UNIVERSIDADE FEDERAL FLUMINENSE
INSTITUTO DE COMPUTAÇÃO
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Bruno Ferreira Pinto

Fernando Campello

Cálculo de similaridade de documentos XML

Niterói
2012

Bruno Ferreira Pinto

Fernando Campello

Cálculo de similaridade de documentos XML

Monografia apresentada ao Departamento de Ciência da Computação da Universidade Federal Fluminense como parte dos requisitos para obtenção do Grau de Bacharel em Ciência da Computação

Orientador: Leonardo Gresta Paulino Murta

Coorientador: Vanessa Braganholo Murta

Niterói

2012

Bruno Ferreira Pinto

Fernando Campello

Cálculo de similaridade de documentos XML

Monografia apresentada ao Departamento de Ciência da Computação da Universidade Federal Fluminense como parte dos requisitos para obtenção do Grau de Bacharel em Ciência da Computação

Aprovado em outubro de 2012.

BANCA EXAMINADORA

_____

Prof. JOSÉ VITERBO FILHO, D.Sc.

_____

Prof. LUIZ ANDRÉ PORTES PAES LEME, D.Sc.

_____

Prof. LEONARDO GRESTA PAULINO MURTA, D.Sc.

Orientador

_____

Prof. VANESSA BRAGANHOLO MURTA, D.Sc.

Coorientador

Niterói

2012

AGRADECIMENTOS

Aos meus pais, os meus maiores heróis. À minha namorada, Elma, que sempre me deu força e apoio ao longo de todo o projeto. Jamais esquecerei vocês.

Aos meus orientadores Leonardo Murta e Vanessa Braganholo pelo apoio, paciência, sabedoria e compreensão em momentos difíceis desse projeto.

(Fernando Campello)

À minha família, que sempre me apoiou, incentivou e tornou possível que eu chegasse até a conclusão da faculdade. Em especial, à minha mãe e aos meus irmãos que sempre estiveram ao meu lado, me incentivando, dando forças e todo o necessário para que eu seguisse em frente. Sem vocês nada seria possível!

À Bianca, minha namorada, que me apoiou e motivou em todos os momentos, sabendo me animar nas horas de desânimo e dando forças para que a faculdade se concretizasse. Você foi fundamental em mais essa conquista.

(Bruno Ferreira Pinto)

RESUMO

Documentos XML possuem uma estrutura diferente dos documentos de texto comuns. Por esta razão, os algoritmos de diferenciação utilizados nos arquivos de texto comuns obtêm resultados insatisfatórios quando utilizados em documentos XML. Ao longo dos anos, algoritmos específicos foram criados para diferenciar este tipo de documento. O objetivo deste trabalho é discutir os principais algoritmos de diferenciação de documentos XML e apresentar um novo algoritmo. A ferramenta Phoenix foi criada com o intuito de implementar o algoritmo aqui discutido. Esta ferramenta aborda o problema de comparação de documentos XML com um diferencial, utilizando o conceito de similaridade. A Phoenix possui algumas limitações que podem ser trabalhadas em projetos futuros. Atualmente, algumas dessas limitações já estão sendo corrigidas por outros projetos.

Palavras Chave:
XML, Diff, SGWfC, LCS, Húngaro.

ABSTRACT

XML documents have a different structure than common text documents. For this reason, the diff algorithms used on common text documents have an unsatisfactory result when applied to XML documents. Some specific algorithms were created to differentiate this kind of document. The objective of our work is to discuss the main algorithms for comparing XML documents and present a new algorithm. The Phoenix tool was conceived to implement the algorithm here discussed. This tool focuses on the problem of comparing XML documents with a differential: the adoption of the concept of similarity. Phoenix has some limitations that can be solved on future projects. Currently, some of these limitations are already being fixed by other projects.

Keywords:

XML, Diff, SGWfC, LCS, Hungarian.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

CAPÍTULO 1 - INTRODUÇÃO

A infraestrutura computacional tem ajudado engenheiros, biólogos, geólogos e cientistas em geral a melhorar a qualidade dos experimentos científicos, reduzindo o tempo necessário para sua execução. Para que experimentos científicos em larga escala possam ser gerenciados, é preciso que um conjunto de funcionalidades esteja presente, dentre as quais estão o apoio à concepção de workflows científicos, a reutilização dos que foram previamente concebidos por outros cientistas, o controle sobre a evolução das diferentes versões do workflow e a coleta de informações que permita identificar a proveniência dos dados de origem (MATTOSO *et al.*, 2009).

Os sistemas de gerência de *workfows* científicos (SGWfC) são ferramentas que fornecem uma infraestrutura para controlar workflows e seus respectivos resultados. A maioria dos SGWfCs usa arquivos XML para armazenar as especificações de workflows científicos. No entanto, nenhum dos SGWfC faz uso de gerência de configuração (ESTUBLIER, 2000) no que diz respeito à diferenciação (*diff*) e mesclagem (*merge*) de dados com uma complexidade maior, adotando somente técnicas simples tanto para versões quanto para rastreabilidade. Esta estratégia acaba se tornando ineficiente e cria limitações na análise dos dados, pois pode culminar em dados inconsistentes ou inválidos.

É muito comum que o cientista precise executar sua experiência repetidamente, alterando seus valores de entrada e analisando os respectivos efeitos no modelo de acordo com as mudanças de seus parâmetros e resultados. Para isso, ele precisa comparar resultados de diversas execuções, e ter uma noção clara de quais mudanças nas entradas originaram as mudanças nos resultados. Por exemplo, se um cientista realizar 20 alterações no workflow até encontrar uma configuração do experimento que obtenha resultados satisfatórios, provavelmente não terá certeza de quais alterações contribuíram para o resultado. Como cada uma das modificações culminou em um documento XML diferente, a possibilidade de comparar esses documentos passa a ser fundamental na compreensão da evolução de experimentos científicos.

Atualmente no mercado, existem várias ferramentas de diferenciação de arquivos e algumas delas já estão integradas aos sistemas de controle de versão, como o Subversion (

COLLINS-SUSSMAN et al. 2008) e CVS (CEDERQVIST, 2003). Os arquivos, de um modo geral, são tratados por esses sistemas como documentos de texto puro, sendo comparados linha a linha e identificando quais linhas são iguais e quais são diferentes. Contudo, caso os arquivos XML sejam tratados como arquivos de textos simples, o resultado pode ser um documento malformado, uma vez que um XML segue uma estrutura hierárquica, com atributos e subelementos, que não podem ser considerados iguais ou diferentes apenas por comparação linha a linha, mas sim por análise de toda a sua estrutura.

Embora a motivação inicial desse trabalho tenha sido a comparação de arquivos XML que representam workflows científicos, o problema de comparar XMLs transcende o domínio de experimentos científicos, e é relevante para todos os domínios que usam XML. Considere o exemplo da Figura 1 que ilustra o problema da comparação de dois arquivos XML por algoritmos textuais. Neste exemplo, é mostrada parte de um arquivo XML que representa o layout de uma página web. Suponha que este arquivo esteja em um repositório remoto, controlado por um sistema de controle de versão. A Figura 2 e a Figura 3 mostram alterações nesse arquivo feitas por dois usuários em paralelo.

```
<h:dataTable id = "tblUsuarios"
             var = "usuário"
             value = "#{UsuarioMB.listaDeUsuario}" >
 ...
```

Figura 1: Exemplo de página do JSF

```
<h:dataTable rendered = "true"
             id = "tblUsuarios"
             var = "usuário"
             value = "#{UsuarioMB.listaDeUsuario}" >
```

Figura 2: Adição de um atributo por um usuário

```
<h:dataTable id= "tblUsuarios"
             var= "usuário"
             value= "#{UsuarioMB.listaDeUsuario}"
             rendered = "false" >
```

Figura 3: Adição do mesmo atributo por outro usuário em paralelo

Após realizar as alterações, o primeiro usuário submete as mudanças ao repositório e então, para que o segundo usuário possa submeter uma nova alteração, este precisa mesclar as suas modificações com as que o primeiro usuário realizou. Caso não faça a

mesclagem, o controle de versão tentará realizá-la automaticamente e, se não for possível, este negará a submissão até que a mesclagem seja feita manualmente. Neste caso, ambos os usuários inseriram o atributo “rendered” no mesmo elemento, porém em ordens diferentes. Essas alterações foram feitas em linhas diferentes do arquivo e, portanto, muitos sistemas de controle de versão executariam a mesclagem automaticamente sem detectar nenhum conflito, resultando no arquivo final ilustrado na Figura 4.

```
<h:dataTable rendered = “true”
             id=”tblUsuarios”
             var= “usuário”
             value= “#{UsuarioMB.listaDeUsuario}”
             rendered = “false” >
```

Figura 4: Resultado da mesclagem executado por um VCS qualquer da Figura 2 e Figura 3

Como pode ser visto, este arquivo é um documento XML malformado, pois o novo elemento passa a ter dois atributos com nomes iguais. O motivo desse comportamento é que esses algoritmos de diferenciação não conseguem tratar arquivos XML de forma apropriada, pois não levam em conta a estrutura hierárquica, conforme dito anteriormente. Assim, novos algoritmos foram criados para suprir esta demanda.

Entre os algoritmos existentes que tratam diferenciação de documentos XML, pode-se citar o X-diff (WANG et al. 2003), o Xydiff (COBENA et al. 2002)(COBENA; S. ABITEBOUL; MARIAN, 2002) e XMLTreeDiff (CURBERA; D. A. EPSTEIN, 1999). Embora estes algoritmos solucionem parcialmente o problema encontrado no que diz respeito à diferenciação de documentos XMLs, estes ainda possuem algumas peculiaridades. Para ilustrar, considere o exemplo de dois workflows científicos, representado na Figura 5.

(a)

```
<workflow>
    <activity>A</activity>
    <activity>B</activity>
    <activity>C</activity>
    <connection in=“A” out=“B” />
</workflow>
```

(b)

```
<workflow>
    <activity>C</activity>
    <activity>A</activity>
    <activity>B</activity>
    <connection out=“B” in=“A” />
</workflow>
```

Figura 5: Exemplo de dois workflows com mesmo conteúdo, mas com nome dos elementos diferente

Conforme ilustra a Figura 5, ambos os documentos são iguais, mudando apenas a ordem em que os seus subelementos aparecem. Ao comparar esses documentos usando os

algoritmos descritos anteriormente, todos identificarão uma diferença já que eles não levam em consideração a ordem em que os elementos aparecem. Dito isso, para muitas aplicações, esse conceito não é prático, pois além de não conseguir detectar a igualdade de um determinado elemento, mesmo que este mesmo elemento apareça mais à frente no outro documento, os algoritmos não são capazes de identificar a similaridade entre eles. Assim, outras propriedades dos documentos, como, por exemplo, o conteúdo dos seus subelementos e atributos, deveriam ser consideradas na análise de comparação. Uma vez identificada a diferença entre os documentos XML surge um novo problema. É preciso exibir estas diferenças detectadas de forma mais intuitiva para o usuário.

A maioria das ferramentas de visualização de diferenças não está integrada ao algoritmo de diferenciação. Essa carência gera uma limitação na apresentação dos dados ao usuário, já que cada algoritmo possui uma forma diferente de evidenciar o documento XML resultante. É de extrema relevância que o usuário possa entender as diferenças detectadas de forma simples com o intuito de avaliá-las.

O objetivo deste trabalho é elaborar um algoritmo para o problema de comparação de documentos XML de um mesmo domínio. A construção de uma ferramenta, a Phoenix, que possibilite a comparação de documentos XML utilizando o algoritmo proposto, com uma interface gráfica, que permita visualizar, na forma de grafos, a diferença entre os documentos comparados, realçando partes iguais, semelhantes e diferentes, valida a proposta do trabalho.

O restante deste trabalho está organizado da seguinte forma. Os algoritmos de comparação de documentos XML discutidos superficialmente na introdução são abordados no Capítulo 2 com maiores detalhes. O Capítulo 2 discute também outros algoritmos de comparação de documentos XML existentes, tanto comerciais como acadêmicos, fazendo uma comparação entre eles. No Capítulo 3 é apresentada a abordagem Phoenix. O Capítulo 4 descreve o funcionamento da Phoenix com um exemplo ilustrativo de comparação de dois documentos XML. Para concluir, o Capítulo 5 compara os algoritmos discutidos no Capítulo 2 com a proposta deste trabalho, apontando possíveis limitações da abordagem e finaliza com uma série de atividades em desenvolvimento e a serem desenvolvidas em trabalhos futuros.

CAPÍTULO 2 - ALGORITMOS DE COMPARAÇÃO

Um documento XML é formado por uma estrutura baseada em árvores. Essa característica aumenta a complexidade da comparação entre documentos deste tipo. Embora existam sistemas de diferenciação textual que utilizem algoritmos de comparação de sequência (COBENA et al. 2002) ou de conjuntos (C. SANTOS e HARA, 2007) obtendo resultados satisfatórios, neste capítulo, são ilustradas algumas limitações no uso exclusivo de um destes algoritmos no âmbito dos documentos XML.

Uma diferença entre os algoritmos baseados em sequência e em conjunto é a forma como estes lidam com a ordenação de seus componentes. Por exemplo, o conjunto {‘a’, ‘b’, ‘c’} é igual ao conjunto {‘b’, ‘a’, ‘c’}, mas as sequências ‘abc’ e ‘bac’ não são iguais. Em documentos XML, a ordenação de seus elementos pode ou não ser relevante dependendo dos tipos dos elementos comparados, como será visto nas próximas seções. Um dos motivos para a realização deste trabalho é possibilitar que estes algoritmos trabalhem com um conceito de similaridade. No exemplo das sequências acima, ao invés destas serem consideradas iguais ou diferentes, passariam a possuir um grau de similaridade que varia de 0% a 100%.

Neste capítulo, são discutidos alguns algoritmos clássicos utilizados na elaboração deste projeto. Também são analisados algoritmos relacionados com a proposta. A Seção 2.1 aborda a comparação de sequências, os tipos de dados que demandam esta comparação e um algoritmo clássico. A Seção 2.2 aborda a comparação de conjuntos, os tipos de dados que demandam esta comparação e um algoritmo. A Seção 2.3 lista algumas abordagens relacionadas a este trabalho. Finalmente, a Seção 2.4 apresenta o resultado da análise dos trabalhos relacionados solidificando a necessidade para a realização deste trabalho.

2.1 COMPARAÇÃO DE SEQUÊNCIAS

Em uma aplicação, a comparação de dois ou mais textos pode ser desejada. O primeiro texto pode ser, por exemplo, T_1 = “comprimento”, enquanto o outro pode ser, por

exemplo, T_2 = “cumprimento”. Um dos objetivos dessa comparação é determinar o quanto os textos são semelhantes. Uma forma de realizar essa análise é verificar a quantidade de alterações necessárias para transformar um texto no outro (CORMEN *et al.*, 2009).

CORMEN *et al.*, (2009) definem uma subsequência de uma determinada sequência como sendo a sequência original com zero ou mais elementos omitidos. No exemplo anterior, subsequências do texto T_1 poderiam ser o próprio texto “comprimento”, “cmprimento”, “t”, “no”, entre outros. No contexto da comparação de sequências, gostaríamos de obter a menor quantidade de alterações necessárias para transformar uma subsequência na outra, e usar essa informação para calcular a similaridade entre os dois textos. A subsequência de letras T_3 com maior grau de semelhança entre os textos em questão seria “cmprimento”, pois para transformar esta em qualquer um dos textos originais é necessária apenas uma alteração (de adição).

Vale ressaltar que as sequências de letras não precisam ser consecutivas, mas que a ordem das mesmas é relevante. Por exemplo, as palavras “ator” e “rato” são compostas pelas mesmas letras, entretanto, como estas possuem ordenações distintas, a sequência comum mais longa obtida pela comparação de sequências seria “ato”. Note também que a comparação não é semântica, ou seja, a similaridade de dois textos não depende do significado de ambos.

Um algoritmo clássico para esse tipo de comparação é o *Longest Common Subsequence*, ou LCS (CORMEN et al., 2009). Este algoritmo é, por exemplo, utilizado nas ferramentas padrão de *diff* do Unix (“GNU Diffutils”,2012) e em problemas de Bioinformática (ACHARD et al., 2001).

O algoritmo LCS cria uma matriz de tamanho M x N, onde cada coluna é um caractere de uma das sequências, de tamanho M, e cada linha é um caractere da outra sequência, de tamanho N. Após gerar esta matriz, o algoritmo calcula a maior subsequência comum entre as sequências comparadas. A Figura 6 exibe a definição formal deste algoritmo.

$$LCS(X_i, Y_j) = \begin{cases} \emptyset & se\ i = 0\ or\ j = 0 \\ (LCS(X_{i-1}, Y_{j-1}), x_i) & se\ x_i = y_j \\ longest\left(LCS(X_i, Y_{j-1}),\ LCS(X_{i-1}, Y_j)\right) & se\ x_i \neq y_j \end{cases}$$

Figura 6: Definição formal do algoritmo LCS

O cálculo do algoritmo LCS pode ser armazenado em uma tabela que contenha duas informações por célula: um número que representa a quantidade de caracteres comuns encontrados pelo algoritmo; e uma seta que indica os caracteres em comum para cada subsequência. A Tabela 1 exemplifica o algoritmo LCS para os textos 'comprimento' e 'cumprimento'.

0	1	2	3	4	5	6	7	8	9	10	11	12	13
1		Ø	C	U	M	P	R	I	M	E	N	T	O
2	Ø	0	0	0	0	0	0	0	0	0	0	0	0
3	C	0	↖ 1	← 1	← 1	← 1	← 1	← 1	← 1	← 1	← 1	← 1	← 1
4	O	0	↑ 1	← 1 ↑	← 1 ↑	← 1 ↑	← 1 ↑	← 1 ↑	← 1 ↑	← 1 ↑	← 1 ↑	← 1 ↑	← 1 ↑
5	M	0	↑ 1	← 1 ↑	↖ 2	← 2	← 2	← 2	↖ 2	← 2	← 2	← 2	← 2
6	P	0	↑ 1	← 1 ↑	↑ 2	↖ 3	← 3	← 3	← 3	← 3	← 3	← 3	← 3
7	R	0	↑ 1	← 1 ↑	↑ 2	↑ 3	↖ 4	← 4	← 4	← 4	← 4	← 4	← 4
8	I	0	↑ 1	← 1 ↑	↑ 2	↑ 3	↑ 4	↖ 5	← 5	← 5	← 5	← 5	← 5
9	M	0	↑ 1	← 1 ↑	↖ 2	↑ 3	↑ 4	↑ 5	↖ 6	← 6	← 6	← 6	← 6
10	E	0	↑ 1	← 1 ↑	↑ 2	↑ 3	↑ 4	↑ 5	↑ 6	↖ 7	← 7	← 7	← 7
11	N	0	↑ 1	← 1 ↑	↑ 2	↑ 3	↑ 4	↑ 5	↑ 6	↑ 7	↖ 8	← 8	← 8
12	T	0	↑ 1	← 1 ↑	↑ 2	↑ 3	↑ 4	↑ 5	↑ 6	↑ 7	↑ 8	↖ 9	← 9
13	O	0	↑ 1	← 1 ↑	↑ 2	↑ 3	↑ 4	↑ 5	↑ 6	↑ 7	↑ 8	↑ 9	↖ 10

Tabela 1: Tabela demonstrativa do algoritmo LCS (CORMEN *et al.*, 2009)

Quando o algoritmo encontra dois caracteres iguais nas posições sendo comparadas, este irá aumentar a quantidade de igualdades encontradas até o momento e armazenar na célula esta quantidade com uma seta apontando para a diagonal superior à esquerda. Quando encontra um elemento diferente, a maior quantidade de igualdades encontradas até o momento será copiada e uma seta será adicionada para a esquerda e/ou para cima dependendo de qual destas células possui o caminho com a maior igualdade até o momento. Para fazer a leitura desta tabela, o algoritmo percorre as setas a partir da célula mais à

direita e mais a baixo [13,13]. Então, o algoritmo irá percorrer o caminho que proporcionou a maior igualdade (seguindo as setas). As células a serem percorridas seriam [12,12], [11,11], [10,10], até chegar à célula [4,4] onde a seta deixa de ser diagonal superior à esquerda (igualdade encontrada) e passa a apontar para a esquerda ou para cima (desigualdade). Nesta célula, o algoritmo poderia escolher tanto a célula a cima quanto a célula à esquerda. É observável que, independente do caminho escolhido, ambos seguirão pela célula [3,3] (igualdade).

No âmbito de documentos XML, para que esse tipo de comparação obtenha resultados desejáveis, é necessário que a comparação seja realizada individualmente nos diferentes níveis de estrutura do documento XML. Considere o exemplo da Figura 7.

(a)

```
<pessoa>
  <nome>João</nome>
  <sobrenome>Silva<sobrenome/>
</pessoa>
```

(b)

```
<pessoa>
  <sobrenome>Silva<sobrenome/>
  <nome>João</nome>
</pessoa>
```

Figura 7: Exemplo de dois documentos XML iguais.

Semanticamente, os dois documentos da Figura 7 são iguais. Entretanto, ao aplicarmos um algoritmo genérico de comparação de sequências, veremos que a ordenação inversa dos elementos “nome” e “sobrenome” fará com que a similaridade entre ambos seja baixa. Um algoritmo mais elaborado, conhecendo as características de um documento XML, poderia dividir os documentos em elementos e atributos, e calcular a similaridade de um documento como a soma da similaridade de suas partes. Dessa forma, ao compararmos individualmente os elementos “nome” e “sobrenome” de ambos os documentos, obteremos um alto nível de similaridade. Posteriormente, ao compararmos o componente mais geral, o elemento “pessoa”, obteremos uma baixa similaridade já que a ordenação de seus elementos não é igual. Assim, a soma da similaridade dos componentes de ambos os documentos seria maior do que a similaridade do documento como um todo, porém esta ainda não seria máxima.

2.2 COMPARAÇÃO DE CONJUNTOS

Analogamente à comparação de sequências, dois conjuntos são semelhantes se o número de alterações necessárias para transformar um no outro é pequeno. A diferença entre os algoritmos de comparação de sequências e os de conjuntos é que a ordem na comparação de conjuntos é irrelevante. Utilizando o mesmo exemplo da Seção 2.1, as palavras “ator” e “rato” são compostas pelas mesmas letras e, embora possuam ordenações distintas, o maior conjunto de letras obtido pela comparação de conjuntos seria {‘r’, ‘a’, ‘t’, ‘o’} ao invés de somente {‘a’, ‘t’, ‘o’}.

Um algoritmo de casamento ótimo, o Algoritmo Húngaro (Kuhn, H.W. 2005), é utilizado para realizar a comparação de conjuntos. Nesse algoritmo, para realizar o casamento de dois conjuntos, calcula-se a similaridade de cada elemento de um conjunto em relação a cada um dos elementos do outro conjunto. Então, o algoritmo Húngaro calcula o casamento entre os elementos dos conjuntos que gera a similaridade máxima. A Tabela 2 exemplifica a execução de um algoritmo de similaridade de conjunto para os conjuntos {‘r’, ‘a’, ‘t’, ‘o’} e {‘a’, ‘t’, ‘o’, ‘r’}. Quando o algoritmo encontra uma similaridade entre os elementos, este irá colocar na célula um valor entre 0 e 1 indicando a similaridade entre cada um dos elementos (0 é desigualdade e 1 é igualdade).

	Ø	R	A	T	O
Ø	–	0	0	0	0
A	0	0	1	0	0
T	0	0	0	1	0
O	0	0	0	0	1
R	0	1	0	0	0

Tabela 2: Tabela demonstrativa do Algoritmo de Conjunto

Assim como na comparação de sequências, no âmbito de documentos XML, para que a comparação de conjuntos obtenha melhores resultados, é necessário que esta seja realizada individualmente nos diferentes níveis da estrutura do documento XML. Considere o exemplo da Figura 8.

(a)	(b)
<pessoa> <nome>João</nome> <sobrenome>Silva<sobrenome/> </pessoa>	<pessoa> <nome>oJoã<nome/> <sobrenome>Silva</sobrenome> </pessoa>

Figura 8: Exemplo de dois documentos XML diferentes.

Semanticamente, os dois documentos da Figura 8 são diferentes. Entretanto, ao aplicarmos um algoritmo genérico de comparação de conjuntos onde a ordenação dos elementos é irrelevante, a similaridade entre ambos os documentos é total dado que o conjunto de letras {‘J’, ‘o’, ‘ã’, ‘o’} é igual ao conjunto {‘o’, ‘J’, ‘o’, ‘ã’}, ou seja, os dois documentos são iguais. Um algoritmo mais elaborado, sabendo das características de um documento XML, poderia dividir os documentos em elementos e atributos, e calcular a similaridade de um documento como um todo como a soma da similaridade de suas partes. Dessa forma, ao compararmos individualmente o atributo “nome” de ambos os documentos, obteremos um baixo nível de similaridade. Posteriormente, ao compararmos o componente mais geral, “pessoa”, obteremos uma similaridade maior já que o atributo “sobrenome” de ambos os documentos são iguais. Dessa forma, ambos os documentos seriam considerados semelhantes, mas não iguais.

2.3 COMPARAÇÃO COM TRABALHOS RELACIONADOS

Conforme foi discutido anteriormente no capítulo 1, a comparação de arquivos XML não deve ser feita de forma textual, pois pode gerar uma análise incorreta devido a não compreensão da sua estrutura. Em virtude disso, novos algoritmos e programas de diferenciação foram criados para suprir esta demanda. Como o objetivo deste trabalho não é comparar todas as ferramentas existentes, apenas algumas foram usadas para que se possa ter um comparativo de como os algoritmos de diferenciação funcionam.

Nessa Seção, são abordados alguns exemplos usando algumas das ferramentas de diferenciação de documentos XML disponíveis no mercado. Como há muitas ferramentas, foi preciso selecionar algumas delas para que possam ser comparadas. Então para reduzir o

conjunto de opções, foi estabelecido que só seriam escolhidos as ferramentas que apresentassem uma interface gráfica capaz de auxiliar os usuários. Entre elas, estão:

- DiffDog (ALTOVA DIFFDOG, 2012);
- ExamXML Pro (A7SOFT 2003);
- DeltaXML (DELTAXML COMPARE, 2010); e
- oXygen (OXYGEN, 2002).

A Subseção 2.3.1 destaca como o estudo foi executado. As subseções posteriores discutem brevemente cada ferramenta falando dos seus pontos positivos e negativos.

2.3.1 PLANEJAMENTO DO ESTUDO

Com o objetivo de avaliar o comportamento de outras ferramentas de comparação de arquivos XML, foram criados dois documentos XML para estudo. Esses documentos são trechos reduzidos de dados para que a comparação seja feita de forma clara e intuitiva já que, para a maioria das ferramentas usadas, a comparação de arquivos acaba gerando uma dificuldade de entendimento para os usuários. Espera-se que o mesmo comportamento para esses trechos também seja observado em estruturas mais complexas, com uma grande quantidade de dados. A Figura 9 ilustra esses trechos reduzidos de dados.

(a)

```
<autores>
 <autor id="01">
   <nome>João da Silva</nome>
   <instituicao>ABC</instituicao>
 </autor>
 <autor id="02">
   <nome>Maria da Silva</nome>
   <instituicao>XYZ</instituicao>
 </autor>
</autores>
```

(b)

```
<autores>
 <autor id="01">
   <nome>Maria da Silva</nome>
   <instituicao>ABC</instituicao>
 </autor>
 <autor id="02">
   <nome>João da Silva</nome>
   <instituicao>XYZ</instituicao>
 </autor>
 <autor id="03">
   <nome>José Silveira</nome>
   <instituicao>XYZ</instituicao>
 </autor>
</autores>
```

Figura 9: Dois trechos de documentos XML usados para estudo de caso das ferramentas.

Os elementos do documento da Figura 9 (a) são exatamente iguais aos elementos do documento da Figura 9(b) com algumas ressalvas: a ordem em que os elementos aparecem é diferente; houve a inclusão de outro elemento no documento da Figura 9 (b); o "id" de alguns elementos são diferentes; e João e Maria mudaram de instituição.

Os dois documentos XML para estudo foram usados em cada uma das ferramentas e ao final, foram discutidos os seus prós e contras. Entre os critérios de comparação, foram considerados os seguintes itens:

- Visualizar o casamento de forma gráfica;
- Comparar arquivos;
- Comparar de forma textual, onde o usuário pode editar um documento ou pode copiar e colar para dentro da ferramenta;
- Comparar diretórios, onde todos os arquivos são comparados um a um;
- Navegar entre os conflitos com o auxilio da ferramenta;
- Informar ao usuário o quanto um documento é similar ao outro;
- Tratar árvores ordenadas; e
- Tratar árvores não ordenadas.

Para que possa ser analisado se os algoritmos de diferença fazem referência às árvores ordenadas ou não ordenadas, algumas modificações nos documentos utilizados para o estudo tiveram que ser feitas. Dentre elas, estão a retirada dos atributos dos elementos, a reorganização dos elementos e a inclusão ou remoção de outros elementos. É possível ver todos os cenários testados no Apêndice A deste trabalho.

2.3.2 RESULTADOS DO DIFFDOG

A primeira ferramenta usada para comparação foi a DiffDog (ALTOVA DIFFDOG, 2012). Entre as funcionalidades disponíveis nesta ferramenta, estão: o tipo de comparação

que deve ser feito (texto, XML, binário); o modo de visualização dos dados (em texto ou em Grid); a escolha do formato dos caracteres; interação do usuário para navegar entre os conflitos e a mesclagem dos elementos de acordo com a sua necessidade. O DiffDog possui uma funcionalidade capaz de mostrar os dados dos documentos XML em formato de tabela.

A Figura 10 mostra a comparação de XML do Diffdog em modo texto, enquanto a Figura 11 mostra em modo Grid (tabela). Os elementos que estão em verde são os elementos que foram identificados como diferentes pelo algoritmo usado. As linhas com colchetes representam quais elementos estão sendo comparados.

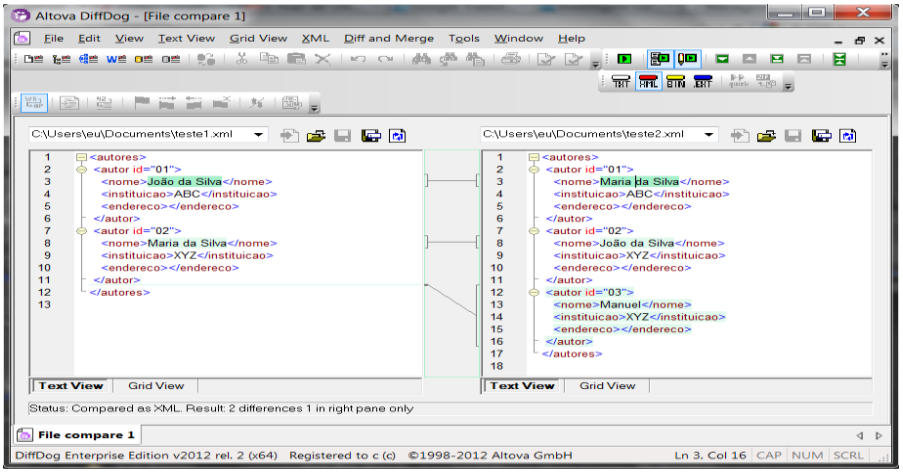

Figura 10: Comparação dos XML usando o DiffDogg em modo texto.

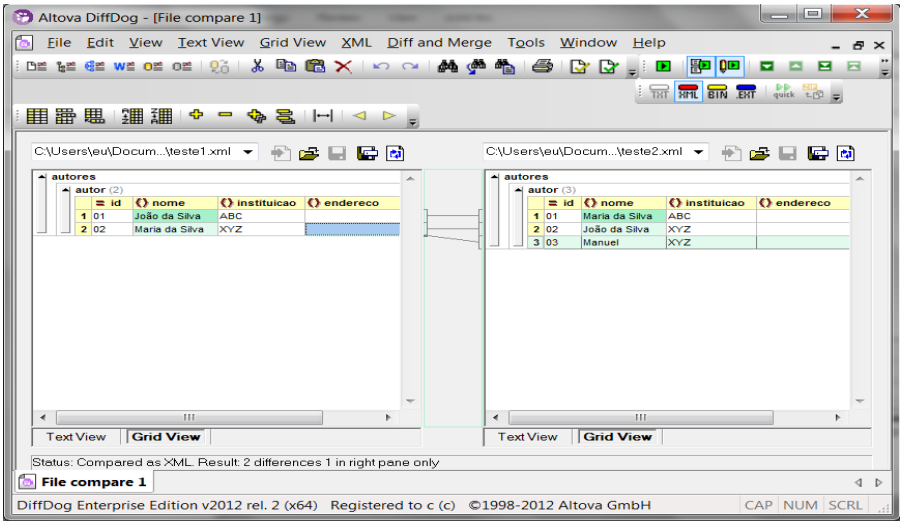

Figura 11: Comparação dos XML usando DiffDog em modo Grid.

A ferramenta DiffDog se mostrou a mais intuitiva das ferramentas aqui apresentadas para identificação de conflitos, pois mesmo que não haja figuras, o DiffDog usou tabelas para mostrar as diferenças e igualdades dos elementos, além de apontar quais deles estão sendo comparados. No entanto, o DiffDog possui algumas limitações.

A diferenciação dos documentos não é feita de forma a verificar se existe ou não a mesma instância daquele elemento na outra estrutura. Ou seja, se forem comparados dois arquivos iguais com os elementos arranjados em ordens diferentes, o DiffDog acusará um conflito. Assim, a ordem em que os elementos aparecem torna-se prioridade para se verificar a diferença entre os documentos. Outra limitação é que os conflitos são mostrados através de cores e linhas, o que pode causar um desconforto para o leitor na visualização dos conflitos.

2.3.3 RESULTADO DO EXAMXML PRO

Assim como o Diffdog, o ExamXML Pro (A7SOFT, 2003) é uma ferramenta desenvolvida para diferenciar, editar e mesclar dados de arquivos XML. O ExamXML Pro possui um algoritmo um pouco mais elaborado que o DiffDog, pois este faz a comparação dos elementos na ordem em que aparecem, enquanto aquele faz a diferenciação procurando referências iguais ao elemento comparado. Ou seja, para o ExamXML Pro a ordem em que os elementos aparecem não é relevante. A Figura 12 mostra a IDE de comparação do ExamXML Pro.

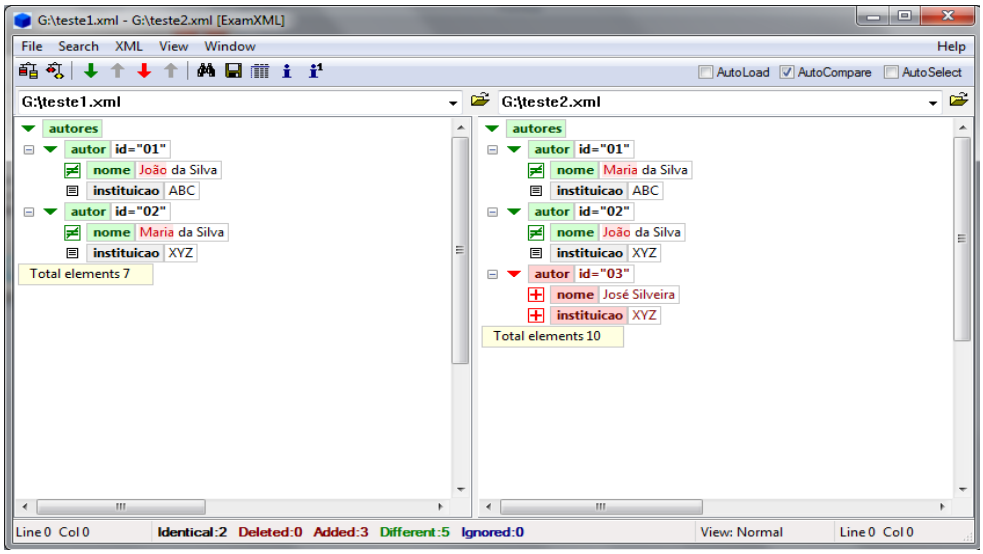

Figura 12: IDE da ferramenta ExamXML Pro.

No Apêndice A são mostradas todas as variâncias utilizadas dos documentos da Figura 12, para que se possa analisar esta ferramenta de uma forma mais apurada. O resultado foi satisfatório, pois o algoritmo conseguiu indicar para o usuário os elementos que são iguais e os elementos que são diferentes independente de sua organização.

Outro ponto positivo que pode ser destacado seria como o conteúdo dos elementos são comparados. O algoritmo não desconsidera o conteúdo dos elementos casados, mesmo que sejam um pouco diferentes. Assim, o usuário é capaz de visualizar que parte da string é igual à outra. A Figura 13 ilustra como o ExamXML Pro trata essas diferenças.

... <nome>**João** Ninguem</nome> ...	... <nome>**João** Da Silva</nome> ...

Figura 13: Identificação de igualdade entre strings de um elemento pelo ExamXML Pro.

O ExamXML Pro possui uma funcionalidade interessante, que é o tipo de visualização que o usuário deseja analisar. Entre elas estão: a visualização normal, que mostra todos os conflitos, igualdades e diferenças entre os documentos; visualização das

diferenças, que mostra apenas as diferenças entre os documentos comparados; e a visualização das similaridades, que mostra apenas as similaridades encontradas entre eles.

Apesar de ter um algoritmo um pouco mais elaborado que o DiffDog, o ExamXML Pro possui alguns pontos negativos. Ele apresenta seus resultados sempre em modo texto. Essa abordagem pode se tornar suscetível a erros, à medida que se aumenta a complexidade dos documentos XML. Uma maneira encontrada para contornar este entrave foi a possibilidade do usuário acompanhar qual elemento está sendo comparado, através do clique do mouse sobre ele.

2.3.4 RESULTADO DO DELTA XML

O DeltaXML (DELTAXML COMPARE, 2010) apresenta uma IDE simples onde o usuário tem a opção de fazer a comparação ou em modo texto, ou em modo arquivo. A Figura 14 mostra os tipos de saída que o usuário pode utilizar para visualizar a diferenciação dos documentos.

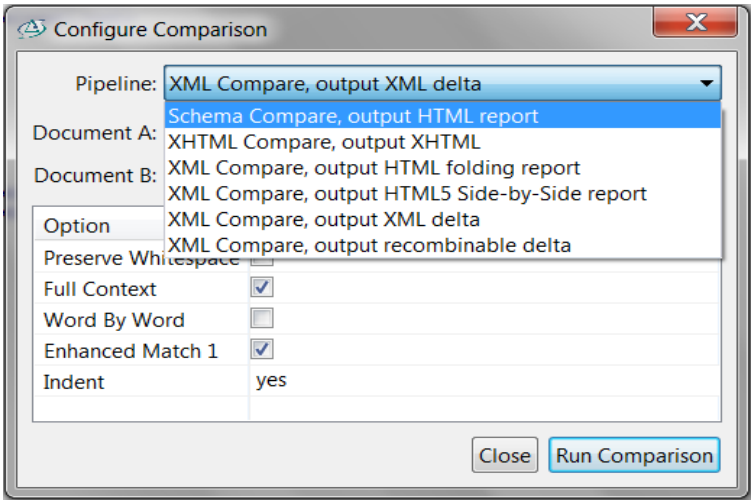

Figura 14: Configuração de output do DeltaXML.

Entre os tipos que apresentam as saídas mais intuitivas, tem-se o “Schema Compare, output HTML report” e o “XML Compare, output HTML5 Side-by-Side report”, conforme ilustram, respectivamente, a Figura 15 e a Figura 16.

Figura 15: Resultado da comparação do DeltaXML com saída em HTML.

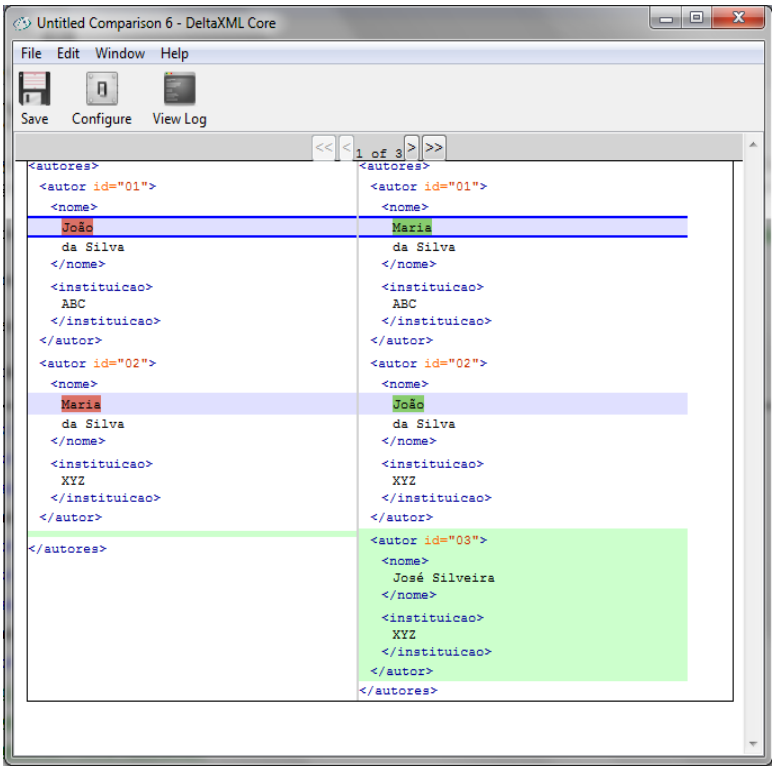

Figura 16: Resultado da comparação do DeltaXML com saída em HTML5.

Um ponto positivo é que a ferramenta é simples e de fácil manuseio. Além disso, possui outras formas de representar os dados, o que pode ser interessante para algumas aplicações, como, por exemplo, web. Porém, possui o mesmo problema discutido em outras ferramentas: o resultado é apresentado em modo texto que pode causar um desconforto para o usuário, aumentando a incidência de erros relacionado à identificação de conflitos.

Outro ponto negativo, referenciado anteriormente na análise da ferramenta ExamXML Pro, é a dificuldade de identificar os conflitos caso aumente a complexidade dos documentos.

2.3.5 RESULTADO DO OXYGEN

O oXygen (OXYGEN, 2002) possui duas IDE para comparação de arquivos. Uma para arquivos e outra para diretórios. Ao executar a diferenciação, é possível que o usuário não só navegue pelos conflitos, mas também mescle os dados à medida que for percorrendo os documentos. De forma equivalente às outras ferramentas apresentadas até o momento, com exceção do ExamXML Pro, o Oxygen faz a diferenciação na ordem que os elementos aparecem. Ou seja, as árvores devem ser ordenadas. A Figura 17 mostra a interface da ferramenta oXygen.

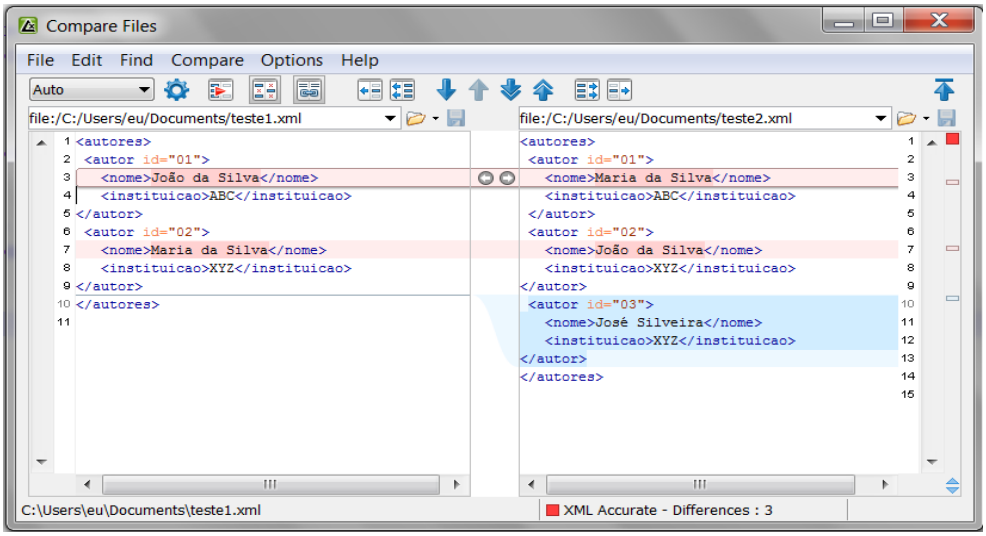

Figura 17: IDE da ferramenta oXygen

2.4 CONSIDERAÇÕES FINAIS

A Tabela 3 apresenta um comparativo das ferramentas discutidas anteriormente.

	DiffDog	ExamlXML Pro	Delta XML	oXygen
Visualizar o casamento de forma gráfica	Sim	±	±	±
Comparar de forma textual, onde o usuário pode editar um documento	Sim	Não	Não	Sim
Comparar diretórios	Sim	Não	Não	Sim
Navegar entre os conflitos	Sim	Sim	Sim	Sim
Informar similaridade	Não	Não	Não	Não
Tratar árvore ordenadas	Não	Não	Não	Não

Tabela 3: Ferramentas de Visualização de Diferenças

Grande parte das ferramentas apresentadas neste capítulo possui outras funcionalidades além da comparação de arquivos XML, como, por exemplo, a mesclagem de dados, seja ela automatizada ou manual, a diferenciação de outros tipos de arquivos, não só em XML, entre outros. Embora sejam importantes, essas funcionalidades fogem do escopo deste trabalho.

CAPÍTULO 3 - ABORDAGEM PHOENIX

Como foi visto no Capítulo 1, alguns sistemas de comparação de documentos XML utilizam comparação textual, com exceção do X-Diff (Wang et al. 2003), XMLTreeDiff (CURBERA; D. A. EPSTEIN, 1999) e o Xy-Diff (MARIAN *et al.*, 2001). Este tipo de comparação agrega uma complexidade que, para um algoritmo de diferenciação textual, dificulta a compreensão de um documento XML impedindo uma análise de forma correta. Isso se dá pelo fato de que um documento XML é formado por uma estrutura hierárquica, com atributos, elementos e subelementos, e não deve ser comparado de forma isolada. Para solucionar estes problemas, foram criados algoritmos de diferenciação específicos para esses documentos, conforme citado no Capítulo 1 e Capítulo 2 .

A proposta desse trabalho é apoiar na comparação e detecção de diferenças, exclusivamente de documentos XML, e apresentar os resultados de forma gráfica, com o objetivo de facilitar o entendimento dos usuários sobre o que está sendo identificado como igual ou diferente. O nome dessa abordagem é Phoenix.

Este capítulo está dividido em cinco seções, além desta introdução. A Seção 3.1 detalha os tipos de comparação que são executadas pelo algoritmo Phoenix. A Seção 3.2 aborda a ponderação dos elementos a fim de classificar o que é mais relevante para o usuário na hora que for comparar dois documentos XML. A Seção 3.3 ilustra como é feita a representação gráfica dos documentos XML. A Seção 3.4 discorre sobre os tipos de interpretação que o usuário pode ter de duas versões. Por último, a Seção 3.5 faz as considerações finais sobre o capítulo.

3.1 COMPARAÇÃO DE DOCUMENTOS XML

A abordagem Phoenix faz a comparação de dois documentos XML em função dos seguintes itens:

- Nomes dos elementos;
- Conteúdo textual;
- Atributos e seus valores; e
- Subelementos.

Para cada item, foi criado um método de comparação com o objetivo de calcular o seu percentual de similaridade. O resultado de cada método é então multiplicado por um peso estabelecido pelo usuário e somado com os demais itens para informar a similaridade dos documentos comparados ao usuário. Caso o usuário não estabeleça o peso, um valor padrão pré-determinado pelo algoritmo é utilizado. A Seção 3.2 explica a ponderação dos itens de comparação citados e como eles interagem para calcular a similaridade total entre os documentos.

A abordagem Phoenix começa verificando se os nomes dos elementos são iguais. Para que se possa dar continuidade à comparação dos outros itens, os elementos devem ter nomes iguais. Caso contrário, o algoritmo entenderá que eles são diferentes e não haverá necessidade de comparar os outros itens. Sendo iguais, os outros itens de comparação serão executados normalmente. A partir deste ponto, cada um dos métodos podem ser executados paralelamente, já que o cálculo de um não influencia no resultado do outro. A Figura 18 representa um diagrama de atividades desta abordagem.

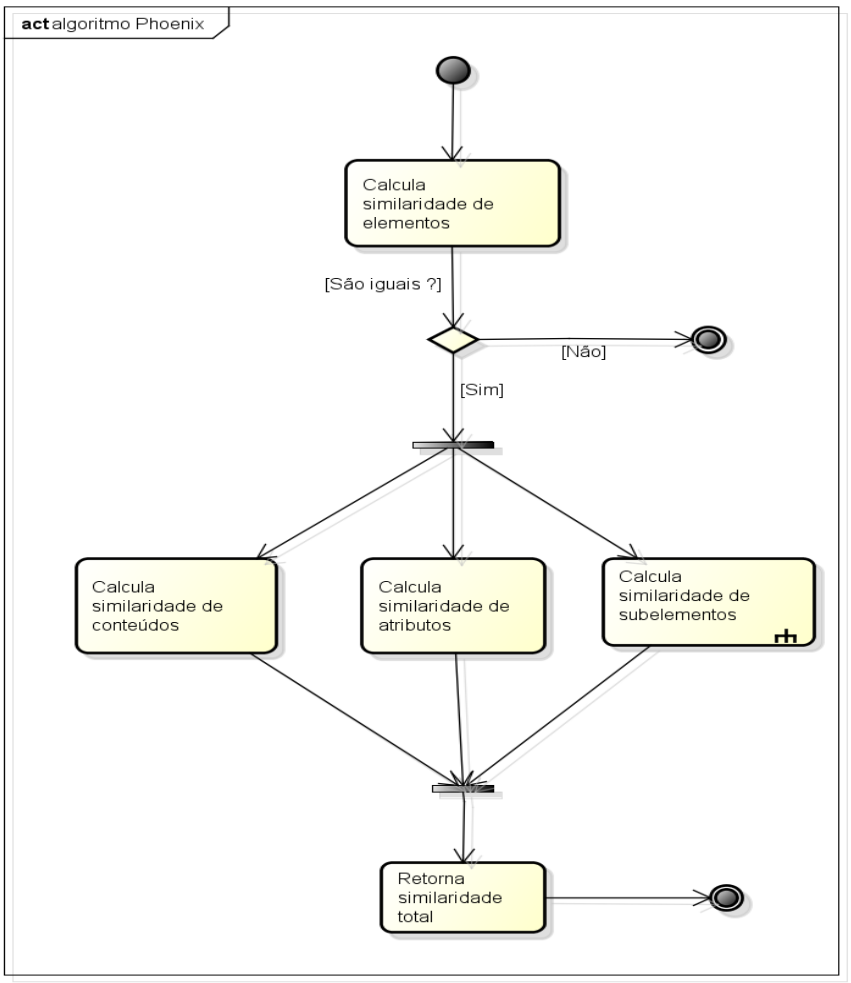

Figura 18: Diagrama de atividade do cálculo de similaridade de documentos XML

As Subseções seguintes discutem cada um dos itens de comparação que a Phoenix executa para calcular a similaridade dos documentos.

3.1.1 COMPARAÇÃO DE NOMES DOS ELEMENTOS

Conforme dito anteriormente, a primeira comparação que o algoritmo faz é pelo nome dos seus elementos. Se eles forem iguais, então o algoritmo continua a executar os demais itens. Caso contrário, é retornado zero por cento de similaridade (desigualdade). Considere o exemplo da Figura 19.

(a)	(b)
<**nome**>Carlos Eduardo</**nome**>	<**nome**>Eduardo da Silva</**nome**>

Figura 19: Exemplo de dois documentos XML com mesmo elemento

Como os nomes dos elementos são idênticos, o algoritmo continuará a executar os demais itens, conforme ilustra a Figura 18. Considerando o uso dos pesos atribuídos pelo próprio sistema (25% cada método), o resultado de similaridade até o momento é igual 25% (100% x 25%). Na seção 3.2, discute como a ponderação dos itens influencia no cálculo de similaridade.

3.1.2 COMPARAÇÃO DO CONTEÚDO TEXTUAL

Outra mensuração que o algoritmo realiza é a comparação do conteúdo dos seus elementos. O algoritmo extrai as informações de ambos os elementos correntes e verifica o percentual de similaridade entre elas. Para o cálculo de similaridade de conteúdo textual, foi utilizado o algoritmo LCS, já discutido no Capítulo 2.

Considerando o próprio exemplo da Figura 19, o algoritmo calcula a similaridade do conteúdo do elemento "nome" de ambos os documentos. Como o algoritmo LCS trabalha apenas com strings, retornando a maior sequência comum, foi preciso transformar esse retorno em valor percentual. Assim, o resultado de similaridade percentual obtido é igual ao tamanho da string encontrada pelo algoritmo LCS dividido pela metade do somatório dos tamanhos das strings comparadas. Neste exemplo, a maior sequência comum encontrada é "Eduardo", tendo 7 caracteres. A quantidade de caracteres da string da Figura 19(a) é igual a 14, enquanto Figura 19(b) é igual a 16. Portanto, o resultado encontrado entre "Carlos Eduardo" e "Eduardo da Silva" é igual a **7 / ((14 + 16) / 2)**, que é aproximadamente igual a 46,67%. Conforme dito anteriormente, este valor será multiplicado pelo peso atribuído pelo usuário, ou pelo próprio sistema.

3.1.3 COMPARAÇÃO DOS ATRIBUTOS E SEUS VALORES

Para examinar se os atributos dos elementos comparados são similares, este método busca todos os atributos que compõem os elementos e os compara. Se não existirem

atributos em ambos, o método retorna 100% de similaridade. Caso contrário, o algoritmo extrai todos eles junto com os seus respectivos conteúdos e calcula seu percentual. Considere o exemplo da Figura 20.

(a)

```
<artigo data="10-01-2012"
        versão="01">
      ...
</artigo>
```

(b)

```
<artigo data="10-01-2012"
        ultima_revisao="15-01-2012"
        versão="02">
      ...
</artigo>
```

Figura 20: Exemplo de dois documentos XML com quantidade de atributos diferentes

Como se pode notar, o documento da Figura 20(b) tem um atributo a mais que o da Figura 20(a). Em virtude disso, o algoritmo agrupa todos os atributos pertencentes a ambos os elementos e cria uma conjunto com todos eles. Com isso, mesmo que o atributo não apareça explicitamente no elemento, o algoritmo entenderá que este existe e que seu conteúdo é nulo. A figura 22 mostra como a abordagem Phoenix trata a comparação de atributos implícitos.

(a)

```
<artigo data="10-01-2012"
        ultima_revisao=""
        versão="01">
      ...
</artigo>
```

(b)

```
<artigo data="10-01-2012"
        ultima_revisao="15-01-2012"
        versão="02">
      ...
</artigo>
```

Figura 21: Exemplo de como a abordagem Phoenix trata a comparação de atributos implícitos

Depois de criar o conjunto, é feita uma verificação de seus conteúdos para computar o percentual de similaridade. Conforme a Seção 3.1.2, o percentual de similaridade de cada atributo é calculado com o uso do algoritmo LCS. Ao final, o resultado obtido é a média aritmética das similaridades de cada atributo. No caso do exemplo da Figura 20, têm-se os valores descritos na Tabela 4. Logo, a similaridade de atributos, sem a multiplicação pelo peso, é de 50%.

Atributos	(a)	(b)	Similaridade
Data	"10-01-2012"	"10-01-2012"	100%
ultima_revisao	""	"15-01-2012"	0%
Versão	"01"	"02"	50%
Resultado			150% / 3 = 50%

Tabela 4: Resultado do cálculo de similaridade dos atributos

3.1.4 COMPARAÇÃO DOS SUBELEMENTOS

O quarto e último método que o algoritmo executa é a comparação dos subelementos. O método isola todos os filhos de cada nó e os compara, um a um, até encontrar o conjunto com a maior similaridade. A comparação ocorre de modo recursivo, onde cada elemento é comparado utilizando os mesmos critérios exemplificados na Figura 18. A recursão é finalizada quando não houver mais filhos para comparar. A Figura 22 mostra o diagrama de atividades da comparação dos subelementos.

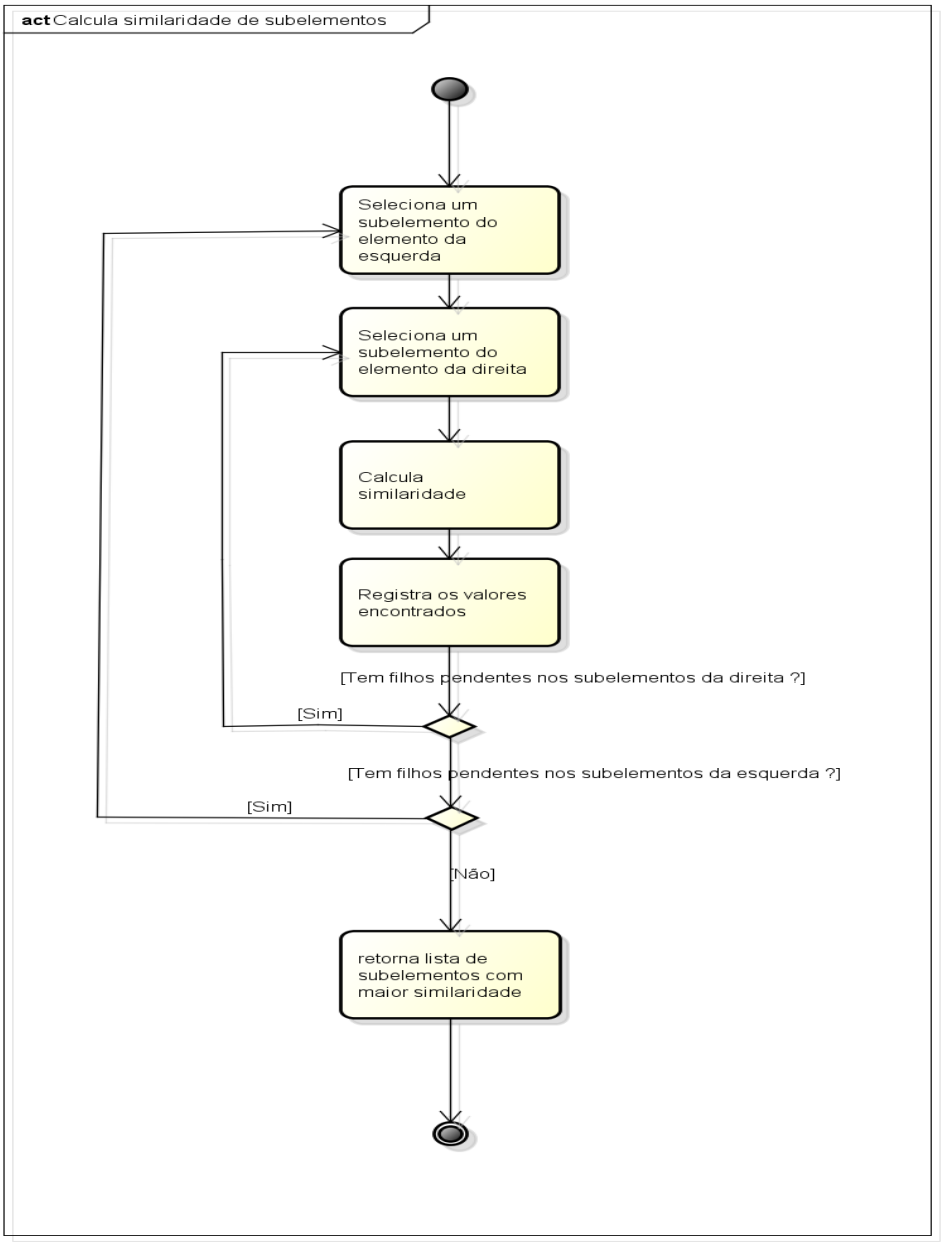

Figura 22: Diagrama de atividades da comparação de similaridade dos subelementos

O algoritmo monta uma matriz com todos os subelementos do nó esquerdo na primeira linha, e todos os subelementos do nó direito na primeira coluna. A comparação dos elementos pode ser vista como um produto cartesiano, onde cada elemento do nó esquerdo é comparado com cada elemento do nó direito. A cada interação, o resultado é armazenado nessa matriz para que futuramente possa ser percorrida com o objetivo de

achar o conjunto de elementos com maior similaridade. O algoritmo usado para processar esta matriz é o algoritmo Húngaro, já discutido anteriormente no Capítulo 2. A Figura 23 ilustra como funciona a comparação de similaridade entre os subelementos dos elementos correntes.

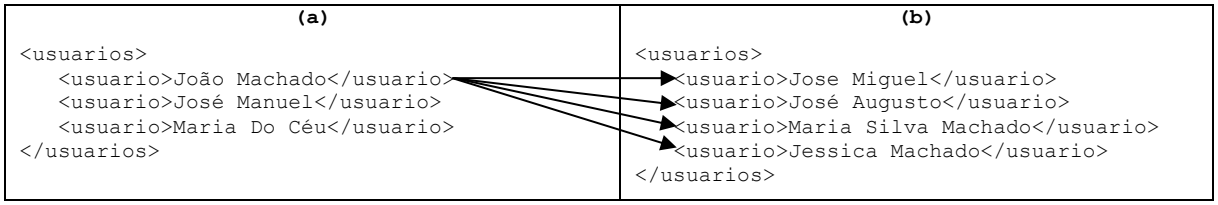

Figura 23: Exemplo da comparação dos filhos entre dois documentos XML

A Tabela 5 mostra todas as comparações feitas pelo algoritmo, junto com o seu percentual de similaridade. Neste exemplo, o documento da Figura 23(a) possui um elemento a menos que o da Figura 23(b). Para contornar esta situação, o método cria um elemento fictício para que seja possível calcular a maior similaridade no conjunto usando o algoritmo Húngaro. Com isso, a criação da tabela de similaridades entre os subelementos sempre será uma matriz quadrada.

Usuarios	usuario(Jose Miguel)	usuario(José Augusto)	usuario(Maria Silva Machado)	usuario(Jessica Machado)
usuario(João Machado)	83,7**%**	83,33 %	87,9%	**91,67**%
usuario(José Manuel)	**93,2**%	88,04**%**	80%	84,61%
usuario(Maria Do Céu)	81,52%	81,25%	**86,3%**	82,41 %
Linha0	0.0%	**0.0**%	0.0%	0.0%

Tabela 5: Resultado da comparação de similaridade entre os filhos do documento da Figura 23 (a) e Figura 23(b)

Conforme dito anteriormente, o resultado dessa tabela servirá de entrada para o algoritmo Húngaro. O resultado deste método é a similaridade do conjunto com maior percentual dividido pela ordem da matriz quadrada. Neste exemplo, têm-se os valores ilustrados na Tabela 6. Logo, a similaridade total dos subelementos é igual a 68,31%. É importante deixar claro que, apesar do exemplo, o Húngaro não é um algoritmo guloso. O algoritmo retorna um casamento ótimo dos elementos comparados, conforme discutido no Capítulo 2.

	usuario	usuario	Usuario	usuario	Resultado
Maior similaridade do conjunto	**93,2**%	**0.0**%	**86,3**%	**91,67**%	271,17%/ 4 = 67,8%

Tabela 6: Resultado similaridade dos subelementos comparados

3.1.5 CÁLCULO DA SIMILARIDADE DO DOCUMENTO

Um fator que contribui para a avaliação da similaridade entre os documentos é o limite inferior da comparação. Para muitos usuários, é importante visualizar apenas o quanto um documento é análogo ao outro. No entanto, se o resultado dessa comparação fosse abaixo do esperado, não haveria necessidade de informá-lo sobre sua similaridade. Assim, é aferido um limite (*threshold*) para que o resultado do cálculo de similaridade possa ser considerado aceito. Abaixo deste limite, o método considera este valor como zero, consequentemente interpretando que foram adicionados elementos distintos nos dois documentos. Esta regra também se aplica a todos os itens de comparação discutidos anteriormente.

A Seção 3.2 descreve uma forma de se obter a similaridade entre dois documentos XML especificando apenas o que é mais relevante para a diferenciação, através da ponderação de seus métodos de comparação.

3.2 PONDERAÇÃO DOS QUESITOS DE SIMILARIDADE

Como discutido anteriormente, um documento XML é composto pelos seguintes itens: elementos; atributos; conteúdo dos elementos (textual); e os subelementos. Desse modo, para se calcular a similaridade entre dois documentos, a abordagem Phoenix faz a comparação levando em consideração todos os quesitos citados acima. Cada um é calculado e somado com o quesito seguinte para obter uma similaridade total resultante, conforme foi visto na Seção 3.1. No entanto, para alguns domínios, alguns itens são mais relevantes do que outros. Para contemplar essa necessidade, foi criada uma ponderação de valores sobre esses itens, onde o usuário informa ao sistema o quão importante é aquele item para ele na comparação de similaridades.

Considere o exemplo de dois documentos XML representados na Figura 24. Assuma que o usuário tenha colocado os seguintes parâmetros para o cálculo de similaridade, conforme descrito na Figura 25.

(a)

```
<usuario nome = "João" idade = "12" >
    <endereço>Ruaqualquer123</endereço>
</usuario>
```

(b)

```
<usuario nome = "José" idade = "15" >
    <CPF>000.000.000-00</CPF>
    <identidade>12345</identidade>
    <endereço>Rua qualquer 123</endereço>
 </usuario>
```

Figura 24: Comparação de dois documentos XML

```
Similaridade dos nomes dos elementos: 40%
Similaridade dos atributos: 40%
Similaridade dos conteúdos dos elementos: 15%
Similaridade dos filhos dos elementos: 5%
```

Figura 25: Exemplo pesos atribuídos pelos usuários

Seguindo o diagrama de atividades expostas na Figura 18 da Seção 3.1, é feita a primeira comparação do algoritmo. Como os dois documentos possuem o mesmo elemento "usuario" então o método retorna 100% de similaridade (igualdade) no nome do elemento. Embora se tenha achado uma igualdade entre eles, o método ainda não calculou o seu resultado final. É necessário que o valor de similaridade encontrado por ele seja multiplicado pelo peso atribuído pelo usuário. De acordo com os valores assumidos na Figura 25, a similaridade dos nomes dos elementos é igual a 40%. Então, a similaridade total encontrada até o momento pelo algoritmo é de 40% (100% x 40%).

A seguir, cada um dos métodos podem ser executados paralelamente de acordo com a Figura 18. Seguindo a ordem, o próximo método de comparação é o da similaridade dos atributos. Ambos os documentos possuem dois atributos (nome e idade), porém seus conteúdos são diferentes. Usando a estratégia da Subseção 3.1.3, é calculado o percentual de similaridade do conteúdo dos atributos. A Tabela 7 ilustra o resultado dos cálculos obtidos.

	Lado esquerdo	Lado direito	LCS	Resultado
Nome	João	José	Jo	2 / ((4 + 4)/2) = 50%
Idade	12	15	1	1 / ((2 + 2)/2) = 50%
Total				(50% + 50%) / 2 = 50%

Tabela 7: Resultado da similaridade dos atributos do elemento “usuario”

Logo, o cálculo da similaridade dos atributos é igual a 20% (50% x 40%). O resultado é somado com a similaridade total encontrada até o momento, que no caso, é igual a 40%. Assim, têm-se 60% (40% + 20%) de similaridade.

O próximo método a ser executado é o da similaridade dos conteúdos dos elementos. Como o elemento “usuario” não possui conteúdo em ambos os documentos, o método retornará 100%. Assim, o valor obtido por este método é igual a 15% (100% x 15%), que somados com a similaridade corrente é igual a 75% (60% + 15%).

O último método de comparação é o da similaridade dos filhos dos elementos. O primeiro e único filho do elemento “usuario” do documento da Figura 24(a), o elemento “endereço” do lado esquerdo, é comparado com todos os filhos do outro elemento (CPF, identidade e endereço). Em consonância com que foi ilustrado na Figura 18, o método só irá prosseguir com os outros itens de comparação se o nome dos elementos forem iguais. Portanto, há apenas um elemento com o mesmo nome, o elemento “endereço”. Para calcular o percentual de similaridade desse elemento, é preciso usar os mesmos passos descritos anteriormente de modo recursivo. O “endereço” possui mesmo nome de elemento, o mesmo conteúdo, não possuem atributos e não há filhos. A Tabela 8**Error! Reference source not found.** mostra a matriz que servirá de entrada para o algoritmo Húngaro.

	CPF	Identidade	Endereço(nó direito)
Endereço(nó esquerdo)	0.0%	0.0%	100%
linha0	0.0%	0.0%	0.0%
linha1	0.0%	0.0%	0.0%

Tabela 8: Matriz gerada para o processar o maior similaridade do conjunto do elemento “usuario”

Como a matriz é de ordem 3, de acordo com o a Subseção 3.1.4, o percentual de similaridade encontrado pelo método é igual a 33,33% (100% / 3).A Tabela 9 mostra a

comparação de cada item de similaridade com os seus respectivos pesos atribuídos pelo usuário de acordo com a Figura 25.

Método de comparação	Percentual encontrado	Peso	Similaride
Por elementos	100%	40%	40%
Por atributos	50%	40%	20%
Por conteudos	100%	15%	15%
Por subelementos	33,33%	5%	1,67%
Resultado			76,67%

Tabela 9: Resultado final da comparação de cada método com os pesos

O percentual de similaridade encontrado do elemento raiz é o somatório das similaridades encontrados em cada um dos métodos. Assim, o valor do elemento "usuario" da figura 25 é igual a 76,67% (40% + 20% + 15% + 1,67%).

É importante deixar claro que esse percentual nunca deverá passar de 100%. Isto porque as ponderações atribuídas a cada item é efetuado de tal maneira que o seu somatório será sempre igual a 100%. Se isso não for obedecido, o cálculo pode gerar resultados insconsistentes do tipo 150% de similaridade.

O peso padrão atribuído pelo sistema, destacado no iníco desta Subseção, é proporcional à quantidade de métodos de comparação discutidos na Seção 3.1. Como são quatro métodos de comparação, cada método possui 25% de importância no cálculo de similaridade. Portanto, fica a critério do usuário a escolha do que é mais relevante através da ponderação dos quesitos de comparação de similaridades citados acima.

3.3 INTERFACE COM O USUÁRIO

Como foi abordado no Capítulo 2, não foi encontrado nenhum algoritmo de diferenciação de documentos XML que possa ilustrar as diferenças entre eles. A abordagem Phoenix propõe a construção de uma interface que seja integrada aos algoritmos de diferenciação abordados na Seção 3.1. Para isso, é necessário estabelecer como um documento XML será representado graficamente.

Para ilustrar como o Phoenix codifica um documento XML em uma interface gráfica, considere o exemplo da Figura 27, uma representação em formato de árvore do documento XML da Figura 26.

```
<empregados>
   <empregado cod="C01" dept="D01">
      <nome>João</nome>
      <sobrenome>Silva</sobrenome>
   </empregado>
   <empregado cod="C02" dept="D01">
      <nome>José</nome>
      <sobrenome>Pires</sobrenome>
   </empregado>
</empregados>
```

Figura 26: Exemplo de XML para criar uma interface gráfica

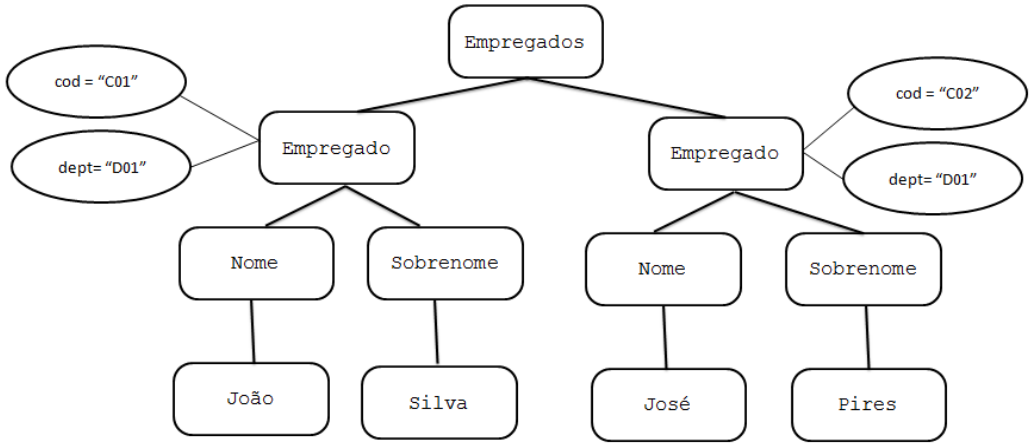

Figura 27: Apresentação gráfica de um documento XML

Para diferenciar uma estrutura da outra, foram criadas cores pré-estabelecidas a fim de que o usuário possa identificar semelhanças e diferenças de forma mais suave. Ao comparar dois documentos XML, cada um deve ser representado com uma cor diferente. Para essa representação, foram escolhidas as cores vermelho e verde. As cores vermelhas representam todos os elementos do documento XML da esquerda, enquanto as cores verdes representam todos os elementos do documento XML da direita da comparação. Os elementos que forem similares são representados por uma variação de cores que vai desde o cinza mais escuro, até o branco absoluto. O que irá ditar esta variação é o grau de similaridade que o dado elemento possuir. Logo, quanto mais similar o elemento for, mais próximo da cor branca ele será.

3.4 COMPARAÇÃO DAS VERSÕES

Para a comparação de duas versões de um documento XML, o usuário poderá ter duas concepções diferentes: a revisão de um documento, ou a variância desse documento. A revisão tem como objetivo verificar o quanto um documento foi modificado. Assim, o usuário que visualiza o "diff" feito pela abordagem Phoenix poderá interpretar que as cores vermelhas representam elementos que estavam na versão anterior e que agora foram removidos, enquanto que as cores verdes representam os elementos que foram adicionadas ao documento. A variância, por outro lado, tem como intenção mostrar o quanto um documento é diferente do outro. Com isso, as cores vermelhas representam os elementos que estão em um documento, enquanto as cores verdes representam os elementos que estão no outro documento.

3.5 CONSIDERAÇÕES FINAIS

Neste capítulo foram discutidos os métodos de comparação entre documentos XML para que a abordagem Phoenix possa identificar as similaridades e discrepâncias entre eles. É importante salientar que estes métodos não são os únicos responsáveis pela verificação de semelhança entre os documentos. A ponderação dos itens também é outro fator que contribui para o cálculo de similaridade.

Por último, foi discutido como abordagem interpreta um documento XML e como o resultado é exposto para o usuário. Com isso, a comparação de duas versões de documentos XML poderá ser interpretada como uma revisão ou como uma variância.

No Capítulo 4, será discutido como o Phoenix agrupa todos os assuntos abordados neste capítulo. O capítulo também apresenta uma introdução às ferramentas usadas para viabilizar a sua construção.

CAPÍTULO 4 - IMPLEMENTAÇÃO

Apesar de existirem outras linguagens de programação que permitiriam o desenvolvimento deste trabalho, a linguagem Java foi escolhida, principalmente por possuir uma quantidade considerável de bibliotecas e pelo próprio conhecimento da equipe.

O framework que se mostrou mais prático para a representação dos dados de forma gráfica foi o Jung (JUNG, 2003), que possui componentes de tela capazes de ilustrar os elementos dos documentos XML de forma hierárquica. O framework Jung, pela sua própria definição, "é uma biblioteca de software que fornece uma linguagem comum e extensível para modelagem, análise e visualização dos dados que podem ser representados como um grafo ou rede" (JUNG, 2003).

Para ser possível mostrar os elementos de ambos os documentos de forma a ilustrar os conflitos e igualdades entre eles, é necessário que os mesmos possam ser manipulados. Entre os frameworks disponíveis em Java para que isto ocorra, foi escolhido o Dom4j API.

Este capítulo descreve como o Phoenix foi elaborado. A Seção 4.1 mostra o funcionamento do Phoenix com todos os frameworks e estratégias discutidas até o momento. A Seção 4.2 faz uma comparação do Phoenix com outras ferramentas existentes no mercado e no meio científico.

4.1 FERRAMENTA PHOENIX

A execução da ferramenta Phoenix inicia com duas áreas de texto em branco para que o usuário insira os documentos XML a serem comparados. A Figura 28 mostra a tela inicial do sistema. Ao clicar no botão "comparar", o usuário tem a possibilidade de visualizar os conflitos identificados nos documentos comparados de forma gráfica, além de enxergar o quanto um documento é similar ao outro, conforme ilustrado na Figura 29.

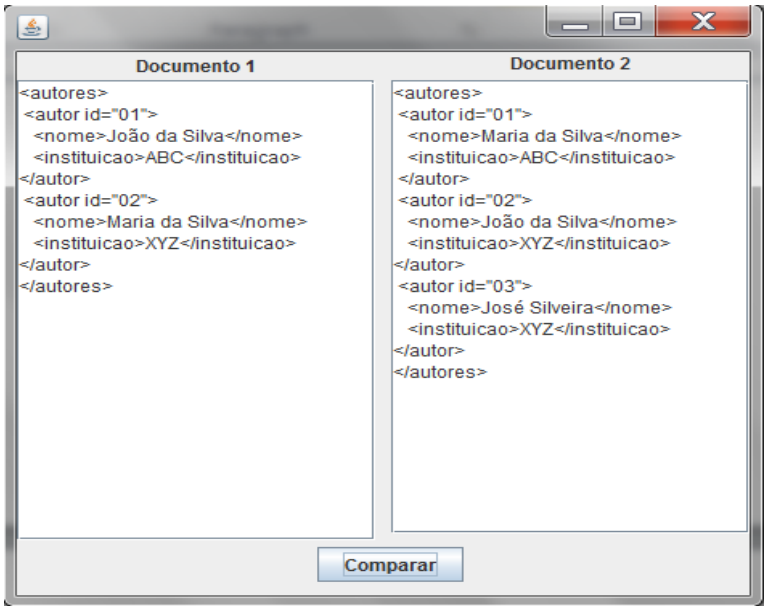

Figura 28: Tela inicial do sistema Phoenix.

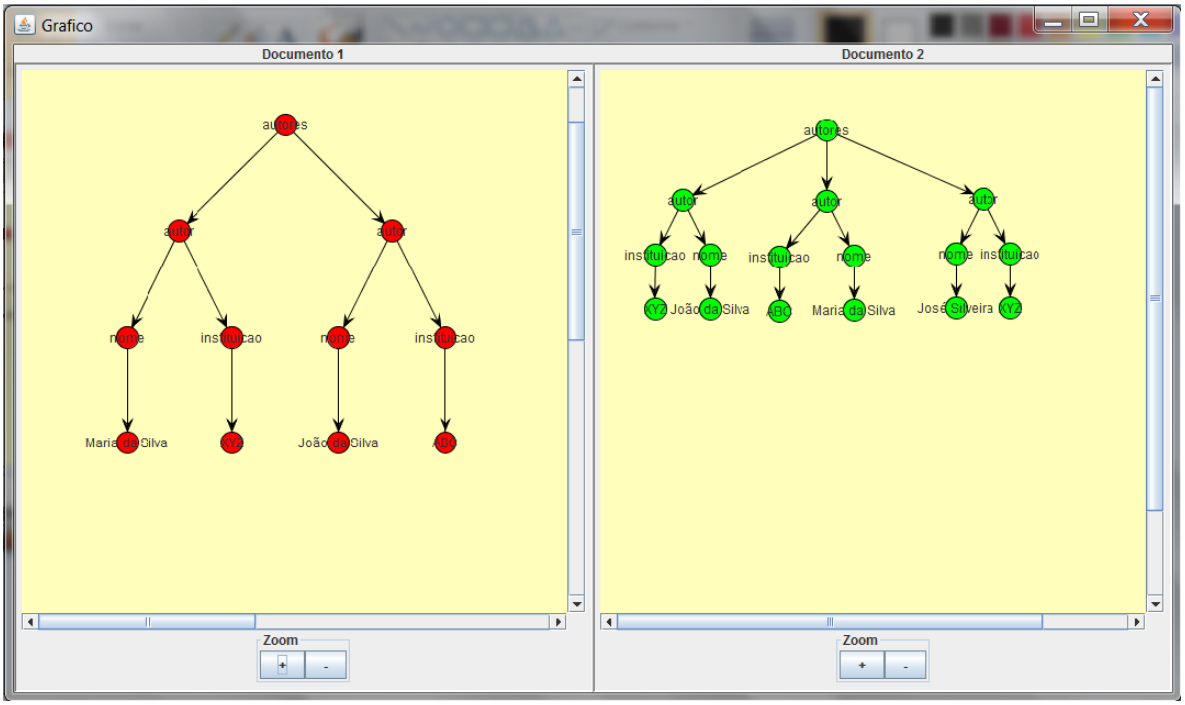

Figura 29: Gráfico que mostra os documentos XML escritos pelo usuário na tela inicial.

Os nós do documento XML à esquerda da tela são representados pela cor vermelha, enquanto os da direita da tela são representados pela cor verde. Ao fazer a comparação entre os documentos, o sistema retorna o valor total de similaridade entre eles, informando quais nós são iguais, e quais são diferentes. As cores usadas para demonstrar os elementos iguais, no entanto, diferem das cores acima, conforme mencionado no Capítulo 3. A cada nó detectado como sendo diferente, o sistema apresenta para o usuário o percentual de similaridade que eles têm entre si ao passar o mouse sobre o elemento. O resultado da comparação pode ser visto na Figura 30.

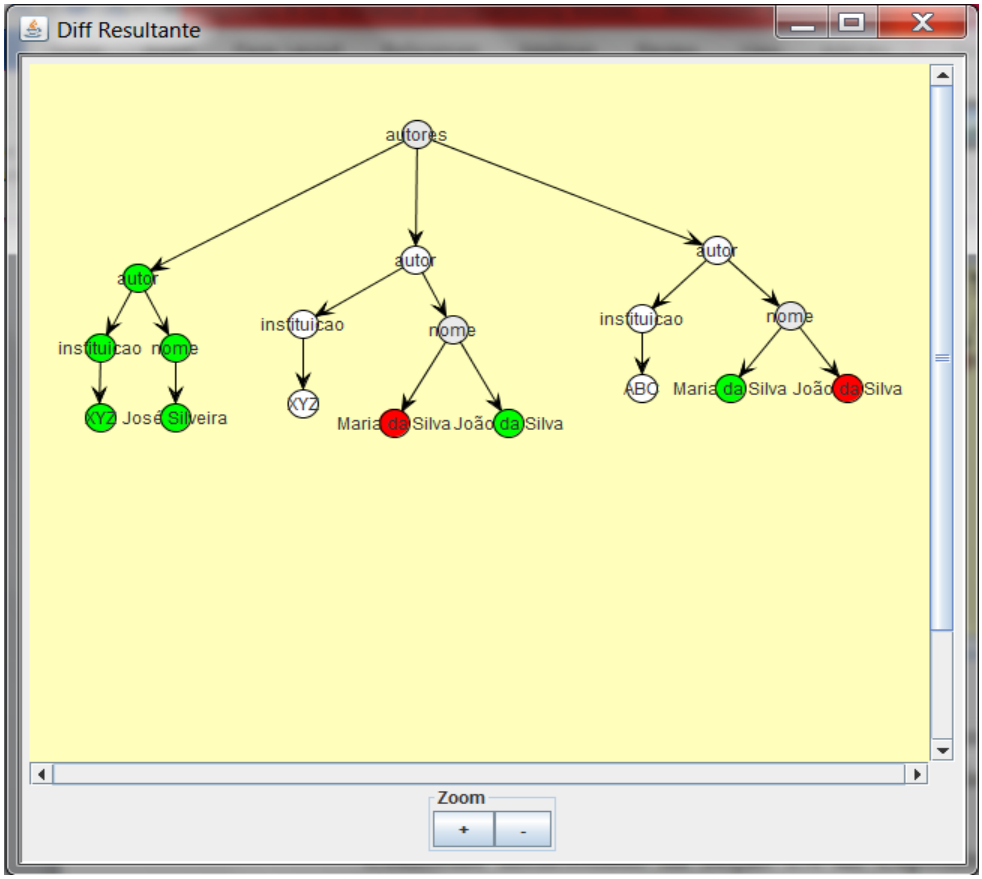

Figura 30: Resultado da diferenciação dos documentos.

Conforme mencionado na Seção 3.4 do Capítulo 3, o usuário pode interpretar a Figura 30 de duas formas diferentes. Os documentos podem ser revisões ou variantes. Sendo o documento 2 uma revisão do documento 1, entende-se que os elementos em verde

foram acrescentados e os elementos em vermelho foram removidos. Como variantes, a cor verde pode ser interpretada como os elementos que pertencem ao documento 2 e a cor vermelha como os elementos que pertencem ao documento 1.

Nesta implementação foi considerado apenas a similaridade dos elementos pertencentes aos documentos XML, sem se preocupar com a ordem na qual estes aparecem. Com isso, a ordem não é um fator crítico de similaridade, embora para algumas aplicações, este seja um requisito imperativo para análise dos resultados. A Seção 4.2 explora os prós e contras do sistema Phoenix fazendo um comparativo com as ferramentas discutidas no Capítulo 2 .

4.2 COMPARAÇÃO DO PHOENIX COM OUTRAS FERRAMENTAS

A Phoenix possui um diferencial que é o cálculo de similaridade entre os elementos dos documentos XML, mas possui algumas restrições. No Capítulo 2, algumas ferramentas foram analisadas e a Tabela 3 foi gerada. A Tabela 10 utiliza essa análise para comparar as caracterísicas então discutidas com a ferramenta Phoenix.

	DiffDog	ExamlXML Pro	Delta XML	oXygen	Phoenix
Visualizar o casamento de forma gráfica	Sim	+	+	+	Sim
Comparar de forma textual, onde o usuário pode editar um documento	Sim	Não	Não	Sim	Sim
Comparar diretórios	Sim	Não	Não	Sim	Não
Navegar entre os conflitos	Sim	Sim	Sim	Sim	Sim
Informar similaridade	Não	Não	Não	Não	Sim
Tratar árvore ordenadas	Não	Não	Não	Não	Não

Tabela 10: Ferramentas de Visualização de Diferenças

O Capítulo 5 discute as contribuições, limitações e propostas futuras de desenvolvimento da ferramenta Phoenix.

CAPÍTULO 5 - CONCLUSÃO

5.1 CONTRIBUIÇÕES

No Capítulo 2, foram descritas e comparadas algumas das ferramentas de análise de documentos XML existentes no mercado, evidenciando suas principais funcionalidades. De acordo com a Tabela 10, a ferramenta Phoenix contribui com as seguintes características:

- Visualização dos documentos de forma gráfica, por meio de grafos;
- Visualização da diferença entre dois documentos de forma gráfica, também por meio de um grafo, usando uma notação de cores;
- Informação de quanto um documento é similar ao outro através de um valor percentual; e
- Capacidade de comparar árvores, onde a ordem em que os elementos aparecem não é relevante para a definição de similaridade.

Além disso, o Phoenix faz uma análise considerando outros quesitos (nome dos elementos, atributos, conteúdos e subelementos) para informar se um elemento é similar ou não ao outro. Assim, é possível atingir uma maior precisão de similaridade entre os documentos já que, dentre as ferramentas estudadas até aqui, nenhuma outra leva em consideração esse tipo de comparação.

5.2 LIMITAÇÕES

A versão atual da Phoenix ainda não consegue verificar se um documento XML é bem formado, ou não. Portanto, para que possa ser executada uma comparação no sistema, os documentos já devem estar bem formados.

Para algumas aplicações, a ordem em que os elementos aparecem é relevante. No entanto, a Phoenix não faz a comparação de similaridade levando em consideração este quesito.

Outra limitação desta versão do Phoenix é que, apesar de possuir uma interface gráfica para melhorar a identificação dos conflitos, a visualização não se mostra confortável para documentos XML grandes.

5.3 TRABALHOS EM ANDAMENTO

O sistema Phoenix está limitado à comparação de apenas dois documentos XML por vez. Em uma extensão do Phoenix desenvolvida por Gleiph Ghiotto, foi possível comparar vários documentos XML e obter um agrupamento de similaridade, onde o sistema informa o quanto um grupo de documentos é similar aos outros. O exemplo da Figura 31 ilustra esse agrupamento.

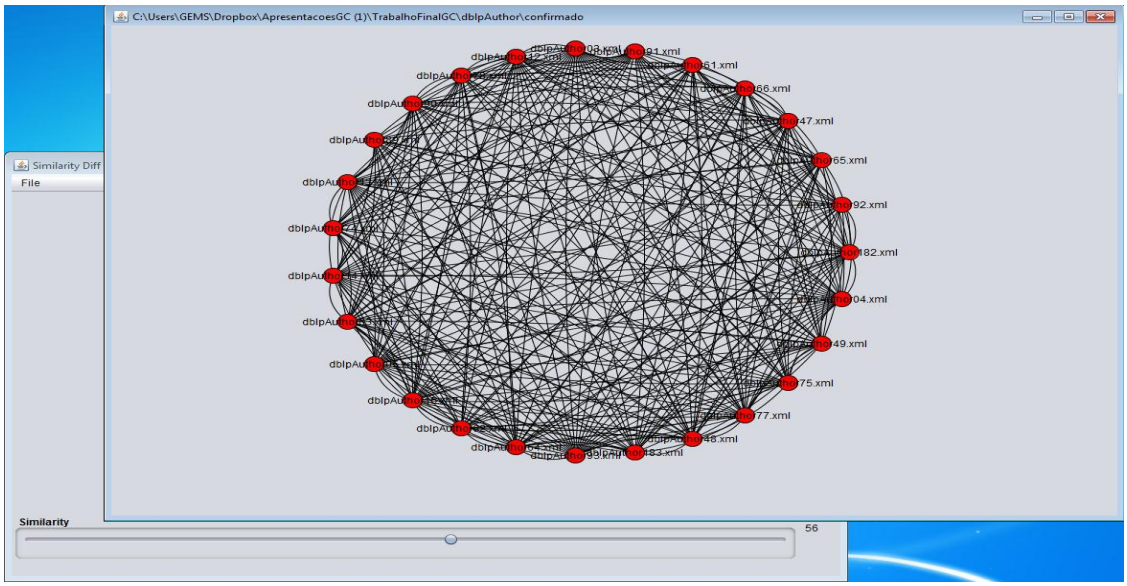

Figura 31: Clusterização com limiar de similaridade baixo (56%).

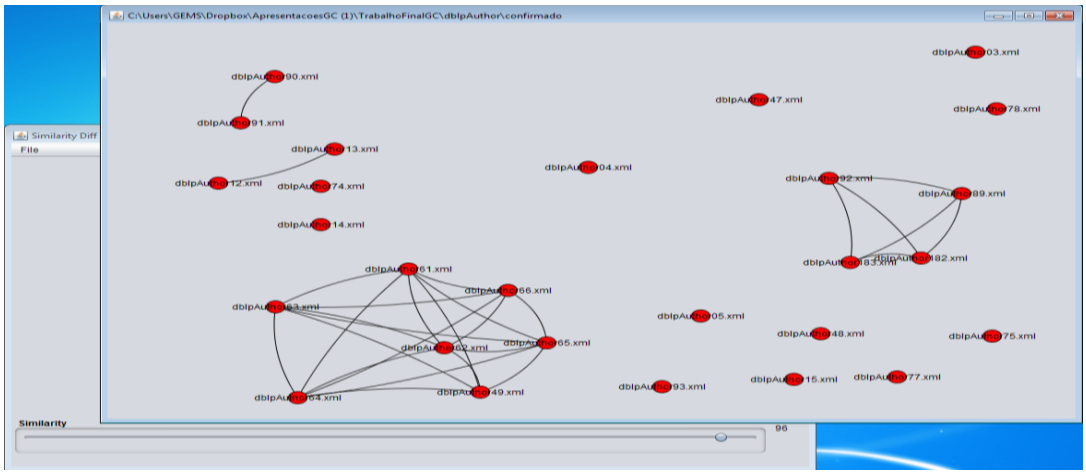

Figura 32: Clusterização com limiar de similaridade alto (96%).

Um limite mínimo de similaridade é adotado entre os documentos do grupo. Cada documento XML comparado está representado no grafo como um nó e, cada aresta entre dois nós (documentos) representa uma similaridade, acima do limite mínimo, entre eles. Sendo assim, os documentos que não possuem nenhuma aresta não possuem uma similaridade mínima com nenhum dos outros documentos.

5.4 TRABALHOS FUTUROS

Como discutido anteriormente no Capítulo 3, os métodos de comparação podem ser executados em paralelo sem que isto afete o resultado final do sistema. Logo, para reduzir a carga computacional envolvida no processo, pode-se utilizar programação paralela para que o cálculo da similaridade de qualquer um dos três métodos – textual, atributos e subelementos – seja feita concomitantemente. Ao final, seus resultados seriam somados com todos os demais e o sistema apresentaria o resultado final da comparação. Essa abordagem é útil para cálculos de similaridades de grandes documentos.

Atualmente, o sistema Phoenix só permite que os usuários possam visualizar os conflitos detectados entre os documentos XML, não sendo possível mesclar suas estruturas. Dessa forma, se o usuário desejar alterar os elementos conflitantes, terá que executar a mesclagem externamente. Isto pode ser uma tarefa dispendiosa para grandes arquivos,

podendo ocasionar inconsistência dos dados ou até mesmo documentos mal formados. Uma solução seria mesclar os elementos à medida que o usuário percorresse o XML resultante.

Como é sabido, vários sistemas de controle de versão utilizam algoritmos de diferenciação. No entanto, atualmente o sistema Phoenix não permite integrar suas funcionalidades em outros sistemas. Assim, um mecanismo de integração deve ser implementado para que outras ferramentas possam utilizar os critérios de comparação discutidos no Capítulo 3. Como resultado desse trabalho, seriam disponibilizados para os usuários dois tipos de arquivos: a ferramenta Phoenix, onde o usuário poderia manipular os dados diretamente através de uma interface gráfica bem definida, e uma biblioteca, onde outros sistemas poderiam se beneficiar com quesitos abordados ao longo deste trabalho.

REFERÊNCIAS

A7Soft, (2003), A7Soft provides tools to compare XML documents. Disponível em: <"http://www.a7soft.com/">. Acesso em: 4 abr 2012.

Achard, F. and Vaysseix, G. and Barillot, E., (2001), "XML, bioinformatics and data integration", *Bioinformatics*, v. 17, n. 2 (jan.), p. 115-125.

Altova DiffDog, (2012), Download Altova DiffDog Enterprise Edition. Disponível em: <"http://www.altova.com/download/diffdog/diff_merge_tool_enterprise.html">. Acesso em: 3 abr 2012.

C. Santos, R. and Hara, C., (2007), *Utilização de Chaves em Algoritmos de Detecção de Diferenças para XML*

Cederqvist, P., (2003), *Version Management with CVS*, Free Software Foundation.

Cobena, G. and S. Abiteboul and Marian, A., (2002), "Detecting Changes in XML Documents". In: *International Conference on Data Engineering (ICDE)*, p. 41-52, San Jose, CA, USA.

Collins-Sussman, B. and Fitzpatrick, B. W. and Pilato, C. M., (2008), *Version Control with Subversion*. Sebastpol, CA, USA, O'Reilly Media.

Cormen, T. H. and Leiserson, C. E. and Rivest, R. L. and Stein, C., (2009), *Introduction to Algorithms*. third edition ed. The MIT Press.

Curbera and D. A. Epstein, (1999), "Fast Difference and Update of XML Documents". In: *Fast Difference and Update of XML Documents*

DeltaXML Compare, (2010), *DeltaXML Compare*.

Estublier, J., (2000), "Software Configuration Management: a Roadmap". In: *The Future of Software EngineeringInternational Conference on Software Engineering (ICSE)*, p. 279-289, Limerick, Ireland.

JUNG, (2003), JUNG, Java Universal Network/Graph Framework. *JUNG, Java Universal Network/Graph Framework*. Disponível em: <"http://jung.sourceforge.net/">.

Kuhn, H.W., (2005), "The Hungarian method for the assignment problem", *Naval Research Logistics*, v. 52, n. 1 (fev.), p. 7-21.

oXygen, (2002), <oXygen/> XML Diff & Merge. Disponível em: <"http://www.oxygenxml.com/xml_diff_and_merge.html#new-version">. Acesso em: 3 abr 2012.

Wang, Y. and De Witt, D. J. and Cai, J., (2003), "X-Diff: An Effective Change Detection Algorithm for XML Documents". In: *International Conference on Data Engineering (ICDE)*, p. 519-530, Bangalore, India.

GNU Diffutils. *GNU Diffutils*. Disponível em: <"http://www.gnu.org/software/diffutils/">. Acesso em: 25 ago 2012.

APÊNDICE A

Nas figuras abaixo, é demostrado as variações de dois documentos XML a fim de estudar o comportamento das ferramentas citados no Capítulo 2.

(a)

```
<autores>
 <autor id="01">
   <nome>João da Silva</nome>
   <instituicao>ABC</instituicao>
 </autor>
 <autor id="02">
   <nome>Maria da Silva</nome>
   <instituicao>XYZ</instituicao>
 </autor>
</autores>
```

(b)

```
<autores>
 <autor id="01">
   <nome>Maria da Silva</nome>
   <instituicao>ABC</instituicao>
 </autor>
 <autor id="02">
   <nome>João da Silva</nome>
   <instituicao>XYZ</instituicao>
 </autor>
 <autor id="03">
   <nome>José Silveira</nome>
   <instituicao>XYZ</instituicao>
 </autor>
</autores>
```

Figura 33: Documentos XML com todos itens(elemento, conteúdo textual, atributos e subelementos)

(a)

```
<autores>
 <autor>
   <nome>João da Silva</nome>
   <instituicao>ABC</instituicao>
 </autor>
 <autor>
   <nome>Maria da Silva</nome>
   <instituicao>XYZ</instituicao>
 </autor>
</autores>
```

(b)

```
<autores>
 <autor>
   <nome>Maria da Silva</nome>
   <instituicao>ABC</instituicao>
 </autor>
 <autor>
   <nome>João da Silva</nome>
   <instituicao>XYZ</instituicao>
 </autor>
 <autor>
   <nome>José Silveira</nome>
   <instituicao>XYZ</instituicao>
 </autor>
</autores>
```

Figura 34: Documentos XML sem os atributos

(a)

```
<autores>
 <autor id="01">
   <nome>João da Silva</nome>
   <instituicao>ABC</instituicao>
 </autor>
 <autor id="02">
   <nome>Maria da Silva</nome>
   <instituicao>XYZ</instituicao>
 </autor>
</autores>
```

(b)

```
<autores>
<autor id="03">
   <nome>José Silveira</nome>
   <instituicao>XYZ</instituicao>
 </autor>
<autor id="01">
   <nome>Maria da Silva</nome>
   <instituicao>ABC</instituicao>
 </autor>
 <autor id="02">
   <nome>João da Silva</nome>
   <instituicao>XYZ</instituicao>
 </autor>
</autores>
```

Figura 35: Documentos XML com todos os itens da Figura 33 em ordem diferente