Qualidade na Publicação Electrónica: como controlá-la?

José CarlosRamalho
PedroRangel Henriques
Universidade do Minho

Abstract

Hoje em dia assiste-se a uma utilização desenfreada de suportes digitais para uma publicação electrónica selvagem e ditada pelas regras de competição do mercado.

Assim, é fácil encontrar CDROMs ou páginas WWW contendo vários erros, que podem ser simples erros ortográficos ou erros semânticos que podem ter consequências graves.

De facto, se nos debruçarmos um pouco sobre o ciclo de vida da publicação electrónica não encontramos nenhum processo de controlo de qualidade. Este, se existe, é composto por ciclos de revisão que são os primeiros a ser eliminados quando surgem restrições temporais no projecto.

A Qualidade em Publicações Electrónicas pode ser vista sob vários pontos de

vista, desde o aspecto visual da interface, o linguístico e literário, à correcção da informação (significado, semântica), isto para além da qualidade inerente ao próprio processo da publicação em si. Neste trabalho vamos focar a nossa atenção neste aspecto, a correcção da informação, discutindo como garanti-la.

Há já alguns anos que a comunidade internacional relacionada com a Publicação Electrónica se vem preocupando com a qualidade e optimização dos processos editoriais. O primeiro problema que enfrentaram foi o da proliferação de formatos subjacentes às ferramentas de software utilizadas e inerente falta de compatibilidade entre plataformas. Das reuniões científicas realizadas para discutir soluções para este e outros problemas menores surgiu um standard internacional para a Publicação Electrónica, o SGML ("Standard Generalized Markup Language").

O SGML veio resolver aquele problema mas veio também dar uma grande contribuição para a qualidade da produção documental - a validação estrutural automática.

No entanto, o SGML veio resolver apenas uma parte do problema. Veio permitir a edição estrutural assistida de documentos e provocou a separação completa entre o conteúdo de um documento e a sua formatação. Hoje existem no mercado muitos ambientes de edição que assistem os seus utilizadores na criação de documentos bem formados e estruturalmente correctos. Mas há uma falha neste sistema, não há possibilidade de se controlar o conteúdo do documento. O autor tem o controle total da informação que está a introduzir. O que nós gostaríamos que o sistema permitisse é a associação de restrições semânticas aos elementos estruturais de um documento. Deste modo, muitos erros poderiam ser evitados, alguns ciclos de revisão poderiam ser anulados e por conseguinte a qualidade aumentaria.

Neste trabalho discutiremos a utilização do SGML como meio de controlar a qualidade na produção documental e, proporemos uma extensão que nos permitirá exprimir restrições sobre o conteúdo dos documentos de modo a possibilitar alguma validação semântica automática mostrando, deste modo, aquele que seria um ambiente ideal com controlo de qualidade para a produção documental.

Por fim propomos um novo ciclo de vida para a Publicação Electrónica e apresentamos um protótipo da sua implementação.

Introdução

Em trabalhos anteriores [RH98] propusemos o processamento algébrico como uma alternativa à edição electrónica existente, com o objectivo de melhorar a qualidade e optimizar todo o processo de edição/publicação. Estas ideias foram mais exploradas num outro trabalho posterior [RAH96].

Entretanto foi-se comparando o que se andava a fazer com aquilo que a comunidade internacional andava a fazer, nomeadamente os comités ISO para a publicação electrónica. Foi fácil constatar que as estas ideias tinham eco nesses comités e que o seu trabalho tinha uma grande área de intersecção com este.

A ideia original era a de utilizar tipos de dados abstractos para exprimir e definir estruturas de documentos através das quais se faria todo o processamento documental.

Com o mesmo fim já existia o SGML [Goldfarb90], um standard ISO para a especificação estrutural de documentos. Existem já no mercado uma série de editores que

a partir daquelas especificações garantem uma validação sintáctica automática do documento aquando da sua edição.

Neste contexto, achou-se que o melhor seria aproveitarmos as nossas ideias para se atacar um problema que até ao momento não tinha sido devidamente tratado e que representa uma falha nos sistemas baseados em SGML: a validação semântica.

Neste artigo, não se irá discutir a definição de funções para o processamento documental, ir-se-á sim colocar todo o ênfase numa metodologia simples e prática que permitirá tratar a validação semântica, sempre com o objectivo final de aumentar a qualidade do produto final - os documentos (neste caso, do ponto de vista da correcção da informação).

A ideia base desta proposta é a fusão do modelo normal de processamento documental baseado em SGML com a abordagem algébrica. Desta maneira, mantem-se toda a funcionalidade subjacente aos sistemas baseados em SGML e vai-se expandi-la com a capacidade de realizar alguma validação semântica durante a edição dos documentos. Esta adição ao modelo visa, como já foi referido, a detecção de informação incorrecta.

Um pouco à semelhança de trabalhos anteriores, mas também devido à nossa formação, utilizou-se o sistema CAMILA (uma linguagem de especificação e um ambiente funcional de prototipagem) [ABNO97] desenvolvido na Universidade do Minho pelo grupo de Fundamentos da Computação, para a implementação do protótipo de validação semântica.

A estrutura deste artigo reflecte um pouco os passos seguidos na realização do trabalho que aqui se apresenta. Assim, na próxima secção faremos um resumo do que é o

SGML e dos conceitos que lhe são inerentes. De seguida, discutiremos as qualidades e as limitações do modelo existente para a seguir propormos a extensão que irá contornar aquelas limitações; basearemos esta discussão nalguns "case studies" que temos em mãos no contexto do projecto GEiRA. O resto do artigo relatará o caminho seguido na implementação do protótipo. Fecharemos o artigo tecendo algumas considerações quanto aos resultados conseguidos e ao trabalho que estamos a desenvolver e que no futuro permitirá que o protótipo aqui apresentado evolua para algo mais sério.

Publicação Electrónica baseada em SGML

Nas últimas duas décadas, a Publicação Electrónica sofreu uma enorme evolução. O evoluir da informática e das tecnologias a ela associadas tem levado a uma substituição gradual mas efectiva do papel pelo suporte digital, magnético ou óptico (diskette, disco de computador ou CDROM).

Nos últimos anos, a explosão da Internet, e a sua cada vez mais fácil acessibilidade, vieram acelerar e aumentar ainda mais a produção documental em suporte digital, consolidando novos tipos e arquitecturas de informação: o hipertexto e a hypermedia [Hytime].

Neste contexto, um documento não é apenas um registo linear de informação, mas sim algo estruturado. E foi a preocupação de tornar visível essa estrutura que deu origem às linguagens de anotação. Daqui em diante, designaremos por linguagem de anotação o conjunto de etiquetas que se colocam ao longo de um texto de modo a marcar a sua estrutura. Como exemplo de uma linguagem de anotação conhecida por todos, temos o HTML, a linguagem para anotar as páginas WWW.

Exemplo de um texto anotado:

<TITULO>1760</TITULO><PARAG>Em 14 de Junho deste anno foi mandado sahir immediatamente da Corte o cardeal <NOME>Accioalolli</NOME> e dentro em 4 dias do Reino, para o que se lhe deu decente transporte e acompanhamento. Era Nuncio de sua Santidade o Papa Clemente 13.</PARAG></PAG>

<PAG><PARAG>Aqui principiou a rotura com a Sé Appostolica. Fes-se publico o motivo pela informação que se mandou a <NOME>Francisco de Almada</NOME> residente na Corte de Roma, dando-se por fundamento maior o não pôr luminarias por ocasião do cazamento que se fes no dia seis de Junho do Senhor <NOME>Infante D. Pedro</NOME> com sua sobrinha a Excelentissima Senhora D. Maria, princesa do Brasil, o que o dito Cardeal não fizera, porque lho não fizerão a saber por carta de officio, quando os mais embaixadores sem isso o fizerão. O caso he que querião afasta-lo da Corte e tambem elle não seguia a conduta della por aquelles tempos.</PARAG> ...

Como se pode ver no exemplo, identificam-se claramente no texto as etiquetas *TITULO, PARAG, PAG e NOME.* Estas etiquetas marcam os limites de certos elementos do documento dando desta maneira estrutura a este.

A anotação dos documentos, longe se ser pacífica, veio introduzir várias alterações no ciclo de vida editorial:

- se a estrutura estiver definida à partida é possível validar se o utilizador que edita o texto está a obedecer à referida estrutura.
- um documento pode responder de diferentes maneiras a diferentes processadores.
- as "queries" numa procura de informação podem assumir uma forma mais inteligente, dirigindo-se para a estrutura.

No meio desta caótica evolução era necessário um standard que impusesse um pouco de ordem na proliferação de formatos proprietários. Esse standard surgiu em 1986 com o nome de SGML - "Standard Generalized Markup Language". Como o próprio nome sugere é uma meta-linguagem que se pode utilizar para a definição de linguagens de anotação.

Em SGML, um documento é visto como uma estrutura lógica que contém um elemento, a raiz de uma árvore de elementos que compõem o conteúdo do documento. Por exemplo, um livro pode conter elementos do tipo capítulo que por sua vez podem conter elementos do tipo parágrafo e/ou imagem.

Os elementos são distinguidos uns dos outros por informação extra (etiquetas) que é adicionada ao conteúdo do documento. Assim, um documento contém dois tipos de informação: dados e etiquetas.

Um documento SGML é composto por três partes:

a declaração SGML

onde são definidos os caracteres da linguagem, os separadores das etiquetas, tamanhos máximos para as etiquetas, ...

o DTD - "Document Type Definition"

é a parte mais importante, onde se define a estrutura do documento, quais os elementos estruturais e qual a sua hierarquia, bem como os atributos de cada um.

a instância do documento

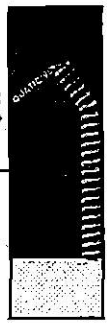
composto pelo texto marcado com as etiquetas.

A primeira parte, a declaração SGML, é criada normalmente no início e serve para todos os documentos (as particularidades de um sistema manter-se-ão). Sempre que não estiver presente é assumida uma por defeito.

O DTD é a parte mais importante do documento. É onde reside a definição da estrutura do documento. Um editor para dar apoio na edição do documento precisa de conhecer esta estrutura. Da mesma maneira, um processador para processar um documento também precisa de conhecer a sua estrutura. Por isso, o DTD surge sempre à cabeça do conteúdo do documento propriamente dito. O DTD é normalmente criado por um analista documental (esta fase é em tudo equivalente à análise que se faz quando se está a conceber uma Base de Dados ou um programa).

A seguir podemos ver um pequeno documento SGML composto pelas duas últimas partes (assumiu-se a declaração por defeito). Trata-se de uma simplificação de uma ficha relativa a um arqueosítio do Noroeste português:

```
<!DOCTYPE  ARQSITS [
<!ELEMENT ARQSITS - - (ARQELEM+)>
<!ELEMENT ARQELEM - - (IDENTI, DESCRI?, LUGAR, FREGUE, CONCEL,
CODADM?,                              LATITU?, LONGIT?, ALTITU?, QUADRO?,)+ >
<!ATTLIST   ARQELEM assunto CDATA #IMPLIED >
<!ELEMENT  IDENTI        - - ( (LIGA | #PCDATA)+ )>
<!ELEMENT  CRONO         - - ( #PCDATA )>
<!ELEMENT  DESCRI        - - ( (LIGA | #PCDATA)+ )>
<!ELEMENT  LUGAR   - - ( (LIGA | #PCDATA)+ )>
<!ELEMENT  (FREGUE, CONCEL, CODADM, LATITU, LONGIT, ALTITU,
QUADRO)                  - - ( (LIGA | #PCDATA)+ )>
```

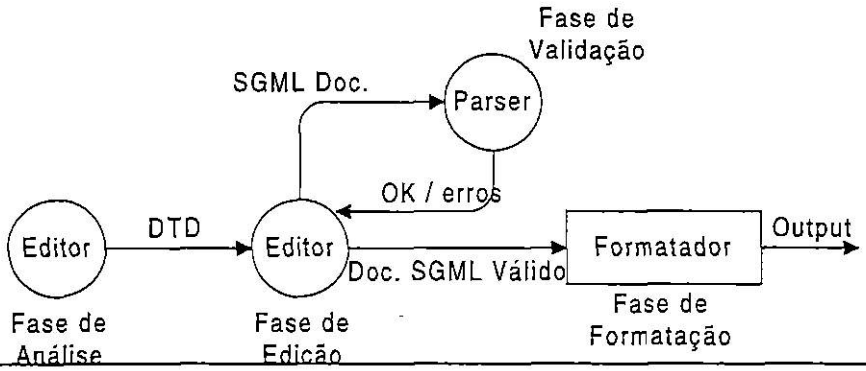


]>
<ARQSITS><ARQELEM assunto="arqueossitio">
<IDENTI> Castro do Caires </IDENTI>
<DESCRI> <LIGA termo="povoado fortificado"> Povoado fortificado</LIGA>
</DESCRI>
<LUGAR> Grovos </LUGAR>
<FREGUE> Caires </FREGUE>
<CONCEL> Amares </CONCEL>
<CODADM> 030105 </CODADM>
<LATITU> 519,9 </LATITU>
<QUADRO> O cabeço onde assenta o <LIGA termo="castro"> castro</LIGA> corresponde a um esporão, de vertentes abruptas, situado praticamente na base da encosta Sul do monte de S. Pedro de Fins. A Norte, um profundo talvegue separa o monte de um outro, mais pequeno, onde se encontram também vestígios de ocupação. </QUADRO>
</ARQELEM>
</ARQSITS>

O Documento apresentado tem duas partes distintas. A primeira iniciada em "*<!DOCTYPE ...*" e terminada em "*]>*" é o DTD. A segunda, que compõe o resto do documento é a instância o documento propriamente dito.

Neste DTD simples podemos observar dois tipos de declarações. As declarações de elementos (*ELEMENT*) que correspondem às partes estruturais identificadas no documento, e as declarações de atributos (*ATTLIST*) que servem para declarar propriedades associadas a cada um dos elementos.

Um sistema de edição baseado em SGML não se limita ao que até aqui foi apresentado, é um pouco mais complexo. A seguir apresentamos um esquema mostrando as várias peças que compõem o sistema e as relações entre elas.



Edição baseada em SGML: figura 1

Como se pode observar, distinguem-se quatro fases na tarefa de criar um documento: análise, edição, validação e formatação.

A fase de análise corresponde, como o próprio nome indica, ao estudo do tipo de documento em causa e à especificação da sua estrutura (que é o resultado desta fase na forma de um DTD).

A seguir temos um ciclo de duas fases, edição e validação. O utilizador vai escrevendo o seu documento e de vez em quando pede ao editor que valide o texto introduzido. O editor, uma vez que conhece o DTD pode proceder à validação estrutural do texto e dar como feedback ao utilizador um OK ou uma lista de erros encontrados.

O documento, depois de editado e validado, passa para um formatador onde irá ser feita a sua formatação para um determinado formato final. Esta formatação é também ela governada por uma especificação, a chamada especificação de estilo que fica

associada a um determinado tipo de documento (DTD). Apesar de existir uma linguagem standard para escrever esta especificação (DSSSL), este ainda não foi adoptado pela indústria. Assim cada editor possui a sua linguagem para especificar o estilo, normalmente associada a um editor gráfico.

Utilizando esta tecnologia já é possível ter algum controlo de qualidade. Torna-se possível a verificação dentro de uma empresa de que todas as pessoas escrevem cartas com a mesma estrutura, ou memos, ou contractos, ...

Mas o aumento na qualidade da informação não se fica por aqui. Se compararmos este método com o tradicionalmente utilizado verificamos que agora temos documentos com estrutura explícita, marcados, e que podem ser manipulados como se de uma base de dados tradicional se tratassem. Este ponto é por demais importante uma vez que os estudos mais recentes indicam que 90% da informação produzida numa empresa está na forma de documentos e apenas 10% se encontra numa forma possível de ser armazenada numa base de dados tradicional.

SGML e a Validação Semântica

Como referimos na secção anterior, a edição baseada em SGML trouxe por si só um aumento na qualidade da produção documental – tornou possível a validação estrutural/sintáctica dos documentos no momento da sua edição.

No entanto, há situações em que este controlo não é suficiente. Documentos estruturalmente correctos podem conter erros que podem deitar por terra a qualidade do mesmo. Estamos a falar de pequenos (e outros menos pequenos) erros semânticos introduzidos por um utilizador menos atento ou menos especializado.

Estas situações surgem com mais frequência no contexto de projectos editoriais

com alguma dimensão. Nestes, os especialistas são envolvidos na fase inicial de preparação dos textos mas, a edição final fica à responsabilidade de técnicos que muitas vezes desconhecem o assunto sobre o qual estão a editar texto. Assim, é muito fácil que haja introdução de informação semanticamente incorrecta.

Como exemplo desta situação podemos referir o caso do CDROM sobre História de Portugal que teve que ser retirado do mercado, pois passados dois meses do seu lançamento verificou-se que continha graves erros semânticos como: errada atribuição de reis a reinados, eventos com datas trocadas, ...

Grande parte desta situação poderia ter sido evitada com a metodologia que aqui propomos.

Resumindo, o SGML permite controlar a estrutura de um documento mas não o seu conteúdo. Na edição baseada em SGML o utilizador/autor tem o controlo total sobre a informação que está a introduzir. No sistema pretendido tal não irá suceder. Nalgumas situações o utilizador terá que se restringir a um conjunto de valores pré-definidos ou que obedeçam a certas condições.

Apresenta-se a seguir um exemplo que ilustra as ideias avançadas:

Exemplo 1: A necessidade de uma validação semântica

Um editor está a preparar um livro sobre literatura portuguesa (usando SGML). Um dos capítulos incidirá sobre autores. A sua estrutura poderia ser definida da seguinte forma:

```
...
<!ELEMENT list-autor - - (autor+)>
<!ELEMENT autor      - - (nome, data-nasc, data-ob,...)>
...
```

Este fragmento do DTD especifica que uma parte do capítulo deverá ser constituída por uma lista de autores, cada um caracterizado por um nome, uma data de nascimento, uma data de óbito, e outros elementos.
A validação estrutural destes elementos irá ser verificada pelo "parser" SGML associado ao editor. No entanto, para além da correcção sintáctica, é também importante garantir um invariante: *o elemento data de óbito de cada autor deverá conter um valor sempre maior que o contido no elemento data de nascimento.*

Este pequeno exemplo dá uma ideia dos problemas que surgem nesta área. Neste caso, o exemplo utilizado foi um livro de literatura mas poderia muito bem ter sido o manual de operações de um avião ou de uma central nuclear e nestes o padrão de qualidade só pode ser um: o máximo.

Nas próximas secções apresenta-se uma extensão ao modelo tradicional do SGML que permitirá resolver parte deste problema.

As Restrições

De modo a preservar certas características semânticas dos documentos é necessário associar restrições aos elementos de um DTD. No modelo que agora se propõe, estas restrições deverão ser especificadas na fase de análise pelo analista. Um processo em tudo semelhante ao da análise de sistemas de informação onde, depois de se obter as entidades e as relações, se vão especificar invariantes sobre os dados.

Sempre que propõe alterações a modelos existentes deve ter-se o cuidado de observar a compatibilidade com o que já existe, principalmente se o modelo é um standard internacional.

Assim, tendo esta preocupação em mente pode-se avançar com duas maneiras de adicionar restrições a um DTD:

- *Secções de comentário especiais* – onde as restrições seriam escritas; sendo comentários o processamento SGML normal irá ignorá-las e actuará como sempre o fez; mais tarde um processador especial poderá ler estas secções e processá-las.

Exemplo 2: DTD com restrições

```
<!DOCTYPE rei [
<!ELEMENT rei -- (nome, cognome, data-nasc, data-ob,
decreto*)>
<!-INV
    inv_rei(r)= ...
->
```

- *Uma referência a um ficheiro externo* – onde seriam colocadas as restrições; esta referência seria também ela colocada num comentário.

Exemplo 3: Restrições através de referência

```
<!- INV: rei.com ->
<!DOCTYPE rei [ ...
```

Recorreu-se aos comentários e colocou-se lá dentro as restrições pretendidas para que as ferramentas SGML existentes continuem a processar estes documentos sem alterações.

O processamento extra necessário para processar estas linhas de comentário especiais irá ser desenvolvido à parte e constituirá uma nova fase do processo.

Das duas abordagens possíveis, apresentadas acima, optou-se pela segunda, uma linha de comentário no início do DTD contendo uma referência a um ficheiro onde serão colocadas as restrições. A primeira abordagem, incluir as restrições no DTD, tornaria os DTDs mais pesados e de leitura difícil (alguns DTDs já descrevem por si só estruturas bem complexas).

Até agora foi apenas discutido o mecanismo de associação das restrições ao DTD. Falta a parte mais importante, a linguagem em que serão escritas essas restrições.

Linguagem de Restrições

As restrições a adicionar a um DTD devem ser especificadas, utilizando-se para tal uma linguagem/formalismo. Aqui a questão é: Que linguagem? Que formalismo?
Para se responder a esta questão é preciso saber qual o grau de complexidade das restrições e que tipo de restrições se irá especificar.

A linguagem poderia ser criada como uma extensão à sintaxe do SGML ou como uma nova linguagem que poderia coexistir com o SGML. Qualquer uma destas soluções tem implicações na implementação como se irá ver mais à frente.

Até ao momento, nos vários projectos onde se utilizou a edição baseada em SGML, as restrições que se sentiu necessidade de especificar são bastante simples: normalmente pretende-se restringir o valor de alguns elementos atómicos a um

determinado domínio, verificar relações entre elementos ou verificar se um determinado valor existe numa dada base de dados.

Provavelmente esta simplicidade deriva do facto da validação estrutural executada pelo "parser" de SGML tornar obsoletas validações nos níveis mais altos da hierarquia estrutural de um documento.

Provavelmente a linguagem necessária para especificar estas restrições será bem simples. No entanto, pode-se distinguir duas fases completamente distintas no modelo de validação semântica que se está a tentar implementar:

- *A definição* – a parte sintáctica do modelo de restrições, as "frases" que expressam as restrições.

- *O processamento* – a parte semântica do modelo de restrições, a interpretação das "frases" que expressam as restrições.

Estas duas fases têm funções distintas e correspondem a diferentes níveis de dificuldade na sua implementação.

A fase de definição implica a criação duma linguagem nova ou a adopção duma existente.

A fase de processamento implica a criação de um motor de processamento com capacidade de processar a linguagem escolhida.

Algures no meio destas duas fases enfrentaremos a necessidade de ter informação "tipada" com toda a complexidade que lhe é inerente (na definição dos elementos num DTD nada é declarado relativamente ao tipo do seu conteúdo). Para os não especialistas na área da compilação a pergunta poderia surgir: "Necessitamos mesmo desta complexidade extra?" Não poderemos passar sem tipos?"

Apresenta-se a seguir um exemplo que ilustra esta necessidade.

Exemplo 4: Será necessário associar tipos aos elementos dum DTD?

Um dos projectos em que se está a aplicar esta metodologia é o dos arqueosítios do Noroeste português (Arqueologia). Para cada arqueosítio é redigida uma ficha, que obedece a um DTD, contendo a informação desse arqueosítio. No futuro, pretende-se ligar cada uma destas fichas a um sistema de informação geográfica (SIG). Para tal, cada ficha contém elementos com informação geográfica: latitude, longitude e altitude. Para que o SIG não contenha erros é preciso que os valores introduzidos estejam correctos – aqui as restrições de domínio podem representar um papel importante:

Doc. SGML

...

<latitude>41.32</latitude>

...

Restrição

latitude > 39 and latitude < 43

Desta maneira podia-se garantir que mesmo que o valor da latitude não fosse o mais correcto pelo menos não "cairia fora do mapa".

Neste exemplo está-se a restringir um valor a um domínio. Está-se a comparar o conteúdo do elemento *latitude* com valores numéricos, que têm um tipo que lhes é inerente (inteiro ou real). Desta maneira, o motor de processamento terá que inferir de alguma maneira que valor do elemento *latitude* é dum tipo numérico.

Exemplos como este são bem simples. O conteúdo numérico é bem conhecido e tem uma forma mais ou menos normalizada. Mas há outros, mais complexos como as datas – há mais de cem formatos diferentes para se escrever uma data (e provavelmente cada um de nós pode inventar facilmente mais um).

Este problema que agora se levanta designa-se por normalização da informação e é também familiar a quem trabalha em sistemas de informação tradicionais.

Para se avançar com o modelo de validação semântica tem-se que dar uma solução a estes dois problemas: inferência de tipos e normalização da informação.

À partida, duas soluções poderiam ser apontadas para tratar daqueles problemas:

- Implementar de raiz ferramentas complexas que tratem de cada um deles.
- Introduzir algumas alterações no DTD que solucionarão o problema da normalização e facilitarão a tarefa de inferência de tipos.

A segunda alternativa é mais simples e permite uma aproximação a algumas soluções parciais já desenvolvidas no seio de projectos internacionais como o "Text Encoding Initiative (TEI)".

A solução é bastante simples tecnicamente mas tem implicações a nível dos intervenientes no processo: adiciona-se um atributo "valor" a todos os elementos para os quais se prevê irem surgir problemas de normalização na introdução da informação; (in)forma-se todos os utilizadores que naqueles elementos podem introduzir o conteúdo na forma que pretenderem desde que preencham o atributo "valor" desse elemento com o valor que se decidiu ser o normalizado.

Pode parecer um pouco anárquica a situação de permitir ao utilizador que introduza os dados como pretender mas, há projectos em que a forma normalizada e forma escrita são necessariamente diferentes: normalmente em documentos históricos,

pretende-se que estes tenham o aspecto visual original mas por detrás tem de haver uma versão normalizada que permita ao computador processá-los. Por exemplo, nos documentos da instauração do reino português, Afonso Henriques aparece referenciado de várias formas: "Afonso", "Affonso" ou simplesmente "o Rei". Se se quiser criar um índice antroponímico para navegação nesses documentos, o processador desse índice precisa de alguma maneira de saber que aquelas três referências correspondem ao mesmo "objecto", neste caso, o rei Afonso Henriques.

Exemplo 5: Normalizando a informação ...

```
...
aconteceu no <data valor="1853.10.05"> quinto dia do mês de
Outubro do ano do Senhor 1853 </data> ...
```

Aqui a forma normalizada adoptada para as datas foi o formato ANSI. E como se pode ver o atributo "valor" tem a data no formato ANSI enquanto o conteúdo do elemento "data" tem a mesma informação num formato (provavelmente original) que daria muito trabalho ao sistema se a quisesse comparar, ordenar ou realizar outras operações sobre ela.

Relativamente à inferência de tipos adoptou-se uma solução semelhante: adicionar um atributo "tipo" a todos os elementos que irão ter restrições associadas; este atributo conterá uma "string" designativa do tipo do valor que poderá ser inserido no elemento.
O exemplo acima ficaria assim:

Exemplo 6: Inferência do tipo

```
...
aconteceu no <data valor="1853.10.05" tipo="data"> quinto dia
do mês de Outubro do ano do Senhor 1853 </data> ...
```

Para simplificação de escrita pode-se definir que sempre que o atributo "valor" não se encontrar definido o conteúdo desse elemento encontra-se na forma normalizada.

Em relação aos tipos a seguinte questão poderia ser levantada: "Será necessário 'tipar' todos os elementos da árvore estrutural de um documento?"

Tentar-se-á responder a esta questão na próxima secção.

O Modelo de Validação Semântica

A questão levantada na última secção pode ser encarada como uma questão de resposta sim ou não. Apesar de parecer inocente qualquer uma das respostas tem implicações bastante pesadas nas fases seguintes.

Uma resposta "sim", implicaria que para além de tipos atómicos tivéssemos tipos estruturados para podermos definir o tipo dos elementos intermédios da árvore do documento. Desta maneira obtinha-se também um mapeamento completo da estrutura do documento num modelo de tipos. As consequências desta abordagem, vantagens e desvantagens, seriam as seguintes:

- Um sistema de tipos mais complexo.
- Seria mais fácil associar tipos aos elementos estruturados partindo do DTD do que da instância do documento; isto implicaria a criação duma ferramenta de conversão.
- Tendo toda a estrutura do documento mapeada num modelo de tipos torna-se possível processar o documento neste novo modelo (utilizando os operadores definidos para esses tipos).

Uma resposta negativa teria as seguintes consequências:

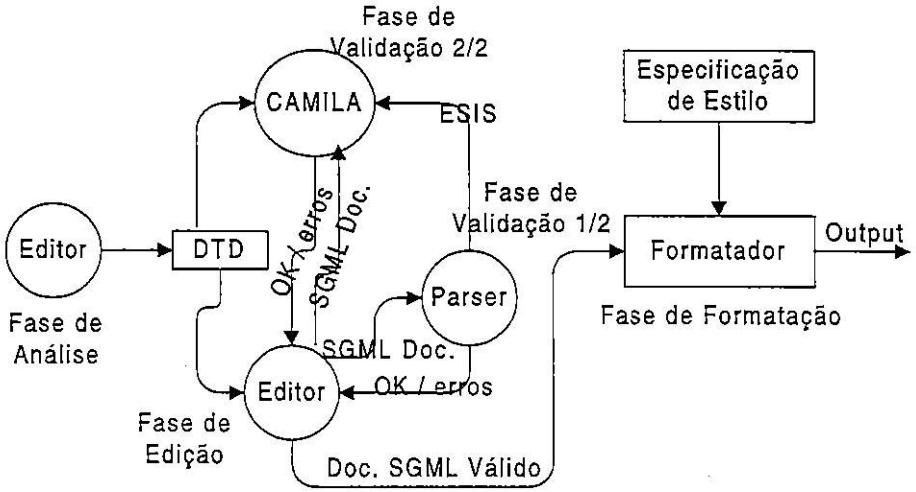
- A linguagem de restrições seria muito simples; provavelmente restrita a tipos atómicos e algumas operações de procura.
- Por conseguinte, o motor de processamento seria também simples e fácil de implementar.
- O modelo de tipos ficaria incompleto o que invalidaria o processamento do documento neste modelo.

Num trabalho anterior (RAH98), apresentou-se uma implementação que seguia a ideia da primeira hipótese ("SIM"). O sistema realizava uma conversão completa da estrutura do documento para um modelo de tipos abstractos de dados. Naquela implementação os problemas de inferência de tipos e de normalização não foram observados. Relativamente aos tipos, tinha-se criado uma ferramenta que convertia o DTD num tipo abstracto de dados.

Foi fácil adaptar aquele sistema para trabalhar com as soluções propostas para a inferência de tipos e normalização.

Apesar de aparentar ser mais complexa foi este o modelo escolhido porque além das restrições permite especificar e processar documentos a um nível muito alto de abstracção.

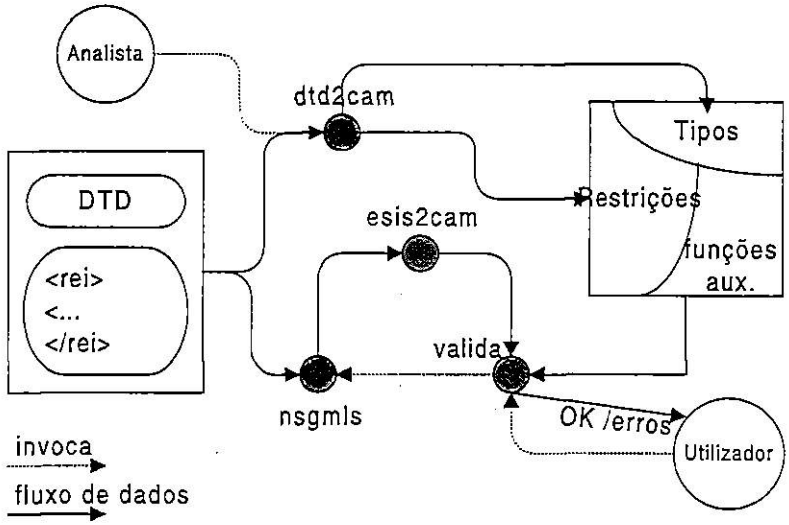
Assim o modelo para validação semântica adoptado pode ser representado pela figura seguinte.



Novo modelo de Edição com SGML: fig. 2

Como se pode ver na figura, foi adicionado um novo processo de validação ao modelo tradicional de edição com SGML. Este novo processo irá tratar de processar as restrições.

Na figura seguinte podemos ver este processo em mais detalhe.



Processo de Validação CAMILA: fig. 3

Este novo processo vai exigir alterações a dois níveis: um relacionado com as pessoas, analista e utilizadores, e outro relacionado com o sofware, como é que vão interactuar as novas rotinas com as já existentes. O primeiro nível será designado por "operacional" e o segundo por "software" e podem ser descritos da seguinte forma:

Nível operacional

No modelo apresentado há dois intervenientes: o analista e o utilizador/autor. No modelo tradicional o analista apenas tinha que desenvolver o DTD. No novo modelo além do DTD, o analista terá de desenvolver ao mesmo tempo a especificação de restrições. No fim da análise uma cópia do DTD será enviada para o Editor da fase de edição e outra juntamente com as restrições serão enviadas para uma nova ferramenta que realizará a validação adicional. Estas alterações são invisíveis para o utilizador do sistema. Ele só sentirá mudanças na

preocupação que agora terá que ter em relação à normalização de informação.

Nível de Software

Como se pode observar na figura 3, este nível é composto por várias peças de sofware. A maior e mais complexa foi designada por **CAMILA** [ABNO97] (devido ao sistema de prototipagem e linguagem em que foi desenvolvida), as outras foram baptizadas de acordo com a sua funcionalidade: **dtd2cam**, **esis2cam** e **parser** [Clark]. O processo é centrado no **CAMILA** que recebe três inputs e produz um resultado. O DTD concebido na fase de análise é enviado a **dtd2cam** que traduz os seus elementos para um tipo abstracto de dados em CAMILA, passando este ao **CAMILA**. As restrições que são escritas directamente na linguagem CAMILA vão directamente para o **CAMILA**. Do outro lado, quando o utilizador quer validar o documento que está a editar, activa a função **valida** que começa por enviar o documento ao **parser**; o output do **parser**, em formato ESIS (formato intermédio utilizado nas aplicações SGML), é enviado ao **esis2cam** que vai usá-lo para preencher a estrutura já definida em CAMILA com o conteúdo do documento. Assim, o **CAMILA** em posse do DTD (traduzido pelo **dtd2cam**), das restrições e do documento (traduzido pelos **parser** e **esis2cam**) pode desencadear a validação semântica, bastando para isso executar as restrições. Como resultado o **CAMILA** envia ao utilizador um OK ou uma lista de mensagens de erro.

Na próxima secção vai-se seguir um documento que irá atravessar o sistema.

Implementação com um exemplo

Os casos de estudo que serviram de base a este trabalho são bastante complexos e cada um tem as suas particularidades o que implica que seja difícil a sua demonstração neste artigo.

Apresenta-se um exemplo fictício (no sentido em que representa um subconjunto pequeno de um dos casos de estudo) mas com todos os ingredientes comuns a uma aplicação normal.

Neste exemplo, os documentos são listas de decretos publicados por um rei português.

Reis e Decretos (DTD)

```
<!DOCTYPE rei (
<!ELEMENT rei -- (nome, cognome, data-nasc, data-ob,
decreto*)>
<!ELEMENT decreto -- (data, corpo)>
<!ELEMENT (nome, cognome, data-nasc, data-ob, data) --
(#PCDATA)>
<!ATTLIST  data-nasc valor CDATA #REQUIRED #FIXED tipo data>
<!ATTLIST  data-ob valor CDATA #REQUIRED #FIXED tipo data>
<!ATTLIST  data valor CDATA #REQUIRED #FIXED tipo data>
<!ELEMENT corpo -- (#PCDATA)>
```

Note-se a definição dos atributos "valor" e "tipo" para valores que irão conter datas. O atributo tem já o seu valor fixado em "data", o utilizador não terá que saber nada

sobre tipos, o analista é que tem de preparar tudo a nível do DTD. O atributo "valor" deverá ser usado pelo utilizador para a introdução de valores normalizados para datas (neste caso ANSI); o processador usará o valor deste atributo em vez do conteúdo do respectivo elemento.

A seguir apresenta-se um documento escrito de acordo com este DTD: trata-se de uma lista de decretos proclamados por D. Dinis.

Os decretos de D. Dinis (dinis.sgm)

```
<rei>
<nome>D. Dinis</nome>
<cognome>o Lavrador</cognome>
<data-nasc valor="1270.09.23">23 Setembro 1270</data-nasc>
<data-ob valor="1370.09.23">23 Setembro 1370</data-ob>
<decreto>
  <data valor="1300.08.15">ao quinto dia do mês de Agosto do
ano 1300</data>
  <corpo>A partir deste dia, apenas bicicletas poderão circular na
cidade de Braga.</corpo>
</decreto>
<decreto>
  <data>1297.07.15</data>
  <corpo>O McDonald's passará a vender vinho verde em vez de
COCA-COLA.</corpo>
</decreto>
</rei>
```

Observando o documento e o DTD pode-se pensar nalgumas propriedades que deveriam ser verificadas:

- A data de cada decreto deverá estar sempre compreendida entre a data de nascimento e a data de óbito do rei que a proclamou.
- O nome do rei deverá existir na base de dados de personalidades do sistema.

Estas restrições são especificadas num ficheiro à parte em linguagem CAMILA e a ligação ao DTD é feita por uma referência colocada numa linha de comentário do mesmo:

```
<!-- Restrições: rei.cam - -
>
<!DOCTYPE rei [ ...
```

Na implementação que se seguiu as restrições são especificadas por um conjunto de regras; cada regra é um par formado por uma condição (a negação da restrição) e respectiva reacção.

As restrições do nosso exemplo são assim especificadas:

```
rei(r) =
{ if( nome_(r) notin BDpersonalidades  -> nome_(r) ++ "não existe
em BDpersonalidades"),
  if( data-nasc_(r) > data-ob_(r)          -> nome_(r) ++ "morreu antes
de nascer"),
  if( data-ob_(r) - data-nasc_(r) > 120 -> nome_(r) ++ "viveu mais de
120 anos"),
  if( !all( x <- decreto_l(r): data-
nasc (r) < data (x) /\ data (x) < data-ob (r) )
```

```
        -> nome_(r) ++ "fez um decreto fora da sua vida" )
};
```

Se se tivesse a seguinte lista de valores:

- r = ("D. Dinis", "o Lavrador", "1265.06.24", "1211.04.12", ...)
- data-nasc_ = "1265.06.24"
- data-ob_ = "1211.04.12"

a condição "if(data-nasc_(r) > data-ob_(r)" seria verdadeira o que teria como resultado a concatenação ("++") do nome do rei ("nome_(r)") com a string "morreu antes de nascer".

Ao longo desta implementação com exemplo viu-se como foi alterada a filosofia de especificação de DTDs de modo a poder acomodar as restrições, e como se extendeu o modelo de processamento de modo a que a verificação das restrições seja possível.

Conclusão

O objectivo deste trabalho foi a melhoria da qualidade e dos mecanismos de controlo a ela associados na publicação electrónica baseada em SGML. Neste contexto, apresentou-se um novo modelo de processamento documental que introduz uma nova linha de controlo: a validação semântica do conteúdo.

A ideia principal é a de que introduzindo algumas restrições semânticas associadas a alguns elementos estruturais dos documentos, pode-se minimizar a incorrecção da informação aumento desta maneira a qualidade do produto final.

Através de um exemplo apresentaram-se os passos a ser seguidos na implementação do novo modelo. Ao longo desse trajecto, identificaram-se problemas e avançaram-se algumas soluções. No fim, apresentou-se o protótipo dum sistema que inclui o novo mecanismo de controlo e o processa automaticamente.

Neste momento, o trabalho prossegue no sentido de transformar aquele protótipo numa aplicação real.

Bibliografia

[ABNO97] J. J. Almeida, L. S. Barbosa, F. L. Neves, and J. N. Oliveira; "CAMILA: Formal Software Engineering Supported by Functional Programming"; Editor: A. De Giusti, J. Diaz, and P. Pesado; Proc. II Conf. Latino Americana de Programacion Funcional (CLaPF97); La Plata, Argentina, October 1997

[Clark] James Clark; "NSGMLS: an SGML parser conforming to ISO 8879", www.jclark.com

[BA95] L. S. Barbosa and J. J. Almeida; "CAMILA: A Reference Manual", Technical Report DI-CAM-95:11:2, DI (U.Minho), 1995

[TEI] C.M. Sperberg-McQueen and Lou Burnard; "Guidelines for Electronic Text Encoding Interchange (TEI P3)"; Chicago: ACH/ACL/ALLC; 1994

[RH98] J. C. Ramalho, J. J. Almeida, and P. R. Henriques; "Algebraic Specification of Documents"; Theoretical Computer Science; Elsevier; 15 June 1998

[RAH96] J. C. Ramalho, J. J. Almeida, and P. R. Henriques; "Document Semantics: two approaches"; SGML'96: Celebrating a decade of SGML, Sheraton-Boston Hotel, Boston, USA; Nov. 1996

[Hytime] Steven DeRose and David Durand; "Hytime: Making Hpermedia Work"; Kluwer Academic Publishers, 1994

[Goldfarb90] Charles Goldfarb, "The SGML Handbook", Clarendon Press – Oxford, 1990

Tema Abrangido: "Métodos de especificação formal para a melhoria da qualidade"

Palavras Chave: Controlo de Qualidade; Publicação Electrónica; SGML; Estandardização;