XTDL, XML Tool Definition Language

Daniela Carneiro da Cruz Pedro Rangel Henriques

{danieladacruz,prh}@di.uminho.pt
Departamento de Informática, CCTC
Universidade do Minho

Resumo

A necessidade de distribuir ferramentas, leva, tipicamente, à criação de um site onde o software em causa esteja disponível e possa ser descarregado. Porém atrás disso vem a necessidade de documentar o produto: o que é; para que foi concedido; como se usa; como se instala. E não basta fazê-lo por um único meio, isto é, através do site criado para o disponibilizar; logo de seguida outras necessidades de documentação vão surgir.
Neste artigo pretendemos apresentar um dialecto XML, XTDL (XML Tool Definition Language) que criámos para descrever ferramentas. Uma descrição em XTDL é independente do tratamento que se lhe pretende dar, fornecendo detalhes sobre autoria, versões em distribuição e documentação disponível. Além disso, permite incluir uma descrição livre e informação técnica, tal como arquitectura, tecnologias utilizadas, etc. XTDL permite ainda descrever um mapa de conceitos sobre o domínio em que a ferramenta se enquadra.
Apresentaremos depois um conjunto de ferramentas, XTS (XTDL Tool Set) que desenvolvemos para processar tais descrições. XTS inclui geradores para construir um site WWW, documentação (por exemplo na forma de um folheto) e uma colecção de diapositivos (por ex., para apoiar uma demonstração ou uma comunicação oral). Caso o mapa de conceitos seja incluído o gerador do site produzirá um grafo navegável, graças ao CMG (Conceptual Map Generator, um compilador de Prolog para dotty do Graphviz desenvolvido no nosso grupo).
O XTS será apresentado num browser da Internet, onde cada utilizador poderá submeter a sua própria descrição XTDL a fim de gerar o site e a documentação que pretenda. No entanto, para que a pessoa possa submeter informação no sistema, e proceder posteriormente à sua consulta e modificação, deverá estar registada no sistema. Com este fim, será utilizado uma base de dados e a tecnologia .NET (linguagem ASP e C#).
O paper inclui ainda, como caso de estudo, a descrição de uma outra ferramenta da nossa autoria, o LISSc (LISS Compiler).

1 Introdução

No contexto deste artigo, chamamos *ferramenta de trabalho em software* (ou simplesmente *ferramenta*, com frequência designada na gíria informática pelo termo inglês *tool*) a uma aplicação informática (conjunto de programas de computador que cooperam para um fim específico) que se destina a ajudar o programador em qualquer uma das fases do ciclo de desenvolvimento de programas, desde a análise ou especificação até aos testes finais ou documentação, auxiliando-o no desempenho da tarefa em causa (o que, em certos casos, pode mesmo significar a geração automática de código).

Desenvolver uma *ferramenta* é uma das possíveis tarefas de um programador, podendo ser motivada por um pedido externo (uma encomenda de determinado cliente), ou por uma necessidade do próprio. Tipicamente os programadores gostam de automatizar as tarefas que executam repetida e sistematicamente. Devido à complexidade que envolve e engenho que requer, esta tarefa é normalmente um desafio que se realiza com entusiasmo e emoção. Construída a *ferramenta* e comprovada a utilidade, surge a questão de a disponibilizar a outros programadores (doravante designados por *utilizadores*), os quais vulgarmente começam por pedir uma cópia que o autor tende a ceder por email ou simplesmente gravando num dispositivo de memória móvel (cdrom, flash, etc.).

A partir daí as coisas complicam-se, quer porque devido ao tamanho a cópia feita por um desses meios não é fácil, quer porque a instalação requer outros procedimentos (instalação de um outro pacote de *software*, escolha de directorias próprias, execução de uma makefile, etc.). Além dessas dificuldades de instalação surgem, depois, as de utilização; é típico que o *utilizador* solicite um guia, manual, ou outra qualquer forma de documentação para perceber a fundo a *ferramenta* e a poder usar correcta e convenientemente.

Face ao que foi exposto, urge encontrar uma solução que permita, por um lado disponibilizar a *ferramenta* de forma a que os potencias *utilizadores* a possam copiar facilmente (sem as limitações de tamanho associadas ao envio por email ou gravação num dispositivo móvel), e por outro lado, possam ter, no mínimo, documentação básica sobre instalação e uso.

A solução mais adequada, para responder com simplicidade a esses dois requisitos, é a criação de um página na Internet, ou seja um site WWW. Mas, como sempre as coisas começam simples e tornam-se complexas; se não cuidamos bem do seu nascimento, rápido se tornam imperfeitas e longe daquilo que nós gostaríamos de manter e os outros de visitar. Daí ser fundamental pensar à priori (antes do nascimento) *o que incluir no* site e *como o organizar*. Além disso, mais rápido do que se pensa, é necessário produzir outros materiais com o mesmo conteúdo mas com fins diversos.

Neste contexto que caracteriza um dos problemas com que o autor de uma *ferramenta* se depara no seu dia a dia, o presente paper vai sistematizar, na secção 2, o que está envolvido na disponibilização e documentação de um *ferramenta*. Depois, na secção 3, apresenta-se um dialecto XML, o XTDL, por nós concebido para descrever uma ferramenta de forma abstracta, isto é, independente do meio de divulgação que se quer produzir. Seguindo a boa tradição do nosso grupo, deve-se começar por especificar

formalmente o objecto com que se está a lidar, para depois surgir a possibilidade de processar a dita especificação e automatizar a produção do resultado. Assim e de seguida, na secção 4, introduz-se um conjunto de processadores, XTS, capazes de tomar como entrada uma descrição XTDL e produzir um site WWW (subsecção 4.1), um folheto (subsecção 4.2) ou um conjunto de diapositivos (subsecção 4.3). Inclui-se, também, a subsecção 5.4 onde se discute uma funcionalidade extra: a geração de um navegador conceptual, para incluir no *site*, a partir da descrição (opcional) do mapa de conceitos do domínio de conhecimento no qual a *ferramenta* se insere. Antes da conclusão, na secção 6, ainda se mostra a aplicação da linguagem especifica XTDL e da família de processadores XTS a uma *ferramenta* por nós desenvolvida, um compilador para a linguagem LISS; a secção 5 será, portanto, dedicada a apresentar este caso de estudo que constituiu a motivação para o trabalho aqui em discussão

2 Disponibilização e Documentação de Ferramentas

Uma *ferramenta* poderá ser descrita por um conjunto de características (ou descriptores) que, quer no site WWW, quer num folheto de apresentação —*datasheet*— quer num conjunto de diapositivos —*presentation*— estão sempre presentes e são a todos eles comuns. De modo a desenhar a linguagem XTDL correctamente—sendo independente do tratamento que se lhe pretende dar—é importante reflectir sobre a informação que deve ser fornecida acerca da ferramenta que se quer distribuir. Embora o sistema desenvolvido seja um contributo para a documentação de *software*, não foi de todo nossa intenção, neste trabalho pragmático, fazer um estudo teórico sobre essa área. Em [Agu03] o leitor pode encontrar uma aprofundada sistematização sobre documentação de *software*, em geral e em casos particulares, e uma extensa bibliografia sobre o tema.

Assumindo que a primeira preocupação é permitir o acesso para *download*, então é óbvio que é preciso *identificar cada uma das versões disponíveis*, associando ao identificador da versão, e à respectiva data, o URL do correspondente ficheiro. Naturalmente que a *identificação da ferramenta—designação* (ou nome abreviado) e *nome completo*—e uma *synopsis* (descrição breve) são também elementos imprescindíveis.

Falar da *equipe de desenvolvimento e manutenção* é, com certeza, outro dos requisitos. Assim será preciso descrever os *autores da ferramenta*, *instituição/empresa* a que estão associados, o *contexto* (projecto, disciplina, etc.) em que surge e *quem a mantém*. Opcionalmente, uma *data de criação* pode ser útil.

Naturalmente, convém explicar *porque foi concebida*, *para que serve*, *como está arquitectada* e *como foi desenvolvida* (estratégia de programação, tecnologia empregue, etc.). Ou seja, importa permitir a inclusão de uma *descrição*, em formato livre, e de *informação técnica*, mais estruturada, com referência à arquitectura e às tecnologias utilizadas.

É também conveniente listar a *documentação—manuais* de instalação e utilização, *exemplos, relatórios e artigos*—que a equipa de desenvolvimento/manutenção, ou outros, tenha produzido sobre a *ferramenta* em causa. Naturalmente que a indicação de *outras fontes relacionadas* é uma opção que deve considerada; sobretudo num site WWW é vulgar e útil encontrar apontadores para outros *sites* afins. Por fim, considerar a possibilidade de incluir (opcionalmente) uma *secção de novidades*, que permita

ao potencial *utilizador* ter uma visão rápida da evolução da *ferramenta* e estar alerta para determinadas questões prementes, afigura-se nos também importante.

Como acréscimo e porque defendemos o uso de *mapas de conceitos* (MCs) para descrever o conhecimento associado ao universo de discurso no seio do qual um determinado trabalho específico se desenvolve, decidimos permitir ainda a definição do MC sobre o domínio em que a ferramenta se enquadra. A intenção é possibilitar a geração de um navegador conceptual, sobre esse MC, caso ele seja especificado, o qual permita facilmente explorar o conhecimento nessa área. O MC a incluir pode ser descrito numa qualquer das linguagem existentes para o efeito. Por isso mesmo e no sentido de reutilizar esforços, decidimos usar uma de nossa autoria (muito simples e baseada em Prolog) para a qual já tínhamos desenvolvido um processador que gera o grafo e o navegador.

3 A linguagem XTDL

Nesta secção, pretende-se apresentar o Schema[Fal01, TMM01, BB01] que define o dialecto XML[BPSMM00, GP01, RH02] para descrição de ferramentas, o XTDL, no sentido de dar resposta aos requisitos identificados na secção anterior.

XTDL é constituída por um conjunto de elementos e atributos que permitem expressar os descritores que definem uma *ferramenta* independentemente da transformação e que foram acima identificados. O Schema que a seguir se apresenta reflecte esses elementos comuns e acrescenta alguns atributos que permitem controlar a geração das diversas formas de documentação contempladas pelo XTS.

3.1 O Schema

Para a geração das diferentes formas de documentar uma *ferramenta* (site que pode incluir um mapa de conceitos, diapositivos e folheto) apenas é necessário um único documento XTDL.

A documentação que se pretende gerar serve objectivos bastante distintos entre si: o site WWW, para consulta de informação geral da ferramenta—versões disponíveis, equipa envolvida no seu desenvolvimento, *download* da ferramenta, etc.—vai ser acedido por um público geral interessado em usar, ou apenas saber mais sobre os detalhes da ferramenta; o folheto (a *datasheet*), para apresentar brevemente a ferramenta—descrição técnica, arquitectura, autores, etc.—vai ser lido por um público, também genérico, que pretende conhecer a ferramenta de uma forma mais superficial; por fim, os diapositivos vão ser vistos e lidos por um público mais técnico com o intuito de conhecer e discutir mais detalhadamente a ferramenta.

Considerando estes aspectos, é unânime que serão necessárias, no nosso documento XTDL, as seguintes *tags* associadas ao elemento-raíz, `tool`, conforme se pode constatar na Figura 1:

- `team` (Equipa de desenvolvimento), o qual se divide em: `authors` (Autores); `filiation` (instituição à qual os autores pertencem); `projectName` (nome do projecto no âmbito do qual se desenvolveu a *ferramenta*); `projectDate` (data de início do desenvolvimento).
- `descritption` (Descrição) que inclui um texto livre (`introd` e `context`) e uma descrição técnica (`tech-desc`), o qual ainda permite juntar alguma explicações organizadas em items (`explanation`) e apresentar a arquitectura (`arch`).

- `distribution` (Versões disponíveis) que engloba três subelementos para identificar a versão, o ficheiro e seu *path* (`version`, `file` e `samples`).
- `docs` (Documentação disponível).
- `news` (Notícias relativas à ferramenta — nova versão, nova actualização, ...).
- `links` (Links relacionados); cada *link* tem um atributo `title` que permite dizer o texto a colocar na âncora.

Importa referir que o nome da *ferramenta* não é um elemento, mas sim um atributo da raiz (`id` e `name`). Seguindo a mesma lógica incluiu-se dois outros atributos associados a `tool`: `logo` e `lastUpdate` para permitir a indicação do logotipo criado para a *ferramenta* e para guardar a data da última actualização da descrição.

Como apresentado na figura 1, associado a cada uma das *tags* são permitidos 3 atributos: `site`; `presentation`; `datasheet`. Cada um destes atributos poderá ter um de dois valores possíveis: `yes` ou `no`. Consoante o valor desses 3 atributos, decidimos como é feita a geração em cada um dos 3 casos previstos. É de notar que cada um destes 3 atributos é opcional, pelo que deveremos pensar que estratégia seguir em caso de omissão. Assim, em caso de omissão de atributos, não é gerada nenhuma documentação. Definiu-se, no entanto, uma estratégia hierárquica para facilitar a escrita do XML.

Consideremos, a título de exemplo, o fragmento de descrição seguinte na descrição técnica (`tech-desc`) da ferramenta, esta é composta por uma explicação (`explanation`) e pela apresentação da sua arquitectura (`arch`):

```
<tech-desc site="yes">
    <explanation>
        <paragraph site="yes" presentation="yes" datasheet="yes">
            A brief description of our tool can be composed by one
            or more paragraphs.
            Each one could have one or more items (they are opcional).
            <item presentation="no" site="no">
                This first item I don't want that show up in my
                datasheet and presentation.</item>
            <item site="yes">The Second item will be shown in
            datasheet generation.</item>
        </paragraph>
        <paragraph site="no">
            Second paragraph
            <item>A simple item that will not appear in the
            site generated</item>
        </paragraph>
    </explanation>
    <arch>
        The figure below illustrates the architecture of our Tool.
        <image label="Old Architecture" site="no">old_arch.jpg</image>
        <image label="New Architecture">new_arch.jpg</image>
    </arch>
</tech-desc>
```

Para este exemplo, ao atributo `site` da *tag* `tech-desc` é associado o valor `yes`, indicando que se pretende que este elemento seja incorporado no *site*. No entanto, o

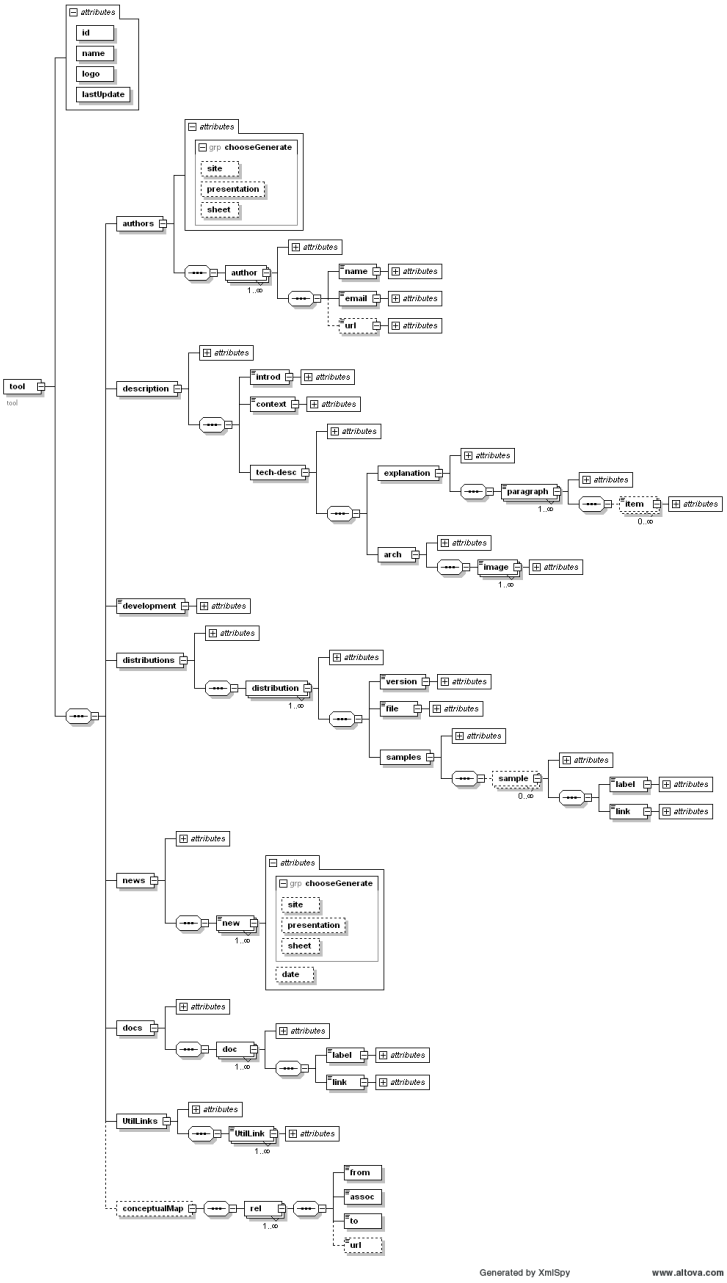

Figura 1: Schema da linguagem XTDL

elemento `explanation` não tem qualquer atributo definido; de acordo com a regra acima, o seu conteúdo seria ignorado na geração do *site* visto que o *default-value* é `no`. Contudo, seguindo uma estratégia hierárquica de herança, este nodo herda o valor do atributo `site` do seu pai (`tech-desc`), pelo que o seu conteúdo continuará a ser incluído no *site*. Apenas no caso do atributo `site` ser re-definido com o valor `no` em algum dos seus descendentes, o conteúdo será ignorado (como acontece no caso do primeiro `item` do primeiro `paragraph`, no segundo `paragraph` e na primeira `image` do elemento `arch`). O resultado da transformação aplicada ao XML apresentado (omitindo formatação própria de *site*, *datasheet* ou *presentation*) seria:

```
A brief description of our tool can be composed by one or more paragraphs.
Each one could have one or more items (they are opcional).
The Second item will be shown in datasheet generation.
The figure below illustrates the architecture of our Tool.
image = new_arch.jpg
```

Para a geração do mapa de conceitos, a linguagem XTDL prevê a definição de um conjunto de relações entre conceitos que serão agrupadas na *tag* `conceptualMap`, sendo cada relação descrita por quatro componentes a saber: `from`, `to`, `assoc` e `url`, servindo os dois primeiros para indicar a origem e destino de cada ligação, o terceiro para indicar o nome da relação que os une e por fim o quarto para associar o endereço da página respectiva.

4 O conjunto de processadores XTDL

Quando precisamos de utilizar uma ferramenta que nos seja útil, esperamos que esta seja simples, fácil de utilizar e intuitiva. Sendo assim, como *front-end* do XTS, para geração de documentação sobre ferramentas, desenvolvemos uma aplicação recorrendo à tecnologia .NET para a criação de uma *Aplicação Web* através da qual o autor pudesse submeter a descrição da sua *ferramenta* em XTDL e procedesse à geração da documentação pretendida, além de dispor da informação necessária à utilização desse conjunto de processadores. Esta *Aplicação Web* deve oferecer uma interface onde o utilizador tome as decisões relativas à geração dos 3 tipos de documentação disponíveis. A transformação de um documento XML usando XSL[Tho03, Hol01] e a apresentação do resultado é uma tarefa bastante simples na linguagem ASP. Apenas precisamos recorrer a 3 classes do C#: System.XSL, System.XML e System.XPath. O seguinte exemplo de código mostra o quão simples pode ser esta tarefa:

```
XslTransform Xslt = new XslTransform();
Xslt.Load(ourStylesheet);
XPathDocument doc = new XPathDocument("user.xml");
StringWriter fs = new StringWriter();
Xslt.Transform(doc, null, fs, null);
```

Essencialmente, após a submissão do documento XML do utilizador, é utilizado um objecto de transformação — XslTransform — onde está presente uma das nossas stylesheet — Xslt.Load(ourStylesheet), para efectuar a transformação — Xslt.Transform(doc, null, fs, null) — armazenando o resultado numa string — `fs` — que depois poderá ser utilizada para escrever a documentação pretendida (dentro das 3 hipóteses fornecidas).

Em cada uma das subsecções seguintes apresentamos a estratégia seguida para a geração de cada documentação final.

4.1 SG: Gerador de Sites

A *linguagem destino* utilizada na geração do *site* foi obviamente HTML [Ish04].

O site WWW será certamente o caso mais abrangente, isto é, aquele no qual utilizaremos a maioria dos elementos definidos no Schema apresentado na subsecção 3.1. Será usada a informação completa — se assim decidido pelo autor — relativamente: aos autores (`name, email, url`); à descrição contextual, histórica e técnica da ferramenta; às versões acessíveis para *download* e aos exemplos de utilização da ferramenta; aos documentos complementares disponíveis; aos *links* definidos para outras páginas; e ao mapa de conceitos.

Na geração desse site, pretendemos seguir um *padrão* que pode ser esquematizado da seguinte forma: o logotipo da *ferramenta* no canto superior esquerdo; o nome da *ferramenta* centrado no topo; o corpo do site subdividido em duas partes — à esquerda, os links para as componentes da descrição eleitas pelo autor para aparecerem na sua página; e no centro, a informação sobre a *ferramenta* correspondente a cada um desses links; na base da página, informação relativa aos contactos relacionados com a *ferramenta*.

Para a geração deste site WWW padrão, são necessárias várias travessias ao documento XTDL:

- a primeira para geração dos links que aparecerão na barra esquerda da página, de acordo com o valor do atributo `site`;
- a segunda para a geração da informação relativa aos *links* da travessia anterior. Esta travessia poderá, na prática, ser subdividida em tantas travessias quantas as páginas diferentes precisamos de gerar para esses links.

Para concretizar a ideia anterior, referimos que no caso do elemento `description` decidimos criar uma nova página, diferente da inicial, onde apresentaremos a descrição técnica da *ferramenta* acompanhada da arquitectura. O mesmo vai acontecer com o elemento `distribution`, para o qual também será gerada uma nova página com a informação e os *links* relativos às versões disponíveis para *download*.

A título de exemplo, mostra-se a seguir a folha de estilo XSL responsável pela primeira travessia.

```
<!-- left column -->
<td width="20%" valign="top">
    <table>
        <xsl:if test="//tool/description[@site='yes']">
            <xsl:apply-templates  select="tech-desc" mode="first" />
        </xsl:if>
        <xsl:if test="//tool/distributions[@site='yes']">
            <xsl:apply-templates  select="distributions" mode="first" />
        </xls:if>
        <xsl:if test="//tool/docs[@site='yes']">...</xsl:if>
        <xsl:if test="//tool/news[@site='yes']">...</xsl:if>
        <xsl:if test="//tool/links[@site='yes']">...</xsl:if>
    </table>
</td><!-- end left column -->

<!-- TECHNICAL DESCRIPTION -->
```

```
<xsl:template match="tech-desc" mode="first">
       <xsl:if test="(count(@site)=0]) or (@site='yes')">
          <tr><td bgcolor="#F4FEFF">
              <a href="#tech-desc">Technical Description</a></td></tr>
        </xsl:if>
    </xsl:if>
</xsl:template>
...
```

4.2 DG: Gerador de Folhetos

A *linguagem destino* utilizada na geração do folheto foi LaTeX[Lam94, GMS94].
Na geração deste documento, a ser impresso em papel para distribuição, a estratégia adoptada será diferente da proposta na secção 4.1, já que o que se pretende neste folheto é apenas uma breve apresentação da *ferramenta* e não a sua descrição por completo. Utilizaremos, além da identificação da *ferramenta* e dos autores envolvidos, a informação contida no elemento `description`—introdução, contexto, explicação técnica e arquitectura—e incluiremos um *link* para a página da *ferramenta*. No entanto, irá ser descartada informação, que neste caso nos parece irrelevante, como os *links* para outras páginas e, claro, os *links* para *download*, quer do executável correspondente às versões disponíveis, quer dos ficheiros relativos aos exemplos.

Para exemplificar o processamento de listas seguindo a herança do valor do atributo de controlo `sheet`, consideremos, em particular, o caso da geração do campo *autores* que deve figurar no folheto. Teremos de ter em atenção que a *tag* `authors` terá de ter o atributo `datasheet="yes"` (só assim deverá ser gerado o fragmento LaTeX `\author`); além deste atributo a verdadeiro, será necessário que pelo menos um dos seus descendentes não tenha o atributo `datasheet` re-definido para `no`.

```
<!-- TEMPLATE FOR "AUTHOR" -->
<xsl:template match="authors">
<xsl:if test="count(((author[@datasheet='yes'])
              | (author[count(@datasheet)=0])) &gt; 0">
  \author{<xsl:for-each select="(author[@datasheet='yes'])
              | (author[count(@datasheet)=0])">
<xsl:apply-templates select="."/>
          </xsl:for-each>}
</xsl:if>
</xsl:template>

<xsl:template match="author">
<xsl:variable name="pos"
     select="count(following-sibling::author/name[@datasheet='yes']
             | following-sibling::author/name[count(@datasheet)=0])
               - count(following-sibling::author[@datasheet='no'])"/>
<xsl:choose>
  <xsl:when test="$pos &gt; 0">
            <xsl:apply-templates select="name"/> \and </xsl:when>
  <xsl:otherwise><xsl:apply-templates select="name"/></xsl:otherwise>
</xsl:choose>
</xsl:template>
```

```
<xsl:template match="name">
<xsl:choose>
  <xsl:when test="(count(@datasheet)=0) or (@datasheet='yes')">
            <xsl:value-of select="."/></xsl:when>
</xsl:choose>
</xsl:template>
<!-- FIM DE TEMPLATE -->
```

4.3 PG: Gerador de Diapositivos

Na geração deste conjunto de diapositivos existem algumas decisões que importa salientar. A primeira de todas é, obviamente, a *linguagem destino* escolhida; neste caso decidimos utilizar na geração dos diapositivos a classe beamer[Tan06] do LaTeX[OS06].

Uma vez que cada diapositivo deverá ter uma quantidade adequada de informação, deveremos pensar que estratégia seguir para não sobrecarregar nenhum deles. Assim, nos dois primeiros diapositivos é pacífico que devemos ter: no inicial, a informação acerca dos autores que irão realizar a apresentação, o tema (nome da *ferramenta*), o local e a data; e no segundo diapositivo, o conteúdo global da apresentação, organizado em tópicos (o que nos é facilitado em LaTeX pela possibilidade de usar o comando `\tableofcontents`, dispensando uma travessia ao XTDL, como acontecia no caso do site WWW para a criação dos *links* activos na página).

A partir deste diapositivo, os restantes deverão ser preenchidos com a informação retirada de cada uma das *tags* do XTDL com o valor `yes` no atributo `presentation`. No entanto, no caso do elemento `description` que se encontra-se dividida em 3 partes—a introdução, o contexto e a descrição técnica—é necessário uma estratégia particular. Assim, cada uma destas partes dará origem a um novo diapositivo: um para a introdução; outro para o contexto; e uma série de diapositivos para a descrição técnica, a qual já apresenta uma estrutura (composta pelos sub-elementos `explanation` e `arch`), pensada para a geração de diapositivos. A *tag* `explanation` é constituída por um conjunto de `paragraph`, em que cada paragraph deverá corresponder a um diapositivo. Por fim a arquitectura (*tag* `arch`) dará origem a uma diapositivo com o texto livre e um outro por cada imagem incluída.

4.4 CMG: Gerador do Mapa de Conceitos

O mapa de conceitos que aqui se pretende gerar, deverá ser descrito através de um conjunto de factos tendo como *linguagem destino* o Prolog. A base de factos assim gerada, será depois submetida a um compilador desenvolvido pelo nosso grupo que produzirá dot[ENea06].

A base de conhecimento a ser gerada em Prolog é composta por um conjunto de factos (`map/4`), que descrevem cada associação entre um par de conceitos, na forma `map(source,association,destination,url).`; este conjunto de factos permite-nos deduzir num grafo de dependências, que pode então ser representado graficamente de modo a que o *utilizador* nele possa navegar.

A stylesheet responsável por esta tradução é a seguinte:

```
<xsl:template match="conceptualMap">
<xsl:apply-templates select="rel"/></xsl:template>
```

```
<xsl:template match="rel">
  <xsl:variable name="from" select="from"/>
  <xsl:variable name="to" select="to"/>
  <xsl:variable name="url" select="url"/>
   map('<xsl:value-of select="$from"/>', <xsl:value-of select="$assoc"/>,
       '<xsl:value-of select="$to"/>', '<xsl:value-of select="$url"/>').
```

5 Caso de estudo: LISSc, O Compilador de LISS

Nesta secção apresentamos um exemplo de aplicação prática a uma das nossas ferramentas: LISSc— um compilador de uma Linguagem de programação imperativa com Inteiros, Sequências e Sets, concebida pelo nosso grupo.
Por sermos um grupo produtor de *ferramentas* e sentirmos as dificuldades de distribuição e disponibilização que explicámos na secção 1, aplicámos o conjunto de processadores XTS à descrição da nossa *ferramenta* em XTDL e nas subsecções seguintes apresentamos o resultado obtido, por aplicação das estratégias adoptadas (secção 4).

5.1 O *site* LISSc gerado

Na secção 4.1, apresentámos aquele que seria o padrão gerado pelo processador de site WWW e também a estratégia a seguir para obter um resultado como o que se mostra na figura 2. O documento XTDL sobre o compilador LISSc foi escrito de acordo com

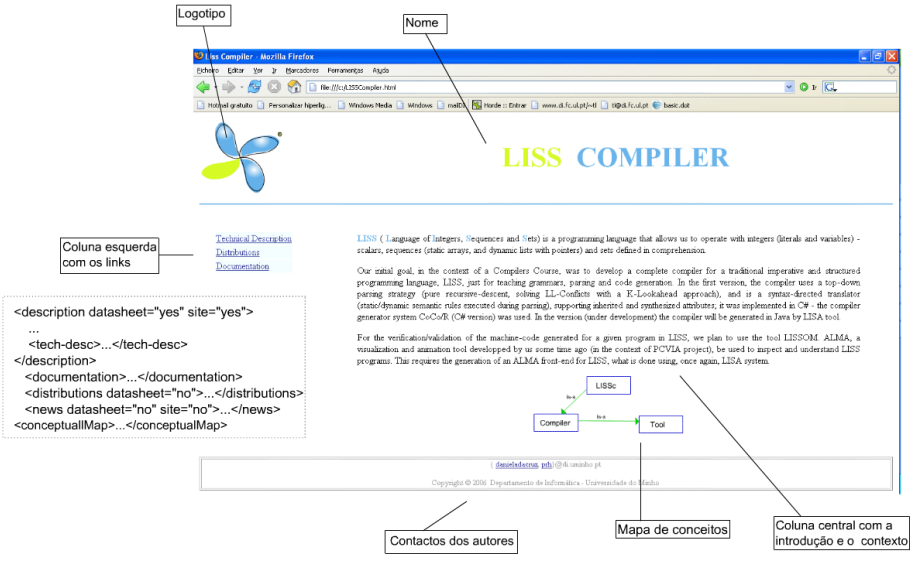

Figura 2: Página Principal do *Site* do LISSc gerado pelo SG

os princípios apresentados anteriormente, dando origem a um conjunto de páginas HTML onde poderemos encontrar a informação descritiva da *ferramenta*, bem como o conteúdo de todas as *tags* que tem valor `yes` no atributo `site`. Como já referido, adoptámos a estratégia de gerar novas página—ligadas à página principal (*homepage* visível na figura 2)—para: as versões acessíveis para *download* (ver figura 3); documentação existente; e *links úteis*.

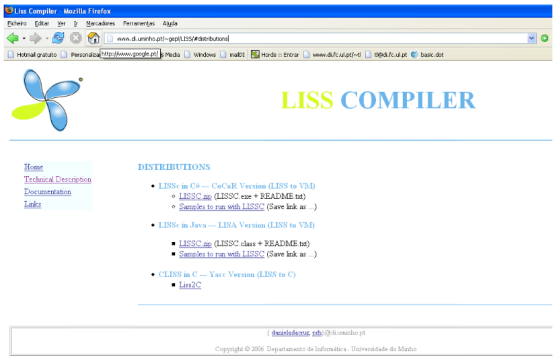

Figura 3: Página para distribuição do LISSc gerada pelo SG

5.2 O Folheto sobre o LISSc

Na secção 4.2 apresentámos a ideia e o objectivo de um folheto: informação breve e simples, destacando-se apenas os pontos relevantes referentes à *ferramenta*. Apresentámos ainda a estratégia adoptada para a geração do respectivo documento em LaTeX.

Nesta subsecção mostramos (abaixo) o resultado parcial desta estratégia, produzido pelo DG na *linguagem destino*:

```
\section{Technical Description}
\subsection{Description}
  LISSc was generated automatically from its Attribute Grammar using
  the Compiler Generator tool LISA.
  LISSc generates pseudo-code for the virtual machine VM.\\
  \begin{itemize}
     \item First version: was used CoCoR (compiler generator) and
           virtual machine MSP.
  \end{itemize}
```

e (na figura 4) o produto final, no formato PDF obtido após compilação do documento LaTeX gerado.

5.3 Os Diapositivos para apresentar o LISSc

Para o gerador de diapositivos de apoio a demos e comunicações a preocupação principal residia, como referido na secção 4.3, na quantidade de informação a distribuir

LISS Compiler

Daniela da Cruz Pedro Rangel Henriques

December 2, 2006

1 Introduction

LISSC is a Compiler for the toy imperative (or procedural) programming language LISS, that stands for Language of Integers, Sequences and Sets.

2 Context

At the beginning the language was conceived with pedagogical purposes, in the context of a Compilers Course. Nowadays it was recovered as a research project is the context of the Language processing group at University of Minho, to explore compiling techniques, generators and virtual machines.

3 Technical Description

3.1 Description

LISSC was generated automatically from its Attribute Grammar using the Compiler Generator tool LISA. LISSC generates pseudo-code for the virtual machine VM.

- First version: using CoCoR

4 Development

LISSC was developed at the Computing Department of he University of Minho by Daniela da Cruz, starting in the summer of 2005. LISS language was designed by Pedro Rangel Henriques and Leonor Barroca

- Version: 3.0

5 Site WWW

http://www.di.uminho.pt/ gepl/LISS

Figura 4: Folheto sobre o LISSc gerado pelo DG

por cada diapositivo.

No entanto, o dialecto XTDL foi pensado para deixar ao cuidado do *utilizador* a capacidade de controlar a situação e ultrapassar este problema, escolhendo a quantidade de informação a colocar em cada uma das *tags* (destacando-se, neste caso, as *tags* `paragraph` e `item`).

Apresentamos de seguida o produto final gerado pelo PG, evidenciando o resultado do critério de separação de diapositivos explicado (distribuição de texto por cada *slide*): a figura 5 mostra a estrutura, ou conteúdo, final da apresentação gerada; a figura 6 ilustra um diapositivo correspondente a um `paragraph` da *tag* `explanation` relativa à `tech-desc` do documento-fonte em XTDL.

Figura 5: Diapositivo que mostra a Estrutura da Apresentação do LISSc gerada pelo PG

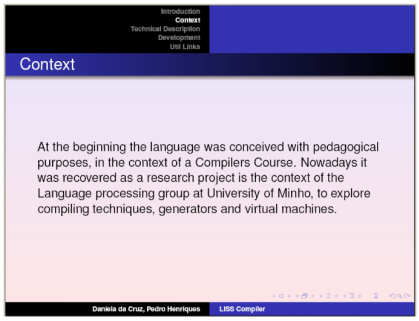

Figura 6: Diapositivo relativo a descrição técnica do LISSc gerado pelo PG

5.4 O Mapa de Conceitos do LISSc

O MC, cujo navegador irá complementar e enriquecer o site WWW, é algo bastante simples, mas que se nos afigura muito útil para a compreensão do domínio onde se enquadra a *ferramenta*. O respectivo gerador limita-se a: (1) transformar relações descritas na *tag* `rel`, do documento XTDL, em factos Prolog. Posteriormente (2) esses

factos Prolog serão traduzidos para dot do Graphviz[ENea06], de modo a que possa ser visto com o visualizador dotty[KGN06] ou WebDot[Ell06], o qual gera um diagrama navegável que é efectivamente o que nos interessa para adicionar ao *site*.

Apresentamos nesta subsecção o resultado das transformações (1) e (2) acima introduzidas. Para isso consideremos o seguinte extracto do documento XTDL referente ao compilador de LISS:

```
<conceptualMap>
     <rel>
         <from>lissc</from><assoc>is-a</assoc><to>compiler</to>
               <url>www.di.uminho.pt/~gepl/LISS</url>
     </rel>
     <rel>
         <from>compiler</from><assoc>is-a</assoc><to>tool</to>
     </rel>
 </conceptualMap>
```

Transformando (1) o mapa de conceitos anterior com a stylesheet apresentada na secção 5.4 obtemos o seguinte resultado:

```
map('lissc',is-a,'compiler', 'www.di.uminho.pt/~gepl/LISS')
map('compiler',is-a,'tool', '')
```

Este mapa de conceitos, submetido ao CMC (Conceptual Map Compiler) por nós desenvolvido, produzirá a imagem cuja descrição em dotty se apresenta a seguir.

```
digraph "LanguageProcessing@di.um.pt"
{
  node[shape=box,color="blue"];
  "LISSc" [ URL = "www.di.uminho.pt/~gepl/LISS" ]
  "Comppiler" [ URL = "" ]
  "Tool" [ URL = "" ]
  "LISSC" -> "Compiler" [ label="is-a" color="green"]
  "Compiler" -> "Tool" [label="is-a" color="green"]
}
```

Na figura2 (mostrada atrás) pode ver-se a dita imagem depois de ser processada pelo Graphviz, já incorporada no site WWW do LISSc.

6 Conclusão

Ao longo deste artigo apresentamos *uma aplicação* com o intuito de mostrar que *a anotação de documentos em XML e o seu processamento com as tecnologias XSL associadas* continua a ser *uma forma elegante e eficiente de programar*—especificar o problema e codificar uma resolução. Em muito pouco tempo e com uma dificuldade reduzida, resolvemos um problema complexo que nos surgiu no âmbito da nossa actividade como construtores de *ferramentas* de *software*, problema esse que é comum aos outros autores de *ferramentas*.

Usando um sistema de anotação (uma linguagem) específico de uma classe de problemas—definido formalmente através de um Schema XML—obtemos um especificação rigorosa

e declarativa de cada problema concreto dentro dessa classe. No caso discutido, descrevemos cada *ferramenta* concreta que pretendemos distribuir em XTDL, o dialecto XML criado para o efeito. É claro que a dita linguagem XTDL e os processadores associados não limitam a sua aplicabilidade à publicação de *ferramentas* de *software*, mas quisemos restringirmo-nos ao caso que serviu de motivação e que estava no nosso âmbito de trabalho.

Essa especificação declarativa enfatiza a estrutura do objecto que se quer tratar (as características da ferramenta a disponibilizar e documentar) e a sua semântica estática (regras contextuais que tem de ser observadas), de uma forma independente da transformação a operar. Além disso e por ser uma anotação específica de um domínio concreto de problemas, torna-se claramente fácil de usar e interpretar por quem trabalha nesse domínio. Recorrendo, depois, a um sistema de produção—formado por regras condição-reacção, também ele escrito rigorosamente num dialecto próprio de XML, o XSL—descreve-se a forma de processar a especificação referida acima.

Esta aproximação permitiu-nos pensar e desenvolver, não um processador, mas um conjunto de processadores que transformam a descrição XTDL de uma *ferramenta* em três tipos diferentes de suportes (meios de comunicação) para divulgação da dita *ferramenta*. É claro que várias outras transformações poderiam ser feitas para produzir outras peças documentais, mas pareceu-nos que as três apresentadas eram suficientes para ilustrar a ideia.

A quantidade de artefactos, actualmente existentes nos vários ambientes de programação, para reconhecer documentos XML e para os processar com folhas de estilo XSL, permitiu que se integrasse muito rápida e facilmente as componentes do XTS de modo a disponibilizá-las através de uma Aplicação Web. Para exemplificar a ideia, utilizaram-se 3 classes standard do ambiente .NET.

O trabalho realizado, como se disse atrás em tempo muito curto, terá, obviamente, de ser apurado no futuro próximo, quer quanto aos elementos previsto no XTDL, quer quanto ao XTS; aí teremos de melhorar a apresentação do resultado produzido por cada um dos geradores e de pensar em incrementar o conjunto, concebendo novos transformadores. Mas a lição retirada a nível da abordagem à resolução de problemas é que importa reter e discutir.

Agradecimentos

Os autores querem expressar o seu agradecimento a Elisabete Ferreira e Patrícia Oliveira pelo seu empenho e ajuda na implementação da linguagem XTDL e das ferramentas da *suite* XTS.

Referências

[Agu03] Ademar Aguiar. *Framework Documentation, a Minimalist Approach.* PhD thesis, DEEC, Faculdade de Engenharia da Universidade do Porto (FEUP), 2003.

[BB01] Paul V. Biron and Paul V. Biron. XML Schema part 2: Datatypes. `http://www.w3.org/TR/xmlschema-2`, May, 2001.

[BPSMM00] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, and Eve Maler. Extensible Markup Language (XML). World Wide Web Consortium, October, 2000. http://www.w3.org/TR/REC-xml.

[Ell06] John Ellson. WebDot Home Page, Dec., 2006. http:///www.graphviz.org.

[ENea06] John Ellson, Stephen North, and et al. Graphviz - Graph Visualization Software, Dec., 2006. http:///www.graphviz.org.

[Fal01] David C. Fallside. XML Schema part 0: Primer. http://www.w3.org/TR/xmlschema-0, May, 2001.

[GMS94] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Addison-Wesley, Reading, Massachusetts, 1994.

[GP01] Charles F. Goldfarb and Paul Prescod. *XML Handbook*. Prentice Hall, 4th edition, 2001.

[Hol01] G. Ken Holman. *Definitive XSLT and XPath*. Prentice Hall, 2001.

[Ish04] Masayasu Ishikawa. HyperText Markup Language (HTML) Home Page. http://www.w3.org/MarkUp/, February, 2004.

[KGN06] Eleftherios Koutsofios, Emden Gansner, and Stephen North. dotty - A Customizable Graph Editor, Dec., 2006. http://www.graphviz.org/cgi-bin/man?dotty.

[Lam94] Lambert Lamport. *LaTeX A Document Preparation System*. Addison-Wesley, Reading, Massachusetts, 2nd edition, 1994.

[OS06] Tobias Oetiker and Alberto Simoes. *Uma não tão pequena introdução ao LaTeX*. Dep. de Informatica, Escola de Engenharia da Universidade do Minho, 2006.

[RH02] José Carlos Ramalho and Pedro Henriques. *XML & XSL Da Teoria à Prática*. FCA Editora, 1.st edition, 2002.

[Tan06] Till Tantau. The LaTeX Beamer Class Homepage, Dec., 2006. http://latex-beamer.sourceforge.net.

[Tho03] Henry Thompson. The Extensible Stylesheet Language (XSL) Family. http://www.w3.org/Style/XSL/, 2003.

[TMM01] Henry S. Thompson, Murray Maloney, and Noah Mendelsohn. XML Schema part 1: Structures. http://www.w3.org/TR/xmlschema-1, May, 2001.