2
Linguagens para Descrição de Documentos Hipermídia

Linguagens de programação podem ser classificadas de modos variados. Uma classificação possível distingue as linguagens entre procedurais (imperativas) e declarativas. Linguagens procedurais exigem que o programador especifique, em detalhes, os passos que o programa deve executar para realizar a tarefa projetada (são escritas muitas linhas de código para indicar os passos exatos a fim de se obter um resultado). Em linguagens declarativas, por outro lado, o programador provê uma definição da tarefa a ser realizada, não estando preocupado com os detalhes de como o computador usará essa definição (é oferecida uma descrição exata do objetivo). Em outras palavras, a diferença entre linguagens procedurais e declarativas é que, na primeira, se especifica como obter a resposta, enquanto, na segunda, se especificam as condições que devem ser satisfeitas para se obter a resposta. Programação declarativa envolve *o que* deve ser computado, mas não necessariamente *como* será computado.

Embora linguagens procedurais possam ser utilizadas na autoria de documentos hipermídia (FLEXTV, 2004a), linguagens declarativas oferecem um nível mais elevado de abstração para a programação, por evitar que o programador (ou autor, no contexto de autoria de documentos) escreva muitos dos detalhes de execução que podem ser inferidos pelo interpretador. Esse interpretador (ou executor) é o elemento responsável por aplicar um algoritmo pré-definido às especificações feitas em uma linguagem declarativa para produzir o resultado almejado (como foi visto, esse elemento, em sistemas hipermídia, é denominado formatador). Assim, um programador declarativo experiente é qualificado para descrever cuidadosamente os resultados esperados, entretanto, mantém-se ignorante de como o código interno é executado para se obter o resultado. Obviamente, a linguagem declarativa deve prover os recursos necessários para que o programador/autor seja capaz de especificar um programa/documento de acordo com sua concepção[8].

[8] Uma comparação entre autoria declarativa e procedural, apontando suas vantagens e desvantagens, pode ser consultada em (FLEXTV, 2004a).

Por apresentarem um nível de abstração mais elevado para a programação, esta dissertação irá tratar apenas de linguagens declarativas para autoria de documentos. Em especial, serão abordadas linguagens de marcação declarativas definidas utilizando o padrão XML.

Este capítulo está estruturado da seguinte maneira. Inicialmente discorre-se sobre a meta-linguagem SGML, padrão ISO (*International Organization for Standardization*[9]) que estabeleceu o conceito de linguagens de marcação para autoria de documentos eletrônicos. Em seguida, é apresentada a meta-linguagem XML, um perfil SGML[10] a partir do qual várias linguagens de marcação foram criadas e vêm sendo utilizadas nas mais diversas áreas. Finalmente, são descritas linguagens declarativas baseadas em XML voltadas para o domínio de autoria de documentos hipermídia, notadamente: XHTML, SMIL e NCL.

2.1. SGML e XML

O padrão *Standard Generalized Markup Language* - SGML (ISO, 1986) permite a definição de documentos em função do domínio de aplicação a que se destinam. Marcações SGML permitem especificar a estruturação desses documentos, independente de como os mesmos devam ser apresentados. A formatação (ou seja, características de apresentação) de um documento SGML deve ser definida externamente ao documento, utilizando, por exemplo, tecnologias de folhas de estilos - como CSS (W3C, 1998a).

Para a definição e interpretação de um documento SGML, é preciso especificar uma gramática formal. Uma gramática SGML define como as marcações devem ser interpretadas, quais as regras que restringem o uso de cada marcação nos diferentes contextos do documento e, quando for relevante, a ordem dessas marcações no documento.

A Figura 2:1 ilustra uma aplicação SGML, composta por *declarações SGML*, uma DTD (*Document Type Definition*) e um *interpretador*. De maneira simplificada, declarações SGML definem um conjunto de regras de sintaxe para uma linguagem, enquanto a DTD especifica sua semântica. Dessa forma, uma

[9] http://www.iso.org/

[10] Como será apresentado, o conceito de perfil SMGL é distinto do conceito de perfil de linguagem apresentado anteriormente.

aplicação SGML[11] tem definida a gramática a ser seguida por documentos de uma linguagem, sendo possível aos interpretadores dessa aplicação, além de verificar a sintaxe desses documentos, também interpretar sua semântica, controlando, inclusive, suas exibições.

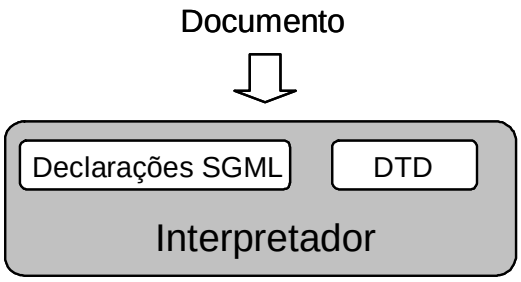

Figura 2:1 - Aplicação SGML.

SGML é suficientemente formal para possibilitar a validação de documentos; tem estrutura suficiente para permitir a especificação e o manuseamento de documentos complexos; e possui mecanismos de extensão capazes de suportar a gestão de grandes repositórios de informação. Entretanto, dada a sua generalidade, a utilização direta de SGML mostrou-se demasiadamente complexa. Visando solucionar parte desse problema, foi definido o padrão XML: um perfil SGML.

Extensible Markup Language - XML (W3C, 2004) restringe alguns dos pontos de flexibilização de SGML, facilitando o desenvolvimento de novas aplicações (linguagens de marcação), assim como o processamento e o intercâmbio de documentos declarados nessas linguagens. XML vem se mostrando um padrão *de fato* para criação de linguagens de marcação, tanto para domínios hipermídia quanto para outros domínios.

De maneira simplificada, XML é menos genérico que o padrão SGML, pois predefine as *declarações SGML* desse padrão (ver Figura 2:1). Assim, é necessária apenas a especificação de DTDs na criação de novas linguagens (aplicações XML). Devido a suas características, XML foi amplamente utilizado, levando à identificação não somente de suas diversas vantagens, mas, também, de algumas de suas limitações. A dificuldade no reuso das definições presentes nas DTDs, por exemplo, foi um dos fatores que impulsionou o desenvolvimento de

[11] Uma aplicação SGML é constituída por um perfil SGML e por uma DTD específica. Ou seja, um interpretador que possui somente *declarações SGML* constitui um perfil (*profile*) SGML.

novas tecnologias relacionadas a XML (W3C, 2001d; W3C, 1999b). Essas novas tecnologias oferecem uma funcionalidade de extrema importância quando se abordam linguagens de marcação baseadas em XML: a criação de linguagens seguindo uma abordagem modular. Módulos agrupam, de forma coerente, elementos e atributos XML[12] que possuam alguma relação semântica entre si.

Diversas são as vantagens da definição de uma linguagem de forma modular. Um primeiro benefício é a possibilidade de criação de perfis de linguagem. Perfis reúnem um subconjunto dos módulos oferecidos pela linguagem, definindo, assim, um subconjunto de funcionalidades apropriadas para a construção de uma determinada classe de documentos. Outra vantagem da estruturação da linguagem em módulos é a facilidade de reutilização. Linguagens podem ser construídas definindo novos módulos e reusando módulos oferecidos por outras linguagens. Mais ainda, é possível a criação de novas linguagens XML através da simples combinação de diversos módulos já existentes em outras linguagens, sem a necessidade de definição de novos módulos. Essas várias vantagens possibilitam que cada linguagem, ou perfil da linguagem, atenda aos requisitos de uma determinada aplicação.

Outra vantagem da abordagem modular é a estruturação da linguagem, que pode ser extremamente útil em linguagens complexas (e com um grande número de elementos e atributos). Assim, módulos podem agrupar os elementos e atributos que possuam semântica relacionada, facilitando, entre outras coisas, a manutenção da linguagem. Como o número de módulos pode ser grande, algumas linguagens definem um segundo nível de estruturação, agrupando seus módulos em *áreas funcionais*.

2.2. HTML e XHTML

HyperText Markup Language - HTML - é a principal linguagem utilizada para autoria de documentos na WWW. Tendo sua primeira versão publicada em 1992 (como uma aplicação SGML), HTML (W3C, 1999a) é, atualmente, uma linguagem declarativa de autoria padronizada pelo W3C.

[12] Elementos e atributos XML serão exemplificados nas próximas subseções.

O modelo hipermídia que fundamenta a linguagem HTML é bastante simples, possibilitando a criação de uma linguagem de fácil autoria (apesar de limitar tanto seu poder de expressão quanto seus recursos para reuso). Páginas HTML representam os *nós* do modelo, enquanto *elos* definem os relacionamentos entre essas páginas. Elos HTML são sempre de referência (do tipo *go-to*) e possuem duas características principais: especificam um relacionamento entre uma única origem e um único destino; e são sempre definidos como parte do conteúdo dos nós (mais especificamente, como parte do conteúdo dos nós de origem do elo).

Essas características impedem a identificação dos elos que fazem referência a uma determinada página; não permitem a separação entre os dados sendo referenciados e as referências propriamente ditas (dificultando a manutenção dos dados e dos elos); não possibilitam a reutilização de nós sem a herança obrigatória dos relacionamentos (por exemplo, não é possível reusar uma página HTML sem herdar todos os seus elos); e impedem a definição de elos em páginas nas quais o autor não possua direitos de escrita.

Apesar das limitações de HTML, sua simplicidade proporcionou uma ampla utilização dessa linguagem. Assim, com a maturidade e simplicidade de HTML e o surgimento do padrão XML, observou-se a possibilidade de definir uma linguagem baseada em XML oferecendo todas as funcionalidades de HTML: a linguagem XHTML (W3C, 2000c). Dessa forma, XHTML possui todos os benefícios de XML (facilidade de intercâmbio de documentos entre aplicações; facilidade na implementação de processadores de documentos nessa linguagem; variedade de bibliotecas de software que oferecem suporte a XML; variedade de tecnologias aplicáveis a XML etc.) agregados à maturidade de HTML.

2.3. SMIL

Synchronized Multimedia Integration Language - SMIL (W3C, 2001b) é uma linguagem de marcação declarativa modular (estruturada em 10 áreas funcionais), derivada de XML, para autoria de documentos hipermídia. Tendo como foco relações de sincronismo temporal, SMIL é um padrão W3C que visa incorporar esse requisito à WWW (ver Capítulo 1), de forma a superar as

limitações de HTML. Diferente de HTML, cujos elos somente possibilitam especificação de relacionamentos de referência entre páginas, SMIL também permite a definição de relacionamentos de sincronismo temporal. Esses relacionamentos são expressos por meio de eventos ou composições.

Composições SMIL contêm um conjunto de nós (objetos de mídia ou outros nós de composição) e possuem semântica temporal: a composição *par* determina que seus nós internos devem ser exibidos em paralelo; a composição *seq* determina a exibição de seus nós em seqüência; e a composição *excl* (exclusiva) especifica que seus nós não podem ser exibidos simultaneamente. Além de composições, relacionamentos em SMIL podem ser definidos através de eventos, como *begin* (início de apresentação), *end* (término de apresentação) e *click* (clique do mouse). A Figura 2:2 apresenta um documento SMIL simplificado. A composição *seq* definida nas linhas 08-18 especifica a exibição, em seqüência, de suas composições internas. A composição *par* (linhas 09-13) determina a exibição em paralelo de um vídeo, um áudio e uma imagem: *video1*, *audio1* e *img1*. O elo da linha 12, definido juntamente com a imagem[13], especifica que a exibição dessa imagem deve terminar (atributo *end*) ao final da exibição de *audio1* (valor do atributo *end* especificado como o evento *audio1.end*). A outra composição *seq* (linhas 14-17) determina que dois áudios sejam exibidos em seqüência. Pela semântica da composição *seq* mais externa, esses dois últimos áudios serão exibidos após o término da composição paralela.

```
01. <smil>
02.  <head>
03.   <layout>
04.    ...
05.   </layout>
06.  </head>
07.  <body id=" ">
08.   <seq>
09.     <par>
10.       <video id="video1" />
11.       <audio id="audio1" />
12.       <img id="img1" end="audio1.end" />
13.     </par>
14.     <seq>
15.       <audio id="audio2" />
```

[13] O nó imagem é definido por meio do elemento XML *img*, enquanto o elo é definido utilizando o atributo XML *end* desse elemento.

```
16.       <audio id="audio3" />
17.     </seq>
18.    </seq>
19.  </body>
20. </smil>
```

Figura 2:2 - Exemplo de um documento SMIL.

Visando incorporar à HTML as facilidades para autoria de documentos com sincronização temporal presentes em SMIL, foi proposta, em 1998, para padronização no W3C, a linguagem HTML+TIME (W3C, 1998c).

2.4. NCL

Nested Context Language - NCL (Muchaluat-Saade et al., 2003; Muchaluat-Saade, 2003) é uma linguagem declarativa modular baseada em XML (e estruturada em 11 áreas funcionais) para especificação de documentos hipermídia. Fazendo uso das funcionalidades XML já descritas, NCL reusou, sempre que possível, módulos definidos pela linguagem SMIL (Muchaluat-Saade, 2003).

Composições em NCL, assim como em SMIL, podem conter um conjunto de nós, que podem ser objetos de mídia ou outros nós de composição. Entretanto, em NCL, composições (elemento *composite*) não possuem semântica temporal, mas de inclusão e, conseqüentemente, sua semântica é dada por seus elos.

A Figura 2:3 mostra um exemplo de um documento NCL similar ao documento SMIL da Figura 2:2. Para se obter a semântica do documento SMIL, elos NCL devem ser especificados, sincronizando os nós de cada composição. Esses elos NCL devem refletir a sincronização paralela e seqüencial das composições SMIL e também refletir o elo SMIL entre *audio1* e *img1*. Elos NCL são definidos separadamente dos nós por eles relacionados, estando agrupados em bases de elos (elemento *linkBase*). Essa abordagem permite, por exemplo, o reuso de nós sem a herança obrigatória de elos. Outra opção para especificar esse documento NCL é através de atribuição semântica, no caso paralela ou seqüencial, a composições NCL. Como será visto, isso é possível através de *templates* de composição.

```
01. <ncl>
02.  <head>
03.   <layout>
```

```
04.    ...
05.   </layout>
06.    ...
07.  </head>
08.  <body id=" ">
09.   <composite>
10.     <composite>
11.       <video id="video1" />
12.       <audio id="audio1" />
13.       <img id="img1"/>
14.       <linkBase>
15.         ...
16.        <link id="link1" xconnector="finishes.xml">
17.         <bind component="audio1" role="on_x_presentation_end"/>
18.         <bind component="img1" role="stop_y"/>
19.        </link>
20.         ...
21.       </linkBase>
22.     </composite>
23.     <composite>
24.       <audio id="audio2" />
25.       <audio id="audio3" />
26.       <linkBase>
27.         ...
28.        <link id="link2" xconnector="meets-start.xml">
29.         <bind component="audio2" role="on_x_presentation_end"/>
30.         <bind component="audio3" role="start_y"/>
31.        </link>
32.         ...
33.       </linkBase>
34.     </composite>
35.     <linkBase>
36.       ...
37.     </linkBase>
38.    </composite>
39.  </body>
40. </ncl>
```

Figura 2:3 - Exemplo de um documento NCL.

Elos NCL são definidos através de conectores hipermídia. Um conector especifica uma relação de forma independente do relacionamento, ou seja, não especifica quais serão os nós relacionados. Elos que representam um mesmo tipo de relação, mas que interligam nós distintos, podem reusar um mesmo conector (aproveitando toda sua especificação). Um conector hipermídia especifica um conjunto de pontos de interface, chamados papéis (*roles*). Para a criação de um elo, faz-se referência a um conector e define-se um conjunto de *binds*, que associam cada extremidade do elo (ponto de interface de um nó) a um papel do

conector. Conectores são especificados através da linguagem XConnector (Muchaluat-Saade, 2003) de NCL.

Na Figura 2:3 é possível observar a especificação de dois elos (por clareza, os demais elos do documento foram omitidos). O elo (elemento *link*) *link1* define que, ao término de *audio1*, deve ser interrompida a apresentação de *img1*. Essa semântica é obtida por meio do conector especificado no documento *finishes.xml*, onde são declarados os papéis *on_x_presentation_end* e *stop_y*. Os elementos do tipo *bind* desse elo fazem a associação do áudio e da imagem com os papeis definidos pelo conector. Já o elo *link2* define que *audio3* deve ser iniciado após o término de *audio2*, utilizando o conector *meets-start.xml*, que possui essa semântica. Os *binds* desse elo mapeiam os dois componentes do tipo áudio com os papéis *on_x_presentation_end* e *start_y* desse conector. Como pode ser observado, o elo *link1* do documento NCL representa o elo SMIL definido por meio de evento no documento da Figura 2:2, enquanto o elo *link2* representa o comportamento seqüencial obtido pelo uso da composição *seq* em SMIL. Ao contrário de SMIL, ambos os relacionamentos são definidos da mesma forma em NCL.

Para melhor ilustrar o uso de conectores, a Figura 2:4 apresenta um conector *R* representando uma relação com três papéis distintos, que significam três tipos de participantes da relação. A Figura 2:4 também mostra dois elos, *l1* e *l2*, reusando *R*. Enquanto o conector define o tipo de relação, o conjunto de *binds* de um elo define os participantes. O elo *l1* especifica três *binds*, interligando os nós *A*, *B* e *C* aos papéis de *R*. O mesmo ocorre com *l2*, só que interligando um conjunto diferente de nós (*B*, *C* e *D*). Os elos *l1* e *l2* definem relacionamentos diferentes, já que interligam conjuntos distintos de nós, mas representam o mesmo tipo de relação, pois usam o mesmo conector.

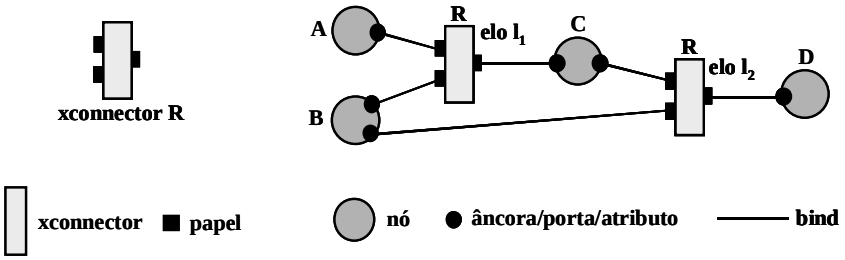

Figura 2:4 - Exemplo de elos NCL e de reuso de conectores.

Um *template* de composição, como mencionado, pode ser utilizado para embutir semântica, por exemplo temporal, em um nó de composição. *Templates* de composição são definidos através da linguagem XTemplate (Muchaluat-Saade, 2003) de NCL. A definição de um *template* de composição é feita através de um vocabulário, que especifica tipos de componentes, tipos de relações (modeladas por conectores) e pontos de interface. A definição também pode incluir restrições sobre elementos do vocabulário e conter instâncias de componentes e relacionamentos entre componentes (modelados por elos).

Um exemplo do uso de um *template* de composição pode ser visto na Figura 2:5, onde é definido que um áudio deve ser sincronizado com um *logo* (sincronismo de início e término de apresentação), e que cada trecho do áudio deve ser sincronizado, de forma similar, com a respectiva legenda[14]. Quando esse *template* é herdado por uma composição com um nó de áudio e três legendas (como na Figura 2:5), a composição passa a conter elos de sincronismo entre as legendas e o áudio, e entre o áudio e o *logo*. É importante ressaltar que o *logo*, definido como instância de componente no *template*, também passou a ser contido pela composição, após o processamento do *template*[15].

[14] Note que o conector utilizado na especificação dos elos de sincronismo para término de apresentação (conector *P*, ou *finishes*, na Figura 2:5) é o mesmo utilizado no exemplo da Figura 2:3.

[15] O *Processador de Templates* é responsável por atribuir a semântica de um *template* à composição que o referencia.

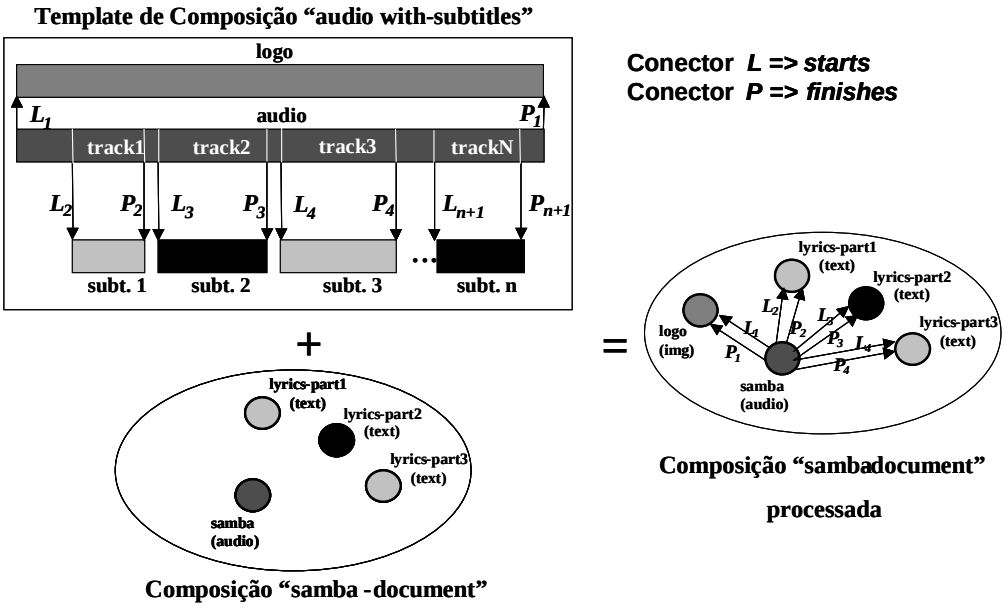

Figura 2:5 - Exemplo do uso de *templates* de composição.

Maiores detalhes sobre as especificações de *templates* de composição serão abordados no próximo capítulo, ao se apresentar a proposta de extensão da linguagem XTemplate.