5
Aspectos de Implementação

Neste capítulo são detalhados os aspectos técnicos da implementação da arquitetura proposta. Todos os componentes foram desenvolvidos na plataforma Microsoft .NET com a linguagem C#. Apesar da adoção da tecnologia Microsoft, o CAS foi modularizado e construído de forma que um componente pudesse ser substituído por outro em qualquer linguagem compatível com o .NET Framework, desde que obedecido ao contrato definido.

5.1.
Modelo de Comunicação

Independentemente de o componente desenvolvido ser um SpeedCar ou um CASIn, obrigatoriamente ele segue o modelo de comunicação adotado, que foi o do *Windows Communication Foundation* (WCF), introduzido pelo .NET Framework 3.0[20] da Microsoft. Apesar de todos os componentes desse trabalho terem sido desenvolvidos em C# com o WCF, eles podem ser reimplementados em qualquer outra linguagem que tenha integração com o .Net. Linguagens com suporte à máquina de execução do .NET, como o Delphi.NET por exemplo, podem utilizar a própria biblioteca de classes do WCF. A interoperabilidade com outras linguagens não suportadas pelo framework .NET, como o Java EE, também é garantida já que os serviços do WCF interagem por meio do protocolo SOAP[21]. Nestes casos as diretrizes que serão apresentadas a seguir podem ser diferentes.

O modelo de programação WCF unifica uma vasta gama de tecnologias como Web Services, .NET Remoting, DCOM e Message Queues em um modelo único de programação orientada a serviço para computação distribuída [Chap07]. Os serviços são definidos como entidades autônomas, apesar de existir um acordo sobre a interface de comunicação entre eles e os clientes. A comunicação entre

[20] http://msdn.microsoft.com/netframework/

processos WCF pode ser realizada dentro da mesma máquina, numa rede ou até pela internet. A hospedagem do serviço pode ser provida pelo servidor *web* do Windows, chamado *Internet Information Services* (IIS); por um servidor específico para esse tipo de tecnologia, disponível nas novas versões do Windows, como o Windows Vista e Windows 2008 Server, chamado *Windows Activation Service* (WAS); ou ainda de forma autocontida, usando a classe *ServiceHost*, permitindo que a aplicação construída baseada em WCF seja totalmente independente de um servidor. Graças ao projeto Mono[22], que visa portar o Microsoft .NET Framework para o Linux, uma aplicação WCF pode inclusive ser hospedada num servidor *web* Apache[23] sem necessidade de modificações no código.

Um serviço WCF é composto de três partes como ilustrado na Figura 5.1: uma classe de serviço (*Service Class*), que implementa de fato o serviço provido; um ambiente de hospedagem (*Host Process Environment*); e um ou mais *Endpoints*, nos quais os clientes se conectam e por meio dos quais toda a comunicação ocorre. Os *Endpoints* especificam contratos que definem quais métodos do serviço estarão disponíveis, sendo que cada um pode expor um conjunto diferente de métodos. Eles também definem *bindings*, que são a especificação da forma e do endereço de comunicação do cliente.

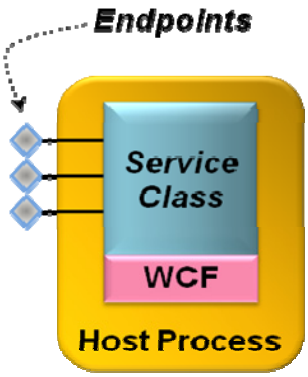

Figura 5.1: Estrutura de um serviço WCF [Chap07].

[21] http://www.w3.org/TR/soap/
[22] http://www.mono-project.com
[23] http://www.apache.org

Uma classe de serviço da WCF implementa alguns serviços como um conjunto de métodos. Ela também deve ter pelo menos um contrato de serviço (*Service Contract*), o qual define as operações que o serviço pode executar. Opcionalmente, podem ser definidos contratos de dados (*Data Contracts*) que descrevem os tipos de dados manipulados pelas operações expostas. Na Listagem 5.1, observa-se um exemplo de um contrato de serviço WCF em C#, que nada mais é do que uma interface com atributos específicos. Qualquer linguagem .NET, apesar de mudar a sintaxe, possuirá os mesmos atributos modificadores.

```
[ServiceContract]
interface ISampleContract
{
    [DataContract]
    struct DataSample
    {
        [DataMember]
        public int i;
        [DataMember]
        public string s;
    }

    [OperationContract]
    bool Boo(int a);

    [OperationContract]
    int Foo(int b);
}
```

Listagem 5.1: Exemplo de um contrato do WCF.

5.2. Componentes Desenvolvidos

Para permitir a extensibilidade do CAS, foram definidos contratos para cada tipo de componente suportado. Nas próximas subseções serão apresentados os aspectos técnicos de cada componente desenvolvido, os principais contratos e as regras que devem ser seguidas em alguns casos específicos.

5.2.1. SpeedCar

Para um componente ser um SpeedCar, ele deve obedecer a três regras básicas:

1. Implementar o contrato de serviço definido na Listagem 5.2.

```
[ServiceContract]
interface ISpeedCar
{
    [OperationContract]
    bool StartCaptureNow(int eventID, string EventPath);

    [OperationContract]
    bool StopCaptureNow();
 }
```

Listagem 5.2: Contrato de serviço dos SpeedCars.

2. Gravar a mídia capturada em uma subpasta dentro da pasta do evento (definida por *EventPath*). Foi uma decisão do projeto não serializar simplesmente os objetos de mídia, pois estes podem ser muito grandes, como acontece com os vídeos. Em contrapartida, introduziu-se a necessidade de todos os componentes, que podem estar distribuídos, gravarem em uma área de rede compartilhada.

3. Gravar junto com a mídia o arquivo SpeedCar.xml com informações gerais da captura e detalhes sobre as mídias. O exemplo da Listagem 5.3 ilustra a gravação de um único fluxo de áudio no dia 20/12/2007 às 14:00:30 para o evento com identificador 31. Os tipos de mídia (*media type*) aceitos no arquivo XML são: *video*, *audio* e *application/vnd.ms-powerpoint*. Estes são um pequeno subgrupo dos tipos MIME (*Multipurpose Internet Mail Extensions*)[24] associados aos SpeedCars desenvolvidos. Com o desenvolvimento de novos SpeedCars espera-se o aumento dos tipos MIME aceitos. A *tag src* indica o nome do arquivo, e a *tag seq* a ordem, caso tenham várias mídias capturadas num mesmo evento.

```
<?xml version="1.0"?>
<CAS>
  <SpeedCar>
    <CaptureInfo>
      <EventID>31</EventID>
    </CaptureInfo>
    <CaptureMedia>
      <Media type="audio" src="audio.wav" seq="1">
        <StartDateTime>20/12/2007 14:00:30</StartDateTime>
```

[24] http://www.iana.org/assignments/media-types/application/

```
      </Media>
    </CaptureMedia>
  </SpeedCar>
</CAS>
```

Listagem 5.3: Exemplo de XML gerado na gravação de um áudio.

Foram desenvolvidos três SpeedCars para esta versão inicial do CAS: um para áudio, um para vídeo e um para captura de *slides* no formato Microsoft PowerPoint®. Estes serão detalhados nas subseções seguintes.

5.2.1.1. SpeedCar Áudio

É um componente C# que utiliza Microsoft DirectSound, uma biblioteca do SDK DirectX[25], para a captura do dispositivo principal de áudio do micro e gravação em formato WAVE com as configurações exibidas na Listagem 5.4.

```
AverageBytesPerSecond = 44100;
BitsPerSample = 16;
BlockAlign = 2;
Channels = 1;
FormatTag = WaveFormatTag.Pcm;
SamplesPerSecond = 22050;
```

Listagem 5.4: Especificações do formato de áudio utilizado.

O uso do formato WAVE para a captura foi apenas uma decisão de projeto para facilitar a construção do componente, não havendo nenhum impedimento na substituição desse componente por outro que já grave em um formato comprimido, como o MP3 (*MPEG-1 Audio Layer 3*) ou o AAC (*Advanced Audio Coding*), também conhecido como MPEG-2 Part 7 ou MPEG-4 Part 3, padrão da TV Digital Brasileira. O serviço de pós-produção converte qualquer formato de áudio suportado pelo Microsoft DirectSound. Caso a mídia já seja o formato comprimido definido ele pula o passo de conversão.

[25] http://msdn.microsoft.com/directX/

5.2.1.2. SpeedCar D-Link Video

Esse componente C# utiliza um objeto COM+ fornecido pelo SDK, da D-Link, para controlar todos os modelos de câmera PTZ (*Pan Tilt Zoom*) disponíveis do fabricante menos o DCS-950G e realizar a captura do vídeo em formato AVI com compressão INDEO5. Apesar de o objeto COM+ usado suportar alguns outros CODECs de vídeo, todos apresentavam qualidade ou fator de compressão piores que o INDEO5. Para não aumentar a complexidade do componente de captura e sobrecarregar em processamento a máquina onde ele for executado, optou-se por postergar a conversão para formatos com maior compressão como o MPEG-2 ou MPEG-4 para a fase de pós-edição.

Como as câmeras da D-Link PTZ são câmeras IP, é necessário informar para o componente COM+ o endereço IP, usuário e senha das mesmas. Como essas informações não estão previstas no contrato de serviço, até porque só seriam úteis para esse componente, elas foram embutidas em um arquivo XML com a extensão *.config* que reside junto do binário conforme ilustrado na Listagem 5.5.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <add key="IP" value="192.168.0.1"/>
  <add key="user" value="demo"/>
 <add key="pass" value="12345"/>
</configuration>
```

Listagem 5.5: Arquivo de configuração do SpeedCar DLink Vídeo.

5.2.1.3. SpeedCar PowerPoint

Componente C# que utiliza o objeto COM+ do Microsoft PowerPoint® para capturar todos os eventos de transição de *slides* em um arquivo XML padrão dos SpeedCars, como ilustrado na Listagem 5.6.

```
<?xml version="1.0"?>
<SpeedCarPPT>
    <PresentationInfo>
        <StartDateTime>20/2/2008 09:14:27</StartDateTime>
        <PPTFileName>Modeling.ppt</PPTFileName>
        <NumberOfSlides>15</NumberOfSlides>
```

```
    </PresentationInfo>
    <SlideShow>
        <SlideChange seq="1">
            <Slide>1</Slide>
            <Start>0</Start>
            <End>158,125</End>
        </SlideChange>
        ...
        <SlideChange seq="17">
            <Slide>15</Slide>
            <Start>4285,15625</Start>
            <End>4372,15625</End>
        </SlideChange>
    </SlideShow>
</SpeedCarPPT>
```

Listagem 5.6: Exemplo de XML gerado na gravação de um PowerPoint®.

O seu funcionamento está baseado no registro de uma função que deve ser chamada a cada evento de troca de *slides* do PowerPoint®.

5.2.2. CASIn

Como já observado, o CASIn (*Capture & Access Service Infrastructure*) é um conjunto de vários componentes, responsável por toda a infra-estrutura do CAS, com atuação em todas as fases de C&A. Na Figura 5.2 observa-se em detalhes a arquitetura do CASIn, seguindo um modelo de n-camadas.

No nível mais baixo encontra-se a camada de banco de dados, responsável pelo armazenamento das meta-informações do evento. Algumas destas são informadas pelo usuário (ex.: assunto, tipo de evento etc.), outras são obtidas automaticamente do CAS (ex.: data, usuário, mídias capturadas etc.) e por fim existem aquelas que são extraídas automaticamente das mídias (ex.: imagem para identificação do evento no *site*, palavras-chave dos títulos dos *slides* etc.). O armazenamento das mídias se dá no sistema de arquivos, e no banco de dados é guardado apenas a referência para a pasta do evento. Esta camada também possui procedimentos armazenados para manipulação destes dados.

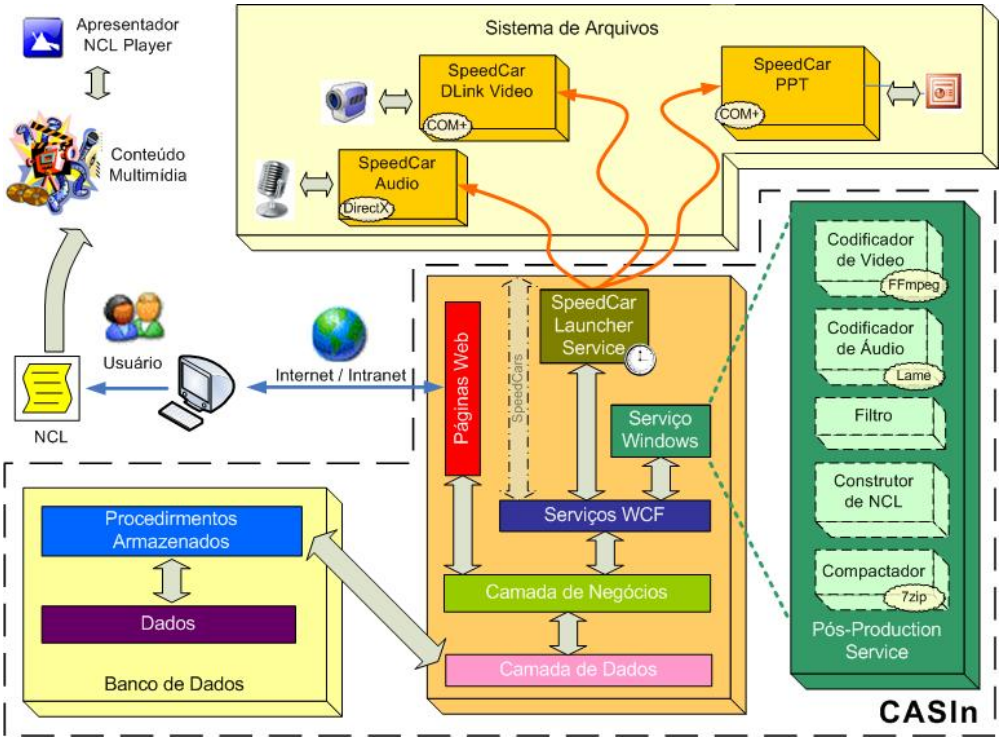

Figura 5.2: Arquitetura detalhada do CASIn.

Uma camada comum de negócio está acima da camada de dados, e é responsável pelo registro dos componentes, processamento de pedidos do usuário no *site* e manutenção do estado dos eventos, dentre outras funcionalidades. Esta camada é a ponte de comunicação entre a camada de acesso a dados e os três módulos principais: a interface *web* de controle e acesso do CAS denominada CASWeb; o serviço Windows de registro e ativação dos SpeedCars, chamado de SpeedCar Launcher Service; e por fim o serviço de pós-produção – Post-Production Service – que é uma composição de artefatos com tarefas específicas para transformar o conteúdo bruto capturado em um documento hipermídia rico e dinâmico. Nas próximas subseções será detalhado cada módulo do CASIn.

5.2.2.1. CASWeb

A camada de apresentação do CASIn é um *web site* desenvolvido em ASP.NET com C#, que centraliza toda a interação do usuário. Pelo *site*, ilustrado na Figura 5.3, os usuários podem acessar os eventos já realizados, por meio de diferentes listagens ou formas de busca. As buscas podem ser realizadas por meio

de informações gerais do evento, como data de realização, título ou nome do apresentador e por meta-informações extraídas automaticamente do evento capturado, como por exemplo, qualquer texto dos *slides* apresentados. Ao selecionar um evento, o usuário é apresentado a uma tela com todos os seus detalhes e pode recuperá-lo de duas formas: por meio de um arquivo compactado com todo o evento gravado para ser visualizado em seu computador. A visualização propriamente dita é feita com a ferramenta Ginga NCL Emulator[26], conforme detalhado na subseção 5.2.3 sobre acesso.

Figura 5.3: Página inicial do CASWeb.

Usuários devidamente autorizados podem agendar gravação de eventos, controlar eventos que estão sendo capturados e até excluir eventos já arquivados pela mesma interface *web*. Na Figura 5.4 é exibida a primeira tela relacionada à

[26] http://www.ncl.org.br/ferramentas/

gravação de um novo evento. Além das informações básicas do evento, os três últimos campos têm funções especiais dentro do CAS. O *template* informa para o sistema qual modelo de apresentação deve ser utilizado na reprodução do evento. Pode-se ter modelos que privilegiem diferentes tipos de mídias, dando mais destaque na exibição. O penúltimo campo – *Context* – define se o usuário irá informar manualmente os contextos, baseado na apresentação enviada; ou se o sistema deverá analisar o arquivo PowerPoint® e identificar automaticamente os contextos, baseado nas técnicas descritas na subseção 5.2.2.4 sobre transformadores. Por fim, o campo *auto record* indica se o CAS deve iniciar automaticamente na data e hora informados a captura, comunicando-se com todos os SpeedCars registrados.

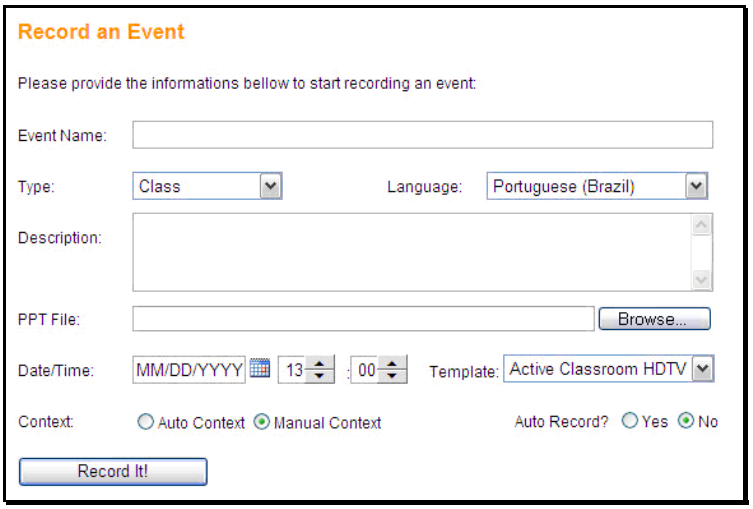

Figura 5.4: Tela de gravação de um evento no CASWeb.

A Figura 5.5 exibe o segundo passo da gravação de um evento que somente é exibido quando o usuário opta pelo contexto manual. O sistema apresenta uma imagem em miniatura de cada *slide* da apresentação enviada, o número do *slide*, e quando possível, o respectivo título. Todas essas informações são apresentadas para facilitar a seleção pelo usuário dos pontos da apresentação onde se inicia um novo contexto. Após marcar todos os *slides* que representam um novo tópico o usuário salva as seleções e é direcionado para a tela inicial.

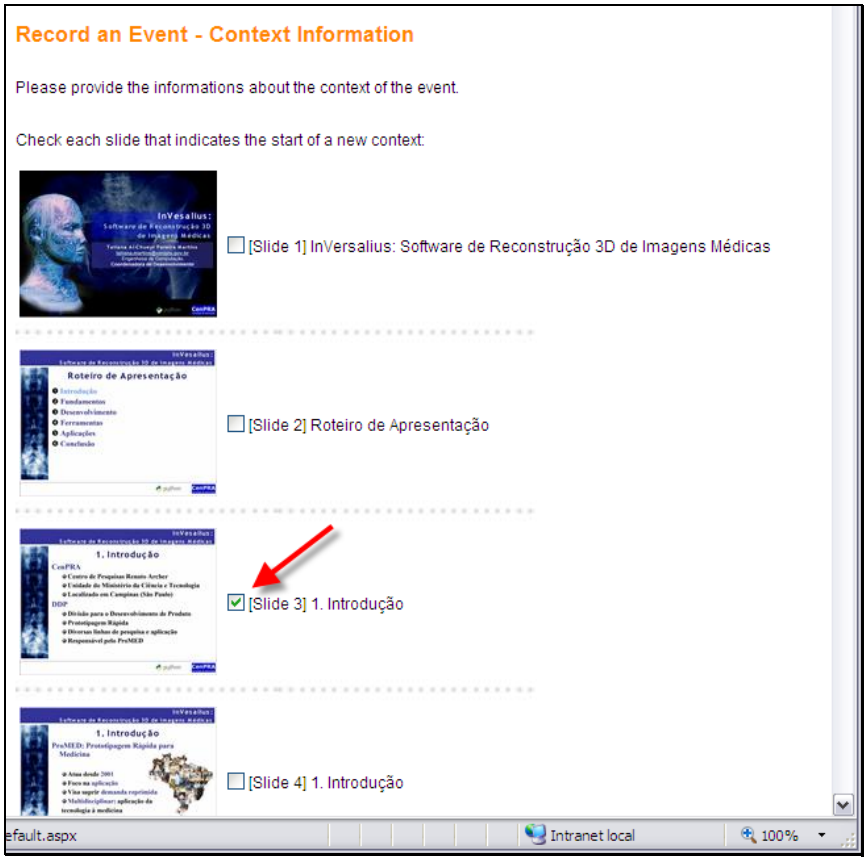

Figura 5.5: Tela de entrada de contexto manual no CASWeb.

No caso da gravação manual o usuário ainda precisará ir à seção de controle das salas inteligentes (*Control an Active Room*) e iniciar manualmente a captura do evento a qualquer momento, desde que outra captura ainda não esteja em andamento. O encerramento da captura pode se dar de duas formas: solicitando o término da gravação do evento manualmente na mesma seção do *site*, ou simplesmente terminando a apresentação. A saída do Microsoft PowerPoint® é interpretada pelo sistema como o término do evento.

5.2.2.2. SpeedCar Launcher Service

Esse componente é implementado como um serviço do Windows (*Windows Service*) e fica em execução constantemente em segundo plano, independentemente da utilização do computador por um usuário. Ele realiza duas tarefas: verifica se há eventos para gravar e então dispara os SpeedCars, e registra os

SpeedCars inicializados no ambiente. Toda a comunicação com a camada de dados e com os componentes de captura é feita por meio do WCF.

Para identificar se há algum evento para capturar ele consome o método GetRecordingEventID(), provido pelo CASWeb em uma função ativada por um *timer*. Ao receber um novo evento ele percorre a sua lista de componentes realizando a chamada ao método StartCaptureNow() definido no contrato de todos os componentes, passando o número do evento e o caminho do compartilhamento no sistema de arquivos onde o componente deve gravar suas saídas.

A lista de componentes é atualizada pelo seu método RegisterComponent(), exposto em seu contrato, que cadastra numa estrutura de lista o componente com o seu WCF *address*, que é um URI (*Uniform Resource Identifier*) como o ilustrado na Listagem 5.7.

```
net.tcp://dell2.ac.inf.puc-rio.br:10001/cas/speedcar
```

Listagem 5.7: Exemplo de endereço de ativação de um SpeedCar.

Vale destacar que o exemplo da Listagem 5.7 só funciona quando o serviço está sendo hospedado pelo Windows Activation Service (WAS) disponível em máquinas com o Windows Vista® ou Windows Server 2008®. Com as demais versões do Microsoft Windows® o protocolo deve ser obrigatoriamente o HTTP/HTTPS, com um *web service* gerenciado pelo IIS.

5.2.2.3. Post-Production Service

O serviço de pós-produção – *Post-Production Service* – é o maior serviço do CASIn e é, na verdade, um fluxo de trabalho (*workflow*) seguido pela ativação de vários componentes em cadeia. Por meio de um *timer* são verificados novos eventos capturados prontos para a fase de pós-produção. Após sua ativação ele busca outros serviços para a realização da seguinte seqüência de macroatividades: recodificação do vídeo e do áudio, ajuste do início das mídias, filtragem do documento e dos eventos capturados, geração da NCL que represente o evento e arquivamento de todo o conteúdo hipermídia, tornando-o disponível no *site* para o

usuário. Da mesma forma que o SpeedCar Launcher Service, a comunicação com a camada de dados é feita por meio de serviços WCF que lhes garantem uma independência de máquina para execução.

Como os SpeedCars implementados, listados na seção 5.2.1, trabalham com formatos de baixa compressão, a primeira tarefa do serviço de pós-produção é recodificar as mídias de áudio e vídeo, utilizando *codecs* compatíveis com os *players* NCL para obter uma melhor taxa de compressão. Para o áudio ele utiliza a biblioteca LAME[27] para gerar um MP3, e para o vídeo ele utiliza o MEncoder[28] para gerar arquivos MPEG4.

Para garantir que na recuperação de uma sessão as mídias armazenadas sejam exibidas de forma síncrona, o segundo passo do serviço de pós-produção do CASIn é sincronizar o início das mídias capturadas. Por diversos motivos, o tempo entre o comando de ativação passado pelo *SpeedCar Launcher Service* e o início efetivo da gravação de uma mídia pode ser diferente para cada componente de captura. Desta forma optou-se por uma abordagem de sincronização em que cada SpeedCar é responsável por gerar um arquivo XML com seus tempos que deve ser disponibilizado junto com a mídia gravada. Neste arquivo é informado o *timestamp* do início da captura utilizado para a sincronização das mídias e os tempos relativos de cada evento registrado. Nesta implementação é preciso garantir a sincronização dos relógios dos diferentes computadores, o que é feito por meio do protocolo NTP. Logo, neste segundo passo são lidos os arquivos XML de todas as mídias e gerado um novo arquivo com todos os tempos relativos entre elas. A mídia que começou a ser gravada mais cedo vira o marco inicial ("tempo zero") e as demais são ajustadas tendo como base essa referência.

Apesar da necessidade de sincronização de relógios, esta abordagem é bem mais simples que as técnicas sofisticadas apresentadas em alguns trabalhos anteriores de C&A. O Cornell Lecture Browser [MS99], por exemplo, gravava o áudio ambiente apenas no canal direito de cada câmera, e no canal esquerdo gravava um sinal garantindo a sincronização dos vídeos. Para sincronizar os *slides*

[27] http://lame.sourceforge.net/

com os vídeos ele realizava um processamento comparando um *stream* de vídeo responsável por filmar a região visual dos *slides* com as imagens dos *slides*. Apesar da complexidade de todo o processo, eles relatam um acerto de aproximadamente 97% da sincronização [MS99]. Com o uso da NCL os SpeedCars são simples, armazenando apenas os *timestamps* e não houve necessidade de se construir um formatador complexo, pois utiliza-se o já disponibilizado para a própria linguagem.

Em seguida são processados todos os transformadores que estiverem ativos. Os transformadores, discutidos em detalhes na subseção 5.2.2.4 a seguir, têm o papel de transformar o documento hipermídia quebrando o paradigma temporal recorrente desse tipo de aplicação e agregar outras formas de visualização do conteúdo.

Após a aplicação dos transformadores, o novo documento XML deve ser transformado num documento hipermídia, que no nosso caso será um documento NCL [ABNT07] – padrão brasileiro de TV Digital. Ao escolher a NCL, e não a criação de um visualizador próprio baseado na própria estrutura em XML como alguns projetos de C&A [AABE98, GRFF01] ou usar outras linguagens hipermídia como em outros projetos [Bult03, CARP03], pode-se com pouco esforço, exibir uma sessão capturada numa TV digital. Esse gerador utiliza informações de tempo dos eventos registradas pelos SpeedCars em conjunto com o *template* escolhido para criar os relacionamentos espaço-temporais entre os objetos de mídia. Além disso, pelos contextos informados pelo usuário ou gerados automaticamente pelos transformadores, esse componente gera os menus de contexto para navegação por tópicos.

A última etapa é a compactação do documento junto com todas as mídias necessárias para o acesso. Esse processo é executado com o auxílio da biblioteca 7-Zip[29]. Ao término da compactação o arquivo é copiado para o servidor *web* e o

[28] http://www.mplayerhq.hu/
[29] http://p7zip.sourceforge.net/

status do evento é modificado no banco de dados para que ele apareça nas buscas dos usuários.

5.2.2.4. CAS Filters

Os transformadores são implementados como serviços WCF e obedecem ao contrato definido na Listagem 5.8. Cada transformador é um método que realiza uma operação com o documento XML de entrada e tem a obrigação de retornar um documento XML válido para o *schema* do CAS detalhado no Apêndice A.

```
[ServiceContract]
interface ICASFilter
{
    [OperationContract]
    XMLDocument Filter(int eventID, XMLDocument originalDoc);
}
```

Listagem 5.8: Contrato de um transformador.

Cada transformador possui seu próprio algoritmo para o processamento do XML. Os três algoritmos que tratam a detecção de contexto alteram o documento XML de entrada, ilustrado pela Listagem 5.9, adicionando atributos de contexto em todos os nós assim identificados como ilustrado pela Listagem 5.10.

```
<?xml version="1.0"?>
<CAS>
  <PresentationInfo>
    ...
    <Slide SlideNumber="3">
      <SlideTitle>1. Introdução</SlideTitle>
    </Slide>
    <Slide SlideNumber="4">
      <SlideTitle>1. Introdução</SlideTitle>
    </Slide>
    ...
    <Slide SlideNumber="13">
      <SlideTitle>Roteiro de Apresentação</SlideTitle>
    </Slide>
    <Slide SlideNumber="14">
      <SlideTitle>2. Fundamentos</SlideTitle>
    </Slide>
    <Slide SlideNumber="15">
      <SlideTitle>2. Fundamentos</SlideTitle>
    </Slide>
    ...
```

Listagem 5.9: Exemplo simplificado de XML do evento de Python (detalhado na subseção 6.1) sem a aplicação de transformadores.

```
<?xml version="1.0"?>
<CAS>
  <PresentationInfo>
    ...
    <Slide SlideNumber="3" BeginOfContext="true">
      <SlideTitle>1. Introdução</SlideTitle>
    </Slide>
    <Slide SlideNumber="4">
      <SlideTitle>1. Introdução</SlideTitle>
    </Slide>
    ...
    <Slide SlideNumber="13" BeginOfContext="true">
      <SlideTitle>Roteiro de Apresentação</SlideTitle>
    </Slide>
    <Slide SlideNumber="14" BeginOfContext="true">
      <SlideTitle>2. Fundamentos</SlideTitle>
    </Slide>
    <Slide SlideNumber="15">
      <SlideTitle>2. Fundamentos</SlideTitle>
    </Slide>
    ...
```

Listagem 5.10: Exemplo simplificado de XML do evento de Python (detalhado na subseção 6.1) com transformador de contexto simples.

Já o transformador de remoção de transições curtas, descrito na subseção 4.2.4.4, simplesmente remove alguns nós “SlideChange” do documento XML, além de alterar o valor “End” do elemento anterior à transição removida. A Listagem 5.11 mostra, de forma simplificada, o XML do evento de Python Científico descrito na subseção 6.1. Do lado esquerdo temos os dados brutos, que representam o evento assim como ocorreu, e no lado direito temos o documento com o transformador de remoção de transições curtas aplicado para tempos menores que um segundo. Podemos observar claramente que a apresentadora no *slide* 7 retornou ao *slide* 5 para alguma explicação adicional e depois retornou à sua apresentação. O transformador removeu corretamente os dois slides que não eram relevantes.

```
<?xml version="1.0"?>
<CAS>
  ...
  <SlideShow>
    ...
    <SlideChange seq="6">
```

```
<?xml version="1.0"?>
<CAS>
  ...
  <SlideShow>
    ...
    <SlideChange seq="6">
```

```
      <Slide>6</Slide>
      <Start>16,98</Start>
      <End>19,85</End>
    </SlideChange>
    <SlideChange seq="7">
      <Slide>7</Slide>
      <Start>19,85</Start>
      <End>24,40</End>
    </SlideChange>
    <SlideChange seq="8">
      <Slide>6</Slide>
      <Start>24,40</Start>
      <End>24,70</End>
    </SlideChange>
    <SlideChange seq="9">
      <Slide>5</Slide>
      <Start>24,70</Start>
      <End>29,81</End>
    </SlideChange>
    <SlideChange seq="10">
      <Slide>6</Slide>
      <Start>29,81</Start>
      <End>30,12</End>
    </SlideChange>
    <SlideChange seq="11">
      <Slide>7</Slide>
      <Start>30,12</Start>
      <End>37,65</End>
    </SlideChange>
    ...
```

```
      <Slide>6</Slide>
      <Start>16,98</Start>
      <End>19,85</End>
    </SlideChange>
    <SlideChange seq="7">
      <Slide>7</Slide>
      <Start>19,85</Start>
      <End>24,70</End>
    </SlideChange>
    <SlideChange seq="9">
      <Slide>5</Slide>
      <Start>24,70</Start>
      <End>30,12</End>
    </SlideChange>
    <SlideChange seq="11">
      <Slide>7</Slide>
      <Start>30,12</Start>
      <End>37,65</End>
    </SlideChange>
    ...
```

Listagem 5.11: Comparativo simplificado entre o XML bruto do evento de Python (detalhado na subseção 6.1) e o XML do mesmo evento após a aplicação do transformador de remoção de transições curtas.

5.2.3. Acesso

Uma das vantagens da utilização da linguagem declarativa NCL é poupar o trabalho de desenvolver uma ferramenta de visualização, usualmente denominada formatador hipermídia, que sincronize as diversas mídias exibidas. A ferramenta Ginga-NCL Emulator[30], desenvolvida pela PUC-Rio, oferece uma infra-estrutura de apresentação para aplicações multimídia/hipermídia sob o paradigma declarativo, escrita em linguagem NCL. Apesar da versão *desktop* não ter o compromisso com a renderização de aspectos avançados da linguagem NCL, como transparências e transições[31], ela oferece todos os demais recursos da NCL

[30] http://www.ncl.org.br
[31] http://www.softwarepublico.gov.br/dotlrn/clubs/ginga/lars-blogger/archive/2007

como sincronização espacial e temporal de objetos de diferentes tipos de mídia, focos de nosso interesse.

O processo de acesso é realizado pela recuperação de um arquivo compactado, do *site,* contendo o arquivo NCL e uma pasta com todas as mídias envolvidas naquele evento. Ao abrir o arquivo NCL do evento com o apresentador de documentos NCL – o Ginga-NCL Emulator – o usuário verá algumas mensagens gerais sobre o sistema e em seguida deverá escolher a forma de navegação pelo evento. Caso o usuário opte por uma navegação contextual ele será apresentado à um índice gerado pelos contextos identificados como ilustrado pela Figura 5.6. Ao selecionar um contexto, ou caso o usuário tenha optado por visualizar todo o evento de forma cronológica no menu anterior, ele visualizará todas as mídias capturadas de forma sincronizada.

Figura 5.6: Menu de seleção para navegação por contexto da terceira palestra do evento de Python (detalhado na subseção 6.1.4) no Ginga NCL Emulator.

Figura 5.7: Reprodução da terceira palestra do evento de Python (detalhado na subseção 6.1.4) no Ginga NCL Emulator com três vídeos em sincronia com os slides.

A apresentação visual do evento pode variar de acordo com o *template* utilizado para a geração da NCL. Os exemplos acima utilizaram o modelo chamado "Active Classroom HDTV" que usa uma imagem de TV de alta definição como fundo para compor as mídias e controles do evento. O *template* também é responsável pela disposição espacial das mídias e controles que permitam mudanças na diagramação pelo usuário durante a visualização. Os controles desse modelo em específico permitem a parada total da reprodução, o retorno ao menu inicial e a alternância entre mídias na área central de destaque.