XSDelta – Uma Ferramenta Visual para Comparação de Esquemas XML

Augusto Belotto Perini, Vincent Nelson Kellers da Silveira e Renata de Matos Galante

Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil

{aperini, vincent, galante}@inf.ufrgs.br

Abstract. *This paper describes XSDelta, a graphical tool designed for comparison of XML schemas expressed through both DTD and XML Schema. By using a diff algorithm for detecting changes in XML documents, XSDelta exposes to the user all operations required to transform an original XML schema into its new version. XSDelta presents schemas textually, making modified parts stand out through distinct colors and identifying the schemas changes performed. As output, XSDelta generates XML files containing identified differences, assisting semi-structured databases administrators in the XML schemas evolution process management.*

Resumo. *Este trabalho apresenta o XSDelta, uma ferramenta gráfica capaz de comparar versões de esquemas XML definidos tanto em DTD quanto em XML Schema. Utilizando um algoritmo de detecção de diferenças em documentos XML, o XSDelta expõe ao usuário todas as operações envolvidas na transformação de um esquema original em sua nova versão. O XSDelta é capaz de exibir textualmente os esquemas, realçando através de cores distintas as partes modificadas e identificando as mudanças de esquema realizadas. Como saída, o XSDelta gera arquivos XML contendo as diferenças identificadas, auxiliando administradores de bancos de dados semi-estruturados na gerência do processo de evolução de esquemas XML.*

1. Introdução

A utilização de esquemas XML como um meio de definir e especificar regras de validação sobre as estruturas e conteúdo de documentos XML tornou a gerência da evolução de esquemas XML um foco de atenção por parte de administradores de bancos de dados semi-estruturados.

Manutenções em bases de dados semi-estruturadas que incluam a modificação de esquemas podem comprometer consultas e a validade de documentos XML [Guerrini 2005]. Essa evolução pode ser entendida como um processo natural de amadurecimento das estruturas de dados que descrevem as informações disponíveis aos sistemas computacionais, podendo ocorrer sob diversas circunstâncias, como refinamento de requisitos de aplicações XML, alteração na modelagem lógica de um banco de dados, eliminação de *bugs*, etc. A detecção da evolução no contexto de esquemas XML possibilita que os administradores de bases de dados semi-estruturadas compreendam as

modificações ocorridas, de modo que elas sejam propagadas aos documentos que referenciam tais esquemas, impedindo que se tornem inválidos [Guerrini 2005].

A Figura 1 é um exemplo de evolução de esquema (representado por uma árvore XML) onde o atributo "**id**" e o elemento "**ano**" são adicionados ao elemento "**artigo**".

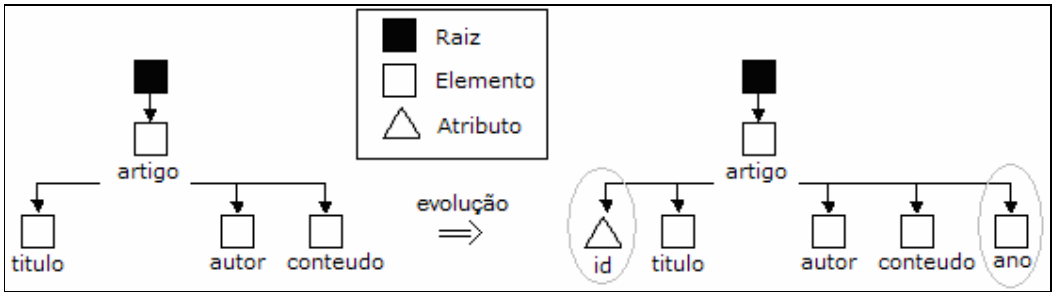

Figura 1. Exemplo de evolução de esquema

Na literatura há diversos algoritmos capazes de identificar diferenças em árvores XML. O XyDiff foi o algoritmo escolhido para implementar no XSDelta o processo de detecção de diferenças em função de sua ordem de complexidade, do conjunto de operações de *diff* suportadas e da implementação *open source* disponível. Cabe ressaltar que o XSDelta é projetado de maneira que o algoritmo escolhido possa ser facilmente substituído por outro de mesmo propósito que se mostre mais adequado segundo os critérios abordados na Seção 2.

O objetivo do XSDelta é detectar as diferenças entre duas versões de um esquema por meio de um algoritmo de detecção de diferenças em documentos XML. Esquemas definidos em XML *Schema* possuem sintaxe XML [W3C 2004A], sendo documentos bem formados e aptos ao processo de detecção de diferenças proposto. Esquemas DTD possuem sintaxe própria. Devido à simplicidade e limitações da DTD, o XSDelta possui a funcionalidade de conversão do formato DTD para o formato XML *Schema*, viabilizando a submissão de esquemas definidos em ambas linguagens ao mesmo processo de detecção de diferenças.

O XSDelta trata a transformação de um esquema em outro como a aplicação de sucessivas operações de modificação nas estruturas das suas árvores XML. As operações em questão incluem a inserção e remoção de nodos, atualização de valores dos nodos e movimentação de sub-árvores. O exemplo na Figura 1 indica a inserção de dois nodos na árvore XML. Além disso, o XSDelta exibe textualmente os esquemas, realçando através de cores distintas as partes modificadas, juntamente com a lista de operações de edição realizadas. Como saída, o XSDelta gera arquivos XML contendo as diferenças identificadas.

Este trabalho está dividido da seguinte maneira: A Seção 2 apresenta as principais características dos algoritmos de detecção de diferenças em documentos XML e justifica a escolha do XyDiff. A Seção 3 apresenta a arquitetura e as funcionalidades do XSDelta. Seção 4 encerra o trabalho com considerações finais, relacionando trabalhos futuros.

2. Algoritmos de detecção de diferenças em documentos XML

Diversos algoritmos surgiram para o cômputo de diferenças entre dados hierarquicamente estruturados, representados sob o formato de árvores. Como saída, eles retornam um *script*, também conhecido como *script delta*, que contém todas as operações para transformação de um documento em outro. Após comparar os esquemas, o XSDelta processa este *script delta* para apresentar as operações ao usuário.

No estudo de [Peters 2005] são comparados dezessete algoritmos que detectam diferenças em árvores XML, que variam quanto ao foco da detecção, vantagens e desvantagens sob diferentes cenários, otimizações e requisitos peculiares. Os itens comparados entre os algoritmos incluem o tratamento de árvores ordenadas ou não-ordenadas (se a ordem entre nodos irmãos é relevante ou não); a corretude (qualidade da saída gerada, o *script delta*), estando relacionada com a busca pelo *matching* ótimo entre nodos; o uso de memória e tamanho das entradas (que pode ser um limitante para certos algoritmos); o uso de semântica para a comparação de nodos, como utilização de chaves ou *id's*, proporcionando uma condição única para o casamento de elementos; tamanho da saída (tamanho dos *scripts delta* gerados, influenciado pelo suporte a determinadas operações de *diff*); suporte a operações especializadas como *move* e *copy,* garantindo *scripts* mais compactos e complexidade de tempo e de recursos.

2.1 XyDiff - Detecting Changes in XML Documents

O XyDiff [Cobéna 2002] mostrou-se adequado ao XSDelta, na medida em que suporta as operações de *diff* do tipo *insert*, *delete*, *update* e *move,* o que contribui para diminuir o tamanho do *script delta*, e que sua complexidade de execução é da ordem de *O*(nlogn), adotando práticas que focam na otimização de uso de memória e tempo de processamento, e lida com árvores ordenadas (considera que cada nodo da árvore contém uma lista ordenada de nodos filhos). Em [ZHANG 1992] demonstra-se que a detecção de mudanças em árvores não-ordenadas é substancialmente mais difícil e custosa. Para o cômputo das diferenças e geração do *delta*, o XyDiff é dividido em cinco fases:

- Fase 1 - busca pelo casamento único entre nodos, por meio de atributos *ID*;
- Fase 2 - atribuição de assinaturas e pesos aos nodos e ordenamento de sub-árvores através de uma fila de prioridade;
- Fase 3 - procura por casamentos priorizando nodos e sub- árvores de maior peso;
- Fase 4 - otimização dos casamentos pela procura por nodos casados cujos pais possuam mesma assinatura, com varredura *bottom-up* e posterior varredura *top-down* das árvores;
- Fase 5 – reconhecimento das operações *insert*, *delete*, *update* e *move* e geração do *script delta*.

3. XSDelta - Uma Ferramenta Visual para Comparação de Esquemas XML

Esta seção descreve o XSDelta, uma ferramenta visual para comparação de esquemas XML desenvolvida para auxiliar administradores de bancos de dados semi-estruturados na gerência do processo de evolução de esquemas XML.

3.1. Arquitetura da Ferramenta

Os quatro casos de uso (Editar esquema, Converter esquema, Detectar diferenças e Gerar delta) ilustrados na Figura 2, identificados no momento da modelagem do XSDelta, encapsulam todas as funcionalidades da ferramenta (vide Seção 3.2).

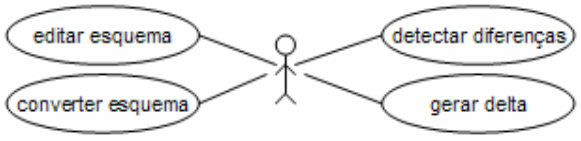

Figura 2. Diagrama de casos de uso do XSDelta

A linguagem de programação Java foi utilizada na implementação do XSDelta. Para a diagramação relativa a parte de análise e projeto foi utilizado o ArgoUML[1], que é uma ferramenta *open source* para modelagem UML que roda sobre a plataforma Java. O ambiente de desenvolvimento utilizado foi o Eclipse 3.1[2].

Para a fase de experimentação da ferramenta XSDelta estão sendo usadas bases de dados oriundas da DBLP[3] (*Digital Bibliography & Library Project*), entre outras, que modelam o domínio de artigos científicos em Ciência da Computação. Estas bases de dados[4] utilizadas na etapa de testes são importantes para o refinamento do XSDelta.

3.2. Funcionalidades do XSDelta

O XSDelta possui três funcionalidades principais: (i) conversão de esquemas DTD para esquemas XML *Schema*; (ii) processamento dos *scripts delta* para exibição visual das operações de evolução; e (iii) geração do arquivo *delta*. Duas perspectivas agrupam essas funcionalidades: esquema para o item (i) e evolução para os itens (ii) e (iii).

O fluxo básico de utilização do XSDelta está detalhado na Figura 3. Os esquemas XML são submetidos ao processo de detecção de diferenças. Esquemas que não são documentos XML (esquemas DTD) seguem o fluxo alternativo, descrito por setas tracejadas, para que sejam convertidos para o formato XML *Schema*.

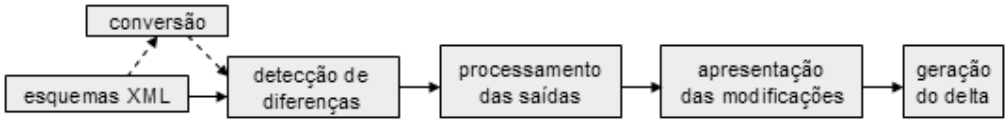

Figura 3. Fluxo básico do XSDelta

Após detectar as diferenças, o XSDelta processa as saídas do algoritmo escolhido para apresentar as modificações entre os esquemas, juntamente com a opção de geração do arquivo XML (geração do delta). O delta gerado pelo XSDelta é um documento XML. Esse documento contém os conteúdos dos esquemas comparados e as operações de evolução detectadas pelo algoritmo de *diff* escolhido.

---

[1] http://argouml.tigris.org/

[2] http://www.eclipse.org/

[3] http://dblp.uni-trier.de/

[4] Bases disponíveis em <http://dblp.uni-trier.de/xml>. Acesso em 03/05/2006.

3.2.1. Conversão de esquemas

A Figura 3 ilustra um esquema em formato DTD (lado esquerdo) sendo convertido para o formato XML *Schema* (lado direito). É através da perspectiva ESQUEMA que o XSDelta viabiliza a análise da evolução de esquemas DTD, pois, após serem convertidos para XML *Schema*, esquemas DTD tornam-se entradas válidas ao algoritmo de detecção de diferenças utilizado na perspectiva EVOLUÇÃO.

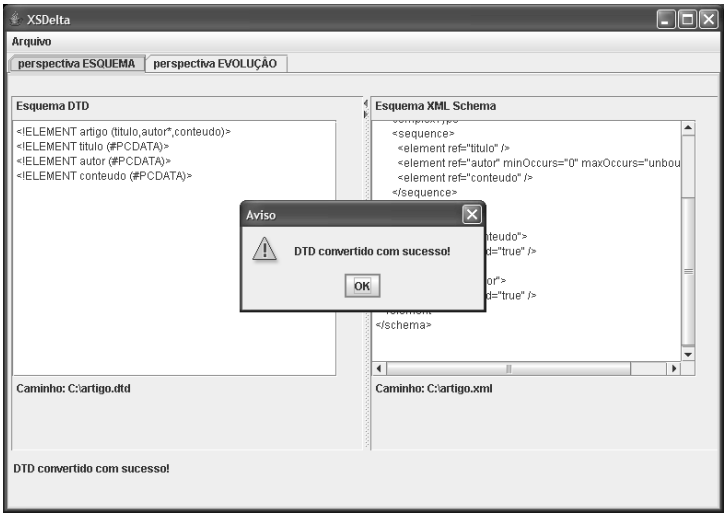

Figura 3. Perspectiva ESQUEMA

3.2.2. Visualização da evolução do esquema

A Figura 4 ilustra o resultado da comparação de um esquema com sua nova versão.

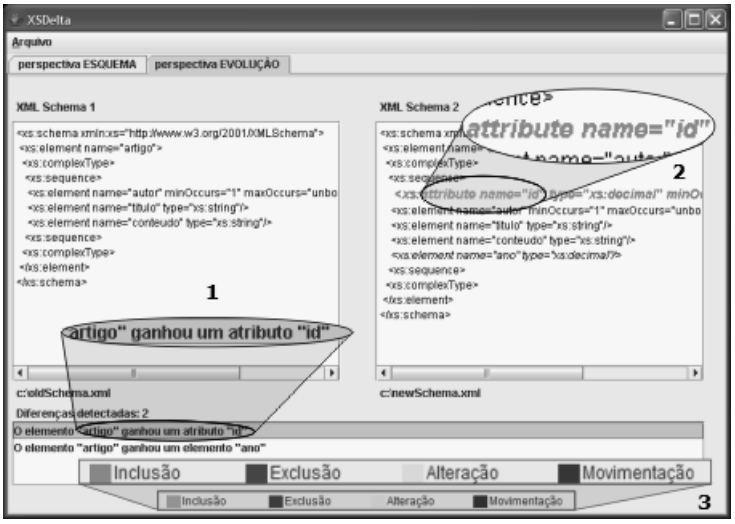

Figura 4. Perspectiva EVOLUÇÃO

Neste exemplo o XSDelta identificou as operações de inserção do novo atributo **id** e do novo elemento **ano**, ambas ao elemento já existente **artigo**. Todas as operações identificadas aparecem na lista de diferenças detectadas. O usuário do XSDelta pode selecionar livremente as operações desta lista (ampliação **1**) para que porções dos

conteúdos dos esquemas XML recebam destaques coloridos (ampliação **2**). Na parte inferior da tela encontra-se a legenda com o significado das cores utilizadas no destaque das porções de interesse dos esquemas (ampliação **3**).

As ampliações numeradas são recursos ilustrativos, que visam facilitar a identificação dos elementos na interface e não integram a implementação do XSDelta.

4. Conclusões e trabalhos futuros

Este trabalho apresentou o XSDelta, uma ferramenta visual capaz de descrever textual e graficamente modificações realizadas em esquemas definidos em DTD e *XML Schema*, no intuito de possibilitar aos administradores de bancos de dados semi-estruturados uma melhor compreensão a respeito do processo de evolução de esquemas XML.

Atualmente os módulos do XSDelta responsáveis por processar a saída do algoritmo de detecção de diferenças em documentos XML escolhido e por gerar o arquivo XML *delta* estão sendo testados e necessitam de refinamentos. O módulo `processamento das saídas` vem sendo projetado para que a integração de outros algoritmos de mesmo propósito seja facilitada. Esse módulo da ferramenta deve ser estendido para cada algoritmo que se deseje incluir suporte, pois eles não retornam o *script* com as operações em formato padrão. A ferramenta não se limita a comparar esquemas XML, podendo inclusive detectar diferenças em documentos XML de conteúdo, dada a natureza dos algoritmos estudados.

Há um trabalho de mestrado em Ciência da Computação no Instituto de Informática da UFRGS em andamento que aborda a propagação de modificações realizadas em esquemas XML para documentos XML, particularmente interessado nos resultados alcançados pelo XSDelta, pois seu primeiro passo consiste na identificação das modificações realizadas nos esquema XML. Este, e outros trabalhos relacionados, representam estímulos à melhoria dos resultados atualmente obtidos pelo XSDelta.

5. Referências

Cobéna, G., Abiteboul, S. and Marian, A. (2002) "Detecting Changes in XML Documents", In: 18th IEEE International Conference on Data Engineering (ICDE), Washington, DC, USA. Proceedings... IEEE Computer Society, p. 41-52.

Guerrini G., Mesiti M., Rossi D. (2005) "Impact of XML Schema Evolution on Valid Documents", In: 7th annual ACM international Workshop on Web Information and data Management (WIDM), Bremen, Germany. Proceedings… ACM Press p. 39-44.

Peters, L. J. (2005) "Change Detection in XML Tress: a Survey", In: 3rd Twente Student Conference on IT, University of Twente, Enschede, Netherlands. June.

W3C. (2004). "Extensible Markup Language (XML) 1.0 (Third Edition)", W3C Recommendation, http://www.w3.org/TR/2004/REC-xml-20040204/, February.

W3C. (2004). "XML Schema Part 0: Primer Second Edition", W3C Recommendation, http://www.w3.org/TR/xmlschema-0/, October.

Zhang, K., Statman, R., Shasha, D. (1992) "On the Editing Distance between Unordered Labeled Trees", In: Information Processing Letters, 42(3), Amsterdam, The Netherlands. Proceedings… Elsevier North-Holland, Inc. p. 133-139.