2
Trabalhos Relacionados

Atualmente, entre os principais middlewares declarativos destacam-se o europeu DVB-HTML (ETSI, 2003), o americano DASE (*DTV Application Software Environment)* declarativo (ATSC, 2003) e o japonês BML (*Broadcast Markup Language*) (ARIB, 2002). As seções a seguir detalham as particularidades de cada um deles e oferece uma análise comparativa.

2.1.
DVB-HTML

O middleware declarativo DVB-HTML faz parte das especificações do padrão MHP (ETSI, 2003), que foi desenvolvido com o objetivo de especificar um middleware, em conformidade aos padrões DVB, para terminais de acesso de TV digital interativa.

2.1.1.
Arquitetura

A arquitetura do middleware MHP, ilustrada na Figura 1 (b), é baseada no uso de uma máquina virtual Java. O padrão define uma API genérica com o objetivo de oferecer acesso aos recursos dessa máquina virtual, bem como aos recursos do terminal de acesso. Uma aplicação Java que faz uso dessa API genérica é chamada de aplicação DVB-J. Como alternativa às aplicações DVB-J, o padrão MHP define (de forma opcional) uma aplicação, comumente denominada *user agent* na terminologia W3C[4], capaz de interpretar documentos declarativos hipermídia que reúnem um conjunto de padrões desenvolvidos para *Web*.

No contexto dos padrões DVB, uma aplicação interpretada pelo *user agent* é chamada de aplicação DVB-HTML. As especificações sobre o *user agent* DVB-

HTML e suas aplicações definem o middleware declarativo europeu DVB-HTML. Por ser opcional, um *user agent* pode ser implementado como um *plug-in* do middleware MHP, como será discutido na Seção 2.1.7. Finalmente, para controlar o ciclo de vida das aplicações DVB-J, bem como das aplicações DVB-HTML, o padrão MHP especifica um gerenciador de aplicações (ETSI, 2003).

A base do *user agent* DVB-HTML é a linguagem XHTML (*eXtensible HyperText Markup Language*) (Pemberton, 2002), versão baseada em XML da linguagem HTML. Por ser baseada em XML, a linguagem XHTML não possui tolerância a documentos HTML mal formados, isso significa que as implementações de um interpretador XHTML podem ser muito mais simples que as implementações de um interpretador HTML.

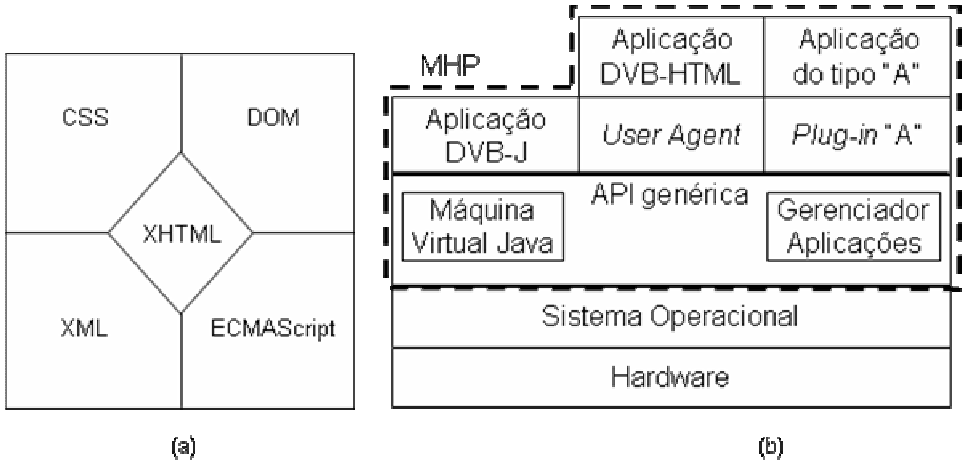

Figura 1: (a) Padrões reconhecidos pelo *User Agent* DVB-HTML.
(b) Arquitetura Middleware MHP (Procedural + Declarativo)

Além de XHTML, um *user agent* DVB-HTML deve dar suporte a outros padrões, conforme ilustra a Figura 1 (a), são eles: CSS (*Cascade Style Sheets*) (W3C, 1998), responsável pela formatação e layout de páginas HTML, ECMAScript (ECMA, 1999), responsável por adicionar recursos de programação (scripts) a documentos XHTML, e DOM (*Document Object Model*) (W3C, 2004), responsável por permitir que código procedural (por exemplo, os scripts) manipule estruturas e conteúdos de documentos XHTML. Esses padrões serão discutidos nas seções a seguir.

[4] http://www.w3c.org

2.1.2. Extensões sobre XHTML

A especificação W3C de XHTML define a linguagem através de módulos, apresentados na Tabela 1. Por ter foco em aplicações para *Web*, alguns desses módulos não possuem utilidade no contexto da TV digital. Assim, as especificações MHP definem para a linguagem DVB-HTML um subconjunto de XHTML, apresentado na Tabela 1, e estendem esse subconjunto através de especificações para um módulo adicional, denominado *DVB intrinsic events*. Os módulos XHTML que pertencem ao conjunto definido pelo padrão DVB são indicados na Tabela 1 através do símbolo "✓", por outro lado os módulos que não pertencem a esse conjunto são indicados com o símbolo "✗". Outras tabelas apresentadas neste capítulo utilizam essa mesma notação.

XHTML	DVB-HTML
Structure	✓
Text	×
Hypertext	✓
List	✓
Applet	×
Presentation	✓
Edit	×
Bidirectional text	✓
Basic forms	×
Forms	✓
Basic tables	×
Tables	✓
Image	✓
Client-side image map	✓
Server-side image map	×
Object	✓
Frames	✓
Target	✓
IFrame	✓
Intrinsic events	×
Meta-information	✓
Scripting	✓
Style sheet	✓
Style attribute	✓
Link	✓
Base	✓
Name identification	×
Legacy	×

Tabela 1: Módulos XHTML utilizados por DVB-HTML.

O módulo *DVB intrinsic events* foi definido para gerar eventos de acordo com o estado da interpretação do documento DVB-HTML. Esse módulo oferece três tipos de eventos para os principais elementos de um documento DVB-HTML (*body* e *frame*): `ondvbdomstable`, `onload` e `onunload`. Esses eventos ocorrem de acordo com as precondições apresentadas na Tabela 2.

Evento	Precondição
dvbdomstable	Quando a estrutura do documento DVB-HTML for completamente recebida e, depois, a interpretação (*parsing*) sobre esse mesmo documento tiver sido realizada (isso significa que a estrutura DOM do documento DVB-HTML está estável). Como conseqüência, uma modificação na estrutura DOM do documento DVB-HTML poderá ser realizada de forma "segura".
Load	Quando todos os recursos (imagens, Xlets, entre outros (ETSI, 2003)) relativos aos elementos do documento DVB-HTML forem recebidos .
unload	Quando o documento DVB-HTML for removido de uma janela ou *frame*.

Tabela 2: Precondições dos Eventos Definidos no Módulo *DVB Intrinsic Events*.

2.1.3. Extensões sobre CSS

Com o objetivo de oferecer acesso à camada gráfica do modelo de apresentação do MHP, DVB-HTML requer conformidade com a recomendação W3C de CSS (W3C, 1998). Assim, o conjunto de regras e propriedades de CSS foi adaptado e estendido para atender às características de um ambiente de TV digital, sempre respeitando a conformidade com a recomendação W3C. As extensões foram especificadas com o objetivo de oferecer:

- **Integração com a camada gráfica**: uma regra, denominada `viewport`, foi criada com propriedades para permitir que uma aplicação DVB-HTML defina sua área de atuação na camada gráfica do modelo de apresentação. Uma das propriedades dessa regra é o `initial block`, que permite definir uma área retangular (com tamanho e posição) interna à `viewport`. Apenas uma regra `viewport` pode ser definida por aplicação. A regra `viewport`, bem como a propriedade bloco inicial são ilustradas na Figura Tabela 32;
- **Gerenciamento da opacidade**: a propriedade `opacity` foi criada para realizar efeitos de translucidez na camada gráfica do modelo de apresentação;
- **Integração com a camada de vídeo**: a propriedade `clip-video` foi criada para permitir que uma área retangular da camada de vídeo seja associada a elementos `object` ou `image` de um documento DVB-HTML. A Figura Tabela 32 apresenta um `clip-video` como uma área retangular equivalente à área do vídeo apresentado;

- **Navegação por controle remoto**: as propriedades `nav-up`, `nav-down`, `nav-left`, `nav-right`, `nav-index` e `nav-first` permitem alternar o foco entre elementos definidos no documento DVB-HTML através de eventos de controle remoto. Por exemplo, se ao especificar um elemento "X" no documento XHTML for atribuída, através de CSS, uma propriedade `nav-left` a "X", quando o evento `nav-left` (seta para esquerda, no padrão de tratamento de eventos de controle remoto (ETSI, 2003)) for gerado pelo controle remoto, o elemento "X" receberá o foco. Analogamente, outros elementos podem receber as propriedades `nav-up`, `nav-down`, e `nav-right`. A propriedade nav-first indica qual elemento deverá receber o foco quando a aplicação for inicialmente exibida e, finalmente, a propriedade `nav-index` atribui um botão específico (exceto aqueles definidos nas propriedades já citadas) do controle remoto ao elemento. Dessa forma, quando o botão atribuído for acionado, o respectivo elemento receberá o foco.

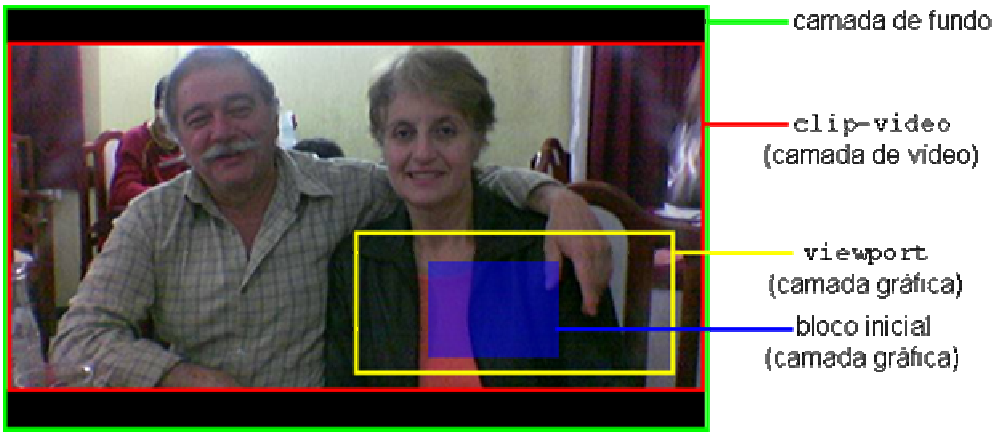

Figura Tabela 32: Propriedades CSS sobre o Modelo de Apresentação MHP

2.1.4. Extensões sobre DOM

Para permitir que o código procedural (na linguagem ECMAScript), presente em um documento DVB-HTML, ou mesmo outras aplicações (inclusive aplicações DVB-J, ou o próprio *user agent*), manipulem de forma dinâmica o

conteúdo de um documento DVB-HTML, o padrão MHP especifica o uso das recomendações W3C do modelo DOM.

Entre os principais objetivos do modelo DOM estão a criação de estruturas lógicas para documentos especificados em linguagens baseadas em XML, bem como a forma com que esses documentos serão acessados e manipulados por códigos procedurais como, por exemplo, Java e ECMAScript (W3C, 2004). Para isso, o modelo DOM especifica uma API (W3C, 2004).

O modelo DOM foi desenvolvido com foco em documentos hipermídia baseados em tecnologias desenvolvidas para Web. Assim, o padrão MHP especifica que o *user agent* DVB-HTML precisa oferecer suporte a apenas um subconjunto do modelo DOM, conforme descrito na Tabela 4.

Módulo DOM		**DVB-HTML**
Pacote	**String**	
Level 2 core	Core	✓
	XML	×
Level 2 HTML	HTML	×
Level 2 views	Views	✓
Level 2 style sheets	StyleSheets	×
Level 2 CSS style sheets	CSS	×
	CSS2	✓
Level 2 events	Events	✓
	UIEvents	✓
	MutationEvents	✓
	HTMLEvents	×
	MouseEvents	×
Level 2 Traversal and Range	Traversal	×
	Range	×

Tabela 4: Módulos DOM utilizados por DVB-HTML.

Além de utilizar os módulos recomendados pelo W3C, apontados pela Tabela 4, cinco novos módulos DOM são especificados no padrão MHP: *DVB-HTML*, *DVB Events*, *DVB Key Events*, *DVB CSS* e *DVB Environment*.

O módulo DOM *DVB-HTML* substitui o módulo DOM *HTML* e foi definido para refletir as modificações nas semânticas dos módulos XHTML tratados pelo *user agent* DVB-HTML, conforme discutido na Seção 2.1.2.

Com o objetivo de permitir que códigos procedurais (scripts no documento DVB-HTML e classes Java, incluindo as classes de aplicações DVB-J) controlem

o ciclo de vida das aplicações DVB-HTML, em conformidade com as especificações MHP, o módulo DOM *DVB Events* especifica a interface denominada *DVBLifecycleEvent*. Essa interface define eventos como, por exemplo, "iniciando aplicação", "pausando aplicação", "retomando aplicação", entre outros (ETSI, 2003). O módulo DOM *DVB Events* possui ainda uma interface, denominada *trigger-event*, que contém atributos para descrever, no modelo DOM, um evento DSM-CC, uma ocorrência no tempo representada por uma estrutura de dados que será discutida na Seção 3.2.2. A interface *trigger-event* permite que o *user agent* notifique a ocorrência de um evento DSM-CC específico às aplicações DVB-HTML interessadas. Esse mecanismo de sincronismo será discutido na Seção 2.1.6.

O módulo DOM *DVB Key Events*, por sua vez, oferece uma interface para eventos de controle remoto, enquanto que o módulo DOM *DVB CSS* tem o objetivo de permitir que códigos procedurais acessem as regras e propriedades CSS estendidas pelo padrão MHP, discutidas na Seção 2.1.3. Finalmente, o módulo DOM *DVB Environment* possui o objetivo de permitir que códigos procedurais acessem variáveis de ambiente definidas em uma aplicação DVB-HTML.

O fato de uma aplicação DVB-J utilizar DOM para modificar o comportamento de uma aplicação DVB-HTML significa a existência de um meio de comunicação do middleware procedural para o middleware declarativo MHP. Esse meio de comunicação é um dos dois tipos definidos pelas especificações MHP. A comunicação no outro sentido, do middleware declarativo para o middleware procedural, será discutida na próxima seção. A utilização de um desses dois meios de comunicação é denominada *bridge*, no contexto MHP.

2.1.5. Código Procedural através de ECMAScript

As especificações MHP determinam que um *user agent* DVB-HTML deve prover suporte a ECMAScript, com o objetivo de incrementar o poder de expressividade das aplicações DVB-HTML. A linguagem ECMAScript teve como origem uma miscelânea de tecnologias, entre as mais conhecidas estão JavaScript (Nestscape) e JScript (Microsoft). Entretanto, ECMAScript possui mais recursos que suas precursoras, sendo uma linguagem de programação

interpretada, que utiliza o paradigma orientado a objetos e pode oferecer suporte a scripts em documentos XML (ECMA, 1999). ECMAScript possui uma propriedade, denominada `Packages`, que a permite acessar objetos Java de um pacote que ela conheça. Por exemplo, para um documento XML que possui ECMAScript acessar uma classe que faz parte do pacote `java.lang`, basta utilizar a seguinte construção ECMAScript: `Packages.java.lang` (Perror, 2001). O padrão MHP especifica que os pacotes e classes do middleware procedural devem ser conhecidos por ECMAScript. Assim, através de ECMAScript, as aplicações DVB-HTML podem utilizar os recursos do middleware procedural MHP, caracterizando um meio de comunicação do middleware declarativo para o middleware procedural. Analogamente, uma aplicação DVB-J pode se tornar "alcançável" ao ECMAScript de uma aplicação DVB-HTML.

2.1.6. Sincronismo

O protocolo DSM-CC, que será discutido na Seção 3.2, é o único mecanismo especificado no padrão MHP para realizar o sincronismo do comportamento das aplicações DVB-HTML com o conteúdo audiovisual transmitido pelo provedor de conteúdo. O mecanismo de sincronismo é utilizado de acordo com as discussões sobre eventos DSM-CC que serão realizadas no próximo capítulo.

Uma das principais funções do *user agent* DVB-HTML ao receber um documento, é realizar a interpretação do mesmo. Durante a interpretação, um modelo DOM desse documento é construído pelo *user agent*. Nesse processo, através de ECMAScript ou mesmo através de uma aplicação DVB-J a qual o documento DVB-HTML faça referência, uma chamada pode estar presente para cadastrar alguma função procedural (associada ao documento DVB-HTML) como *listener* de um determinado tipo de evento, conforme ilustrado pelo código da função setupEventListeners, na Figura 3 (ETSI, 2003).

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//DVB//DTD XHTML DVB-HTML 1.0//EN"
   "http://www.dvb.org/mhp/dtd/dvbhtml-1-0.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
   xmlns:dvbhtml="http://www.dvb.org/mhp">
   <head>
      <script type="text/ecmascript">
         // event listener declaration
         function handleEvent(evt) { /* Handle the event */ }

         // the listener is positioned on the document root node,
         // i.e. the html node
         function setupEventListeners () {
            var htmlNode = document.documentElement;
            htmlNode.addEventListener("myTriggerEvent", handleEvent, true);
         }
      </script>
   </head>
   <body dvbhtml:onload="setupEventListeners()">
   </body>
</html>
```

Figura 3: Cadastro de aplicação DVB-HTML em um evento DOM, que pode ser a tradução de um evento DSM-CC

A partir de uma especificação de eventos, chamada de *fábrica de eventos* (*event factory*), o middleware MHP pode ser capaz de mapear eventos DSM-CC em eventos DOM, e dessa forma ter as aplicações DVB-HTML sincronizadas com os fluxos elementares transmitidos pelas emissoras. No exemplo, um evento DSM-CC poderia ser associado ao evento DOM denominado "*myTriggerEvent*". Além disso, o documento DVB-HTML interpretado pode ter um elemento que possui uma função procedural (através de ECMAScript ou mesmo através de uma aplicação DVB-J a qual o documento DVB-HTML faça referência) que cria um evento da interface DOM *trigger-event*. Essa interface, além de possuir atributos para descrever um evento DSM-CC no modelo DOM, conta ainda com um atributo, denominado *target*, que contém um evento DOM a ser gerado. A estrutura DOM do documento interpretado passa a possuir então um elemento *trigger-event*, com as informações (atributos) necessárias para mapear um evento DSM-CC em um evento DOM. Ou seja, quando um evento DSM-CC ocorrer, o *user agent* notificará as aplicações DVB-HTML interessadas (aquelas registradas, através de código procedura, para o tipo de evento DSM-CC que está ocorrendo). A notificação consiste em acessar o elemento *trigger-event* que corresponde ao evento DSM-CC que está ocorrendo e gerar o evento DOM presente no atributo *target* desse elemento.

2.1.7. Alternativas de Uso

Um *user agent* pode ser implementado como um *plug-in* do middleware procedural, a ser enviado pelo provedor de conteúdo, como um Xlet, quando for necessária a apresentação de uma aplicação DVB-HTML. Um *plug-in* é um conjunto de funcionalidades que podem ser adicionadas a uma plataforma MHP a fim de possibilitar a interpretação de formatos de conteúdo de aplicações ainda não suportadas nessa plataforma. Uma vez que o *plug-in* está em estado operacional, a plataforma deverá comportar-se como se o formato do conteúdo já fosse suportado de modo nativo, sem o uso do *plug-in*.

É importante notar que as especificações MHP não possibilitam a implementação de um ambiente que possua apenas o middleware declarativo DVB-HTML. Ele é considerado, pelo padrão DVB, dependente do middleware procedural MHP. O *user agent* utiliza a API genérica definida pelo middleware procedural, ou seja, um *user agent* tem acesso aos recursos do terminal de acesso, ou mesmo da máquina virtual Java, através da API procedural MHP. Além disso, o gerenciador de aplicações, que também foi definido no middleware procedural, é responsável por controlar o ciclo de vida das aplicações DVB-HTML. Segundo (MHP, 2005), é impossível a implementação de um ambiente puramente declarativo DVB-HTML em conformidade com as especificações MHP. O objetivo é evitar uma fragmentação no mercado, garantindo que o MHP seja sempre oferecido, também nas versões futuras, de forma compatível (MHP, 2005).

2.2. DASE Declarativo

O padrão DASE foi desenvolvido nos Estados Unidos pelo grupo ATSC[5] (*Advanced Television Systems Comittee*), tendo sua primeira versão sido concluída em 2002.

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

2.2.1. Arquitetura

A arquitetura DASE, ilustrada na Figura 4, possui uma camada superior formada por aplicações DASE, que podem ser declarativas ou procedurais, utilizando tecnologias como XHTML, CSS, Java, JMF, entre outras. Tais tecnologias são disponibilizadas pelo middleware DASE, que é dividido em quatro módulos.

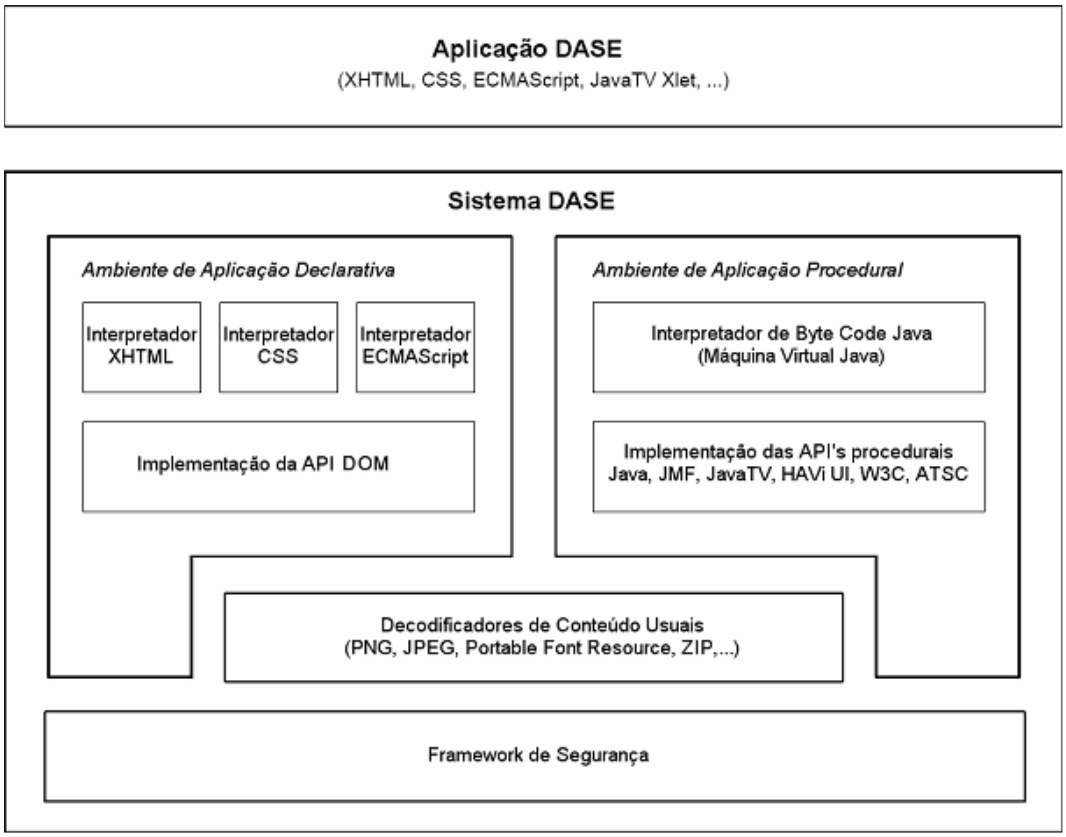

Figura 4: Arquitetura do Middleware DASE (Declarativo + Procedural)

O módulo do Ambiente de Aplicação Declarativa (DAE – *Declarative Application Environment*) consiste na parte declarativa do middleware DASE, sendo um subsistema lógico que, de forma semelhante ao middleware declarativo DVB-HTML, interpreta documentos XHTML com folhas de estilos CSS, scripts ECMAScript e possui suporte a DOM. Assim, uma aplicação que faz uso do DASE declarativo utiliza uma linguagem, oferecida por ele, para ter acesso aos recursos do terminal de acesso. Analogamente, o módulo do Ambiente de Aplicação Procedural (PAE – *Procedural Application Environment*) oferece sua API baseada na linguagem Java às aplicações procedurais. O módulo Decodificadores de Conteúdo atende às necessidades tanto do ambiente de

aplicação procedural quanto do ambiente de aplicação declarativa. Os seguintes decodificadores podem estar presentes nesse módulo: PNG, JPEG, ZIP, entre outros (ATSC, 2003). Finalmente, o módulo Framework de Segurança é responsável por questões de criptografia dos dados transmitidos entre o provedor do conteúdo e o terminal de acesso do usuário (ATSC, 2003).

2.2.2. Extensões sobre XHTML

As especificações DASE definem uma linguagem, denominada XDML (*Extensible DTV Markup Language*), que consiste em um subconjunto dos módulos XHTML recomendados pelo W3C. Os módulos selecionados por DASE são apontados na Tabela 5. Nenhuma extensão XHTML foi especificada pelo padrão DASE.

XHTML	XDML
Structure	✓
Text	✓
Hypertext	✓
List	✓
Applet	×
Presentation	✓
Edit	×
Bidirectional text	✓
Basic forms	×
Forms	✓
Basic tables	×
Tables	✓
Image	×
Client-side image map	✓
Server-side image map	×
Object	✓
Frames	✓
Target	✓
IFrame	×
Intrinsic events	✓
Meta-information	✓
Scripting	✓
Style sheet	✓
Style attribute	✓
Link	×
Base	×
Name identification	✓
Legacy	×

Tabela 5: Módulos XHTML utilizados por XDML.

2.2.3. Extensões sobre CSS

Com o objetivo de oferecer acesso à camada gráfica do modelo de apresentação DASE, suas especificações exigem conformidade com a recomendação W3C de CSS (W3C, 1998). Assim, de forma semelhante ao middleware declarativo europeu, um conjunto de regras e propriedades CSS foi definido para atender às características de um ambiente de TV digital, sempre

respeitando a conformidade com a recomendação W3C. As extensões definidas no padrão DASE, entretanto, são mais restritas que as especificadas pelo MHP. Apenas duas regras e uma propriedade foram definidas como extensão: uma regra para navegação por controle remoto (especificada da mesma forma que em MHP, discutida na Seção 2.1.3); uma propriedade para atualização dinâmica dos elementos de um documento XDML, denominada `atsc-dynamic-refresh` – a propriedade `atsc-dynamic-refresh` permite especificar quais elementos de um documento XDML (i.e. uma imagem, um objeto, ou mesmo todo o documento) devem ser dinamicamente atualizados quando um recurso qualquer da aplicação declarativa (i.e. qualquer elemento da aplicação descrito por XDML, CSS ou ECMAScript) em execução for atualizado; e, finalmente, uma regra para especificação de cores e opacidade de elementos, denominada `atsc-rgba` (vermelho, verde, azul e a porcentagem de opacidade).

2.2.4. Extensões sobre DOM

DOM foi adotado em DASE com os mesmos objetivos do padrão MHP: permitir que scripts (na linguagem ECMAScript), ou código procedural de outras aplicações (aplicações procedurais DASE, bem como ECMAScript de outras aplicações declarativas), manipulem de forma dinâmica o conteúdo de um documento declarativo DASE, caracterizando-se em um meio de comunicação do middleware procedural DASE para o middleware declarativo. Um subconjunto do modelo DOM foi especificado pelo padrão DASE, e modificado com a definição de novas interfaces específicas para TV digital. A Tabela 6 apresenta quais módulos do modelo DOM têm suporte no DASE declarativo.

As principais modificações feitas pelas especificações DASE concentram-se em definir interfaces adicionais em dois módulos DOM: *Core* e *View*. No módulo *Core*, a interface *DOMExceptionExt* acrescenta dois tipos de exceções, VALIDATION_ERR e NO_CLOSE_ALLOWED_ERR. A exceção VALIDATION_ERR ocorre quando uma aplicação tenta modificar um documento de forma a torná-lo inválido. Já a exceção NO_CLOSE_ALLOWED_ERR ocorre quando uma aplicação tenta fechar uma janela que não foi criada por ela. No módulo *View*, a interface *DocumentViewExt* foi criada para definir o espaço de coordenadas para as aplicações DASE

declarativas, ou seja, determinar características (resolução, posição, entre outras (ATSC, 2003)) da camada gráfica do modelo de apresentação DASE.

Módulo DOM		**DASE declarativo**
Pacote	**String**	
Level 2 core	Core	✓
	XML	✓
Level 2 HTML	HTML	✓
Level 2 views	Views	✓
Level 2 style sheets	StyleSheets	✓
Level 2 CSS style sheets	CSS	✓
	CSS2	×
Level 2 events	Events	✓
	UIEvents	✓
	MutationEvents	✓
	HTMLEvents	✓
	MouseEvents	✓
Level 2 Traversal and Range	Traversal	×
	Range	×

Tabela 6: Módulos DOM utilizados por DASE.

2.2.5. Código Procedural Através de ECMAScript

As especificações DASE definem ECMAScript com o mesmo objetivo do middleware declarativo DVB-HTML: incrementar a expressividade na construção das aplicações declarativas, permitindo que essas aplicações utilizem recursos de uma linguagem procedural, bem como do middleware procedural DASE, caracterizando um meio de comunicação do middleware declarativo para o middleware procedural. Da mesma forma que em DVB-HTML, uma aplicação DASE procedural pode se tornar "alcançável" ao ECMAScript de uma aplicação DASE declarativa, caracterizando a existência de aplicações híbridas.

2.2.6. Sincronismo

O mecanismo para realizar o sincronismo do comportamento das aplicações declarativas DASE com o conteúdo audiovisual transmitido pelo provedor de

conteúdo é baseado nos eventos DSM-CC. Entretanto, algumas modificações foram realizadas através das especificações A/93 (ATSC, 2002).

O padrão A/93 define duas entidades: *triggers* e *targets*. Os *triggers* são estruturas de dados, baseadas no descritor de eventos DSM-CC, utilizadas para especificar o momento de disparo de um alvo (uma aplicação DASE, por exemplo). Esse alvo, denominado *target*, é identificado dentro do *trigger* por meio de uma referência. Os *triggers* e *targets* podem ser enviados pelo provedor de conteúdo através de um carrossel de objetos (esse mecanismo será discutido na Seção 3.2.1).

Um *trigger* é formado basicamente por quatro campos: um identificador do *trigger*; uma referência temporal para o instante de "disparo" do *trigger* (que pode ser instantânea, como nos eventos "*do it now*", que serão discutidos na Seção 3.2.2); uma referência para o *target*; e dados específicos para o *target*. Assim, no mecanismo de sincronismo DASE, não é necessário que uma aplicação se registre como *listener* a um determinado tipo de evento (como em DVB-HTML).

Além de aplicações DASE, o campo *target* pode referenciar um evento DOM definido na estrutura lógica do documento DASE. De forma semelhante a DVB-HTML, ao interpretar um documento declarativo DASE, o DAE gera uma estrutura DOM para o mesmo. Note que o mecanismo definido nas especificações A/93 (ATSC, 2002) simplifica a forma de mapear um evento DSM-CC em um evento DOM, comparando com o mecanismo definido em DVB-HTML. Isso porque, ao contrário das aplicações DVB-HTML, as aplicações declarativas DASE não se preocupam com os tipos de eventos DSM-CC que estão sendo gerados, uma vez que elas são alvos (*targets*) desses eventos.

2.2.7. Alternativas de Uso

Diferentemente do middleware declarativo europeu, o DASE declarativo pode ser utilizado em plataformas que não possuem o DASE procedural instalado (CENELEC, 2003; Whitaker & Benson, 2003). As especificações DASE não definem o conceito de *plug-ins*. Entretanto, baseado na definição de uma classe Xlet, deve ser possível enviar uma implementação Xlet do DASE declarativo, com o objetivo de interpretar aplicações DASE declarativas em terminais que, originalmente, não possuem o ambiente declarativo (DAE) implantado.

Finalmente, o middleware declarativo DASE pode ser utilizado também de forma conjunta com o middleware procedural.

2.3. BML

O padrão japonês de TV digital, ISDB (*Integrated Services Digital Broadcasting*), foi desenvolvido em 1999 pela ARIB (*Association of Radio Industries and Broadcast*), um grupo de empresas, fabricantes e operadoras de televisão e de telecomunicações, fomentado pelo governo japonês. Seu desenvolvimento deu-se como segundo movimento de um trabalho iniciado em 1995 com o objetivo de digitalizar todos os sistemas de transmissão de televisão. O middleware japonês, também chamado de ARIB, foi definido através das especificações desse padrão. Diferentemente dos middlewares MHP e DASE, o middleware japonês foi inicialmente concebido contendo apenas um ambiente declarativo. Posteriormente, foi definido no padrão ARIB um middleware procedural (ARIB, 2002).

2.3.1. Arquitetura

De forma semelhante à arquitetura do middleware DASE, a arquitetura do middleware japonês, ilustrada na Figura 5, é composta por um ambiente procedural e um ambiente declarativo, independentes. O middleware procedural é baseado no uso de uma máquina virtual Java; o padrão que especifica o middleware procedural (ARIB, 2004) define uma biblioteca genérica com o objetivo de oferecer acesso aos recursos dos terminais. De forma semelhante ao middleware procedural europeu, *plug-ins* podem ser enviados pelo provedor de conteúdo a fim de possibilitar a interpretação de formatos de conteúdo de aplicações ainda não suportadas pela plataforma. O padrão do middleware declarativo (ARIB, 2002) define uma linguagem de marcação, denominada BML (*Broadcast Markup Language*), que, assim como as linguagens DVB-HTML e XDML, é baseada em XHTML, com suporte a CSS2, ECMAScript e DOM. Algumas extensões, bem como algumas restrições, foram especificadas pelo padrão. Essas especificações serão discutidas na próxima seção.

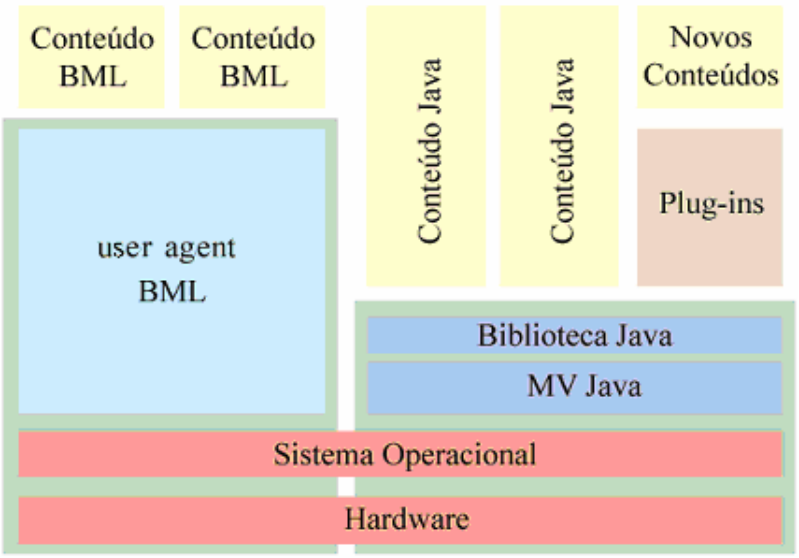

Figura 5: Arquitetura do Middleware ARIB (Declarativo + Procedural)

2.3.2. Extensões sobre XHTML

A linguagem BML consiste em um subconjunto dos módulos XHTML recomendados pelo W3C. Os módulos selecionados no padrão japonês são destacados na Tabela 7. Um único módulo de extensão, denominado *módulo BML*, foi especificado pelo padrão. Os principais elementos definidos nesse módulo são discutidos a seguir.

XHTML	XDML
Structure	✓
Text	✓
Hypertext	✓
List	✓
Applet	×
Presentation	✓
Edit	✓
Bidirectional text	✓
Basic forms	✓
Forms	✓
Basic tables	✓
Tables	✓
Image	✓
Client-side image map	✓
Server-side image map	✓
Object	✓
Frames	✓
Target	✓
IFrame	✓
Intrinsic events	✓
Meta-information	✓
Scripting	✓
Style sheet	✓
Style attribute	✓
Link	✓
Base	✓
Name identification	×
Legacy	×

Tabela 7: Módulos XHTML utilizados por BML.

Para permitir que o sincronismo entre o comportamento de uma aplicação BML e programas transmitidos pelo provedor de conteúdo sejam especificados através de código declarativo, BML define dois elementos para o controle de eventos: `bevent` e `beitem`. O elemento `bevent` possui todos os elementos `beitem` que serão processados no documento. Um elemento `beitem`, por sua vez, possui a capacidade de associar, através de seus atributos, código procedural à ocorrência de um evento específico. Cada elemento `beitem` é identificado através de um atributo chamado de *id*. Além disso, um elemento `beitem` possui um atributo *type*

que especifica o tipo de evento ao qual o elemento se refere. Os principais tipos de eventos, e suas respectivas semânticas, são apresentados na Tabela 8. Um terceiro atributo do elemento `beitem`, denominado *onoccur*, possui a identificação da chamada ao código procedural que será executado quando o evento específico ocorrer.

Tipo de Evento	Semântica
EventMessageFired	Notifica a ocorrência de um determinado evento DSM-CC transmitido pelo provedor de conteúdo.
EventFinished	Notifica a ocorrência do fim de um programa de TV.
EventEndNotice	Pré-notifica a ocorrência do fim de um programa de TV.
Abort	Notifica que a apresentação de um conteúdo foi abortada.
ModuleUpdated	Notifica que foi detectada uma versão atualizada de um módulo específico de um determinado carrossel.
CCStatusChanged	Notifica que a linguagem da legenda sendo exibida foi alterada.
MainAudioStreamChanged	Notifica que um novo fluxo de áudio principal foi selecionado.
NPTReferred	Notifica que existe a possibilidade de adquirir a referência temporal corrente.
MediaStopped	Notifica que a decodificação de uma mídia específica (vídeo, áudio, caracteres, imagem estática, entre outras (ARIB, 2002)) foi completamente realizada.
MediaStarted	Notifica que a decodificação de uma mídia específica foi iniciada.
MediaRepeated	Notifica que a decodificação de uma mídia específica foi reiniciada.
DataButtonPressed	Notifica que um botão específico foi pressionado.

Tabela 8: Tipos de eventos `beitem` e suas respectivas semânticas.

Além dos elementos para controle de eventos, o padrão ARIB define atributos para o elemento `object`, específicos para o controle da exibição dos fluxos transmitidos:

- ***streamstatus***: determina o estado de um fluxo. Os possíveis valores para esse atributo são "*stop*", "*play*" e "*pause*". O valor "*stop*" para um fluxo de áudio, por exemplo, significa na verdade um estado "*mute*";
- ***streamposition***: indica a posição temporal durante a exibição de um fluxo. Quando o fluxo está no estado "*pause*", o valor desse atributo é constante. Quando o fluxo está no estado "*stop*", é atribuído o valor zero para esse atributo. Finalmente, quando o fluxo está no

estado "*play*", o valor representa a unidade de tempo corrente do conteúdo de mídia em exibição (ARIB, 2002);

- ***streamlooping***: determina o número de vezes que um fluxo será exibido novamente;
- ***streamspeednumerator*** e ***streamspeeddenominator***: definem um fator, através de numerador e denominador, respectivamente, para a velocidade de exibição de um fluxo;
- ***streamlevel***: determina o volume de um fluxo. Note que esse atributo é principalmente útil para fluxos de áudio;
- ***streamstartposition*** e ***streamendposition***: determinam o início e o fim, respectivamente, de um fluxo que está armazenado no terminal de acesso.

A Figura 6 apresenta um exemplo que exibe um arquivo de áudio com o nome "sample.aiff", que deverá ser exibido dez vezes, a uma velocidade de exibição com fator "1/2", no volume máximo suportado pelo terminal de acesso. Note que o estado inicial do fluxo é "*stop*", isso porque um fluxo de áudio deve ser iniciado através de scripts.

```
<body>
  <object id="a" data = "sample.aiff"
   streamstatus = "stop"
   streamposition="0"
   streamspeednumerator = "1"
   streamspeeddenominator = "2"
   streamlooping = "10"
   streamlevel="100" />
</body>
```

Figura 6: Exemplo do Uso de Atributos para Controle de Exibição de Fluxo

2.3.3. Extensões sobre CSS

Com o objetivo de oferecer acesso ao modelo de apresentação ARIB, suas especificações exigem conformidade com a recomendação W3C de CSS (W3C, 1998), de forma semelhante aos middlewares declarativos MHP e DASE.

Além de propriedades para definir resolução e cores de elementos, algumas propriedades para controlar eventos de controle remoto foram definidas. Entre elas estão propriedades para seleção de foco entre elementos (nav-index, nav-up,

nav-down, nav-left e nav-right) especificadas da mesma forma que em MHP e DASE. Entretanto, uma propriedade interessante é definida no padrão ARIB, denominada used-key-list, para determinar os tipos de teclas que serão capturadas pelo *user agent* BML. Por exemplo, quando essa propriedade possui o valor "*basic*", os eventos de teclas numéricas do controle remoto são interpretados como seleção de um canal e não são considerados pelo *user agent* BML. Por outro lado, quando essa propriedade possui o valor "*numeric-tuning*", e um elemento que espera entrada de eventos do controle remoto possui o foco, os eventos de teclas numéricas do controle remoto são interpretados como uma entrada numérica para esse elemento e não como uma seleção de canais.

2.3.4. Extensões sobre DOM

DOM foi especificado em ARIB com objetivos semelhantes aos dos padrões MHP e DASE: permitir que o código procedural manipule, de forma dinâmica, o conteúdo de um documento BML. Um subconjunto do modelo DOM foi especificado pelo padrão ARIB, e modificado com a definição de novas interfaces. A Tabela 9 apresenta quais módulos do modelo DOM devem ter suporte, dado pelo *user agent* BML. As novas interfaces foram definidas apenas para refletir no modelo DOM as extensões sobre XHTML e CSS especificadas. Por exemplo, uma interface, denominada *BMLEvent*, possui atributos (ARIB, 2002) correspondentes à definição de um elemento `beitem`, discutido na Seção 2.3.2.

Módulo DOM		BML
Pacote	**String**	
Level 2 core	Core	✓
	XML	×
Level 2 HTML	HTML	✓
Level 2 views	Views	×
Level 2 style sheets	StyleSheets	×
Level 2 CSS style sheets	CSS	×
	CSS2	✓
Level 2 events	Events	✓
	UIEvents	✓
	MutationEvents	✓
	HTMLEvents	×
	MouseEvents	×
Level 2 Traversal and Range	Traversal	×
	Range	×

Tabela 9: Módulos DOM utilizados por BML.

2.3.5. Código Procedural Através de ECMAScript

As especificações ARIB definem ECMAScript com o objetivo de incrementar o poder de expressividade na autoria das aplicações BML. No contexto ARIB, ECMAScript é utilizado apenas para alterar dinamicamente o conteúdo de um documento BML, através da API DOM (Hori & Dewa, 2006). Ou seja, diferentemente de DVB-HTML e DASE declarativo, BML não define o uso de ECMAScript para acessar aplicações procedurais ou mesmo classes do middleware procedural japonês.

2.3.6. Sincronismo

Para realizar o sincronismo do comportamento das aplicações BML com os fluxos dos programas transmitidos pelo provedor de conteúdo, o padrão ARIB oferece ao autor de documentos uma abstração através de declarações BML. Essas declarações são formadas pelas extensões sobre a linguagem XHTML discutidas na Seção 2.3.2.

Ao interpretar um documento BML, o *user agent* gera uma estrutura DOM para o mesmo. Quando um elemento `beitem` é encontrado no documento, a estrutura DOM do documento interpretado passa a possuir um elemento *BMLEvent*, com as informações (atributos) necessárias para mapear o evento ocorrido a uma ação específica (atributo *onoccur*, discutido na Seção 2.3.2).

O *user agent* BML é responsável por monitorar a ocorrência dos tipos de eventos definidos na Tabela 8 (Seção 2.3.2), bem como notificar a ocorrência desses eventos às aplicações BML que possuam ações determinadas para esses eventos. Note que o mecanismo definido pelas especificações ARIB simplifica a forma de realizar o sincronismo do comportamento das aplicações com o conteúdo audiovisual transmitido, comparando com os mecanismos definidos em DASE declarativo e DVB-HTML, por exemplo, no caso da Figura 3, a função *setupEventListeners* seria substituída pelo código declarativo.

2.3.7. Alternativas de Uso

O middleware declarativo japonês é completamente independente de um middleware procedural, mas pode trabalhar em conjunto com o mesmo. Curiosamente, apesar de utilizar o mesmo conceito de *plug-in* do padrão europeu, o padrão ARIB não apresenta nenhuma especificação ou restrição para que seu middleware declarativo seja enviado como um *plug-in* do middleware procedural. Vale notar que o padrão japonês não define formas de comunicação entre o middleware declarativo e o procedural.

2.4. Análise Comparativa

Os principais middlewares declarativos existentes utilizam tecnologias desenvolvidas para *Web*, mais especificamente XHTML com suporte a CSS, DOM e ECMAScript. É um conjunto de tecnologias interessantes, mas ao mesmo tempo, segundo as discussões que serão realizadas a seguir, denotam os principais pontos fracos desses middlewares.

A linguagem XHTML favorece, geralmente, a autoria em função da simplicidade e da disseminação do HTML como linguagem para especificação de documentos na *Web*. Entretanto, as funcionalidades de XHTML são restritas

quando existe a necessidade de especificar o sincronismo e a interatividade, uma vez que XHTML restringe-se à formatação para apresentação e a definição de relações de referência entre documentos. Para superar essa limitação, as linguagens DVB-HTML e XDML adotam XHTML em conjunto com a linguagem ECMAScript. A linguagem BML, por sua vez, especifica extensões sobre XHTML, definindo marcações para descrever algum tipo de sincronismo. Entretanto, a ação dessas marcações geralmente são especificadas através de ECMAScript, exigindo que o autor programe, em detalhes, os passos que a aplicação deve executar para atingir seu objetivo. Além disso, outros relacionamentos de sincronização espaço-temporal, que não se encaixam nas extensões definidas por BML, também devem ser definidos através do uso de ECMAScript, por meio de uma especificação embutida, direta ou indiretamente, em um objeto XHTML. Como toda linguagem procedural, scripts apresentam um propósito mais geral, oferecendo uma maior flexibilidade para construção dos programas. No entanto, o nível de abstração oferecido para os autores é mais baixo, obrigando que os criadores de documentos hipermídia, como os programas de TV interativa, trabalhem, na realidade, como programadores de software.

Dada sua natureza de propósito geral, scripts não apresentam mecanismos diretos para a especificação de sincronismo, exigindo mecanismos de mais baixo nível para a sincronização entre objetos. Isso torna a programação em scripts mais flexível, porém suscetível a falhas quando há mudanças nas estruturas de dados em diferentes partes do escopo das funções. É também difícil identificar pelo código as várias estruturas de sincronismos utilizadas pelo autor. Tudo isso torna extremamente difícil modularizar e manter uma apresentação usando esse paradigma. Quando se deseja reusar parte de uma apresentação, não é claro onde uma cópia deve começar ou terminar. Especificação por script é geralmente útil para apresentações pequenas, mas a manipulação e edição de grandes documentos torna-se difícil sem a definição de uma estruturação. Além disso, a especificação detalhada de atividades paralelas apresenta os mesmos problemas da maioria das linguagens de programação.

Nas linguagens DVB-HTML, XDML e BML, as características de apresentação espacial dos objetos de mídia podem ser especificadas de forma independente através do uso de CSS. Entretanto, baseado em experiências com os navegadores existentes, é complexo conseguir o mesmo resultado na apresentação

de um mesmo documento com CSS em implementações diferentes (Morris & Chaigneau, 2005). Por exemplo, um mesmo documento com CSS interpretado pelos navegadores Mozilla e Internet Explorer, possivelmente, apresentará resultados distintos (apresentação de seus elementos dispostos de maneira diferente). Além disso, segundo (Morris & Chaigneau, 2005), aplicações que utilizam CSS demandam consumo de recursos elevado, levando a apresentações lentas em plataformas onde esses recursos são escassos.

No contexto geral, as linguagens DVB-HTML, XDML e BML utilizam principalmente o protocolo DSM-CC para sincronizar o comportamento de suas aplicações com os fluxos de programas transmitidos pelas emissoras. A sincronização é realizada de acordo com uma semântica embutida em eventos DSM-CC e um mapeamento desses eventos para eventos das linguagens declarativas. Esse é um mecanismo relativamente complexo, quando comparado a linguagens declarativas que vislumbram o sincronismo de mídias.

O fato das limitações dos principais middleware declarativos serem atribuídas às linguagens definidas por eles demonstra a importância de uma escolha criteriosa da linguagem declarativa. A linguagem NCL (Muchaluat-Saade, 2003) é uma alternativa interessante, por ser uma linguagem que vislumbra o sincronismo de mídias, baseada em um modelo conceitual que soluciona algumas limitações do modelo conceitual das linguagens baseadas em HTML (Soares, 2004). Através de um paradigma de causalidade/restrição (Soares et al, 2003), NCL possibilita a especificação das relações de sincronização temporal e espacial entre os objetos dispensando o uso de scripts e ao mesmo tempo oferecendo uma rica expressividade.