, and others corresponding the different views of the instructor. The students may choose to focus their attention into one specific stream presented in one large (main) window while the other streams are presented in small windows – and may swap the video presented in the main window with any other. The change in spatial composition is also a requirement for reactive applications that must change their layout in response to user interactions, such as when the orientation of the device changes or the display is resized.

The problem on implementing this functionality lies in keeping track the “window” in which each media is presented. Moreover, it may be necessary to declare links for the many possible combinations of media and positions in order to react properly to user’s interactions. We implemented the changes in spatial composition using a finite-state machine (FSM). Each state in the FSM corresponds to a spatial composition the medias can assume. The user’s interactions (or internal events) are the transitions in the FSM.

Figure 2.30 illustrates a FSM for a composition of 3 videos. There are 3 states, each one with one of the videos as the main (larger) video. When a user clicks on a video presented in a small screen, the corresponding a state transition is triggered and that stream is shown in the main video. We implemented this in NCL by using a variable to store the current state. When a interaction corresponding to a transition in the FSM is performed, a link for each transition related to that event is triggered. These links check the current state variable, if the current state has a transition that is triggered by that interaction, the corresponding link performs its actions, changing the video bounds and setting the current state variable to meet the new state configuration.

Note that in the configuration illustrated in Figure 2.30, if the current state is the one with the video2 as the main video and the user clicks over the video2 two links are triggered. However, since the current state does not have transitions related to this event, none of the links perform their actions.

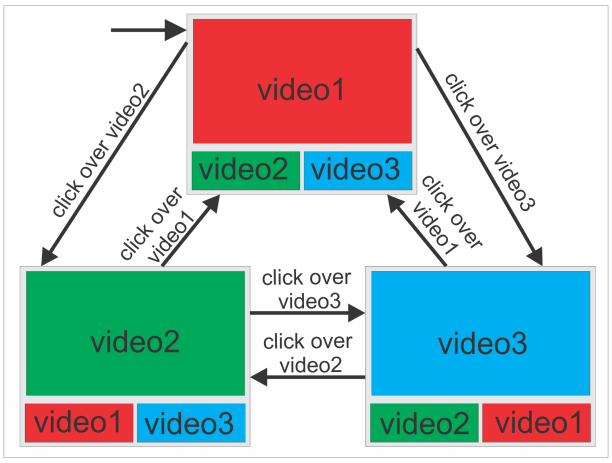


Figura 2.30: FSM for Spatial Composition

Annotation

User annotation is an important requirement for multimedia applications. In our approach annotations are based on a timeline: the user marks some media segment as important, or adds some comment at a given instant of the media, etc.

One main concern about annotations in multimedia presentations is how to store and retrieve the annotations created by end users. A possible approach is to modify the multimedia presentation on-the-fly, adding new media and links to represent the annotations (Cattelan et al., 2008; Jansen et al., 2010; Pimentel et al., 2010). This could be implemented in NCL by using the Ginga-NCL live editing API (Costa et al., 2006) and dumping the modified NCL document into a new XML file. A version control mechanism for these annotated NCL files would also be required.

We propose a different approach: a Web Service is used to store and retrieve the annotations. A Lua script accesses the Web Service and, using the data retrieved, draws the annotations over the timeline. In order to add a new annotation, the Lua script must know the current time of the media in which the annotation will be performed. This can be obtained from the Lua script that controls the Timeline Navigation.

The interface between the Lua script and the NCL document is implemented using some virtual anchors, as depicted in Listing 2.3.

Listing 2.3: Anchor Listing

```
<media id="timeline" src="timeline.lua">
    <!-- ... -->
    <area id="important" />
    <area id="doubt" />
    <area id="comment" />

    <area id="next_annotation" />
    <area id="previous_annotation" />

    <area id="annotation1" />
    <area id="annotation2" />
    <!-- ... -->
    <area id="annotation50" />

    <property name="annotation1_bounds"
            value="None" />
```

```
17      <property name="annotation2_bounds"
18              value="None" />
19      <!-- ... -->
20      <property name="annotation50_bounds"
21              value="None" />
22  </media>
```

We use images as buttons to allow users to create annotations while interacting with the presentation, as suggested by the watch-and-comment paradigm (Cattelan et al., 2008). For instance, a button may be used to express that the segment is important and other to add that a textual comment. When these buttons are clicked, a link performs a start action over the correspondent virtual anchor of the Lua object (Listing 2.3, lines 3-5).

The Lua script uses the information about the current time to save the annotation in its internal structures (using the discrete segments of the timeline). It also saves the annotation in the web server. The annotations stored in the Lua internal structures are then drawn in the timeline.

It is possible to navigate among the annotations using the next and previous buttons. This is implemented using the Lua script as a proxy (similar to the timeline navigation). The buttons trigger links that perform start actions over the Lua object command anchors (Listing 2.3, lines 7-8) and the Lua script performs a start action over the appropriate Lua virtual temporal anchor.

In order to allow mouse navigation in the annotations, it is necessary to add virtual invisible nodes. Because at the time of the creation of the NCL document we do not know which annotations a continuous media will receive, it is necessary to add some media in advance for this purpose. However, we also do not know which segments will be annotated, which means that the positioning the invisible media in the timeline cannot be made when the NCL document is created.

The positioning of annotation media nodes are computed by a Lua script. The positions are sent to the NCL document when the Lua script performs a set action over one of the annotation bounds anchor properties (Listing 2.3, lines 15-21). The value the Lua script sets in these anchors are the positions in which the associated virtual nodes must be placed. Back to the NCL, a link that is triggered when a value is set to the annotation bounds anchor sets the bounds property of the associated virtual node. When the user clicks on these media nodes, a link performs a start action over corresponding virtual temporal anchors of the Lua object (Listing 2.3, lines 10-13). Finally, the Lua object performs a start action over the proper timeline segment anchor.

When a presentation begins, the Lua script retrieves all the previous annotations made by the user from the Web Service and set their positions in the timeline.

2.4.4 Components

Despite the fact that the features discussed in the previous section can be found in many classes of multimedia presentations, the corresponding implementations in NCL are non-trivial. In order to ease the authoring, we designed and implemented components in Python that generate NCL + Lua code for these features. Figure 2.31 depicts a simplified UML class diagram of the components.

The components are grouped in a layered architecture inspired in the Model-View-Controller (MVC) design pattern. The bottom layer is the **Multimedia Layer**, wherein the media nodes used in the multimedia application are declared. In the case of a NCL document, they are the media and context nodes. The middle layer is the **Controller layer**, which holds the components that define the types of navigation and synchronization (both temporal and spatial) it is possible to perform with the media nodes. The top layer is the **View Layer**, which contains the components responsible for allowing users to interact with the applications.

In this layered architecture, which we refer to as *Multimedia-View-Controller* (MMVC) architecture, media nodes from the multimedia layer are manipulated by the components of the Controller Layer. The user, via components of the View Layer, may interact with the media. The view layer sends messages to the controller layer which performs the actions to manipulate the media.

Note that in Figure 2.31 the components **SpatialSlotManager** and **UserQuery** are above the line that separates the View and the Controller Layer. This is because the synchronizations defined in these components are intrinsically related to user interactions. For instance, the **SpatialSlotManager** component allows some media nodes to exchange the spatial position among themselves after a user selection. It would be possible to split it in two components the parts responsible for the interaction and responsible to define the behavior, each one in one layer, but these two components would, almost always, be used together.

The components **TimeIndex**, **AnnotataionBase** and **Timeline** implement the interface **NavigableIndex**. These components allow navigation operations such as *"go to the next point"* or *"move forward to the previous point"*.

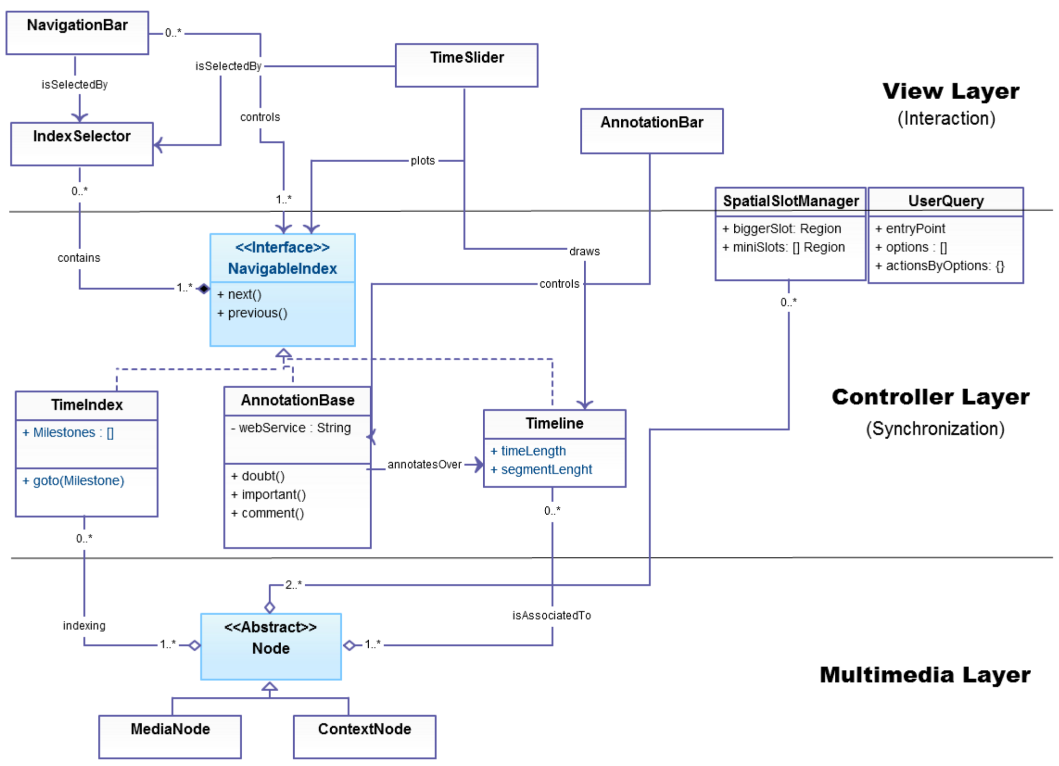


Figura 2.31: Class Diagram of the components

The **TimeIndex** allows the creation of an array of time milestones for a media or a group of medias. For instance, Listing 2.4 depicts the creation of a time index for a media node. It is possible to bind media nodes via their unique ids in the NCL document (Listing 2.4, line 2) or via an instance of Media class (Listing 2.4, line 3-4). The milestones can be passed as a list of times in seconds or individually. A **NavigationBar** is a visual component that presents buttons like next and previous and allows user navigation by any type of **NavigableIndex** instance. Since it is a visual component, it is necessary to define the position in which the buttons must to be displayed. This is done by passing a NCL region to the component (Listing 2.4, line 7)

By combining the **TimeIndex** and the **NavigationBar** components, it is possible to implement the Time-related Navigation (Section 2.4.3).

Listing 2.4: TimeIndex

```
1 index = TimeIndex()
2 index.bind_media('video0')
3 index.bind_media(Media(id='video1',
4                src='video1.mp4'))
5 index.add_milestones([5, 48, 58.5])
6 index.add_milestone(78)
7 bar = NavigationBar('rNaviBar')
```

```
8 bar.add_index(index)
```

The **Timeline** component bounds a media or group of medias to a single timeline. It also enables the user to navigate in the media using a timeline. In order to do this, is necessary to bound it to a **TimeSlider** component. The **TimeSlider** is a visual component that draws a timeline and can be used to access any instant of a media or group of media. Together, these components implement the Timeline Navigation (Section 2.4.3).

Listing 2.5 depicts the instantiation of a **TimeLine** for a group of media nodes and show how to bind it to a **TimeSlider** visual component. The duration of the **TimeLine** and the length of the each discrete segment (in seconds) are informed in the constructor (Listing 2.5, line 1). It is possible to bind a media node and to define the time instant in which the timeline begins (for instance, the second 0 of the timeline corresponding to the second 80 from the media *video1* (Listing 2.4, line 3). It is necessary to define the NCL region of the **TimeSlider** component, as well the **TimeLine** instance it will draw and the means by which users can interact with the component (Listing 2.5, lines 4-6).

Listing 2.5: Timeline

```
1 my_line = Timeline(duration=600, segment=1)
2 my_line.bind_media('video0')
3 my_line.bind_media('video1', begin=80)
4 slider = TimeSlider(mouse_interaction=True,
5             timeline=my_line,
6             region='rTimeline')
```

Listing 2.6 illustrates the utilization of the components **AnnotationBase** and **AnnotatonBar**. The **AnnotationBase** allows to create and retrieve annotations made over a **TimeLine**. If, instead of a **TimeLine**, a continuous media is passed in the **AnnotationBase** constructor, a **TimeLine** will be automatically created and bounded with the media (Listing 2.6, lines 2). It is also necessary to pass the URL of the Web Service that will store the annotations (Listing 2.6, lines 3). Since **AnnotationBar** is a visual component, it is necessary to inform the NCL region in which it will be displayed (Listing 2.6, lines 4). These components implement the annotation functionality (Section 2.4.3).

Listing 2.6: Annotation

```
1 note_base = AnnotationBase(
2         timeline='video0',
3         url='localhost:8080/service')
```

```
note_bar = AnnotationBar(region='rBar',
        base=note_base)
```

Both the **NavigationBar** and the **TimeSlider** can support more than one **NavigableIndex** instance. The **IndexSelector** component is necessary to allow the user chooses which of the possible index he or she wishes to use for navigation.

Listing 2.7 illustrates the use of a **SpatialSlotManager** component. This component displays a “maximized” media and a group of “minimized” in defined spatial slots. It is possible to maximize any of the minimized medias by clicking on them. The slots (both the bigger and smalls) are defined by a tuple with the slot spatial position (represented by a NCL region) and the media initially bounded to that slot. This component implements common cases of Spatial Composition Changes (Section 2.4.3).

The **UserQuery** can be used to take actions based on options queried to the user (Section 2.4.3). It is necessary to define an anchor that, when activated, will present the component to the user. The options are defined as a list of text or images. After the user choice, among one of the options, the corresponding action is performed.

Listing 2.7: Annotation

```
ssm = SpatialSlotManager()
ssm.bigger_slot = ('rMainSlot', 'video0')
ssm.add_mini_slot( ('rslot1', 'video1') )
ssm.add_mini_slot( ('rslot2', 'video12) )
```

2.4.5 Validating the Components

We have used the components described in the previous section to build an application that generates NCL documents from captured lecture-style presentations. The result of the capture is a multi-video multimedia containing different views of the lecture, with video streams generated from capturing the slide presentation, the computers used by the instructor, various views of the classroom, etc. There is also an XML file which holds metadata from the captured lecture, such as the time instants in which slide transitions occur, or the lecturer used her computer or the (electronic or traditional) whiteboards. The orchestration of capture process and the metadata extraction is introduced (Viel et al., 2013b) and detailed (Viel et al., 2013c)elsewhere.

Figure 2.32 shows an example of an NCL document generated from the captured presentations. The NCL documents offer facilities that includes the synchronization of the captured video streams. The document in this example has two video streams captured from cameras (Figure 2.32(1) and Figure 2.32(3)), one stream capturing the slide presentation (Figure 2.32(2)) and the instructor's computer screen (Figure 2.32(4)). They all have the same length and their synchronization is handled by a **Timeline** component.

Users may interact with the presentation and access any segment of the synchronized video streams via a **TimeSlider** component (Figure 2.32(6)). There are also temporal indexes implemented by instances of **TimeIndex** component. Indexes are provided for slide transition, lecturer's close and computer interactions. User may select an index for navigation using the **IndexSelector** component (Figure 2.32(7)), and the index-based navigation uses the **NavigationBar** component (Figure 2.32(5)).

Users can also annotate the document via the **AnnotationBar** component (Figure 2.32(8)). The annotations are stored in the **AnnotationBase** which saves the information in a Web-Service.

Similar to co-located lectures, wherein students may focus their attention to different contents (the lecturer, whiteboard, slide presentation, the textbook, etc.), the NCL document allows users to choose which video stream to see in detail in detail. One option is to show the video in the main window using the **SpatialSlotManager** component (Figure 2.32(1-4)); the other option is via the full-screen mode.

The **TimeSlider** component can be used to show annotations and marks or milestones (such as slides transitions) to the timeline interface. If none of the index are selected in the **IndexSelector** component, the timeline will bear no marks as in Figure 2.33(a). When a **TimeIndex** is selected, marks representing the milestones will be added to the timeline (Figure 2.33(b)). When an annotation index is selected, the annotations will be displayed in the timeline (Figure 2.33(c)).

2.4.6 Analysing the NCL Document

To illustrate the complexity of the NCL documents generated with the aid of the components, Table 2.2 summarizes the size the document generated automatically from a captured lecture lasting 1 hour and 18 minutes.

The NCL document uses 48 video clips: four of them can be played back simultaneously. Besides the videos, the document include 1197 other media items, mostly images. There are also 1898 links responsible for controlling all navigation alternatives (including timeline and index-based).

The NCL document itself has about 3.6 MB of size, not considering the

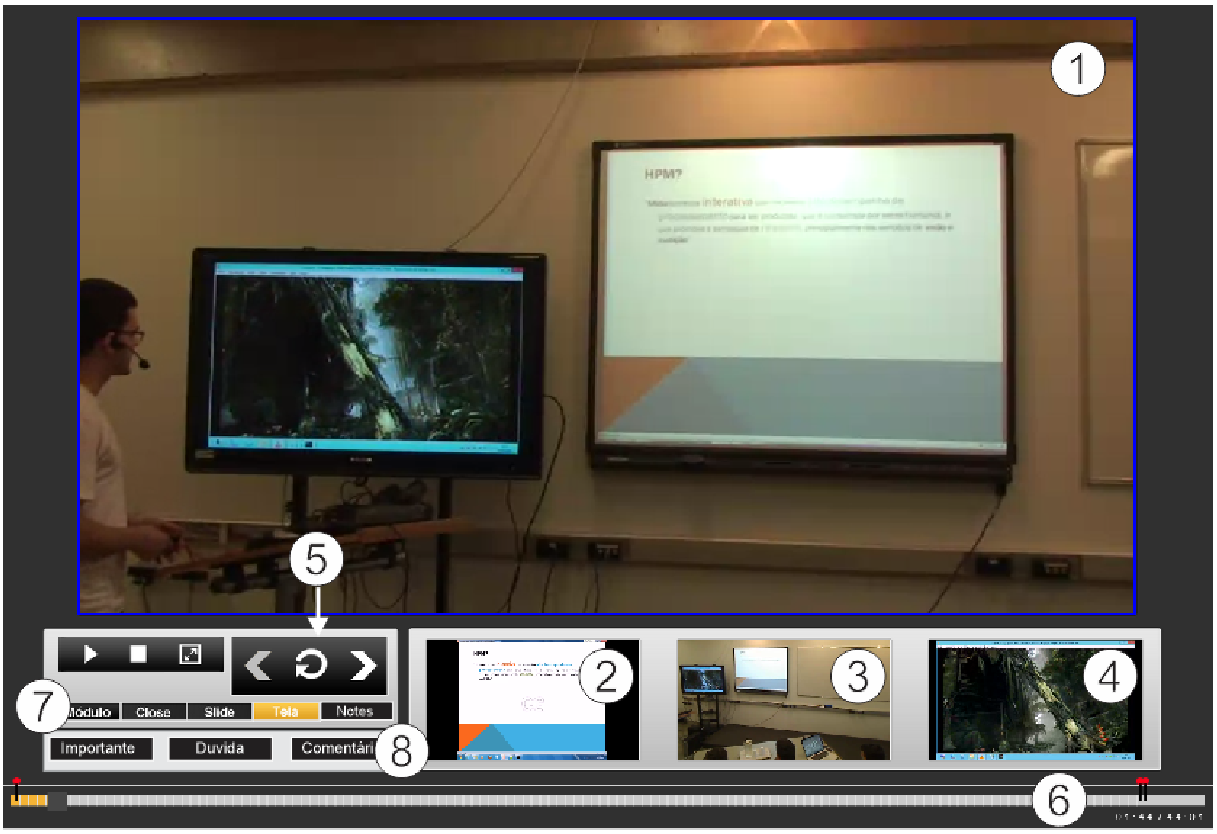
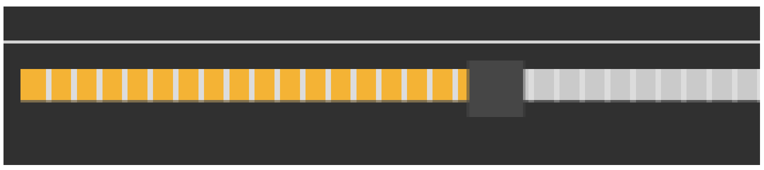
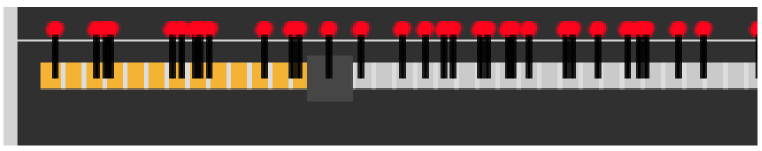


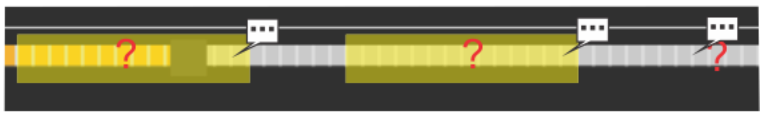
Figura 2.32: Multi-video multimedia object captured from a lecture



(a) Empty



(b) Indexed



(c) Annotated

Figura 2.33: Timeline

videos, images and other media items. The total size of the multimedia document, considering all media, is about 1GB [15]. All the media, anchors, links and the static elements of the NCL (regions, descriptors) add up to a total of 65148 lines of code.

We use the WebNCL presentation machine to render the interactive NCL Documents generated from captured lecture. Although the document sizes may suggest problems in its execution, the WebNCL has been able to reproduce the NCL Document smoothly.

Some improvements in the components may reduce the size of the generated NCL documents. The current version of the components generate a human readable code for debugging proposes, but renaming the NCL entities with concise names and removing white spaces can significantly reduce the document size. Other optimizations in the NCL code structure can also help reducing the document size.

2.4.7 Final Remarks

Declarative languages for media synchronization were conceived to ease the development of multimedia presentations. However, declarative languages for hypermedia synchronization are usually designed to meet common requirements and constrains of the platforms they are originally intended for. The reuse of applications developed in declarative languages in other platforms may be negatively impacted by these limitations.

In this work we elected NCL, a declarative language designed for DTV and IPTV systems, to develop applications that stand beyond the boundaries of such platforms. As result, we identified some features, common in general purpose multimedia applications, that are non-trivial to be implemented

[15]Each video has a version coded in H264 AVC and other coded in VP8, with resolution of 854x480 pixels and 24 frames per second. Note that some of the video streams, such as the on corresponding to the slide presentation, have many static segments which lead to small video size when codecs that remove temporal redundancy such as the H264 and VP8 are used.

Tabela 2.2: NCL Document Numbers

Total Duration	78 minutes
NCL Document Size	3.6 MB
Lines of Code	65148
Links	1898
Media Node	1245
Videos	48

in NCL. Even though, we keep the choice to take advantage of several interesting features of such declarative language for multimedia synchronization. Although theoretically possible, implementing such features using only standard NCL resources would imply a complexity far from the expected for a declarative language.

In order to provide these features, keeping the ease of development, we implemented components organized in a Multimedia-View-Controller architecture that generate NCL + Lua code which implements the non-trivial functionality. We illustrate the use of the components generating a complex and highly-interactive multimedia presentation from a presentation generated with information captured from lectures.

Although the components presented here are not a comprehensive listing capable of implement many classes of multimedia applications, they are construction blocks that can be used in many common applications.

As a future work, we plan to use the components presented in this work as the base for a layered framework for authoring complex and platform-independent multimedia applications. It is important to note that the solutions proposed here, although implemented for NCL, are sufficiently generic to be exploited in similar situations found in other declarative languages for multimedia applications.

2.5 Considerações finais

O objetivo desta tese é apresentar indícios de que aplicações multimídia interativas e potencialmente atrativas podem ser construídas com a incorporação automática ou semiautomática, a uma mídia principal, de informações de contexto representadas em diferentes mídias. As mídias incorporadas podem ser resultado de autoria humana, com apoio de facilitadores, da captura automática de informações de contexto.

Neste capítulo foram apresentados trabalhos que evidenciam a possibilidade de construção dessas aplicações multimídia interativas de maneira automática e semi-automatizada.

Na Seção 2.1 foram discutidos os problemas relacionados à engenharia de documentos multimídia interativos, incluindo aqueles relativos à autoria de documentos multimídia em fase de exibição e interação (edição ao vivo). Foram propostas soluções relacionadas à engenharia de documentos, interação e casos de uso de autoria ubíqua. Através do trabalho relatado na seção foi possível avaliar a viabilidade da autoria ubíqua de documentos multimídia baseadas nas interações do usuário (complementando o conteúdo apresentado na TV). Foram observados também que o mecanismo de interação típico da TV (controle remoto) não oferecia uma experiência de interação adequada aos usuários e que o caráter "social"da TV (usuário geralmente assiste TV em ambientes coletivos) apresenta oportunidades e desafios para que a autoria do conteúdo se dê de maneira colaborativa.

Foram exploradas na Seção 2.2 a discriminação de momentos e intervalos em mídia usando-se dispositivos móveis pessoais e a autoria ubíqua de documentos multimídia colaborativa, aplicadas ao domínio de TV Digital. O trabalho relatado nessa seção evidenciou que dispositivos móveis podem apoiar o processo de autoria e que a autoria colaborativa, baseada em dispositivos pessoais, pode produzir aplicações potencialmente atrativas.

Foram realizados experimentos com interação colaborativa através de dispositivos móveis, sincronizada com o conteúdo da TV e a utilização de *stickers* como um elemento de interatividade. Foram estudados modelos de sincronização, avaliando maneiras alternativas à tradicional, baseada linhas temporais e interações com o controle remoto. Foi, ainda, observado que a construção de aplicações com duas telas não se trata de uma atividade trivial, especialmente quando são necessárias tecnologias diferentes (no caso dos protótipos desenvolvidos: JavaME e NCL). Esse trabalho apontou para a relevância de uma máquina de apresentações capaz de ser executada nos dois ambientes e que tenha facilitadores para novos modelos de sincronização. Isso reforçou a motivação da proposta das soluções apresentadas nos Capítulos 3 e 4.

No decorrer do trabalho foi observado que a autoria ubíqua de documentos multimídia poderia ser aplicada a outros domínios além da TV Digital Interativa, em ambientes com uma riqueza ainda maior de informações contextuais que podem agregar valor a uma apresentação multimídia. Isso motivou a proposta e desenvolvimento de uma ferramenta de autoria capaz de gerar um documento multimídia, de maneira automática, a partir da captura de informações contextuais (Seção 2.4). A ferramenta foi desenvolvida no escopo do projeto Presente e avaliada no cenário educacional. Esse trabalho reforçou a tese de que aplicações potencialmente atrativas podem ser construídas com a incorporação de mídias a uma mídia principal e que a autoria pode se dar através de informações contextuais. Foram realizados experimentos que indicaram que um conteúdo multimídia interativo pode ser produzido de maneira ubíqua, através da captura de informações contextuais e que essas informações agregam valor à mídia, tornando o conteúdo atraente para os usuários finais. O trabalho mostrou, ainda, que para uma efetividade na interatividade o conteúdo deveria estar disponível aos usuários de maneira simples, sem a necessidade da instalação de ferramentas ou aplicações específicas. Mostrou também a necessidade de novos mecanismos de sincronização de mídias. Esses desafios foram estudados e estão apresentados nos capítulos subsequentes.

Foi observado também que alguns requisitos em documentos multimídia são recorrentes em diversas aplicações. Como medida acessória na construção das aplicações multimídia interativas foi proposta uma abordagem baseada em componentes para documentos multimídia, apoiada por linguagem imperativa (Seção 2.4). Essa abordagem permitiu uma redução na complexidade de construção dessas aplicações sem perda dos benefícios de autoria das linguagens declarativas.

Capítulo 3

Apresentação de conteúdo multimídia em múltiplas plataformas

Durante o desenvolvimento de alguns projetos de P&D no Laboratório Lince/UFSCar foi utilizada a implementação de referência do *middleware* Ginga, executada em um emulador de *software* (máquina virtual), como meio de apresentação de conteúdo multimídia codificado através de documentos **NCL** (*Nested Context Language* Soares et al. (2006)). Entretanto a experiência de uso do emulador não era satisfatória em decorrência de fatores como os seguintes:

- O processo de instalação do emulador era complexo, sendo necessário o *download* e instalação de uma plataforma de virtualização que permita a execução da máquina virtual. Além disso o usuário devia também realizar o download da máquina virtual que encapsula o *middleware*;
- Para iniciar uma aplicação o usuário devia interagir com a máquina virtual (muitas vezes via linha de comando);
- A execução de vídeos através de um emulador não era um processo fluido. Os processadores e placas de vídeo mais recentes possuem recursos de decodificação de vídeo em *hardware*, entretanto a máquina virtual realizava todo o processo de decodificação em *software*, resultando em experiência de baixa qualidade para o usuário, especialmente quando utilizados vídeos com alta resolução;
- A customização do software (*middleware*) em execução na máquina vir-

tual era um processo complexo, o que dificultava a sua customização por desenvolvedores menos experientes, tornando a validação de artefatos produzidos em projetos de pesquisa mais custosa.

Foi observado, ainda, que os recursos trazidos pelo *middleware* Ginga, especificamente a linguagem **NCL** e sua máquina de apresentação, permitem a construção de aplicações multimídia bastante sofisticadas, que podem ser aplicadas a outros domínios além do ambiente de TV.

Uma das hipóteses apresentadas nesta tese para justificar a baixa utilização de conteúdo multimídia interativo refere-se aos custos de desenvolvimento de aplicações específicas para as mais diversas plataformas de *hardware* (*tablets, smartphones, smartTVs, PCs, set-top-boxes*) e *software* (sistemas operacionais). Nos projetos foi observado também que os usuários finais têm dificuldade em lidar com emuladores e instalação de aplicações em seus dispositivos para lidar com conteúdo multimídia interativo.

Essas dificuldades observadas levaram o autor desta tese a propor um novo mecanismo para apresentação de documentos multimídia baseados na linguagem NCL. Esse mecanismo de apresentação proposto baseia-se em tecnologias *web* e busca minimizar os problemas mencionados, contribuindo para que aplicações multimídia possam ser mais acessíveis. A proposta foi especialmente útil para viabilizar a ferramenta de autoria descrita na Seção 2.3 e a aplicação descrita no Capítulo 5.

A Seção 3.1 apresenta o resultado dessa pesquisa. Seu conteúdo trata-se da transcrição do artigo **WebNCL: A Web-based Presentation Machine for Multimedia Documents**, de autoria de **Erick Lazaro Melo**, Caio César Viel, Cesar Augusto Camillo Teixeira, Alexandre Coelho Rondon, Daniel de Paula Silva, Danilo Gasques Rodrigues, and Endril Capelli Silva. O artigo foi publicado no *Simpósio Brasileiro de Sistemas Multimídia e Web - Webmedia* (2012), tendo obtido a premiação de melhor artigo da conferência.

3.1 WebNCL: A Web-based Presentation Machine for Multimedia Documents

Presentation machines for multimedia declarative languages – especially the ones related with Interactive Digital TV (iDTV) and Internet Protocol TV (IPTV) – are usually embedded in devices and strongly coupled with the platforms when native code and API for the device's platform are used. Since much of the complexity to implement presentation machines lies on presenting and controlling different types of media (video, audio, image, text), and given that most of the modern browsers natively support those requirements, it becomes interesting to implement presentation machines using Web technologies to reduce their coupling with platforms. In this paper we discuss the advantages of a presentation machine for declarative multimedia languages implemented on top of Web technologies. As a proof of concept we implemented the WebNCL, a lightweight NCL presentation machine based on the web technologies stack (HTML 5/ JavaScript/ CSS). By using WebNCL, NCL documents can be presented in any device that has a HTML5 compatible browser, such as tablets, smartphones, smart TVs and PCs.

3.1.1 Introduction

Implementations of presentation machines for multimedia declarative languages – especially the ones related with Interactive Digital TV (iDTV) and Internet Protocol TV (IPTV) – are usually embedded in devices and strongly coupled with the platforms when the native code and API for the device's platform are used. For instance, the iDTV middleware Ginga-NCL (Soares et al., 2007, 2010a) contains a Nested Context Language (NCL) (Soares et al., 2006) presentation machine.

The Hypertext Markup Language (HTML) and other languages used to build Web pages, such as JavaScript and Cascading Style Sheets (CSS), are platform independent, because they are interpreted by Web browsers. Given the wide variety of browsers present in many platforms (PCs, smartphones, tablets, smartTVs, videogame consoles), it is becoming common to implement applications on top of the Web technologies to make them platform independent.

Since much of the complexity to implement presentation machines lies on presenting and controlling different types of media (video, audio, image, text), and given that most of the modern browsers natively support these requirements, it may be interesting to implement presentation machines using Web technologies to reduce their coupling with platforms.

Furthermore, there are several advantages in having a presentation machine for multimedia declarative language objects based on standard Web technologies. First it allows the reuse of declarative multimedia presentations developed for specific environment in other platforms. In addition, in a Web development perspective, a presentation machine built on top of web technologies stack (HTML5-JavaScript-CSS) is a framework which offers facilities for creating Web-applications with media synchronization requirements. The integration of multimedia declarative language objects, such as NCL documents into Webpages can enhance the capabilities of web applications and reduce the development costs.

Since a Web-based presentation machines are simpler to be used and integrated with other applications than native implementations, a more flexible and portable toolkit for development (testing and prototyping) of multimedia presentations can be provided. The presentation machine could be integrated with a text editor and provide real-time feedback or allow the presentation of multimedia objects without the necessity of sending into real or emulated devices.

Web browsers are available in many platforms. They can be found in smart TVs, smartphones, tablets and other devices. The device manufacturers can reuse the existing infrastructure provided to web browsers to integrate a multimedia presentation machine to their products. For instance, a web-based presentation machine provides a less invasive solution to add support for declarative applications in STB than embedding a dedicated middleware to present declarative multimedia documents, which usually runs on system space and communicate directly with the operation system, the presentation machine can run as a Web-application in the user space.

Finally, a Web-based presentation machine allows the use of rich declarative multimedia languages, such as NCL, to create classes of presentations that are not common in embedded environments due to platform restrictions.

In this paper we discuss the advantages of a presentation machine for declarative multimedia languages implemented on top of Web technologies. As a proof of concept we have implemented the WebNCL, a NCL presentation machine based on the web technologies stack (HTML5/JavaScript/CSS). By using WebNCL, NCL documents can be presented on any device that has a HTML5 compatible browser, such as tablets, smartphones and PCs.

The remaining of this paper is organized as follows: in Section 3.1.2 we discuss related works; Section 3.1.3 describes the development process of the WebNCL and important project decisions; Section 3.1.4 presents the architecture of the WebNCL; Section 3.1.5 describes how WebNCL can be integrated into a webpage and presents the compatible platforms; in Section 3.1.6 we discuss some aspects related to embedding WebNCL; Section 3.1.7 presents the conclusions and future works.

3.1.2 Related Work

The integration of web technologies with media synchronization languages has been investigated in some works. Cazenave et al. 2011 propose a solution to integrate SMIL Timesheets (Vuorimaa et al., 2008) into HTML documents.

Their solution allows web applications to use the SMIL temporal synchronization features through the engine *Timesheet.js*, implemented in JavaScript. A similar solution was adopted by Vuorimaa 2007. The solutions proposed by these authors include only a subset of SMIL and the focus is on providing synchronization mechanisms to web pages. The creation of full portable multimedia presentation machines was not investigated by them.

Jansen & Bulterman 2008 and Jansen & Bulterman 2009 present a solution for integrating SMIL documents to web pages through the use of browser plugins. A bridge between the Ambulant Annotator (César et al., 2006b) and a *webkit*-based browser was created to perform time synchronization in web pages.

Gaggi & Danese 2011 proposes the *SmilingWeb*, a SMIL presentation engine that runs on any browser that supports HTML5. Our work takes this direction and uses NCL as base for discussions and proposals. The implications of this approach on DTV systems and mobile devices are also pointed in our paper.

In the context of Digital TV systems and NCL Language, Cruz et al. 2008 describes a reference implementation of Ginga-NCL for mobile devices based on the Symbian platform. The focus of that study is to validate the Ginga-NCL architecture and to verify how it can be used to port the reference implementation to mobile devices. Ferreira et al. 2010 present an implementation of middleware Ginga-NCL for mobile devices based on Android platform. Their paper describes the steps for porting the middleware to that platform and evaluates the transmission on applications for mobile devices using the Digital Storage Media — Command and Control (DSM-CC) (Crinon, 1997) protocol through IP networks. Although they present the steps to port the middleware, they do not present solutions for cross platform development.

Other Digital TV middlewares have adopted web technologies for their declarative subsystems. This is the case of Broadcast Markup Language (BML) that is based on XHTML, CSS, Document Object Model (DOM) and ECMAScript (ECMA, 1999). The proposal described in this paper can solve some of the known problems of these systems, such as restrictions of adaptability, support for multiple devices and live presentation editing. As an example, WebNCL could be integrated to those middlewares, allowing them to display multimedia documents specified in NCL language by relying on technologies already supported (web-technologies).

3.1.3 Project Decisions

The WebNCL was developed using agile methods. Although a specific method was not used, we used a mix of SCRUM (Schwaber & Beedle, 2001) and XP (Beck & Andres, 2004).

We used the NCL ABNT Standards (ABNT Associação Brasileira de Normas Técnicas, 2007) as our main reference during the development. We also consulted Soares & Barbosa 2009 to clarify some NCL features and concepts which were not clear in the standards. Our previous experience ((Pimentel et al., 2010),(Freitas & Teixeira, 2009),(Teixeira et al., 2010b)) modifying the middleware Ginga-NCL reference implementation (Soares et al., 2007) contributed to define the requirements of our presentation machine as well as its architecture.

Before starting the development itself, we conceived some proofs-of-concept to determine the viability of implementation and tested some approaches. For instance, we developed an event manager to test the NCL links condition-action mechanisms. We also performed some stressing-tests playing several videos side by side and generating events in a high rate.

After the viability tests, we have conceived an architecture for the WebNCL presentation machine. At the beginning it was similar to the Ginga-NCL reference implementation architecture (Soares et al., 2007), but it suffered many changes during the process due to the JavaScript language particularities and in order to meet project decisions.

The first goal was to develop essential and core parts of the presentation machine, such as the NCL Document parser, NCL Representation Model and the Event Manager. The remaining parts of the implementation was driven by features. We chose features, such as transition or some descriptor propriety, and added support to it. After that, a simple NCL document was written for testing purposes. That NCL document was executed into the Ginga-NCL Virtual Machine and in our presentation machine running in different browsers (Chrome, Firefox, Opera and Safari) and in different operational systems (Linux, Windows, Mac OS, Android and iOS) to verify the implementation accuracy.

Languages and Platforms

In the project conception phase, we considered implementing our NCL presentation machine based on Adobe Flash Platform[1]. Different from the HTML5

[1]http://www.adobe.com/products/flashplayer.html

stack, Adobe Flash is already a mature platform. It also has great media support (especially video) including well-proved streaming protocols and technologies. In addition, its video formats are homogeneous – all up-to-date Adobe Flash Players support the same video formats. Another advantage is that by using the Adobe Air[2], we could easily develop a standalone desktop presentation machine.

However, a HTML5-based implementation can be used in more platforms since the HTML5 stack is based on open standards. For instance, iOS devices, such as iPhone, iPad and AppleTV, do not support Adobe Flash, but they have HTML5-compatible browsers.

We also believe that HTML5 is an ascending technology. Although it is only a draft and has heterogeneous support across browsers, it is already being used to build interesting and complex Web Application.

Computer power in presentation machines is required more to media presentation than to media orchestration. Since the web-browsers are responsible for media presentation and are usually implemented in native code, we believe the performance difference between WebNCL and the native-code presentation machine may be irrelevant.

Frameworks

Although HTML5 has native support for playing and controlling videos, we decided to use the Popcorn.js framework[3] for video playback. Porcorn.js is a JavaScript framework developed by the Mozilla Foundation for media synchronization and control; it also has an abstraction layer that allows developers to use HTML5 native videos via <video> tag and Adobe Flash video by a Flash Player in an agnostic way.

Popcorn.js offers a higher API for control media than pure HTML5. It is also well-tested in different browsers and platforms, which makes the presentation machine consistent across different browsers. Its abstraction layer can be used to add support for Adobe Flash based videos in our NCL presentation machine. It can be used to add features that cannot yet be used with HTML5 native videos, such as video streaming by Real Time Messaging Protocol (RTMP) [4].

A potential drawback when adopting Porcorn.js is the increased size and execution time of the JavaScript code. The size of a minified version of the framework core is 18,4KB and a version with all official plugins is 143KB.

[2]http://www.adobe.com/products/air.html
[3]Popcorn.js - http://popcornjs.org/
[4]RMTP (Real Time Messaging Protocol) is an Adobe proprietary streaming protocol

This overhead is not significant considering usual broadband Internet connection speed. We have not observed performance differences when compared a HTML5-pure and a Popcorn.js implementation. Also, the JavaScript code computing cost time is negligible when comparing to the cost of the video playback.

We also adopted the JQuery framework[5] since it makes the JavaScript development simpler and enhances presentation machine portability and reliability across different browsers and platforms.

Time-based Transition Handling

The Ginga-NCL reference implementation uses a scheduler to track the NCL presentations time-based transitions, such as the beginning of an anchor or the end of a video media. This is carried out by a thread that calculates all the presentation transitions and creates a timeline with those transitions. The thread waits the first transition and then executes the appropriated link actions. Sometimes it's necessary to recalculate the transitions timeline because the actions of the links can start or stop medias, changing the order of the transitions.

Given the asynchronous and event-driven characteristics of the JavaScript language, we adopted a different strategy to process time-based transitions in our presentation machine. The responsibility to inform when transitions occur is distributed among the players. Each media player has the information about the time-based transitions associated with its media (anchors). When a transition occurs the player triggers an event to the Event Manager.

If there are links that use a transition as condition, the Event Manager creates a listener for the corresponding event triggered by the player. When an event is triggered, the listener calls the functions for each link that depends on such a condition. If all the conditions of a link have been met, the link will execute its actions, triggering events to media players. The media players listen to events performed over its medias and execute the corresponding actions.

This approach provides a decoupled architecture, since the player only needs to know the transitions related to its media and the actions that can be performed over it. The Event Manager is agnostic and does not need to know anything about the media nature.

[5]JQuery Framework - http://jquery.com

3.1.4 Architecture

The WebNCL was designed using the Model-View-Controller (MVC) (Gamma et al., 1995) design pattern. Figure 3.1 depicts its architecture.

The Model Layer (Figure 3.1 bottom) consists of two components: NCL Representation Model (Figure 3.1 - component 1) and NCL Parser (Figure 3.1 - component 2). The NCL Representation Model contains JavaScript objects to represent each entity of NCL Documents (link, media, connector and etc.) and their values. The NCL Parser is responsible for translating an NCL Document represented in Extensible Markup Language (XML) code into objects of the NCL Representation Model component.

The View Layer (Figure 3.1 top) contains the media players. Those players are implemented in HTML5 and JavaScript code using the Popcorn.js framework as base. There is an Audio and a Video Player (Figure 3.1 - component 3), an Image Player (Figure 3.1 - component 4), a Plain Text Player (Figure 3.1 component 5) and a HTML Player (Figure 3.1 - component 6). The media players are responsible for playing their medias and for triggering events on transitions.

In addition to the media players, there is a Context Player (Figure 3.1 - component 7) that is located in the Control Layer (Figure 3.1 middle). The Context Player is not related to any HTML5 media - it represents an NCL context node. It contains references to the players of its nodes – Media or Context. The context player is placed on this layer because it is not directly related to "visible" elements in the presentation — it has a controller role.

The Control Layer also contains the Event Manager (Figure 3.1 - component 8), Interaction Manager (Figure 3.1 - component 9), Player Manager (Figure 3.1 - component 10) and Synchronization Manager (Figure 3.1 - component 11).

The Event Manager is the core of the presentation machine and implements the condition-action mechanism of NCL links. The Interaction Manager is responsible for handling the user's interaction via keyboard, virtual remote control, mouse and touchscreen devices. The Player Manager controls the creation and destruction of players objects and keeps track of all active media players. Finally, the Synchronization Manager is responsible for keeping the presentation synchronized when delays or errors occur due to network problems.

The remaining of this section explains how some important mechanisms work in the WebNCL implementation.

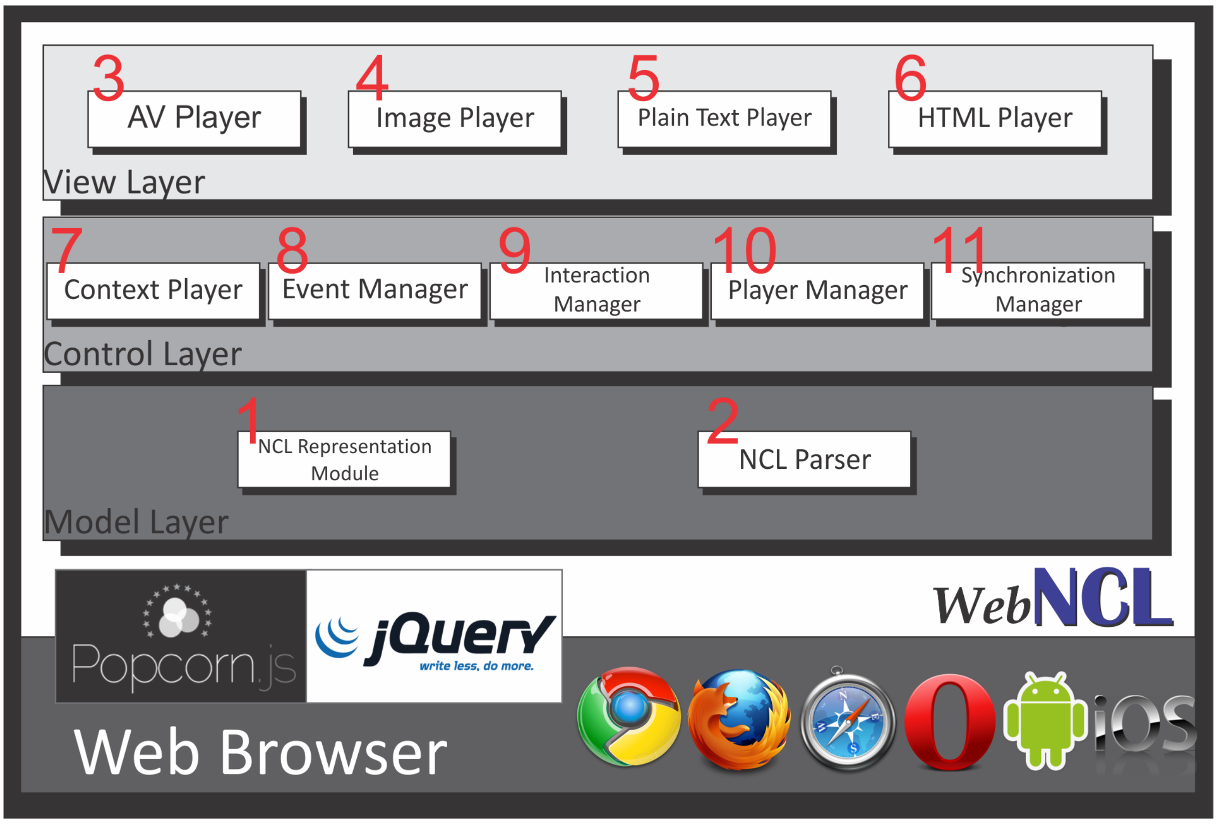


Figura 3.1: WebNCL Architecture

Initializing a Presentation

When an NCL document is loaded into the WebNCL, it is transformed into JavaScript objects from the *NCL Representation Model* component by the *NCL Parser*. Once the document has been translated, the *Player Manager* creates a *Context Player* for the top NCL document context node (<body>). It also creates players for all child nodes (medias and other contexts) of the NCL top context node. The media players start preloading the media, but the nested context players are not completely initialized – their internal nodes are only initialized when the context receives a play event. Each media player is associated with a HTML5 <div> element that contains a media representation tag (<img>, <audio>, <video>, etc.). The initial values properties and other specific properties of the media assume the values of its associated descriptor.

After the body's nodes have been initialized, the *Event Manager* starts to create the event listeners (for transition and user interaction) for the conditions of the body's links. It also creates a function for each link and register it to the corresponding event listeners.

The presentation starts only when the *Synchronization Manager* notifies that all media players are ready. When this takes place, the Event Manager triggers start events over the medias that are referred by the body's ports.

Link Processing

Listing 3.1 depicts an NCL link element named *sampleLink* and the connector *onBeginStart*; Figure 3.2 illustrates the processing of this link by the presentation machine using a simplified sequence diagram.

Listing 3.1: NCL Link

```
<causalConnector id="onBeginStart">
  <simpleCondition role="onBegin"/>
  <simpleAction role="start" max="unbounded"
    qualifier="par"/>
</causalConnector>

<link id="sampleLink" xconnector ="onBeginStart" />
  <bind component="video1" interface="area1"
    role="onBegin" />
  <bind component="image1" role="start" />
</link>
```

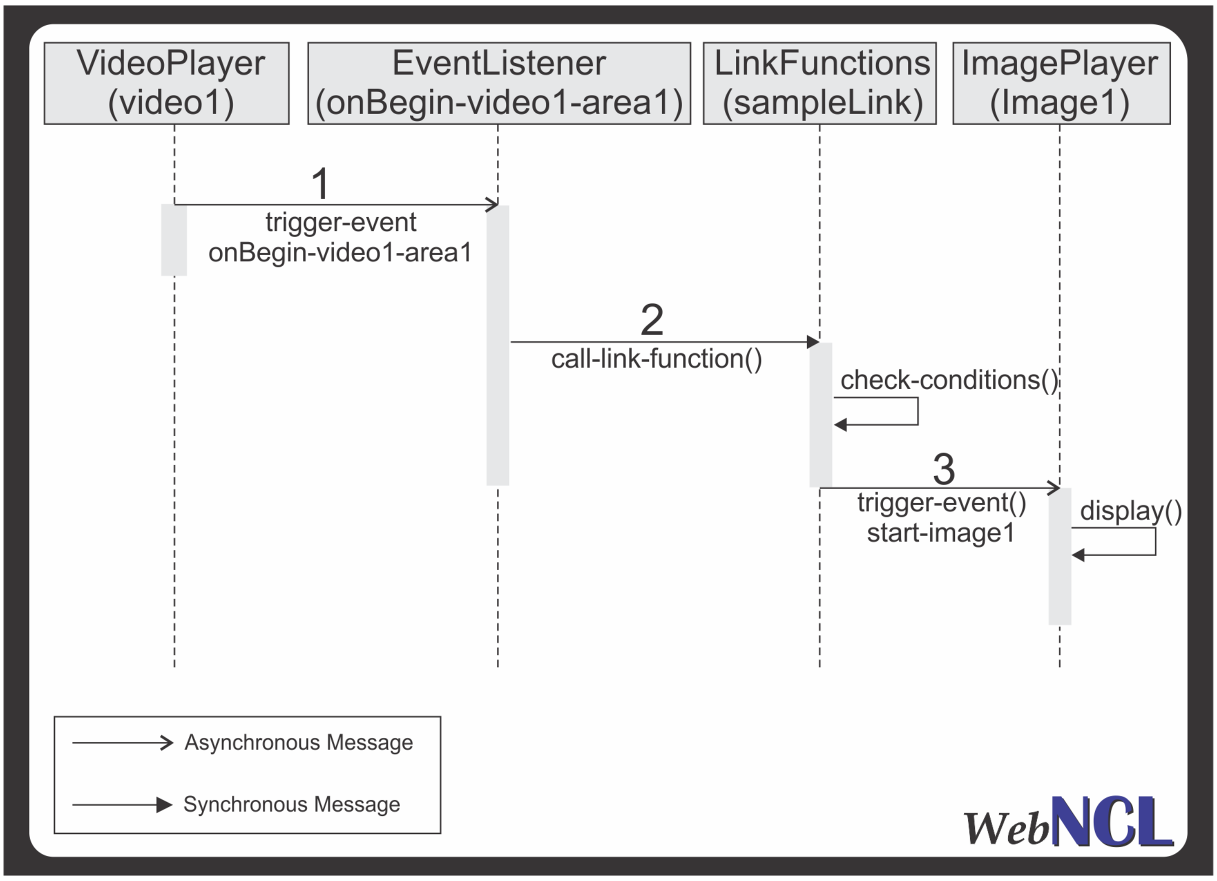


Figura 3.2: Link Processing Sequence

When the Video Player of the media named *video1* reaches the initial point of the anchor *area1*, it triggers an event (Figure 3.2 - step 1). Since there is a link that uses this transition as a condition (Listing 3.1, lines 7-8), the Event Manager creates a listener to that event beforehand. The listener executes the registered link functions, for instance, by calling the function *sampleLink* (Figure 3.2 - step 2).

The *sampleLink* function checks if the link's conditions have been met. When the condition are met, the link functions trigger events to their actions. In this example, *sampleLink* has just one condition and it was met, so the *sampleLink* function will trigger events for its actions (Figure 3.2 - step 3). *Image1* Player receives the event and processes it, making *image1* show up.

This process is more complex when the link has a compound condition. For compound conditions that uses the AND operator the link function keeps track of the conditions met using a condition table. It registers the time when the first condition is met. When the link function is called by the event listener, it verifies the registered time — if it took place more than one second earlier, the function clears the condition table, registers the current time and checks the table for the condition entrance — in other words, conditions which are too old are discarded. If the time stored is not older than one second, the function checks the condition table for that condition. When the condition table is fully checked, the link function clears the condition table and processes the link actions.

For compound conditions that use the OR operator, the link function registers the last time when the actions were performed. When an event listener calls the link function, it verifies if the time registered took place more than one second earlier; if so, the link function processes the link actions and stores the current time. If not, the link functions just returns —- in other words, the link trigger is ignored because it has just been executed.

The presentation machine also supports links that construct complex conditions by cascading different types of compound conditions.

For actions, the presentation machine supports both PAR and SEQ operators. For actions that use the PAR operator, the link function triggers the events and lets the browser scheduler handles the actions as parallel as it can. For actions that use the SEQ operator, only the first event is triggered, but this event carries a callback function. When the event is processed by Media Players, the callback function is called. This callback function triggers the next event, and this event has a callback function that will trigger the next event, and so on. Using those callback functions the presentation machine is capable of processing link actions in a sequential way.

We adopted a compound condition window of one second to trigger the actions because triggering condition events are not atomic operations. The events may be triggered in a very short window time, making us consider these events as simultaneous. Similar strategy is found in others implementations, such as the Ginga-NCL reference implementation (Soares et al., 2007).

User's interactions are implemented as they were transition events. When the user interacts with the presentation, an event is triggered. If there is a link that has user interactions as its conditions, the event will be handled by an event listener.

Synchronization Mechanisms

To keep media synchronization, even under network fluctuations, a sync mechanism was developed to pause the presentation when one or more players face some problem and resume it when all players are ready again.

The mechanism is first activated when a context starts – including the body, as this is the moment when the players start fetching the media to play. The presentation is paused and waits until all players have their media ready. The mechanism will also be activated when any player — usually continuous media players receiving their media by streaming —- triggers an event notifying a problem to play its media.

The *Synchronization Manager* receives the event and forces a pause in the presentation. When the player triggers another event notifying that it is ready again, the *Synchronization Manager* resumes the presentation.

3.1.5 Embedding WebNCL in a webpage

In order to present an NCL Document using WebNCL, it is necessary to embed it on a web page, as seen on the HTML page depicted in Listing 3.2.

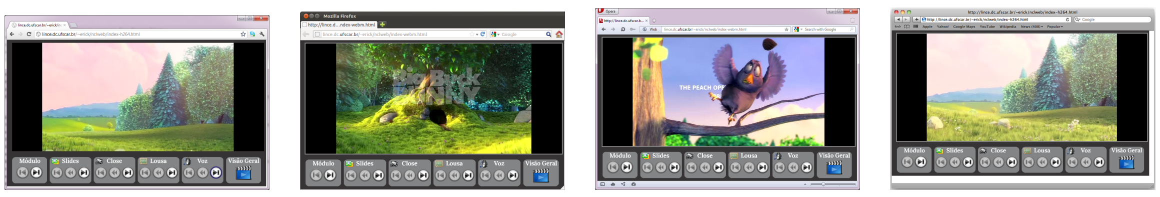
In Lines 3-5 in Listing 3.2, the WebNCL source code to be imported. Between lines 6 and 14 the settings of appearance and positioning are defined as CSS style. Line 18 instantiates the presentation machine, passing the path of the NCL Document to be presented and the id of the HTML <div> where the presentation will be displayed. Finally, in line 21, the presentation is started. More than one WebNCL presentation machine can be embedded in the same HTML page.

During the experiments, an NCL Document with about 4,800 lines and 173 media nodes (videos and images) ran smoothly when submitted to WebNCL. The WebNCL was tested in several browsers and platforms. Figure 3.3 depicts the WebNCL running in different browsers in PC enviroment: Chrome in Win-

Listing 3.2: HTML code to embed the WebNCL

```
<html>
  <head>
    <script type="text/JavaScript"
      src="js/WebNCL.js">
    </script>
    <style type="text/css">
      .nclPlayer {
        position: fixed;
        left: 0px;
        top: 0px;
        width: 854px;
        height: 480px;
      }
    </style>
  </head>
  <body>
    <div id="ncl" class="nclPlayer"/>
    <script type="text/JavaScript">
      var p = new WebNclPlayer("main.ncl",
        "ncl");
      p.start();
    </script>
  </body>
</html>
```

dows (Figure 3.3(a)), Firefox in Linux (Figure 3.3(b)), Safari in Mac OSX (Figure 3.3(d)) and Opera in Windows (Figure 3.3(c)). Figure 3.4 depicts the WebNCL running in an Android device and Figure 3.5 ilustrates the execution in an iOS device. Any combination of browsers and platforms is possible.



(a) Chrome (b) Firefox (c) Opera (d) Safari

Figura 3.3: WebNCL running on PC browsers

User Interaction

Given that many NCL applications use a TV remote control as its primary input device, we have to provide a similar way to interact with these applicati-ons. This can be done using different approaches like mapping the TV remote

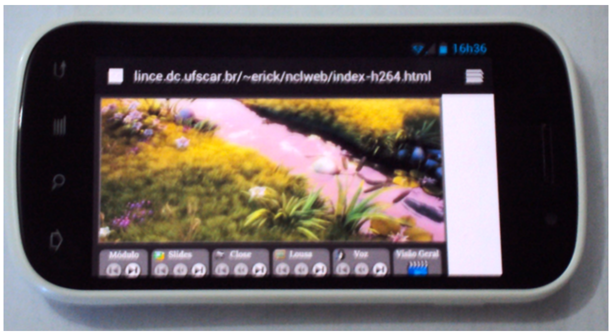


Figura 3.4: WebNCL running on Android

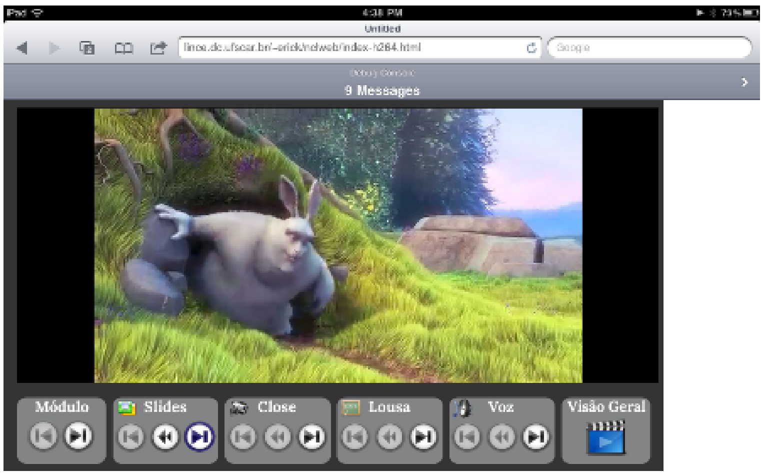


Figura 3.5: WebNCL running on iPad

control on keyboard and via a virtual remote control. We implemented both solutions in the WebNCL.

Table 3.1 describes the mapping of TV remote control on a standard PC keyboard. When a mapped key is pressed, the Interaction Manager triggers the corresponding event.

TV Remote Control Button	Keyboard Key
Numbers 0-9	Numbers 0-9
Arrows	Arrows
Enter | OK	Enter
RED	Q
GREEN	W
YELLOW	E
BLUE	R

Tabela 3.1: Keys mapping

The virtual remote control is built externally to the presentation machine using HTML and JavaScript. It uses the presentation machine JavaScript API to post key events. This decoupling allows creating different virtual TV remote controls without modifying the presentation machine. For instance, Figure 3.6(a) presents a more complete remote control while 3.6(b) presents a simple.

We also added support to interaction via mouse and touchscreen devices, since the WebNCL can run in environments with richer input devices. When the mouse is over a media, that media receives the focus. If the left or right button is pressed, an OK button event is triggered. In touchscreen devices, tapping in a media without the focus is equivalent to change the focus to that media. A second tap (or double tap) on that media triggers an event equivalent to pressing the OK button.

Although we used rich input devices just to mimic interactions which are already possible with TV remote controls, it is possible to extend the interaction paradigm to support new types of interaction which are more natural in PC or tablet environment (long tap, swipe, double click, drag and drop). In work in progress we are adding support to these interactions in WebNCL.

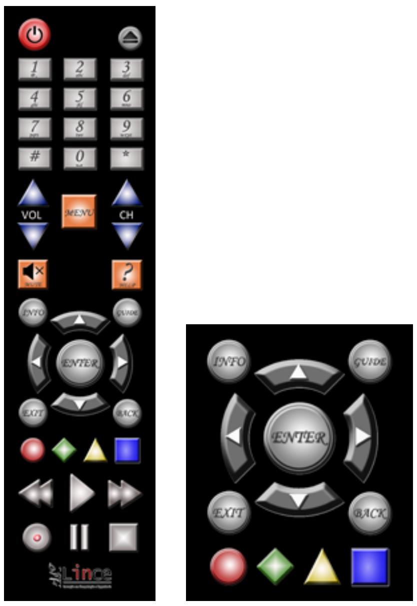

(a) Complex (b) Simple

Figura 3.6: Virtual remote controls

3.1.6 Discussions

In this paper we present a solution for building a declarative presentation machine based on web-technologies. Although the WebNCL can allow the execution of NCL documents in any platform that offers a HTML5-compatible browser, it also can also be used as a component of embedded DTV middleware, since the web stacks are compatible with many embedded platforms.

However, the execution of HTML based applications is usually done in a sandbox environment. The application does not have access to the same lower-level APIs available to native codes. Efforts have been made to provide some new APIs to web-browsers, allowing the applications to access accelerometers, GPS, camera and local storage. It accomplishes many requirements of some applications but for some we still need to access native resources. This is the case of the DTV environment – there is a common core that is very platform dependent and that provides very specific services required by DTV applications (for instance: access to the EPG data).

In order to use WebNCL as a component of an iDTV middlware, we must provide low level service access to the presentation machine. Some approaches may be considered in order to extend the WebNCL to provide specific DTV services:

- The HTML5 engine may be embedded in native software, using a HTML rendering engine. This engine can be extended to provide new APIs that bridges the HTML application to the native code. This strategy has been adopted in the mobile environment with the PhoneGap framework (Charland & Leroux, 2011), which provides some APIs to handle mobile common features, allowing the same code base to run across many platforms (iOS, Android, Symbian, WebOS, Windows Phone).

- Some web-browsers provide ways to create native extensions. The most common way is using the Pepper Plugin API[6] which is a cross platform API to build native client web browser plugins. It allows the user to execute native compiled code in the browser, which runs in a sandbox environment that tries to avoid malicious codes from damaging the system.

- Existing plugins available may be used, such as Java Applets. Specifically in the DTV environment this strategy can be used to integrate declarative applications with procedural Java applications based on GEM Framework (ETSI, 2005) or Java DTV API (JavaDTV, 2010). Java applets allow the execution of Java applications that are able to access protected resources (out of the browser sandbox).

HTML5 browser issues

WebNCL relies on the web-technologies stack. The browser imposes restrictions regarding the supported media types and its behavior. Currently, there are some limitations imposed by the browsers:

- The iOS devices are limited to playback a single audio or video stream at any time. Playback more than one video at same time is not allowed. The pre-fetching of streamings is also not supported by that platform.

- The video in HTML5 is not supported in the same way by different browsers. Only some browsers support the H.264 encoder, while others support the WebM and OGG encoders. The same goes for <audio> tag.

- The CSS3 properties are handled in different ways by the browsers. For instance, it may impact on how NCL transitions will be presented.

- Browsers eventually implement the Same Origin Policy[7], which avoid a web application to access local files. In some cases, to run local NCL applications it is necessary to access the WebNCL from a Hypertext Transfer Protocol (HTTP) Web Server or disable that browser security mechanism.

[6]https://developers.google.com/native-client/overview
[7]http://www.w3.org/Security/wiki/Same_Origin_Policy

3.1.7 Final Remarks

This paper discusses the possibility of increasing the portability of declarative synchronization languages presentation machines by using web-based implementations. As a proof-of-concept we developed the WebNCL, an NCL presentation machine that presents NCL documents in HTML5 compatible browsers.

The WebNCL brings some implications and possibilities. TV stations may allow users to access the same interactive content broadcasted on TV in the web environment. The same interactive experience can be offered to web users. For instance, a soap-opera can use the same application for viewing characters profiles in both platforms. Another possibility is to offer the same interactive merchandising application in iDTV and Web.

The development of a toolkit based on WebNCL for NCL document authoring is interesting, since, according to the authors experience, the difficulties to setup an devlopment environment may be reduced to developers when compared to traditional virtual machines and emulators.

Given the possibility to embed the NCL presentation machine into webpages, many applications can be conceived. For instance, the Club NCL[8] can provide features to online previewing NCL documents. Educational tools could incorporate interactive videos in a lightweight approach using WebNCL. The traditional video lectures may be replaced by interactive multimedia lectures. This approach is being exploited by the authors.

WebNCL also helps the exploration of the multiple devices NCL support. As WebNCL runs on any HTML5 compatible browsers (which are available in most modern mobile devices) the efforts to integrate the devices — set-top-boxes and mobiles — can be smooth.

Currently, we are extending WebNCL to provide support to the Lua scripting language and implementing Lua APIs specified in ISBD-T. Through these efforts we hope that applications developed for the Ginga-NCL environment can be entirely reused in the web environment.

WebNCL is open-source and the source code and demo applications are available at
http://webncl.org.

[8]Club NCL is a portal to share NCL applications - http://club.ncl.org.br/

3.2 Considerações Finais

A promoção da interação do usuário com aplicações multimídia potencialmente atrativas depende diretamente da disponibilidade dessas aplicações. Um dos fatores de atratividade e satisfação do usuário refere-se à disponibilidade das aplicações.

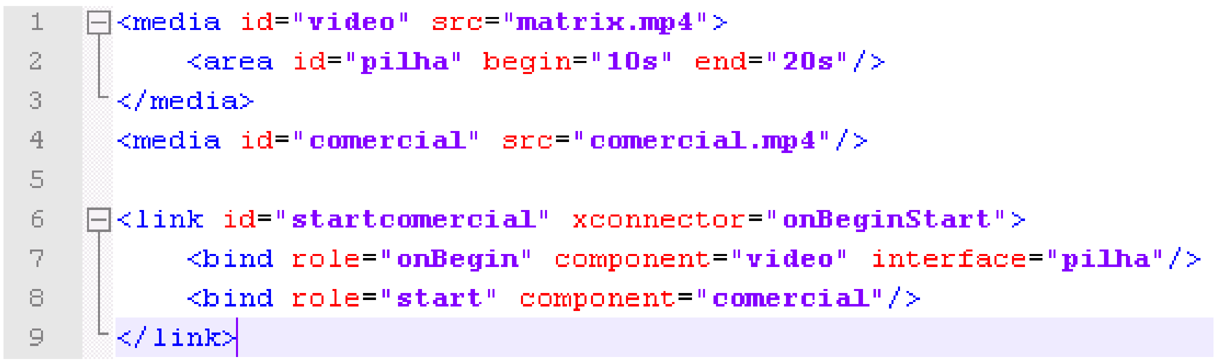
Os resultados descritos neste capítulo viabilizam a difusão de aplicações multimídia em uma grande variedade de dispositivos e plataformas, contribuindo para que as aplicações estejam mais acessíveis ao usuário. Além disso *WebNCL* é um facilitador para a utilização de aplicações multimídia interativas em ambientes distintos da TV Digital Interativa, como plataformas móveis e na *web*, que possuem um apelo ainda maior à interatividade que a TVDi.

CAPÍTULO 4

Sincronização Multimídia não Convencional

Tradicionalmente a sincronização de aplicações multimídia é baseada em relações temporais. As linguagens de sincronização multimídia, de maneira geral, trabalham com os mesmos conceitos. Na linguagem NCL a definição de âncoras de sincronismo ocorre em função de informações temporais (Figura 4.1).

Na fase de autoria o autor de um documento multimídia em muitos casos define âncoras em função do conteúdo apresentado na TV (Figura 4.2). A representação temporal dessas âncoras trata-se, em muitos casos, de um artifício tecnológico. A intenção original do autor do documento refere-se ao conteúdo, que na especificação do documento é codificado como uma relação temporal. Neste caso o contexto a qual o autor se refere pode ser perdido.



```
<media id="video" src="matrix.mp4">
    <area id="pilha" begin="10s" end="20s"/>
</media>
<media id="comercial" src="comercial.mp4"/>

<link id="startcomercial" xconnector="onBeginStart">
    <bind role="onBegin" component="video" interface="pilha"/>
    <bind role="start" component="comercial"/>
</link>
```

Figura 4.1: Definição de âncoras temporais em NCL

Não existe uma relação explícita entre as âncoras e o contexto em que estão inseridas.

A estratégia do emprego de marcações temporais para a especificação de âncoras tem sido adotada em larga escala principalmente pela complexidade da construção de máquinas de apresentação. A sincronização baseada no tempo é amplamente estudada e existem diversas técnicas conhecidas. Uma das principais vantagens dessa abordagem refere-se ao baixo consumo de processamento, característica importante para a execução do software em sistemas embarcados. Entretanto é perceptível o aumento da capacidade dos dispositivos. Esse desenvolvimento tecnológico permite a experimentação de novas técnicas para sincronização multimídia.

O modelo tradicional de sincronização, em que as âncoras são definidas apenas no tempo cria uma dissociação entre a especificação do documento multimídia e o contexto considerado pelo autor do documento – uma cena é diferente de um *timestamp*. É comum no ambiente de TV existirem diversas versões do mesmo conteúdo, como compactos de eventos esportivos e versões de filmes com cortes de cenas. Neste caso as relações temporais são comprometidas. É necessário um esforço adicional de edição do documento multimídia para que a intenção do autor seja preservada.

O reúso de aplicações/documentos fica prejudicado nesse caso. Sempre que uma das mídias envolvidas no processo tiver que ser editada, e essa edição alterar a *timeline* do vídeo, é necessário que o documento também seja editado já que este é fortemente atrelado à *timeline*.

O processo de autoria também fica pouco natural, já que a especificação em alto nível envolve o estabelecimento de relações entre mídias baseadas no conteúdo, no que está sendo apresentada, e não em instantes de tempo.

Um exemplo desse problema pode ser encontrado em ambientes de compartilhamento de vídeo e legendas, em que o usuário pode localizar legendas em diversos idiomas para determinado filme (geralmente codificadas segundo padrões como SRT). Os usuários geralmente têm muitos problemas para sincronizar as legendas com o vídeo, já que por vezes a mídia usada para a construção da legenda é diferente da mídia obtida pelo usuário, embora apresentem o mesmo conteúdo. Os softwares para reprodução de vídeo geralmente apresentam recursos para contornar esse problema, dando ao usuário a opção de sincronizar a legenda com o vídeo manualmente. O processo de sincronização consiste, basicamente, em incrementar/decrementar o relógio da legenda em alguns segundos. Esse processo manual funciona quando existe apenas um deslocamento da legenda, mas é impraticável quando existem cortes em segmentos intermediários do vídeo.

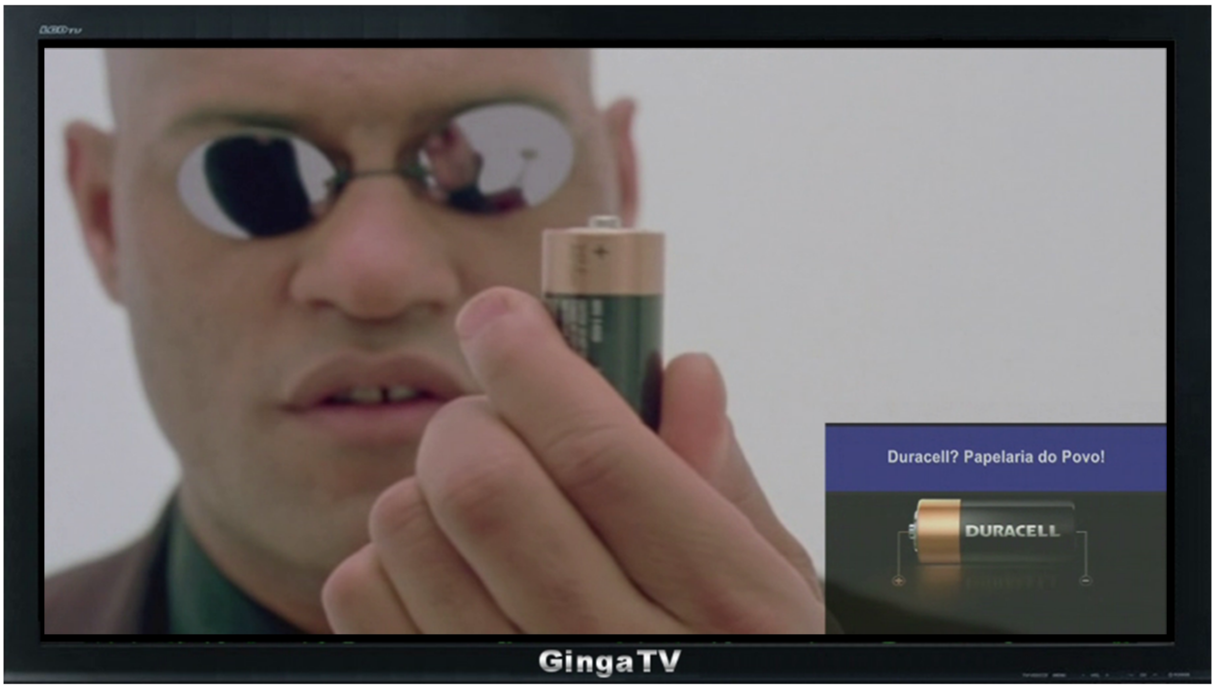


Figura 4.2: Sincronização relacionada com o conteúdo (Soares et al., 2007)

Neste capítulo são apresentadas estratégias para sincronização de mídias de maneira não convencional. A sincronização pode ser realizada tanto localmente quanto de maneira distribuída, em tempo real ou utilizando-se técnicas de pré-processamento.

Na Seção 4.1 são propostas técnicas para a sincronização em tempo real (conteúdo ao vivo), baseadas no conteúdo das mídias.

Na Seção 4.2 é apresentada uma abordagem para o controle síncrono de apresentações multimídia distribuídas. Essencialmente foi proposto um mecanismo para que máquinas de apresentações multimídia sejam controladas remotamente. O conteúdo dessa seção refere-se ao artigo **An Approach for Controlling Synchronous Remote Instances of a Multimedia Presentation**, que encontra-se em fase de submissão.

Na Seção 4.3 é apresentada uma proposta de integração de mídias de alta performance, produzidas por *clusteres*, a apresentações multimídia. Nele são discutidos tanto os aspectos de incorporação desse tipo de objeto multimídia a apresentações quanto mecanismos que permitam a sincronização entre a máquina de apresentação local e o conteúdo em execução no *cluster*. O conteúdo dessa seção refere-se ao artigo **Multimedia Presentation Integrating Interactive Media Produced in Real Time with High Performance Processing**, de autoria de Caio César Viel, **Erick Lazaro Melo**, Arthur Pedro Godoy, Diego Roberto Colombo Dias, Luis Carlos Trevelin e Cesar Augusto Camillo

Teixeira, publicado nos anais do *Simpósio Brasileiro de Sistemas Multimídia e Web - Webmedia* (2012).

Em todos os casos o objetivo principal é viabilizar a agregação de mídias complementares a uma mídia principal, preferencialmente contextualizadas, visando enriquecer a experiência do usuário. Mídia complementar pode ser tanto um objeto sintético (Seção 4.3) quanto uma interação realizada via comunicação síncrona de vídeo (Seção 4.2).

4.1 Técnicas de Sincronização Multimídia Baseada no Conteúdo

O conteúdo multimídia possui características que permitem a sua identificação de diversas maneiras, de acordo com a sua natureza. A identificação desse conteúdo pode se dar com base em elementos da mídia como o texto apresentado, áudio, imagens ou metadados.

Alguns tipos de mídia, como vídeos, apresentam características que permitem a sua distinção de múltiplas maneiras. Tipicamente a indexação e busca de vídeo consiste no processamento do sinal de áudio e de imagens.

Os estudos realizados no escopo desta tese para sincronização baseada em conteúdo estão focados primariamente em mídias do tipo vídeo. Especificamente foram experimentados vídeos de TV disponibilizados via *broadcast* em transmissões terrestres.

Para a implementação do conceito de sincronização multimídia baseada em conteúdo, neste trabalho, explorou-se inicialmente a utilização de sinais de áudio (extraídos do vídeo). Através do processamento de sinais de áudio é possível a extração de características do sinal, permitindo a comparação com outros sinais e o estabelecimento de uma relação de similaridade.

A indexação de sinais de áudio tem sido objeto de diversos estudos acadêmicos. A biblioteca jAudio (McKay et al., 2005) é resultado de um desses estudos. A biblioteca apresenta um conjunto de recursos para a extração de características de sinais de áudio e algoritmos para a comparação entre sinais dessa categoria. O trabalho proposto por Dzhambazov (Dzhambazov, 2009), estende a biblioteca jAudio, apresentando uma solução para a indexação e recuperação de músicas – dada uma amostra de uma música (pode ser inclusive cantada pelo próprio usuário) ele é capaz de localizar músicas similares, indicando, inclusive, o grau de similaridade. Na solução apontada a amostra de áudio é divida em janelas de intervalo de cerca de 1 segundo e para cada janela é calculada a métrica de contorno de melodia (é utilizada a frequência com mais potência para determinação do tom). É importante ressaltar que muitas das métricas de áudio obtidas são tolerantes a interferências na captação do áudio, podendo ser utilizadas, inclusive, em ambientes coletivos (como aeroportos, hospitais e outros ambientes públicos).

Com base na biblioteca jAudio e no trabalho de Dzhambazov foi desenvolvida, no escopo deste tese, uma prova de conceito que permite a sincronização de elementos multimídia com base no sinal de áudio. Embora a utilização dessa abordagem tenha apresentado aspectos positivos, algumas restrições se mostraram:

- O processo para extração das características do sinal não demanda elevado uso de recursos computacionais, entretanto o conceito de "janelas" utilizado introduz um atraso para a identificação das âncoras. Foram feitos experimentos com janelas menores que 1 segundo, aplicados a conteúdo de TV, mas os resultados obtidos foram insatisfatórios – os melhores resultados foram obtidos com janela de 2 segundos. Desta forma tem-se um atraso de pelo menos 2 segundos para a identificação das âncoras.

- Existe perda de informação quando o objeto de sincronização é um vídeo, já que apenas o sinal do áudio é processado. Quando se considera conteúdo de TV (novelas, programas de auditório e comerciais) foi notado que é comum esses conteúdos apresentarem músicas de fundo, que muitas vezes se repetem em conteúdos diversos. Isso acaba gerando inconsistências no processo de sincronização, com a identificação inapropriada de âncoras.

Para a sincronização fina essa abordagem mostrou-se inviável. Entretanto em cenários em que existe uma tolerância maior a atrasos (requisitos de qualidade de serviço mais fracos) essa estratégia pode ser aceita. Os experimentos também apontam para a necessidade de utilização de técnicas adicionais para a identificação inequívoca de âncoras de conteúdo de TV, já que o áudio nem sempre é uma característica singular.

A sincronização multimídia, especialmente no ambiente de TV, está bastante relacionada à sincronização de um vídeo com outras mídias. O vídeo tem uma particularidade interessante que é a informação visual, muitas vezes mais rica que apenas o sinal de áudio. Através de processamento de imagem assinaturas consistentes do conteúdo podem ser obtidas, permitindo a identificação de âncoras de maneira mais efetiva e, em tempo suficientemente rápido para não prejudicar aplicações que demandem diferenças de sincronismo na faixa de décimos de segundo.

Através do processamento de imagem podem-se definir âncoras vinculadas ao conteúdo que estejam associadas à imagem transmitida pela TV ou a partes dela. Esse tipo de técnica permite realizar processamento da imagem, para localizar a correspondência entre imagens ou partes dela. Desse modo é possível realizar uma busca mais refinada, tentando localizar pontos de sincronismo dentro de um frame de vídeo, como por exemplo, um marcador ou uma parte da imagem. Pode, ainda, ser considerada a geração de *fingerprints* das imagens e a indexação e buscas baseadas em assinaturas, reduzindo-se assim a complexidade computacional para a busca.

Pesquisas têm sido desenvolvidas para o desenvolvimento de técnicas para indexação multimídia. A biblioteca OpenCV (Bradski, 2000) implementa alguns desses algoritmos e tem sido largamente empregada em aplicações de visão computacional.

O escopo desta tese não consiste em pesquisa relacionada ao processamento de imagens e sinais. Todavia a implementação do conceito de sincronização multimídia baseada em conteúdo requer a utilização de tais técnicas para sua viabilização. Foram realizadas pesquisas na literatura a respeito do assunto e as soluções encontradas apresentam alto grau de sofisticação, endereçando problemas além daqueles descritos neste trabalho, demandando recursos computacionais incompatíveis com os requisitos de execução em tempo real (*streaming*) em *hardware commodity*. Diante disso foram experimentadas duas abordagens para a implementação do conceito de sincronização baseada em conteúdo através do processamento de imagens, uma baseada na comparação de matrizes e outro no cálculo de assinaturas.

Os algoritmos descritos a seguir apoiam a solução do problema, mas não se tratam de soluções exclusivas. Outras técnicas podem ser aplicadas para a implementação da sincronização baseada em conteúdo.

4.1.1 Algoritmo baseado na comparação de matrizes

Uma das técnicas avaliadas no escopo desta tese consiste na busca baseada na comparação de matrizes. Algumas etapas foram definidas para a realização da busca, conforme Figura 4.3.

A aplicação de algoritmos de *matching* de imagens com resolução na ordem das transmitidas pelas emissoras de TV (em geral superiores a 720x480 *pixels*) implica em elevado custo computacional. Visando contornar esse problema, foi experimentada a redução dos *frames* de maneira decrescente até a dimensão de 12x8 *pixels*, mantendo-se a imagem com as 3 componentes RGB. Com isso o tempo de execução do algoritmo de *matching* foi reduzido sensivelmente,

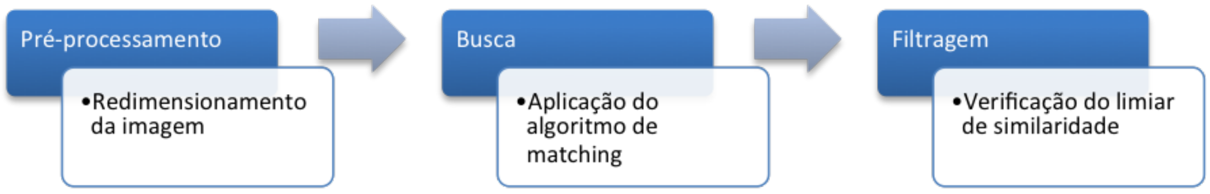

Pré-processamento
•Redimensionamento da imagem
Busca
•Aplicação do algoritmo de matching
Filtragem
•Verificação do limiar de similaridade

Figura 4.3: Etapas para verificação de similaridades entre frames de vídeo

viabilizando a sua utilização em aplicações de tempo real.

Foi utilizado um dos algoritmos para cálculo de similaridade entre matrizes disponibilizados pela biblioteca OpenCV para o processo de comparação entre *frames* de vídeo (método CV_TM_CCORR_NORMED).

A resposta obtida pelo algoritmo aplicando-o à imagem redimensionada não foi sensivelmente afetada, se comparada à imagem original. Quando é aplicada uma redução maior ao frame, como 6x4 pixels, o tempo de *matching* não é sensivelmente afetado, mas existe uma perda significante na precisão dos resultados – falsos positivos são gerados com frequência.

A Figura 4.4 apresenta os resultados dos experimentos realizados. No eixo vertical é indicada a quantidade de comparações por segundo entre *frames* obtida, de acordo com a resolução da imagem utilizada (eixo horizontal).

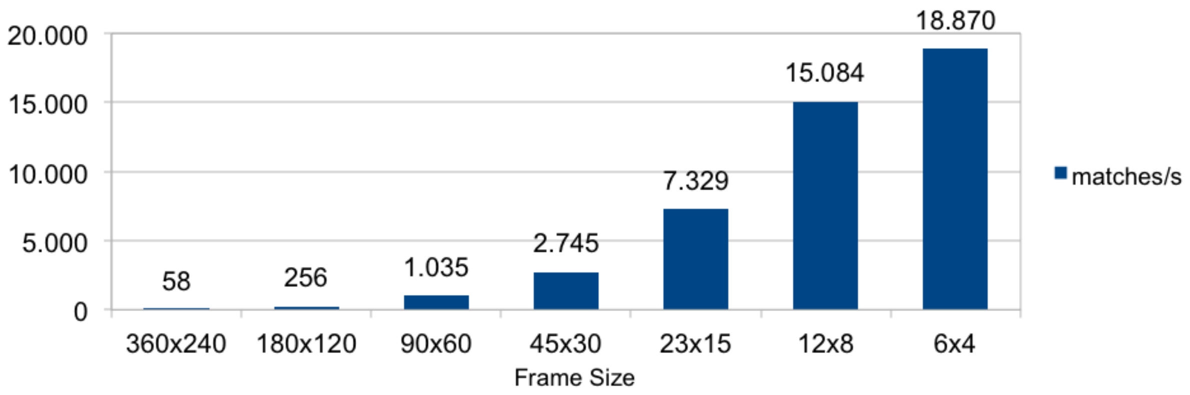


Figura 4.4: Taxa de comparação de frames

A implementação do mecanismo de comparação foi desenvolvida em linguagem C++. Os experimentos foram executados em um computador com processador Intel Core i5, utilizando uma única *thread*.

Considerando um fluxo de vídeo a uma taxa de 30 *frames* por segundo é viável comparar cada frame capturado com cerca de 630 *frames* candidatos. A quantidade de candidatos pode ser aumentada se for considerada uma taxa menor de *frames* por segundo. Através de experimentos foi possível notar que, excetuando-se as transições de cena, em geral existe pouca diferença entre um frame e seus vizinhos – o nível de similaridade entre eles é elevado. Para o conteúdo de TV experimentado (24 horas contínuas de vídeo capturado em alguns dos principais canais abertos em audiência de TV brasileiros em audiência: Globo, SBT e Band), uma taxa de 3 *frames* por segundo foi considerada aceitável, permitindo com que cada *frame* capturado pudesse ser comparado com outros 6.290 candidatos. A taxa de *frames* a ser considerada deve levar em conta a dinâmica do conteúdo, requisitos de tempo de resposta e disponibilidade de CPU do ambiente.

Tais limitações devem ser consideradas ao se adotar essa estratégia, já que podem torná-la inviável em certas aplicações que exigem pequena latência no reconhecimento mas que requerem taxa de *frames* elevada para melhor precisão.

4.1.2 Algoritmo baseado no cálculo de assinaturas

Com o objetivo de se identificar em tempo real segmentos de vídeo como propagandas, vinhetas ou cenas de um filme em fluxos contínuos de vídeo, como transmissões via radiodifusão terrestre de televisão, foi projetado e implementando um algoritmo de reconhecimento capaz de realizar buscas em grandes bases de vídeo com uma baixa latência. O algoritmo considera uma base composta dos segmentos de vídeo a serem reconhecidos. Os vídeos dessa base são comparados com o fluxo de vídeo contínuo.

A busca dos segmentos de vídeo é feita a partir da comparação dos *frames* dos vídeos contidos na base com os *frames* capturados do fluxo de vídeo contínuo. Ao invés de realizar a comparação a nível de *pixels* (o que demanda grande esforça computacional e tempo, quando o tamanho da base de amostra é elevado), o algoritmo extrai uma métrica de cada frame dos vídeos e os organiza em uma estrutura de dados otimizada para busca.

O algoritmo baseia-se na similaridade entre *frames* e não na comparação entre *pixels*. Essa estratégia busca minimizar os problemas relativos a interferências na transmissão (perda de *pixels*) e diferenças no contraste (diferentes canais podem apresentar o mesmo conteúdo com sutis diferenças de contraste). Uma sequência contínua de *frames* semelhantes é considerada, pelo algoritmo, uma sequência de *frames* iguais. Essa estratégia reduz o tamanho da estrutura de dados de busca e minimiza problemas relativos a interferências em algum *frame* causada por falhas na transmissão.

O algoritmo é divido em três fases: extração da métrica, construção da árvore e identificação de segmentos. Cada etapa é detalhada nas seções subsequentes.

Extração da Métrica

A técnica adotada para extração de uma métrica de cada *frame* dos vídeos da base de segmentos baseia-se na divisão do *frame* em quadrantes, como ilustrado na Figura 4.5(a). Para cada um dos quadrantes é calculada a média dos valores de seus pixels. Considerando-se apenas *frames* em tons de cinza e a divisão em 16 quadrantes cada frame do vídeo é representado por um vetor de 16 números, cada número representando a média de cores de um dos quadrantes.

O algoritmo foi inicialmente concebido para calcular a métrica transformando as imagens em tons de cinza e considerando os 16 quadrantes. Entretanto, durante os testes realizados, versões diferentes do método para extração das métricas foram experimentadas, como por exemplo, a utilização apenas da parte central do *frame* (Figura 4.5(b)); a divisão do *frame* em 9 quadrantes ao invés de 16 (Figura 4.5(c)) e a utilização de *frames* em cores, ao invés de tons de cinza.

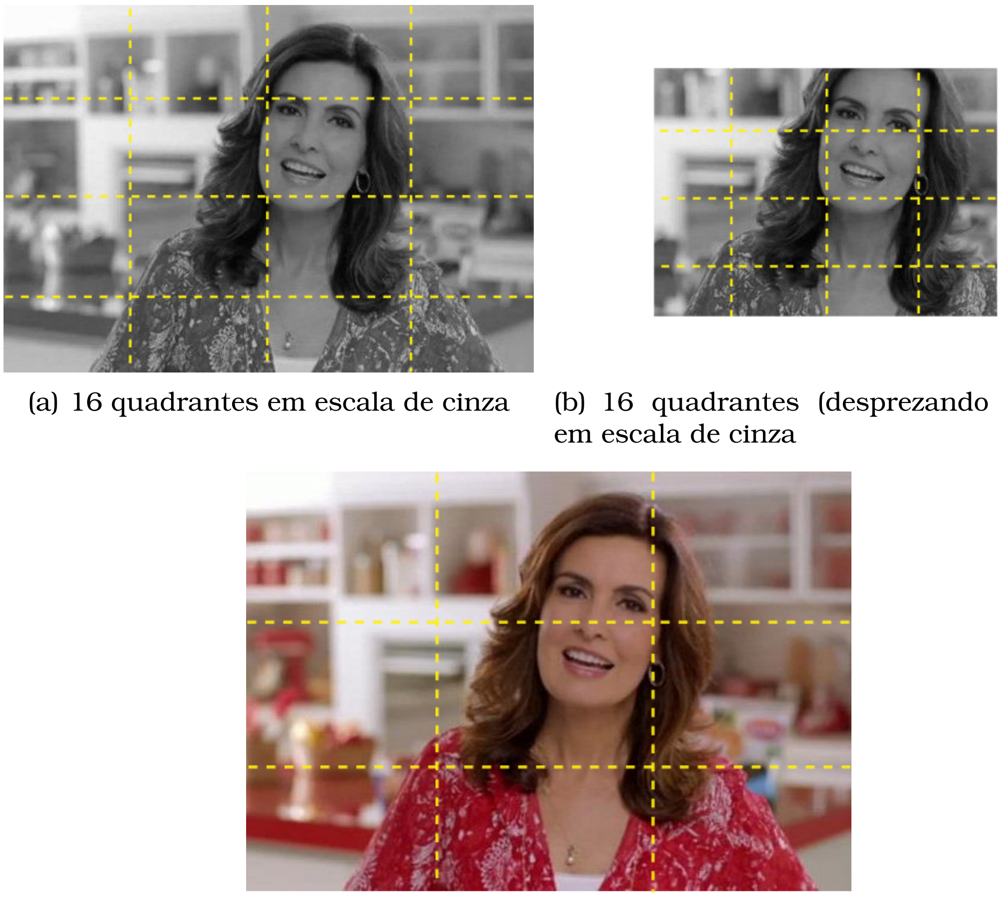
No caso da utilização dos *frames* em cores, 3 médias distintas são calculadas para cada quadrante, uma para cada um dos componentes Red, Green, Blue (RGB) que compõem os pixels. No caso de um *frame* dividido em 16 quadrantes, a métrica extraída de um quadro em cores seria composta por 48 números (16x3).

Construção da Árvore de Busca

Com o intuito de otimizar a busca dos *frames* e evitar a necessidade de comparar cada um dos *frames* do fluxo de vídeo contínuo com os *frames* da base de segmentos, os *frames* dos vídeos da base são organizados em uma árvore de busca.

Durante experimentos foi observado que a escala de cores utilizadas em programas de TV não possui uma amplitude muito elevada. Isso significa que a escala de cores (0 a 255) não é utilizada com uma distribuição homogênea. Um divisão na escala ao meio (128) poderia levar a um desbalanceamento da árvore, impactando a performance das buscas.

Para minimizar esse problema, é calculada a média geral para cada quadrante, considerando-se todos os *frames* de todos os vídeos da base. A média geral, que é utilizada para se construir a árvore pode ser calculada somando todos os vetores de todos os *frames* na base e dividindo-se o vetor resultante pelo número total de *frames*. Com isso se obtém um ponto da escala que


(a) 16 quadrantes em escala de cinza (b) 16 quadrantes (desprezando bordas) em escala de cinza

(c) 9 quadrantes (desprezando bordas) em cores

Figura 4.5: Divisão do *frame* de vídeo em quadrantes

otimiza o balanceamento da distribuição dos nós na árvore.

Os nós do primeiro nível da árvore (a raiz) contêm o valor da primeira posição do vetor de média geral, os do segundo nível contêm o valor da segunda posição do vetor e assim sucessivamente. Os *frames* da base são inseridos nos nós folhas.

O processo de inserção de um *frame* na base é baseado na comparação da métrica extraída daquele quando com os valores da média geral. O valor da primeira posição do vetor de métrica é comparado com o valor contido na raiz da árvore (que corresponde ao valor da primeira posição do vetor da média geral). Se for menor, o *frame* é enviado para o nó-filho à esquerda, se ele for maior, ele é enviado para o nó-filho à direita. No caso do valor ser igual ou muito próximo, ele é enviado para os dois nós-filhos. Nos nós do segundo nível o valor da segunda posição do vetor de métrica é comparado com o valor da segunda posição do vetor da média geral e é enviado para os nós-filhos de forma análoga ao que acontece com a raiz. Essa comparação dos valores continua até que as folhas da árvore sejam atingidas, onde então o *frame* é inserido, em um ou mais nós-folha.

É importante destacar que esse mecanismo de enviar o *frame* para os dois nós-filhos caso ele seja igual ou próximo do valor da média existe para garantir uma maior tolerância a flutuações no fluxo continuo de vídeo que é capturado, pois como se pode tratar de um fluxo enviado pelo ar (sem garantia de entrega), pequenas imperfeições no sinal recebido podem mudar o valor da média. Desta forma, mesmo que as flutuações façam um valor que deveria ser ligeiramente menor que a média se tornar maior, o algoritmo de busca ainda é capaz de encontrar o *frame* na árvore. Um outro efeito dessa abordagem consiste na possibilidade de suportar ligeiras diferenças nos contrastes de cores, comum em diferentes emissoras de TV.

Considerando que os valores de cada *pixel* podem variar entre 0 e 255, optou-se por utilizar uma tolerância de proximidade da média de 5 unidades, (cerca de 2%). Dessa forma, se a média geral para um dado quadrante for 100, *frames* cujo valor daquele quadrante estejam entre 95 e 105 serão enviados para os dois nós-filhos.

A Figura 4.6 ilustra o processo de inserção de um *frame* na árvore de busca. Como o primeiro valor da métrica extraída do frame é menor que a média geral, o frame é enviado para a esquerda. O segundo valor é maior que a média, então é enviado à direita. O terceiro e quarto valores são iguais ou próximos do valor da média, fazendo com que o *frame* seja enviado para ambos os nós-filhos.

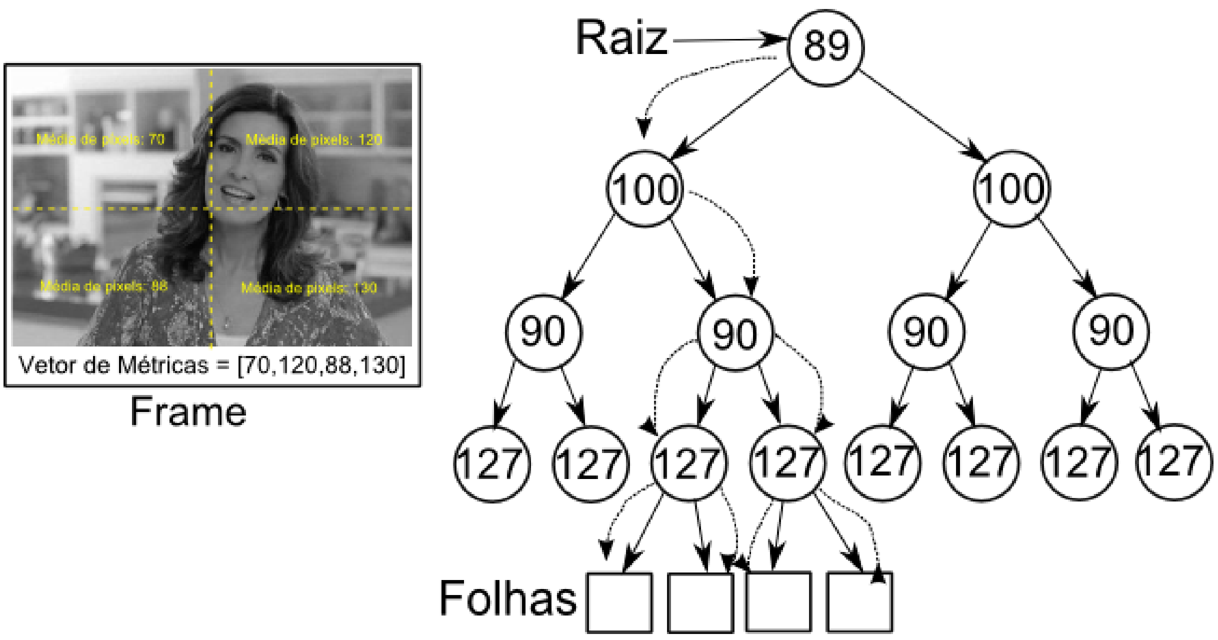


Figura 4.6: Montagem da Árvore de Busca

Por se tratar de uma árvore binária, o número de nós-folhas possíveis para a árvore é igual a 2 elevado ao tamanho do vetor da métrica extraída do *frame*. No exemplo está sendo utilizado apenas tons de cinza e os *frames* estão sendo divididos em 4 quadrantes, resultando num vetor de 4 posições e uma árvore com um total de 16 nós folhas.

No final do processo de inserção mostrado na Figura 4.6, o *frame* foi inserido em 4 dos 16 nós folhas possíveis. No pior caso, em que todos os valores de métrica sejam iguais ou próximos dos da média geral, o frame seria inserido em todos os 16 nós folhas.

Dentro dos nós folhas os *frames* são ainda divididos em duas listas. Uma das listas, chamada de "Lista de Início de Segmento", contém somente os *frames* que pertencem ao início de algum segmento de vídeo da base. A outra lista, chamada de "Lista de Final de Segmento", contém os demais *frames*. A separação dos *frames* em duas listas é feita para agilizar o processo de identificação dos segmentos, detalhado a seguir.

O limiar que define se um *frame* pertence ao início de um segmento ou ao final é uma variável do algoritmo que determina o quão tolerante a flutuações do fluxo de vídeo contínuo ele é. Por exemplo, se forem considerados como *frames* do início os pertencentes ao primeiro segundo de cada segmento de vídeo e no momento em que um determinado segmento estiver começando a ser transmitido houver uma flutuação muito brusca com duração de um segundo ou mesmo um corte do primeiro segundo do segmento devido a algum erro da emissora de TV, o segmento não será reconhecido. Se esse valor for muito grande, contudo, o algoritmo se tornará mais propício a encontrar falsos positivos, sobretudo se houver segmentos de vídeo com pouca duração (por exemplo, inferiores a 3 segundos) na base.

Identificação de Segmentos

Uma vez que a árvore esteja construída com todos os *frames* dos vídeos da base de segmentos inseridos nos nós-folha, pode-se iniciar o processo de identificação dos segmentos. Para isso um *frame* do fluxo de vídeo é capturado respeitando a mesma taxa de *frames* por segundo dos vídeos contidos na base. Repete-se o processo de extração da métrica para o *frame* capturado, resultando num vetor com os valores da média de *pixel* para cada quadrante.

Realiza-se uma busca na árvore, utilizando o mesmo mecanismo de inserção de *frames* de comparação dos valores da métrica extraída com a média geral. Como resultado da busca obtém-se um ou mais nós-folhas cuja métrica dos *frames* lá inseridos é semelhante à métrica do *frame* capturado do fluxo de vídeo contínuo.

Caso algum dos nós-folha encontrados possua *frames* na "Lista de Início de Segmento", os segmentos de vídeo correspondentes àqueles *frames* são adicionados à "Lista de Candidatos", referenciando o primeiro *frame* e contabilizando 1 ao número de sucessos e 0 ao número de falhas, mesmo que o *frame* reconhecido não tenha sido exatamente o primeiro.

É realizada também uma checagem se os *frames* localizados nos nós-folha podem ser a continuação de algum dos segmentos que já estavam presentes na "Lista de Candidatos". Para isso o algoritmo conta com uma janela deslizante de *frames* esperados.

A janela deslizante começa no *frame* corrente de cada candidato e se estende por certo número de *frames*. O tamanho da janela deslizante deve ser no mínimo do mesmo tamanho do limiar de início de segmento.

Sempre que houver pelo menos um *frame* nos nós-folha (independente se ele estiver na lista de início ou final de segmento) que esteja na janela deslizante de *frames* esperados de algum dos candidatos da lista de candidatos, o número de sucessos daquele candidato é incrementado. Caso contrário, o número de falhas daquele candidato é incrementado. Em ambos os casos, a referência ao *frame* corrente é incrementada, o que consequentemente faz a janela deslizante se deslocar.

Se o número de falhas de um determinado candidato se torna muito grande, o candidato é descartado e removido da "Lista de Candidatos". Caso o número de sucessos de um candidato atinja certo limiar, o segmento correspondente àquele candidato é reconhecido pelo algoritmo, e todos os candidatos referentes àquele segmento são removidos da "Lista de Candidatos".

Vale ressaltar que é possível haver várias entradas para um mesmo segmento na lista de candidatos, pois sempre que um *frame* capturado do fluxo contínuo chega em um nó-folha com *frames* na lista de início de segmento, um novo candidato é criado independente da existência de candidatos para aquele segmento na lista de candidatos.

Implementação e Testes

O algoritmo foi implementado em Python, porém a manipulação das imagens e vídeos é feita através da biblioteca OpenCV, que é implementada em C++ e oferece *bindings* para a linguagem Python. Dessa forma, todo o processamento de imagens é realizado através de código compilado em linguagem de máquina e somente a parte de controle e lógica do algoritmo é realizada em Python de forma a minimizar os efeitos da utilização de uma linguagem interpretada no desempenho do algoritmo.

Apesar de uma estrutura de árvore ser utilizada para armazenar os *frames*, na implementação utiliza-se uma estrutura de *hash* para se representar as folhas. A métrica extraída de cada frame é transformada em um ou mais números inteiros que são as chaves para o *hash*. Esse número é construído com base na navegação da árvore; se o frame é enviado para o filho à esquerda do nó raiz, é atribuído o valor "0"ao primeiro bit do número, se for enviado à direita, o bit recebe valor "1". No segundo nível da árvore, o segundo bit é atribuído de forma análoga. No caso do frame ser enviado para ambos os lados, são gerados dois números diferentes, um com o bit em "0"e outro com o bit em "1".

Ao invés de uma estrutura de *hash* para armazenar os *frames*, poderia-se utilizar um vetor, porém muitos dos nós folhas permaneceriam vazios e haveria um desperdício de memória ao se criar um vetor. Isso poderia ser solucionado por meio de um vetor esparso, porém a operação em um vetor esparso tende a ser mais lenta do que um *hash* alocado de forma dinâmica. A decisão de implementação está diretamente ligada às estruturas nativas da linguagem de programação escolhida.

A primeira versão do algoritmo considerava *frames* em tons de cinza (os *frames* coloridos capturados do fluxo de vídeo continuo eram convertidos pelo OpenCV para tons de cinza) e utilizava 16 quadrantes. Os testes iniciais com esse algoritmo mostraram que ele encontrava muitos falsos positivos, especialmente com segmentos de vídeo muitos claros, escuros ou estáticos.

Notou-se que em boa parte dos segmentos de vídeo os *pixels* próximos da borda do frame apresentavam pouca variação, sendo estáticos na maior parte do segmento. Para tentar melhorar o desempenho do algoritmo, passou-se a considerar apenas a parte central dos *frames* na hora de extrair a métrica. Com essa modificação foi possível a obtenção de resultados melhores, porém ainda eram encontrados muitos falsos positivos.

Passou-se então a utilizar os *frames* em cores. Nos testes iniciais, com uma base pequena, a estratégia de cores mostrou-se bastante eficaz e reduziu drasticamente o número de falsos positivos. Entretanto, quando foi utilizada uma base maior, observou-se o problema de estouro de memória durante a construção da árvore.

Como as métricas extraídas tinham 48 números (3 cores para cada um dos 16 quadrantes), a árvore poderia ter até 281.474.976.710.656 nós-folha e o número de bifurcações na árvore (situações em que o valor da métrica era igual ou próximo ao da média geral) era bastante elevado. Por exemplo, em um dos testes encontrou-se que um mesmo frame havia sido inserido em mais de 1000 nós-folha diferente. O tamanho da árvore ocasionou o estouro de memória, inviabilizando a sua utilização em larga escala.

Buscando solucionar o problema de memória foi reduzido o número de quadrantes para 9 e mantida a utilização das cores. Essa estratégia mostrou-se bastante eficaz e eficiente em relação ao uso de memória, entretanto foi a combinação da estratégia de considerar apenas a parte central dos *frames* com a estratégia de 9 quadrantes e em cores que produziu os melhores resultados nos testes realizados.

A Tabela 4.1 resume os resultados obtidos utilizando-se as diferentes estratégias de cálculo da métrica. Note que a linha "Memória Utilizada"expressa o tamanho da estrutura "Map"que armazena a árvore já montada. Durante a montagem da árvore muitas estruturas temporárias são armazenadas em memória (como os vetores de métricas) aumentando consideravelmente a memória gasta.

Foram utilizados para a composição da base um total de 222 amostras de vídeo capturadas de canais de TV aberta brasileiros (Globo, SBT e Band). Cada amostra refere-se a um comercial completo de TV ou vinheta de programa. No total foram considerados 74.251 *frames*, o que equivale a 4.950 segundos de vídeo (1:22:30). Os vídeos foram capturados a uma taxa de 15 fps.

O falso negativo obtido na configuração de 9 quadrantes com cores é uma vinheta com menos de 3 segundos de duração e bastante estática. Deste modo, pode-se concluir que, apesar do algoritmo com essa configuração ser eficiente e eficaz, ele ainda não é capaz de identificar certos tipos de trechos de vídeo. Outras técnicas podem ser combinadas a essa de maneira a aumentar a eficácia do algoritmo. Em todas as execuções o tempo de busca manteve-se em valores inferiores a 1ms.

Métrica	16q (cinza)	16q (cinza) sem bordas	16q (color)	9q (color)	9q (color) sem borda
Tempo para Cálculo das Médias (s)	370,19	204,86	629,87	637,44	334,09
Tempo para Montagem da Árvore (s)	5,39	6,42	ND	12,61	11,50
Número Máximo de folhas da Árvore	65.536	65.536	$2{,}81 \times 10^{14}$	134.217.729	134.217.729
Folhas da Árvore Ocupadas	34.504	46.842	ND	660.974	611.048
Maior número de frames encontrados em uma folha	777	928	ND	489	398
Memória Utilizada (kb)	156,9	172,1	ND	486,9	460
Falsos Positivos	16	10	ND	1	0
Falsos Negativos	2	1	ND	1	1

Tabela 4.1: Métricas obtidas nos testes com variações do algoritmo (q=quadrantes / ND = não disponível)

4.2 An Approach for Controlling Synchronous Remote Instances of a Multimedia Presentation

Keeping in touch and sharing experiences with family and friends about TV shows and other multimedia content are becoming a trend. The enjoyment of such shared activities depends on how synchronized the users' contents are. Many works propose architectures and algorithms to provide continuous media synchronization in order to enhance the quality of shared media consumption. However, in some applications, such as interactive multimedia presentations, the content synchronization depends on many factors, some of which can be affected by users' interactions. In these applications, in order to provide a synchronous shared experience among users, the interactions of one user should not affect only her presentation, but the presentation of all users which are engaged with her in the same activity. In this paper we propose an approach for engage users in synchronous shared experience for interactive multimedia presentations. Our approach is based on replicating the interactions of one specific user in all presentation instances, in order to keep the presentations synchronized. This means that a master user, like a pilot, controls not only her local presentation, but the presentations of all users. We present some challenges that need to be overcome in order to implement this approach and, for each challenge, we list some possible solutions and their pros and cons. We also present a proof-of-concept for NCL Documents and illustrate its use with highly-interactive multi-video presentations.

4.2.1 Introduction

The popularity of social networks, the spread of high-speed broadband connections, and the consolidation of video on the Web and Social TV are collaborating to transform media consumption into a synchronous shared experience(Geerts et al., 2011; Vaishnavi et al., 2011). Instead of watching media content alone in their own devices, users wish to engage in this activity with their families, friends or coworkers located elsewhere — and meanwhile they are watching, they are talking about the content, enriching their experience.

A common approach for allowing users share the media consumption experience is connecting two or more users through a text chat or video conferencing while they are watching the same content, such as a soccer match or a political debate (Weisz et al., 2007). This approach is enough to provide synchronous shared experience when the content is not interactive or its interactivity does not affects content's playback or its meaning.

However, when considering recorded interactive multimedia presentations — composed of several videos, images, audios and texts with temporal and spatial synchronization relationships, such as applications written in media synchronization languages such as SMIL or NCL — users usually do not only watch the content passively, but interact with it. For some classes of multimedia presentations, such as non-linear video, the content playback and therefore users' experience are dependent on user's interactions. In these applications, in order to provide a synchronous shared experience among users, the interactions of one user should not affect only his presentation, but the presentations of all users which are engaged with him in the same activity.

A recorded lecture is an appropriate example of interactive multimedia presentation to be consumed together. In order to better explain some subject, a lecture may be composed of one or more videos (teacher's face, teacher's body, whiteboard, complementary videos, etc), images, texts, animations, etc. The teacher and her students download the presentation into their devices and, at a previously scheduled time, they join in a video-conference. When the teacher is presenting a view of a specific moment of the lecture, in the role of pilot of the presentation, everybody (teacher and student) must see the same in their devices. If the teacher comments, via video-conference, the presented moment of the multimedia lecture, all users will be synchronized in that moment, ready to follow the explanation. If the teacher pauses her video, all students' videos should be paused in order to keep the synchronization with the teacher' presentation. The role of pilot may be passed temporally to a student. With this status, it is the student now who determines what everybody will see. This may be an opportunity for him to play some part of the lecture and ask related questions.

Other possible scenario is a guided virtual visit to an art exhibition. The virtual exhibition is represented by an interactive multimedia presentation, mimicking a real museum, with high-fidelity reproductions of pieces of art such as paintings and sculptures. Although users can explore the presentation by themselves, some might prefer being guided by an art expert, who can be a friend or a museum's employee. Visitors and the guide connect via a video-conference and the tour begins. Visitors' presentations follow the guide's presentation. When the guide is seeing a painting (and explaining via video-conference), all the visitors should be seeing the same painting -– otherwise they would become confused. Although the visitors' presentations follow the guide's one, they could still perform some interactions with the art works, such as rotate a sculpture or zooming in a painting.

The two scenarios described above are examples of applications in which, for synchronous shared experience, the interactions can affect the content's meaning. They also follow a pattern: only the interactions of one user (per time) with the presentation need to be replicated by other users' presentations. In this paper, this approach for live presentations synchronization (of recorded multimedia objects) is called *Remote Pilotage*, since one user controls or "pilots" all presentation instances. As the role of *pilot* may be hold by different users over the time, Remote Pilotage may be applied in a wide range of scenarios, providing synchronous shared experience with interactive multimedia presentations.

We describe the Remote Pilotage approach and discuss some challenges that need to be overcome to implement it. As a proof-of-concept, we extend a Nested Context Language (NCL) presentation machine in order to add the support for Remote Pilotage. We evaluate the proof-of-concept by using the Remote Pilotage approach to control multiple instances of a high-interactive multi-video presentation in different scenarios.

Section 4.2.2 presents related work. Section 4.2.3 describes in detail the Remote Pilotage model. Section 4.2.4 presents some challenges that must be overcome in order to implement the Remote Pilotage and some solutions. Section 4.2.5 presents the proof-of-concept implemented. In Section 4.2.6 we present the remarks and future works.

4.2.2 Related Work

The work of Repplinger et al. (Repplinger et al., 2009) adds to X3D support for distributed media playback. It enables the synchronous playback of continuous media into distributed X3D scenarios and allows simple media controls such as play, pause and choosing the media to be played.

In Boronat et al. (Boronat et al., 2012), the authors propose modifications in RTP/RTCP in order to allow media synchronization in different domains. Thus, two users watching a soccer match from different content providers, such as cable television and a video streamed over a 3G connection, would have their TV show synchronized, despite the difference in latency between content providers.

The works of Vaishnavi et al. (Vaishnavi et al., 2011) and Geerts et al. (Geerts et al., 2011) discuss human and technical aspects in order to provide a satisfactory experience on watching video or TV shows together over IP networks.

All the aforementioned works only consider distributed media synchronization for one media stream. They do not address complex multimedia presentation, in which the user interaction is part of the shared experience.

Mauve et al. (Mauve et al., 2004) and Fleury et al. (Fleury et al., 2010) present mechanisms to maintain the consistency in distributed applications. Mauve's paper addresses scenarios in which one application state is replicated to all other instances. The work of Fleury is more concerned with collaborative environment, in which all the users can affect the virtual environment simultaneously. Although these works present techniques for synchronous applications, the techniques are not discussed in the context of multimedia presentations.

In Viel et al. (Viel et al., 2011), the authors present a framework that allows user from interactive Digital TV (iDTV) and mobile platforms interact with remote instances of virtual reality (VR) applications. Techniques to reduce or fix possible inconsistencies between the remote VR application and the users' view are discussed. In Viel et al. (Viel et al., 2012) that work is extended by allowing the use of remote VR applications as a media in a multimedia presentation. However, these works do not address coherence among different media in a distributed multimedia presentation.

Neto et al. (Marques Neto & Santos, 2008) presents an approach in which broadcasters can control, during live TV shows, multimedia presentations in their clients' set-top-boxes (STB). That work can be viewed as an instantiation of the remote pilotage, but it lacks the generality of the approach proposed by this work.

In Boronat et al. (Boronat et al., 2009) the authors present a survey of techniques for multimedia synchronization, including scenarios in which the users' interactions can affect playback. However, that work does not address declarative synchronization languages.

4.2.3 Proposed Approach

A *Remote Pilotage Session* can be defined as a group of users interacting with the same multimedia presentation at the same time. Only one of the users, per time, is in the role of pilot of the presentation. The users might be related to each other, like friends in a social network or students of the same course. They might also be strangers with a common interest, such as fans of the same sport team.

Note that in a Remote Pilotage Session, the multimedia presentations watched by the users do not need to be strictly equal. They need only to be equivalent, which means they should be composed by the same (or equivalent) media and offer the same types of interactions in order to be controlled in a similar way. For instance, presentations developed for different platforms, such as iDTV and mobile devices, are different. Their medias are of different resolutions and may be presented in different spatial dispositions, but the presentations are considered equivalent if the content of their medias are similar and if they share the same interactions. Other example is a presentation with higher contrast to users with some visual impairment, which is different but equivalent to a presentation with normal contrast values. Still, different captions or audio streams may be adopted by equivalent presentations.

Users that engage in a Remote Pilotage Session can assume 3 different roles, as detailed in the UML User Case depicted in Figure 4.7.

The *Manager User* is responsible for creating the session (e.g. choose which multimedia presentation will be presented). He has also the responsibility to determine the moment in which the session begins and ends. When session begins, the *Manager* also holds the role of *Pilot*. He can promote one of the *Spectator Users* to a *Pilot user*, which makes him lose the *Pilot* role. However, he can always retake the *Pilot* role, which makes the previous *Pilot user* becomes a *Spectator user* again.

One and only one *Pilot* user per time, during the session, controls the presentation. The *Pilot* is the unique user that can fully interact with the presentation. His or her interactions are captured and sent to all others users' presentations. In fact, the *Pilot user* does not interact only with his or her presentation, but he or she is in the control (or piloting) of all the presentations. The *Pilot user* can also broadcast his or her webcam audio/video to the *Spectator users* and communicate via a text chat.

Besides the *Manager* and the *Pilot user*, all other users engaged in a session are *Spectators*. they have no control of their presentations; they just watch the result of *pilot user*'s interactions. *Spectators* can see the Pilot webcam image/audio (if available) and communicate with other users via text chat. They can also request to the *Manager user* the permission to become a *Pilot*.

Note that presentations may contains two different interactions categories, named as *Session Interactions* and *Local Interactions*. *Session Interactions* are interactions that affect all the session's presentations (e.g. video seek) and may be performed only by the *Pilot User*. On the other hand, *Local Interactions* only affect the presentation in which the user is locally interacting with (e.g. zooming in an image) and can be performed both by *Pilot* and *Spectator users*. The *pilot* user can enable and disable *local interactions*. The determination of which category an interaction is depends on the application and should be analyzed on a case by case basis.

Figure 4.8 depicts different scenarios in which the Remote Pilotage can be used. A basic scenario is 1-1, in which there are one *Pilot* and one *Spec-*

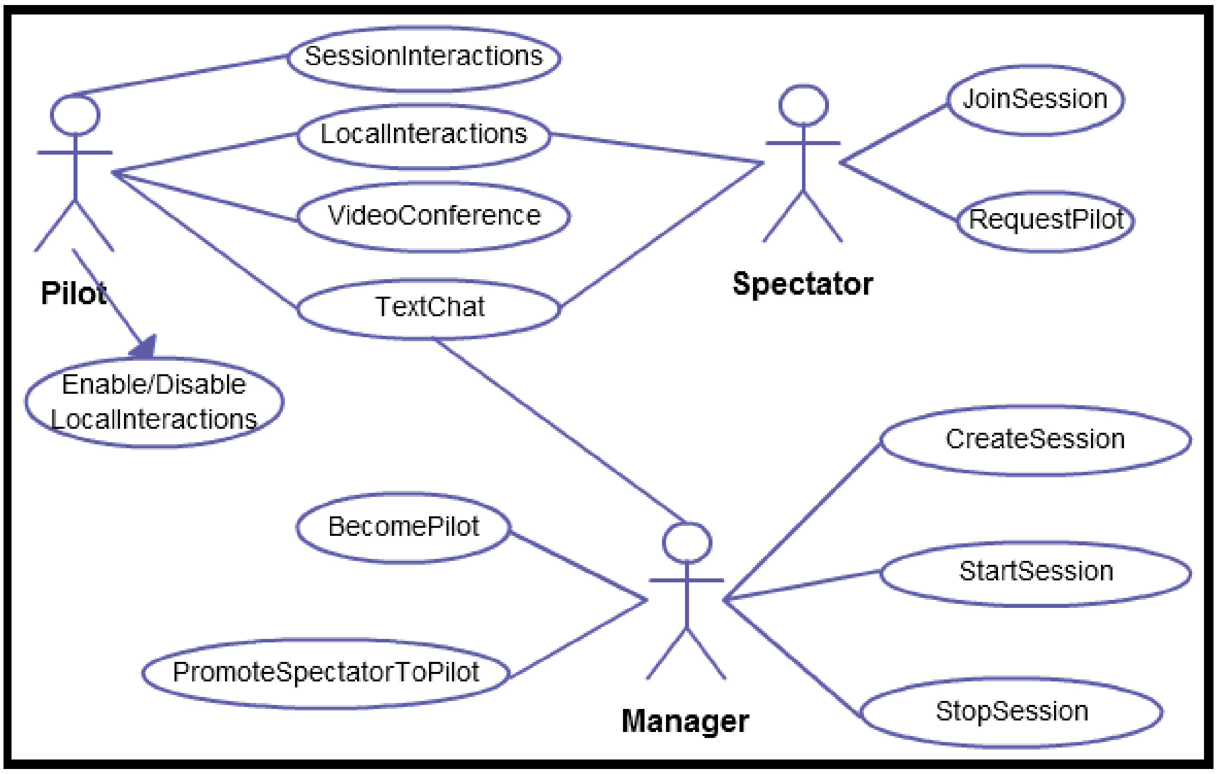


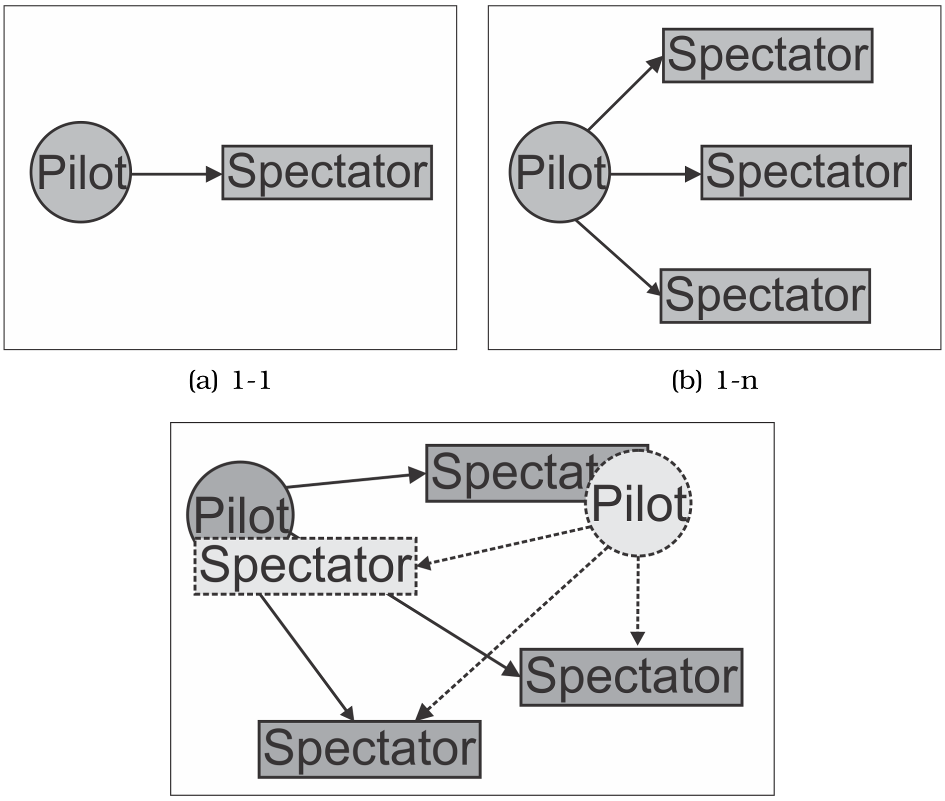
Figura 4.7: User Case Diagram

tator user (Figure 4.8(a)). In the 1-N scenario (Figure 4.8(b)) the *Pilot user* pilots many different *Spectator users*' presentations. In the N-N scenario (Figure 4.8(c)), the *Pilot user* changes over the time, allowing different users control all the presentations, but not at the same time.

4.2.4 Challenges

The Remote Pilotage has two main features: (i) the possibility of one user to control the presentation of all other users; and (ii) the users communicate in real-time with each other by a text chat or video conference. The former requires that continuous media, such as video, audio or animation be synchronized; and that media's spatial-temporal relationship, and the result of user's interactions, be consistent among all the presentation instances.

In order to implement (ii), it is necessary a *Low-Latency Communication Channel*, meanwhile (i) requires *Distributed Media Synchronization* and *Inter-media Synchronization*. There is also the necessity of allowing the *Later Join* in a remote Pilotage Session, e.g. allows a user to join in a session that has already started. The remaining of this section describes how these challenges can be overcome with technologies available at the date this paper was written.



(a) 1-1 (b) 1-n

(c) n-n

Figura 4.8: Scenarios

Low-Latency Communication Channel

For a low-latency communication channel with video or audio conference, it is necessary the use of low-delay audio and video codecs and decoder. G.279 and Speex are some of the low-delay encoders used in commercial Voice Over Internet Protocol (VOIP) applications. These codecs present low-bitrate and are good for voice encoding, however they present a poor audio quality. High-fidelity low-delay audio codec, such as Mpeg-4 Advanced Audio Coding Enhanced Low-Delay (AAC-ELD) (Schnell et al., 2008) and CELT[1] (Valin et al., 2009) could also be considered, however these codecs present higher bitrates.

For video, a common approach is to optimize broadly-adopted video codec for low-delay communication, usually with specialized hardware aid. For instance, the Mpeg-4 H.264 Advanced Video Coding (AVC) can be optimized for low-delay communication by reducing its internal buffers or avoid using B frames ((Viel et al., 2011; Kegel et al., 2012)). The coded audio and video must also be packaged and transported by some real-time protocol, such as Real-Time Transport Protocol (RTP) or Real-Time Streaming Protocol (RTSP) for IP networks.

[1]CELT is now part of the IETF Opus Codec http://opus-codec.org/

Another option is RTMP[2], which was developed by Macromedia and later acquired by Adobe to provide real-time communication between a Flash player and a media server. It supports audio, video and data streaming over IP network such as the Internet. It has also variations that include security, encrypted connections and packets encapsulated within HTTP requests to avoid firewall security.

A video conference during a remote pilotage session with hundreds of users, such as in Massive Open Online Courses (MOOCs), may become unfeasible. A hierarchical structure of tutors may be a solution. Students may be able to see and hear the teachers explanations, but they are allowed to interact only with the tutor designated to help the group in which they are included.

Distributed Media Synchronization

Continuous media, such as video, audio and animation, must be synchronized when multiple users are watching the content together. It means that the videos of each presentation instance of a remote pilotage session should be playing the same frame. Some works report that differences in video's playback up to 1 second do not impact on users' experience, but differences bigger than that should be corrected (Geerts et al., 2011; Vaishnavi et al., 2011).

There are three classics approaches to keep continuous medias synchronized: master–slave, sync-maestro and distributed control. In the master-slave, one presentation instance is elected as master and periodically broadcasts to the other (slaves) instances the current timestamps of its continuous medias. If a slave instance finds out that it is not synchronized with the master, it corrects itself. The sync-maestro approach gathers timestamps from all instances and calculates a guideline timestamp which is then broadcasted back to the instances. In the distributed control, all instances broadcast their timestamps and each instance chooses one instance to be synchronized with.

By using one of the three approaches, the presentation's instances can detect when they are desynchronized and need to correct themself. There are three ways to perform the re-sync: (i) time warp (Mauve et al., 2004), (ii) waiting sync and (iii) playback speed adjustment. In time warp, the instance instantaneously seeks the continuous medias to the correct time in order to synchronize with other instances. In the waiting sync, the most time-forward instances are paused until the late ones reach them. In playback speed adjustment, the speed playback of the continuous media are slightly accelerated or reduced until the medias become, gradually, synchronized again.

[2]Real-Time Messaging Protocol

The approach (i) is the easier to implement, but it may impact on users' experience. If the time adjustment is too big, the user may lose some important information in the presentation. The approach (ii) is a global approach and affects all the presentation instances. It may be very annoying to users if one instance lose synchronization and become later often. The approach (iii) may lead impacts on inter-media synchronization and the speed changes may impacts on users' experience.

Inter-media Synchronization

Multimedia presentations usually have spatial-temporal relationship among its medias. For instance, in SMIL or NCL it is possible to specify that when a continuous media reach a certain time, another media must be displayed in the screen. The relationship among medias is also affected by users interactions. For instance, in NCL it is possible to specify that when a button is pressed a media playback must be paused. Although the given examples are quite simple, the spatial-temporal relationship among medias may be very complex, involving many medias and conditions.

One approach that might be used in order to keep inter-media synchronization among different presentations instances is rely on the local presentation machines of each presentation instance. In this approach, hereinafter called **Local Synchronization**, only the unpredictable events (such as user interactions) are sent to spectators' instances.

Other possibility is disabling any formatting and scheduling functionality of the spectators' instances and uses the local presentation machines only as media players. In this approach, hereinafter called as **Centralized Maestro**, not only the unpredictable events are broadcasted from the pilot's instance, but all programmed actions that modify presentation, such as an start or stop action over a media.

The main drawback in **Local Synchronization** is that this approach is more likely to result in inconsistencies among the presentations, especially due to corrections of distributed media synchronization. Suppose, for example, a program that imposes to a video a resize when it reaches 20s. In a 5s time-forward spectator instance the video would reach 20s and would be resized, but the the correction to synchronize it with the other session's presentations, using the time wrap approach, would move the video to 15s but would keep its new dimensions.

In the **Centralized Maestro**, all media behavior and playback are controlled by a single presentation machine, so it is possible to avoid some inconsistencies caused by the local presentation machines. In addition, since the spectators' instances do not need to handle the inter-media synchronization by themselves, they can be simpler than the pilot's presentation machine. Note that this may prevent some users to become pilots.

One drawback of **Centralized Maestro** is that it may flood the network with lots of control messages. It also may be ineffective in scenarios in which the network delay is high. For instance, if the network delay between the pilot's and a spectator's machine is 1s, the control message that would make the video to be resized at the second 20, from the previous example, would only reach its destination in second 21.

A solution for this delay problem is the employment of the **local lag** technique (Mauve et al., 2004). In local lag, user's interactions performed locally are only resolved after a certain time; usually the estimated network latency. However, for time-based events, such as the video resize at second 20, it is not enough. It would only make the video be resized in the second 21 in both pilot and spectator's presentation. In order to perform the resize in second 20, it requires a prediction engine in the pilot's instance that send the control messages before the event occurs.

Later Join

For a user to join a remote pilotage session after it has already started, it is necessary to know the current state of the other presentations instances. The state of an interactive multimedia presentation can be associate to a node in a temporal graph, such as in SMIL State (Jansen & Bulterman, 2009) or NCL Eventline (Cerqueira Neto et al., 2012).

A possible solution to allow later join is to keep the current state of the presentation of the temporal graph in the pilot's instance. When a new user joins the presentation, the current state is informed to his or her presentation machine and, by the temporal graph, it restores the presentation state. The problem of this solution is that temporal graph tends to become very complex, especially in highly interactive presentations, and it is not a trivial task restore the presentation state from the graph's state in another presentation machine.

Instead of keep the current state, it is possible to store the events that lead to transitions in the temporal graph, as an event stream. When a new user joins the presentation, the event stream is sent to its presentation instance and it reproduces the events in order to achieve the current presentation state. Note that some optimizations in the event stream can be performed. For instance, if there are two seeks in a video, only the second one needs to be carried out.

Rather than consider the presentation state in whole, we can consider each media's state individually. Each media has a number of properties, such as spatial position, playback state (playing, paused, stopped, etc.) and, for continuous media, current playback time. It is possible, when a user joins the session, dump all the media's properties from the pilot's presentation and restore them into the new spectator instance.

Another solution is splitting the presentation into independent sub-presentations. A sub-presentation is not affect by any interaction performed by users in the other sub-presentations. The state in the temporal graph in which a sub-presentation starts is predictable and can be reproduced easily, no matter if it is a presentation that is already engaged in a remote pilotage session or a new presentation instance that just joined. These states are like milestones in the temporal graph. In this approach, when a new user wishes to join an already started session, it should wait until the beginning of next sub-presentation to engage the activity.

All the aforementioned solutions for later join can be implemented by the presentation machine itself and they work regardless of the multimedia presentation (although the sub-presentation approach expects a certain presentation organization). It is also possible to conceive application-bounded solution for later join. The presentation's state would be defined by a couple of well-known variables. When a new user joins the session, his or her presentation machine would receive the value of these variables and the associate application would restore the state. The application-bounded solution may be easy to be implemented for most multimedia presentation, however it requires imperative code. It may be tricky to implement in declarative synchronization languages, such as SMIL and NCL.

4.2.5 Proof-of-Concept

We have instantiated the Remote Pilotage approach for NCL documents. Our focuses in the proof-of-concept was the exploration of inter-media synchronization and later join. We have also implemented a low-latency instant messenger with support for text chat and video-conferencing based on Adobe Flash platform. Figure 4.9 depicts an overview of the proof-of-concept.

We have extended the WebNCL (Melo et al., 2012), a NCL presentation machine based on the Web-stack (HTML5/Javascript/CSS3). The extension consists of a new API, the Event Listener API, in which a function can be registered for listening to specific events that may occur in a NCL presentation, such as input events or media events (starting, pausing, etc.). We also extended the Interaction API (the API from which WebNCL receives input events, such as

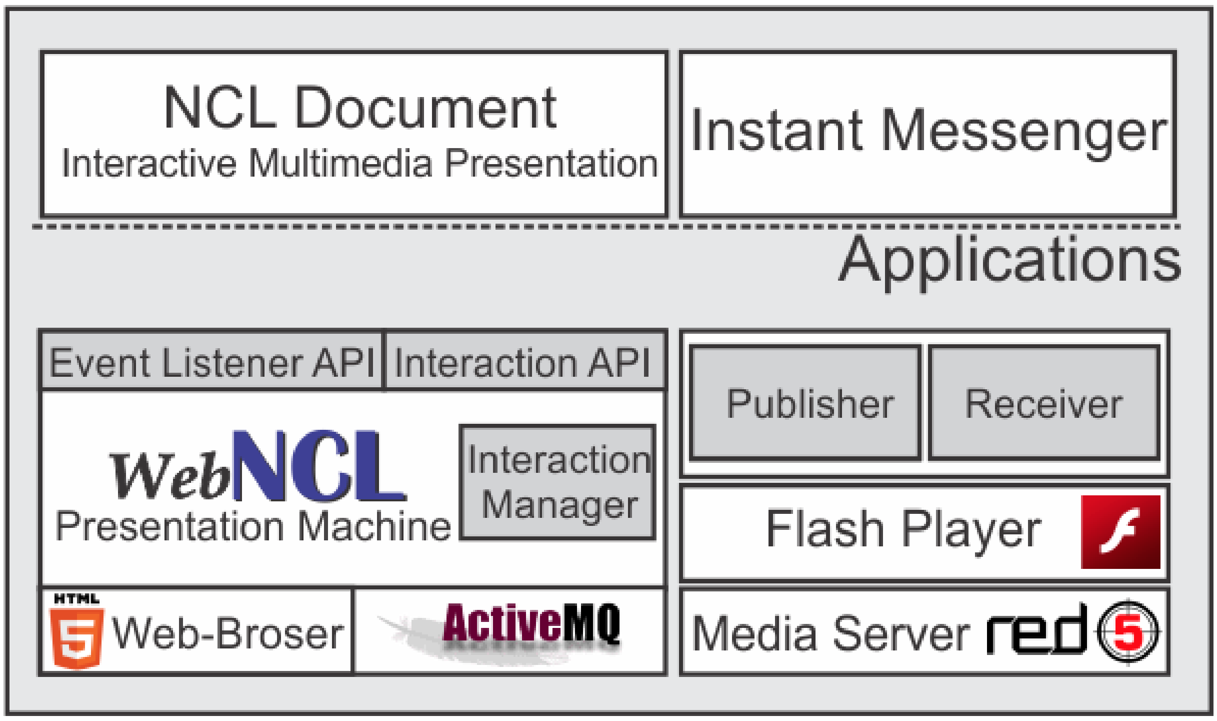


Figura 4.9: Proof-of-Concept Architecture

from virtual remote control) to add the passive mode. In this mode the user cannot interact with the presentation, e.g. events detected by the WebNCL's Interaction Manager are ignored.

Listing 4.1 illustrates the use of the extended WebNCL's API. In Lines 10-12 the WebNCL is instantiated; in Lines 13-17 a function for listening to *input* (such as focus changes and key pressed) and *presentation* (high level-events, such as the beginning and the end of the presentation) events is registered. In line 18, the passive mode is enabled in the WebNCL instance, which means the player will ignore any local user's interactions.

Listing 4.1: WebNCL's API using

```
1  <html>
2  <head>
3      <script type="text/javascript"
4              src="webncl.deb.js"></script>
5      ...
6  </head>
7  <body>
8      <div id="playerDiv"></div>
9      <script>
10         var player = new WebNclPlayer(
11                 "main.ncl",
                "playerDiv");
        player.addListener(
            function(evt) {
                console.log(evt);
            }, ['input','presentation']);
        };
        player.setPassiveMode(true);
    </script>
</body>
</html>
```

By using these APIs we implemented the Inter-Media synchronization, based on the Local Synchronization approach. Registers for input and presentation events are done in the pilot's instance. These events are encoded in JSON and sent to the spectators' presentations by means of an ActiveMQ [3] message broker via the stomp protocol over a websocket. The spectator's presentations are all in the passive mode, but when they receive the input events from the pilot's presentation via broker, they post events using WebNCL's input API.

The control messages are also encoded in JSON and sent via broker to all presentation machines. For example, when a spectator instance is promoted to pilot, we register a function for listening events in the new pilot, setting the passive mode to false. The previous pilot's presentation passive mode is set to true and it becomes a spectator instance.

We have implemented the later join using the sub-presentation approach. The NCL document must implicitly declare which are its sub-presentations. A sub-presentation must be an NCL <context> children from the <body> node with the anchor property *isMilestone* set as *true*, as illustrated in Listing 4.2. When a <context> node, which is a sub-presentation, starts, a presentation event is sent by the pilot's presentation. All the presentation instances, that are waiting, start when a sub-presentation begins. But instead of starting from the <body> context node, they start from the sub-presentation <context> node.

Listing 4.2: Sub-Presentation as <context> node

```
<ncl>
<body>
    <context id="subpresentation1">
        <property name="isMilestone"
            value="true" />
```

[3]Apache ActiveMQ - http://activemq.apache.org/

```
        ...
    </context>

    <context id="subpresentation2">
        <property name="isMilestone"
            value="true" />
        ...
    </context>
</ncl>
```

The instant communication is developed using flash components developed in the scope of Tidia-Ae project (Gaspar et al., 2009). The media server used in this proof-of-concept is the open source Red5 project [4].

The audio and video real-time communication of the instant messenger needs a publisher and one or more receivers. The connection between the publisher and a receiver is accomplished by the *streamID* created by the publisher when publishing a streaming. The publisher sends audio and video data encapsulated in RTMP protocol to the media server. The server is responsible to distribute data streams to each subscribed receiver. The developed proof-of-concept creates a combination of publisher and receiver, so both the pilot user and the spectator users can send and receive audio and video.

In our proof-of-concept, only the pilot user may publish his video at the beginning of the session. A spectator user must require the permission to publish his or her video. If a pilot or manager grants to the spectator user the permission for publish his or her video, then the entire session will receive the spectator user's stream. Text chat, however, is always available for all users.

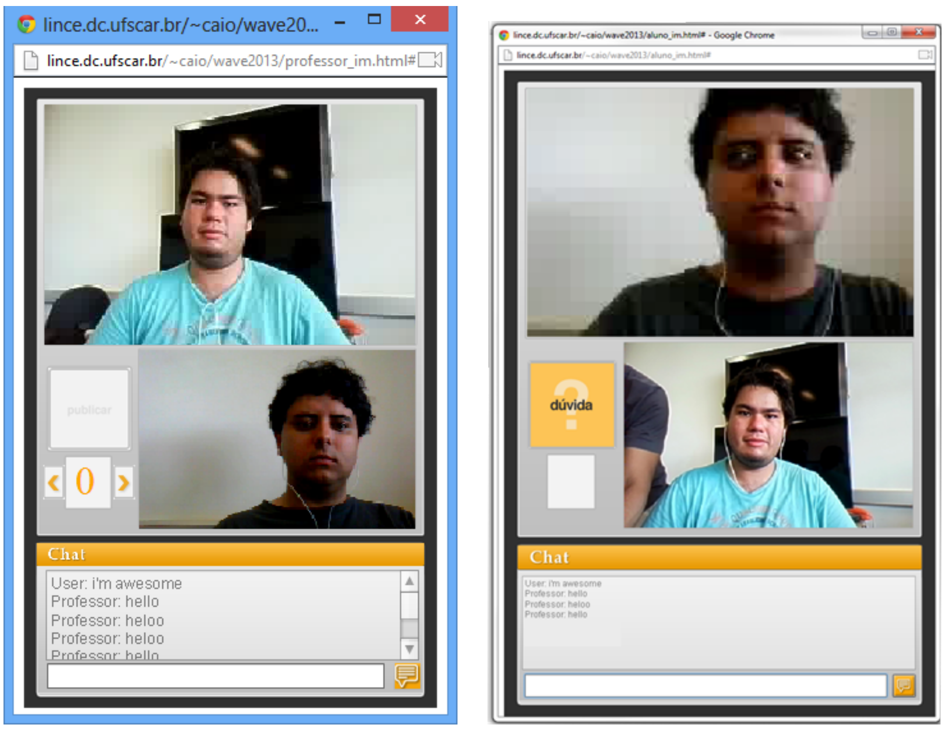
Figure 4.10 depicts two instances of the instant messenger interface when a spectator user is publishing his or her video. Figure 4.10(a) is the pilot view and Figure 4.10(b) is the spectator view.

Using the Remote Pilotage

We have used the Remote Pilotage in order to control complex multi-video and highly-interactive NCL document automatically generated from the capture of a lecture-style live-presentation. The capture and generation process of these presentation are described elsewhere (Viel et al., 2013b).

These multimedia presentations are compounded by four synchronized video streams. The user may select which video he or she wishes to see in

[4]Red5 Media Server - http://www.red5.org/


lince.dc.ufscar.br/~caio/wave20...
lince.dc.ufscar.br/~caio/wave2013/professor_im.html#
publicar
0
Chat
lince.dc.ufscar.br/~caio/wave2013/aluno_im.html# - Google Chrome
lince.dc.ufscar.br/~caio/wave2013/aluno_im.html#
dúvida
Chat

User: i'm awesome
Professor: hello
Professor: heloo
Professor: heloo
Professor: hello

User: i'm awesome
Professor: hello
Professor: heloo
Professor: hello

(a) Pilot User (b) Spectator User

Figura 4.10: Instant Messenger Interface

detail in a bigger window. He may also navigate in the presentation using the timeline or by points of interest, such as slide transitions.

Figure 4.11 shows a picture with two presentation instances. The presentation at the right (notebook) is the pilot instance and it is controlling the left one (desktop), which is a spectator instance. All the events performed in the pilot instance, such as choosing the main video or navigating in the presentation, were reproduced with fidelity in the spectator instance; keeping the presentations synchronized.

In order to validate the proof-of-concept prototype, we have evaluated: (i) the coherence among the multimedia presentation instances; (ii) how long an interaction performed in the pilot's instance takes to affect the spectators' instance; (iii) audio and video quality of the video conference; and (iv) the communication latency.

We tested the remote pilotage proof-of-concept in four different scenarios: (i) LAN with a local broker and media server; (ii) LAN with a remote broker and media server; (iii) users geographically distant with the broker and media server hosted in the pilot super node; and (iv) users geographically distant with the broker and media server hosted in a cloud.

In our tests, we have used computers with enough processing power to present the multi-video multimedia presentation smoothly (such as Core 2

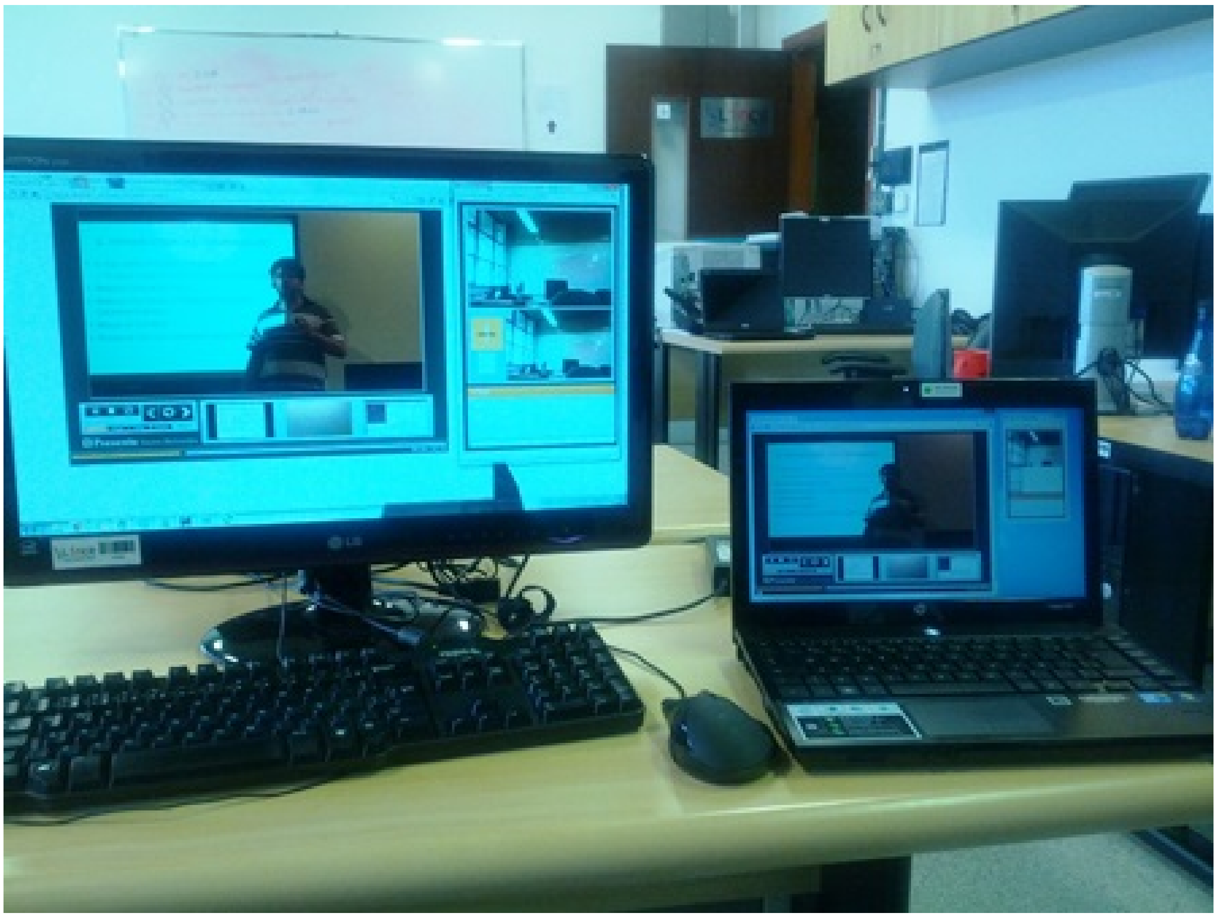


Figura 4.11: A Remote Pilotage Session

Duo 2.6 Ghz and 4 GB of memory or superior). They were also connected through a broadband network.

Table 4.2 summarizes the results. In scenarios (i) and (ii) we have used up to 5 computers running an instance each. In scenario (i) we have not seen any noticeable incoherence or interaction delay in the instances. We could set audio and video quality to high and we did not notice latency in communication. In scenario (ii), the instances were located in São Carlos - SP, Brazil and the broker and media server were located in an Amazon Server in Ireland. Although no coherence losses were noticed, a delay up to 500ms between the pilot and the spectators was observed. The quality of the audio and video was reduced to achieve a low latency in the video conference.

Scenarios (iii) and (iv) were tested with a node located in São Carlos - SP, Brazil, a node located in San Diego - CA, USA, and a node located in Corfu, Greece. In scenario (iii) the broker and the media server were located in Brazil, and the node in Brazil was the pilot. In Scenario (iv) the broker and media server were located in a Amazon Server in Ireland. We did not notice any coherence loss in both scenarios, but there was a delay up to 300 between the pilot and the spectators. The quality of the audio and video was reduced to achieve a low latency in the video conference.

4.2.6 Final Remarks

Synchronized distributed multimedia consumption is a facility which may enable the conception of interesting applications. When the interaction can affect the content's playback, only the synchronization of continuous media is not enough. The Remote Pilotage approach, presented and evaluated in this paper, is an effective contribution to reach a solution for such facility.

We presented a proof-of-concept that implements the Remote Pilotage model for NCL documents. The instant messenger is built using flash components. The inter-media synchronization is based on Local Synchronization approach and Later Join are enabled by Sub-Presentation strategy.

Scenario	Coherence	Interaction Delay	Quality	Com. Latency
LAN; servers local	Yes	Unnoticeable	High	Unnoticeable
LAN; servers remote	Yes	Yes, < 500ms	Low	Yes, < 500ms
Remote nodes; servers in pilot	Yes	Yes, < 300ms	Low	Yes, < 300ms
Remote nodes; severs in a cloud	Yes	Yes, < 300ms	Low	Yes, < 300ms

Tabela 4.2: Tests in Different Scenarios

In our tests the video conference presented a low latency, but at the cost of audio and video quality. Other technologies for real-time communication, in special the WebRTC API, a HTML5 emerging feature that enables video-conferencing in the Web without the aid of plugins, are going to be investigated in the continuity of this work.

In our study cases, we have noticed that users' communication, both by text chat and audio/video, really improves the experience when interacting with multimedia presentations. As a future work, we intend to investigate means for enriching the multimedia presentation with the interactions performed by the users during a remote pilotage sessions, as suggest by the works of Cattelan et. al. (Cattelan et al., 2008), Pimentel et. al. (Pimentel et al., 2007a) and Mullher and Ottman (Müller & Ottmann, 2000).

4.3 Multimedia Presentation Integrating Interactive Media Produced in Real Time with High Performance Processing

Sophisticated interactive animation may be an interesting element to compose a multimedia document. However, potential demand for high performance computing can make this option impractical in digital interactive TV and mobile device environments, due to computing power restrictions of these platforms. In previous work we proposed a solution to overcome such restrictions based on video streaming. Moving forward the solution, it is described in this paper how to take advantage of media-agnostic characteristic, when present in the multimedia presentation machine, to manage this new type of media in multimedia documents. As a proof of concept we extended and tested an NCL presentation engine to add suport to this new type of media.

4.3.1 Introduction

Multimedia presentations composed by different types of media can enrich the user experience when watching, listening and interacting with them. An interesting type of media to be considered is interactive animations with a high degree of realism and complexity, having visual and audio information following user's interactions. In this paper this type of media, wich usually requires real time and high performance processing to be produced, is called *High Performance Media (HPM)*.

One can imagine interesting applications of HPM objects in the TV environment. For example, consider a TV show based on an interactive expedition of a tourist spot, such as *Machu Picchu*. The application could be composed of a main video presenting an overview of the site and an HPM object representing a realistic 3D model of *Machu Picchu*. With the HPM object the user could explore areas and objects that were not explored or highlighted by the overview. The HPM object would be synchronized with the video of the overview. When the reporter move to another scenario, the application would move too. It would be even possible for the user to meet the news crew while navigating through the virtual environment. In addition, another media in the application, as high-resolution images or environment sound, could be added and synchronized with the user experience on HPM object. When the user enter a particular room, a specific song could start playing, for example.

Another interesting application could be a virtual contest between residents of different cities competing in a variety show. In addition to the audio and video of the program with a television presenter, participants and audience attendance, the application could also include an HPM object offering a virtual environment with competitions and events to viewers. Some viewers of the competing cities could be awarded with the opportunity to participate in the contest, competing from their homes, while all the others could follow the contest by choosing different visions of the virtual environment.

The framework RV-MTV was presented in previous work (Viel et al., 2011). It uses a strategy based on streaming video to allow devices with low computing power - smartphones, tablets or Set-Top Boxes (STB) [5] of DTV and Internet Protocol TV (IPTV) - to interact with applications running remotely. The applications run on high-performance rendering servers. Visual and audio outputs produced by the application are captured and encoded using compression techniques. They are sent on the fly to client devices via streaming through IP or air or cable broadcast.

Using the RV-MTV one can build applications that allow viewers from environments with restricted processing power, as DTV, mobile and IPTV environments, to interact with HPM objects executed remotely. Controlling the HPM object requires writing an application to send messages according to some specific protocol to the HPM object. Despite the possibility of interacting with an HPM object, it would not be possible with this model to integrate HPM objects into a multimedia presentation. An HPM object would be represented by a independent video, not related to other media and subject only to the mechanisms implemented specifically for its control.

[5]The STB term is used in the text to refer to both the set-top box for IDTV and to TVs which have embbeded the features of these set-top box

This paper presents a transparent solution to incorporate HPM objects into multimedia presentations. A dedicated player for HPM objects is proposed. The player performs the interface between the multimedia presentation machine and the HPM object, translating the instructions received from the presentation machine to messages for the HPM object. It is also proposed a way for mapping semantic concepts, common to declarative synchronization languages (anchors, links) in HPM objects. As a proof of concept, we extended the WebNCL (Melo et al., 2012), a presentation machine for Nested Context Language (NCL) (Soares et al., 2007), to add support to HPM objects. A multimedia presentation, integrating an HPM object generated by a graphic cluster, was used for testing purposes.

The remainder of this paper is organized as follows: section 2 presents some related works; section 3 defines the HPM object and discusses different strategies to enable them to run on devices with low computing power; section 4 presents the proposed mechanisms to aggregate HPM object to multimedia presentations; section 5 describes a proof of concept developed and finally section 6 presents conclusions and future works.

4.3.2 Related Work

Cesar, Jansen and Bulterman proposed in (César et al., 2006a) a way to enrich the multimedia content broadcasted in the context of IDTV. The proposal is to allow viewers to add their comments to the content transmitted by the broadcaster. The paper also discusses conceptually the incorporation of this facility in a declarative language, however it does not presents or make any reference to a proof of concept of the ideas.

Another approach to integrate HPM objects to multimedia presentations in the context of DTV, adopted in some works as in Dias et al (Souza et al., 2010), is to rely on local high performance computation. This approach may be a solution when the hardware, usually quite limited, of STB evolve to larger processing power, memory and graphics capability. This fact could lead to an increase in the cost of these devices and the need to rewrite applications for the languages supported by the embedded execution engines, which may not always be possible because of third-party resource dependencies, such as libraries and frameworks. A solution based only in local resources do not allow collaborative applications with HPM objects and remote users.

The strategy of remote execution of applications based on HPM objects is also treated in the work of Jurgelionis et al (Jurgelionis et al., 2009). In that paper the authors propose strategies for implementation, capture, transmission and remote control of video games, allowing computers with low computational power to interact with applications. In another related work (Viel et al., 2011), the authors extend this concept to IDTV and mobile devices and present a framework that enables one to interact with remote applications through a strategy of streaming video. In both articles, applications are not integrated into the context of multimedia presentations. They are only remotely controlled by commands built with imperative languages.

In the paper of Schmitz (Schmitz, 2002), the objective is also to enhance multimedia presentations defined by a declarative language. The approach proposes an extension of the language using a model that is able to manage hybrid objects, between animation and video. Thus, it combines the control of a video sequence with the flexibility of an animation, highlighting the importance of synchronization of user interactions in a multimedia environment. The solution is restricted to the web and, as the generation of the hybrid media, respecting user interaction, is performed locally, collaborative interventions between remote users over a shared media are not supported.

Rahman, Hossain and Lornav (Rahman et al., 2004) propose the inclusion of a 3D media in multimedia objects using a declarative language. However, the 3D media is treated standalone. The authors do not address its integration with other media on the composition of a multimedia object.

4.3.3 HPM and IDTV

The class of HPM objects defined in the context of multimedia systems refers to interactive applications that produce results perceived to the human senses, usually sight and/or hearing, of very high quality and realism. The results arise from the reaction to unpredictable external stimuli caused by human interaction, by natural events or generated by other applications. The results are consumed by humans and must be created with time constraints (short delay between stimulus and response) in order to keep the quality of experience (soft real time).

Due to its characteristics, HPM objects require high performance computers or clusters to run. Thus, environments with restricted processing and/or memory, such as smartphones, STB and tablets are not suitable to run HPM objects.

To make feasible the presentation of HPM objects in IDTV environment, two approaches may be adopted. The first one is the inclusion of high-performance graphics chipsets in STB, aiming the local processing of HPM objects. The second approach is to run HPM objects remotely on high performance clusters, and transmit the visual and audible results to the STB.

The approach of local processing may prevent the reuse of legacy HPM objects, developed for running on clusters or high performance computers - environments very flexible as development platforms - due to constraints like differences on the programming languages supported. It would not be possible, for example, the reuse of HPM objects written in C / C + + in an environment that only understand IDTV applications written in Java. Moreover, this approach would not be scalable as new types of HPM objects could be designed requiring more processing power than would be available, making chipsets obsolete.

With the approach of remote processing using clusters, the restriction to reuse legacy HPM objects no longer exist. In addition, it becomes easier to build collaborative applications, since the execution of HPM objects of different users can be held in the same cluster. However, this approach requires that visual and / or audible results from an HPM object execution be transmitted via data network to the STB, which can increase the delay in consumption of the results.

Considering the remote processing approach for HPM objects that return high quality visual experience to the users, one can use different ways to transmit the output produced to the STB: (1) transmitting uncompressed frames, (2) transmission of pre-rendered frames and (3) streamming of compressed digital video .

In (1) latency time would be reduced as there would be no delay due to the compression and decompression processes. However, transmitting high quality uncompressed frames would require a large bandwidth not compatible with the found nowadays on the Internet or terrestrial broadcasting systems.

When using pre-rendered frames (2), there would be a negligible increase in latency and a significant reduction in bandwidth needed to transmit the high quality visual information back to the viewer. However, this approach assumes that local machines have sufficient processing capabilities to complete the renderization process, which can not be true with many STB and mobile devices.

Streaming compressed digital video (3) allows the transmission of high quality video over a reasonable bandwidth network. STB and mobile devices usually already have hardware support to perform video decompression. However, the problem with this approach is the increased latency due to the time spent in processes of compressing and decompressing video.

There are studies in the literature that seek ways to reduce the latency of video encoding processes. In the Holub et. al. (Holub et al., 2006) high-definition videos, which are more costly to be encoded, are considered. It is also important to consider the increased response time introduced by running applications remotely, which may be critical for some classes of HPM objects. Barker and Shenoy (Barker & Shenoy, 2010) explore this issue through empirical analysis of the performance of collaborative applications running on cloud computing.

In this work it was adopted the approach of implementing HPM objects remotely in clusters and transmitting results via streaming compressed digital video. The framework RV-MTV, which is able to perform the transmission of objects with latencies as low as one second (Viel et al., 2011), was used for capturing the HPM object video output, encoding and transmitting the video. While applications like action video game, that require instant feedback, fit the definition of HPM, the proof of concept implemented in this work obtain better results with HPM classes of objects more tolerant to response time, like virtual tours in museums and those presented as example in Section 1. Strategies to minimize inconsistencies in HPM object that could prejudice the quality of experience, when the deadline for response is difficult to be fulfilled, are addressed in this paper in Section 4.

4.3.4 HPM as a Multimedia Presentation Component

Multimedia Objects, which define the behavior of multimedia presentations and interactions, can be produced using imperative languages such as Java or Lua, declarative languages, such as SMIL and NCL, or combinations of both, in which imperative scripting languages provide support to the declarative one, as NCL and Lua in the Ginga middleware (Soares et al., 2007) or HTML and JavaScript.

The combination, declarative language supported by imperative scripting language, is a good compromise since, at the same time it eases the specification of synchronism between media it also provides facilities to define specific situations that may not be adequately supported by the declarative language, but may be implemented with the scripting language.

For the integration of HPM objects to multimedia presentations, this paper explores the possibilities of a declarative environment with support of imperative language. Two alternatives may be considered. The first, referenced here as Imperative, is based on an API in the imperative language that offers facilities for HPM objects manipulation, such as synchronization, and communication and delay control. The second approach, referenced as Declarative, as the declarative language resources are extensively exploited, defines direct interactions between the object and the HPM presentation machine. In this case, the HPM objects are in sync with the rest of the media presentation. The advantages and disadvantages of each alternative are presented below.

Declarative vs. Imperative Approach

One advantage of the Imperative approach is the flexibility for future changes and specifications due to the ease of modifying the developed API to add new features. Another advantage of this approach is a better decoupling of the API implementation and the presentation machine, which do not need to be modified. Furthermore, an imperative language always gives greater freedom for software development.

However, the Imperative approach creates a distance between the presentation machine and the HPM objects. This gap hinders the integration and interaction of HPM objects with other supported media, because the management of the HPM object is no longer under control of the presentation machine. Furthermore, the development of multimedia presentations using the Imperative approach is more complex, as it requires greater skill in programming.

In contrast with the imperative approach, declarative implementation uses the concept of presentation integrated to media management, leading to a more cohesive and natural manipulation of the multimedia document. In this approach the HPM can be used as a conventional media (such as video or audio), with support to specific demands of HPM objects, such as the address of the remote machine and the communication channel configuration. Synchronization between HPM objects and other media from multimedia object with this approach is straightforward.

The integration of HPM objects to multimedia presentations is more coupled to the presentation machine in the declarative than in the imperative approach. Specifications of the HPM objects behavior can be carried out using the same semantics of the declarative language applied to other media, making use, for example, of interaction concepts common to multimedia declarative languages, such as anchors, regions, etc. In the imperative approach, delegating the handling of HPM objects to an external controller would require re-implementation of these concepts in a language which was not designed for this.

Considering a declarative language such as NCL, for example, which allow transparent addition of new types of media , one can add support for HPM objects without the need to change the grammar of the language (Soares & Barbosa, 2009) . As a result of the delegation of responsibility for interpretation of the new elements to the presentation machine, this approach strengthens the link between multimedia presentations (with HPM objects) and the presentation machine, since the the presentation machine must be extended to support HPM objects.

Integration Proposal

Due to the advantages of the Declarative approach presented before, that was the way we adopted for the integration of HPM objects to multimedia presentations, promoting a development model more suitable for multimedia contexts. It is necessary that the declarative language be media-agnostic, which means be able of handling and synchronizing transparently different media, without knowing the media. Examples of languages that presents such characteristic are NCL and SMIL (Bulterman et al., 2005) .

This class of languages uses transitions in the state machine of each media to manage synchronization. For example, synchronization points may be the beginning, end or pause of a specific media. Another intrinsic characteristic of these languages are the concepts of media properties and media anchors. The anchors refer to portions of media, such as a time interval in a video. The properties are features that can be accessed and changed during the presentation, for example, the volume of a music.

To promote the incorporation of HPM objects in multimedia presentations, some changes must be done in the presentation machine to support this new class of objects. It is necessary to map the concepts of properties, anchors and transitions of the state machine into HPM objects. The interactions between the presentation machine and the HPM object mus be encapsulated in control messages exchanged through the communication channel.

The event manipulation and transposition of an anchor type interface are illustrated in Figure 4.12. A syntax similar to NCL is use to exemplify the integration.

The anchor Paint1 is declared in item 1; item 2 defines its relationship with the HPM object via a link triggered in the anchor by the event onSelectionStart. The HPM object, defined in item 3 is then interpreted by the presentation machine that performs the necessary steps for synchronization, in this case pausing the return video, running the waiting resource and blocking the treatment of any other event until finishing the wait period. Then (item 4) passes to the remote application the event generated, along with control information (current timestamp) and triggers the response timeout. The remote application will apply the result of the event in the virtual environment (in this case modifying the current position of the user to the location of the anchor Paint1), thereby updating the return streaming video. The remote applicaltion also send an acknowledgment to the presentation machine, which resume the video removing it from the pause state.

Synchronization

Due to the delay inherent to the process of video encoding, decoding and transmission, the video presented to the viewer does not reflect the current state of the application running on the remote server. Although the synchronization between the states is not critical as HPM objects require only a sufficient synchronization to human perception, the latency in response time can affect the quality of experience. The upgrade of the presentation machine to integrate HPM objects must provide mechanisms to circumvent this problem.

To reduce the problem of states inconsistency, a synchronization mechanism between the state of the presentation and the state of the HPM object, using the clock of the involved machines (timestamp), is proposed. An initial calibration is required to synchronize the clock of local and remote machines. The calibration is repeated periodically to guarantee the sync. This synchronization does not need to be exact, tolerating differences not relevant to humans senses.

The application keeps, for each rendering iteration, the current state of its objects (eg user three-dimensional position and point of view). The presentation machine contributes with the sync process adding a timestamp in the messages when mapping events to messages and stop accepting new event requests until get confirmation message from the remote HPM object.

Further, to counterbalance the video coding/decoding delay, it is necessary that the remote application be able to evaluate the time spent in the process. Adding this time with the difference between the timestamps of control messages, the application can return back to the state the presentation was when sent the event.

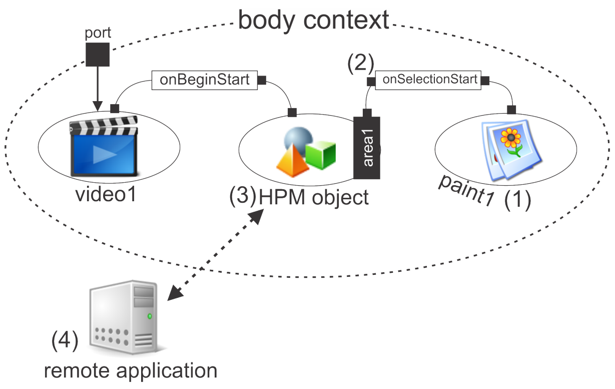


Figura 4.12: HPM Object in a multimedia presentation

There is also the need to address possible remote application freezes, discovered by the lack of response message. To do that, the extended machine triggers a timer waiting for a confirmation. In the event of a timeout, the presentation machine takes steps to alert the user, to reestablish communication or to exchange the remote server.

Application Manifest

To make simple the integration of HPM objects into multimedia presentations, we use an XML file to specify the characteristic of the HPM object. This file can hold information about the video stream address generated from the HPM object and the ways by which messages can be sent to the remote application that is generating the HPM object. It also lists the proprieties and anchors of the HPM object by which the multimedia presentation can interact with the HPM object. This Manifest File should be provide by the authors of the HPM object.

The Listing 4.3 contains a manifest file sample. In the lines 1 and 2 is defined the address of the video stream produced by the remote application. Between lines 3 and 8 the HPM object anchors and propriery are listed. Between lines 9 and 13 is specified how the communication with the remote application will be performed - in this case, via a message broker. Between lines 14 and 17 is specified the alternative content that will be displayed by the latency dissimulating mechanism.

4.3.5 Proof of Concept

Our proof of concept for the proposed integration of HPM objects into multimedia objects is composed of two parts. The first part is the extension of an NCL presentation machine to suport HPM objects as a new media type which was done by adding a new media player for HPM objects. The second part is building a distributed application, consisting of an application running on high-performance clusters that procudes HPM object and a NCL document which embedded this HPM object.

Adding an HPM Player to the presentation machine

The presentation machine we choose to extend was the WebNCL (Melo et al., 2012), an NCL presentation machine developed on top of web stack technologies (HTML/CSS /JavaScript) that allows the execution of NCL documents into HTML5 compatible browsers.

Listing 4.3: NCL Link

```
<manifest class="HPM">
  <media type="video"
    mrl="rtmp://mediaserver/streaming" />
  <anchors>
    <area name="PAINT_001" />
    <area name="PAINT_002" />
    <area name="PAINT_003" />
    <property name="speed" />
  </anchors>
  <communication protocol="stomp">
    <address brokerURI=
    "failover:(tcp://broker:61616)" />
    <topic dest="HPM_APP" />
  </communication>
  <holdOn>
    <waitingTrick type="image"
      mrl="loading.gif" />
  </holdOn>
</manifest>
```

The WebNCL's Event Manager is responsible for controlling the processing of NCL links. It is media-agnostic-media, which means it does not need to know if the media is an image or a video - or even non-standard type such as HPM objects - to perform the processing of events. The responsibility to inform the event manager about the transitions occurred in the media presentation that can represent conditions in NCL links - as beginning of an anchor - is delegated to players of each media. Also, the media players are responsible for receiving and processing the actions triggered by links - as modify a media property.

Because of this decoupling between the event manager and the media players in WebNCL, to add support for HPM objects in WebNCL we need to implement a new media player. This new media is responsible for playing and controlling HPM objects. This approach also does not require modifications in the presentation machine core.

Figure 4.13 depicts the WebNCL's architecture and details the HPM object player that was added to the presentation machine. The HPM player was inserted in the presentation layer, the same layer where the other media players are situated. It is composed of components responsible for to parse the manifest file (Manifest Parser), to handle and to present the video streamed by the remote application (Video Stream Player), to exchange messages with the remote application (Communication Module) and to synchronize and to perform dissimulation delay mechanisms (Synchronization Module).

The HPM object player receives video stream using RTMP (Real Time Messaging Protocol)[6] and display the video frames on the screen in the position defined by the presentation machine. Whenever an action event is triggered over an HPM object media node - as to start a certain anchor - the HPM player encodes that event in a JSON (JavaScript Object Notation) [7] message and sends it to the remote application.

In a similar way, when the remote application notices any transition that can be used as a condition to NCL links - such as HPM object's anchors starts - it sends a message to the HPM player. The HPM player decodes the message received and triggers the corresponding events to the presentation machine.

The HPM player also implements synchronization mechanisms to works around possible delay resulting from the video encoding strategy. When a user's stimulus can generate some inconsistencies in the presentation due to the difference between what he sees and the HPM object state in the cluster, the HPM player sends a message to the remote application with the video packets timestamp. The remote application uses the timestamp to retrieve the HPM object state for the moment that user's stimulus has been happened. In this state, the remote application processes the stimulus, producing the right visual output. While the cluster processes the user's stimulus and adjust the HPM object state, the HPM player can display some alternative content to dissimulate the latency.

[6]RTMP - http://www.adobe.com/devnet/rtmp.html
[7]JSON - http://www.json.org

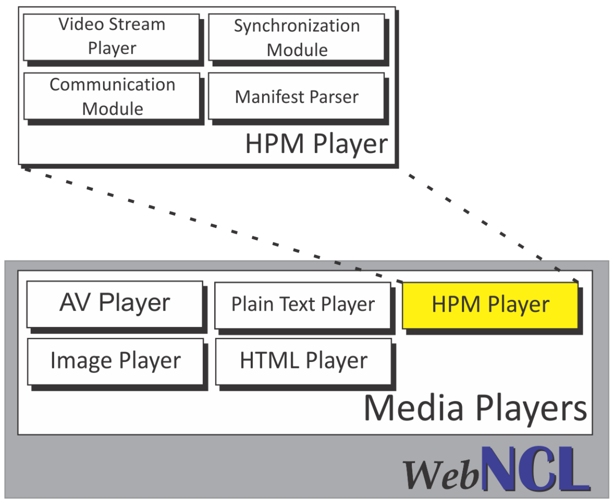


Figura 4.13: NCLweb and HPM Player Architecture

HPM object embedded in a NCL presentation

Figure 4.14 dipcts the infrastructure used for the integration of HPM objects with an NCL presentation.

The graphic cluster is an instantiation of a Cave Automatic Virtual Environment (CAVE) (Soares, 2005) called Mini CAVE (Dias et al., 2011). The graphics rendering applications is done in a distributed way. Thus complex environments can be presented in a high frame rate.

The running application was developed in C++ using the OGRE 3D [8], a graphic engine. It runs on a cluster of six nodes, equipped with Intel i7 processors and NVIDIA GPU.

The application responsible for generating the HPM object is an 3D modeling of a painting of Sonia Menna Barreto, a Brazilian artist, called *Torre de Papel*. The environment has some predefined animations and other paintings spread by the environment, such as an art exhibition of her works. Users can navigate in the virtual environment, using traditional devices (mouse, keyboard) and unconventional (mobile devices, Kinect and Wii Remote).

The application has been modified to incorporate the framework RV-MTV (Viel et al., 2011) . Using the framework, the generated frames are captured and encoded, generating a video stream that is sent to the HPM player.

The anchors of the HPM object were defined as certain positions within the 3D environment (the coordinates x, y, z and the camera angle by which the image is seen). Each anchor points to an environment location that faces one of the exhibited paintings. When the application receives a message from the HPM player informing an *play event* in a certain anchor, the application adjusts its display to show the painting associated with the anchor. One can also set the speed of predefined animations by modifying the property value

[8]OGRE 3D - http://www.ogre3d.org/

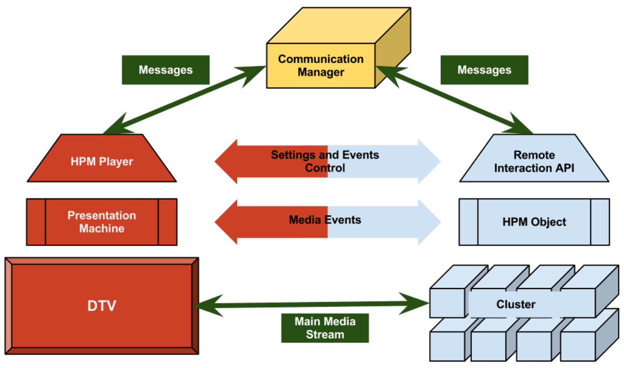


Figura 4.14: HPM Architecture

speed through events sent by HPM player.

Also, it was implemented a new interaction mechnism, allowing navigation through NCL application. If the user navigating in the environment, approaches to a painting associated to an anchor, the application will send an event to the HPM player, informing the start of that anchor. When the user leaves the area, an event indicating the end of an anchor will also be sent in an analogous manner. Figure 4.15 shows the application running on the cluster graph.

A module for controlling the messages exchanged between the HPM Player and the HPM object running on the cluster is important. We chose to use the ActiveMQ broker [9] to accomplish the management of these messages.

The advantages of a using a broker for message exchange are its reliability and security. Furthermore, as libraries for integration with ActiveMQ are present in many different languages, the construction of distributed applications on different platforms and languages, as occurs in this proof of concept, is facilitated.

We used the STOMP protocol (Simple Text Message Oriented Protocol) [10] to realize the communication between applications, encoding messages in JSON format. Streaming video produced by the application is sent to a remote media server. The media server then retransmits the video for the presentation machine using a multimedia streaming protocol (RTMP).

The NCL document which embed the HPM object allows users to navigate among the anchors through a menu. When the user is viewing any HPM anchor, a high-resolution image of the artwork is displayed. The user can also change the speed of the animations in HPM object using the colored buttons in the remote control.

[9]ActiveMQ - http://activemq.apache.org
[10]STOMP - http://stomp.github.com

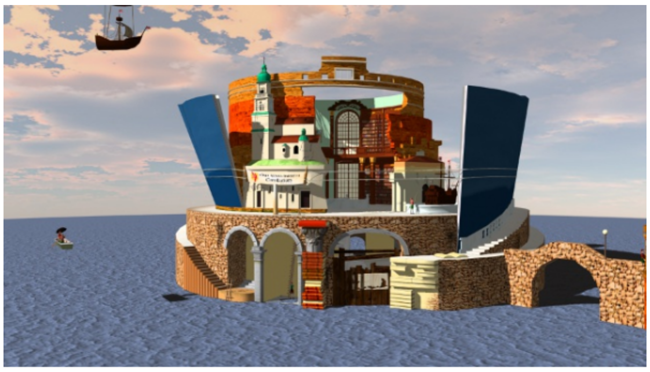


Figura 4.15: HPM object running.

Listing 4.4: NCL Link

```
<media id="interactivePaint"
    src="HPM_manifest.xml"
    descriptor="dInteractivePaint">
  <area id="paint1" text="PAINT_001" />
  <area id="paint2" text="PAINT_002" />
  <area id="paint3" text="PAINT_003" />
  <property name="speed" value="1.0" />
</media>

<link id="lGoPaint1"
    xconnector="onSelectionStopStart">
  <bind component="menuItem1"
    role="onSelection" />
  <bind component="interactivePaint"
    role="stop" />
  <bind component="interactivePaint"
    interface="paint1" role="start" />
</link>

<link id="lRedKey"
    xconnector="onKeySelecionSet">
  <bind component="interactivePaint"
    role="onSelection">
        <bindParam name="keyCode"
    value="RED"/>
  </bind>
  <bind component="interactivePaint"
    interface="speed" role="set">
        <bindParam name="setValue"
    value="1.5"/>
  </bind>
</link>

<link id="lShowPaint2"
    xconnector="onBeginStart">
  <bind component="interactivePaint"
    interface="paint2" role="onBegin" />
  <bind component="plainPaint2"
```

```
    role="start" />
</link>

<link id="lHidePaint2"
    xconnector="onEndStop">
  <bind component="interactivePaint"
    interface="paint2" role="onEnd" />
  <bind component="plainPaint2"
    role="stop" />
</link>
```

Listing 4.4 presents some excerpts from the NCL document. Between lines 1 and 8 an HPM media node is defined. The media's source is the HPM manifest (HPM_manifest.xml). It describes the possible properties, anchors and how to communicate with the HPM object (the address of streaming and other information that defines the object). Inside the media node are defined three anchors and a property.

The link lGoPaint1 (lines 10-18) defines an action that causes the HPM object to be moved to the anchor Paint1 when a particular menu icon is selected. The link lRedKey modify the value of HPM oject speed property to 1.5 when the red button is pressed, in other words, increasing the speed of the animations in 50 %.

The links lShowPaint2 (lines 34-40) and lHidePaint2 (lines 42-48) make the image plainPaint2, a high-resolution image of an artwork, be presented when the user is near to the equivalent painting in the HPM object.

Figure 4.16 shows the NCL document being presented. The user can interact with the application using the arrow keys to navigate between artworks from a side menu. When the user selects a menu item, the HPM object is moved to its anchor and a high resolution image of the artwork is displayed (Figure 4.17).

4.3.6 Final Remarks

The work presented in this paper confirms the feasibility of the proposed approach for embedding HPM objects in multimedia presentations. From a multimedia presentation platform with declarative language, complemented by imperative scripting language, interpretation capabilities, the approach adopted was to promote changes in the presentation machine so that the new media could be fully managed by the declarative part of the environment only. Thus, HPM objects can be better integrated with the multimedia presentation,

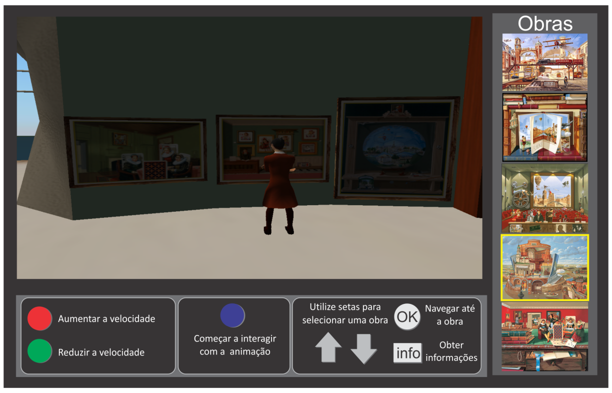


Figura 4.16: Navigation in the virtual environment

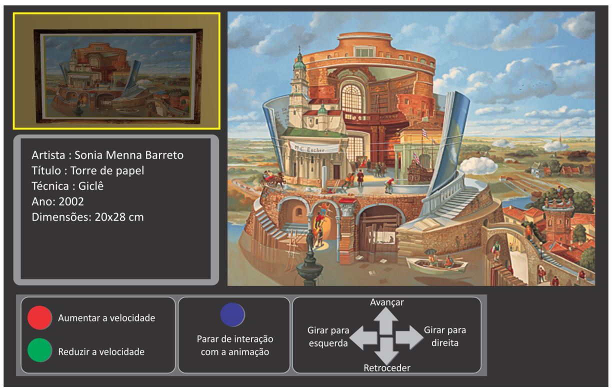

Artista : Sonia Menna Barreto
Título : Torre de papel
Técnica : Giclê
Ano: 2002
Dimensões: 20x28 cm

Figura 4.17: HPM synchronization with other media

as all its management is kept under the control of the presentation machine. It was also shown the feasibility of incorporating HPM objects on TVDi and mobile devices environments, using a strategy of video streaming.

It was presented as proof of concept the upgrade of a NCL presentation machine needed to enable it to run multimedia presentations that have HPM objects as one of their media. The new media is managed by the presentation machine generically, as it was an image or a video. Upgrading the presentation machine demanded no modification in the grammar of NCL as it is a media-agnostic language. A NCL multimedia application, including an HPM object generated remotely by a graphics cluster, was also developed and tested. This experience showed that creating a multimedia presentation with HPM objects is natural for developers who already have experience in authoring NCL documents.

The main issues that may be addressed in future works, in order to improve the proposed approach of embedding HPM objects in multimedia applications for IDTV and mobile devices environments, are related to the delay of encoding and streaming video. One can explore an approach based on the generation of pre-rendered frames at the cluster, leaving the final rendering to the local application. This process can be improved with embedded GPUs as OpenGL-ES [11] or even a web version like WebGL.

4.4 Considerações finais

A aplicações multimídia interativas e potencialmente atrativas podem ser construídas com a incorporação automática ou semiautomática, a uma mídia principal, de informações de contexto representadas em diferentes mídias. Nos estudos realizados observou-se também que a utilização de uma segunda tela pode ser um elemento facilitador da interatividade.

A integração de mídias de diversas naturezas em diferentes dispositivos introduz novos desafios à sincronização multimídia. Os mecanismos de sincronização investigados contribuem para o avanço do estado da arte, mostraram-se adequados para alguns propósitos:

- Execução de atividades colaborativas em apresentações multimídia, em que usuários geograficamente distribuídos podem compartilhar a experiência de uma interatividade colaborativa com uma apresentação multimídia

- Integração de mídias não convencionais, produzidas através de *clusters* computacionais a apresentações multimídia, permitindo, inclusive a definição de âncoras para a interatividade;

- A sincronização multimídia pode ser realizada através da extração de informações do conteúdo de cada mídia. Isso permite que novos horizontes de interatividade possam ser experimentados, já que as dificuldades pela não existência de um relógio comum e redes de transmissão heterogêneas podem ser superadas;

- Permitir o reúso de documentos multimídia, preservando os elementos de sincronização, mesmo em casos em que as mídias originais (como vídeo) sejam customizadas;

As pesquisas descritas neste capítulo apoiam o objetivo da tese ao promover facilitadores para que mídias possam ser incorporadas a uma mídia principal de maneira simples, mesmo que estas sejam produzidas em tempo real por *clusteres* computacionais ou que estejam atreladas a uma interação síncrona e colaborativa.

[11]OpenGL-ES - http://www.khronos.org/registry/gles/

CAPÍTULO

5

Ad-Avoidance Technology como catalisadora da interatividade na TV

O modelo de negócios de Televisão (TV) está em processo de fortes mudanças motivadas por evolução tecnológica. Após o grande impacto da televisão colorida de algumas décadas atrás, a TV passou por uma longa fase de estabilidade. A evolução das redes de computadores e da Web, o advento da TV Digital e do paradigma de aplicações em dispositivos móveis têm propiciado a exploração de novos modelos, com uma TV muito mais interativa e social (César & Geerts, 2011).

A interação no modelo da TV Digital passa a ser possível, pois o televisor digital incorpora um processador e os programas de TV podem enviar, junto com seus conteúdos tradicionais (shows, filmes, novelas, comerciais, etc.), programas de computação, também denominados aplicações. As aplicações promovem a interação.

Infelizmente, tanto no Brasil como em outras partes do mundo, a interatividade não tem sido promovida em larga escala pelas emissoras de TV no contexto da TV Digital. Além de problemas mercadológicos e do receio a mudanças da conservadora indústria de TV, as poucas experiências com aplicações interativa não têm obtido o êxito esperado.

Tipicamente o ato de "assistir TV"constitui um momento de entretenimento, em que o usuário se coloca em um estado de passividade, apenas consumindo conteúdo - daí a palavra tel*espectador*.

As aplicações interativas disponíveis, de maneira geral, trazem para a TV uma interatividade próxima daquela praticada na web. Mesmo quando aspectos de interação e usabilidade são tratados (tamanho de fontes, disposição de mídias, etc), o conteúdo é essencialmente tradicional e não acrescenta elementos inovadores capazes de estimular o usuário a sair do seu estado de passividade e interagir. Vale lembrar que o modelo de TV vigente vem sendo mantido desde o surgimento da TV, na década de 1920. Não se conhecem aplicações que sejam inovadoras e que realmente estimulam a interatividade na TV.

Essa situação motivou a busca por alternativas que pudessem alavancar a interatividade na TV. Os projetos de P&D (Seção 1.2), coordenados pelo autor da tese, tiveram como objetivo comum a busca do enriquecimento de conteúdo multimídia e por consequência uma melhor experiência para usuários finais. Nos projetos foi observado um potencial de aplicabilidade das pesquisas para a construção de aplicações interativas inovadoras com potencial de estímulo à interatividade.

Acredita-se que a possibilidade de promover interações com a TV através de dispositivos móveis, tendo a Web como meio de intermediação, ou através de TVs conectadas (smartvs), sejam boas opções para se alcançar grande parte do público. A disseminação de *smartphones* e *tablets* na população tem ocorrido de maneira bastante acelerada. Além disso, parece muito mais cômodo e menos invasivo que a interação seja feita de forma individual, cada telespectador com seu dispositivo móvel pessoal, ao invés de se usar um controle remoto único e ter o retorno da interação exibido na tela da TV, eventualmente incomodando outros telespectadores no mesmo ambiente. Diversas experiências, principalmente no contexto empresarial, têm sido realizadas nessa linha (Gondo & Sakamoto, 2011; Sarosi, 2011).

Observa-se também que é cada vez mais comum as pessoas assistirem TV acompanhadas dos seus dispositivos móveis e produzindo conteúdo atrelado aos programas de TV em redes sociais. Isso pode ser observado através dos *trending topics* to Twitter, nos quais é frequente a presença de comentários sobre programas de TV (Digital, 2012).

Brian Monahan, em “When It Comes to Ad Avoidance, the DVR Is Not the Problem” (Monahan, 2011) adverte que, para anunciantes, pior que os DVRs (“digital video recorders”) que permitem gravar o programa da TV e eliminar os comerciais, são os smartphones e tablets, que tiram a atenção do telespectador durante os comerciais, já que é um bom momento para o telespectador navegar na web, interagir em redes sociais, etc.

Dada a resistência dos fabricantes de TVs e *set-top-boxes* em facilitar o ingresso de novos *players* (desenvolvedores independentes de aplicações interativas) e a tendência de assistir TV acompanhado de um dispositivo móvel, foi proposto no escopo desta tese o projeto Kaeptor, que introduz um novo modelo de difusão de aplicações interativas. Esse modelo baseia-se na utilização do dispositivo móvel para a difusão de aplicações interativas sincronizadas com o conteúdo da TV, e integradas a redes sociais.

Foi proposta uma aplicação do tipo *ad-avoidance* como meio de promoção da interatividade. Entende-se como aplicação *ad-avoidance* aquela que faculta ao usuário (telespectador) a opção de evitar comerciais de TV.

A escolha desse tipo de aplicação foi resultado de estudos sobre o modelo de negócio da TV e na observação empírica de que a existência de blocos de comercial na TV traz um certo incômodo aos usuários. A Seção 5.2 apresenta a fundamentação teórica, difundida na literatura, que confirma essa hipótese.

A aplicação desenvolvida baseia-se em um paradoxo: se por um lado os telespectadores não estão satisfeitos com o modelo de ter um programa de TV interrompido para a exibição de blocos comerciais, por outro lado a sustentabilidade do modelo da TV (principalmente TV aberta) depende do conteúdo publicitário para custear a produção dos programas.

O modelo de programação baseada em "blocos de comercial"(também chamados de *breaks*, na linguagem de TV) pode ser uma razão para acentuar o descontentamento do usuário. Por questões de custo o anunciante é compelido a despejar uma "avalanche"de conteúdo em um curto espaço de tempo de um anúncio de TV(tipicamente 15 a 30 segundos). Isso dificulta a fruição do ato de "assistir TV".

Muitas vezes o comercial veiculado é algo que interessa ao usuário, mas ele não tem a oportunidade e o tempo adequados para absorver efetivamente o que foi apresentado: ex: preços de um produto em promoção, validade da oferta, um website com mais informações, etc.

A aplicação proposta, baseada nesse paradoxo, apresenta duas características contraditórias mas complementares, indo além da funcionalidade de *ad-avoidance*:

1. Permite ao usuário evitar os comerciais de TV. A aplicação se propõe a oferecer a possibilidade do usuário ser informado quando a programação de um determinado canal voltou. Isto é, a aplicação se propõe a avisar ao usuário quando um bloco de comerciais termina. Com isso o usuário tem a liberdade para realizar outras atividades durante o tempo "inútil"dos comerciais: apertar a tecla *mute*, trocar de canal, realizar atividades domésticas, etc. A proposta é que essas atividades possam ser executadas com naturalidade, sem a preocupação de ter a fruição do programa de TV prejudicada (como perder uma cena importante em um

filme após o retorno dos comerciais). É dada a liberdade ao usuário de optar por assistir ou não comerciais.

2. Permite que o usuário possa consumir os comerciais veiculados na TV de maneira mais confortável. Através da aplicação o usuário tem a possibilidade de visualizar detalhes do comercial, como preços, datas e outras informações relacionadas ao comercial apresentado na TV. O conteúdo fica disponível ao usuário de maneira sincronizada à programação da TV - assim que o comercial é veiculado na TV ele é disponibilizado ao usuário no seu dispositivo pessoal. A aplicação permite que o usuário possa navegar por um comercial no tempo que achar necessário, bem como marcá-lo para visualização futura.

Hoje já é comum, principalmente entre jovens, assistir TV e ouvir música ou participar ao mesmo tempo de redes sociais utilizando notebooks, tablets ou smartphones (Wallis, 2006; Tokan et al., 2011). A aplicação, que baseia-se na companhia do smartphone enquanto se assiste TV, alinha-se a essa tendência.

A Seção 5.2 apresenta a formalização da hipótese que motivou o desenvolvimento da aplicação.

A arquitetura da aplicação, os desafios científicos-tecnológicos impostos para sua construção e detalhes sobre o desenvolvimento da aplicação estão descritos na Seção 5.3.

Considerações finais sobre as contribuições desta aplicação aos objetivos da tese são tecidas na Seção 5.4.

5.1 Hipótese

"A utilização de uma Ad-Avoidance Tehcnology (AAT) adequada pode promover a TV Interativa." Essa AAT deve ter as seguintes características:

1. Estar à mão do usuário, preferencialmente no seu dispositivo pessoal;

2. Induzir a interação em tempo real;

3. Promover uma mudança cultural, criando o hábito de interação com a TV;

4. Induzir o usuário a interagir com outras aplicações pela comodidade e facilidade de acesso, já que a opção interação estará à mão.

A hipótese levantada foi explorada através do desenvolvimento de um conjunto de ferramentas que permite a realização de experimentos e medições com vistas à comprovação da hipótese.

5.2 Modelos de Negócio da TV comercial: conceitos e conflitos

A exploração da difusão/distribuição de programas de TV pela iniciativa privada, sendo ou não uma concessão estatal, é movida ao lucro. O modelo de negócios da TV é um exemplo típico de "two-sided market". O termo "two-sided"ou de maneira mais geral "multi-sided markets"é utilizado para definir mercados em que uma ou várias plataformas viabilizam a interação entre usuários procurando manter os dois (ou mais lados) satisfeitos no negócio. Através de tarifação adequada para cada lado e reconhecendo quanto cada lado se beneficia ou sofre com a interação, o intermediário busca maximizar o seu lucro (Rochet & Tirole, 2003; Bagwell, 2007; Ginsburgh & Throsby, 2006). No caso da TV, o que caracteriza o "two-sided market"é que a emissora vende dois tipos de produtos, o programa para os telespectadores e a audiência para os anunciantes (Ginsburgh & Throsby, 2006).

Outro conceito importante para se entender o modelo de negócios de TV é o de "network externality". Esse conceito tem se tornado popular na literatura econômica e jurídica desde meados da década de 80. A ideia fundamental é que o ato de se aderir a uma "network"resulta benefício a todos os demais integrantes da "network"(Page & Lopatka, 1999). Mais especificamente, "network externality"é definida como a alteração em benefícios que um agente obtém pelo consumo de um produto em função da variação do número de outros agentes consumindo o mesmo produto (Liebowitz & Margolis, 1998). As externalidades ("externalities") podem ser positivas ou negativas. Um caso típico é a adesão a um plano de telefonia em que o assinante pode falar gratuitamente com todos os demais assinantes que também aderiram ao plano da mesma operadora. A adesão ao plano de pessoas com quem o assinante se comunica frequentemente resulta em externalidade positiva. A externalidade negativa acontece em situação inversa, a desvinculação ao plano dessas pessoas.

No caso da TV, um "two-sided market", uma externalidade positiva flui dos telespectadores aos anunciantes. Da mesma forma, telespectadores podem considerar os anúncios como um incômodo, e se o custo do incômodo supera os benefícios que estejam associados com o anúncio, uma externalidade negativa flui a partir dos anunciantes para os telespectadores (Bagwell, 2007).

Fluxos de externalidades positivas e negativas podem ocorrer também entre telespectadores com atuação de anunciantes e emissoras. A grande audiência a um programa, e principalmente aos comerciais, podem, em tese, resultar em externalidades positivas, traduzidas por melhoria do programa, melhoria da qualidade dos comerciais, diminuição da quantidade de comerciais.

Em um "two-sided market"o lucro do intermediário pode advir de apenas um ou de ambos (múltiplos) os lados. Na TV aberta a emissora é financiada apenas pelos anunciantes. Na TV paga, a emissora além de cobrar dos assinantes, pode sustentar seu negócio também com anúncios pagos, auferindo remuneração, portanto, dos dois lados. A dupla possibilidade de remuneração da TV paga permite à operadora promover maior diferenciação de conteúdo. Já a emissora aberta tem mais argumentos para inserção de maior quantidade de comerciais durante a programação (Peitz & Valletti, 2008).

O fato é que uma emissora financiada exclusivamente por assinantes deve se preocupar exclusivamente com as preferências de seus assinantes. Os programas devem ter a qualidade esperada (comprada) pelos assinantes. Entretanto, o caso das emissoras/operadoras financiadas por anunciantes é bem diferente. Pesquisas sugerem que as preferências dos anunciantes influenciam mais fortemente a emissora do que as preferências dos telespectadores (Wilbur, 2008). O resultado prático dessa política é que o interesse da emissora é manter o telespectador para que ele consuma o anúncio e não o programa. Em "The Economic Analysis of Advertising"(Bagwell, 2007), Kyle Bagwell destaca: *"The broadcast company must then ensure that consumers stay on board, by bundling the ads with entertainment"*.

Simon P. Andersone e Jean J. Gabszewicz cunham em"The media and advertising: a tale of two-side markets": *"Entertainment and content are the bait to get prospective purchasers of consumer goods to be exposed to advertisements"* (Anderson & Gabszewicz, 2006).

No processo de maximização de lucros é comum emissoras se excederem na quantidade de comerciais apresentados por hora de programação, entre outras medidas prejudiciais ao público. Estudos procuram identificar distorções de mercado e eventuais necessidades de intervenção governamental (Choi, 2006).

O telespectador não fica passivo a abusos - ele tem suas armas: tecnologias para evitar comerciais - AATs (Ad-AvoidanceTechnologies). A tecnologia mais básica para evitar comercial é sair da sala, ir ao banheiro, buscar outra atividade... Talvez a mais utilizada seja o controle remoto. Durante o intervalo é o momento comum para vasculhar o que se passa em outros canais ("zapping") ou apenas colocar o som em "mute". O DVR ("digital video recorder)

TiVo[1], um dos líderes de mercado em AATs, dispõe de tecnologia que permite retirar os comerciais de programas gravados. Outras tecnologias descritas na literatura (Hua et al., 2005) utilizam-se, para remoção de comercias, de padrões estabelecidos em determinados países para inserção de comerciais, como delimitação por frames pretos ou brancos e redução do som.

Diversos estudos mostram que durante a programação e, principalmente, nos intervalos comerciais, o telespectador deixa temporariamente o ambiente da TV ou muda de canal (Zhou, 2004; Tse & Lee, 2001; Rojas-Mendez et al., 2009; Brennan & Syn, 2001; Danaher, 1995). A mudança de canal é também fortemente estimulada pelo aumento de quantidade de canais que se observa, com operadoras oferecendo na ordem de uma centena, ou mais, de canais.

A referência a Knealy, D., 1988, 'Zapping of TV Ads Appears Pervasive,' Wall Street Journal, April 25, p. 30', apresentada em (Zhou, 2004), mostra um impacto interessante do modelo de blocos comerciais de TV:

> "In 1952, the water commissioner in Toledo, Ohio, noticed that whenever I Love Lucy was shown on the city's only television station, each advertising break was marked by a huge drop in water pressure as thousands of toilets flushed at once."

Estudos mostram ainda que muitos telespectadores sequer esperam o início do comercial para trocarem de canal (Kent, 2013). O descontentamento dos consumidores com anúncios na TV é tal que 42% dos telespectadores entrevistados em pesquisa realizada por Greg Stuart (You Can't Avoid Ad Avoidance (Stuart, 2008)) estariam dispostos a pagar 20 dólares a mais por mês para evitar os anúncios (isso além do que já pagam de assinatura de canais).

Do ponto de vista de um anunciante, não é o tamanho da audiência durante os programas que é importante, mas sim o tamanho da audiência durante os intervalos de comerciais. Assim, para proteger seus interesses econômicos as emissoras procuram adotar estratégias contra AATs. Uma prática é a sincronização de intervalos comerciais entre diferentes emissoras, principalmente quando o mesmo tipo de programa esteja sendo transmitido simultaneamente entre diferentes canais (Zhou, 2004). Outra proposta, veiculada como matéria jornalística "The End of the 2-Minute Commercial Break" por Daniel Frankel (Frankel, 2009) sugere a redução dos intervalos comerciais para 30s e a utilização dos próprios atores do programa, quando for o caso, para realizar o comercial, de maneira a não se perceber muita quebra na sequência do programa. Nos comentários a essa matéria os leitores manifestam, em sua ampla

[1]http://www.tivo.com/what-is-tivo/tivo-is/index.html

maioria, grande desconforto a comerciais de TV. Em relação a intervalos menores, mas com maior frequência destaca-se a seguinte observação de uma leitora:

> "What I passionately hate is not the length of commercials but the frequency. Because it is the most gut wrenching feeling when you are so focused on a show and then a commercial comes and breaks that focus. If there were just two breaks, one in the middle of a show and one at the end, that would be perfect. But a break every 10 minutes....I just can't enjoy a show!

Alguns trabalhos alertam que a prática de se evitar comerciais na TV pode levar a um desequilíbrio no sistema resultando em produções muito piores e volume ainda maior de publicidade.

- "As the media market is a two-sided market, increased ad-avoidance reduces advertisers' value of placing an ad." (Stühmeier & Wenzel, 2011)
- "...viewer purchases of devices to avoid ads may cause a disproportionate share of the ad nuisance to fall on the remaining audience. As these are views less adverse to ads, this causes broadcasters to increase advertising levels. This result is in line with observed facts. The bypass option may cause total welfare to fall. We demonstrate that higher penetration of such technologies may cause program content to be of lower quality."(Anderson & Gans, 2006)
- "When AAT becomes available, consumers who purchase AAT gain by avoiding ads; the remaining consumers suffer through more advertisements and reduced content. Advertisers reach fewer viewers but at a lower cost. Content provider profits suffer greatly.... AAT penetration, in the case of television, reduces niche program availability and overall program quality." (Anderson & Gans, 2010)

Já outros autores apostam em um equilíbrio. A atitude de telespctadores de evitar comerciais pode levar a um realinhamento entre lucro de emissoras, qualidade e duração de comerciais (Wilbur, 2008).

Tom Rogers, fundador da CNBC em 1989 e atual CEO e presidente da TiVo, fez em 2008 o seguinte alerta aos participantes da Association of National Advertisers Conference:

> "In the next two to three years the television industry is going to face an advertising crisis more severe than our current financial crisis. You have sufficient warning about television commercial avoidance and the growing epidemic of fast forwarding thru ads and if marketers do not quickly come to terms with the solutions to commercial avoidance, most viewers will be fast forwarding the majority of television ads." (Rogers, 2008)

É relevante a opinião de que AATs são fatos reais que não podem ser ignorados, que são um sério problema para anunciantes e emissores e que eles devem levar isso em conta ao planejar suas campanhas publicitárias (El-Adly & Aicinena, 2010). Uma das contribuições desta tese consiste no desenvolvimento de uma AAT que incorpora, além das funções para evitar comerciais, também funcionalidades de incentivo à atividade de publicidade. Acredita-se com isso ser possível promover fluxos de externalidades positivas tanto no sentido anunciante telespectador como no sentido telespectador anunciante, embora esta não seja uma questão a ser explorada nesta tese.

5.3 Aplicação

A AAT permite que o usuário seja informado a respeito do começo e término de programas e intervalos comerciais em canais de TV. Como esse tipo de informação não é provido de maneira automática pelas emissoras, foi desenvolvido um provedor de informações independente, capaz de identificar o início e término de programas e intervalos comerciais. Esse provedor de informações também proporciona conteúdo complementar relacionado ao conteúdo transmitido na TV, permitindo uma interação independente das aplicações construídas pela emissora. Esse modelo, entretanto, não é exclusivo. É considerada a possibilidade de obtenção dessas informações a partir das próprias emissoras. O ferramental desenvolvido consiste em:

1. Desenvolvimento de uma AAT que contemple os requisitos elencados na hipótese. Essa AAT tem como foco a execução em dispositivos móveis, permitindo que o usuário seja notificado sempre que blocos comerciais de programas de TV sejam finalizados (de acordo com os canais selecionados);

2. Desenvolvimento de uma ferramenta para a autoria de conteúdo complementar. Essa ferramenta permite a geração de conteúdo complementar ao apresentado na TV. Esse conteúdo pode estar relacionado a blocos

comerciais, onde é inserida alguma interatividade relacionada a um segmento de vídeo da TV. A demarcação de segmentos que permite a identificação de início/fim de blocos comerciais, necessárias à ATT, também é apoiada por esta ferramenta;

3. Desenvolvimento de uma aplicação para a geração automática dos sinais de sincronização necessários para o funcionamento da AAT e da aplicação que permite a interatividade com comerciais;

4. Desenvolvimento de uma aplicação que facilita o suporte humano na geração semiautomática de sinais de sincronização. Em situações que não é possível a utilização de técnicas automatizadas é necessário algum apoio humano para permitir a sincronização das mídias. Uma ferramenta foi desenvolvida para facilitar esse apoio;

5. Desenvolvimento de APIs que permitem a integração à AAT de conteúdo complementar provido por terceiros. Esse conteúdo pode ser desde sinais de sincronização indicando início e fim de blocos comerciais, a conteúdo interativo. Através dessas APIs a AAT é dotada de capacidade para promoção de uma "externalidade" positiva;

6. Desenvolvimento de uma aplicação Web que permita ao usuário interagir com o conteúdo complementar através de computadores. Um dos problemas levantados na literatura é a perda de foco do usuário no conteúdo da TV, quando utilizando smartphones. Com a aplicação desenvolvida é possível que esse impacto seja minimizado, já que o usuário pode realizar outras interações com o conteúdo complementar "a posteriori";

7. Desenvolvimento de um sistema de retaguarda que permite a comunicação entre as demais aplicações propostas e que seja robusto para atender o volume de usuários típicos de ambientes de TV.

O processo de notificação do usuário a respeito de ocorrências em um canal depende da identificação do canal em que a TV do usuário estiver sintonizada. Isso pode ser realizado de algumas maneiras:

1. Monitoramento do sinal de áudio, através do microfone do dispositivo móvel, para identificar o canal de TV com base em algoritmos de processamento de áudio.

2. Comunicação ubíqua entre o dispositivo e o aparelho de TV, buscando obter as informações a respeito do canal sintonizado;

3. Interação explícita do usuário, informando o canal que deseja receber notificação sobre início de programas.

A abordagem 1 (monitoramento do áudio), embora ubíqua, possui aspectos relacionados à privacidade do usuário bastante sensíveis. O foco da aplicação é a identificação do canal de TV, mas colateralmente podem ser obtidos outros dados. A segurança da transmissão do fluxo de áudio é critica, pois podem ser capturadas informações sigilosas do usuário, como diálogos. A largura de banda demandada para a transmissão do fluxo de áudio para processamento remoto também pode ser um fator de dificuldade, já que as conexões em dispositivos móveis geralmente são limitadas.

No escopo desta AAT optou-se por abordagens menos invasivas à privacidade do usuário. Foi dada preferência à captura do canal através de uma interação explícita do usuário (opção 3), apoiada por informações de outros dispositivos, como smartvs, quando disponíveis no ambiente. A opção da utilização da AAT associada a *smartvs* é explorada no escopo de um projeto de mestrado em andamento.

5.3.1 Desafios

A viabilização da aplicação proposta passa pela superação de desafios científicos e tecnológicos. Os resultados das pesquisas descritas nos capítulos anteriores colaboram para que fosse viável a construção da aplicação proposta:

1. **Discriminação de momentos**: a ferramenta de autoria desenvolvida para a AAT é baseada nos resultados descritos na Seção 2.2 com relação aos operadores para a discriminação de momentos e demarcação de âncoras. O conceito de *stickers* apontados nessa seção apoiam a aplicação de geração semiautomática de sinais de sincronização (aplicação 4);

2. **Utilização de dispositivos móveis como elementos de interação** : os experimentos realizados no trabalho descrito na Seção 2.2 com relação à execução de uma sessão de interação colaborativa, baseada no dispositivo móvel como elemento de interação, apoiaram o desenvolvimento da aplicação móvel da AAT;

3. **Captura de *streams***: os processos de autoria da ATT demandam a captura do *stream* de TV. A interação com dispositivos de captura foi explorada na Seção 2.3. Os processos de manipulação de vídeo descritos naquela sessão também foram adotados na construção da AAT;

4. **Apresentação de conteúdo multimídia com suporte a múltiplas plataformas**: o trabalho descrito na Seção 3.1, em que é proposta a construção de uma máquina de apresentações NCL com suporte a execução em múltiplas plataformas, atuou como um facilitador para que o conteúdo multimídia pudesse ser apresentado na segunda tela (dispositivo móvel). Uma única linguagem para a especificação de apresentações multimídia pôde ser utilizada para a especificação do conteúdo, contribuindo para redução da complexidade e custos de desenvolvimento;

5. **Sincronização não convencional**: Os trabalho descritos no Capítulo 4 contribuem para a viabilização da sincronização distribuída (aplicação 3), independente da emissora. Os artefatos técnicos produzidos (especialmente a Seção 4.1) foram utilizados na implementação da AAT.

5.3.2 Arquitetura da Aplicação

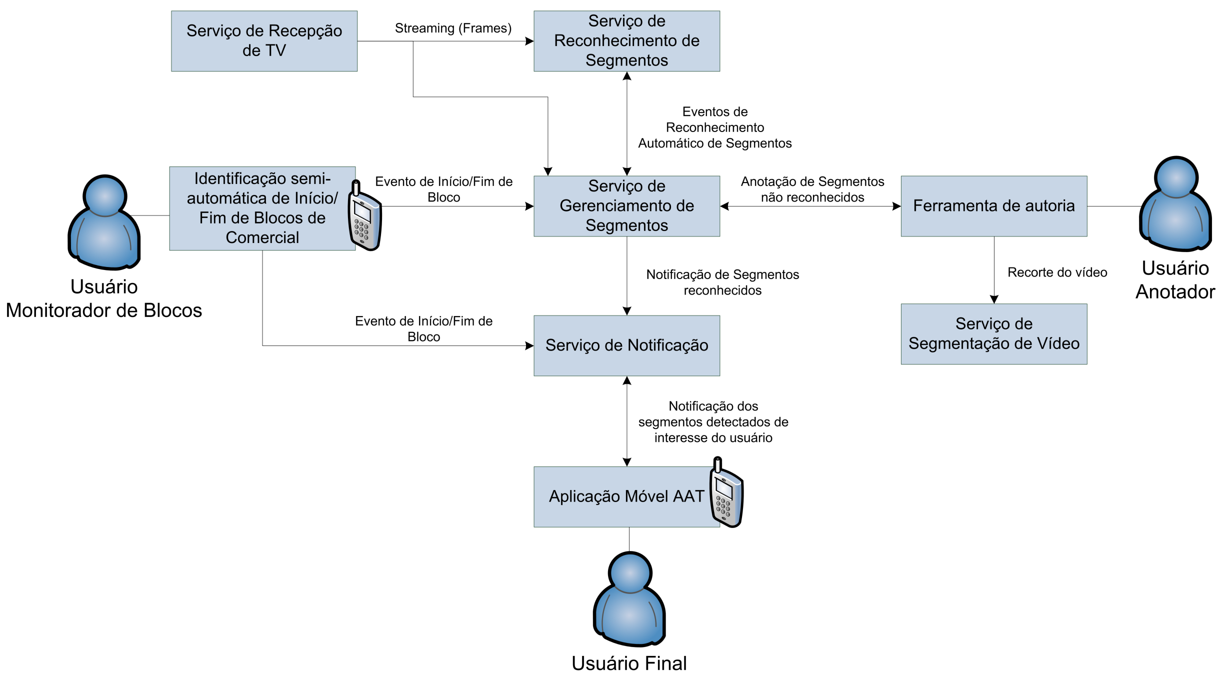

Serviço de Recepção de TV
Streaming (Frames)
Serviço de Reconhecimento de Segmentos
Eventos de Reconhecimento Automático de Segmentos
Identificação semi-automática de Início/Fim de Blocos de Comercial
Evento de Início/Fim de Bloco
Serviço de Gerenciamento de Segmentos
Anotação de Segmentos não reconhecidos
Ferramenta de autoria
Usuário Monitorador de Blocos
Usuário Anotador
Notificação de Segmentos reconhecidos
Recorte do vídeo
Evento de Início/Fim de Bloco
Serviço de Notificação
Serviço de Segmentação de Vídeo
Notificação dos segmentos detectados de interesse do usuário
Aplicação Móvel AAT
Usuário Final

Figura 5.1: Arquitetura da AAT

A AAT baseia-se na captura ininterrupta do *stream* enviado por uma emissora de TV (*broadcast*). O *stream* é processado, em tempo real, visando o reconhecimento de segmentos de vídeo (comerciais, vinhetas de programas) previamente anotados e armazenados em um banco de dados. Paralelamente a AAT é assistida por um operador humano, que sinaliza o início e fim de blocos de comercial.

Considera-se blocos de comercial um conjunto de segmentos adjacentes, delimitado por trechos de programa (também conhecidos como *breaks* na linguagem de TV). Um segmento é definido, no escopo da AAT, como um trecho de vídeo com conteúdo bem definido, autocontido e com valor semântico ao usuário.

Blocos de comercial passam por um processo de reconhecimento automático, identificando aqueles segmentos previamente anotados. Segmentos não reconhecidos são postos em uma fila para anotação, para que um usuário-anotador realize a delimitação do segmento e a autoria do conteúdo a ele relacionado. O segmento resultante da autoria é inserido no banco de dados utilizado para o reconhecimento de segmentos.

Os segmentos de vídeo reconhecidos no *stream* de TV têm seu conteúdo associado (resultado da autoria) apresentado ao usuário em seu dispositivo móvel (*smartphone*), lhe facultando a possibilidade de visualizar maiores detalhes do comercial. A sinalização de fim de bloco de comercial também é enviada ao usuário, caso este tenha interesse na notificação (característica de *ad-avoidance*).

A AAT foi projetada com base em uma arquitetura orientada a serviços (Figura 5.1). A adoção desse tipo de arquitetura facilitou o desenvolvimento do projeto, que possui componentes desenvolvidos em diferentes linguagens de programação, escolhidas de acordo com o propósito do serviço e disponibilidade de componentes e bibliotecas que pudessem ser reusados. O funcionamento de cada serviço da AAT é detalhado a seguir.

Serviço de recepção de TV

Serviço capaz de sintonizar canais de TV, capturar o *framebuffer* do *hardware* receptor e gravar o conteúdo recebido. A gravação é realizada ininterruptamente, para possibilitar a implementação da janela de segmentos. No caso da prova de conceito foi utilizada uma janela de 5 segundos antes e após as marcas de início e fim de segmento. Essa janela permite que o autor do conteúdo possa realizar possíveis correções dos limites do segmento na fase de autoria, evitando a perda de conteúdo em caso de atrasos na detecção do segmento (que pode ser realizada tanto por um humano quanto de maneira automatizada). O serviço foi implementado em Python, apoiado por uma biblioteca desenvolvida para o serviço em C++ (responsável pela comunicação com o receptor e sintonização de canais). A codificação de vídeo foi realizada com o apoio da biblioteca LibAV[2], no formato H.264.

[2]https://libav.org/

Serviço de Detecção de Segmentos

Serviço capaz de detectar automaticamente a existência de um segmento de vídeo em uma base de segmentos previamente anotados. A detecção é realizada com base no algoritmo descrito na Seção 4.1.2. O serviço foi implementado em linguagem Python. A comunicação com o serviço de recepção de TV é implementada através de memória compartilhada por questões de performance (o fluxo de *bytes* é gerado a uma taxa muito elevada, o que inviabiliza a utilização de outros protocolos). Esse serviço reside tipicamente no mesmo nó (computador) em que é feita a recepção de TV, por questões de latência e largura de banda. A notificação da ocorrência de detecção de segmentos é enviada através de *broker* de mensagens baseado no protocolo AMQP[3].

Aplicação de Marcação de Início e Fim de Segmento

Aplicação de apoio para a delimitação de início e fim de segmentos de vídeo. A aplicação desenvolvida possui a mesma natureza daquela descrita na Seção 2.2. A implementação desenvolvida foi baseada em tecnologias Web, para execução em dispositivos móveis. Nela pode-se identificar o início/fim de bloco comercial de determinado canal de TV. Esses eventos alimentam um gerenciador de segmentos, responsável por iniciar o processo de detecção de segmentos para anotação e por notificar usuários interessados em eventos de início/fim de bloco comercial - apoiando a característica de *ad-avoidance* da aplicação.

A utilização dessa aplicação é opcional. Alguns canais de TV implementam a política de *frames* pretos, brancos ou redução de som para sinalizar o início de blocos de comercial (Hua et al., 2005). Outra política que pode ser considerada é a detecção de vinhetas, realizadas com o apoio do Serviço de Detecção de Segmentos. Como a emissora pode alterar a política de sinalização dos blocos sem aviso prévio, essa aplicação é um importante elemento para a resiliência e confiabilidade da AAT.

A Figura 5.2 apresenta imagem da tela do protótipo desenvolvido. A notificação dos eventos de início/fim de bloco é realizada através de protocolo HTTP com o *backend* da AAT. No *backend* o evento é repassado ao *broker* de mensagens, que repassa a informação aos demais módulos interessados.

Serviço de Gerenciamento de Segmentos

O serviço é capaz de orquestrar a comunicação entre os serviços do sistema. O serviço recebe os eventos de detecção de segmentos (através de detecção

[3]Advanced Message Queuing Protocol - http://www.amqp.org/

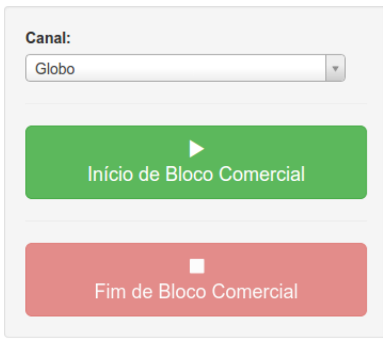


Figura 5.2: Protótipo da Aplicação de Marcação de Início e Fim de Segmento.

automática ou humana), notificando a ocorrência de um segmento de vídeo (por exemplo um comercial). Segmentos passíveis de anotação também são detectados por este serviço, que se encarrega de disponibilizar o segmento para anotação na ferramenta projetada para esse fim.

A inserção de novos segmentos na base é realizada através de comunicação com este serviço, que atualiza a base de segmentos utilizada pelo "Serviço de Detecção de Segmentos".

A implementação deste serviço foi realizada em plataforma Grails[4] (Java). O serviço disponibiliza interfaces RESTful e se conecta aos outros serviços, que recebem eventos *push* via *broker* de mensagens.

Serviço de Notificação

Serviço capaz de notificar os dispositivos dos usuários da AAT acerca do início e fim de bloco comercial e da identificação de determinado segmento. Este serviço possui alguns requisitos:

- Deve ser capaz de escalar de maneira horizontal, permitindo a incorporação de uma infraestrutura de hardware adicional para suprir picos de demanda;

- Diversos dispositivos estarão integrados à plataforma, com os mais variados sistemas operacionais. Esse serviço deve ser capaz de permitir que

[4]http://www.grails.org

esses dispositivos heterogêneos se comuniquem de maneira homogênea, respeitando os seus protocolos de comunicação nativos;

- Dispositivos móveis estarão integrados e nesses casos existem restrições quanto ao uso de bateria. O serviço deve permitir a comunicação com um baixo custo computacional, minimizando o impacto na bateria do dispositivo;

- Cada conteúdo complementar possui os seus requisitos de qualidade de serviço. É importante que o serviço seja capaz de responder a esses requisitos de maneira satisfatória, garantindo que a sincronização seja mantida dentro dos limites estabelecidos;

- Deve levar em consideração a distribuição geográfica dos usuários, evitando rotas de comunicação demasiadamente custosas. A distribuição geográfica da infraestrutura é desejável para mitigar aumento na latência.

Para atender aos requisitos listados foram consideradas duas estratégias não-exclusivas: (i) utilização de um serviço (*software as a service*) nativo do próprio dispositivo, como o *Google Cloud Messaging* para dispositivos Android e *Apple Push Notification* para dispositivos iOS; (ii) utilização de um *cluster* de servidores de mensagens (*brokers*) especializado para a AAT Figura 5.3.

As abordagens propostas são complementares já que nem todos os dispositivos, como *smartvs*, possuem integração nativa a serviços de *push*.

Para a implementação da estratégia (ii) foi escolhido o broker de mensagens Apache ActiveMQ Apollo . A escolha se deu pelos seguintes fatores:

- O ActiveMQ Apollo é executável em qualquer ambiente que suporte JVM 1.4 ou superior, dando liberdade de escolha do ambiente de execução

- Em testes de estresse pôde-se notar a capacidade do ActiveMQ Apollo de entregar mensagens a uma taxa superior a 300.000 mensagens por segundo, em uma única instancia do broker.

- Suporta um conjunto elevado de linguagens e protocolos. Existem implementações disponíveis de conectores nas linguagens mais populares como Java, C++, Objective C, Python, JavaScript, PHP, C#.

- Possui conectores específicos para plataformas web, permitindo a publicação e consumo de mensagens através de AJAX, REST e WebSockets

- O ActiveMQ Apollo tem nativamente suporte à construção de clusters de brokers, permitindo que a entrega de mensagens seja distribuída e tolerante a falhas.

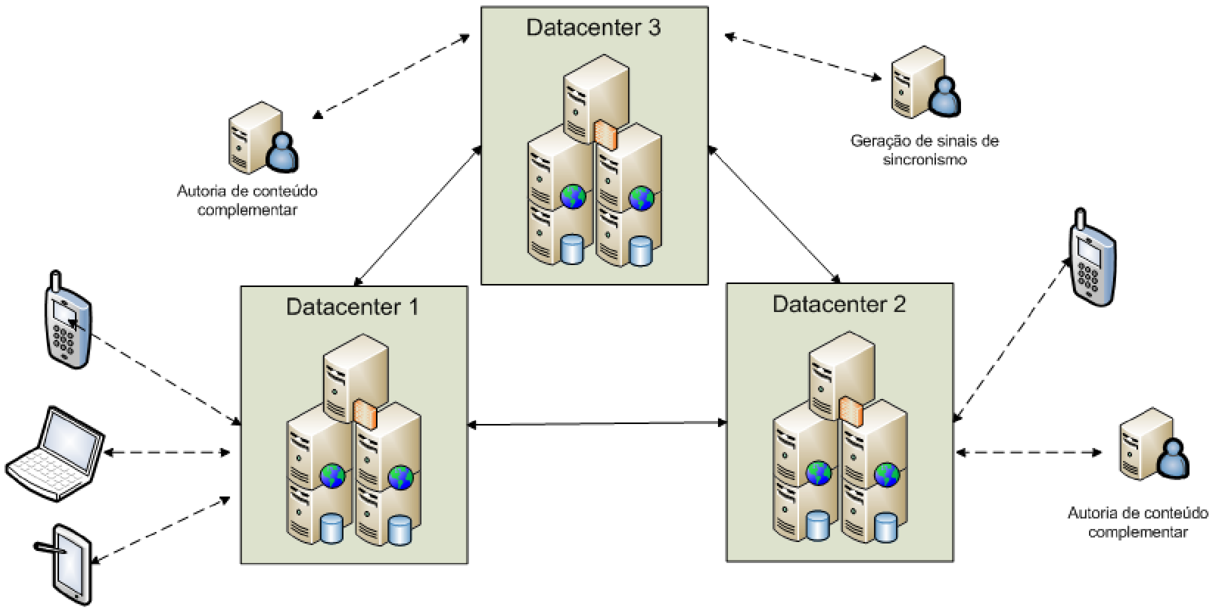

Datacenter 3
Autoria de conteúdo complementar
Geração de sinais de sincronismo
Datacenter 1
Datacenter 2
Autoria de conteúdo complementar

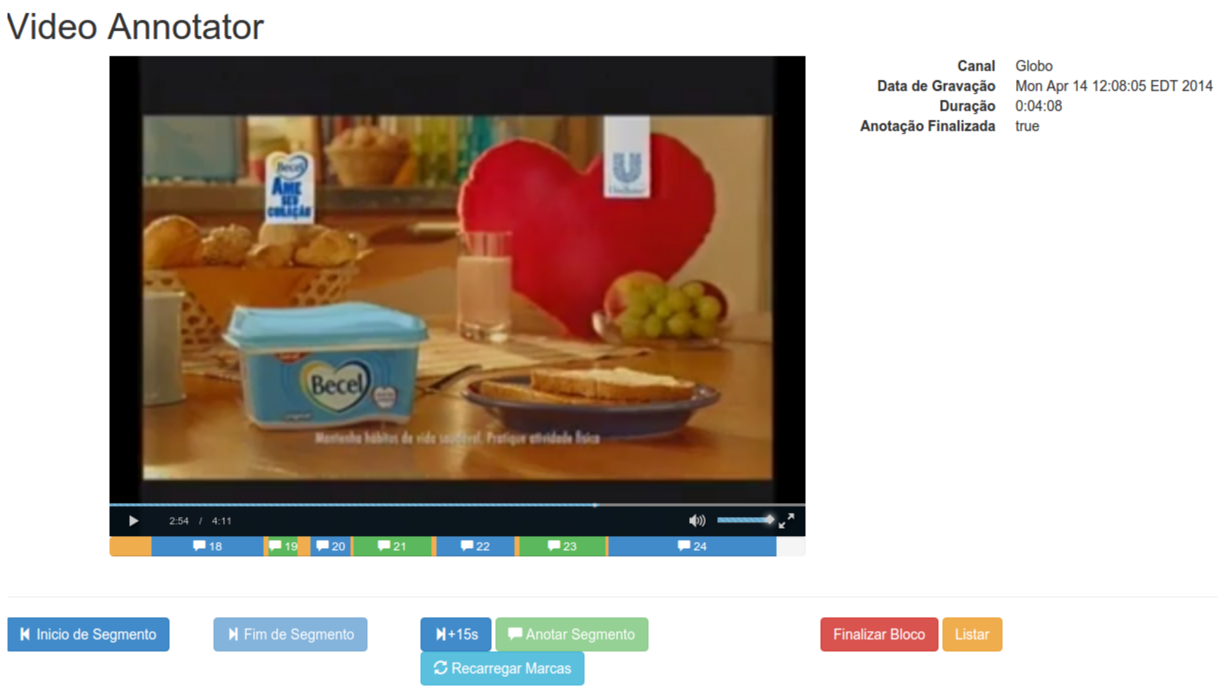
Figura 5.3: Cluster de servidores de mensagem

Ferramenta de Autoria

A delimitação de segmentos e autoria de conteúdo complementar requer o uso de ferramental adequado. Foi desenvolvida uma ferramenta que permite ao usuário-anotador delimitar o início/fim de um segmento e associar conteúdo ao mesmo. O conteúdo gerado pode ser de natureza textual, imagens extraídas do próprio vídeo do segmento, ou aplicações multimídia interativas (incluindo aquelas baseadas em NCL, possível de serem apresentadas através da WebNCL (Seção 3.1)).

A Figura 5.4 retrata a execução da ferramenta. Nela o usuário/anotador tem a opção de delimitar o início/fim de um segmento, visualizar quais segmentos já foram delimitados e associar conteúdo a um segmento (Figura 5.5). Uma *timeline* é apresentada listando quais segmentos do vídeo foram reconhecidos ou anotados.

A ferramenta foi desenvolvida em HTML5, permitindo a sua utilização em diversas plataformas. Ao usuário-anotador demarcar um segmento e inserir as anotações relativas ao mesmo a ferramenta realiza o corte do vídeo (através do Serviço de Segmentação de Vídeo). O segmento é enviado ao Serviço de Gerenciamento de Segmentos que realiza os seguintes procedimentos:


Video Annotator

Canal	Globo
Data de Gravação	Mon Apr 14 12:08:05 EDT 2014
Duração	0:04:08
Anotação Finalizada	true

Figura 5.4: Demarcação de segmentos

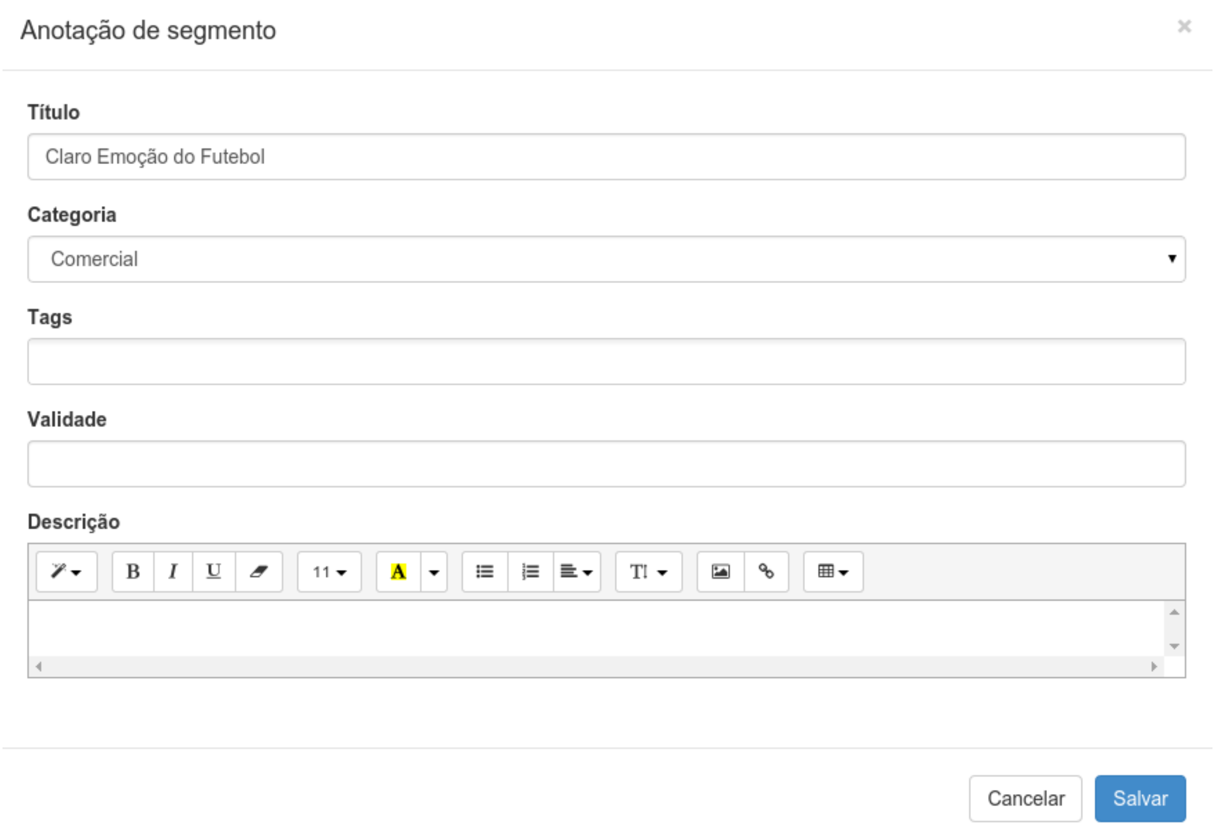
- Verifica se o segmento já não foi anotado de maneira concorrente por outro anotador. Caso tenha sido o usuário é avisado, para solucionar possíveis conflitos;
- Insere o segmento em uma base de segmentos utilizada para o reconhecimento automatizado, tornando-o disponível para o Serviço de Detecção de Segmentos;
- Percorre toda a base de segmentos pendentes de anotação, buscando o reconhecimento do referido segmento nos demais blocos pendentes de anotação. Isso minimiza as chances de um mesmo segmento ser enviado para anotação duas vezes.

Serviço de Segmentação de Vídeo

Serviço capaz de seccionar um vídeo, dado o instante de início e sua duração. O serviço também é capaz de codificar o vídeo segmentado em diversos codificadores e resoluções de imagem.

Foram realizadas duas implementações do serviço:

1. Uma implementação utilizando *software livre*, baseada na biblioteca LibAV. A codificação do vídeo foi realizada no protocolo H.264. Entretanto o vídeo produzido não se adequa ao processo de *streaming progressivo*, dificultando os processos de *skip* do vídeo, quando o *download* está em


Anotação de segmento

Título

Claro Emoção do Futebol

Categoria

Comercial

Tags

Validade

Descrição

Figura 5.5: Anotação de segmento

andamento. Além disso essa implementação apresenta uma demanda elevada por recursos computacionais para a codificação de vídeo.

2. Uma implementação baseada no serviço *AWS Transcoding*, fornecido pela Amazon. O vídeo produzido é totalmente adequado ao processo de *streaming progressivo* e os custos envolvidos para a codificação são inferiores ao custo de manter um servidor dedicado ao processo de codificação (comparando com os custos de um servidor virtual na Amazon).

Serviço de retaguarda - backend

O serviço de retaguarda apoia o funcionamento da AAT em diversas etapas. Por questões de clareza do diagrama este serviço não está listado, mas ele atua provendo serviço a todos os outros módulos, orquestrando a comunicação entre os mesmos.

Foi desenvolvida uma API para a publicação e consumo de conteúdo. Essa API também é capaz de permitir a inserção de novos tipos de conteúdos a serem sincronizados com o fluxo de TV em tempo real.

O armazenamento do conteúdo complementar é realizado através de um banco de dados não relacional orientado a documentos. Um dos requisitos levantados para a plataforma é a sua adaptabilidade a diversos tipos de conteúdos complementares. Bancos de dados orientados a documentos são *schemaless*. Isso significa que não existe um único esquema de dados para uma coleção de documentos. Cada documento pode ter um conjunto de propriedades distintas. Outro aspecto importante dessa categoria de banco de dados é a ausência de controle de transações e chaves estrangeiras – não existem JOINs como no modelo relacional. Isso permite que o banco de dados atenda a um conjunto maior de consultas concorrentes, se comparado a bancos de dados relacionais.

Na implementação do *backend* do sistema foi adotado o banco de dados MongoDB . Trata-se de um banco de dados orientado a documentos implementado em linguagem C++ e que possui conectores disponíveis para uma grande quantidade de linguagens, incluindo Java e Python. Os dados são organizados na forma de coleções (similares às tabelas relacionais). Entretanto cada coleção pode ter variações de esquemas de dados. Isso dá a flexibilidade de armazenamento dos conteúdos complementares e evolução da plataforma, com a inserção de novos tipos de conteúdo estruturados sem que seja necessária a alteração de *schema*.

Outro aspecto importante do banco de dados MongoDB é o recurso de sharding e replicação, que permitem a adição de novos nós (servidores) na rede, balanceando a carga do banco de dados de maneira automática, aumentando a sua confiabilidade e escalabilidade.

Finalmente foi considerado o uso de nuvens de servidores para abrigar a aplicação. Essas nuvens podem estar distribuídas geograficamente, permitindo que o usuário tenha acesso ao serviço com a menor latência possível. Um efeito positivo disso é a possibilidade de acompanhar os picos de audiência demandados por determinados programas de TV, permitindo que a infraestrutura utilizada acompanhe a demanda, reduzindo custos. A tarifação desses servidores na nuvem é dada de acordo com a utilização da infraestrutura, sendo possível um aprovisionamento de recursos em tempo real.

Aplicação Móvel AAT

A implementação da prova de conceito se deu através de tecnologias baseadas na web (HTML 5, JavaScript, CSS) devido à possibilidade de utilização da mesma base de código em diversas plataformas – dispositivos móveis, web e smartTVs. Para acesso a recursos dependentes de plataformas, como notificações em dispositivos móveis, foi adotado o framework Apache Cordova.

Um requisito importante para dispositivos móveis é a economia de recursos de banda (rede) e bateria. Como muitos dados da aplicação são requisitados diversas vezes, como imagens e outros arquivos estáticos a adoção de uma estratégia de cache para a minimização de acessos a recursos externos ao dispositivo, de maneira recorrente, pode ser adequada.

Dada a disponibilidade tecnológica da plataforma HTML5, optou-se por realizar a implementação das estruturas de armazenamento local baseadas na tecnologia WebSQL, com a utilização de índices para otimização das buscas. Entretanto foram encontradas limitações para o armazenamento nativo de dados binários (como é o caso de imagens). Para contornar esse problema adotou-se a conversão de dados binários para uma representação textual, em base 64. No caso das imagens a conversão é feita nativamente através do método toDataUrl(), implementado no objeto Canvas do HTML. Por razões de desempenho optou-se por utilizar a API de WebStorage (armazenamento de pares chave-valor) para dados binários.

Foram experimentadas três abordagens para a comunicação. A primeira se deu através da funcionalidade de WebSockets do padrão HTML5. Entretanto foi notado que ainda são poucos os navegadores que a implementam em plataformas móveis. Diante das limitações tecnológicas optou-se pela utilização de uma estratégia baseada em *polling*, na qual o dispositivo checa periodicamente a existência de mensagens no servidor. Essa abordagem pode gerar um aumento na latência da comunicação, já que o tempo de entrega de mensagens está diretamente relacionado ao tempo de *polling*. Uma outra abordagem experimentada foi a utilização de um serviço (*software as a service*) nativo do próprio dispositivo, como o *Google Cloud Messaging* para dispositivos Android e *Apple Push Notification* para dispositivos iOS.

Para dispositivos móveis Android e iOS a utilização do serviço nativo se mostrou a solução mais apropriada, devido à possibilidade de notificar o usuário mesmo que a aplicação esteja inativa (em *background*, por exemplo). A existência de suporte a esses serviços através de *plugins* do Apache Cordova foi um facilitador do desenvolvimento.

A implementação da interface gráfica foi apoiada pelos seguintes *frameworks*:

1. jQuery Mobile: apresenta uma abordagem declarativa, em que as interfaces são majoritariamente baseadas em código HTML5

2. Sencha Touch: apresenta uma abordagem imperativa, em que as interfaces são majoritariamente baseadas em componentes JavaScript.

Foram realizadas duas implementações distintas, uma utilizando cada *framework*. A fluidez da aplicação foi insatisfatória (principalmente as transições) com o *framework* jQuery Mobile. A implementação baseada em Sencha Touch produziu uma experiência do usuário mais consistente, sem travamento das transições. A Figura 5.6 apresenta uma visão da aplicação.

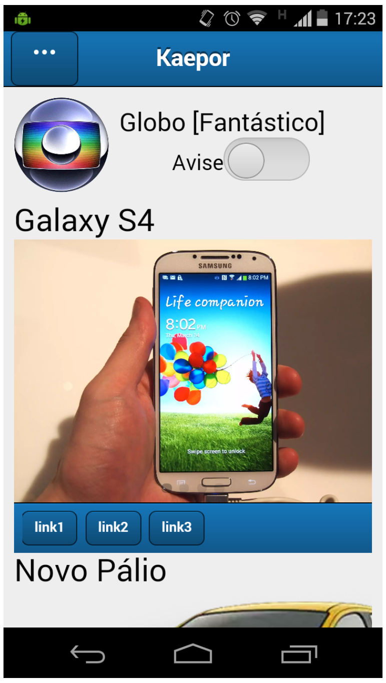


Figura 5.6: Aplicação para interação com AAT no dispositivo móvel

5.4 Considerações Finais

A aplicação do tipo *Ad-Avoidance* explorada neste capítulo tem grande potencial para contribuir com o aumento da satisfação do usuário. A construção do conteúdo multimídia é realizada de maneira semi-automatizada. A mídia produzida é apresentada ao usuário de maneira contextualizada, constituindo conteúdo complementar àquele em exibição na TV.

A hipótese de que *"A utilização de uma Ad-Avoidance Tehcnology (AAT) adequada pode promover a TV Interativa"* foi explorada através do desenvolvimento do ferramental necessário à sua viabilização. A comprovação da hipótese carece, ainda, de estudos com usuários. Os estudos com usuários serão conduzidos e seus resultados disponibilizados em artigos científicos.

Capítulo 6
Considerações Finais

O objetivo principal da tese é apresentar indícios que aplicações multimídia interativas e potencialmente atrativas podem ser construídas com a incorporação automática ou semiautomática, a uma mídia principal, de informações de contexto representadas em diferentes mídias.

Destacam-se também alguns objetivos secundários, que caracterizam as diversas etapas que foram necessárias no trabalho. Essas etapas referem-se a aspectos de áreas como autoria multimídia, difusão de aplicações multimídia em múltiplas plataformas e sincronização de mídias.

Foram propostas e exploradas ferramentas para facilitar a autoria de conteúdo multimídia, tanto de maneira automática quanto de maneira semi-automatizada. No Capítulo 2 foram apresentados trabalhos que evidenciam a viabilidade da construção de aplicações multimídia interativas de maneira automática e semi-automatizada.

Na Seção 2.1 foram discutidos os problemas relacionados à engenharia de documentos multimídia interativos, incluindo aqueles relativos à autoria de documentos multimídia em fase de exibição e interação (edição ao vivo). Foram propostas soluções relacionadas à engenharia de documentos, interação e casos de uso de autoria ubíqua. Foi possível avaliar a viabilidade da autoria ubíqua de documentos multimídia baseadas nas interações do usuário (complementando o conteúdo apresentado na TV). Foram observados também que o mecanismo de interação típico da TV (controle remoto) não oferece uma experiência de interação adequada aos usuários e que o caráter "social"da TV (usuário geralmente assiste TV em ambientes coletivos) apresenta oportunidades e desafios para que a autoria do conteúdo se dê de maneira colaborativa.

Dando sequência aos estudos, a interação e autoria colaborativa foi estudada na Seção 2.2, em que é explorada a discriminação de momentos e intervalos em mídia usando-se dispositivos móveis pessoais. O trabalho relatado nessa seção evidenciou que dispositivos móveis podem apoiar o processo de autoria e que a autoria colaborativa, baseada em dispositivos pessoais, pode produzir aplicações potencialmente atrativas. Foram realizados experimentos com interação colaborativa através de dispositivos móveis, sincronizada com o conteúdo da TV e a utilização de *stickers* como um elemento de interatividade. Foram estudados modelos de sincronização, avaliando maneiras alternativas à tradicional, baseada linhas temporais e interações com o controle remoto.

Os estudos realizados, apoiados por projetos de P&D, mostraram que a integração de diferentes dispositivos, cooperando para a apresentação de conteúdo de maneira síncrona, requer mecanismos não convencionais para a manutenção das relações de sincronismo e apresentação do conteúdo multimídia. Os projetos e experimentos com usuários apontaram para a relevância de uma máquina de apresentações capaz de ser executada em múltiplas plataformas, além de indicarem que a autoria ubíqua de documentos multimídia poderia ser aplicada a outros domínios além da TV Digital Interativa, em ambientes com uma riqueza ainda maior de informações contextuais que podem agregar valor a uma apresentação multimídia.

Foi proposta e desenvolvida uma ferramenta de autoria capaz de gerar um documento multimídia, de maneira automática, a partir da captura de informações contextuais (Seção 2.3). A ferramenta, avaliada no cenário educacional, reforçou a tese de que aplicações potencialmente atrativas podem ser construídas com a incorporação de mídias a uma mídia principal e que a autoria pode se dar através de informações contextuais. Foram realizados experimentos que indicaram que um conteúdo multimídia interativo pode ser produzido de maneira ubíqua, através da captura de informações contextuais e que essas informações agregam valor à mídia, tornando o conteúdo potencialmente atraente para os usuários finais. O trabalho mostrou, ainda, que para uma efetividade na interatividade o conteúdo deveria estar disponível aos usuários de maneira simples, sem a necessidade da instalação de ferramentas ou aplicações específicas.

A promoção da interação do usuário com aplicações multimídia potencialmente atrativas depende diretamente da disponibilidade dessas aplicações. Um dos fatores de atratividade e satisfação do usuário refere-se à disponibilidade das aplicações. Nos projetos foi observado que os usuários têm dificuldade em lidar com emuladores e instalação de aplicações em seus dispositivos para lidar com conteúdo multimídia interativo. Em resposta a esses desafios foi proposta e desenvolvida uma máquina de apresentação de documentos multimídia baseada em tecnologias web, capaz de ser executada em qualquer *browser* com suporte a HTML5 (*smartphones, tablets, smartvs, PCs* (WebNCL - Seção 3.1), que viabiliza a difusão de aplicações multimídia em uma grande variedade de dispositivos e plataformas, contribuindo para que as aplicações estejam mais acessíveis ao usuário. Essa máquina de apresentações pode ser um facilitador para a utilização de aplicações multimídia interativas em ambientes distintos da TV Digital Interativa, como plataformas móveis e na *web*, que possuem um apelo ainda maior à interatividade que a TVDi.

Conteúdo complementar pode se tratar, além dos casos convencionais (vídeo, áudio, texto), de um objeto sintético (Seção 4.3) e interação realizada via comunicação síncrona de vídeo (Seção 4.2). Foram estudadas estratégias para a integração desse tipo de conteúdo complementar a apresentações multimídia.

A integração de conteúdo multimídia (conteúdo complementar) em diferentes dispositivos, cooperando para a apresentação de conteúdo de maneira síncrona, requer mecanismos não convencionais para a manutenção das relações de sincronismo. O modelo tradicional de sincronização, em que as âncoras são definidas apenas no tempo cria uma dissociação entre a especificação do documento multimídia e o contexto considerado pelo autor do documento – uma cena é diferente de um *timestamp*. É comum no ambiente de TV existirem diversas versões do mesmo conteúdo, como compactos de eventos esportivos e versões de filmes com cortes de cenas. É necessário um esforço adicional de edição do documento multimídia para que a intenção do autor seja preservada. O processo de autoria também fica pouco natural, já que a especificação em alto nível envolve o estabelecimento de relações entre mídias baseadas no conteúdo, no que está sendo apresentada, e não em instantes de tempo.

Foram estudadas estratégias para sincronização de mídias de maneira não convencional. A sincronização pode ser realizada tanto localmente quanto de maneira distribuída, em tempo real ou utilizando-se técnicas de pré processamento. Na Seção 4.1 são descritas algumas técnicas para a sincronização em tempo real (conteúdo ao vivo), baseadas no conteúdo das mídias. Os resultados obtidos facilitam a execução de atividades colaborativas em apresentações multimídia, em que usuários geograficamente distribuídos podem compartilhar a experiência de uma interatividade colaborativa com uma apresentação multimídia em tempo real. São facilitadores para a integração de mídias não convencionais, produzidas através de *clusters* computacionais, a apresentações multimídia, permitindo, inclusive a definição de âncoras para a interatividade; o reúso de documentos multimídia, preservando os elementos de sincronização, mesmo em casos em que as mídias originais (como vídeo) sejam customizadas.

Durante os estudos, observou-se que é cada vez mais comum as pessoas assistirem TV acompanhadas dos seus dispositivos móveis e produzindo conteúdo atrelado aos programas de TV em redes sociais. Isso pode ser observado através dos *trending topics* to Twitter, nos quais é frequente a presença de comentários sobre programas de TV (Digital, 2012). Observou-se também que as aplicações interativas disponíveis, de maneira geral, trazem para a TV uma interatividade próxima daquela praticada na web. Mesmo quando aspectos de interação e usabilidade são tratados (tamanho de fontes, disposição de mídias, etc), o conteúdo é essencialmente tradicional e não acrescenta elementos inovadores que estimulam o usuário a sair do seu estado de passividade e interagir.

Essa situação motivou a busca por alternativas que pudessem alavancar a interatividade na TV. Foi proposta uma aplicação do tipo *ad-avoidance* (Capítulo 5) como meio de promoção da interatividade. Entende-se como aplicação *ad-avoidance* aquela que faculta ao usuário (telespectador) a opção de evitar comerciais de TV. A escolha desse tipo de aplicação foi resultado de estudos na literatura sobre o modelo de negócio da TV, onde foi observado que blocos de comercial na TV trazem incômodo ao usuário e que aplicações do tipo *ad-avoidance* provocam um sentimento de satisfação no usuário. Os estudos permitiram a formulação da hipótese de que *"A utilização de uma Ad-Avoidance Tehcnology (AAT) adequada pode promover a TV Interativa"*.

A aplicação desenvolvida baseia-se em um paradoxo: se por um lado os telespectadores não estão satisfeitos com o modelo de ter um programa de TV interrompido para a exibição de blocos comerciais; por outro lado a sustentabilidade do modelo da TV (principalmente TV aberta) depende do conteúdo publicitário para custear a produção dos programas.

Como elemento catalisador de aplicações interativas foi construída uma aplicação para *smartphones* que o telespectador possa usar para evitar comercial na TV. Para procurar diminuir o conflito com emissoras e anunciantes a aplicação interativa usada para comprovar o efeito catalisador é voltada a incentivar a curiosidade e, eventualmente ao consumo, dos produtos anunciados.

A aplicação apresenta duas características contraditórias mas complementares: (i) permite ao usuário evitar os comerciais de TV, possibilitando ao usuário ser informado quando a programação de um determinado canal voltou; (ii) permite que o usuário possa consumir os comerciais veiculados na

TV de maneira mais confortável, possibilitando a visualização de detalhes do comercial, como preços, datas e outras informações de maneira sincronizada à programação da TV .

A viabilização da aplicação passou pela superação dos desafios científicos e tecnológicos relacionados a autoria, apresentação e sincronização explorados ao longo da tese.

A hipótese de que *"A utilização de uma Ad-Avoidance Tehcnology (AAT) adequada pode promover a TV Interativa"* foi explorada através do desenvolvimento do ferramental necessário à sua viabilização. A comprovação da hipótese carece, ainda, de estudos com usuários. Os estudos com usuários serão conduzidos e seus resultados disponibilizados em artigos científicos.

6.1 Trabalhos Futuros

Como trabalhos futuros, no âmbito das atividades de empreendedorismo iniciadas em conjunto com esta pesquisa, pretende-se tornar a aplicação AAT efetivamente um produto, promovendo a integração com anunciantes para a disponibilização do conteúdo complementar relativo aos comerciais.

Para a continuidade da pesquisa científica vislumbra-se a realização de estudos com usuários, com vistas à comprovação da A hipótese de que *"A utilização de uma Ad-Avoidance Tehcnology (AAT) adequada pode promover a TV Interativa"*.

As técnicas de sincronização multimídia não convencional podem ser incorporadas a linguagens de especificação de apresentações multimídia, como a NCL. Âncoras de conteúdo podem ser incorporadas a essas linguagens, proporcionando relações de sincronismo mais semânticas.

Outros estudos interessantes para a continuidade desta pesquisa referem-se à autoria ubíqua de apresentações multimídia com informações de contexto. Novas técnicas para a extração de informações contextuais podem ser exploradas, como a mineração de dados e processamento de sinais para a identificação de âncoras.

Apêndice A

Publicações

A.1 Conferências

1. Caio Cesar Viel, **Erick Lazaro Melo**, Maria da Graça Pimentel, and Cesar Augusto Camillo Teixeira. 2013. *Multimedia multi-device educational presentations preserved as interactive multi-video objects.* In Proceedings of the 19th Brazilian symposium on Multimedia and the web (WebMedia '13). ACM, New York, NY, USA, 51-58.

2. Caio Cesar Viel, **Erick Lazaro Melo**, Maria da Graça Campos Pimentel, and Cesar Augusto Camillo Teixeira. 2013. *Go beyond boundaries of iTV applications.* In Proceedings of the 2013 ACM symposium on Document engineering (DocEng '13). ACM, New York, NY, USA, 263-272.

3. Caio Cesar Viel, **Erick Lazaro Melo**, Maria da Graça Campos Pimentel, and Cesar Augusto Camillo Teixeira. 2013. *How are they Watching Me -Learning from Student Interactions with Multimedia Objects Captured from Classroom Presentations.* In Proceedings of the 15th International Conference on Enterprise Information Systems (ICEIS 2013). 5-16

4. **Erick Lazaro Melo**, Caio César Viel, Cesar Augusto Camillo Teixeira, Alexandre Coelho Rondon, Daniel de Paula Silva, Danilo Gasques Rodrigues, and Endril Capelli Silva. 2012. *WebNCL: a web-based presentation machine for multimedia documents.* In Proceedings of the 18th Brazilian symposium on Multimedia and the web (WebMedia '12). ACM, New York, NY, USA, 403-410.

5. Caio César Viel, **Erick Lazaro Melo**, Arthur Pedro Godoy, Diego Roberto Colombo Dias, Luis Carlos Trevelin, and Cesar Augusto Camillo Teixeira. 2012. *Multimedia presentation integrating interactive media produced in real time with high performance processing.* In Proceedings of the 18th Brazilian symposium on Multimedia and the web (WebMedia '12). ACM, New York, NY, USA, 115-122.

6. Diogo Pedrosa, José Augusto C. Martins, Jr., **Erick L. Melo**, and Cesar A. C. Teixeira. 2011. *A multimodal interaction component for digital television.* In Proceedings of the 2011 ACM Symposium on Applied Computing (SAC '11). ACM, New York, NY, USA, 1253-1258.

7. Kamila R. H. Rodrigues, **Erick L. Melo**, Patricia I. Nakagawa, and Cesar A. C. Teixeira. 2010. *Interação com conteúdo complementar para apoio ao entendimento de programas televisivos.* In Proceedings of the IX Symposium on Human Factors in Computing Systems (IHC '10). Brazilian Computer Society, Porto Alegre, Brazil, Brazil, 91-100.

A.2 Periódicos

1. Caio César Viel, Kamila Rios da Hora Rodrigues, **Erick Lazaro Melo**, Renato Bueno, Maria da Graça Campos Pimentel, Cesar Augusto Camillo Teixeira. (2014). *Interaction with a Problem Solving Multi Video Lecture: Observing Students from Distance and Traditional Learning Courses.* International Journal Of Emerging Technologies In Learning (IJET), 9(1), pp. 39-46.

2. Cesar A. C. Teixeira, **Erick Lazaro Melo**, Giliard B. Freitas, Celso A. Santos, and Maria Da Pimentel. 2012. *Discrimination of media moments and media intervals: sticker-based watch-and-comment annotation.* Multimedia Tools Appl. 61, 3 (December 2012), 675-696.

3. Cesar A. C. Teixeira, **Erick Lazaro Melo**, Renan G. Cattelan, and Maria Da Pimentel. 2010. *Taking advantage of contextualized interactions while users watch TV.* Multimedia Tools Appl. 50, 3 (December 2010), 587-607

4. Maria da Graca Campos Pimentel, Renan G. Cattelan, **Erick Lazaro Melo**, Antonio Francisco Prado, and Cesar Augusto Camillo Teixeira. 2010. *End-user live editing of iTV programmes.* Int. J. Adv. Media Commun. 4, 1 (December 2010), 78-103.

Referências Bibliográficas

ABNT Associação Brasileira de Normas Técnicas (2007). Digital terrestrial television standard 06: Data codification and transmission specifications for digital broadcasting. Technical report, Associação Brasileira de Normas Técnicas.

ABNT Associação Brasileira de Normas Técnicas (2008). *NBR 15606-2:2007, Parte 2: Ginga-NCL para receptores fixos e móveis - Linguagem de aplicação XML para codificação de aplicações, Versão corrigida - 07.04.2008.*

Abowd, G. D., Atkeson, C. G., Feinstein, A., Hmelo, C., Kooper, R., Long, S., Sawhney, N., & Tani, M. (1996). Teaching and learning as multimedia authoring: the classroom 2000 project. In *Proceedings of the fourth ACM International Conference on Multimedia*, MULTIMEDIA '96, pages 187–198, New York, NY, USA. ACM.

Abowd, G. D., Dey, A. K., Brown, P. J., Davies, N., Smith, M., & Steggles, P. (1999). Towards a better understanding of context and context-awareness. In *Handheld and ubiquitous computing*, pages 304–307. Springer.

Abowd, G. D., Mynatt, E. D., & Rodden, T. (2002). The human experience. *IEEE Pervasive Computing*, 1(1):48–57.

Allen, J. F. & Hayes, P. J. (1990). Moments and points in an interval-based temporal logic. *Computational Intelligence Journal*, 5(4):225–238.

Anderson, S. P. & Gabszewicz, J. J. (2006). The media and advertising: a tale of two-sided markets. *Handbook of the Economics of Art and Culture*, 1:567–614.

Anderson, S. P. & Gans, J. S. (2006). Tivoed: The effects of ad-avoidance technologies on broadcaster behavior. Technical report, University of Virginia. Acesso em: 30 de maio de 2012.

Anderson, S. P. & Gans, J. S. (2010). Platform siphoning: Ad-avoidance and media content. In *CEPR Discussion Papers*. SSRN.

Azevedo, R. G. D. A. & Soares, L. F. G. (2012). Embedding 3d objects into ncl multimedia presentations. In *Proceedings of the 17th International Conference on 3D Web Technology*, Web3D '12, pages 143–151, New York, NY, USA. ACM.

Bagwell, K. (2007). The economic analysis of advertising. *Handbook of industrial organization*, 3:1701–1844.

Baladrón, C., Javier, A., Belén, C., Sánchez-Esguevillas, A., Baldauf, M., Fröhlich, P., Musialski, P., Falcarin, P., Rodriguez Rocha, O., Costabello, L., Goix, L.-W., Cadenas, A., Paganelli, F., Parlanti, D., Giuli, D., Pimentel, M. d. G. C., Cattelan, R. G., Melo, E. L., Teixeira, C. A. C., & Raibulet, C. (2008). Integrating user-generated content and pervasive communications. *IEEE Pervasive Computing*, 7(4):58–61.

Barker, S. K. & Shenoy, P. (2010). Empirical evaluation of latency-sensitive application performance in the cloud. In *Proceedings of the first annual ACM SIGMM conference on Multimedia systems*, MMSys '10, pages 35–46, New York, NY, USA. ACM.

Bauerlein, M. (2011). The chronicle of higher education - lectures on the benefits.

Beck, K. & Andres, C. (2004). *Extreme programming explained: embrace change*. Addison-Wesley Professional.

Bianchi, M. (2004). Automatic video production of lectures using an intelligent and aware environment. In *Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia*, MUM '04, pages 117–123, New York, NY, USA. ACM.

Boll, S. (2003). Mm4u - a framework for creating personalized multimedia content.

Boronat, F., Lloret, J., & García, M. (2009). Multimedia group and inter-stream synchronization techniques: A comparative study. *Information Systems*, 34(1):108 – 131.

Boronat, F., Montagud, M., Stokking, H. M., & Niamut, O. (2012). The need for inter-destination synchronization for emerging social interactive multimedia applications. *Communications Magazine, IEEE*, 50(11):150–158.

Bouyakoub, S. & Belkhir, A. (2011). Smil builder: An incremental authoring tool for smil documents. *ACM Trans. Multimedia Comput. Commun. Appl.*, 7(1):2:1–2:30.

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Brennan, M. & Syn, M. (2001). Television viewing behaviour during commercial breaks. In *2001 Australian & New Zealand Marketing Academy Conference (Anzmac 2001)*.

Brooks, C., Thompson, C., & Greer, J. (2013). Visualizing lecture capture usage: A learning analytics case study. In *Proc. Work. Analytics on Video-based Learning (WAVe)*.

Brotherton, J. A. & Abowd, G. D. (2004). Lessons learned from eclass: Assessing automated capture and access in the classroom. *ACM Trans. Comput.-Hum. Interact.*, 11(2):121–155.

Bulterman, D. C. A. (2003). Using SMIL to encode interactive, peer-level multimedia annotations. In *ACM DocEng'03 – Proceedings of the 2003 ACM Symposium on Document Engineering*, pages 32–41. ACM Press.

Bulterman, D. C. A. (2007). User-centered control *within* multimedia presentations. *Multimedia Syst.*, 12(4-5):423–438.

Bulterman, D. C. A., Grassel, G., Jansen, J., Koivisto, A., Layaida, N., Michel, T., Mullender, S., & Zucker, D. (2005). Synchronized Multimedia Integration Language (SMIL 2.1). *http://www.w3.org/TR/2005/REC-SMIL2-20051213/*.

Bulterman, D. C. A. & Rutledge, L. W. (2008). *SMIL 3.0: Flexible Multimedia for Web, Mobile Devices and Daisy Talking Books*. Springer Publishing Company, Incorporated, 2nd ed. edition.

Cabrer, M. R., Redondo, R. P. D., Vilas, A. F., Pazos Arias, J. J., & Duque, J. G. (2006). Controlling the Smart Home from TV. *IEEE Transactions on Consumer Electronics*, 52(2):421–429.

Canessa, E., Fonda, C., & Zennaro, M. (2009). One year of ictp diploma courses on-line using the automated eya recording system. *Comput. Educ.*, 53(1):183–188.

Canessa, E., Tenze, L., Fonda, C., & Zennaro, M. (2013). Apps for synchronized photo-audio recordings to support students. In *Proceedings of the LAK 2013 Workshop on Analytics on Video-based Learning*, pages 29–33.

Cattelan, R. G., Baldochi, L. A., & Maria da Graça, C. P. (2003). Experiences on building capture and access applications. In *In Proceedings of the 9th Brazilian Symposium on Multimedia and Hypermedia Systems*, pages 112–127.

Cattelan, R. G., Teixeira, C., Goularte, R., & Pimentel, M. D. G. C. (2008). Watch-and-comment as a paradigm toward ubiquitous interactive video editing. *ACM Trans. Multimedia Comput. Commun. Appl.*, 4(4):28:1–28:24.

Cazenave, F., Quint, V., & Roisin, C. (2011). Timesheets.js: when smil meets html5 and css3. In *Proceedings of the 11th ACM symposium on Document engineering*, DocEng '11, pages 43–52, New York, NY, USA. ACM.

Cerqueira Neto, J. R., Mesquita Santos, R. C., Soares Neto, C. S., & Teixeira, M. M. (2012). Eventline: representation of the temporal behavior of multimedia applications. In *Proceedings of the 18th Brazilian symposium on Multimedia and the web*, WebMedia '12, pages 215–222, New York, NY, USA. ACM.

Charland, A. & Leroux, B. (2011). Mobile application development: web vs. native. *Communications of the ACM*, 54(5):49–53.

Choi, J. P. (2006). Broadcast competition and advertising with free entry: Subscription vs. free-to-air. *Information Economics and Policy*, 18(2):181–196.

Chou, H.-P., Wang, J.-M., Fuh, C.-S., Lin, S.-C., & Chen, S.-W. (2010). Automated lecture recording system. In *System Science and Engineering (ICSSE), 2010 International Conference on*, pages 167 –172.

Coppens, T., Trappeniers, L., & Godon, M. (2004). AmigoTV: Towards a social TV experience. In *Proc. EuroITV'04*.

Corallo, A., Caputo, E., & Cisternino, V. (2007). Business modelling language: a framework supporting interoperability in cluster of smes. In *Digital EcoSystems and Technologies Conference, 2007. DEST '07. Inaugural IEEE-IES*, pages 107 –112. IEEE.

Costa, M., Correia, N., & Guimarães, N. (2002). Annotations as multiple perspectives of video content. In *MULTIMEDIA '02: Proc. ACM International Conference on Multimedia*, pages 283–286.

Costa, R. M. d. R., Moreno, M. F., Rodrigues, R. F., & Soares, L. F. G. (2006). Live editing of hypermedia documents. In *ACM DocEng'06 – Proceedings of* *the 2006 ACM Symposium on Document Engineering*, pages 165–172. ACM Press.

Crinon, R. J. (1997). The dsm-cc object carousel for broadcast data services. In *Consumer Electronics, 1997. Digest of Technical Papers. ICCE., International Conference on*, pages 246–247. IEEE.

Cruz, V. M., Moreno, M. F., & Soares, L. F. G. (2008). Ginga-ncl: implementaçao de referência para dispositivos portáteis. In *Proceedings of the 14th Brazilian Symposium on Multimedia and the Web*, pages 67–74. ACM.

César, P., Bulterman, D. C., & Jansen, A. (2006a). An architecture for end-user tv content enrichment. *Journal of Virtual Reality and Broadcasting*, 3(9):14.

César, P., Bulterman, D. C., & Jansen, J. (2009). Leveraging user impact: an architecture for secondary screens usage in interactive television. *Multimedia Systems Journal*, 15(3):127–142.

César, P., Bulterman, D. C. A., & Jansen, A. J. (2006b). The ambulant annotator: empowering viewer-side enrichment of multimedia content. In *ACM DocEng'06 – Proceedings of the 2006 ACM Symposium on Document Engineering*, pages 186–187. ACM Press.

César, P., Bulterman, D. C. A., & Jansen, A. J. (2008). Usages of the secondary screen in an interactive television environment: Control, enrich, share, and transfer television content. In *EuroITV'08 - European Interactive TV Conference (LNCS 5866)*, pages 168–177.

César, P., Bulterman, D. C. A., Obrenovic, Z., Ducret, J., & Cruz-Lara, S. (2007). An architecture for non-intrusive user interfaces for interactive digital television. In *EuroITV'07 (LNCS 4471)*, pages 11–20.

César, P. & Geerts, D. (2011). Past, present, and future of social tv: a categorization. In *Consumer Communications and Networking Conference (CCNC), 2011 IEEE*, pages 347–351. IEEE.

Damasceno, J., dos Santos, J., & Muchaluat-Saade, D. (2011). Editec: hypermedia composite template graphical editor for interactive tv authoring. In *Proceedings of the 11th ACM symposium on Document engineering*, DocEng '11, pages 77–80, New York, NY, USA. ACM.

Danaher, P. J. (1995). What happens to television ratings during commercial breaks? *Journal of Advertising Research.*

Dao, M.-S. & Babaguchi, N. (2010). A new spatio-temporal method for event detection and personalized retrieval of sports video. *Multimedia Tools and Applications*, 50(1):227–248.

Darnell, M. J. (2008). Making digital tv easier for less-technically-inclined people. In *UXTV'08: Proceedings of the 1st International Conference on Designing Interactive User Experiences for TV and video*, pages 27–30. ACM.

Dey, A. K. (2001). Understanding and using context. *Personal and ubiquitous computing*, 5(1):4–7.

Dias, D. R. C., Brega, J. R. F., de Paiva Guimarães, M., Modesto, F., Gnecco, B. B., & Lauris, J. R. P. (2011). 3d semantic models for dental education. In *ENTERprise Information Systems*, pages 89–96. Springer.

Dickson, P. E., Arbour, D. T., Adrion, W. R., & Gentzel, A. (2010). Evaluation of automatic classroom capture for computer science education. In *Proc. Innovation and technology in computer science education*, ITiCSE '10, pages 88–92, New York, NY, USA. ACM.

Dickson, P. E., Warshow, D. I., Goebel, A. C., Roache, C. C., & Adrion, W. R. (2012). Student reactions to classroom lecture capture. In *Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education*, ITiCSE '12, pages 144–149, New York, NY, USA. ACM.

Digital, C. (2012). Tv tem a maior influência nos trending topics do twitter. http://convergenciadigital.uol.com.br/cgi/cgilua.exe/sys/start.htm?infoid=31341&sid=4. Acesso em: 29 de Agosto de 2014.

Dzhambazov, G. (2009). Comparisong: Audio comparison engine. *International Book Series*, page 33.

ECMA (1999). Ecmascript language specification.

El-Adly, M. I. & Aicinena, S. (2010). The impact of advertising attitudes on the intensity of tv ads avoiding behavior. *International Journal of Business and Social Science*, 1(1):9–22.

Elliott, B. & Özsoyoglu, Z. M. (2008). Annotation suggestion and search for personal multimedia objects on the web. In *CIVR'08: Proceedings of the 2008 International Conference on Content-based Image and Video Retrieval*, pages 75–84. ACM.

ETSI, T. (2005). 102 819 v1. 3.1 (2005) - digital video broadcasting (dvb): Globally executable mhp (gem) specification 1.0. 2. *European Telecommunications Standards Institute, TS*, 102(819):V1.

Ferreira, G., Nogueira, G., Comarela, G., Fabris, F., Martinello, M., & P Filho, J. (2010). Ginga-ncl em dispositivos portáteis: Uma implementação para a plataforma android. In *Proc. Brazilian Symposium on Multimedia and Web (WebMedia)*.

Fleury, C., Duval, T., Gouranton, V., & Arnaldi, B. (2010). Architectures and mechanisms to efficiently maintain consistency in collaborative virtual environments. *Proc. of Software Engineering and Architectures for Realtime Interactive Systems-SEARIS*, pages 87–94.

Frankel, D. (2009). End of the 2 minute commercial break. http://www.thewrap.com/tv/article/standard-ad-pods-disappearing-tv_3733. Acesso em: 30 de maio de 2012.

Freitas, G. B. & Teixeira, C. A. C. (2009). Ubiquitous services in home networks offered through digital tv. In *Proceedings of the 2009 ACM symposium on Applied Computing*, pages 1834–1838. ACM.

Gaggi, O. & Danese, L. (2011). A smil player for any web browser. In *Proceedings of International Conference on Distributed Multimedia Systems*, pages 114–119, Firenze, Italy.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design patterns: elements of reusable object-oriented software*. Pearson Education.

Gaspar, T. C., do Prado, A. F., & Teixeira, C. A. C. (2009). Software product lines for web 2.0 synchronous collaboration. In *Proceedings of the XV Brazilian Symposium on Multimedia and the Web*, WebMedia '09, pages 11:1–11:8, New York, NY, USA. ACM.

Geerts, D., Vaishnavi, I., Mekuria, R., van Deventer, O., & César, P. (2011). Are we in sync?: synchronization requirements for watching online video together. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 311–314, New York, NY, USA. ACM.

Ginsburgh, V. A. & Throsby, D. (2006). *Handbook of the Economics of Art and Culture*, volume 1. Elsevier.

Gondo, S. & Sakamoto, N. (2011). Multiple screen scenario - aditional use case and key issues. http://www.w3.org/2011/09/webtv/papers/Position_

Paper_for_The_third_W3C_Web_and_TV_Workshop__TOSHIBA_.pdf. Acesso em: 30 de maio de 2012.

Goularte, R., Cattelan, R. G., Camacho-Guerrero, J. A., Inácio, Jr., V. R., & Pimentel. Maria da Graça, C. (2004). Interactive multimedia annotations: enriching and extending content. In *DocEng '04: Proc. ACM Symposium on Document Engineering*, pages 84–86.

Guimarães, R. L., Costa, R. M. R., & Soares, L. F. G. (2008). Composer: Authoring Tool for iTV Programs. In *EuroITV'08 (LNCS 5066)*, pages 61–71.

H.761, R. I.-T. (2009). Nested context language (ncl) and ginga-ncl for iptv services. Technical report, ITU-T.

Halawa, S., Pang, D., Cheung, N.-M., & Girod, B. (2011). Classx: an open source interactive lecture streamingsystem. In *Proceedings of the 19th ACM international conference on Multimedia*, MM '11, pages 719–722, New York, NY, USA. ACM.

Harter, A., Hopper, A., Steggles, P., Ward, A., & Webster, P. (2002). The anatomy of a context-aware application. *Wireless Networks*, 8(2/3):187–197.

Hölbling, G., Rabl, T., Coquil, D., & Kosch, H. (2008). Interactive tv services on mobile devices. *IEEE MultiMedia*, 15(2):72–76.

Holub, P., Matyska, L., Liška, M., Hejtmánek, L., Denemark, J., Rebok, T., Hutanu, A., Paruchuri, R., Radil, J., & Hladká, E. (2006). High-definition multimedia for multiparty low-latency interactive communication. *Future Generation Computer Systems*, 22(8):856 – 861.

Hua, X.-S., Lu, L., & Zhang, H.-J. (2005). Robust learning-based tv commercial detection. In *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*, pages 4–pp. IEEE.

Huang, E. M., Harboe, G., Tullio, J., Novak, A., Massey, N., Metcalf, C. J., & Romano, G. (2009). Of social television comes home: a field study of communication choices and practices in tv-based text and voice chat. In *CHI'09: Proc. ACM International Conference on Human Factors in Computing Systems*, pages 585–594.

Hupp, D. & Miller, R. C. (2007). Smart bookmarks: automatic retroactive macro recording on the web. In *UIST '07: Proceedings of the 20th annual ACM symposium on User interface software and technology*, pages 81–90. ACM.

Hürst, W. (2003). Indexing, searching, and skimming of multimedia documents containing recorded lectures and live presentations. In *Proceedings of 11th ACM International Conference on Multimedia*, MULTIMEDIA '03, pages 450–451, New York, NY, USA. ACM.

ISO/IEC International Organization for Standardization (1998). *13818-6:1998, Generic Coding of Moving Pictures and Associated Audio Information - Part 6: Extensions for DSM-CC.*

Jansen, J. & Bulterman, D. C. (2008). Enabling adaptive time-based web applications with smil state. In *Proceeding of the eighth ACM symposium on Document engineering*, pages 18–27. ACM.

Jansen, J. & Bulterman, D. C. (2009). Smil state: an architecture and implementation for adaptive time-based web applications. *Multimedia Tools and Applications*, 43(3):203–224.

Jansen, J., César, P., & Bulterman, D. C. (2010). A model for editing operations on active temporal multimedia documents. In *Proceedings of the 10th ACM symposium on Document engineering*, DocEng '10, pages 87–96, New York, NY, USA. ACM.

JavaDTV, A. (2010). Java dtv api 1.3 specification, sun microsystems (2009).

Jurgelionis, A., Fechteler, P., Eisert, P., Bellotti, F., David, H., Laulajainen, J.-P., Carmichael, R., Poulopoulos, V., Laikari, A., Perälä, P., et al. (2009). Platform for distributed 3d gaming. *Int. J. Comput. Games Technol.*, 2009:1:1–1:15.

Kegel, I., César, P., Jansen, J., Bulterman, D. C., Stevens, T., Kort, J., & Färber, N. (2012). Enabling 'togetherness' in high-quality domestic video. In *Proceedings of the 20th ACM international conference on Multimedia*, MM '12, pages 159–168, New York, NY, USA. ACM.

Kent, R. J. (2013). Switching before the pitch: Exploring television channel changing before the ads even start. *Journal of Marketing Communications*, 19(5):377–386.

King, P., Schmitz, P., & Thompson, S. (2004). Behavioral reactivity and real time programming in xml: functional programming meets smil animation. In *Proceedings of the 2004 ACM Symposium on Document Engineering*, DocEng '04, pages 57–66, New York, NY, USA. ACM.

Lampi, F., Kopf, S., & Effelsberg, W. (2008). Automatic lecture recording. In *Proceedings of the 16th ACM international conference on Multimedia*, MM '08, pages 1103–1104, New York, NY, USA. ACM.

Lanir, J., Booth, K. S., & Tang, A. (2008). Multipresenter: a presentation system for (very) large display surfaces. In *Proceedings of the 16th ACM International Conference on Multimedia*, MULTIMEDIA '08, pages 519–528, New York, NY, USA. ACM.

Liebowitz, S. J. & Margolis, S. E. (1998). Network externalities (effects). *The New Palgrave's Dictionary of Economics and the Law*.

Lienhard, J. & Lauer, T. (2002). Multi-layer recording as a new concept of combining lecture recording and students' handwritten notes. In *Proceedings of the 10th ACM International Conference on Multimedia*, MULTIMEDIA '02, pages 335–338, New York, NY, USA. ACM.

Lin, J., Drake, J., Kim, H., & Song, E. (2012). A framework-based approach for interactive multimedia application development. In *Proceedings of the 2012 ACM Research in Applied Computation Symposium*, RACS '12, pages 364–370, New York, NY, USA. ACM.

Liu, T. & Kender, J. R. (2004). Lecture videos for e-learning: current research and challenges. In *Multimedia Software Engineering, 2004. Proceedings. IEEE Sixth International Symposium on*, pages 574 – 578.

Lucena, V. F., Filho, J. E. C., Viana, N.S., & Maia, O. (2009). A home automation proposal built on the ginga digital tv middleware and the osgi framework. *IEEE Transactions on Consumer Electronics*, 55(3):1254–1262.

Luo, H., Gao, Y., Xue, X., Peng, J., & Fan, J. (2008). Incorporating feature hierarchy and boosting to achieve more effective classifier training and concept-oriented video summarization and skimming. *ACM Trans. Multimedia Comput. Commun. Appl.*, 4(1):1–25.

Lux, M., Marques, O., Schoffmann, K., Boszormenyi, L., & Lajtai, G. (2010). A novel tool for summarization of arthroscopic videos. *Multimedia Tools and Applications*, 46(2–3):521–544.

Maria da Graça, C. P., Goularte, R., Cattelan, R. G., Santos, F. S., & Teixeira, C. (2008). Ubiquitous Interactive Video Editing via Multimodal Annotations. In *EuroITV'08 (LNCS 5066)*, pages 72–81.

Marques Neto, M. C. & Santos, C. A. (2008). An event-based model for interactive live tv shows. In *Proceedings of the 16th ACM international conference on Multimedia*, MM '08, pages 845–848, New York, NY, USA. ACM.

Marshall, C. C. (1998). Toward an ecology of hypertext annotation. In *Proceedings of the 9th ACM Conference on Hypertext and Hypermedia: links, objects, time and space – structure in hypermedia systems*, pages 40–49. ACM.

Mauve, M., Vogel, J., Hilt, V., & Effelsberg, W. (2004). Local-lag and timewarp: Providing consistency for replicated continuous applications. *Multimedia, IEEE Transactions on*, 6(1):47–57.

McGreal, R. (2004). Learning objects: A practical definition. *International Journal of Instructional Technology and Distance Learning*, 1(9):21–32.

McKay, C., Fujinaga, I., & Depalle, P. (2005). jaudio: A feature extraction library. In *Proceedings of the International Conference on Music Information Retrieval*, pages 600–3.

Meixner, B. & Kosch, H. (2012). Interactive non-linear video: definition and xml structure. In *Proceedings of the 2012 ACM Symposium on Document Engineering*, DocEng '12, pages 49–58, New York, NY, USA. ACM.

Meixner, B., Köstler, J., & Kosch, H. (2011). A mobile player for interactive non-linear video. In *Proceedings of the 19th ACM international conference on Multimedia*, MM '11, pages 779–780, New York, NY, USA. ACM.

Meixner, B., Siegel, B., Hölbling, G., Lehner, F., & Kosch, H. (2010). Siva suite: authoring system and player for interactive non-linear videos. In *Proceedings of the international conference on Multimedia*, MM '10, pages 1563–1566, New York, NY, USA. ACM.

Melo, E. L., Viel, C. C., Teixeira, C. A. C., Rondon, A. C., Silva, D. d. P., Rodrigues, D. G., & Silva, E. C. (2012). WebNCL: a web-based presentation machine for multimedia documents. In *Proc. Brazilian symposium on Multimedia and the web*, WebMedia '12, pages 403–410. ACM.

Monahan, B. (2011). When it comes to ad avoidance, the dvr is not the problem. http://adage.com/article/adagestat/smartphones-a-bigger-distraction-dvrs/227725/. Acesso em: 30 de maio de 2012.

Moreno, M. F., Batista, C., & Soares, L. F. G. (2010). Ncl and itu-t's standardization effort on multimedia application frameworks for iptv. *ACM, October*.

Müller, R. & Ottmann, T. (2000). The "authoring on the fly" system for automated recording and replay of (tele)presentations. *Multimedia Systems*, 8(3):158–176.

Nagai, T. (2009). Automated lecture recording system with avchd camcorder and microserver. In *Proceedings of the 37th annual ACM SIGUCCS fall conference*, SIGUCCS '09, pages 47–54, New York, NY, USA. ACM.

Nathan, M., Harrison, C., Yarosh, S., Terveen, L. G., Stead, L., & Amento, B. (2008). Collaboratv: making television viewing social again. In *UXTV'08: Proc. International Conference on Designing Interactive User Experiences for TV and Video*, pages 85–94. ACM.

Nitta, N., Takahashi, Y., & Babaguchi, N. (2009). Automatic personalized video abstraction for sports videos using metadata. *Multimedia Tools and Applications*, 41(1):1–25.

Omojokun, O., Pierce, S., Isbell, L., & Dewan, P. (2006). Comparing end-user and intelligent remote control interface generation. *Personal Ubiquitous Comput.*, 10(2-3):136–143.

Page, W. H. & Lopatka, J. E. (1999). Network externalities. *Encyclopedia of law and economics*, 760:952–980.

Pang, D., Halawa, S., Cheung, N.-M., & Girod, B. (2011). Classx mobile: region-of-interest video streaming to mobile devices with multi-touch interaction. In *Proceedings of the 19th ACM international conference on Multimedia*, MM '11, pages 787–788, New York, NY, USA. ACM.

Patel, M., Gossweiler, R., Sahami, M., Blackburn, J., Brown, D., & Knight, A. (2008). Google tv search: dual-wielding search and discovery in a large-scale product. In *UXTV '08: Proceeding of the 1st International Conference on Designing Interactive User Experiences for TV and Video*, pages 95–104, New York, NY, USA. ACM.

Pedrosa, D., Martins, Jr., J. A. C., Melo, E. L., & Teixeira, C. A. C. (2011). A multimodal interaction component for digital television. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, SAC '11, pages 1253–1258, New York, NY, USA. ACM.

Peitz, M. & Valletti, T. M. (2008). Content and advertising in the media: Pay-tv versus free-to-air. *International Journal of Industrial Organization*, 26(4):949–965.

Petersen, M. K. & Butkus, A. (2008). Modeling emotional context from latent semantics. In *UXTV '08: Proceeding of the 1st International Conference on Designing Interactive User Experiences for TV and Video*, pages 63–66, New York, NY, USA. ACM.

Pimentel, M. d. G. C., Baldochi, L. A., & Cattelan, R. G. (2007a). Prototyping applications to document human experiences. *IEEE Pervasive Computing*, 6(2):93–100.

Pimentel, M. d. G. C., Cattelan, R. G., Freitas, G., Melo, E. L., & Teixeira, C. A. C. (2009). Watch-and-comment as an approach to collaborative annotate points of interest in video and interactive-TV programs. In Marcus, A., Roibás, A. C., & Sala, R., editors, *Mobile TV: Customizing Content and Experience*, pages 349–366. Springer.

Pimentel, M. d. G. C., Cattelan, R. G., Melo, E. L., Prado, A. F., & Teixeira, C. A. C. (2008a). Ubiquitous end-user live editing of interactive multimedia programs. In *WebMedia '08: Proceedings of the 14th Brazilian Symposium on Multimedia and the Web*, pages 123–129, New York, NY, USA. ACM.

Pimentel, M. d. G. C., Cattelan, R. G., Melo, E. L., Prado, A. F., & Teixeira, C. A. C. (2010). End-user live editing of iTV programmes. *Int. J. Adv. Media Commun.*, 4(1):78–103.

Pimentel, M. d. G. C., Cattelan, R. G., Melo, E. L., & Teixeira, C. A. (2008b). End-user editing of interactive multimedia documents. In *DocEng'08: Proc. ACM Symposium on Document Engineering*, pages 298–301. ACM.

Pimentel, M. d. G. C., Goularte, R., Cattelan, R. G., Santos, F. S., & Teixeira, C. A. C. (2007b). Enhancing multimodal annotations with pen-based information. In *Workshop on New Techniques for Consuming, Managing, and Manipulating Interactive Digital Media at Home*, volume 2, pages 207–212.

Quek, F., Bryll, R., Kirbas, C., Arslan, H., & McNeill, D. (2002). A multimedia system for temporally situated perceptual psycholinguistic analysis. *Multimedia Tools and Applications*, 18(2):91–114.

Rahman, M. A., Hossain, M. A., & Saddik, A. (2004). Lornav: A demo of a virtual reality tool for navigation and authoring of learning object repositories. In *Distributed Simulation and Real-Time Applications, 2004. DS-RT 2004. Eighth IEEE International Symposium on*, pages 240 – 243.

Ramos, G. & Balakrishnan, R. (2003). Fluid interaction techniques for the control and annotation of digital video. In *UIST '03: Proc. ACM Symposium on User Interface Software and Technology*, pages 105–114.

Repplinger, M., Löffler, A., Schug, B., & Slusallek, P. (2009). Extending x3d for distributed multimedia processing and control. In *Proceedings of the 14th International Conference on 3D Web Technology*, Web3D '09, pages 61–69, New York, NY, USA. ACM.

Rice, M. & Alm, N. (2008). Designing new interfaces for digital interactive television usable by older adults. *Comput. Entertain.*, 6(1):1–20.

Rochet, J.-C. & Tirole, J. (2003). Platform competition in two-sided markets. *Journal of the European Economic Association*, 1(4):990–1029.

Rodrigues, K. R. d. H., Pereira, S. d. S., Pimentel, M. d. G. C., & Teixeira, C. A. C. (2012). Find: facilitating the identification of intervals and moments for incorporation of additional content in continuous media. In *Proceedings of the 18th Brazilian symposium on Multimedia and the web*, WebMedia '12, pages 265–268, New York, NY, USA. ACM.

Rodrigues, R. F. & Soares, L. F. G. (2003). Inter and intra media-object qos provisioning in adaptive formatters. In *ACM DocEng'03: Proceedings of the 2003 ACM symposium on Document Engineering*, pages 78–87. ACM.

Rogers, T. (2008). Tv industry faces ad avoidance crisis more severe than financial crisis. http://www.jackmyers.com/jackmyers-think-tank/31599939.html. Acesso em: 30 de maio de 2012.

Rojas-Mendez, J. I., Davies, G., & Madran, C. (2009). Universal differences in advertising avoidance behavior: A cross-cultural study. *Journal of Business Research*, 62(10):947–954.

Ronchetti, M. (2013). Videolectures ingredients that can make analytics effective. In *Proc. WAVe'2013*.

Sakamoto, N., Muguruma, K., Koshino, N., Chiba, S., & Sakurai, M. (2005). A digital HDTV receiver with home networking function and digital content storage. In *ICCE'05: International Conf. Consumer Electronics - Digest of Technical Papers.*, pages 39–40. IEEE.

Santos, J. A. & Muchaluat-Saade, D. C. (2012). Xtemplate 3.0: spatio-temporal semantics and structure reuse for hypermedia compositions. *Multimedia Tools Appl.*, 61(3):645–673.

Sarosi, G. (2011). User interface development for smarttv using web technology and cea2014. http://www.w3.org/2011/09/webtv/papers/webandtvSept2011-twc-v1.0.pdf. Acesso em: 30 de maio de 2012.

SBTVD (2006). Brazilian Digital TV System Reference Model. *http://sbtvd.cpqd.com.br*.

Schmitz, P. (2002). Multimedia meets computer graphics in smil2.0: a time model for the web. In *Proceedings of the 11th international conference on World Wide Web*, WWW '02, pages 45–53, New York, NY, USA. ACM.

Schnell, M., Schmidt, M., Jander, M., Albert, T., Geiger, R., Ruoppila, V., Ekstrand, P., & Bernhard, G. (2008). Mpeg-4 enhanced low delay aac - a new standard for high quality communication. In *Audio Engineering Society Convention 125*.

Schulte, O. A., Wunden, T., & Brunner, A. (2008). Replay: an integrated and open solution to produce, handle, and distributeaudio-visual (lecture) recordings. In *Proceedings of the 36th annual ACM SIGUCCS fall conference: moving mountains, blazing trails*, SIGUCCS '08, pages 195–198, New York, NY, USA. ACM.

Schwaber, K. & Beedle, M. (2001). *Agile software development with Scrum*, volume 18. Prentice Hall.

Schwerdt, G. & Wuppermann, A. C. (2011). Sage on the stage: Is lecturing really all that bad? *Education Next*, 11(3):62–67.

Silva, H. V. O., Rodrigues, R. F., Soares, L. F. G., & MuchaluatSaade, D. C. (2004). NCL 2.0: integrating new concepts to XML modular languages. In *ACM DocEng'04: Proceedings of the 2004 ACM symposium on Document engineering*, pages 188–197. ACM.

Soares, L. F. G. & Barbosa, S. D. J. (2009). *Programando em NCL 3.0: desenvolvimento de aplicações para middleware Ginga: TV digital e Web*. Elsevier.

Soares, L. F. G., Moreno, M. F., & De Salles Soares Neto, C. (2010a). Ginga-ncl: Declarative middleware for multimedia iptv services. *Communications Magazine, IEEE*, 48(6):74–81.

Soares, L. F. G., Rodrigues, & Rodrigues, R. F. (2006). Nested context language 3.0 part 8–ncl digital tv profiles. *Monografias em Ciência da Computação do Departamento de Informática da PUC-Rio*, 1200(35):06.

Soares, L. F. G., Rodrigues, R. F., Cerqueira, R., & Barbosa, S. D. (2010b). Variable and state handling in ncl. *Multimedia Tools Appl.*, 50(3):465–489.

Soares, L. F. G., Rodrigues, R. F., & Moreno, M. F. (2007). Ginga-ncl: the declarative environment of the brazilian digital tv system. *Journal of the Brazilian Computer Society*, 12(4):37–46.

Soares, L. F. G. & Souza Filho, G. L. d. (2007). Interactive Television in Brazil: System Software and the Digital Divide. In *EuroITV'07: Proceedings of the European Conference on European Interactive Television Conference (LNCS 4417)*, pages 41–44. Springer.

Soares, L. P. (2005). *Um Ambiente de multiprojeção totalmente imersivo baseado em aglomerados de computadores*. PhD thesis, USP, São Paulo.

Soares Neto, C. S., Pinto, H. F., & Soares, L. F. G. (2012). Tal processor for hypermedia applications. In *Proceedings of the 2012 ACM symposium on Document engineering*, DocEng '12, pages 69–78, New York, NY, USA. ACM.

Souza, D. F., Tavares, T. A., Machado, L. S., & Souza Filho, G. L. (2010). Incorporating 3d technologies to the brazilian dtv standard: a study of integration strategies based on middleware ginga. In *Proceedings of the 8th international interactive conference on Interactive TV&#38;Video*, EuroITV '10, pages 251–258, New York, NY, USA. ACM.

Stuart, G. (2008). You can't avoid ad avoidance. http://www.adweek.com/news/advertising-branding/you-cant-avoid-ad-avoidance-96852. Acesso em: 30 de maio de 2012.

Stühmeier, T. & Wenzel, T. (2011). Getting beer during commercials: Adverse effects of ad-avoidance. *Information Economics and Policy*, 23(1):98–106.

Teixeira, C. A., Melo, E. L., Freitas, G. B., Santos, C. A. S., & Pimentel, Maria da Graça, C. (2012). Discrimination of media moments and media intervals: sticker-based watch-and-comment annotation. *Multimedia Tools and Applications*, 61(3):675–696.

Teixeira, C. A. C., Freitas, G., & Pimentel, M. d. G. C. (2010a). Distributed discrimination of media moments and media intervals: a Watch-and-Comment approach. In *SAC'10: Proceedings of the 2010 ACM Symposium on Applied Computing*, pages 1929–1935.

Teixeira, C. A. C., Melo, E. L., Cattelan, R. G., & Maria da Graça, C. P. (2010b). Taking advantage of contextualized interactions while users watch tv. *Multimedia Tools and Applications*, 50(3):587–607.

Tkachenko, D., Kornet, N., & Kaplan, A. (2004). Convergence of iDTV and home network platforms. In *IEEE Consumer Communications and Networking Conference*, pages 624–626.

Tokan, F. et al. (2011). *Media as multitasking: An exploratory study on capturing audiences' media multitasking and multiple media use behaviours*. PhD thesis, Aalto University.

Tonndorf, K., Knieper, T., Meixner, B., Kosch, H., & Lehner, F. (2012). Challenges in creating multimedia instructions for support systems and dynamic problem-solving. In *Proceedings of the 12th International Conference on Knowledge Management and Knowledge Technologies*, i-KNOW '12, pages 33:1–33:4, New York, NY, USA. ACM.

Tse, A. C. B. & Lee, R. P. (2001). Zapping behavior during commercial breaks. *Journal of Advertising Research*, 41(3):25–30.

Ursu, M. F., Thomas, M., Kegel, I., Williams, D., Tuomola, M., Lindstedt, I., Wright, T., Leurdijk, A., Zsombori, V., Sussner, J., Myrestam, U., & Hall, N. (2008). Interactive tv narratives: Opportunities, progress, and challenges. *ACM Trans. Multimedia Comput. Commun. Appl.*, 4(4):1–39.

Vaishnavi, I., César, P., Bulterman, D. C. A., Friedrich, O., Gunkel, S., & Geerts, D. (2011). From iptv to synchronous shared experiences challenges in design: Distributed media synchronization. *Sig. Proc.: Image Comm.*, pages 370–377.

Valin, J.-M., Terriberry, T. B., & Maxwell, G. (2009). A full-bandwidth audio codec with low complexity and very low delay. In *17th European Signal Processing Conference. Glasgow, Scotland.*

Vega-Oliveros, D. A., Martins, D. S., & Pimentel, M. d. G. C. (2010). "this conversation will be recorded": automatically generating interactive documents from captured media. In *Proceedings of the 10th ACM symposium on Document engineering*, DocEng '10, pages 37–40, New York, NY, USA. ACM.

Viel, C. C., Melo, E. L., Godoy, A. P., Dias, D. R. C., Trevelin, L. C., & Teixeira, C. A. C. (2012). Multimedia presentation integrating interactive media produced in real time with high performance processing. In *Proceedings of the 18th Brazilian symposium on Multimedia and the web*, WebMedia '12, pages 115–122, New York, NY, USA. ACM.

Viel, C. C., Melo, E. L., Pimentel, Maria da Graça, C., & Teixeira, C. A. C. (2013a). How are they watching me: learning from student interactions with multimedia objects captured from classroom presentations. In *Proc. International Conference on Enterprise Information Systems*, ICEIS '13.

Viel, C. C., Melo, E. L., Pimentel, Maria da Graça, C., & Teixeira, C. A. C. (2013b). Presentations preserved as interactive multi-video objects. In *Proc. Workshop on Analytics on Video-Based Learning*.

Viel, C. C., Melo, E. L., Pimentel, M. d. G. C., & Teixeira, C. A. (2013c). Multimedia multi-device educational presentations preserved as interactive multi-video objects. In *Proceedings of the 19th Brazilian symposium on Multimedia and the web*, pages 51–58. ACM.

Viel, C. C., Melo, E. L., Pimentel, M. d. G. C., & Teixeira, C. A. C. (2013d). Go beyond boundaries of itv applications. In *Proc. ACM Symposium on Document Engineering (DocEng)*, pages 263–272. ACM.

Viel, C. C., Melo, E. L., Trevelin, L., & Teixeira, C. A. C. (2011). Rv-mtv: Framework para interaçao multimodal com aplicaçoes de realidade virtual em tv digital e dispositivos móveis. In *Proceedings of the 17th Brazilian symposium on Multimedia and the web*.

Vuorimaa, P. (2007). Timesheets javascript engine.

Vuorimaa, P., Bulterman, D., & Cesar, P. (2008). Smil timesheets 1.0. *W3C Working Draft*.

Wallis, C. (2006). The multitasking generation. *Time Magazine*, 167(13):48–55.

Weibel, N., Norrie, M. C., & Signer, B. (2007). A model for mapping between printed and digital document instances. In *ACM DocEng'07 – Proceedings of the 2006 ACM Symposium on Document Engineering*, pages 19–28.

Weiser, M. (1991). The computer for the 21st century. *Scientific American*, 265(3):94–104.

Weisz, J. D., Kiesler, S., Zhang, H., Ren, Y., Kraut, R. E., & Konstan, J. A. (2007). Watching together: integrating text chat with video. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, pages 877–886, New York, NY, USA. ACM.

Wilbur, K. C. (2008). A two-sided, empirical model of television advertising and viewing markets. *Marketing Science*, 27(3):356–378.

Yi, H., Rajan, D., & Chia, L.-T. (2005). Automatic generation of mpeg-7 compliant xml document for motion trajectory descriptor in sports video. *Multimedia Tools and Applications*, 26(2):191–206.

Zhou, W. (2004). The choice of commercial breaks in television programs: the number, length and timing. *The Journal of Industrial Economics*, 52(3):315–326.