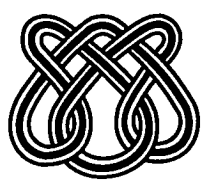
UNIVERSIDADE DE SÃO PAULO

xRot: Um Guia de Referência

Daniel Facciolo Pires
Maria da Graça Campos Pimentel

Nº 46

NOTAS DIDÁTICAS



Instituto de Ciências Matemáticas de São Carlos

UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação
ISSN 0103-2585

xRot: Um Guia de Referência

Daniel Facciolo Pires
Maria da Graça Campos Pimentel

Nº 46

NOTAS DIDÁTICAS

São Carlos – SP
Out./2000

xROT: UM GUIA DE REFERÊNCIA

Um roteiro para autoria de aplicações que manipulam documentos XML na Web, denominado *xRot*, foi criado com o objetivo de apoiar o desenvolvimento de aplicações que se preocupam com o intercâmbio de informações. Procurando facilitar a utilização desse roteiro por parte dos desenvolvedores de aplicações, este guia de referência para o roteiro *xRot* apresenta as 4 fases do *xRot* de forma sucinta e ilustrativa através de exemplos e figuras do estudo de caso C2000ML. O C2000ML foi desenvolvido utilizando o roteiro *xRot* (Pires, 2000). Inicialmente é apresentada a *ArqGDE* subjacente às aplicações geradas com o auxílio do *xRot*.

Arquitetura *ArqGDE*

A Figura 1 ilustra uma arquitetura típica de aplicações na Internet que geram dinamicamente documentos HTML como resposta a uma solicitação do *browser* - cliente. Nessa figura: **(A)** ilustra a requisição enviada pelo *browser* (cliente) ao servidor WWW; **(B)** ilustra a ativação, pelo servidor, de programas de acordo com a interface CGI; **(C)** representa o fato de que estes programas muitas vezes realizam consultas a banco de dados associados à aplicação; **(D)** ilustra a passagem ao servidor dos documentos HTML gerados dinamicamente pelos programas; e **(E)** representa o envio dos documentos pelo servidor ao *browser* cliente.

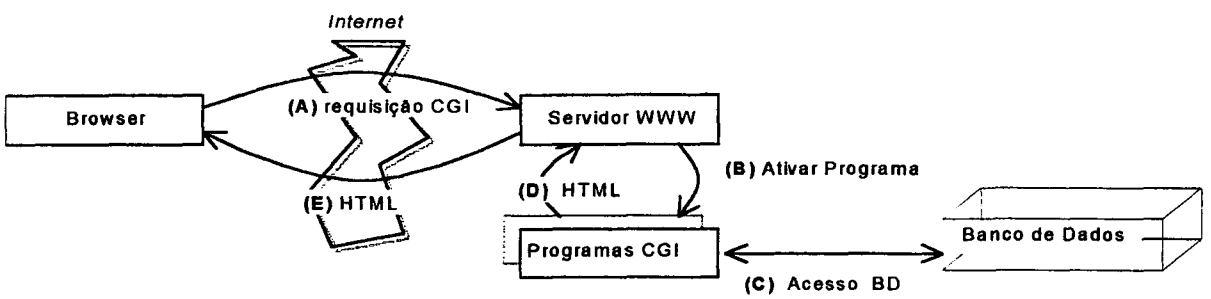


Figura 1 - Arquitetura típica de aplicações na Internet (Pimentel et al., 2000)

Já a Figura 2 apresenta a *Arquitetura para Geração de Documentos Estruturados - ArqGDE*. Essa arquitetura estende a arquitetura típica explorando a utilização de especificações XML (na forma de documentos DTD) e XSL na geração dos documentos por parte das aplicações. A *ArqGDE* é subjacente às aplicações geradas com o auxílio do roteiro *xRot*. Na figura: **(A)** representa a requisição enviada pelo *browser* (cliente) ao servidor WWW; **(B)** representa a ativação, pelo servidor, de programas de acordo com a interface CGI; **(C)** indica a seleção, pela aplicação, de um entre vários DTDs suportados; **(D)** indica a utilização do DTD apropriado pelo programa; similarmente, **(E)** e **(F)** indicam a seleção e utilização, respectivamente, de uma entre várias *style sheets* XSL; como no caso da arquitetura típica, a aplicação pode utilizar os recursos de um gerenciador de banco de dados, interação essa ilustrada na Figura 9 por **(G)**, para obtenção da informação a ser repassada ao cliente; **(H)** indica que a aplicação combina a especificação dada no DTD com a informação obtida junto ao banco de dados e a *style sheet* XSL apropriada para a produção do documento XML — que são repassados ao servidor WWW; e finalmente, ***I*** indica que essas informações são passadas ao *browser* (cliente) de modo transparente pelo servidor — isto é, o servidor não dá tratamento especial aos dados enviados.

Portanto, no contexto da *ArqGDE*, é função do cliente processar os documentos XML e XSL recebidos. De fato, os *browsers* mais recentes trazem embutidos ou podem ser estendidos com

processadores XML e XSL que permitem apresentar a informação correspondente de forma transparente ao usuário.

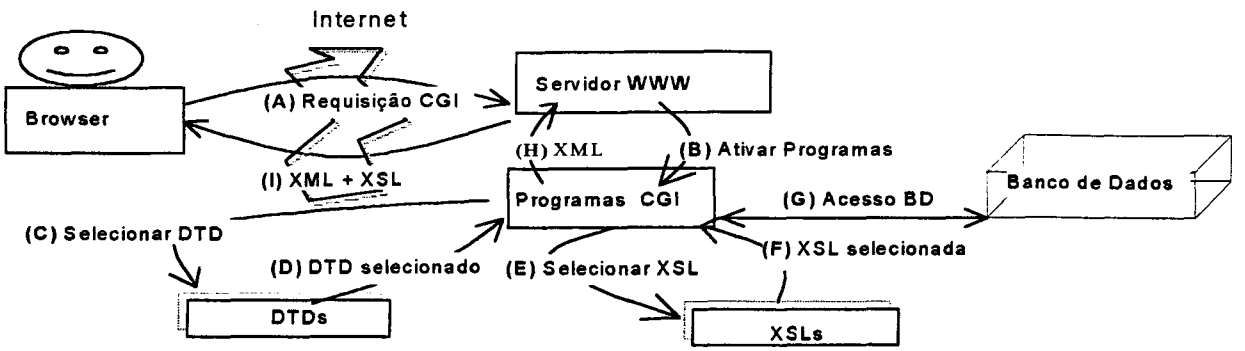


Figura 2 - Extensão da arquitetura típica - *ArqGDE* (Pimentel et al., 2000)

1ª Fase: Definição de Documentos XML

A FASE 1 do *xRot* estabelece que para cada tipo de documento HTML resultante dos serviços oferecidos pela aplicação (passo **(D)** da Figura 1), deve ser definido um documento XML que corresponda a essa instância. Os programas computacionais que serão criados na Fase 4 do roteiro serão responsáveis pela construção dos documentos XML definidos neste fase.

Como exemplo, considere o caso de uma aplicação que disponibiliza três diferentes formas de material didático no servidor: documentos contendo teoria, exercícios propostos e exercícios resolvidos. Neste caso, três tipos diferentes de documentos HTML podem ser gerados e, conseqüentemente, devem ser definidos três tipos diferentes de documentos XML correspondentes. Tomando o C2000ML como outro exemplo, foram definidos 4 tipos diferentes de documentos XML. Neste guia de referência, faremos referência a esses documentos XML como documentos X, Y, Z e T por motivos de simplificação. O nome dos documentos DTD e XSL também serão simplificados. Mais detalhes sobre os documentos definidos no C2000ML estão em (Pires, 2000).

2ª Fase: Criação de Documentos DTD

Definidos os diferentes tipos de documentos XML oferecidos pela aplicação na fase anterior, nesta fase são especificados os documentos DTD correspondentes (para cada tipo de documento XML definido deve-se criar um documento DTD) — os quais fornecem uma especificação formal da estrutura dos documentos XML.

Segundo (Bos, 1997), um documento DTD pode ser definido com base no conjunto de tabelas do banco de dados correspondente aos elementos relacionados à composição de um documento XML. As tabelas e as colunas do banco de dados são os elementos e os atributos, respectivamente, do documento DTD. Deve-se, então, analisar quais serão os elementos/entidades que irão compor o documento XML e como esses elementos/entidades estão definidos no banco de dados. Ainda, (Pires, 2000) afirma que o modelo de dados da aplicação também deve ser utilizado na criação dos documentos DTD, uma vez que o modelo representa o relacionamento entre os elementos/entidades envolvidos. A utilização deve ser feita da seguinte forma: 1) a todo relacionamento N:N indicado pelo modelo de dados conceitual da aplicação, o elemento correspondente no DTD é definido como uma composição dos elementos referentes às entidades envolvidas no relacionamento, bem como qualquer atributo exclusivo do relacionamento, e 2) a todo elemento participante de um relacionamento 1:N do lado "*1*" indicado pelo modelo de dados conceitual da aplicação, o elemento correspondente no

DTD é definido como uma composição do outro elemento referente a entidade envolvida no relacionamento do lado "*N*". Além disso, a cardinalidade dos elementos que compõem os elementos dos dois casos é definida de acordo com a cardinalidade dada pelo relacionamento — 0:1, 0:N, 1:1 e 1:N correspondem, respectivamente, a *entidade?*, *entidade**, *entidade* e *entidade+*. Para tal, relacionamentos N:N devem ser especificados pelos seus relacionamentos 1:N e N:1 correspondentes.

Como exemplo, tomemos o estudo de caso C2000ML onde os elementos *Course*, *Lecture*, *Slide*, *Handwritings*, *Images*, *LectureEdit* e *Session* compõem o documento XML X definido na Fase 1. O modelo conceitual do C2000ML da Figura 3 apresenta o relacionamento entre os elementos. Os elementos *Course*, *Lecture* e *Slide* são definidos como em (A), (B) e (C) também presente nessa figura. Os elementos *Course* e *Lecture*, por exemplo, estão assim definidos no banco de dados: `Course (CourseID integer, CourseCode char(8), CourseName char(50), Instructor char(50));` e `Lecture (LectureID integer, Date date, LectureTitle char(50), StartTime date, EndTime date, AudioFile char(30), VideoFile char(30))`. Dessa forma, o documento DTD chamado de XML do documento X apresenta-se como na Figura 4

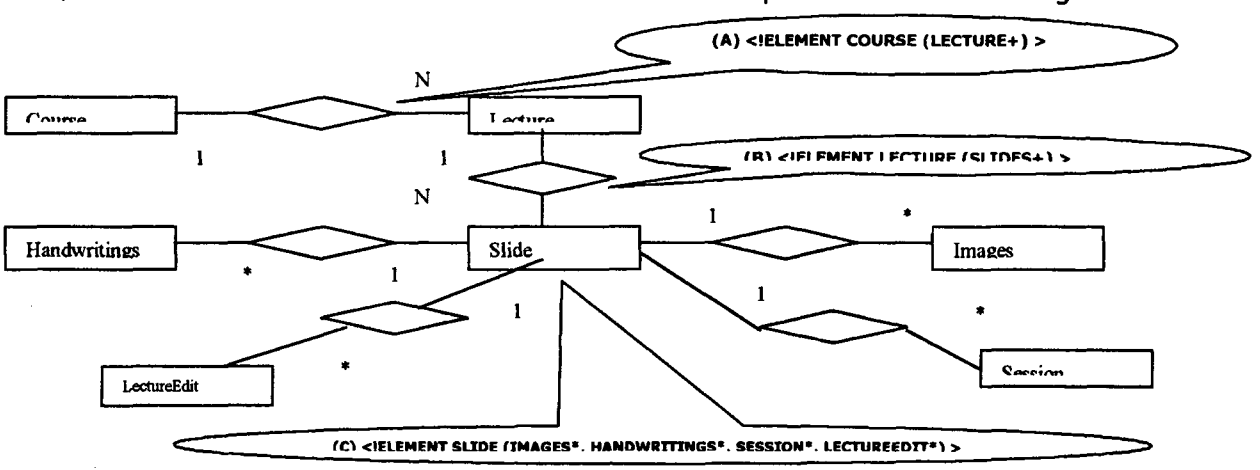


Figura 3 – Modelo Conceitual do exemplo C2000ML

```
<!ELEMENT COURSE (LECTURE+) >
<!ATTLIST COURSE
        COURSEID      ID      #REQUIRED
        COURSECODE    CDATA   #REQUIRED
        COURSENAME    CDATA   #REQUIRED
        INSTRUCTOR    CDATA   #REQUIRED >
<!ELEMENT LECTURE (SLIDE+) >
        LECTUREID           ID          #REQUIRED
        DATE                CDATA       #REQUIRED
        LECTURETITLE  CDATA             #REQUIRED
        STARTTIME           CDATA       #REQUIRED
        ENDTIME  CDATA       #REQUIRED
        AUDIOFILE           CDATA       #REQUIRED
        VIDEOFILE           CDATA       #REQUIRED >
<!ELEMENT SLIDE (IMAGES*, HANDWRITINGS*, SESSION*, LECTUREEDIT*) >
        . >     !- atributos do elemento slide de acordo com especificação no banco de dados
<!ELEMENT IMAGES EMPTY>
        . > !- atributos do elemento images de acordo com especificação no banco de dados
<!ELEMENT HANDWRITINGS EMPTY>
        . >!- atributos do elemento handwritings de acordo com especificação no banco de dados
        ...
```

Figura 4 - Documento DTD XML do exemplo C2000ML

Da mesma forma como foi definido o documento XML, os documentos DTD YML, ZML e TML também foram criados para os documentos Y, Z e T, respectivamente.

O exemplo da Figura 5 ilustra o caso onde se tem um relacionamento N:N envolvendo os elementos de um documento DTD. Um aluno pode se matricular em N disciplinas, e em uma disciplina podem ser

matriculados N alunos. Nesse caso, os elementos *Aluno* e *Disciplina* são definidos como em (A) e (B) também presente na Figura.5.

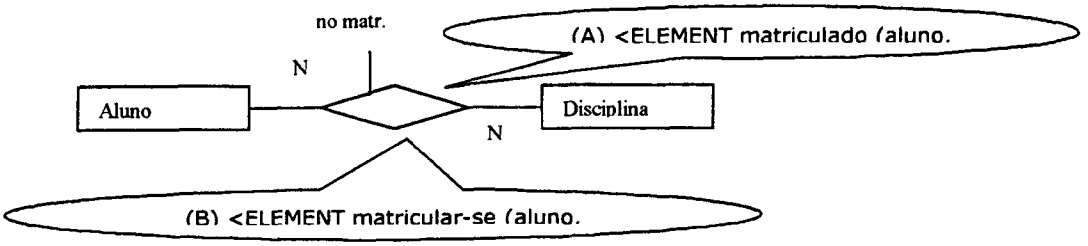


Figura 5 – Relacionamento N:N entre as entidades Aluno e Disciplina

3ª Fase: Criação de Documentos XSL

Esta fase corresponde à definição de *style sheets* responsáveis pela transformação de documentos XML em documentos a serem manipulados pelo cliente (passo (I) da Figura 2). Assim, se o objetivo é a apresentação, no browser do usuário, de documentos HTML, as *style sheets* definidas apresentam as declarações que, quando processadas por um processador XSL, transformam o documento XML no documento HTML esperado pelo cliente. Por outro lado, se o cliente é uma aplicação associada a um serviço de impressão, a *style sheet* deve especificar a transformação do documento XML em um documento RTF ou PostScript, por exemplo. Como exemplo complementar, se o cliente corresponde a uma aplicação interessada no documento XML propriamente dito — como o objetivo único de intercâmbio — nenhuma *style sheet* precisa ser definida nesta fase.

Uma *style sheet* XSL contém um conjunto de declarações que determinam como um documento XML deve ser transformado. Utilizando o exemplo de um cartão de visita que deve gerar um documento XML com informações pessoais de um visitante, a Figura 6a apresenta um documento XML de cartão de visita — essa figura contém, no início, a especificação de uma *style sheet* que deve ser utilizada em sua apresentação.

```
<?xml version="1.0"?>
<xsl:stylesheet
xmlns:xsl="http://www.intermidia
.icmc.sc.usp.br/~dfpires/xsl">
<cartao>
<nome> Jose da Silva </nome>
<instituição> ICMC/USP
</instituição>
<titulo>doutor</titulo>
</cartão>
```

```
<?xml version="1.0"?>
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/cartao">
<HTML><HEAD> <BODY> <H1>
Nome do Visitante: <xsl:value-of
select="nome"/></H1>
<H2>
Instituição do Visitante: <xsl:value-of
select="instituicao"/></H2>
<H3>
Titulo do Visitante:
<xsl:value-of select="titulo"/>
</xsl:template></H3>
</BODY>
</HEAD>
</HTML>
```

```
<HTML><HEAD>
<BODY><H1>
Nome do Visitante:
Jose da Silva</H1>
<H2>Instituição do
Visitante:
ICMC/USP</H2>
<H3>Titulo do
Visitante: Doutor
</H3> </BODY>
</HEAD></HTML>
```

Figura 6 - (a) documento XML, (b) documento XSL para transformação em HTML e (c) documento HTML resultante

A Figura 6b apresenta a *style sheet* XSL referenciada, a qual contém especificações de transformação do documento XML da Figura 6a para HTML. Nessa figura, a declaração: *<xsl:template match="/cartao">* indica que a todo elemento *<cartao>* encontrado no documento XML são aplicadas as declarações existentes nesse próprio template — isto é, são aplicadas as declarações encontradas entre este ponto e o *</xsl:template>* correspondente. Como outro exemplo, a declaração: *<xsl:value-of select="nome"/>* especifica que sua ocorrência deve ser substituída pelo conteúdo do elemento *<nome>* do documento XML. A Figura 6c apresenta o resultado da

transformação do documento XML da Figura 6a em documento HTML — de acordo com as especificações dadas no documento da Figura 6b.

Ao final desta fase, uma especificação XSL deve ter sido criada para cada possível forma de apresentação dos diferentes documentos XML produzidos pela aplicação. Isso significa que, para um mesmo documento XML (e seu DTD), mais de uma especificação XSL pode ser produzida. Como ilustração, o estudo de caso C2000ML definiu 5 especificações XSL para o documento XML X (X1, X2, X3, X4 e X5), e outras 7 especificações XSL para os documentos Y, Z e T.

4ª Fase: Criação de Programas para Geração de Documentos XML

Programas computacionais podem ser escritos com o objetivo de gerar documentos estruturados XML com conteúdo de uma base de dados e a partir da análise de um documento DTD. Esses programas correspondem à seqüência de passos de **(C)** a **(H)** da Figura 2. As principais atividades desses programas são: 1) o estabelecimento e fechamento de conexão com banco de dados, 2) a obtenção dos valores que são passados de acordo com interface CGI, e 3) a construção do documento XML utilizando o algoritmo GERAXML. No roteiro *xRot* devem ser construídos um programa computacional para cada tipo de documentos XML a ser gerado.

A atividade 3 descrita anteriormente corresponde a um dos passos importantes da *ArqGDE* (ilustrado por ***(H)*** na Figura 2). Essa atividade inclui a implementação do algoritmo *GERAXML* apresentado na Figura 7, que indica os passos a serem executados para que a aplicação, combinando a especificação do DTD com a informação obtida no banco de dados, realize a produção do documento XML associado a *style sheet* XSL apropriada.

```
FUNCAO GERAXML (refDTD, idXSL, Ps)
INICIO
      SE ELEMENTO_RAIZ
         ENTAO ESCREVA "<?xml version="1.0" encoding="ISO-8859-1"?>"
               ESCREVA "<?xml-stylesheet type=\"text/xsl\" href=\"nome_XSL.xsl\" ?>"
               ESCREVA "<!DOCTYPE aulaml SYSTEM \"nome_DTD.dtd\">"
      FIMSE
      ESCREVA "<ELEMENTO";
// o valor de ELEMENTO deve ser buscado com um consulta ao BD, podendo ter ou não Ps
      ENQUANTO (ATRIBUTO) FACA
         ESCREVA "ATRIBUTO="VALOR_ATRIBUTO"";
// o valor de ATRIBUTO deve ser buscado com um consulta ao BD, podendo ter ou não condições Ps
      FIMENQUANTO
      SE (ELEMENTO = EMPTY)
         ENTAO ESCREVA "/>";
      SENAO ESCREVA ">";
      ENQUANTO (SUBELEMENTO) FACA
         SE (SUBELEMENTO OU SUBELEMENTO?)
            ENTAO GERAXML(SUBELEMENTO)
         SENAO SE (SUBELEMENTO* OU SUBELEMENTO+)
                   ENTAO ENQUANTO (SUBELEMENTO) FACA
                            GERAXML(SUBELEMENTO)
                         FIMENQUANTO
               FIMSE
         FIMSE
      FIMENQUANTO
      ESCREVA </ELEMENTO>;
      FIMSE
FIM
```

Figura 7 - GERAML: Algoritmo para geração de documentos estruturados XML

O algoritmo *GERAXML* recebe como parâmetros o documento DTD, a referência a um documento XSL, bem como valores correspondentes à solicitação de serviços por parte do usuário (indicado por

Ps na lista de parâmetros). As estruturas de condição e de repetição do algoritmo foram baseadas nas regras de um documento DTD. O algoritmo *GERAXML* recebe, inicialmente, o elemento raiz do documento DTD — nesse caso, a referência para a *style sheet* é também processada. Os demais elementos definidos pela hierarquia do DTD são processados de maneira recursiva. As variáveis VALOR_ATRIBUTO e VALOR_ELEMENTO representam o conteúdo dos atributos e elementos, respectivamente, do documento XML. Esse conteúdo é buscado a partir do banco de dados da aplicação. As buscas podem ser condicionadas aos valores dos parâmetros *Ps*.

Tomando como exemplo o estudo de caso C2000ML, foram criados 4 programas Java para gerar os documentos XML X, Y, Z e T. A atividade de obtenção dos valores passados ao programa Java que gera documentos X de acordo com a interface CGI é mostrada na declaração a seguir :

String CourseID = Cs3302-a; String LectureID = 22; idPresent = X1.xsl; idDTD = XML.dtd

Essa atividade faz com que o programa Java receba os parâmetros associados à geração do documento XML X e repasse ao algoritmo GERAXML o elemento DTD guardado na variável *idDTD* (XML.dtd), a *style sheet* armazenada na variável *idPresent* (X1.xsl) e os parâmetros *Ps* associados à escolha do usuário armazenados nas variáveis *CourseID* e *LectureID* (Cs3302-a e 22, respectivamente). De posse desses valores, o algoritmo GERAXML gera o documento XML apropriado, passos de **(C)** a **(H)**. Finalmente, o documento X é devolvido ao *browser* (passo **(I)**) da Figura 2, e processado utilizando a *style* sheet X1.xsl para apresentação ao usuário. A Figura 8 ilustra o documento XML gerado pelo programa Java neste caso. O documento XML contém informações de *Course, Lecturer's Note* e quais os *Slides* existentes para a *Lecture*, elementos já utilizados em exemplos anteriores deste documento e que compõem o documento X.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="X1.xsl" ?>
<!DOCTYPE C2000ML SYSTEM "XML.dtd">
<course id="1" coursecode="cs3302a" coursename="Section A: Introduction to Software
Engineering - CS3302" instructor="Gregory Abowd"><lecture id="1" date="Jan.06.1999"
lecturetitle="Introduction" starttime="13:06:13" endtime="13:56:49" part="1"
audiofile="Jan.06.1999.1.ra" videofile="Jan.06.1999.1.rm">
<slide id="521" number="1" update="1" title="">
<imagens
src="http://c2000.cc.gatech.edu/zenpad/classes/cs3302a_99_Winter/Jan.6.1999.1/gif/
.gif"/>
</slide>
<slide id="530" number="10" update="0" title="Exams">
<imagens
src="http://c2000.cc.gatech.edu/zenpad/classes/cs3302a_99_Winter/Jan.6.1999.1/gif/
0.gif"/>
</slide>
.
.
</lecture>
</course>
```

Figura 8 - Documento XML gerado por um programa formado pelo GERAXML

Considerações Finais

Espera-se que o *xRot Card* tenha realmente facilitado a utilização do roteiro *xRot* por parte dos desenvolvedores de aplicações cujo objetivo é construir aplicações que manipulam documentos XML. Informações adicionais do roteiro *xRot* são encontrados em (Pires, 2000) (Pimentel e al., 2000).

Referências

(Bos, 1997) Bos, Bert. XML Representation of a Relational DataBase. [online]. Disponível na Internet em: http://www.w3.org/XML/RDB.html. 1997.

(Pimentel et al., 2000) Pimentel, M.G.C., Pires, D.F., Teixeira, C.A.C. xRot: Um Roteiro para Autoria de Aplicações Interoperáveis para a Web. IN: Simpósio Brasileiro de Sistemas Multimídia e Hipermídia, Natal - RN, 2000.

(Pires, 2000) Pires, D.F. *xRot*: Um Roteiro para Autoria de Aplicações que Manipulam Documentos XML na Web. Dissertação de Mestrado. ICMC/USP. São Carlos. Maio. 2000.