Sistema de Análise de Complexidade de Documentos XML

Ricardo Andrade, Ruben Costa

FCT / UNL

Resumo — A adopção de documentos XML tem sido hoje em dia bastante utilizado, tornando-se cada vez mais como um standard para troca de informação entre vários sistemas. Os documentos XML encontram-se estruturados das formas mais diversas, o que por vezes pode tornar difícil para o ser humano interpretar o seu conteúdo. A extracção de métricas desses documentos permite avaliar a sua complexidade daí que este documento sugere uma abordagem baseada no populamento de um meta-modelo apresentando uma ferramenta capaz de obter tais resultados sobre todo o tipo de ficheiros XML.

Palavras Chave — Bases de Dados, Meta-Modelo, Métricas, OCL, USE, XML

1 INTRODUÇÃO

Os sistemas modernos de bases de dados melhoram em larga escala as capacidades dos sistemas de bases de dados tradicionais, através das suas capacidades de manipulação de qualquer tipo de dados, incluindo texto, imagens, áudio, e vídeo, e possibilitam o acesso as esses dados via linguagens SQL. Hoje em dia, os sistemas de bases de dados têm um papel fundamental na web, uma vez que podem providenciar informação para os gestores de conteúdos de páginas web. Desde a altura em que os sistemas de bases de dados começaram a trocar informação entre si, houve um interesse particular na linguagem XML [1] como formato padrão na troca de informação. Como resultado, vários sistemas de bases de dados relacionais permitem a exportação / importação de dados em formato XML. O XML está em vias de se tornar no meio de comunicação standard utilizado na web. Adicionalmente, tem-se assistido cada vez mais à tendência de armazenar dados XML em sistemas de bases de dados. Ao se adoptar esta abordagem torna-se mais fácil o acesso e a manutenção dos dados. No entanto, alguns sistemas de bases de dados comerciais são específicos no armazenamento, manutenção e facilidade no acesso a documentos XML.

Toda esta proliferação de documentos XML com grandes volumes de dados associados, leva a que muitas vezes seja difícil para o ser humano ter uma fácil interpretação de como o documento se encontra organizado, e muitas vezes saber em que parte do documento se encontra a informação que é realmente importante. Todos estes factores conjugados levam a que seja necessário a criação de métricas para documentos XML. Estas métricas permitem dar ao utilizador informação sobre: de que forma o documento se encontra estruturado, qual o grau de complexidade do documento, qual a dimensão do documento em termos de elementos que o constituem, qual o factor de homogeneidade do documento face a outros documentos conhecidos, entre outras.

O trabalho subjacente a este artigo foi desenvolvido no âmbito da disciplina de Qualidade do processo e do produto, leccionada no Mestrado de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, sob orientação do professor Fernando Brito e Abreu[2].

Este artigo, encontra-se dividido em vários capítulos:

- O presente capítulo, onde se encontra descrito o âmbito do trabalho;
- O segundo capítulo, onde se encontram descritos todos os passos inerentes à realização do respectivo trabalho;
- No terceiro capítulo, encontram-se descritas algumas referências a vários trabalhos relacionados na mesma área;
- No quarto e último capítulo, estão descritas as conclusões ao trabalho, bem como, algumas visões sobre o trabalho futuro a ser desenvolvido nesta área

2 APLICAÇÃO

2.1 USE

O *use* é um sistema utilizado para especificação de sistemas de informação, o qual se baseia num subgrupo do UML (Unified Modeling Language) [3]. Uma especificação *use* contém uma descrição textual do modelo utilizando propriedades intrínsecas dos diagramas de classes do UML (classes, associações, etc.). As expressões escritas em OCL (Object Constraint Language) [4] são utilizadas para especificar restrições a nível de integridade no modelo. Um modelo poderá ser ilustrado de forma a validar a sua especificação em função dos requisitos não formais. Os estados do sistema poderão ser criados e manipulados durante a visualização do modelo. Em cada instante as restrições em OCL são automaticamente verificadas. Toda a informação sobre os vários estados do sistema é dada sob a forma gráfica. As expressões em OCL são introduzidas e verificadas de forma a ter noção sobre a informação do sistema num dado estado [5]. A figura seguinte ilustra o uso de uma abordagem *use*.

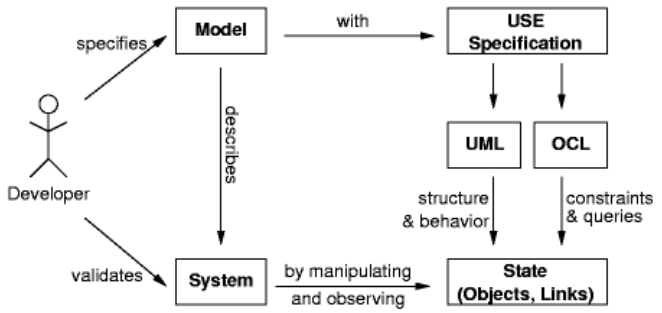

Figura 1 - Utilização da abordagem USE

O nosso sistema irá tirar partido das funcionalidades do USE para que através do populamento de um meta-modelo descrito em UML que represente a organização típica dos ficheiros XML possamos construir expressões OCL que executem consultas que representarão as métricas que queiramos definir. Para tal será necessário também construir um analisador léxico-sintáctico que, para cada ficheiros XML, produza um ficheiro compatível com o populamento do modelo da aplicação USE.

2.2 Analisador Léxico-Sintáctico

Esta aplicação tem como principal objectivo a criação de um ficheiro com extensão ".cmd" que irá ser utilizado para popular o meta-modelo do XML [6], que se encontra em formato ".use". O meta-modelo do XML é instanciado com o recurso à aplicação *use*, que executa o ficheiro ".cmd" gerado pelo analisador léxico-sintático, carregando os objectos para o meta-modelo do XML.

O analisador léxico-sintático foi desenvolvido utilizando o framework .NETda Microsoft, tendo como linguagem de base o C# [7]. A figura seguinte, ilustra a interface gráfica disponibilizada pela aplicação ao utilizador. Sob esta interface, o utilizador poderá escolher qual o ficheiro XML que pretende analisar, bem como escolher quais as métricas associadas ao documento que pretende analisar.

Figura 2 - Analisador Léxico-Sintático

Depois do utilizador escolher qual o ficheiro XML e as métricas que pretende analisar, a aplicação carrega toda a estrutura do ficheiro em DOM (Document Object Module) onde é efectuada a navegação sobre a estrutura do documento XML. À medida que são encontrados elementos, elementos raíz, nós, nós de texto ou atributos, são criados comandos *USE* no ficheiro gerado pela aplicação, que permite a criação dos vários objectos e definição do respectivo

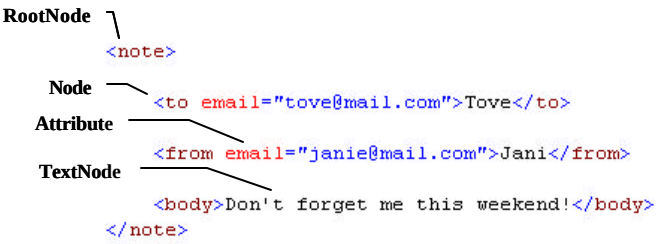

Figura 3 - Exemplificação dos objectos reconhecidos pelas classes do meta-modelo XML

estado, bem como as várias ligações entre eles que irão servir de base para posterior preenchimento do meta-modelo do XML.

Foi adoptado neste estudo o meta-modelo do XML especificado na figura 4.

Numa breve descrição, este modelo considera que toda a informação contida num ficheiro XML pode ser considerada um elemento que terá um nome do tipo *String*. Derivando da classe abstracta *Element* podemos ter Atributos (clas-

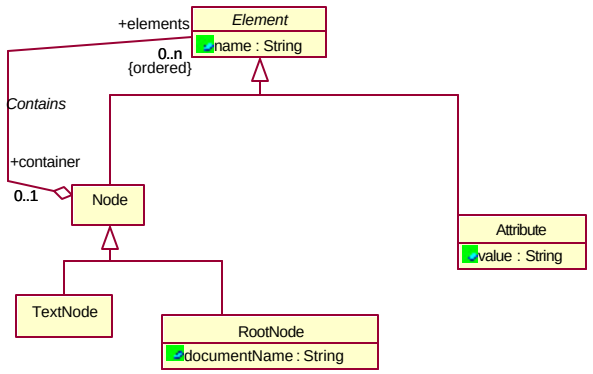

Figura 4 - Especificação do meta-modelo do XML adoptado

se *Atribute*) ou Nós (classe *Node*). Os Atributos para além do nome terão também um valor do tipo *String*. As classes *TextNode* e *RootNode* são casos particulares da classe *Node*, representando respectivamente o texto dentro de um elemento e o nó da raíz do documento. A associação "Contains" permite expressar a relação de paternidade que um nó pode ter sobre outros, sejam eles atributos ou outros nós.

Quando a expressividade de um modelo descrito em UML é insuficiente podemos usar expressões OCL que nos irão permitir impor restrições ao modelo desenhado de modo que, ao implementarmos este meta-modelo com a ajuda da ferramenta USE, decidimos impor três invariantes que qualquer ficheiro XML deve cumprir:

1. Só pode haver um *RootNode*.
2. Um objecto do tipo *TextNode* não pode ter filhos
3. Um objecto não pode ser filho dele próprio;

Quando a navegação do documento termina, a aplicação gera no ficheiro ".cmd" os comandos de invocação de funções em OCL, que representam as métricas associadas à escolha do utilizador no menu inicial. Depois da criação do ficheiro ".cmd" e do carregamento do modelo no USE será necessário que ler o ficeiro ".cmd" na mesma sessão que carregou o modelo de forma a ser efectuada a instanciação

e no final a apresentação dos resultados das métricas. Na interface do *USE* o utilizador poderá escolher métricas para nós específicos que pretenda analisar em maior detalhe.

Usando o documento da figura 3 como exemplo e com base no ficheiro ".cmd" gerado (figura 6), foi criado o diagrama de objectos a partir da ferramenta *USE*, ilustrado na figura 5.

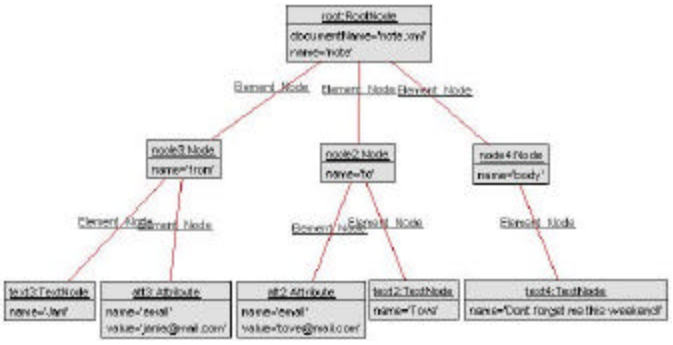

Figure 5 - Diagrama de Objectos

Como se pode analisar da figura 6, as primeiras três linhas de código correspondem à criação do elemento raíz do documento XML analisado, associado com o nome do documento XML. Em seguida, é criado o primeiro elemento nó pertencente ao elemento raíz do documento. Depois de criado o elemento nó, são criados os atributos respeitantes a esse nó. Uma vez criado o elemento nó e os seus atributos é efectuada a associação do elemento nó ao elemento raíz e dos atributos ao elemento nó. A criação e associação de elementos texto seguem a mesma abordagem.

2.3 Métricas

Tendo o meta-modelo definido, seria necessário pensar que tipo de dados seriam passíveis de se retirar e que uso teriam esses dados isoladamente ou no seu conjunto.

Todas as métricas têm como ponto de referência o nó da raíz, sendo como que métodos da classe RootNode, e foram expressas usando as capacidades da linguagem OCL[8].

Começámos por calcular números absolutos de nós, atributos e nós de texto:

1. **Número Total de Nós:**
   TE(): Integer = Node.allInstances->
   select(oclIsTypeOf(Node))->size()
   Isoladamente este valor irá fornecer informação directamente relacionada com a dimensão da estrutura do ficheiro e servir de base para futuras comparações.

2. **Número Total de Nós filhos da Raíz:**
   EN(): Integer = elements->select(oclIsTypeOf(Node))->size()
   A informação pode estar armazenada a vários níveis de profundidade de elementos, contudo o nó que representa a raiz tem propriedades que o definem como referência para todos os outros. Ter a informação do número de nós que lhe pertence pode, em conjugação com outro tipo de dados, pode permitir decisões sobre o número de iterações necessárias para a recolha da informação presente no documento.

```
reset
!create root: RootNode
!set root.documentName='note.xml'
!set root.name = 'note'
!create node2: Node
!set node2.name = 'to'
!create att2: Attribute
!set att2.name = 'email'
!set att2.value = 'tove@mail.com'
!insert (att2, node2)into Element_Node
!insert (node2,root)into Element_Node
!create text2: TextNode
!set text2.name = 'Tove'
!insert (text2, node2)into Element_Node
!create node3: Node
!set node3.name = 'from'
!create att3: Attribute
!set att3.name = 'email'
!set att3.value = 'janie@mail.com'
!insert (att3, node3)into Element_Node
!insert (node3,root)into Element_Node
!create text3: TextNode
!set text3.name = 'Jani'
!insert (text3, node3)into Element_Node
!create node4: Node
!set node4.name = 'body'
!insert (node4,root)into Element_Node
!create text4: TextNode
!set text4.name = 'Dont forget me this weekend!'
!insert (text4, node4)into Element_Node
check
?root.maxDepth(1)
?root.depth()
?root.parent()
?root.EN()
?root.TE()
?root.TA()
?root.TTN()
?root.DifE()
?root.DifA()
?root.DifTN()
?root.DifER()
?root.DifER()
?root.NELevel(1)
?root.RatElem()
?root.RatAttribElem()
?root.RatTextElem()
?root.RatTextUText()
?root.RatAttribUAttrib()
?root.SameAtt()
?root.SameParent()
?root.SameElem()
?root.UniqueRootChild()
```

Figure 6 - ".cmd" gerado

3. **Número Total de Atributos:**
   TA(): Integer = Attribute.allInstances->size()
   Ter a noção de quantos atributos existem em todo o documento tem como principal objectivo perceber se a

organização do XML passou pela vasta utilização de atributos para ajudar no armazenamento da informação e servir de base para várias comparações.

4. Número Total de Nós de Texto:
TTN(): Integer = TextNode.allInstances->size()
Em conexão com a métrica anterior, convém também ter a noção do tipo de utilização dos nós de texto voltando este dado a ser usado para vários tipos de comparações entre elementos.

No geral estas métricas darão alguma percepção da dimensão do documento, conseguindo já ter uma primeira noção se a informação está contida em Atributos ou em Nós de Texto.

Outro conjunto de valores que tem interesse em calcular e que irá complementar o anterior é o conjunto de métricas que se baseiam no número total de elementos diferentes, sendo que o que os identifica será sempre o seu nome. Desta forma calculámos:

5. Número de Nós diferentes:
DifE():Integer = Node.allInstances->
select(oclIsTypeOf(Node))->collect(name)->asSet->size()
Se pensarmos que cada tipo de nó representará uma entidade, podendo transpor esse conceito para objectos ou tabelas, este dado irá fornecer uma rápida ideia do número de entidades diferentes representadas no ficheiro XML analisado.

6. Número de Nós diferentes que são filhos da Raíz:
DifER():Integer = elements->select(oclIsTypeOf(Node))->
collect(name)->asSet->size()
Se aplicarmos o mesmo tipo de raciocínio utilizado na anterior definição percebemos que com esta métrica poderemos descobrir o número entidades de tipos diferentes a raiz da estrutura contém.

7. Número de Atributos diferentes:
DifA():Integer = Attribute.allInstances->
collect(name)->asSet->size()
Apesar de já sabermos através de diferentes métricas a quantidade total de atributos usada poderá de ser de maior importância saber o número de atributos diferentes usados para poder chegar a conclusões sobre a repetição deste tipo de elementos de informação.

8. Número de Nós de Texto diferentes:
DifTN():Integer = TextNode.allInstances->
collect(name)->asSet->size()
Um dos factores mais difíceis de contornar no desenho das estruturas XML é a factorização da informação de forma a evitar a duplicação de informação na base de dados daí que seja importante tirar este tipo de informação que posteriormente seja processada por forma a ser possível detectar este tipos de casos.

Deste conjunto vale a pena voltar a destacar a possível de tirar conclusões referentes à heterogeneidade dos filhos da raíz. Isto pode ser importante para que numa recolha automática de informação contida em ficheiros XML seja possível perceber se estamos perante um documento que descreve uma colecção de diferentes tipos de objectos ou apenas descreve um tipo de objectos.

A intersecção destes dois últimos grupos de métricas resulta no cálculo de rácios que tentam exprimir a relação entre alguns destes totais apresentados. Os rácios são no fundo uma forma de uniformizar a comparação diferentes documentos XML. No nosso trabalho calculámos cinco rácios, mas será sempre possível calcular muitos outros.

9. Número de Nós diferentes /Número total de Nós:
RatElem():Real = 1 - DifE()/TE()
Permite obter o factor de repetição dos nós do documento permitindo perceber se todos os nós são únicos ou com que frequência se pode achar nós de tipos repetidos.

10. Número de Atributos diferentes /Número total de Atributos:
RatAttribUAttrib():Real = 1 - DifA()/TA()
Permite obter o factor de repetição dos atributos permitindo perceber se existe algum tipo de reaproveitamento dos atributos mencionados.

11. Número de Nós de Texto diferentes/Número total de Nós de Texto:
RatTextUText():Real = 1 - DifTN()/TTN()
Permite obter o factor de repetição dos nós de texto o que pode levar a ter noção da existência de duplicação de informação e, caso seja apropriado, aplicar mecanismos de facturização de informação.

12. Número total de Atributos / Número total de Nós:
RatAttribElem():Real = TA()/TE()
Permite calcular a média de atributos por nó, o que traduzirá numa medida de utilização do elemento atributo ao longo de todo o documento.

13. Número total de Nós de Texto / Número total de Nós):
RatTextElem():Real = TTN()/TE()
Permite calcular a média de nós de texto por nó, o que leva a tirar conclusões sobre a utilização dos nós de texto para o armazenamento de texto. No caso extremo de ter o valor um significa que cada nó terá uma informação guarda em nó de texto associada e no caso contrário, valor 0 deste cálculo, significará que nenhum elemento tem nós de texto associados

Foram também pensadas métricas que incidem sobre a profundidade dos elementos, pelo que foi necessário definir o método que calculasse a profundidade de um elemento e partindo desse dado foi fácil chegar a métricas como:

14. Profundidade máxima do documento:
maxDepth(d:Integer):Integer = if elements-> se-
lect(oclIsTypeOf(Node))->size()=0 then d else elements->
select(oclIsTypeOf(Node))->iterate(elem; acc : Integer = d |
acc.max(elem.oclAsType(Node).maxDepth(d+1)))endif
Importante para conhecer o tipo de estrutura que o documento apresenta, começando a ter uma noção se esta é do tipo achatada ou imbricada. Em conjunto com outros dados podem ser feitas médias relacionando os elementos com a profundidade máxima.

15. Número de Nós à profundidade X:
NELevel(d:Integer):Integer = if d<=1 then 1 else Node.allInstances->select(oclIsTypeOf(Node) and depth() = d)->size() endif
No seguimento da anterior medida este valor permite-nos compreender a densidade de informação presente nos vários níveis de hierarquia do documento.

Durante todos estes cálculos nunca perdemos de vista um dos nossos maiores objectivos, conseguir perceber se o documento apresenta uma estrutura regular e homogénea. Isto é relevante pois se pensarmos numa recolha automatizada de informação, tendo por base ficheiros XML, é importante perceber se estamos perante um documento que respeita constantemente a mesma estrutura de nós e atributos, ou se a sua organização não é de todo rígida com casos que se distinguem dos outros.

Do nosso ponto de vista, existem várias formas de definir o que é um documento XML homogéneo. Podemos dizer que para ser considerado como tal basta que todo um nó de nome X tem de ter sempre os mesmos tipos de atributos e filhos ao longo de todo o documento, mas se quisermos podemos considerar que todos os filhos da raíz devem ser iguais para que o tratamento seja uniformizado sobre o mesmo tipo de objectos ou podemos ir ainda mais longe defendendo que é também necessário que um nó de nome Y deve, ao longo de todo o documento tenha sempre o mesmo pai.

Para analisar o cumprimento de cada uma destas condições de homegeneidade oferecemos três predicados booleanos:

16. Todos os nós têm os mesmos filhos e atributos ao longo do documento:
SameElem():Boolean=Node.allInstances-> select(oclIsTypeOf(Node))->collect(name)->asSet()-> forAll(n: String | Node.allInstances->select(oclIsTypeOf(Node) and name=n)->forAll(n1,n2:Node | n1.elements-> select(not oclIsTypeOf(TextNode))->collect(name) = n2.elements ->select(not oclIsTypeOf(TextNode))-> collect(name)))
Verifica se todos os nós respeitam a estrutura típica do seu tipo, apresentando constantemente o mesmo conjunto de atributos e nós filhos.

17. A raíz só tem filhos do mesmo tipo:
UniqueRootChild():Boolean = DifER() = 1
Verifica se a raíz tem apenas um tipo de nó associado compreendendo assim que se trata de uma colecção específica de um determinado tipo de dados e não vários tipos de dados que não se relacionam estruturalmente uns com os outros.

18. Todos os nós têm o mesmo pai ao longo do documento:
SameParent():Boolean = Node.allInstances-> select(oclIsTypeOf(Node))->collect(name)-> sset()-> forAll(n: String |Node.allInstances-> select(oclIsTypeOf(Node) and name=n)->forAll(n1,n2:Node | n1.parent() = n2.parent()))
Verifica ser todos os nós respeitarem as mesmas relações de parentesco o que pode exprimir uma regularidade ainda maior das relações na estrutura XML.

A definição de homogeneidade pode ser construída através de uma composição destes três predicados, satisfazendo assim a necessidade do momento.

Algumas destas métricas, principalmente a que consta em primeiro deste grupo, revelaram uma fraca performance ao serem executadas com documentos XML de tamanho considerável (>2Mb).

A abertura para se exprimir outro tipo de métricas é bastante. Uma vez preenchido o meta-modelo, qualquer utilizador com conhecimentos de OCL pode enriquecer o conjunto de métricas aqui explanado com as que vão de encontro ao seu interesse.

3 EXEMPLOS DE INTERPRETAÇÕES

Vamos nesta secção tentar interpretar alguns resultados obtidos com o nosso sistema em duas situações diferentes e tentar perceber que tipo de ficheiro estamos a analisar em cada uma das situações.

3.1 Primeiro Caso

Imaginemos que obteríamos as seguintes métricas de um ficheiro XML que desconhecíamos. Que conclusões se poriam tirar?

Métrica	Valor
Nº total de nós	30
Nº total de nós filhos da raíz	30
Nº total de atributos	90
Nº total de nós de texto	0
Nº de nós diferentes	1
Nº de nós filhos da raíz diferentes	1
Nº de atributos diferentes	3
Factor de repetição dos nós	0.97
Factor de repetição dos atributos	0.97
Média de atributos por nós	3
Média de nós de texto por nós	0
Profundidade Máxima	1
Todos os nós com os mesmos atributos e filhos	True
Raíz só tem filhos do mesmo tipo	True
Todos os nós com o mesmo pai	True

Se começarmos por analisar a métrica que nos indica a profundidade máxima da estrutura apercebemo-nos que o documento tem uma estrutura rasa com apenas um nível abaixo da raíz. Para confirmar esta assunção temos que todos os 30 nós existentes são filhos da raiz e mais podemos conhecer sobre esses nós ao observarmos que apenas um tipo de nós existe e que a raiz só tem um tipo de nós chegando à conclusão que o nosso ficheiro terá para já uma estrutura semelhante à seguinte:

```
<root>
    <node ...>...</node>
    <node ...>...</node>
    <node ...>...</node>
    ...
</root>
```

Se agora formos observar outro tipo de métricas que nos permitam caracterizar ainda mais o nosso ficheiro, podemos ver que não existem nós de texto pelo que a informa-

ção a ser guardada será nos atributos. Olhando então sobre as métricas que manipulam os atributos temos um total de 90 atributos dando isto uma média de 3 atributos por nó. Facilmente apercebemos que se todos os nós respeitam o mesmo conjunto de atributos e como só existem 3 tipos de atributos diferentes, a estrutura do ficheiro terá o seguinte aspecto:

```
<root>
    <node attX=... attY=… attW=… />
    <node attX=... attY=… attW=… />
    <node attX=... attY=… attW=… />
    ...
</root>
```

Adicionalmente sabemos que a estrutura <node attX=... attY=… attW=… /> se repetirá 30 vezes.

3.2 Segundo Caso

4 Trabalho Relacionado

Apesar do extenso trabalho já produzido sobre métricas de software e modelo relacionais de bases de dados, ainda não são muitas as pesquisas realizadas sobre métricas e extracção de características qualitativas de documentos XML.

Um dos primeiros trabalhos neste área foi o artigo "Everthing you ever wanted to know about DTDs but were afraid to ask" from Arnaud Sahguet [Sah00][9] que descreve as principais características de um DTD e realiza métricas que podemos chamar de dimensão sobre os vários componentes que um DTD identifica. Este documento foi estendido pelo artigo "DTD mining labor day report" [CS00][10] que apresenta mais de 60 DTDs nas suas formas mais usuais, classificando os erros mais comuns que podem ocorrer na sus construção e aproveitando para acrescentar mais algumas métricas sobre DTDs.

O artigo "A few tips for good XML design" [Cho00][11], para além de sugestões de como desenhar ficheiros XML explica como algumas medidas simples podem ajudar a caracterizar seguindo o tamanho, a clareza, o *Parsing*, a navegabilidade, o mapeamento e a modularização; ainda outro artigo de relevância para o nosso estudo é o "Métrics for XML document collection"[12] de Meike Klettk, L. Shneider e A. Heuer que apresenta um pequeno mas útil conjunto de métricas exemplificando o seu cálculo e explicando a sua aplicabilidade.

O projecto de investigação *ToX* que significa *TOronto Xml server* que se encontra em fase de desenvolvimento, por supõe a construção de um servidor de dados XML que suporte o armazenamento de grandes volumes de dados e que seja escalável. O principal objectivo é fornecer a mesma funcionalidade para os dados XML que as actuais DBMS (Database Management Systems) providenciam para os tradicionais tipos de dados. As questões que a equipa de desenvolvimento se depara neste momento, têm a ver com o armazenamento eficiente de dados; estrutura de índices para dados/documentos XML; optimização de técnicas de *querys* para dados XML; métricas para a estrutura de documentos XML e também análises de armazenamento e *querys* para dados XML.

4 Conclusões & Trabalho Futuro

Este trabalho permitiu-nos concluir que é possível através das métricas desenvolvidas, obter de uma forma coerente a caracterização de documentos XML.

A metodologia usada permite bastante flexibilidade ao nível da aplicação, pois é sempre possível estender um conjunto de métricas consoante as necessidades dos utilizadores. A criação das métricas apresentadas permite a recolha de uma forma automática e simples da informação contida nos documentos XML.

A utilidade deste tipo de ferramentas tem bastante aplicabilidade em vários domínios da engenharia do software. Pode servir de base para a concpeção de um sistema de simples catalogação de documentos XML; pode ser incluída num sistema de apoio à decisão que fornecça orientações sobre qual a melhor forma de modelar um repositório de dados de acordo com a estrutura do documento XML analisado; ou pode, até mesmo, estar integrada com um sistema que produza de forma automática um repositório de dados mediante a estrutura de qualquer ficheiro XML.

Contudo será ainda necessário o melhoramento do desempenho de algumas métricas apresentadas, bem como a criação de outras igualmente úteis.

Referências

[1] – http://www.w3.org/XML/
[2] - http://ctp.di.fct.unl.pt/mei/qpp/
[3] – OMG Unified Modeling Language Specification, Version 1.3, June 1999. Object Management Group, Inc., Framingham, Mass., Internet: http://www.omg.org, 1999
[4] – http://www.omg.org
[5] – http://www.db.informatik.uni-bremen.de/projects/USE/
[6] – http://java.sun.com/products/jmi/download.html
[7] – http://msdn.microsoft.com/vcsharp/
[8] – Abreu, Fernando Brito. "Using OCL to formalize object oriented metrics definitions". NESC Technical Report, Junho 2001
[9] – Arnaud Sahuguet. Everthing you ever wanted to know about DTDs but were afraid to ask. In *WebDB-2000*, 2000
[10]– Byron Choi, Arnaud Sahuguet. "DTD Mining Labor Day Report". September 2000
[11]– Byron Choi "A few tips for good XML design" 2000.
[12] – Meike Klettk, L. Shneider, A. Heuer. "Métrics for XML document collection". EDBT Workshops 2002.