Universidade do Minho

Opção III - Projecto

X.*f*.X.

XML to Form to XML

<orientador> Jorge Rocha </orientador>

<autor email="jmaferreira@hotmail.com"> José Miguel Araújo Ferreira </autor>

<autor email="pgabidf@portugalmail.pt"> Pedro Gabriel Dias Ferreira </autor>

<autor email="ruiguerra@hotmail.com"> Rui Manuel A. S. R. Guerra </autor>

14 de Fevereiro de 2002

Resumo

Devido à crescente importância da eXtensible Markup Language, XML, no contexto da World Wide Web e como forma de organizar e partilhar dados, cresce de igual modo a importância de aplicações e ferramentas que visam facilitar o trabalho de programadores e utilizadores desta linguagem.

A XfX é uma aplicação que pretende contribuir tornando fácil e intuitiva a utilização da XML, como standard no armazenamento e organização de informação estruturada.

Ao fornecer uma interface amigável e num formato familiar a todos os utilizadores da Internet, acostumados a interagir com documentos HTML, a XfX liberta os seus utilizadores de terem que aprender ou conhecer todos os mecanismos necessários para gerar um ficheiro XML válido e bem formado.

A facilidade de inserção e edição de informação, bem como a clara definição da forma como esta se relaciona é o objectivo primordial da XfX.

Conteúdo

Capítulo 1

Introdução

A XML pertence à família de linguagens de *anotação*[1]. Estas linguagens funcionam todas elas de modo semalhante. De forma genérica, são compostas por um conjunto de pares de etiquetas (tags) que são colocadas à volta de palavras ou frases de forma a alterar o seu aspecto ou significado. Estes pares de etiquetas formam os chamados elementos.

A linguagem XML foi concebida para ser utilizada em outras áreas para além da web. O objectivo da sua concepção é de tornar esta linguagem num formato largamente aceite de partilhar informação entre as mais variadas aplicações. Por exemplo, se um jornalista ou escritor desejar partilhar as suas histórias na Internet, ou imprimi-la num formato próprio, ou até partilhá-la com amigos, pode anotar a sua história segundo as definições da XML Standard. Uma vez anotada, a sua história pode ser facilmente convertida noutros media.

Uma vez que a XML é definida num ficheiro em formato de texto, a criação de um ficheiro XML ou a anotação de um texto já existente pode ser feita facilmente através de um editor de texto. Tal solução deixa de ser prática ou até mesmo viável, quando se trata grandes volumes de informação, levando frequentemente à mal formação dos elementos a anotar, i.e., a etiquetas mal encadeadas e ao esquecimento de porções de informação.

Para ultrapassar estas dificuldades existem aplicações que têm por objectivo editar documentos XML. A *XfX* (*XML to FORM to XML*), é uma aplicação que pertence à família dos editores de XML.

Actualmente existem no mercado diversos editores de XML. No entanto apresentam quase sempre as mesmas limitações.

São normalmente editores com uma interface rudimentar ou de difícil utilização, bastante rígidas na forma como permitem a manipulação da informação no documento XML.

[1] Markup languages.

Um dos principais defeitos dos actuais editores de XML tem a ver com o facto do utilizador que pretende escrever ou alterar um documento XML tem necessariamente que conhecer a sua síntaxe. Um utilizador convencional tem portanto que obter formação para poder editar um documento XML. O processo não é, portanto, tecnológicamente transparente.

Além disso, estes editores integram normalmente aplicações que trabalham sobre todas as áreas tecnológicas do XML, sendo por consequência aplicações demasiado dispendiosas e que vão para além daquilo que se pretende obter.

A *XfX* é uma aplicação que pretende ultrapassar algumas dessas limitações, tendo-se no entanto a consciência da dificuldade que representam e por vezes a impossibilidade da sua total resolução. Sabendo tratar-se de um projecto ambicioso, foi aceite como um desafio, esperando que este projecto seja de grande utilidade para todos os seus utilizadores.

As linhas mestras que orientaram a concepção da aplicação XFX, foram:

- Utilização fácil e intuitiva;
- Não implicar conhecimentos profundos da tecnologia XML e outras tecnologias associadas;
- Portabilidade e independência de operação;
- Restringir-se ao objectivo de edição e anotação de informação em formato XML;
- Certificar a boa formação e validação do documento XML consoante o definido pela respectiva gramática;
- Sugestividade visual da forma como a informação se relaciona no documento;

1.1 O que é o X*fX*

A designação *XfX*, resulta da frase *XML to FORM to XML*.

Dito de outra forma a aplicação recebe um ficheiro XML que contém a estrutura e definição dos dados a representar, i.e., define a gramática do documento. Alternativamente recebe um ficheiro XML com a estrutura já definida para o documento, contendo eventualmente alguma informação já anotada.

De seguida, é gerado de forma dinâmica um formulário com uma representação visual diferenciada para cada estrutura e tipo de dados do documento, com especial atenção à forma como estes se relacionam ao longo do documento. Neste

formulário é possível proceder à inserção ou modificação dos dados no documento. Note-se que esta fase do trabalho é crítica, uma vez que o formulário deve reflectir a estrutura que o documento pode assumir. Tal constitui uma forma de certificação da boa formação e validação do mesmo, restringindo as possibilidades de edição aos elementos existentes no formulário. Devido à possibilidade de existirem estruturas de dados bastante complexas o formulário deve ser suficientemente flexível de modo a prever e oferecer soluções para todos os casos possíveis.

O formulário é gerado na linguagem HTML, que possui um aspecto simples e familiar aos utilizadores da WWW. Devido à necessidade de se alterar o código HTML gerado dinamicamente é utilizada uma linguagem de "scripting", a JavaScript, que permite ultrapassar as limitações do HTML que é uma linguagem estática por natureza.

Por fim é gerado o ficheiro XML com toda a informação anotada, respeitando as regras de boa formação e validação descritas no seu documento definidor. Este ficheiro está pronto a ser utilizado por outras aplicações ou partilhado entre utilizadores. Podendo ser reeditado a qualquer momento através do *XfX*.

Capítulo 2

Tecnologias utilizadas

De seguida faz-se uma breve introdução a cada uma das tecnologias utilizadas, às suas limitações e/ou adequações ao âmbito deste projecto e quais as razões que levaram à sua escolha.

2.1 XML (eXtensive Markup Language)

Já não há dúvidas de que a XML veio para ficar. Esta conseguiu alterar a *Web* e a forma como as coisas são feitas para nela funcionar. Além disso a XML veio confirmar uma forma já existente de guardar e organizar informação.

"Os criadores da XML quiseram que esta fosse usada em muitos outros sítios para além da Web. Era intenção destes que a XML se tornasse uma forma amplamente aceite de partilhar dados entre diferentes aplicações." [BAA00]

A XML é por muitos conhecida como *"prima"* da HTML e subconjunto ou *"filha"* da SGML. Esta linguagem foi concebida para ultrapassar as limitações da SGML mas sem a sua complexidade. De uma forma mais clara podemos dizer que a SGML é uma meta-linguagem, podendo ser utilizada para definir outras linguagens. HTML é uma linguagem fixa que não pode ser extendida ou criar subconjuntos de ela própria. XML é um subconjunto de SGML e pode ser utilizada para definir outras linguagens, tornando-a numa meta-linguagem.

Podemos utilizar a XML para criar estruturas que descrevam o conteúdo, sem atender à forma como este será exibido. Por outro lado o HTML descreve a forma como o documento é exibido, sem conter informações acerca daquilo que este trata.

Seguem-se algumas propriedades da XML[BAA00]:

- Suportada por uma variadade de aplicações;

- Compatível com SGML;
- Facilidade de escrever programas que processem XML;
- Os documentos são facilmente inteligiveis;
- Os documentos são de rápida e fácil concepção;
- Não existem ambiguidades;

2.2 HTML (HyperText Markup Language)

A HTML é utilizada para escrever documentos com o objectivo primordial de disponibiliza-los na *Web*. Esta linguagem consiste num conjunto de etiquetas que define como o documento é estruturado. Este tipo de linguagem de *anotação* permite definir as partes dos documentos, mas não a sua formatação, sendo esta definida pelo browser de forma a se adaptar as propriedades do mesmo, garantindo assim que o documento é exibido de forma correcta no ecrã do utilizador. Então pode dizer-se que a linguagem HTML corresponde a elementos ou *tags*, escritas em volta de blocos de texto, para definir a que partes do documento estes blocos correspondem.

A HTML é uma linguagem fácil de aprender e como constitui já um standard apresenta um aspecto familiar a todos os que utilizaram a *Web* pelo menos uma vez. Além disso, todos os sistemas operativos possuem pelo menos um *browser* compatível que permite visualizar este tipo de ficheiros com apenas um duplo clique no documento. Esta portabilidade, independência de plataforma e pelo facto de ser um standard levou à escolha desta linguagem como ferramenta para definir o formulário de preenchimento de edição de dados do *XfX*.

2.3 DTD (Document Type Definition)

O DTD pode ser visto como uma definição do documento, ao estabelecer regras para o parser seguir. Descreve o tipo de elementos que podem existir no documento e a forma como estes se relacionam. Um documento XML que esteja conforme a especificação do seu DTD é considerado válido.

Apesar de as ideias que estão na base do DTD serem bastante simples, a escrita deste nem sempre é intuitiva. Tal acontece porque usa uma sintaxe diferente da sintaxe da XML. Principalmente por esta razão decidiu-se não utilizar um DTD como elemento definidor da gramática do documento, recaindo essa opção sobre o *Schema* cujas vantagens sobre o DTD se explicam na próxima secção.

2.4 Schema

Uma das principais limitações dos DTD é a de que são escritas noutra *linguagem* diferente da XML, requerendo por isso duas tecnologias de *parsing* diferentes. Além disso o utilizador necessita de aprender uma linguagem adicional e sintaxe associada só para trabalhar com DTDs. Os *Schemas* são escritos utilizando as regras do XML standard.

Apresentam-se de seguida outras desvantagens dos DTDs em relação aos *Schemas*[BAA00]:

- Os DTDs não suportam namespaces (espaço de nomes), o que os vai tornar num futuro próximo de pouco utilidade, devido à crescente interconexão entre diferentes ficheiros XML;
- Outra desvantagem dos DTDs é a falta de mecanismos para suportar herança e sub-classes, sendo estes mecanismos facilmente implementados através de *Schemas*.
- Outra grande falha dos DTDs é a falta do conceito de tipo de dados, para além de PCDATA e CDATA, sendo os *Schemas* um poderoso meio para definir estruturas de dados simples ou complexas.

Estas e outras razões indicam que os Schemas serão o futuro standard de definição de documentos XML. Por isso, se justifica a escolha dos *Schemas* como elemento definidor de estruturas XML, a ser utilizada pelo *XfX*.

2.5 XSL (eXtensible Stylesheet Language)

A eXtensible StyleSheet Language é uma aplicação da tecnologia XML, que permite criar stylesheets especiais para formatar ficheiros XML. Ao contrário das CSS (Cascading Style Sheets) que consistem numa simples transformação de elementos do documento em atributos HTML, a XSL oferece um poderoso mecanismo de transformação e adaptação de estilos.

XSL utiliza regras padrão-reacção para condicionalmente aplicar formatação aos diferentes elementos, permitindo converter documentos XML num qualquer vocabulário de *anotação*. Ao longo deste projecto a XSL é utilizada para transformar condicionalmente o conteúdo XML em HTML, ou seja, para gerar dinamicamente os formulários a partir de um *schema*.

Cada stylesheet é um documento XML bem formado e válido. A XSL manipula os dados utilizando um conjunto de elementos e métodos invocados segundo o espaço de nomes *xsl*.

Um documento XML nunca é visualizado directamente no *browser*. Se este é utilizado sem nenhuma instrução explícita acerca da forma como deve ser visualizado, uma stylesheet por omissão será utilizada pelo *browser*. Caso contrário, existe uma instrução no documento que indica qual a stylesheet a ser utilizada para visualizar o mesmo.

Assim a aplicação receptora, recebe dois ficheiros: o documento XML e a respectiva stylesheet. Esta aplicação não necessita de ser um *browser* da *Web* mas caso o seja o resultado será um ficheiro HTML.

No âmbito do *XfX* a linguagem XSL é utilizada para gerar dinamicamente o formulário em HTML, com código *JavaScript* embebido. Tal solução foi considerada a mais indicada visto esta ser uma linguagem bastante poderosa e flexível e estar já de alguma forma standardizada, sendo suportada pela maioria dos *browsers* actualmente existentes.

2.6 JavaScript

A HTML fornece uma grande flexibilidade aos criadores de páginas. No entanto esta linguagem é estática por natureza.

A criação de CGI, que correm do lado dos servidores *Web*, permitem criar sites mais interessantes e mais interactivos.

Algumas aplicações exigem que o documento dinâmico seja executado do lado do cliente. Isto porque, permite uma maior portabilidade do mesmo, uma vez que não estamos dependentes de nenhum servidor *Web* específico.

A *JavaScript* é uma linguagem interpretada, ou seja, o computador tem que correr o código sempre que o programa é executado. Ao embeber comandos de *JavaScript* na página HTML, qualquer browser que suporte *JavaScript* pode interpretar os comandos e executar os mesmos. Desta forma, estes pequenos *scripts* são executados no lado do cliente em vez de serem executados do lado do servidor.

A *JavaScript* fornece um leque alargado de funções pré-definidas e comandos. Os *scripts* desta linguagem permitem modificar o documento HTML exibido pelo *browser*, efectuar cálculos matemáticos, abrir novos URLs, etc...

O código para executar estas funções pode ser embebido na página e executado quando a página é carregada. Alternativamente podem ser escritos métodos que contém código que será despoletado por eventos especificados.

Foi este poder que nos levou à escolha da *JavaScript* como a linguagem de *scripting* ultrapassando assim as características estáticas da HTML.

Capítulo 3

Arquitectura

3.1 Criação de um formulário a partir de um Schema

Um mesmo stylesheet (xsd2html.xslt) processa *schemas* e ficheiros *.xfx*. Assim se este stylesheet for aplicado ao *schema* o procedimento é criar o formulário em branco, pronto a preencher. No caso de ser aplicado ao ficheiro *.xfx*, o stylesheet lê a localização do *schema* e abre esse ficheiro.

A primeira missão do stylesheet é descobrir qual é a raíz do documento. Se nos for fornecido o *.xfx*, a solução da questão é óbvia, ou seja, a raiz do *.xfx* é a raíz do documento. No caso de nos apresentarem um *schema* com múltiplas raízes possíveis, o stylesheet escolhe a primeira raíz que aparece no schema.

Depois de conhecermos a raíz, a travessia do schema é feita saltando de referências em referências e não linearmente como é costume. Isto acontece porque o *schema* é um tipo de XML com características muito particulares.

A tarefa seguinte é criar o formulário, mas agora, preenchido com os valores lidos do ficheiro *.xfx*. Assim, temos de atravessar o schema em paralelo com o *.xfx*.

3.1.1 Sub-conjunto do schema considerado

Devido à extensão e complexidade de um *schema*, foi necessário definir um sub-conjunto do schema que o X*f*X iria suportar. Assim, as seguintes limitações foram consideradas durante a realização do X*f*X:

- O XML Schema possibilita referenciar outros ficheiros. Para tal, existem as etiquetas *include* e *import*. O X*f*X não suporta qualquer tipo de referenciamento externo. Assim se o *schema* estiver repartido por vários ficheiros, o ideal será compila-los todos num só e apagar as etiquetas *include* e *import*.

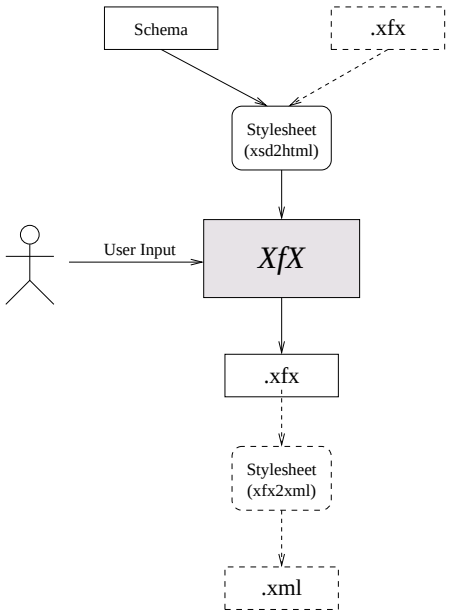

Figura 3.1: Arquitectura do X*f*X.

- O XML Schema prevê a possibilidade de redefinir determinados elementos. Para tal usa-se a etiqueta *redefine*. O *XfX* não inclui esta funcionalidade, assim a etiqueta *redefine* é simplesmente ignorada.
- Um elemento pode ser bastante complexo. Podem existir elementos que incluem outros elementos, atributos, etc. As etiquetas que prevêm esta funcionalidade são *simpleContent* e *complexContent*. Devido ao grau de complexidade inerente a estes elementos, o *XfX* não permite o uso deste tipo de etiquetas no schema.
- Os elementos do tipo simpleType são bastante simples. Porém a sua definição é bastante extensa. O *XfX* apenas resolve o caso *restriction*/*enumeration*, sendo este, provavelmente, o caso mais complexo. Assim, todos os restantes tipos de restrições que podem ser impostos no elemento simpleType não serão tratados.

3.1.2 Porquê um ficheiro de projecto

Aquando do desenvolvimento do *XfX* tivemos a necessidade de recorrer a utilização de um ficheiro de projecto. Este ficheiro é de extrema utilidade, quando queremos reescrever um ficheiro XML. Nestes casos, o formulário é gerado normalmente, mas os campos são preenchidos baseados com os valores encontrados no ficheiro *.xfx*. Não nos podemos guiar por um XML porque as etiquetas podem

aparecer por outra ordem que não a dos campos do formulário, ou seja, poderão existir diferentes derivações da gramática (schema) que originem o mesmo XML.

Neste caso, duas soluções seriam possíveis. Utilizar um ficheiro intermédio onde guardamos o XML com alguma informação sobre o schema, ou recorrer a um parser de XML que nos indicasse todas as derivações utilizadas para a validação do XML Os parsers disponibilizados até a data não permitem fazer isso.

3.2 Geração dinâmica de HTML

3.2.1 Tabelas como representantes atómicos de elementos

Um formulário HTML é composto por diversos elementos visuais, tabelas, input boxes, imagens, etc. Todo o elemento de um documento XML é representado no formulário por uma tabela, que pode ou não ser visível.

Determinadas tabelas são mantidas invisíveis, embora existam, para não confundir o utilizador com excesso de informação.

Cada tabela representa um determinado elemento do schema (sequence, all, complexType, choice, elementos simples, etc.).

Esta abstracção é fundamental na medida em que é preciso transportar alguma informação acerca de cada elemento do documento XML final.

Essas informações são tão variadas como:

- Os dados do próprio elemento
- MaxOccour e minOccour
- Xpath
- ChoiceItem
- elementType

Essas informações são transportadas como valores de atributos do elemento table do HTML. Assim, uma table é o elemento HTML mínimo suficiente para representar um determinado elemento XML. Isto acontece, pois é necessário agrupar vários elementos do documento HTML como sendo um único e indivisível elemento do XML. Como exemplo de alguns desses elementos podemos apontar o agrupamento de uma input box, que contém os dados do elemento e o atributo que indica qual o tipo desses dados.

3.2.2 HTML Dinâmico

Um documento HTML é, por natureza, estático, ou seja, os seus componentes visuais não são facilmente alterados durante a consulta desse mesmo documento.

Esta limitação teria que ser obrigatoriamente ultrapassada uma vez que o nosso formulário HTML, obtido através da aplicação de uma stylesheet ao schema argumento, deveria conter porções do seu conteúdo capazes de ser modificadas.

Isto acontece, principlamente, devido ao facto de alguns elementos poderem ocorrer mais do que uma vez no documento XML resultante. Este comportamento é assegurado no schema dum determinado XML através do uso dos atributos *maxOccours* e *minOccours*.

Assim, um elemento definido no schema que possua os atributos *maxOccours* e *minOccours* diferentes de um (1 é o seu valor por omissão), poderão ser replicados ou eliminados do formulário HTML e por conseguinte, do documento XML resultante.

Ex.:

```
<xs:element name="telefone" type="xs:string" maxOccurs="unbounded"/>
```

Neste pequeno exemplo, podemos constatar que o elemento telefone pode ocorrer um número indeterminado de vezes no documento XML, portanto o formulário HTML que o representa deverá permitir a introdução de vários números de telefone. A solução para este problema passa por permitir ao utilizador a replicação do elemento telefone de modo a que este possa introduzir os diversos números de telefone que deseja.

3.2.3 DOM (Document Object Model)

Esta alteração dinâmica dos conteúdos do documento HTML apenas são possíveis devido à capacidade que o Javascript possui de manipular o DOM de documentos XML, documentos estes cujo HTML é apenas um exemplo.

Assim, todo e qualquer documento HTML que vemos no nosso ecrã foi previamente processado por um browser e organizado numa estrutura de dados em forma de árvore à qual chamamos DOM.

A manipulação do DOM fica a cargo de um conjunto bem definido de métodos e objectos.

(Para mais informações acerca da API que permite manipular o DOM, por favor consultar DOM Definition on W3C `http://www.w3.org/TR/DOM-Level-2-Core/core.html`)

Para além de alterar a estrutura do documento HTML em runtime, o objecto DOM (Microsoft.XMLDOM), disponibilizado pelo sistema operativo, permite a manipulação de documentos XML. O DOM é nada mais nada menos que o resultado da aplicação de um parser ao documento XML, resultando assim num, por vezes chamado, grove.

Este objecto permite a aplicação de transformações aos documentos XML. Estas transformações são definidas em stylesheets, que são uma peça fundamental da realização deste projecto.

3.2.4 XPath como representação plana dum XML

Um programa realizado em Javascript ressente da falta de estruturas de dados capazes de suportar informação complexa para além dos simples tipos escalares. Sentimos a necessidade de reconhecer o estado do formulário HTML e qual a sua relação com o documento XML representado.

Esta relação deveria ser o mais simples e directa possível, pelo que optamos por representar a estrutura do XML (tipicamente em árvore) numa estrutura de dados plana (flat) que a representasse completamente sem perda de informação.

Assim, os elementos HTML que representam determinado elemento no XML final serão marcados com um atributo *xpath*, que representa a localização desse mesmo objecto na árvore do XML.

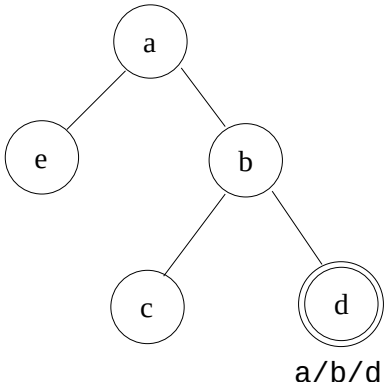

Figura 3.2: Exemplo de um *xpath*.

3.2.5 Replicação de elementos

A replicação dos elementos foi um dos problemas mais complexos de resolver, não pela complexidade visual que representam, mas sim pela complexidade estrutural da sua representação.

Replicar um determinado elemento visual (uma table com todos os seus elementos filhos, por exemplo) é bastante simples. Consegue-se através da aplicação de dois ou três métodos disponibilizados pelo DOM. O problema encontra-se em manter a coerência de nomes dos elementos replicados. Uma simples cópia não é suficiente, uma vez que iríamos ter mais do que um elemento com o mesmo xpath.

O problema prende-se com o facto de ser impossível determinar a que nível dois elementos são irmãos. Por exemplo, se considerarmos a seguinte árvore representativa de um documento XML, podemos facilmente constar que os dois elementos folha serão identificados pelo mesmo *xpath,* ou seja, **/a/ b/c**.

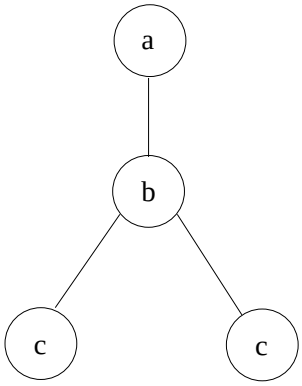

Figura 3.3: Árvore XML cujos nós folhas possuem o mesmo *xpath* (/a/b/c).

Se tentarmos reconstruir a árvore inicial baseando-nos apenas na informação de *xpath* obtida, rapidamente constataremos que poderão existir dúvidas durante essa reconstrução. O mesmo *xpath* poderá originar duas árvores diferentes.

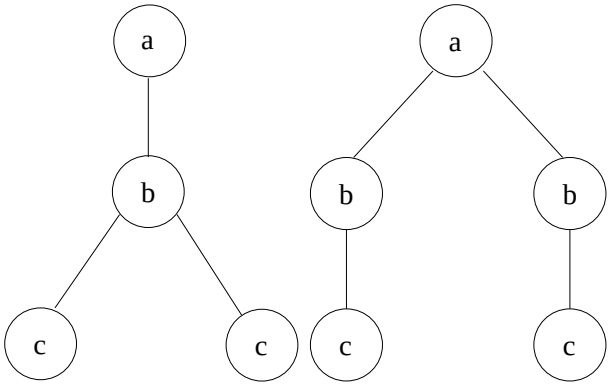

Figura 3.4: O mesmo *xpath* (/a/b/c) pode dar origem a duas árvores diferentes.

A solução passa pela introdução de informação contextual em cada nível de um *xpath.* Através da introdução de índices em cada um dos níveis da estrutura XML, deixam de existir dúvidas sobre quem é que é irmão de quem.

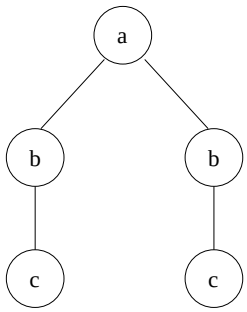

Figura 3.5: Árvore XML que representa os nós /a[1]/b[1]/c[1] e /a[1]/b[2]/c[1].

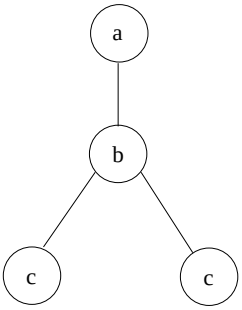

Figura 3.6: Árvore XML que representa os nós /a[1]/b[1]/c[1] e /a[1]/b[1]/c[2].

A introdução de informação contextual, baseada em índices, eliminou a incerteza existente durante construção das árvores XML.

3.2.6 Validação dos tipos de dados

O Javascript relevou-se fundamental noutra tarefa não menos importante - a validação dos dados.

O schema introduz diversos tipos de dados para além do PCDATA e CDATA que acompanhavam as especificações de gramáticas dos DTD.

Para a além da boa formação do documento, é fundamental que este seja, também, um documento válido, ou seja, que todos os dados respeitem os tipos definidos no schema e as restrições à sua estrutura sejam respeitadas.

A validação dos dados é conseguida usando Javascript e expressões regulares.

Um conjunto significativo de funções de validação foi criado de modo a garantir a conformidade dos dados com o seu tipo. Eis alguns exemplos:

```
function isValidBoolean(data, pattern, whiteSpace) {

  return /^true|false|1|0$/.test(data);
}

function isValidDecimal(data, totalDigits, fractionDigits, pattern, whiteSpace, enumeration, maxInclusive,
maxExclusive,
```

```
minInclusive, minExclusive) {

  return /^(\+|-)?\d+\.\d+$/.test(data);

}


function isValidDuration(data, pattern, enumeration, whiteSpace, maxInclusive, maxExclusive, minInclusive,
minExclusive){

  if(/^-?P$/.test(data)) return false;

  else return /^(-?P(\d+Y)?(\d+M)?(\d+D)?(T(\d+H)?(\d+M)?(\d+(\.\d+)?S)?)?)$/.test(data);
}
```

É de salientar que nesta versão do XfX apenas é efectuada a validação dos tipos em termos da sua sintaxe e não da sua semântica. É possível na especificação de um schema impor restrições aos tipos de dados tomados como base. Por exemplo, se estivermos a documentar a temperatura de uma sala de refrigeração industrial, podemos definir um novo tipo de dados tendo como base o tipo *integer* e que permita apenas valores de temperatura inteiros entre -273 e 0.

Este tipo de validação (às restrições) não é efectuada, no entanto o tipo base será sempre verificado aquando da validação.

Se algum dos dados introduzidos não estiver de acordo com o tipo definido no schema, o utilizador será avisado. Um documento XML só poderá ser gerado se todos os tipos de dados forem respeitados.

Capítulo 4

Conclusões

4.1 Crítica ao Javascript

Embora uma peça fundamental na realização deste projecto, o Javascript apresenta algumas limitações que nos levaram algum tempo a sobrepor. O facto de ser uma linguagem extremamente simples pode facilitar a vida a muitos programados inexperientes, no entanto, impossibilita a realização de alguns problemas sem termos que nos socorrer a truques e esquemas elaborados que podiam ser facilmente solúveis caso o Javascript fosse um pouco mais poderoso.

O Javascript disponibiliza alguns objectos verdadeiramente úteis, como é o caso do **DOM**, **RegExp** (para aplicação de expressões regulares), **Math**, etc., no entanto a criação de novos objectos é demasiado rudimentar.

A noção de classes em Javascript é conseguida através da adaptação de uma simples função ao paradigma de objectos, através da referência a funções auxiliares que são elegidas à qualidade de métodos.

4.2 Crítica ao XSL

A linguagem por excelência para transformar XML é a linguagem XSL. É bastante útil para fazer formatações simples da informação. No caso de um programa com a complexidade do XfX a linguagem oferece muitos entraves, ao desenvolvimento.

Uma linguagem deve ser uma extensão do nosso cérebro, uma representação do nosso pensamento, no entanto possui alguns problemas estruturais básicos. Devido à sua simplicidade, os programas facilmente se tornam extensos. O XSL desaconselha o uso de funções, considerando que estas são desnecessárias. No

entanto, consideramos que é uma óptima forma de resumir e organizar código que de outra forma se tornaria muito extenso[1].

O XSL está preparado para tratar ficheiros XML convencionais de informação anotada. No nosso caso, o XML que pretendíamos tratar era um *schema* genérico, pelo que dificultou a resolução do problema em duas frentes distintas.

Primeiro, os *schemas* têm uma lógica organizacional distinta, sendo necessário alterar a ordem normal da recursividade do processador de XSL. O XSL não prevê estes casos. A solução não é a mais intuitiva.

A segunda questão diz respeito ao facto de tentarmos tratar um *schema* genérico. O XSL é muito útil quando temos XML com estruturas fixas. No caso do *schema*, é indispensável prever uma grande variedade de casos, onde muitas vezes, mais do que um *schema* representam um mesmo XML.

Os parsers existentes encontram-se em fase embrionária. Entre outros problemas temos o facto das mensagens de erro serem praticamente inexistentes, o que dificulta e desencoraja a sua utilização.

4.3 O estado da aplicação

A aplicação encontra-se operacional. Após testada com exemplos de diversos níveis de complexidade a aplicação revelou-se robusta e estável.

Os seguintes axiomas foram alcançados:

- **Transparência**, i.e., não são necessários conhecimentos mais do que superficiais sobre a tecnologia XML para se conseguir gerar/editar um documento XML.
- **Portabilidade**, i.e., a aplicação funciona em qualquer browser numa qualquer plataforma, desde que suporte as tecnologias utilizadas, nomeadamente, XSL, Javascript e DOM. Esta aplicação foi desenvolvida e testada sobre o Microsoft Internet Explorer 6.0.
- ***User-friendly***, i.e., utiliza formulários em HTML com um aspecto familiar a qualquer utilizador da *World Wide Web*.

[1] No início do nosso curso aprendemos a lei *"divide e conquista"*. Será que já não tem validade?

Capítulo 5

Sugestões para continuação

As seguintes sugestões poderão ser tomadas em futuros desenvolvimentos:

- Validação mais exacta dos tipos de dados
- Adequar a aplicação para operar do lado do servidor
- Melhorar o formato *.xfx*
- Adicionar outros modelos de interface gráfica para permitir uma melhor navegação na informação, i.e., *modo wizard,* inclusão de índices remissivos para acesso directo , etc.
- Alargar o sub-conjunto do *schema,* eventualmente para o suporte total

Bibliografia

[HP96] HoneyCutt, Pike, et al. "Using the Internet, Third Edition". QUE, New York, 1996

[BAA00] Braakse, Mark et al. "Professional ASP XML". Wrox, Birmingham (U.K.), 2000

[SM01] Stephen Mohr et al. "Professional XML Schemas". Wrox, Birmingham (U.K.), 2001

[JCR00] Ramalho, José. "Anotação Estrutural de Documentos e sua Semântica". Braga, 2000

[NB00] Bradley, Neil. "XSL Companion". Addison-Wesley, New York, 2000

[W3CXSL] W3C Recommendation, XSL Transformations (XSLT) Version 1.0. 16 November 1999. http://www.w3.org/TR/1999/REC-xslt-19991116/

[W3CSchema0] W3C Recommendation, XML Schema Part 0: Primer. 2 May 2001. http://www.w3.org/TR/2001/REC-xmlschema-0-20010502/

[W3CSchema1] W3C Recommendation, XML Schema Part 1: Structures. 2 May 2001. http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/

[W3CSchema2] W3C Recommendation , XML Schema Part 2: Datatypes. 2 May 2001. http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

[W3CPATH] W3C Recommendation XML Path Language (XPath) Version 1.0. 16 November 1999. http://www.w3.org/TR/1999/REC-xpath-19991116