UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

FELIPE DOS SANTOS WILKE

X-Spread 2 - Uma ferramenta operacional para propagação da evolução de esquemas XML para documentos XML

Trabalho de Graduação.

Prof. Dra. Renata Galante
Orientadora

Porto Alegre, dezembro de 2010.

AGRADECIMENTOS

A meus pais, Maria Helena e Flaviano, por todos ensinamentos e pela enorme dedicação para que eu tivesse todas as condições necessárias para concluir esse curso. A minha vó Dione e ao meu tio Celso, por muitas vezes assumirem o papel de pais, por todo apoio e pelos conselhos e exemplos, fundamentais para a minha formação como um homem de caráter. A minha irmã Clarissa por ser especial e por ser um grande exemplo de personalidade, determinação e superação. A Verônica pela inspiração, pela paciência e pela certeza de que devemos viver intensamente. Aos meus familiares e amigos, por todos os momentos passados juntos e por ser a grande fonte de energia para enfrentar os desafios da vida. A IDEA Tecnologia e o trio Pitoni, Gustavo e Flávio, pela oportunidade, pelo aprendizado e por ser decisiva na minha continuação e conclusão desse curso. A Soft Design e PROCERGS, em especial a Osmar, Joel e Gilson, pela oportunidade e pela compreensão, fundamental nesse último ano de curso. A minha orientadora Prof. Dra. Renata Galante e ao Prof. Dr. Marcelo Pimenta pela grande contribuição na elaboração deste trabalho. Aos meus professores do Instituto de Informática e da UFRGS pelo ensinamento de qualidade ímpar.

A todos vocês, muito obrigado.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
DOM	*Document Object Model*
DTD	*Data Type Definition*
SGBD	*Sistema de Gerenciamento de Banco de Dados*
SGML	*Standard Generalized Markup Language*
XML	*Extensible Markup Language*
W3C	*World Wide Web Consortium*

LISTA DE FIGURAS

RESUMO

Esquemas e documentos XML são alterados ao longo do tempo para acomodar suas especificações que estão em constante evolução. A característica evolutiva dos esquemas pode por vezes comprometer a validade dos documentos a eles associados. Este trabalho propõe o X-Spread 2, uma ferramenta operacional, que propaga a evolução de esquemas XML para um conjunto de documentos XML. É realizada uma adaptação em um conjunto de documentos XML, a partir da detecção de diferenças entre duas versões de esquemas XML, informados pelo usuário, com o objetivo de tornar esse conjunto de documentos XML compatível com a versão mais recente do esquema XML e disponibilizá-lo novamente ao usuário. Este trabalho está baseado no protótipo não operacional implementado para o X-Spread (Silveira 2007, Silveira e Galante 2008). X-Spread é uma abordagem para a propagação automática da evolução de esquemas para documentos XML. Este TCC implementou os conceitos especificados para o X-Spread, revisando suas definições e incorporando novos algoritmos e idéias para a adaptação de documentos XML.

Palavras-chave: XML, esquemas, evolução, adaptação.

X-Spread 2 - An operational tool for propagating changes in XML schemas for XML documents

ABSTRACT

Schemas and XML documents are changed over time to accommodate their specifications that are constantly evolving. The evolutionary feature of the schemes can sometimes compromise the validity of the documents associated with them. We propose X-Spread 2, an operational tool that propagates the schema evolution for a set of XML documents. First, we identify the differences between two XML schema versions previously selected by the user. Then, we perform an adaptation process by making a set of XML documents compatible with the latest XML schema version. Finally, we make the set of XML documents available back to the user. This work is based on non-operational prototype implemented for the X-Spread (Silveira 2007, Silveira and Galante 2008). X-Spread is an automatic approach for propagating schema evolution for XML documents. The operational tool implemented specified here adopts the concepts specified for the X-Spread, by reviewing its definitions and incorporating both new ideas and algorithms for the adaptation of XML documents.

Keywords: XML, schema, evolution , adaptation.

1 INTRODUÇÃO

Aplicações podem sofrer alterações ao longo do tempo por diversos motivos, como, por exemplo, mudanças nas funcionalidades existentes ou o acréscimo de novas funcionalidades, podendo ser necessárias mudanças nas bases de dados associadas a essas aplicações. Logo, as bases de dados utilizadas por essas aplicações estão sujeitas a este processo de evolução.

O processo de evolução em bases de dados semi-estruturados difere das bases de dados relacionais, devido à ausência de um sistema gerenciador de banco de dados (SGBD) e a disjunção entre o esquema e as instâncias. Compostas por documentos e esquemas XML, nas bases de dados semi-estruturadas são as instâncias (documentos) que referenciam os esquemas, enquanto estes, por sua vez, possuem as definições de suas estruturas internas. Se uma estrutura do esquema de uma base de dados semi-estruturados é alterada, é necessária a propagação desta modificação às instâncias existentes, com o objetivo de garantir a integridade da base de dados. No caso de bancos de dados relacionais, alterações que colocam a base de dados em um estado inconsistente são bloqueadas pelo SGBD. Já em uma base de dados semi-estruturados, alterações nos esquemas podem ser feitas livremente, podendo tornar essa base de dados inconsistente. Portanto, mecanismos para verificar e garantir a integridade das bases de dados semi-estruturados são importantes para manter a consistência das instâncias frente ao esquema a elas associado.

Este trabalho propõe o X-Spread 2, uma ferramenta operacional que propaga a evolução de esquemas para um conjunto de documentos XML com o objetivo de tornar uma base de dados semi-estruturados novamente consistente. O X-Spread 2 realiza uma adaptação em um conjunto de documentos XML a partir da detecção de diferenças entre duas versões esquemas XML informados pelo usuário, com o objetivo de tornar esse

conjunto de documentos XML compatível com a versão mais recente do esquema XML e disponibilizá-lo ao usuário. Cabe ressaltar que este trabalho é proposto para documentos XML orientados a dados (*data centric*). Documentos XML orientados a dados em geral são exportados de bancos de dados relacionais ou outros dados estruturados, no qual os nomes dos nodos tipicamente descrevem semântica, ao contrário dos documentos XML orientados a documento (*document centric*), no qual os nodos tipicamente descrevem formatação (Liu e Chen 2010).

O restante do texto encontra-se organizado da seguinte forma: o capítulo 2, *Definições Conceituais*, introduz os conceitos importantes que servem como base para o X-Spread 2. Os conceitos apresentados abordam principalmente documentos semi-estruturados e a evolução de esquemas XML.

O capítulo 3, *X-Spread 1*, descreve o trabalho especificado em (Silveira 2007), destacando suas características, arquitetura e objetivos. Apesar do protótipo X-Spread 1 não ser operacional e abordar o controle e a propagação da evolução de esquemas de uma maneira automática, algumas de suas ideias e algoritmos especificados foram incorporados e utilizados na elaboração deste trabalho.

O capítulo 4, *X-Spread 2*, define uma ferramenta operacional que propaga a evolução de esquemas XML para um conjunto de documentos XML. Neste capítulo é especificado o objetivo, o funcionamento e a arquitetura do X-Spread 2, destacando a especificação de como é realizada a adaptação de um conjunto de documentos XML a partir da detecção de diferenças entre dois esquemas XML.

Finalmente, o capítulo 5, *Conclusões*, encerra o trabalho com as considerações finais, revisando as contribuições e apresentando possibilidades de trabalhos futuros.

2 DEFINIÇÕES CONCEITUAIS

Este capítulo apresenta os principais conceitos que servem de base para a ferramenta implementada nesse trabalho, visando delimitar o escopo do problema abordado. Primeiramente, são apresentados os conceitos de XML e esquemas XML. Em seguida, o principal problema tratado neste trabalho é definido: evolução de esquemas e evolução de esquemas XML. Por fim, são definidos os processos de validação e adaptação de instâncias, que representam o processo que soluciona o problema abordado.

2.1 Documentos semi-estruturados

O XML (Extensible Markup Language) é um formato de texto simples e muito flexível para geração de documentos com dados organizados de forma hierárquica. Definido pela W3C (World Wide Web Consortium) e derivado do SGML (Standard Generalized Markup Language, ISO 8895) o XML inicialmente foi criado para suprir aos desafios das publicações eletrônicas em larga escala. Porém, atualmente, desempenha um papel cada vez mais importante, sendo utilizado para diversos fins, como, armazenamento de dados e na troca de uma ampla variedade de dados dentro e fora da Web, devido principalmente a sua portabilidade, ou seja, a sua independência das plataformas de hardware e software.

Os esquemas XML são estruturas que tem por objetivo restringir os valores e o formato presentes em documentos XML. Ao contrário dos bancos de dados relacionais, um esquema XML não possui a informação da quantidade de documentos XML que o referencia. O conjunto composto por um esquema XML e os documentos XML que o referenciam possui um aspecto distribuído, pois documentos XML podem referenciar esquemas XML disponibilizados em uma rede de dados. Esse conjunto composto por um esquema XML e documentos XML que o referenciam é nomeado de base de

dados semi-estruturados (COHEN, 1999). O DTD (*Data Type Definition*) e o *XML Schema* são, atualmente, os dois principais formatos utilizados para a representação de esquemas XML. Esses formatos possuem uma função semelhante, porém suas capacidades são diferentes.

O DTD é caracterizado por definir hierarquicamente a estrutura de documentos XML, através de uma lista de elementos e atributos. Como um esquema DTD permite a definição de valores contidos nos atributos dos elementos de um documento XML, o DTD permite uma estruturação básica da informação. O DTD pode ser declarado internamente no documento XML ou pode ser uma referência externa. A representação de um esquema DTD não é realizada no formato XML, logo ele não é considerado um documento XML.

O XML *Schema* é caracterizado como uma alternativa e uma evolução do esquema DTD. O XML *Schema* também define hierarquicamente a estrutura de documentos XML, porém, possui um maior poder de expressividade que o DTD, além de admitir a validação de atributos e valores de elementos dos documentos XML utilizando a especificação de expressões regulares. Diferentemente do DTD, o XML *Schema* é representado no formato XML.

2.2 Evolução de esquemas

Uma aplicação pode sofrer transformações e evoluir ao longo do tempo por diversos motivos. As bases de dados utilizadas por essas aplicações estão sujeitas a este processo de evolução. Consequentemente, em uma base de dados semi-estruturados o esquema e os documentos XML podem evoluir.

A evolução de um esquema pode prejudicar a integridade de uma base de dados e a integridade do relacionamento entre a aplicação e a base de dados acessada. Uma base de dados pode se tornar inconsistente devido a evolução de esquemas quando alterações nas estruturas do esquema são incompatíveis com os dados já existentes. Por exemplo, quando o nome de uma estrutura de um esquema é alterado, essa operação pode acarretar a inconsistência das instâncias existentes na base de dados. A relação de integridade entre um esquema de dados e as aplicações também pode ser comprometida quando o esquema torna-se incompatível com as representações internas desse artefato. Por exemplo, quando o tipo de um

atributo de uma estrutura de um esquema é alterado para um novo tipo que torne as instâncias da aplicação incompatíveis com esse esquema.

2.3 Evolução de esquemas XML

Atualmente é crescente o número de aplicações que utilizam bases de dados semi-estruturados. Em base de dados semi-estruturados as instâncias podem se encontrar distribuídas em diferentes pontos de uma rede de dados e diferentemente das bases de dados relacionais, são as instâncias que referenciam aos esquemas que possuem as definições de suas estruturas internas. Consequentemente, os esquemas semi-estruturados não possuem a informação da quantidade de instâncias que os referenciam.

Diferentemente das bases de dados relacionais, não existe um SGBD para dados semi-estruturados. Logo, se uma estrutura de dados do esquema XML é alterada, é necessária a propagação desta modificação aos documentos XML existentes com o objetivo de garantir a integridade da base de dados. No caso de bancos de dados relacionais, alterações que colocam a base de dados em um estado inconsistente são bloqueadas pelo SGBD. Como bases de dados semi-estruturadas usualmente não possuem um SGBD, quaisquer alterações no esquema são permitidas, podendo tornar as bases de dados inconsistentes.

Ao contrário de bases de dados orientadas a objetos e objeto-relacionais, os esquemas XML não associam métodos a estruturas de dados definidas nos esquemas. Consequentemente, a evolução de esquemas XML não se refere a métodos, e sim ao conjunto de aplicações que opera com instâncias desses esquemas. Logo, essas aplicações precisam ser sinalizadas de algum modo sobre alterações na estrutura dos dados que podem levar as aplicações a estados inconsistentes.

O conjunto de operações (H. Su, D. Kramer, L. Chen, K. Claypool e E. A. Rundensteiner, 2001) que podem ser realizadas em esquemas XML é grande. Por exemplo, podem ser realizadas operações como:

- adição de um novo elemento;
- exclusão de um elemento;
- alteração do nome de um elemento;

- alteração do tipo de dado de um elemento;
- adição de um novo atributo;
- exclusão de um atributo;
- alteração do nome de um atributo;
- alteração do tipo de dado de um atributo;
- alteração da cardinalidade de um elemento;
- movimentação de um elemento na hierarquia de dados.

Considerando que qualquer esquema XML válido pode ser gerado após a execução de uma sequência de operações de modificações, é possível afirmar que, se forem respeitadas as referências a componentes já existentes no esquema XML, essas operações garantem a integridade interna de um esquema XML.

A classe de esquemas que pode ser representados pelo formato DTD é contida na classe de esquemas que pode ser representada pelo formato XML *Schema* (RAGHAVACHARI; SHMUELI,2004). Logo, é possível afirmar que o formato XML *Schema* é mais genérico, com maior poder de expressividade. Portanto, todas as operações de modificação de esquemas XML no formato DTD podem ser aplicadas ao formato XML *Schema*.

2.4 Revalidação de documentos XML

O processo de revalidação de documentos XML é o processo de validação de um documento XML frente ao esquema XML e que sofreu modificações. Logo, a revalidação permite identificar a validade desse documento XML após a evolução do esquema XML referenciado.

2.5 Adaptação de documentos XML

Em bases de dados relacionais, modificações nos esquemas relacionais são validadas pelo SGBD com o objetivo de garantir a integridade dessas bases.

Devido à ausência de um SGBD em bases de dados semi-estruturados, não há um mecanismo que identifique a validade das alterações realizadas em esquemas XML. Ainda, devido às características de distribuição e disjunção de

esquemas e documentos XML, modificações em esquemas XML podem levar o conjunto formado por esquema e documentos XML que referenciam este esquema a um estado inconsistente.

Portanto, após a realização de uma alteração em um esquema XML, é necessário propagar essa modificação para todos documentos XML que referenciam esse esquema, com o objetivo de manter essa base de dados semi-estruturados consistente. O processo de adaptação de documentos XML consiste em um conjunto de modificações realizadas sobre os documentos XML para que esses documentos XML sejam válidos para nova versão do esquema XML referenciado. Esse conjunto de modificações são baseadas nas diferenças detectadas entre a versão atual e a versão anterior do esquema XML.

3 X-SPREAD

X-Spread (Silveira 2007, Silveira e Galante 2008) é uma uma abordagem para a propagação automática da evolução de esquemas para documentos XML. O principal objetivo do X-Spread é re-estabelecer a consistência de um conjunto de documentos XML quando da evolução do esquema associado a esses documentos, através de adaptações nesses documentos tornados inválidos pelas modificações sofridas pelo esquema. Foi desenvolvido um protótipo preliminar simplesmente para testar as principais premissas especificadas no X-Spread.

Este capítulo apresenta uma revisão das principais características especificadas no X-Spread com vistas a implementação de uma ferramenta operacional da abordagem proposta no X-Spread. Para clareza do texto, nomeia-se a partir de agora X-Spread 1 a abordagem especificada em (Silveira 2007, Silveira e Galante 2008) e o protótipo não operacional e X-Spread 2 a ferramenta operacional especificado e implementado neste TCC.

3.1 X-Spread: uma visão geral

X-Spread 1 é a especificação de uma abordagem automático para a propagação da evolução de esquemas para documentos XML. A arquitetura do X-Spread 1 e o fluxo geral entre os componentes é ilustrada na Figura 3.1.

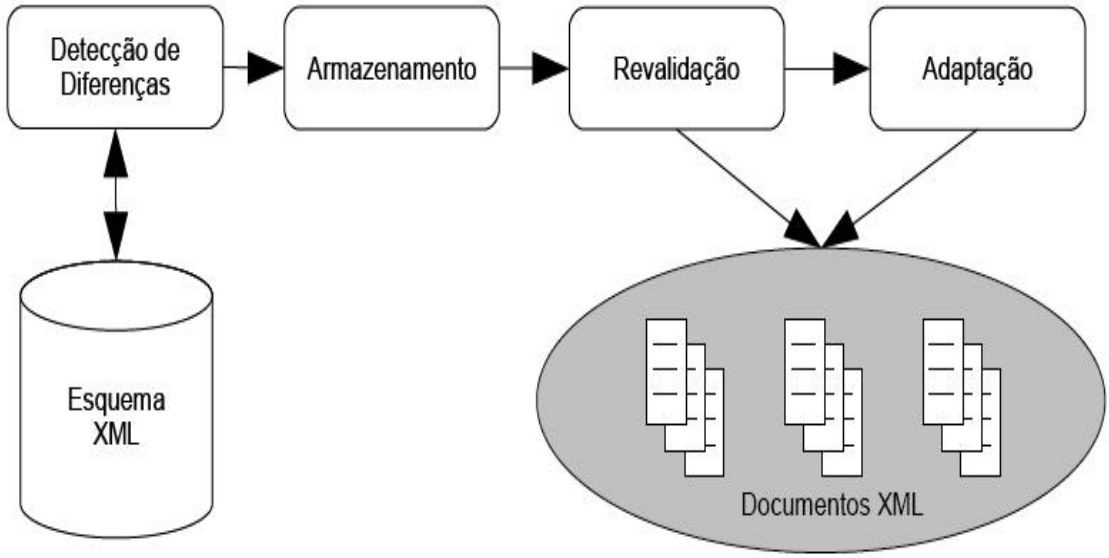

Figura 3.1: Arquitetura e fluxo geral do X-Spread 1.

3.1.1 Módulo de Detecção de Diferenças

O objetivo do módulo de detecção de diferenças é monitorar as atualizações em esquemas XML configurados pelo administrador da base de dados e identificar quais modificações foram realizadas nesses esquemas. O monitoramento dos esquemas é feito através de consultas periódicas realizadas em intervalos definidos pelo gerenciador do banco de dados. Na primeira consulta a um esquema XML, ele será armazenado na base de dados do X-Spread 1, assim como o valor resultante da aplicação de funções de dispersão ao esquema, como o MD5.

Nas consultas subsequentes, o conteúdo completo do esquema XML é recuperado e novamente é aplicada a função de dispersão ao esquema. Caso esse novo valor obtido seja diferente daquele valor já armazenado na base de dados do X-Spread 1, essa versão do esquema e o valor MD5 correspondente são armazenados, dando início ao processo de detecção de diferenças.

As duas versões do esquema são comparadas através da aplicação de algoritmos de detecção de diferenças entre documentos XML e o X-Spread 1 realiza a normalização e o armazenamento das saídas. O XyDiff (Cobéna 2002) é o algoritmo de detecção de diferenças entre documentos XML utilizado no X-Spread 1. Na normalização, são identificadas as partículas afetadas pela modificação do esquema e outras informações dependendo tipo de modificação, como o antigo ancestral, o novo ancestral, o antigo valor e o novo

valor. Com a identificação dos itens afetados através da atualização do esquema, como partículas modificadas e seus ancestrais, é possível realizar um mapeamento desses itens para operações de modificação especificadas pelo X-Spread. Essas operações serão utilizadas posteriormente nas etapas de revalidação e adaptação dos documentos XML associados ao esquema do processamento atual.

3.1.2 Módulo de Armazenamento

O objetivo do módulo de armazenamento é armazenar configurações do sistema, versões e meta-dados de esquemas além de disponibilizar interfaces de armazenamento e consulta dessas informações aos outros módulos do X-Spread 1. O administrador da base de dados poderá especificar o conjunto de esquemas XML que devem ser monitorados, o intervalo de tempo entre as consultas que verificam a existência de atualizações dos esquemas e definir quais as operações de adaptação que devem ser ignoradas pelo sistema.

3.1.3 Módulo de Revalidação

O objetivo do módulo de revalidação é revalidar um conjunto de documentos XML em relação a um conjunto de modificações sofridas pelos esquemas XML referenciados. Primeiro, é analisada a necessidade de cada documento XML ser submetido ao processo de revalidação, através da verificação de inconsistência desses documentos causada pelas modificações do esquema. Se o documento XML está inconsistente em relação à última versão do esquema, o X-Spread 1 armazena as informações referentes às modificações sofridas pelo esquema que causaram a inconsistência e submete esse documento ao processo de adaptação com o objetivo de torná-lo compatível com a última versão do esquema.

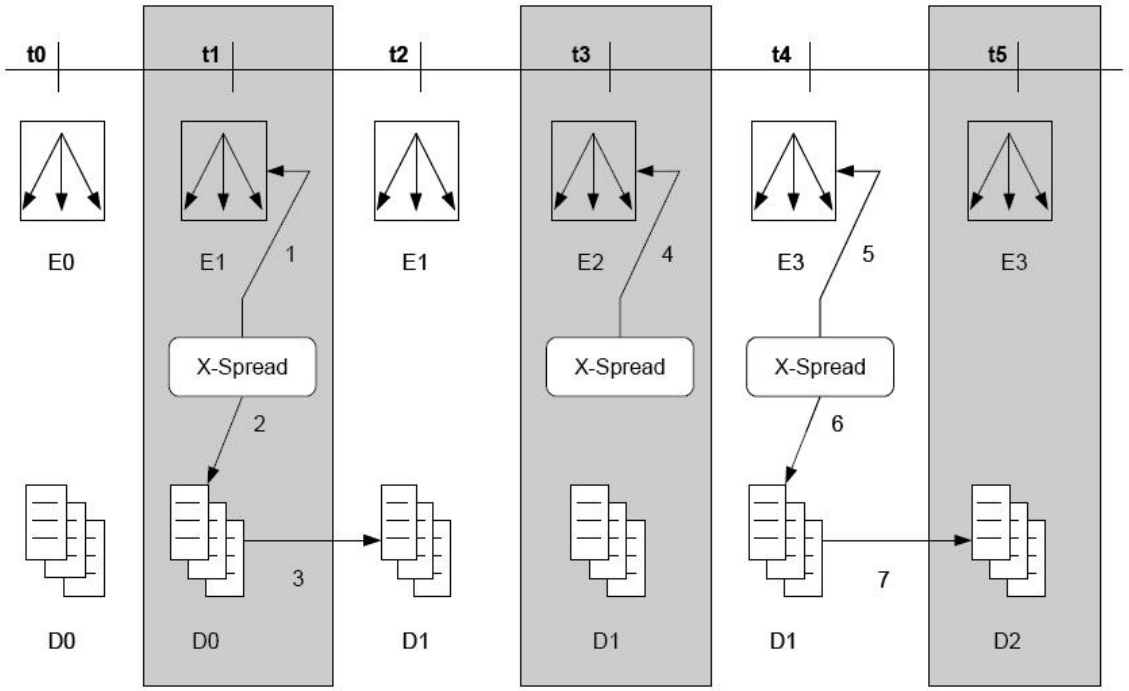

Figura 3.2: Fluxo de revalidação e adaptação de documentos XML.

A figura 3.2 ilustra o fluxo de evolução de um esquema e as ações dos processos de revalidação e adaptação sobre o conjunto de documentos XML que referenciam esse esquema. Nessa figura, são representados seis momentos de tempos distintos na evolução de um esquema:

- t0 – o esquema E0 e o conjunto dos documentos XML D0 consistentes e associados a esse esquema são monitorados pelo X-Spread 1;
- t1 – uma nova versão E1 do esquema E0 é gerada, correspondendo a última versão modificada do esquema E0. Após a identificação dessa nova versão E1 do esquema, o X-Spread 1 executa diferentes ações:

1. detecção das diferenças entre os esquemas E0 e E1;
2. revalidação do conjunto de documentos XML D0 em relação às diferenças detectadas entre os esquemas E0 e E1;
3. geração do conjunto de documentos XML D1 a partir das adaptações realizadas no conjunto de documentos D0;

- t2 – nesse momento é encerrada a adaptação do conjunto de documentos XML D0, gerando o conjunto de documentos XML D1 compatíveis com o esquema E1;
- t3 – uma nova versão de esquema XML E2 é gerada a partir de modificações sobre o esquema E1. A detecção de diferenças entre os

esquemas E1 e E2 é representada pela ação 4. Nenhuma adaptação no conjunto de documentos XML D1 é necessária, pois não foi detectada nenhuma modificação no esquema E2 que torne algum documento do conjunto D1 inconsistente.

- t4 – uma nova versão de esquema XML E3 é criada a partir de alterações sobre o esquema E2. Após a identificação dessa nova versão E3 do esquema, o X-Spread 1 executa as seguintes ações:

1. detecção de diferenças entre os esquemas XML E2 e E3;
2. revalidação do conjunto de documentos XML D1 a partir das diferenças detectadas entre os esquemas E2 e E3;
3. geração do conjunto de documentos XML D2 após as adaptações realizadas no conjunto de documentos D1;

- t5 – os documentos XML do conjunto D2 são compatíveis com o esquema XML E3.

3.1.4 Módulo de Adaptação

A finalidade do módulo de adaptação é transformar um conjunto de documentos XML, inicialmente inconsistentes em relação a um esquema XML, em um conjunto de documentos XML compatíveis a esse esquema XML. O módulo de adaptação recebe um conjunto de modificações realizadas, um esquema e um conjunto de documentos XML que serão submetidos ao processo de adaptação com o objetivo de transformá-los em documentos XML compatíveis a última versão desse esquema.

3.2 Protótipo do X-Spread 1

O Protótipo do X-Spread 1 foi construído a partir da especificação da arquitetura, definida na seção 3.2.1. O objetivo dessa implementação foi simplesmente testar as principais premissas especificadas na abordagem proposta Silveira e não implementar um protótipo operacional. As seguintes tecnologias, bibliotecas e aplicativos foram utilizados na construção desse protótipo:

- Sun Java 5 (SUN MICROSYSTEMS, 2004);

- jaxen (CODEHAUS, 2004) - biblioteca com suporte a pesquisas XPath em Java, utilizadas nos módulos de adaptação e revalidação;
- MySQL (AB, 2007) – banco de dados utilizado internamente no X-Spread 1, acessado pelo módulo de armazenamento.

O X-Spread 1 consulta periodicamente um conjunto de documentos e esquemas configurados pelo administrador do banco de dados, através de execuções de uma *thread* específica. Esta busca também é realizada quando a configuração de esquemas e documentos monitorados pelo X-Spread 1 sofre alterações, resultando na atualização do *cache* com as configurações mais recentes, e quando a consulta a esquemas é iniciada manualmente pelo gerenciador da base de dados. Os principais componentes do protótipo X-Spread 1 e os fluxos de comunicações existentes entre esses componentes são apresentados na figura 3.3.

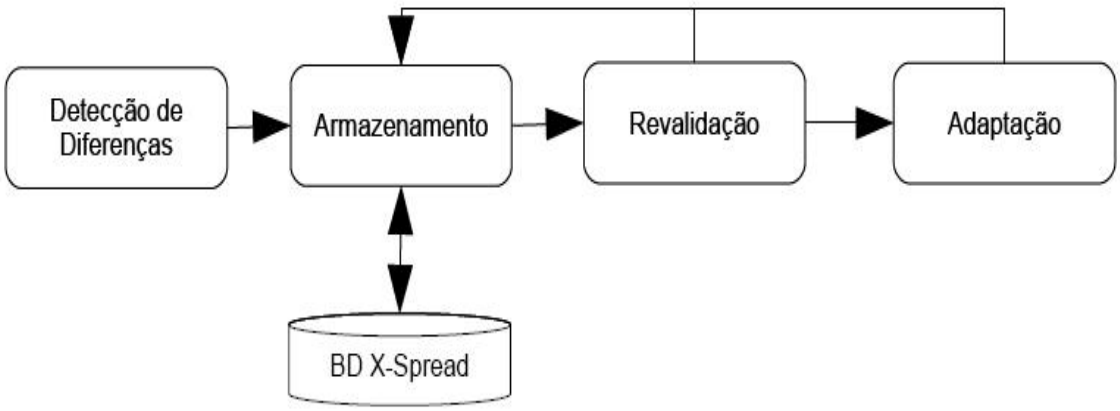

Figura 3.3: Arquitetura do protótipo X-Spread e o fluxo entre os componentes.

3.2.1 Arquitetura

A figura 3.4 ilustra a arquitetura do X-Spread 1 e o fluxo de chamadas possíveis entre os módulos na revalidação de um conjunto de documentos XML. Para fins de simplicidade, o banco de dados do X-Spread 1 é encapsulado pelo módulo de armazenamento. As ações realizadas pelo X-Spread 1, conforme numeração da figura, são:

1. o *cache* interno do X-Spread 1 é atualizado pelo módulo de armazenamento com as informações de esquemas e documentos monitorados, de acordo com a especificação do gerenciador da base de dados;

2. os esquemas a serem monitorados são consultados no *cache* do X-Spread 1 pelo módulo de detecção de diferenças;
3. o módulo de detecção de diferenças realiza uma busca das versões mais recentes dos esquemas presentes no *cache* do X-Spread 1 no módulo de armazenamento;
4. o módulo de detecção de diferenças verifica a existência de versões mais recentes dos esquemas e realiza uma comparação com as versões armazenadas no banco de dados do X-Spread 1;
5. o módulo de armazenamento recebe do módulo de detecção de diferenças os esquemas que possuem uma versão mais recente disponível, assim como as modificações realizadas, e armazena essas informações no banco de dados do X-Spread 1;
6. o módulo de revalidação recebe alterações desestabilizadoras enviadas pelo módulo de armazenamento;
7. o módulo de revalidação realiza uma consulta no cache do X-Spread 1 para identificar quais documentos devem ser revalidados;
8. o módulo de revalidação busca esses documentos XML identificados no cache do X-Spread 1 e os processa;
9. o módulo de revalidação identifica os documentos inválidos e os envia ao processamento do módulo de adaptação;
10. o módulo de adaptação realiza a adaptação desses documentos inválidos e armazena os documentos adaptados nos seus respectivos sistemas de arquivos, disponibilizando a nova versão adaptada destes artefatos para os demais usuários do banco de dados.

No primeiro passo é realizada a preparação do ambiente de execução do X-Spread, com atualização do *cache* interno. Periodicamente são realizadas as ações compreendidas pelos passos 2 a 4. Se nenhuma nova versão dos esquemas é identificada, o X-Spread 1 não executa nenhuma ação adicional. O passo 5 é executado caso uma atualização de esquema seja identificada. As ações compreendidas pelos passos 6 a 8 são executadas somente se alguma modificação desestabilizadora tenha sido realizada no esquema. Finalmente, a adaptação, compreendida pelos passos 9 e 10 é realizada com o objetivo de deixar o banco de dados em um estado íntegro.

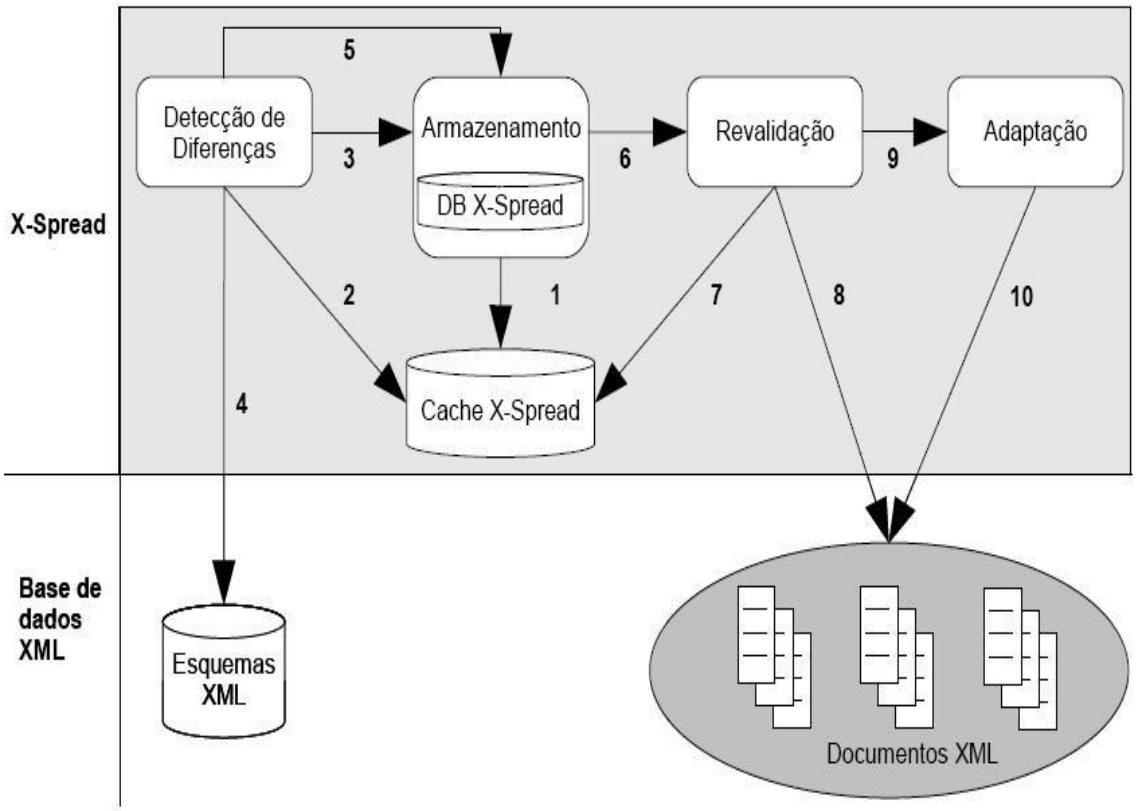

Figura 3.4: Arquitetura do protótipo X-Spread 1 e o fluxo de ações entre os componentes.

3.2.2 Implementação do X-Spread 1

O protótipo não operacional foi desenvolvido para testar as principais premissas especificadas no X-Spread. Consequentemente, algoritmos e funcionalidades foram implementados visando o funcionamento automático, periódico e a utilização de um banco de dados interno para a adaptação dos documentos XML. Destacando a implementação de:

- Diversas classes e algoritmos para leitura e manipulação de documentos e esquemas XML;
- Armazenamento dos esquemas e documentos XML que deveriam ser monitorados no banco de dados interno;
- Execução do algoritmo XyDiff sobre dois esquemas XML;
- Armazenamento no banco de dados interno das diferenças detectadas entre dois esquemas XML;
- Esboço da detecção das operações a serem realizadas na adaptação dos documentos XML;

Diferentemente do X-Spread 2, especificado na seção 4, o protótipo X-Spread 1 não possuía uma interface e utilizava um banco de dados interno para o armazenamento de dados necessários para sua execução automática e periódica. Ainda, o X-Spread 1 não era operacional, ou seja, não realizava a adaptação de documentos XML a partir das diferenças detectadas entre dois esquemas XML, devido, principalmente, à ausência da detecção de quais operações deveriam ser realizadas sobre o conjunto de documentos XML para torná-lo novamente consistente. Porém, diversos algoritmos e idéias do X-Spread 1 foram incorporados na elaboração da ferramenta operacional especificada nesse trabalho, destacando a detecção das operações a serem realizadas sobre os documentos XML como uma das principais inovações implementadas.

4 X-SPREAD 2

O X-Spread 2 é uma ferramenta operacional que realiza a propagação da evolução de esquemas XML para um conjunto de documentos XML, ou seja, realiza a adaptação de um conjunto de documentos XML a partir das diferenças detectadas entre dois esquemas XML. O X-Spread 2 possibilita ao usuário informar, através de sua interface, dois esquemas XML e um conjunto de documentos XML, inicialmente válidos para o primeiro esquema XML e possivelmente inválidos para o segundo esquema XML. O objetivo do X-Spread 2 é realizar a adaptação desse conjunto de documentos XML para torná-lo válido para o segundo esquema XML e disponibilizar o novo conjunto de documentos XML gerados para o usuário. Este trabalho está baseado no protótipo não operacional do X-Spread 1, especificado em (Silveira 2007, Silveira e Galante 2008), sendo incorporados novos algoritmos e idéias para a adaptação de documentos XML.

4.1 Arquitetura

O X-Spread 2 é uma aplicação Web desenvolvida a partir das seguintes tecnologias e APIs:

- JAVA EE 6
- JavaServer Faces 1.2
- Apache Tomcat 6.0
- DOM
- JDOM
- jaxen
- XyDiff

A aplicação possui sua interface desenvolvida a partir da tecnologia JavaServerFaces 1.2 e possui o Apache Tomcat 6.0 como servidor. As APIs

DOM, JDOM, jaxen e XyDiff foram utilizadas para manipular e validar documentos e esquemas XML durante todo o processo de adaptação dos documentos XML do X-Spread 2.

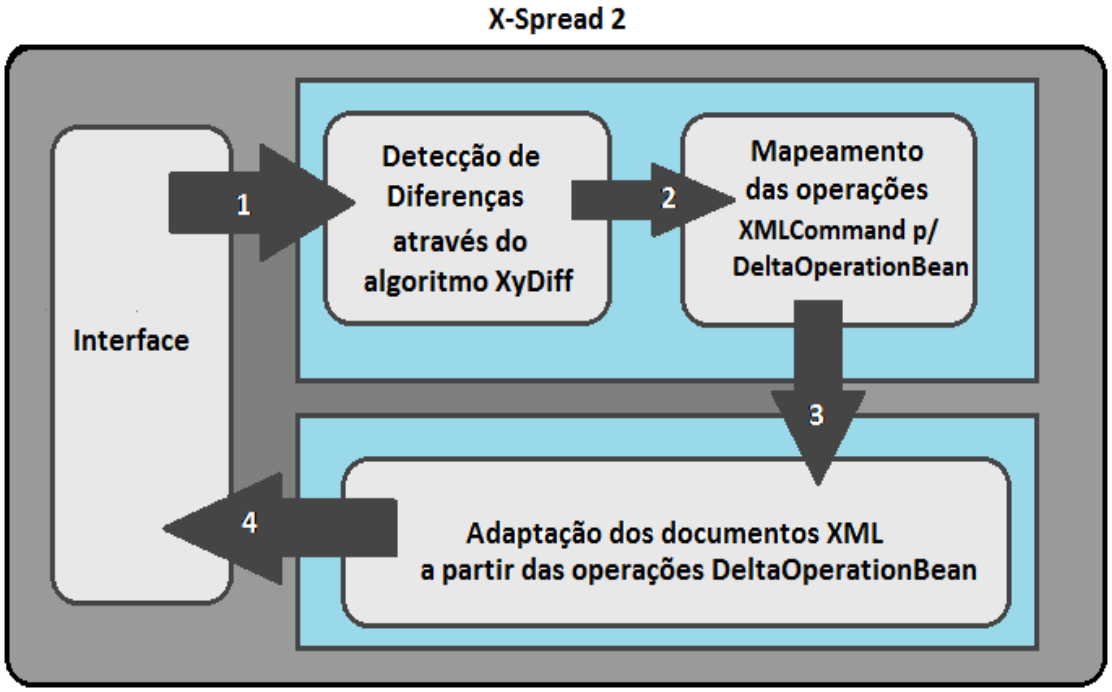

Figura 4.1: Arquitetura e fluxo geral do processo de adaptação de documentos XML.

Conforme a arquitetura e o fluxo geral da adaptação de documentos XML realizados pelo X-Spread 2, ilustrados na figura 4.1, são realizados os seguintes passos:

1. através da interface, o usuário informa os esquemas XML A e B, documentos XML que referenciam e são válidos para o esquema XML A e outras informações necessárias para a adaptação desses documentos XML;

2. realiza-se a detecção de diferenças através da execução do algoritmo *XyDiff* sobre os dois esquemas XML A e B informados pelo usuário. A saída do algoritmo *XyDiff* é um conjunto de operações *XMLCommand* que transformam o esquema XML A no esquema XML B;

3. realiza-se o mapeamento do conjunto de operações *XMLCommand* para um conjunto de operações *DeltaOperationBean*, que são operações que transformam cada documento XML válido para o esquema XML B;

4. para cada documento XML, realiza-se um conjunto de modificações a partir do conjunto de operações *DeltaOperationBean* e cria-se um novo documento XML. Posteriormente, realiza-se uma validação desse novo documento XML para o esquema XML B, informando ao usuário o sucesso ou a falha no processo de adaptação de cada documento XML.

Os passos listados anteriormente serão detalhados durante o restante desta seção.

4.2 Processo de adaptação de documentos XML

O processo de adaptação de documentos XML tem como objetivo detectar as diferenças entre dois esquemas XML e aplicar as transformações necessárias sobre um conjunto de documentos XML, inicialmente válidos para o primeiro esquema XML e possivelmente inválidos para o segundo esquema XML, para tornar esse conjunto de documentos XML compatível com o segundo esquema XML.

A figura 4.2 ilustra o formulário da adaptação de documentos XML do X-Spread 2. Nesse formulário, o usuário deve obrigatoriamente informar:

- a url do esquema XML A;
- a url do esquema XML B;
- o diretório onde os documentos XML serão gerados;
- pelo menos uma url de um documento XML válido para o esquema XML A e que deve ser adaptado, com o objetivo de torná-lo válido para o esquema XML B;

Ao clicar no botão *Adaptar documentos XML*, X-Spread 2 irá detectar as diferenças existentes entre o esquema XML A e o esquema XML B e realizar as transformações necessárias para que cada documento XML informado no formulário seja válido para o esquema XML B.

Figura 4.2: Formulário da adaptação de documentos XML.

Se o usuário informou corretamente os dados, o X-Spread 2 realizará as transformações necessárias sobre o conjunto de documentos XML a partir das diferenças detectadas entre os esquemas XML A e B. Para cada documento XML do conjunto de documentos XML, será criado no diretório informado pelo usuário, um novo documento XML com o mesmo nome acrescentado pelo sufixo default *_novo* do X-Spread 2 ou pelo sufixo informado pelo usuário. Se o novo documento XML foi criado com sucesso, ou seja, se o documento é válido para o esquema B, uma mensagem de sucesso será exibida ao usuário conforme ilustra a figura 4.3. Se algum erro aconteceu na geração do novo documento XML uma mensagem de erro será exibida ao usuário.

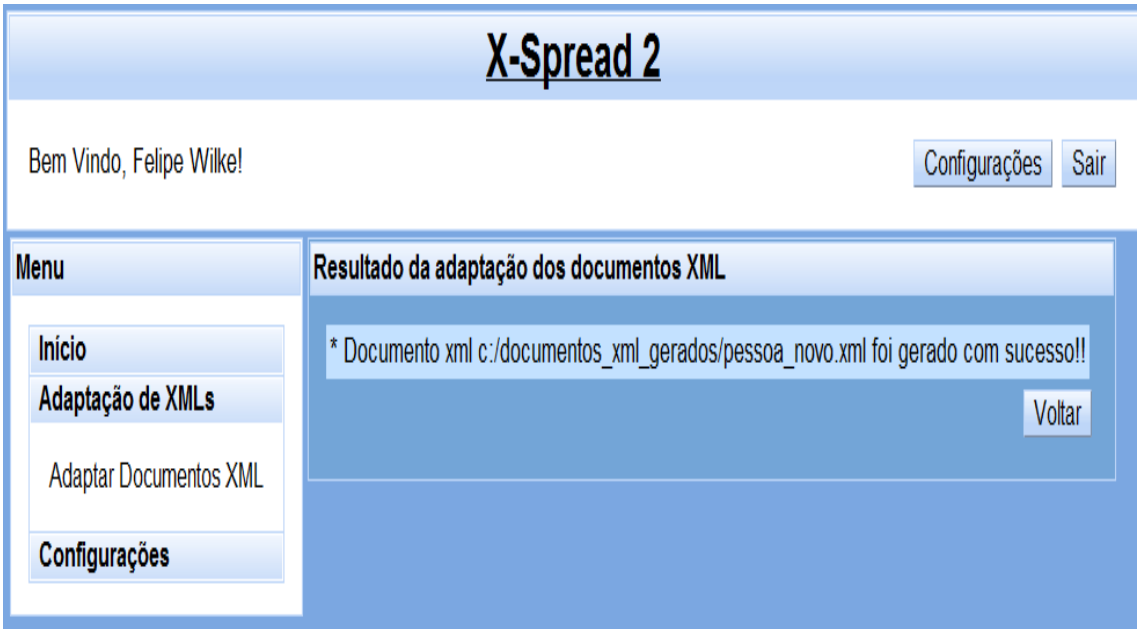

Figura 4.3: Mensagem de sucesso do processo de aptação de um documento XML.

Caso algum campo não tenha sido informado corretamente, o X-Spread 2 exibirá uma mensagem alerta na parte superior do formulário, conforme a figura 4.4, informando qual informação não foi fornecida corretamente.

Conforme ilustrado na figura 4.5, referente ao formulário expandido de adaptação de documentos XML, o usuário poderá adicionar vários documentos XML clicando no botão *Adicionar +1 Documento* e limpar o valor de todos os campos preenchidos clicando no botão *Limpar*. Além disso, ao clicar no botão *Configurações*, o formulário será expandido e o usuário poderá informar o valor das seguintes configurações:

- sufixo dos novos documentos XML;
- valor default para novo elemento do tipo 'string';
- valor default para novo elemento do tipo 'int';
- valor default para novo elemento do tipo 'boolean';
- valor default para novo elemento do tipo 'date';
- valor default para novo elemento do tipo 'dateTime';
- valor default para novo elemento do tipo 'double';
- valor default para novo elemento do tipo 'float';
- valor default para novo elemento do tipo 'hexBinary';
- valor default para novo elemento do tipo 'time'.

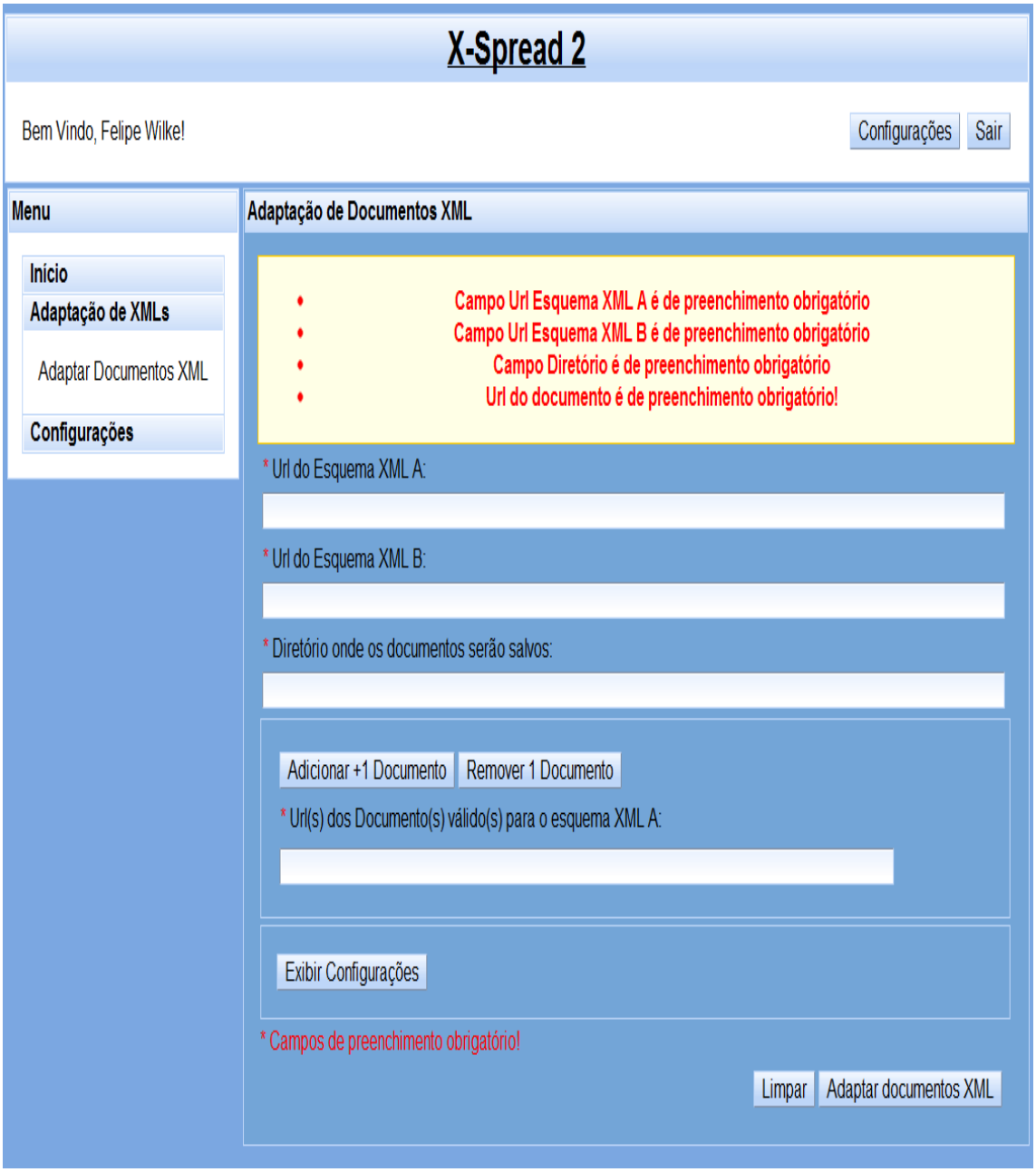

Figura 4.4: Mensagem de alerta do formulário de adaptação de documentos XML.

As configurações citadas anteriormente não são obrigatórias, logo, se o usuário não preencher alguma dessas configurações, o X-Spread 2 utilizará os valores default dessas configurações:

- sufixo dos novos documentos XML: *_novo*;
- valor default para novo elemento do tipo 'string': ;
- valor default para novo elemento do tipo 'int': *0*;
- valor default para novo elemento do tipo 'boolean': *false*

- valor default para novo elemento do tipo 'date': ;
- valor default para novo elemento do tipo 'dateTime': ;
- valor default para novo elemento do tipo 'double': *0.0*;
- valor default para novo elemento do tipo 'float': *0.0*;
- valor default para novo elemento do tipo 'hexBinary': ;
- valor default para novo elemento do tipo 'time': *00:00:00*;

Figura 4.5: Formulário expandido da adaptação de documentos XML.

4.2.1 Detecção de diferenças entre os esquemas XML A e B

Na etapa de detecção de diferenças entre os esquemas XML A e B, primeiramente é realizado um teste, no qual é verificado se o conteúdo do esquema XML A é diferente do conteúdo do esquema XML B, através da aplicação da função de dispersão *MD5,* com o auxílio de uma instância *java.security.MessageDigest.getInstance("MD5"),* sobre o conteúdo dos esquemas XML A e B. Se os conteúdos dos esquemas XML A e B são diferentes, são gerados dois arquivos temporários com os conteúdos dos esquemas XML A e B.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
<xs:element name="pessoa">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="nome" type="xs:string"/>
      <xs:element name="idade" type="xs:int"/>
      <xs:element name="endereco" type="tipoEnderecoBrasil"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:complexType name="tipoEnderecoBrasil">
    <xs:sequence>
      <xs:element name="tipo" type="xs:string"/>
      <xs:element name="nomeEndereco"   type="xs:string"/>
      <xs:element name="complemento"   type="xs:string"/>
      <xs:element name="cidade"   type="xs:string"/>
      <xs:element name="estado"  type="xs:string"/>
      <xs:element name="cep"    type="xs:decimal"/>
    </xs:sequence>
</xs:complexType>
</xs:schema>
```

Figura 4.6: Esquema XML A pessoa.xsd.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
<xs:element name="pessoa">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="nome" type="xs:string"/>
      <xs:element name="sobrenome" type="xs:string"/>
      <xs:element name="idade" type="xs:int"/>
      <xs:element name="endereco" type="tipoEnderecoBrasil"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:complexType name="tipoEnderecoBrasil">
    <xs:sequence>
      <xs:element name="tipo" type="xs:string"/>
      <xs:element name="nomeEndereco"   type="xs:string"/>
      <xs:element name="complemento"   type="xs:string"/>
      <xs:element name="cidade"   type="xs:string"/>
      <xs:element name="estado"  type="xs:string"/>
      <xs:element name="cep"    type="xs:decimal"/>
    </xs:sequence>
</xs:complexType>
</xs:schema>
```

Figura 4.7: Esquema XML B pessoaV2.xsd.

O algoritmo *XyDiff* (Cobéna 2002), desenvolvido pela equipe GEMO no INRIA (Instituto Nacional de Pesquisa em Ciência da Computação e Controle da França), possui o objetivo de detectar diferenças entre documentos XML. Logo, pode ser utilizado para detectar as diferenças entre dois esquemas XML. O próximo passo realizado é obter uma instância da classe *fr.loria.ecoo.so6.xml.xydiff.XyDiff* passando como parâmetros os conteúdos dos esquemas XML A e B. Posteriormente, é executado o método *XyDiff.diff()* dessa instância, que aplica o algoritmo de detecção de diferenças sobre os esquemas XML A e B, retornando uma instância da classe *fr.loria.ecoo.so6.xml.xydiff.DeltaConstructor* populada com uma lista de comandos *fr.loria.ecoo.so6.xml.xydiff.XMLCommand*. Essa lista de comandos *XMLCommand* corresponde as operações que o algoritmo *XyDiff* julga necessárias para transformar o esquema XML A em esquema XML B.

```
DeleteNode: <xs:element name="endereco" type="tipoEnderecoBrasil"/> path 0:0:0:0:0:2
DeleteNode: <xs:element name="idade" type="xs:int"/> path 0:0:0:0:0:1
DeleteNode: <xs:element name="nome" type="xs:string"/> path 0:0:0:0:0:0
InsertNode: <xs:element name="nome" type="xs:string"/> path 0:0:0:0:0:0
InsertNode: <xs:element name="sobrenome" type="xs:string"/> path 0:0:0:0:0:1
InsertNode: <xs:element name="idade" type="xs:int"/> path 0:0:0:0:0:2
InsertNode: <xs:element name="endereco" type="tipoEnderecoBrasil"/> path 0:0:0:0:0:3
```

Figura 4.8: Saída do XyDiff: Lista de operações para transformar o esquema XML A no esquema XML B.

A figura 4.8 ilustra a saída do algoritmo XyDiff executado sobre o esquema XML A pessoa.xsd, ilustrado na figura 4.6, e o esquema XML B pessoaV2.xsd, ilustrado na figura 4.7. Essa saída corresponde a uma lista de comandos *fr.loria.ecoo.so6.xml.xydiff.XMLCommand*. Logo, de acordo com o algoritmo *XyDiff*, para transformar o esquema XML A em esquema XML B, devem ser realizadas as operações nessa ordem:

- remoção do elemento *endereco*, que se encontra na posição 0:0:0:0:0:2;
- remoção do elemento *idade*, que se encontra na posição 0:0:0:0:0:1;
- remoção do elemento *nome*, que se encontra na posição 0:0:0:0:0:0;
- inserção do elemento *nome*, do tipo *xs:string*, na posição 0:0:0:0:0:0;
- inserção do elemento *sobrenome*, do tipo *xs:string*, na posição 0:0:0:0:0:1;
- inserção do elemento *idade*, do tipo *xs:int*, que se encontra na posição 0:0:0:0:0:2;
- inserção do elemento *endereco*, do tipo *tipoEnderecoBrasil*, que se encontra na posição 0:0:0:0:0:3.

O cálculo do algoritmo *XyDiff* para determinar as posições de cada elemento é realizado a partir do elemento raiz até o elemento em questão, conforme ilustrado na figura 4.9.

Figura 4.9: Cálculo do XyDiff para as posições dos elementos.

Nesse momento do processo de adaptação de documentos XML, o X-Spread 2 possui um conjunto de operações necessárias para realizar a transformação do esquema XML A no esquema XML B. Porém, o objetivo é adaptar cada documento XML do conjunto de documentos informados pelo usuário para que esse documento XML seja válido para o esquema XML B. Logo, o X-Spread 2 deve obter as operações necessárias para realizar a adaptação de cada documento XML.

4.2.2 Detecção das operações realizadas sobre os documentos XML

O conjunto operações *OX* que serão realizadas sobre os documentos XML são obtidas a partir do conjunto de operações *OE* necessárias para transformar o esquema XML A no esquema XML B. O primeiro passo realizado para obter esse conjuntos de operações *OX* é obter uma instância da classe *org.jdom.Document* a partir da instância da classe *DeltaConstructor,* obtida na

seção 4.1.1, que contém as operações que transformam o esquema XML A em B.

Para cada *fr.loria.ecoo.so6.xml.xydiff.XMLCommand*, da instância *DeltaConstructor*, cria-se uma instância *com.xspread.util.DeltaOperationBean* com as informações necessárias para realizar uma operação de *Remoção de Elemento*, *Adição de Elemento*, *Alteração do Nome de Elemento* ou *Alteração do Tipo de Elemento*. Cada *DeltaOperationBean* possui os campos:

- int operationId;
- String initialValue;
- String finalValue;
- String name;
- String type;
- String parentName;

Para a operação de *Remoção de Elemento* são obtidos o nome do elemento e o nome do pai do elemento a ser removido. Respectivamente são populados os campos *name* e *parentName* do *DeltaOperationBean* que representa essa operação de *Remoção de Elemento,* identificada através do campo *operationId*.

Para a operação de *Adição de Elemento* são obtidos o nome, o tipo e o nome do pai do elemento a ser removido. Respectivamente são populados os campos *name, type* e *parentName* do *DeltaOperationBean* que representa essa operação de *Adição de Elemento*.

Para a operação de *Alteração do Nome de Elemento* são obtidos o nome inicial e o nome final do elemento a ser atualizado. Respectivamente são populados os campos *initialValue* e *finalValue* do *DeltaOperationBean* que representa essa operação de *Alteração do Nome de Elemento.*

Para a operação de *Alteração do Tipo de Elemento* são obtidos o tipo inicial, o tipo final e o nome do elemento a ser atualizado. Respectivamente são populados os campos *initialValue, finalValue* e *name* do *DeltaOperationBean* que representa essa operação de *Alteração do Tipo de Elemento.*

Conforme ilustrado na figura 4.10, no final dessa etapa obtemos uma lista de *com.xspread.util.DeltaOperationBean,* onde cada *DeltaOperationBean*

é uma operação a ser realizada para adaptar os documentos XML informados pelo usuário e torná-los válidos para o esquema XML B.

```
DeltaOperationBean [finalValue=null, initialValue=null, name=endereco,
 operationId=3, parentName=pessoa, type=tipoEnderecoBrasil]
DeltaOperationBean [finalValue=null, initialValue=null, name=idade,
 operationId=3, parentName=pessoa, type=xs:int]
DeltaOperationBean [finalValue=null, initialValue=null, name=nome,
 operationId=3, parentName=pessoa, type=xs:string]
DeltaOperationBean [finalValue=null, initialValue=null, name=nome,
 operationId=1, parentName=pessoa, type=xs:string]
DeltaOperationBean [finalValue=null, initialValue=null, name=sobrenome,
 operationId=1, parentName=pessoa, type=xs:string]
DeltaOperationBean [finalValue=null, initialValue=null, name=idade,
 operationId=1, parentName=pessoa, type=xs:int]
DeltaOperationBean [finalValue=null, initialValue=null, name=endereco,
 operationId=1, parentName=pessoa, type=tipoEnderecoBrasil]
```

Figura 4.10: Lista de DeltaOperationBean obtida a partir da lista de XMLCommand gerada para os esquemas XML pessoa.xsd e pessoaV2.xsd.

4.2.3 Adaptação de documentos XML

Nessa etapa é realizada a adaptação do conjunto de documentos XML informados pelo usuário a partir das operações *DeltaOperationBean* obtidas na seção 4.1.2. Ao final dessa etapa, os novos documentos XML gerados estarão disponíveis no diretório informado pelo usuário.

Para cada documento XML é realizado o mesmo procedimento. Primeiramente, uma instância da classe *org.w3c.dom.Document* é obtida através do método *javax.xml.parsers.DocumentBuilder.parse*, passando como parâmetro o documento XML. Depois, para cada operação *DeltaOperationBean* é verificado o seu tipo, pois, para cada tipo de operação, uma sequência de ações é realizada. Os seguintes tipos de operação são considerados no X-Spread 2:

- Remoção de Elemento;
- Adição de Elemento;
- Alteração do Nome de Elemento;

- Alteração do Tipo de Elemento.

Para o tipo de operação *Adição de Elemento* é realizada a inserção de um elemento *org.w3c.dom.Element* com o nome *DeltaOperationBean.getName()* . Primeiramente, verifica-se se algum elemento *org.w3c.dom.Element* com esse nome já foi removido anteriormente, ou seja, se encontra-se na lista auxiliar citada na operação de *Remoção de Elemento*. Se o elemento já foi removido anteriormente, então adiciona-se o elemento da lista de removidos como filho do seu elemento pai. Se o elemento não tinha sido removido anteriormente, então encontra-se o elemento pai através do nome *DeltaOperationBean.getParentName()* e adiciona-se um novo elemento filho com o nome *DeltaOperationBean.getName()* e com um valor correspondente ao tipo *DeltaOperationBean.getTypeElemento(),* que é o valor default do X-Spread 2 ou o valor informado pelo usuário no formulário de adaptação de documentos XML.

Para exemplificar esse tipo de operação, podemos considerar o esquema XML A como o esquema *pessoa.xsd*, ilustrado na figura 4.6, e o esquema XML B como o esquema *pessoaV2.xsd*, ilustrado na figura 4.7. Os quatro últimos *DeltaOperationBean* obtidos e ilustrados na figura 4.10 são referentes à operações de *Adição de Elemento.* Após as transformações realizadas sobre o documento XML pessoa.xml, ilustrado na figura 4.11, é criado um novo documento XML pessoa_V2.xml, ilustrado pela figura 4.12.

```
<?xml version="1.0" encoding="UTF-8"?>
<pessoa xsi:noNamespaceSchemaLocation="pessoa.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<nome>felipe</nome>
<idade>20</idade>
<endereco>
  <tipo>rua</tipo>
  <nomeEndereco>jose bonifacio</nomeEndereco>
  <complemento>23</complemento>
  <cidade>porto alegre</cidade>
  <estado>rs</estado>
  <cep>90050000</cep>
</endereco>
</pessoa>
```

Figura 4.11: Documento XML pessoa.xml.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<pessoa xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="c:/schemas/pessoaV2.xsd">

<nome>felipe</nome>
<sobrenome/>
<idade>20</idade>
<endereco>
  <tipo>rua</tipo>
  <nomeEndereco>jose bonifacio</nomeEndereco>
  <complemento>23</complemento>
  <cidade>porto alegre</cidade>
  <estado>rs</estado>
  <cep>90050000</cep>
</endereco>
</pessoa>
```

Figura 4.12: Documento XML pessoa_V2.xml.

Para o tipo de operação *Remoção de Elemento* é realizada a remoção de cada elemento *org.w3c.dom.Element* com o nome *DeltaOperationBean.getName()*. Primeiramente, cada elemento a ser removido é encontrado através do método *org.w3c.dom.Document.getElementsByTagName(nome)* passando o nome *DeltaOperationBean.getName()* do elemento como parâmetro. Cada elemento *org.w3c.dom.Element* encontrado é armazenado em uma lista auxiliar, assim como o nome do elemento pai, pois posteriormente esse elemento pode ser adicionado numa operação do tipo *Adição de Elemento*. Finalmente, cada elemento *org.w3c.dom.Element* é removido através do método *org.w3c.dom.Element.getParentNode().removeChild()*. Os três primeiros *DeltaOperationBean* ilustrados na figura X.X são referentes a operações de *Remoção de Elemento.*

Para o tipo de operação *Alteração do Nome de Elemento* é realizada a atualização do nome do elemento. Primeiramente, o elemento que deve ter o

nome alterado é encontrado através do método *Document.getElementsByTagName(nome)*, onde o parâmetro passado é *DeltaOperationBean.getInitialValue()*. Finalmente, a alteração do nome do elemento é realizada através do método *org.w3c.dom.Document.renameNode(nome)* onde é passado como parâmetro o nome atualizado do elemento, ou seja, *DeltaOperationBean.getFinalValue()*. Para exemplificar esse tipo de operação, podemos considerar o esquema XML A como o esquema *pessoa.xsd*, ilustrado na figura 4.6, e o esquema XML B como o esquema pessoaV3.xsd ilustrado na figura 4.13.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
<xs:element name="pessoa">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="nome" type="xs:string"/>
      <xs:element name="idade" type="xs:int"/>
      <xs:element name="enderecoResidencial" type="tipoEnderecoBrasil"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:complexType name="tipoEnderecoBrasil">
    <xs:sequence>
      <xs:element name="tipo" type="xs:string"/>
      <xs:element name="nomeEndereco"   type="xs:string"/>
      <xs:element name="complemento"   type="xs:string"/>
      <xs:element name="cidade"   type="xs:string"/>
      <xs:element name="estado"  type="xs:string"/>
      <xs:element name="cep"    type="xs:decimal"/>
    </xs:sequence>
</xs:complexType>
</xs:schema>
```

Figura 4.13: Esquema XML pessoaV3.xsd.

A saída do algoritmo *XyDiff* executado sobre os esquemas *pessoa.xsd* e *pessoaV3.xsd* é ilustrada na figura X.Z. Essa saída contém um *XMLCommand* que representa a operação a ser realizada: a atualização do valor do atributo *name* de *endereco* para *enderecoResidencial*. Como o atributo a ser alterado é o atributo *name* essa operação é de *Alteração do Nome de Elemento.*

```
UpdateAttribute: name name path 0:0:0:0:0:2 from endereco to enderecoResidencial
```

Figura 4.14: Saída do algoritmo XyDiff executado sobre os esquemas XML pessoa.xsd e pessoaV3.xsd.

```
DeltaOperationBean [finalValue=enderecoResidencial, initialValue=endereco,
 name=endereco, operationId=2, parentName=null, type=null]
```

Figura 4.15: DeltaOperationBean obtido a partir do XMLCommand gerado para os esquemas XML pessoa.xsd e pessoaV3.xsd.

O próximo passo realizado é a criação de um *DeltaOperationBean,* ilustrado na figura 4.15, identificando a operação de *Alteração do Nome de Elemento* através do campo *operationId*. Os campos *initialValue e finalValue* do *DeltaOperationBean* representam nessa operação, respectivamente, o valor inicial do nome do elemento e o valor final do nome do elemento a ser atualizado. Logo, esses campos recebem respectivamente os valores *endereco* e *enderecoResidencial*. Após as transformações realizadas o documento XML *pessoa_v3.xml*, ilustrado pela figura 4.16, é gerado.

```
<?xml version="1.0" encoding="UTF-8"?>
<pessoa xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="c:/schemas/pessoaV3.xsd">
<nome>felipe</nome>
<idade>20</idade>
<enderecoResidencial>
  <tipo>rua</tipo>
  <nomeEndereco>jose bonifacio</nomeEndereco>
  <complemento>23</complemento>
  <cidade>porto alegre</cidade>
  <estado>rs</estado>
  <cep>90050000</cep>
</enderecoResidencial>
</pessoa>
```

Figura 4.16: Documento XML pessoa_V3.xml.

Para o tipo de operação *Alteração do Tipo de Elemento* realiza-se a atualização do valor do elemento de acordo com o novo tipo do elemento. Primeiramente, o elemento que deve ter o tipo alterado é encontrado através do método *Document.getElementsByTagName(nome)*, onde o parâmetro passado é *DeltaOperationBean.getName()*. Depois, verifica-se se o valor atual desse elemento, obtido através do método *org.w3c.dom.Element.getTextContent()*, é válido para o novo tipo do elemento, *DeltaOperationBean.getFinalValue().* Se o valor atual do elemento é válido para o novo tipo, esse valor é mantido. Caso contrário, esse elemento recebe o valor default do X-Spread 2 ou valor informado pelo usuário para o novo tipo.

Para exemplificar a operação de *Alteração do Tipo de Elemento*, consideramos o esquema XML A como o esquema pessoa.xsd, ilustrado pela figura 4.6 e o esquema XML B como o esquema pessoaV4.xsd, ilustrado pela figura 4.17.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
<xs:element name="pessoa">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="nome" type="xs:string"/>
      <xs:element name="idade" type="xs:int"/>
      <xs:element name="endereco" type="tipoEnderecoBrasil"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:complexType name="tipoEnderecoBrasil">
    <xs:sequence>
      <xs:element name="tipo" type="xs:int"/>
      <xs:element name="nomeEndereco"   type="xs:string"/>
      <xs:element name="complemento"   type="xs:string"/>
      <xs:element name="cidade"   type="xs:string"/>
      <xs:element name="estado"  type="xs:string"/>
      <xs:element name="cep"    type="xs:decimal"/>
    </xs:sequence>
</xs:complexType>
</xs:schema>
```

Figura 4.17: Esquema XML pessoaV4.xsd.

A saída do algoritmo *XyDiff* executado sobre os esquemas XML *pessoa.xsd* e *pessoaV4.xsd* é ilustrada na figura 4.18. Essa saída contém um *XMLCommand* que representa a operação a ser realizada: a atualização do valor do atributo *type* de *xs:string* para *xs:int*. Como o atributo a ser alterado é o atributo *type* essa operação é de *Alteração do Tipo de Elemento.*

```
UpdateAttribute: name type path 0:0:1:0:0 from xs:string to xs:int
```

Figura 4.18: Saída do algoritmo executado sobre os esquemas XML pessoa.xsd e pessoaV4.xsd.

```
DeltaOperationBean [finalValue=xs:int, initialValue=xs:string,
 name=tipo, operationId=12, parentName=null, type=null]
```

Figura 4.19: DeltaOperationBean obtido a partir do XMLCommand gerado para os esquemas XML pessoa.xsd e pessoaV4.xsd.

O próximo passo realizado é a criação de um *DeltaOperationBean,* ilustrado na figura 4.19, identificando a operação de *Alteração do Tipo de Elemento* através do campo *operationId*. Os campos *initialValue, finalValue* e *name* do *DeltaOperationBean* representam nessa operação, respectivamente o valor inicial do tipo do elemento, o valor final do tipo do elemento e o nome do elemento a ser atualizado. Logo, esses campos recebem respectivamente os valores *xs:string, xs:int* e *tipo*.

O nome do elemento a ser atualizado é obtido através de uma busca do elemento representado pela posição, nesse exemplo a posição *0:0:1:0:0* é referente ao elemento *tipo*.

Realiza-se uma validação sobre o valor inicial *rua* do elemento *tipo* para o tipo final *xs:int*. Como o valor inicial *rua* é inválido para o tipo *xs:int*, o X-Spread 2 atualiza o elemento *tipo* com o valor default para o tipo *xs:int,* correspondente a *0*. Atualmente, o X-Spread 2 só realiza a validação para o tipo *xs:int*, logo, para os outros tipos é necessária a implementação da validação para um correto funcionamento do X-Spread 2. Após as transformações realizadas o documento XML *pessoa_V4.xml*, ilustrado pela figura 4.20, é gerado.

```
<?xml version="1.0" encoding="UTF-8"?>
<pessoa xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="c:/schemas/pessoaV4.xsd">
<nome>felipe</nome>
<idade>20</idade>
<endereco>
  <tipo>0</tipo>
  <nomeEndereco>jose bonifacio</nomeEndereco>
  <complemento>23</complemento>
  <cidade>porto alegre</cidade>
  <estado>rs</estado>
  <cep>90050000</cep>
</endereco>
</pessoa>
```

Figura 4.20: Documento XML pessoa_V4.xml.

4.2.4 Validação dos documentos XML gerados

Após as alterações realizadas, o documento adaptado é salvo no diretório informado pelo usuário, através do método *Transformer.transform*, com o nome do documento original acrescentado do sufixo default do X-Spread 2 ou com o sufixo informado pelo usuário.

Posteriormente, é realizada uma validação, verificando se o documento XML gerado é válido para o esquema XML B. Primeiramente, obtem-se uma instância da classe *javax.xml.parsers.DocumentBuilderFactory*. Nessa instância, o atributo *http://java.sun.com/xml/jaxp/properties/schemaSource* recebe a url do esquema B como valor através do método *DocumentBuilderFactory.setAttribute.*

A validação é realizada através da execução do método *javax.xml.parsers.DocumentBuilder.parse(documentoXML)*, passando como parâmetro a url do novo documento XML gerado. Se o novo documento XML gerado é inválido para o esquema XML B, a mensagem contendo o erro encontrado será exibida ao usuário na tela de saída da adaptação de documentos XML. Caso contrário, uma mensagem de sucesso é exibida ao usuário, conforme ilustra a figura 4.4.

4.3 Metodologia de desenvolvimento

O desenvolvimento ágil foi a metodologia que mais se aproximou do desenvolvimento utilizado nesse trabalho, uma vez que vários módulos ou partes do código poderiam ser desenvolvidos independentes e demandavam um tempo de desenvolvimento teoricamente curto. Logo, foram listados os algoritmos e funcionalidades que deveriam ser implementados e, dependendo do andamento do trabalho, um conjunto dessas tarefas era escolhido para ser desenvolvido.

Grande parte do trabalho foi desenvolvido como trabalho da disciplina de Laboratório de Engenharia de Software, ministrada pelo professor Marcelo Pimenta. Como eram realizados encontros periódicos com o professor Pimenta e com a orientadora Renata Galante, eram definidos os objetivos para os próximos encontros. Consequentemente, um conjunto de tarefas eram desenvolvidas entre os encontros e a cada encontro, o protótipo aproximava-se do objetivo final. Portanto, a forma de desenvolvimento do X-Spread 2 foi similar às iterações propostas pelo desenvolvimento com métodos ágeis.

5 CONCLUSÕES

A ferramenta operacional X-Spread 2 atingiu os objetivos propostos, uma vez que realiza a adaptação de um conjunto de documentos XML a partir da detecção de diferenças entre dois esquemas XML informados pelo usuário. As diferenças detectadas são mapeadas para as operações de modificação de *Adição de Elemento, Adição de Elemento, Alteração do Nome de Elemento* e *Alteração do Tipo de Elemento* que são a base das transformações realizadas sobre o conjunto de documentos XML.

O X-Spread 2 também possui uma interface simples, prática e clara, que possibilita ao usuário informar dois esquemas XML, um conjunto de documentos XML e outras configurações para a realização da adaptação desse conjunto de documentos XML, com o objetivo de tornar esse conjunto de documentos XML compatível com o segundo esquema XML informado. Para esclarecimento ao usuário, são exibidas mensagens de alerta quando o usuário não fornece corretamente os dados. Após a geração dos novos documentos XML, para cada documento XML gerado corretamente é exibida uma mensagem de sucesso. Caso algum o documento gerado não seja válido para o esquema XML desejado, uma mensagem com o erro ocorrido é exibida ao usuário.

A ferramenta X-Spread 2 é operacional e possui uma interface que permite e auxilia o usuário a realizar a adaptação de um conjunto de documentos XML. Porém, novas funcionalidades e algoritmos podem e devem ser incorporados para que esquemas XML e documentos XML mais complexos possam ser utilizados na adaptação de documentos XML do X-Spread 2. Algumas melhorias que poderiam ser realizadas:

- Expandir o conjunto de operações de modificações detectadas, por exemplo, acrescentando a detecção da operação de mudança da cardinalidade de um elemento;

- Disponibilizar ao usuário um arquivo de log com as informações das operações realizadas, incluindo, por exemplo, a saída do algoritmo XyDiff e as operações de modificação detectadas;
- Tornar o X-Spread 2 um Web Service, ou seja, transformar a adaptação de documentos XML em um serviço que poderá ser utilizado por outras aplicações;

Portanto, acrescentando novas funcionalidades e aperfeiçoando o X-Spread 2, essa ferramenta poderá ser muito útil para o auxílio em pesquisas e trabalhos sobre a evolução de esquemas XML e a propagação da evolução de esquemas para documentos XML.

REFERÊNCIAS

Apache Tomcat 6.0. Disponível em: <http://tomcat.apache.org/tomcat-6.0-doc/index.html>. Acesso em: nov. 2010.

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

CODEHAUS. jaxen: universal jaxa xpath engine. Disponível em: <http://jaxen.org/>. Acesso em: nov. 2010.

Hong Su, Diane Kramer, Li Chen, Kajal T. Claypool, Elke A. Rundensteiner: XEM: Managing the evolution of XML Documents. RIDE-DM 2001 103-110

Java EE 6. Disponível em: <http://download.oracle.com/javaee/6/api/>. Acesso em: nov. 2010.

JavaServer Faces Technology. Disponível em: <http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html>. Acesso em: nov. 2010.

JDOM. Disponível em : <http://www.jdom.org/>. Acesso em: nov. 2010.

LIU, Z.; CHEN, Y. Return specification inference and result clustering for keyword search on XML. ACM Trans. Database Syst., [S.l.], v.35, n.2, 2010

PERINI, A. B.; SILVEIRA, V. N. K. da; GALANTE, R. M. XSDelta - Uma ferramenta visual para comparação de esquemas XML. In: SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, SBBD, 21., 2006, Florianópolis. Anais da III Sessão de Demos. Florianópolis: SBC, 2006. p. 7–12.

RAGHAVACHARI, M.; SHMUELI, O. Efficient Schema-Based Revalidation of XML. In: EDBT, 2004. Proceedings. . . Berlin: Springer, 2004. p.639–657. (Lecture Notes in Computer Science, v.2992).

SILVEIRA, V. N. K. da. Estudo sobre evolução de esquemas XML. 2006. Trabalho Individual (Mestrado em Ciência da Computação)—Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

SILVEIRA, V. N. K. da; GALANTE, R. M. X-Spread - Um mecanismo para propagação da evolução de esquemas para documentos XML. In: WORKSHOP DE TESES E DISSERTAÇÕES EM BANCO DE DADOS, WTDBD, 5., 2006, Florianópolis, SC. Anais. . . Florianópolis: Sociedade Brasileira de Computação, 2006. p.26–31.

Silveira, V. N. K. da. X-Spread - Um mecanismo automático para propagação da evolução de esquemas para documentos XML. 2007. Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre,BR–RS, 2007.

Silveira, V. (2007). X-Spread - Um mecanismo automático para propagação da evolução de esquemas para documentos XML;

Vincent Nelson Kellers da Silveira, Renata Galante. *X-Spread - A software modeling approach of schema evolution propagation to XML documents.* In: XML: Aplicações e Tecnologias Associadas - XATA, February 2008, Évora, Portugal.

XyDiff Tools, Detecting changes in XML Documents. Disponível em: <http://leo.saclay.inria.fr//software/XyDiff/cdrom/www/xydiff/index-eng.htm)>. Acesso em: nov. 2010.

W3C. W3C XML Schema. Disponível em: <http://www.w3.org/XML/Schema>. Acesso em: nov. 2010.

W3C. XML TECHNOLOGY. Disponível em: <http://www.w3.org/standards/xml/>. Acesso em: nov. 2010.

W3C. W3C Document Object Model. Disponível em: <http://www.w3.org/DOM/>. Acesso em: nov. 2010.