Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Bacharelado em Ciência da Computação

XChangeSim: Compreensão de Mudanças de Documentos XML baseada em Similaridade

Danúbia Vieira Dias de Oliveira

JUIZ DE FORA
MARÇO, 2013

XChangeSim: Compreensão de Mudanças de Documentos XML baseada em Similaridade

Danúbia Vieira Dias de Oliveira

Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Departamento de Ciência da Computação

Bacharelado em Ciência da Computação

Orientador: Alessandreia Marta de Oliveira

JUIZ DE FORA

MARÇO, 2013

XChangeSim: Compreensão de Mudanças de Documentos XML baseada em Similaridade

Danúbia Vieira Dias de Oliveira

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

Alessandreia Marta de Oliveira
M. Sc. (UFRJ)

Jairo Francisco de Souza
D. Sc. (PUC - RJ)

Victor Stroele de Andrade Menezes
D. Sc. (UFRJ)

JUIZ DE FORA

26 DE MARÇO, 2013

Aos meus amigos e irmãos.

Aos pais, pelo apoio e sustento.

Resumo

As aplicações se apoiam cada vez mais na linguagem XML para representar dados semiestruturados e, consequentemente, uma grande quantidade de documentos XML é gerada. Um problema relacionado é que os documentos XML evoluem ao longo do tempo. Desta forma, o controle de mudanças em documentos XML é uma necessidade cada vez mais crescente. Diante disso, este trabalho apresenta o XChangeSim, uma abordagem para a compreensão das mudanças efetuadas em versões de documentos XML baseada em similaridade. A proposta é capaz de detectar mudanças semânticas entre duas versões de um documento XML e possibilita inferir a intenção do usuário ao criar a segunda versão de um documento.

Palavras-chave: Controle de Mudanças, Inferência, Similaridade, XML

Agradecimentos

Agradeço a Deus pelas inúmeras oportunidades e o sucesso no curso. Aos meus parentes e amigos pelo incentivo e presença.

Aos meus pais, José e Nilse, por todo amor e confiança nas escolhas feitas. À minha irmã, Anamália, pela cumplicidade em todos os momentos.

À professora Alessandreia pela orientação, dedicação, amizade e principalmente, pela paciência, sem a qual este trabalho não se realizaria.

Agradeço aos amigos do Grupo de Educação Tutorial da Computação(GETComp UFJF), em especial ao Carlos Oliveira, Célio Larcher, Guilherme Martins e Márcio Júnior, pelo apoio durante o desenvolvimento deste trabalho.

Aos professores do Departamento de Ciência da Computação pelos seus ensinamentos e aos funcionários do curso, que durante esses anos, contribuíram de algum modo para o nosso enriquecimento pessoal e profissional.

"Queira! Basta ser sincero e desejar profundo, você será capaz de sacudir o mundo".

Raul Seixas (Tente outra vez)

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

AFD	Autômato Finito Determinístico
DCC	Departamento de Ciência da Computação
DOM	Document Object Model
DTD	Document Type Definition
GCS	Gerência de Configuração de Software
LCS	Longest Common Subsequence
SCM	Sistema de Controle de Mudanças
UFJF	Universidade Federal de Juiz de Fora
XML	eXtensible Markup Language

1 Introdução

A utilização da Internet vem aumentando progressivamente no que diz respeito ao número de usuários e quantidade de informação disponível em formato digital. Um dos formatos mais conhecidos é o XML (eXtensible Markup Language), usado para troca de dados na *Web*.

A linguagem XML foi originalmente projetada para atender os desafios da publicação eletrônica em larga escala e ganhou grande espaço, pois uma vez pré-determinada a estrutura do documento, facilita o intercâmbio de dados entre sistemas de informação (BRAY et al., 2008). Entretanto, os documentos XML possuem estruturas heterogêneas e devido a isso, o grande desafio está na integração dos dados representados e na verificação de semelhança entre eles. O processamento da equivalência não é uma tarefa trivial (DING, 2002).

Provar tal equivalência consiste na busca por similaridade entre os documentos. Esse tema vem sendo alvo de pesquisas em grandes áreas da Ciência da Computação, como por exemplo em Banco de Dados e Inteligência Artificial. Dentro da Engenharia de Software, as pesquisas de similaridade podem gerar resultados a fim de apoiar a gerência de mudanças ocorridas em um projeto.

O controle de mudanças procura acompanhar uma alteração desde o momento em que a necessidade de mudança é reconhecida e solicitada, até a disponibilização da nova versão do item modificado. Durante todo o processo, cada alteração é documentada a fim de aumentar a compreensão da evolução de determinado item, e do projeto como um todo (GARCIA, 2012). Neste contexto, estudar técnicas mais eficientes tem como objetivo, dentre outros fatores, inferir significados às mudanças realizadas durante o projeto.

1.1 Motivação

A linguagem XML oferece um método simples e ao mesmo tempo poderoso para a representação de documentos. Devido a simplicidade de sua estrutura, documentos XML

podem sofrer alterações constantes, por isso a necessidade de controlar sua evolução.

Apesar de muitas abordagens apontarem modificações estruturais entre versões de um documento, ainda há uma ausência de técnicas eficientes para compreender, semanticamente, as informações resultantes do controle de mudanças. O uso de similaridade nestas técnicas, torna-se uma alternativa para produzir resultados mais precisos e com caráter semântico.

1.2 Justificativa

Existem muitos algoritmos que buscam gerenciar a evolução de documentos XML, porém grande parte destes avaliam apenas o controle sintático. Compreender as mudanças realizadas em um documento resulta no controle semântico, ou seja, atribuir significados as modificações ocorridas.

No contexto dos documentos XML, a organização hierárquica dos dados facilita o controle estrutural e semântico. Para gerenciar este tipo de dados, uma alternativa eficiente é utilizar técnicas de similaridade, a fim de verificar a equivalência de conteúdo entre as versões de um documento. Desta forma, é possível recuperar informações referentes ao contexto de negócio dos documentos analisados.

1.3 Objetivo

Esta pesquisa tem como objetivo compreender as alterações realizadas em documentos XML, apoiando-se no uso de similaridade. Para isto, é proposta uma abordagem, denominada XChangeSim, que busca a semelhança entre os conteúdos de duas versões de um documento XML, no contexto do projeto XChange. Verificar a equivalência com precisão, garante melhores casamentos entre as *tags* das versões analisadas. Desta forma, pode-se encontrar os elementos correspondentes e inferir o que mudou ao longo do processo de evolução do documento.

1.4 Organização do trabalho

Este trabalho está organizado em quatro capítulos, além desta introdução. No Capítulo 2 são apresentados os conceitos para compreensão desta pesquisa. São eles: os tipos de análise de similaridade estrutural e semântica, o algoritmo LCS, o controle de mudanças em âmbito geral e suas características específicas para documentos XML e, por fim, compreensão de mudanças em documentos XML usando similaridade.

O Capítulo 3 apresenta a abordagem proposta, assim como os trabalhos relacionados ao tema desta pesquisa. O capítulo também descreve o relacionamento existente entre a literatura relacionada e um comparativo das características relevantes entre todas as abordagens apresentadas.

O Capítulo 4 expõe uma apresentação da abordagem XChange. Posteriormente, é descrito o XChangeSim e seu funcionamento. Através da sequência de ações apresentadas, pode-se demonstrar como o uso de similaridade se adequa na realização de inferências. Desta forma, é possível justificar, sob a perspectiva semântica, as mudanças ocorridas em diferentes versões de um documento XML.

Por fim, o Capítulo 5 faz as considerações finais sobre este trabalho, bem como algumas sugestões de trabalhos futuros.

2 Similaridade: Conceitos em torno da abordagem

A abordagem proposta neste trabalho, foi desenvolvida utilizando técnica de similaridade, com o propósito de compreender as mudanças realizadas ao longo do processo de evolução de documentos XML. Para que o mesmo seja exposto de forma clara, antes, torna-se necessário a apresentação dos principais termos que compõem esta pesquisa.

A seção 2.1 apresenta as perspectivas para avaliar o grau de equivalência entre documentos semiestruturados. A seção 2.2 apresenta o algoritmo LCS e seu funcionamento, o qual, posteriormente, será utilizado como parte da abordagem aqui proposta. Por fim, a seção 2.3 faz referência ao controle de mudanças, às particularidades deste tema em documentos XML e à compreensão das inferências realizadas através do controle de mudanças neste tipo de documentos.

2.1 Análises de similaridade

Buscar similaridade consiste em comparar documentos ou dados através de algoritmos e métricas que avaliem a semelhança entre os conteúdos analisados. O conceito de semelhança varia conforme o contexto e os tipos de dados, por isso para os documentos semiestruturados são usadas duas perspectivas para avaliar o grau de equivalência: a estrutural e a semântica.

2.1.1 Análise estrutural

A perspectiva estrutural avalia as relações entre os termos dos documentos comparando suas cadeias de caracteres (NOLL et al, 2007). Nesta análise, abstraem-se as informações contidas e considera-se apenas o conteúdo estrutural dos documentos, seja em sua forma original ou após alguma conversão, visando facilitar a análise de equivalência. Desta forma, não é necessário conhecer o significado do conteúdo ou o que aquele documento

representa.

O Edit Distance (LEVENSHTEIN, 1966) é um algoritmo clássico para análise de conteúdo estrutural, uma vez que avalia duas cadeias de caracteres considerando o número mínimo de operações necessárias para transformar uma sequência em outra.

2.1.2 Análise semântica

A perspectiva semântica concentra-se no significado e na correlação conceitual entre os termos comparados (NOLL et al, 2007). Para este tipo de análise, sugere-se a utilização de um *Tesauro* para avaliar relações terminológicas entre conceitos. Um *Tesauro* trata-se de uma lista de palavras com significados semelhantes, dentro de um domínio específico de conhecimento (SOUZA et al, 2008). Sob esta perspectiva, elementos semanticamente equivalentes podem ter nomes diferentes em esquemas distintos.

Na análise semântica, alguns algoritmos verificam as relações hierárquicas de subconceito e superconceito do termo em questão, formando um conjunto de conceitos pertencentes a sua hierarquia.

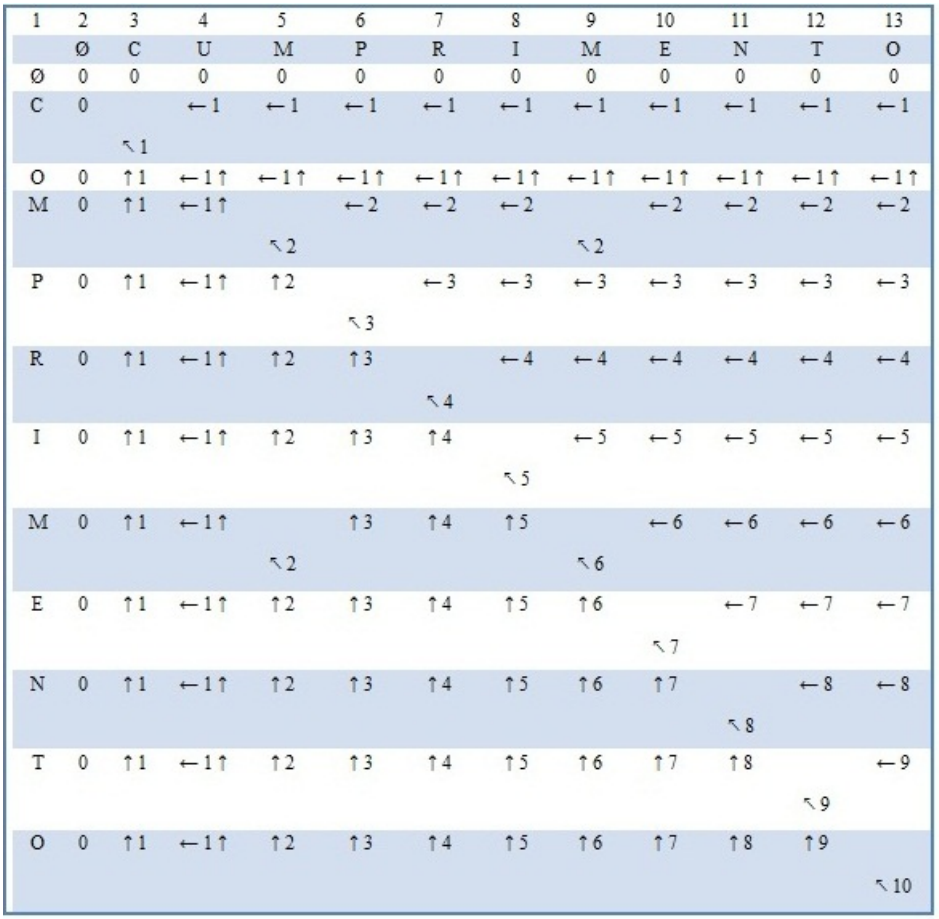
2.2 Longest Common Subsequence

O *Longest Common Subsequence* (LCS) é um algoritmo clássico de manipulação de *strings*, cujo objetivo é encontrar a maior subsequência comum a duas sequências X e Y (CORMEN et al., 2009). O LCS é a base para diversos algoritmos de comparação de arquivos e ferramentas de *diff*.

O algoritmo LCS cria uma matriz de tamanho M x N, em que cada coluna representa um caractere de uma das sequências, de tamanho M e cada linha representa um caractere da outra sequência, de tamanho N. Após criar esta matriz, o algoritmo calcula a maior subsequência comum entre as sequências comparadas.

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 Figura 2.1 apresenta o resultado gerado pelo algoritmo LCS para

as sequências “comprimento” e “cumprimento”.



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

Figura 2.1: Resultado do cálculo do algoritmo LCS (CORMEN et al., 2009)

.

O algoritmo, ao detectar dois caracteres iguais nas posições comparadas, aumenta o número de igualdades encontradas até o momento. A partir disto, armazena na célula esta quantidade e aponta uma seta para a diagonal superior à esquerda. Quando encontra um elemento diferente, a maior quantidade de igualdades descobertas até o momento será copiada. Em seguida, adiciona-se uma seta para a esquerda e/ou para cima, tendo em vista, verificar qual delas possui o caminho com a maior igualdade até o momento.

A leitura desta tabela faz-se da seguinte forma: o algoritmo percorre as setas a partir da célula mais à direita e mais a baixo [13,13]. A seguir, percorre o caminho que proporcionou a maior igualdade (seguindo as setas). As células a serem percorridas [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, o algoritmo pode escolher tanto a célula a cima quanto a célula à esquerda. É notável que, independente do caminho escolhido, ambos seguirão pela célula [3,3] (igualdade).

No caso de documentos XML, para que tal comparação obtenha resultados desejáveis, é necessário que seja realizada individualmente nos diferentes níveis de estrutura do documento.

2.3 Inferência em documentos XML usando similaridade

Documentos XML possuem uma estrutura diferente quando comparados a um documento comum. Por possuir estrutura heterogênea, onde a informação é organizada de acordo com cada tipo de sistema, o XML é um dos formatos mais conhecidos na troca de dados na *Web*. Devido a essa característica, estes documentos são passíveis de alterações frequentes e o grande desafio está em gerenciar as mudanças sofridas à medida que estes vão evoluindo em sua utilização.

2.3.1 Controle de mudanças

Mudanças são cada vez mais constantes em um projeto de desenvolvimento de software. Tais mudanças ocorrem por diversos fatores, como: mudanças nas regras de negócio, novas necessidades dos usuários, alterações preventivas e corretivas, reorganização e refatoração de código, entre outros.

Gerenciar as mudanças ocorridas com a evolução de um projeto é o objetivo de um dos subsistemas da Gerência de Configuração de Software (GCS). A GCS surgiu no fim dos anos 70, trata-se de uma área da Engenharia de Software que utiliza procedimentos técnicos e administrativos para detectar e documentar as características dos elementos de um projeto, ou itens de configuração. Esta área também é responsável por controlar as alterações nestes componentes, armazenar e relatar o processamento das modificações de cada estágio da implementação e verificar a compatibilidade com os requisitos especificados (IEEE, 2005).

Os projetos de Engenharia de Software à medida que evoluem necessitam do acompanhamento de suas alterações, a fim de evitar um ambiente de trabalho desorganizado e uma série de erros. Os sistemas de controle de mudanças (SCM) combinam procedimentos humanos e funcionalidades automatizadas (PRESSMAN, 2011).

Um sistema de controle de mudança trabalha documentando, ao longo do processo de evolução de um determinado projeto, os dados gerados pelas alterações. Estes dados ficam armazenados de forma sistemática para que, posteriormente, sejam fornecidos aos responsáveis pelo projeto. O processo de armazenar e controlar as modificações, permite aos interessados inferir informações como, por exemplo, o motivo das mudanças, os dados alterados e os usuários que as fizeram.

As principais ferramentas de GCS possuem sistemas de controle de mudanças com foco no histórico das modificações e são responsáveis por acompanhar as alterações desde o momento em que elas são solicitadas até o instante em que elas forem disponibilizadas. Desta forma, cada SCM visa adaptar os conceitos de GCS ao tipo de documento analisado.

2.3.2 Controle de mudanças em documentos XML

No contexto de documentos semiestruturados, especificamente XML, o controle de modificações deve fornecer apoio para a gerência das instâncias XML e informações sobre a evolução do item. Os procedimentos para controle de mudanças usados pelos SCM não possuem diferenças entre os tipos de documentos. As ferramentas mais conhecidas seguem esta idéia e não apresentam uma análise diferente para documentos XML, todavia a estrutura hierárquica destes documentos possibilita resultados mais expressivos e com valor semântico diante das informações obtidas pelo controle de mudanças.

As informações obtidas são uma valiosa fonte de conteúdo para entender as modificações ocorridas no projeto. Entretanto, são necessários métodos especializados em extrair conhecimento de documentos semiestruturados e isso não ocorre nos sistemas que utilizam métodos baseados em pesquisa textual (GARCIA, 2012).

Métodos especializados em automatizar a busca e extrair informações expressivas são essenciais em projetos com grande volumes de dados, porém explorar estes mecanismos e os resultados que eles geram tem se tornado um desafio. Diante disso, notou-se a

necessidade em analisar e compreender as mudanças geradas através dos processos de controle na evolução de documentos XML.

2.3.3 Compreensão de mudanças em documentos XML usando similaridade

Ao gerenciar as mudanças ocorridas nos documentos XML, deseja-se mais do que obter informações sobre o que mudou ou quem foi responsável pelas alterações. Espera-se sobretudo, entender a razão das mudanças, isto é, inferir significados para justificar as modificações (OLIVEIRA et al., 2012).

O uso de similaridade visa detectar o grau de equivalência entre os documentos comparados para inferir melhores resultados e, posteriormente, extrair informações implícitas presentes em versões de um documento XML. Desta forma, além de auxiliar o controle de mudanças, é possível analisar e entender a semântica das alterações dentro do contexto de negócio que o documento XML representa.

2.4 Conclusão

Os conceitos apresentados neste capítulo fornecem a base teórica para o entendimento da abordagem proposta. Compreender a diferença entre as análises de equivalência estrutural e semântica, o funcionamento do algoritmo LCS e os aspectos que envolvem o controle de mudanças dentro da Gerência de Configuração de Software são fundamentais para entender como o XChangeSim atua na compreensão de mudanças em documentos XML através do uso de similaridade.

3 Controle de mudanças baseado em similaridade

Há trabalhos que estudam técnicas para identificar e controlar alterações em documentos XML. Estes são compostos por abordagens específicas para diferentes estruturas de dados. A abordagem proposta trata do uso de similaridade para apoiar o controle e a compreensão de mudanças em documentos XML.

Diante disso, o presente capítulo visa mostrar os trabalhos relacionados ao estudo proposto. A seção 3.1 aponta trabalhos presentes na literatura e que estão associados ao tema desta pesquisa. Na seção 3.2 são descritas as características comuns entre as propostas apresentadas na seção anterior. Em seguida, a seção 3.3 faz referência à abordagem proposta, isto é, ao controle de mudanças baseado em similaridade. A seção 3.4 faz um comparativo das abordagens descritas. Por fim, a seção 3.5 apresenta a conclusão deste capítulo.

3.1 Trabalhos relacionados

Os estudos apresentados a seguir estão classificados em abordagens baseadas em chaves e similaridade. Nos trabalhos XyDiff (COBÉNA et al., 2002), X-Diff (WANG et al., 2003) e XKeyDiff (HARA et al., 2007), que usam a estratégia de chaves XML, os documentos comparados são representados em estruturas de árvores para que os nodos sejam percorridos, a fim de casar os elementos das versões de acordo com seus valores identificadores.

As abordagens PathSim (Vinson, 2007) e XML-SIM (MADRIA et al., 2010), baseadas em similaridade, consistem em comparar documentos XML através de algoritmos que são construídos usando métricas que avaliam o quão semelhantes são.

3.1.1 XyDiff

COBÉNA et al. (2002) referencia este item através do algoritmo XyDiff, proposto para

identificar as mudanças entre versões de documentos XML no contexto do projeto Xyleme (XYLEME, 2002). O objetivo é a detecção de mudanças para indexar por palavras, grandes volumes de documentos XML. O algoritmo XyDiff utiliza heurísticas e está dividido em cinco fases:

1. busca pelo casamento único entre nodos;
2. atribuição de assinaturas e pesos aos nodos e ordenamento de subárvores;
3. procura por casamentos priorizando nodos e sub-árvores de maior peso;
4. otimização dos casamentos;
5. geração do script delta.

A abordagem consiste em encontrar a maior subárvore idêntica em ambos os documentos, e a partir desta, percorrer seus nodos identificando quais sofreram alterações e quais permaneceram iguais. O XyDiff faz o cálculo de valores de pesos e *hash*. Estes cálculos são realizados ao associar os elementos das versões em análise, a fim de determinar um identificador para cada nodo.

A etapa inicial do algoritmo verifica os elementos que possuem um identificador e, através deste atributo, o XyDiff encontra os nodos similares na versão comparada. O atributo ID é predefinido no DTD (*Document Type Definition*) do documento XML e os elementos que não possuem este atributo precisam ser analisados a partir de seu contexto.

Na segunda fase, o XyDiff calcula o *hash* e o peso de cada nodo. O cálculo do peso é definido pelo tamanho do conteúdo armazenado no nodo e pelo peso dos nodos filhos correspondentes. Ao final, o algoritmo realiza a ordenação por pesos, de tal forma que na fila de subárvores, as primeiras posições estão destinadas aos nodos de maior peso e, consequentemente, estes são os primeiros a serem testados na verificação de equivalência.

A partir da fila de subárvores, a etapa seguinte consiste em encontrar as equivalências. O algoritmo, por conhecer o valor *hash* de cada nodo, compara as subárvores que possuem mais diferenças entre si, e em seguida, aquelas onde foram encontradas alterações mínimas. O valor *hash* é equivalente à assinatura do nodo. Portanto, a comparação das assinaturas dos nodos é o que define a equivalência entre eles.

A quarta fase do XyDiff é caracterizada pelo mapeamento na árvore. Nesta etapa, realizam-se dois tipos de caminhamento: o *bottom-up*, onde a árvore é percorrida dos nodos folhas até o topo. Em seguida, utiliza-se o caminhamento *top-down*, onde a árvore é visitada a partir da raiz até as folhas. O mapeamento visa detectar os nodos de mesmo nome e que possuem pais equivalentes para evitar a identificação de deleções e inserções desnecessárias.

Na etapa final, o algoritmo calcula o *delta script* e classifica os resultados entre nodos de inserção, remoção e atualização (mudança de conteúdo). Uma característica importante do XyDiff é que o algoritmo identifica, além das operações básicas, os nodos que mudaram de posição na árvore. A saída do algoritmo é o *delta script* gerado com as informações de inserção, remoção, atualização e movimentação.

A abordagem proposta é eficiente em desempenho e velocidade. Entretanto, a qualidade dos resultados é um problema que foi tratado em trabalhos posteriores. O XyDiff detecta as alterações entre as versões de documentos XML, mas não utiliza informações de caráter semântico a fim de justificar as alterações encontradas.

3.1.2 X-Diff

Em WANG et al. (2003) é proposto um algoritmo para detecção de diferenças entre documentos XML que utiliza o conceito de árvores não-ordenadas, denominado X-Diff. Neste algoritmo, a detecção de mudanças é substancialmente mais difícil do que usando o modelo ordenado, mas o resultado das alterações é mais preciso. Esta característica, faz com que o X-Diff possua maior qualidade quando comparado com o algoritmo XyDiff (COBÉNA et al., 2002). O X-Diff está dividido em cinco etapas e considera somente o relacionamento pai-filho.

A primeira etapa do X-Diff é a transformação das entradas. O algoritmo converte cada documento XML em uma árvore, executando o *parse* dos documentos, isto é, a leitura e análise dos documentos para transformá-los em estruturas denominadas *Xtrees*. Esta estrutura consiste em um subconjunto da interface para manipulação de documento existente no DOM (*Document Object Model*).

A etapa seguinte visa a computação de assinaturas. Em ambas as árvores, ao

realizar o *parse*, o algoritmo faz uso de uma função especial de *hash* para calcular a assinatura. Esta função, chamada *XHash*, representa a subárvore enraizada no nodo. A particularidade é que a ordem entre irmãos é desconsiderada.

A terceira fase do X-Diff, realiza o casamento dos nodos entre as versões do documento XML. No início desta etapa, visando reduzir o espaço de busca, o algoritmo avalia os valores *hash* dos elementos do segundo nível das árvores. Desta forma, as subárvores inteiras são casadas, evitando a comparação de seus nodos. Em seguida, é realizada a comparação das distâncias, para cada folha com os nodos da outra árvore XML. Os resultados desta comparação ficam armazenados em uma tabela para, posteriormente, avaliar os melhores casamentos.

Ao terminar o processo de comparação, executa-se o mesmo procedimento com os nodos pais. Em cada ascensão de nível na árvore, é feita a comparação dos valores *hash* a fim de diminuir o espaço de busca. Após realizar estes procedimentos, a avaliação dos resultados armazenados na tabela começa com os nodos raízes, partindo para seus descendentes. O X-Diff realiza uma avaliação *top-down* nas estruturas de árvore, analisando os melhores candidatos coletados e casando-os com os respectivos nodos.

A quarta fase desta abordagem trata-se de uma etapa opcional, uma vez que consiste na otimização do processo. Utiliza-se um limiar (*threshold*) para selecionar o melhor candidato entre os coletados, evitando a comparação com todos os candidatos possíveis. Isto é, aquele candidato que possuir um limiar menor é escolhido, desconsiderando a comparação com os demais.

Na etapa final do X-Diff, após os casamentos, assim como no algoritmo XyDiff (COBÉNA et al., 2002), o *delta script* é construído. Este documento descreve as operações de inclusão, remoção e atualização. Ao contrário do XyDiff (COBÉNA et al., 2002), o *delta script* do X-Diff não apresenta a operação de movimento.

Detectar mudanças em árvores não-ordenadas torna-se uma tarefa mais complexa, porém, os autores defendem que desta forma o resultado é mais exato e adequado para aplicações como banco de dados em XML.

3.1.3 XKeyDiff

Em HARA et al. (2007) é apresentado o algoritmo XKeyDiff. Esta abordagem é composta por duas etapas e tem como objetivo guiar a comparação entre documentos XML. A primeira etapa consiste em casar os elementos das versões comparadas através de seus valores identificadores. Em seguida, é feita uma análise estrutural para detectar as diferenças entre eles e gerar o *edit script*. O *edit script*, ou *delta*, representa as mudanças entre as versões analisadas, ou seja, a sequência de operações necessárias para transformar uma versão do documento em outra.

Os documentos XML podem ser representados em forma de árvore. Os nodos da árvore podem ser de três tipos: elemento, atributo ou texto. O algoritmo recebe como parâmetros de entrada duas versões, v1 e v2, de árvores XML e um conjunto de chaves XML que são satisfeitas por ambas as versões. Para definir uma chave XML é preciso especificar três aspectos: o contexto no qual a chave deve ser satisfeita; o conjunto alvo que define os elementos sobre os quais a chave é definida; e os valores identificadores de elementos do conjunto alvo. A saída do algoritmo é um arquivo *delta*.

O XKeyDiff é composto por dois módulos: o XKeyMatch e um algoritmo de *diff*. Neste contexto, pode-se utilizar qualquer algoritmo de *diff* existente na literatura, sendo necessário implementar uma comunicação desta abordagem com a abordagem escolhida. O algoritmo XKeyMatch tem como objetivo gerar um conjunto de pares de elementos [n1, n2], onde n1 é um nodo em v1 que refere-se à mesma entidade que o nodo n2 em v2 de acordo com o conjunto de chaves XML.

O XKeyMatch é baseado na construção de um Autômato Finito Determinístico (AFD) a partir do conjunto de chaves XML, denominado KeyDFA. Um AFD é uma máquina de estados finitos onde, para cada par de estados e símbolo de entrada, existe um próximo estado determinístico. Através do AFD é possível processar ao mesmo tempo, todo o conjunto de chaves XML, e todos os casamentos baseados neste conjunto podem ser gerados com um percurso em pré-ordem em v1 e v2. De forma específica, cada passo no percurso da árvore XML corresponde a uma mudança de estado do autômato e as informações sobre chaves armazenadas em cada estado são utilizadas para coletar os nodos que são candidatos aos casamentos.

Após coletar todos os candidatos, o algoritmo cria pares de nodos, um de cada versão, de acordo com seus valores identificadores. O conjunto de pares gerados trata-se da entrada para o algoritmo de *diff* que, baseado nos casamentos, compara as versões v1 e v2 e gera o *edit script*.

O XKeyDiff realizou um estudo experimental para determinar a importância que a fase de pré-processamento tem sobre o processo de identificação de diferenças. Notou-se que, embora o algoritmo exija que o usuário conheça o conteúdo dos documentos comparados, quando as chaves XML expressam conteúdo semântico, o algoritmo produz resultados mais significativos que outros baseados somente em similaridades de valor e estrutura.

3.1.4 PathSim

O trabalho de Vinson (2007) apresenta um algoritmo de similaridade entre caminhos XML, nomeado PathSim. O algoritmo realiza o cálculo de similaridade entre dois caminhos baseado no menor número de operações de edição (inserção, remoção e substituição) necessárias para transformar um caminho no outro. Para complementar a pesquisa, duas variações da abordagem são apresentadas, uma incrementada com comparações entre combinações de nomes de elementos, chamada PathSimC, e a outra auxiliada por técnicas de alinhamento, nomeada PathSimA.

O algoritmo recebe como entrada dois caminhos XML e computa como saída um valor de similaridade entre eles. O resultado é normalizado no intervalo [0;1], onde 0 indica desigualdade completa, enquanto 1 representa que as entradas são similares para a função. O PathSim tem como base o algoritmo Edit Distance (LEVENSHTEIN, 1966) e está dividido em três etapas.

Na fase incial, o algoritmo PathSim (Vinson, 2007) cria e inicializa a matriz que será utilizada para armazenar e propagar os custos das operações de transformar um caminho no outro. De forma análoga ao algoritmo Edit Distance (LEVENSHTEIN, 1966), cada célula da matriz armazena o custo de transformar um sub-caminho de uma entrada em um sub-caminho da outra entrada.

Os valores associados às células que não fazem parte da primeira linha ou pri-

meira coluna da matriz, isto é, aquelas que não foram inicializadas, são calculados a partir de valores associados a algumas células vizinhas (superior, esquerda e diagonal superior esquerda), acrescidas do custo da operação correspondente (inserção, remoção ou substituição). Os custos de inserção e remoção são calculados pela adição do valor 1 ao valor da célula na matriz de custos, de acordo com a estrutura determinada pelo algoritmo.

Na segunda fase do algoritmo, o cálculo do custo de substituição de um nome de elemento é definido pela adição do valor de similaridade entre os nomes de elementos correspondentes à célula da matriz de custos. Por isso, o PathSim (Vinson, 2007) aplica os valores reais no intervalo [0;1], e não somente suas extremidades, para definir o custo da operação de substituição. O objetivo de utilizar esse intervalo é representar uma substituição parcial. Substituição parcial significa que os nomes dos elementos são diferentes, mas não em totalidade, o que representa com mais exatidão a similaridade entre os caminhos.

Na terceira etapa, ao preencher totalmente a matriz de custos, a similaridade final entre os caminhos é dada por uma fórmula análoga à utilizada pelo Edit Distance. O cálculo de similaridade normaliza, no intervalo [0; 1], a medida do máximo número de operações possíveis para converter um caminho no outro, subtraído do menor número de operações necessárias para efetuar a mesma transformação (calculado pela matriz, representado na última célula). A normalização é feita pela divisão do valor calculado na subtração, pelo máximo de operações possíveis para transformação dos caminhos. A partir do algoritmo PathSim apresentado, foram criadas duas variações: o PathSimC e o PathSimA.

A primeira variação, nomeada PathSimC, permite que sejam computadas diferenças entre combinações de nomes de elementos no cálculo do custo da operação de substituição. Se o nome de um elemento de um caminho for uma combinação dos nomes de alguns elementos do outro, o algoritmo PathSimC apresenta um valor de similaridade superior ao PathSim.

A variante PathSimA, define onde subsequências de nomes de elementos do caminho mais extenso começam e terminam, através de técnicas de alinhamento de cadeias de caracteres, de maneira a definir semelhança em relação à seqüência de nomes do caminho

mínimo. Assim, o algoritmo PathSimA quando comparado ao PathSim, apresenta maiores valores finais de similaridade, por eliminar as diferenças nos níveis de detalhamento e de escopo dos caminhos.

O trabalho de Vinson (2007) baseado na abordagem do algoritmo Edit Distance busca apresentar um algoritmo onde os resultados são mais precisos. O PathSim permite utilizar qualquer função de similaridade entre nomes de elementos para calcular o custo da operação de substituição. Portanto, uma implementação do algoritmo pode ser incrementada com funções específicas de um domínio, além de poder beneficiar-se do uso de dicionários de sinônimos, *thesaurus* e ontologias de termos. Combinações de funções de similaridade também podem ser usadas para computar a similaridade entre dois nomes de elementos. As variantes do PathSim, acrescentam desempenho ao algoritmo de acordo com a aplicação a ser analisada.

3.1.5 XML-SIM

XML-SIM é um algoritmo que propõe a melhoria dos algoritmos XDoI (MADRIA et al, 2008) e XDI-CSSK (MADRIA et al., 2009) para detectar similaridade entre dois documentos XML (MADRIA et al., 2010).

Os trabalhos anteriores XDoI (MADRIA et al, 2008) e XDI-CSSK (MADRIA et al., 2009) são métodos onde as subárvores semelhantes são calculadas primeiro com base na similaridade de conteúdo, comparando o número de valores comuns em nível nó folha, sem considerar a sua estrutura. Se os cálculos resultarem em casamentos múltiplos então a similaridade estrutural é tomada em consideração para distinguir qual o par de subárvores é mais semelhante. O algoritmo gasta um tempo elevado para o cálculo, uma vez que todos os nós folhas são comparados, mesmo aqueles que não se assemelham em termos de seus tipos de dados e semântica. Diante disso, o XML-SIM (MADRIA et al., 2010) foi proposto, através de quatro etapas, para resolver o problema detectado nos métodos anteriores.

A primeira etapa da abordagem do XML-SIM (MADRIA et al., 2010), consiste no armazenamento dos documentos XML. Estes documentos são armazenados em um banco de dados relacional, o que aumenta a escalabilidade, de modo que a limitação de

carregamento de árvores XML muito grandes para a memória principal não é restrição.

Em segundo lugar, os documentos XML são agrupados em subárvores usando pais nó-folha que são verificados pela integridade da subárvore. Em outras palavras, um pai nó-folha é um nó que tem pelo menos um filho, como um nó de folha. Este nó é considerado como uma raiz de subárvore no processo de agrupamento.

Após o agrupamento, a etapa seguinte consiste em definir as chaves XML com base em um valor de nó folha correspondente para todos os valores de nó exclusivo com o mesmo caminho de assinatura. A chave é usada mais tarde na correspondência de subárvores. No entanto, a equivalência de chaves pode resultar em casamentos múltiplos, uma vez que uma chave identificada pode ser parte de uma ou mais subárvores diferentes. Esta informação pode ser usada no processo de filtro de subárvore, a fim de livrar-se de subárvores inapropriados. Neste ponto, tem-se filtradas as subárvores apropriadas de ambos os documentos XML para serem comparadas na estrutura e detecção de similaridade de conteúdo.

Na quarta etapa da abordagem XML-SIM (MADRIA et al., 2010), as estruturas das subárvores definidas são medidas para encontrar a semelhança semântica-estrutural com base no analisador taxonômico. A semelhança semântica-estrutural é explorada, a fim de comparar a igualdade de conteúdo. A similaridade de conteúdo é determinado por meio da comparação de valores de nó folha que têm estrutura semântica equivalente. Finalmente, o sistema resulta em melhores correspondências de pares das subárvores.

Ao avaliar a eficácia do XML-SIM (MADRIA et al., 2010), os autores defendem, por avaliações experimentais, que esta abordagem supera os trabalhos XDoI (MADRIA et al, 2008) e XDI-CSSK (MADRIA et al., 2009) em termos de tempo de execução, bem como as taxas de falsos positivos. Propõe-se como trabalho futuro, estender esta pesquisa para encontrar a similaridade entre várias versões de documentos XML.

3.2 Abordagem proposta

Visando apoiar a compreensão de mudanças em documentos XML, este trabalho propõe o XChangeSim para complementar a funcionalidade principal da abordagem XChange, utilizando similaridade para detecção de diferenças entre documentos XML.

O XChangeSim consiste na implementação do protótipo de PORTELA et al (2012) na abordagem XChange. PORTELA et al (2012) apresentam a integração do algoritmo Phoenix (CAMPELLO et al., 2012) e o projeto de evolução semântica (OLIVEIRA et al., 2012). O Phoenix (CAMPELLO et al., 2012) foi elaborado para realizar a comparação e detecção de conflitos entre documentos XML, através dos seguintes itens presentes nos documentos: nomes dos elementos, conteúdo textual, atributos e seus valores, e por fim, os subelementos.

Uma vez que um documento XML tem atributos característicos a um documento de texto, torna-se propício a falhas comuns, como erros de digitação. Devido a isso, pequenas mudanças como acentuação ou presença de letras maiúsculas e minúsculas, nas versões que estão sendo analisadas, podem ser responsáveis por concluir que não se tratam do mesmo elemento. Para resolver este tipo de problema, o Phoenix (CAMPELLO et al., 2012) faz uso de um cálculo matemático para concluir quão similar são dois documentos, atribuindo pesos para cada um dos quatros itens a serem verificados pelo algoritmo. De forma que, para cada item que é equivalente, soma-se 25% ao valor da similaridade total.

O XChangeSim recebe como entrada dois documentos do tipo XML, como por exemplo, os fragmentos apresentados nas listagens 3.1 e 3.2.

Listagem 3.1: Versão Base

```
1  <company>
2     <employee>
3        <ssn>123-54-4410</ssn>
4        <name>John</name>
5        <phone>591-888-1234</phone>
6        <father>Darrell</father>
7        <mother>Eleanor</mother>
8        <salary>1600</salary>
9        <job>System Analyst</job>
10       <department>Information Technology</department>
11       <branch>Washington</branch>
12    </employee>
13    ...
14 </company>
```

Listagem 3.2: Versão Modificada

```
<company>
   <employee>
      <ssn>123-54-4410</ssn>
      <name>John</name>
      <phone>591-888-1234</phone>
      <father>Darrell</father>
      <mother>Eleanor</mother>
      <salary>2500</salary>
      <job>Full System Analyst</job>
      <department>Information Technology</department>
      <branch>Washington</branch>
   </employee>
   ...
</company>
```

Os documentos XML do exemplo, descrevem o cadastro de funcionários de uma empresa. Dentro deste cenário, as informações são organizadas no documento XML para controle do setor de recursos humanos. Periodicamente, pode ser necessária a verificação das versões de um documento, a fim de analisar as mudanças ocorridas no quadro de funcionários da empresa.

A aplicação desenvolvida utiliza o projeto Phoenix (CAMPELLO et al., 2012) como biblioteca para fazer uso das chamadas de seus métodos. Os documentos de entrada do XChangeSim são enviados como parâmetros para o algoritmo. Como saída, o XChangeSim exibe o resultado da execução do Phoenix (CAMPELLO et al., 2012). Trata-se de documento com um novo atributo intitulado "*similarity*" que define o grau de similaridade entre os elementos das duas versões. Caso a similaridade seja inferior a 1, isto é, não ocorra casamento exato, são apresentados no campo "*value*" que contêm como atributos "*left*" e "*right*", os valores das versões 1 e 2, respectivamente.

O resultado da execução do algoritmo sobre os documentos das listagens 3.1 e 3.2 é apresentado abaixo, na Listagem 3.3. A partir do cálculo de similaridade realizado, deseja-se usar os valores de equivalência entre os atributos para inferir as mudanças ocorridas entre as versões.

Listagem 3.3: Resultado do algoritmo sobre os documentos de entrada.

```
<company xmlns:diff="ic.uff.br/xmldiff" xmlns:left=
"ic.uff.br/xmldiff" xmlns:right="ic.uff.br/xmldiff"
diff:similarity="0.9698364">
 <employee diff:similarity="0.9954756">
  <ssn diff:similarity="1.0">123-54-4410</ssn>
  <name diff:similarity="1.0">John</name>
  <phone diff:similarity="1.0">591-888-1234</phone>
  <father diff:similarity="1.0">Darrell</father>
  <mother diff:similarity="1.0">Eleanor</mother>
  <salary diff:similarity="0.875">
   <diff:value diff:left="1600" diff:right="2500"/>
  </salary>
  <job diff:similarity="0.96212125">
   <diff:value diff:left="System␣Analyst"
   diff:right="Full␣System␣Analyst"/>
  </job>
  <department diff:similarity="1.0">Information Technology</department>
  <branch diff:similarity="1.0">Washington</branch>
 </employee>
  ...
</company>
```

O trabalho de PORTELA et al (2012) propõe como solução utilizar o grau de similaridade de cada elemento para, em seguida, gerar identificadores únicos nos documentos. A Listagem 3.4 corresponde a versão 1 com o novo subelemento ID, e de forma análoga, a Listagem 3.5 corresponde ao resultado para versão 2.

O projeto de OLIVEIRA et al. (2012) tem como objetivo inferir que dois documentos XML que possuem estruturas diferentes entre si, podem tratar-se do mesmo objeto, sendo possível também encontrar diferenças no contexto semântico, ao longo do processo de evolução. Baseado nesta abordagem, o XChangeSim utiliza as novas versões geradas dos documentos e através da aplicação de regras de inferência em linguagem Prolog, busca revelar informações relevantes dentro do contexto de negócio em que os documentos são utilizados.

Listagem 3.4: Versão Base com subelemento ID.

```
<company>
  <employee>
    <id>1</id>
    <ssn>123-54-4410</ssn>
    <name>John</name>
    <phone>591-888-1234</phone>
    <father>Darrell</father>
    <mother>Eleanor</mother>
    <salary>1600</salary>
    <job>System Analyst</job>
    <department>Information Technology</department>
    <branch>Washington</branch>
  </employee>
  ...
</company>
```

Listagem 3.5: Versão Modificada com subelemento ID.

```
<company>
  <employee>
    <id>1</id>
    <ssn>123-54-4410</ssn>
    <name>John</name>
    <phone>591-888-1234</phone>
    <father>Darrell</father>
    <mother>Eleanor</mother>
    <salary>2500</salary>
    <job>Full System Analyst</job>
    <department>Information Technology</department>
    <branch>Washington</branch>
  </employee>
  ...
</company>
```

A aplicação proposta, tal como a técnica utilizada, tem importante papel dentro da abordagem XChange, uma vez que resolve problemas anteriormente encontrados. No

caso da abordagem "*Context Key*"do sistema, se o usuário informasse o valor errado de apenas um atributo em uma das versões XML e, este atributo fosse eleito como chave de contexto, todo o elemento seria considerado diferente. Como exemplo, caso na versão 1 um subelemento "SSN"tenha valor "123 -54 -4410"e na versão 2, o mesmo atributo tenha valor "123 -54 -4411", e "SSN"seja a chave escolhida, a abordagem considera que não há semelhança entre os elementos. Por sua vez, o XChangeSim avalia todos os subelementos e ao verificar que os demais possuem semelhanças em ambas as versões, de acordo com o grau de similaridade informado pelo usuário, conclui que os elementos são equivalentes.

Este trabalho também facilita o entendimento da aplicação, tornando o processo intuitivo dentro do contexto de negócio. Assim, não é necessário que o usuário do sistema seja conhecedor das abordagens que conduzem os processos, como por exemplo, a linguagem Prolog ou até mesmo, o conceito de chave de contexto.

3.3 Análise das propostas

Esta seção mostra os aspectos comuns entre a literatura relacionada. Também é apresentado um quadro comparativo entre estas abordagens e a proposta deste trabalho, verificando as características relevantes para análise das pesquisas.

3.3.1 Relacionamento entre as abordagens

Os trabalhos relacionados a abordagem proposta possuem características comuns. A Figura 3.1 mostra o relacionamento entre as propostas.

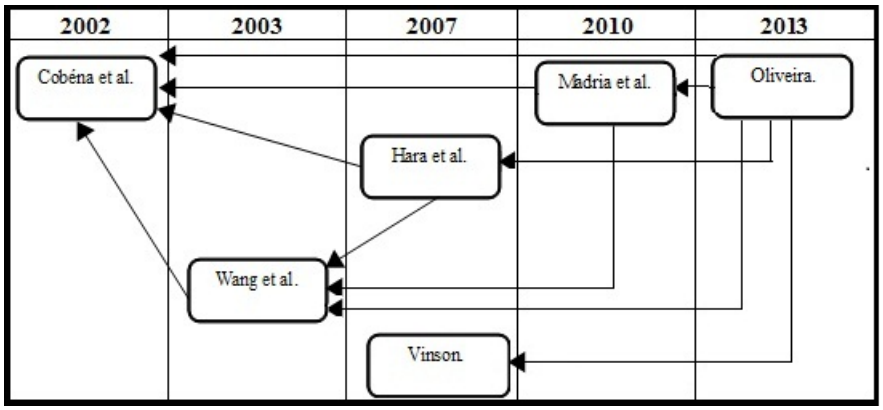


Figura 3.1: Relacionamento entre as propostas.

Como apresentado na Figura 3.1, o trabalho de COBÉNA et al. (2002) baseado no conceito de chaves em árvores ordenadas para detecção de diferenças em documentos XML serviu como base para que WANG et al. (2003) apresentassem um algoritmo com o mesmo objetivo. Entretanto, WANG et al. (2003) trabalham com o conceito de árvores não-ordenadas, a fim de obter resultados mais precisos do que o algoritmo proposto por COBÉNA et al. (2002).

A abordagem de HARA et al. (2007) está relacionada com os trabalhos de COBÉNA et al. (2002) e WANG et al. (2003), pois também apresenta um algoritmo que visa detectar diferenças entre versões de um documento XML e que faz uso do mesmo conceito de chaves em estruturas de árvores para gerar como saída um *delta script* das operações realizadas. Entretanto, o diferencial desta abordagem é a fase de pré-processamento que tem como meta, casar os elementos das versões comparadas através de seus valores identificadores. Só ao final desta fase, dá-se o início da análise estrutural.

O trabalho de MADRIA et al. (2010) também encontra-se relacionado às abordagens propostas por COBÉNA et al. (2002) e WANG et al. (2003), uma vez que o algoritmo, assim como os demais, trabalha com subárvores para detectar similaridades entre documentos XML.

A abordagem proposta por Vinson (2007), apesar de assim como MADRIA et al. (2010) utilizar técnicas baseadas em similaridade, não tem relacionamento direto com as demais abordagens, pois trabalha com a estrutura de matriz e tem como base o algoritmo Edit Distance que avalia cadeia de caracteres.

3.3.2 Comparativo

A Tabela 3.1 apresenta itens relevantes para a comparação entre a abordagem proposta e seus trabalhos relacionados, levando em consideração a proposta deste estudo.

A partir da análise da Tabela 3.1 pode-se constatar, no contexto de controle de mudança, que o uso da técnica de similaridade faz com que a abordagem proposta apresente vantagens sobre as demais. Dentre estas vantagens, pode-se nomear a realização de consultas, a interação do usuário através de uma interface gráfica e sobretudo, a técnica utilizada garante resultados precisos e que envolvem a estrutura e o conteúdo

Tabela 3.1: Comparativo entre as abordagens apresentadas.

	Cobéna et al	Wang et al	Hara et al	Vinson	Madria et al	Oliveira
Abordagem	XyDiff	X-Diff	XKeyDiff	PathSim	XML-SIM	XChangeSim
Baseada em	Chaves	Chaves	Chaves	Similaridade	Similaridade	Similaridade
Tipo de análise	Estrutural	Estrutural	Estrutural	Estrutural	Estrutural	Estrutural e Semântico
Interface	Não	Não	Não	Não	Não	Sim
Consultas	Não	Não	Não	Não	Não	Sim
Eficiência	Desempenho e velocidade	Resultados mais precisos	Resultados mais precisos	Resultados mais precisos	Tempo de execução e taxa de falsos positivos	Resultados mais precisos
Resultados	Delta Script	Delta Script	Delta Script	Valor normalizado no intervalo [0,1]	Melhores correspondências entre pares de subárvores	Semântica das mudanças detectadas nos documentos XML

do documento como um todo. A abordagem proposta é a única que produz resultados que envolvem a semântica do documento para justificar as mudanças detectadas.

3.4 Conclusão

Neste capítulo foi possível analisar alguns trabalhos presentes na literatura que visam detectar diferenças entre versões de um documento XML utilizando variadas técnicas. Entretanto, a maioria das abordagens existentes não utilizam semântica para justificar os resultados encontrados a partir da estrutura e conteúdo dos documentos. O proposto neste trabalho, através de similaridade, visa detectar mudanças entre dois documentos XML sob perspectiva semântica.

4 XChangeSim: Um exemplo de utilização

Diante da abordagem apresentada, pretende-se fornecer significados semânticos ao contexto em que se realizaram as mudanças entre os documentos XML.

O presente capítulo expõe uma descrição da abordagem XChange, para em seguida, apresentar o XChangeSim e seu funcionamento. O funcionamento do XChangeSim é descrito através das ações que o usuário do XChange deve executar, a fim de exemplificar de forma prática a utilização da técnica de similaridade dentro do contexto da aplicação.

4.1 XChange

O XChange tem por objetivo principal realizar inferências a fim de apoiar o controle de mudanças em documentos XML. A aplicação foi desenvolvida em linguagem de programação Java e em sua versão atual, está dividida em duas abordagens: "*Context Key*" e "*Similarity*".

A abordagem "*Context Key*" realiza inferência através de um atributo chave, do contexto de negócio que o documento XML representa. Portanto, espera-se que o usuário, conhecedor do domínio de negócio em questão, escolha um atributo cujo valor seja único em cada elemento (GARCIA, 2012). A segunda abordagem baseia-se na proposta apresentada neste trabalho e será apresentada neste capítulo.

A interface do XChange foi modificada para receber a funcionalidade de inferência por similaridade. Uma nova opção foi criada para que o usuário possa definir qual técnica deve ser utilizada para realizar inferências nos documentos XML de entrada.

O usuário deve carregar dois documentos XML através das opções "*Base version*" e "*Modified version*", e para iniciar o processo de inferência através da abordagem XChangeSim, deve selecionar a opção "*Similarity*" no painel lateral esquerdo.

4.2 XChangeSim

A abordagem XChangeSim foi desenvolvida para apoiar a funcionalidade principal do XChange. XChange é executado através dos seguintes passos e está representado na Figura 4.1:

1. Abrir a versão 1 do documento XML.
2. Abrir a versão 2 do documento XML.
3. Inserir o valor do grau de similaridade considerado aceitável.
4. Gerar as versões com os identificadores (subelemento ID).
5. Traduzir as novas versões em fatos Prolog.
6. Abrir o arquivo de regras Prolog ou criá-las manualmente.
7. Gerar as inferências.

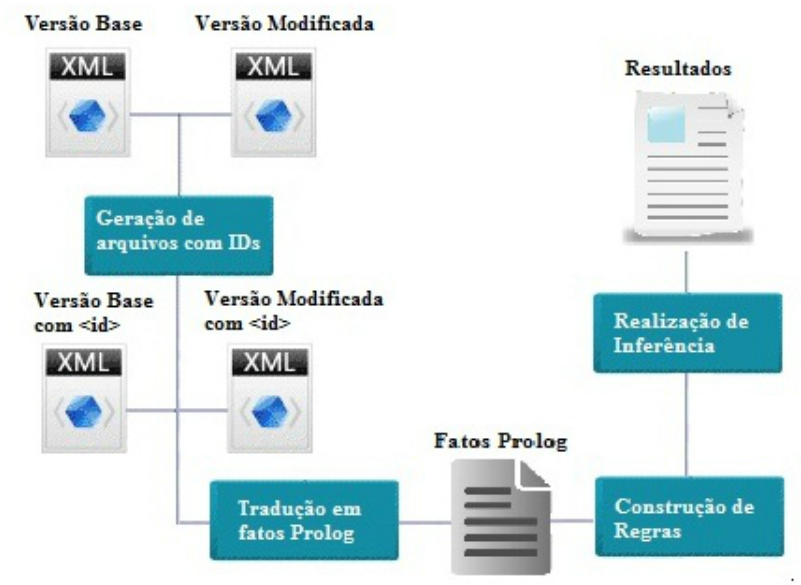


Figura 4.1: Fluxo de execução do XChangeSim

4.2.1 Definição da taxa de similaridade

Quando o módulo de inferência por similaridade é escolhido, o campo de texto "*Similarity Rate*" é ativado para que seja informado um valor aceitável para o grau de similaridade

entre os documentos. Este valor, que deve estar entre 0 e 1, é usado posteriormente, a fim de definir se os elementos presentes nos documentos de entrada tratam-se do mesmo elemento. A Figura 4.2 apresenta o XChange onde podem ser visualizadas duas versões de documento XML no contexto do cadastro de funcionários. Além disso, pode-se observar a definição da taxa de similaridade

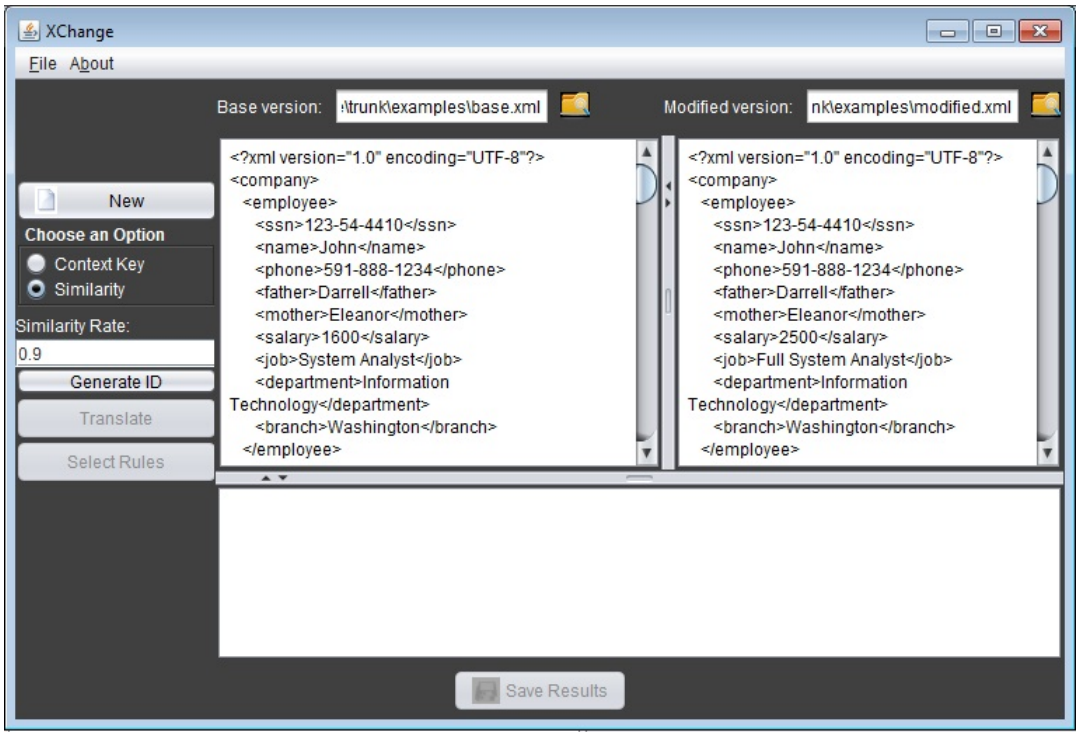


Figura 4.2: Definição do grau de similaridade.

4.2.2 Geração de IDs

Ao definir os documentos de entrada e a taxa de similaridade, o usuário deve começar o processo de geração de identificadores através do botão "*Generate ID*" localizado no painel lateral esquerdo. Nesta etapa, os documentos XML de entrada são passados como parâmetros para que o algoritmo LcsXML do Phoenix (CAMPELLO et al., 2012) possa calcular o grau de similaridade entre eles. A execução do algoritmo LCS foi descrita na seção 2.2 deste trabalho. Como saída, o Phoenix (CAMPELLO et al., 2012) gera o documento XML mostrado na Listagem A.3, em anexo na Apêndice.

Através da API DOM usada no *parser* do XML, pode-se caminhar por toda a árvore do documento XML retornado, verificando se o grau de similaridade é maior do

que o informado no campo "*Similarity Rate*" da interface. Se o grau de similaridade for maior, então é considerado como o mesmo elemento. Caso contrário, os elementos comparados recebem no subelemento ID valores diferentes para cada uma das versões. Isto demonstra que os elementos não correspondem ao grau de similaridade aceitável.

Após o processo de verificação inicial, se os elementos comparados forem o mesmo, verifica-se se este possui filhos. Caso possua, o elemento ganha um subelemento ID, cujo valor é igual nas duas versões de documentos XML resultantes. Entretanto, se o mesmo não possuir nós descendentes, isto é, caso possua apenas valores e atributos ou o seu único filho seja o elemento "*value*", ele não recebe o subelemento ID e é mapeado seguindo outra regra.

A regra para o último caso é que se o atributo de similaridade tiver valor igual a 1, as duas versões XML recebem o mesmo elemento, mas se o valor do atributo de similaridade for menor que 1, a versão 1 do documento XML recebe o elemento com o valor do atributo "*left*" do elemento "*value*" e a versão 2 do documento XML recebe o elemento correspondente ao valor do atributo "*right*". Os novos documentos XML gerados após estas verificações são exibidos para que o usuário acompanhe o que foi realizado neste processo.

Concluída esta etapa, retorna-se ao fluxo padrão do XChange e agora, os fatos e regras podem ser baseados em um ID que garante a unicidade dos elementos nas duas versões. Para os documentos de entrada do exemplo da Figura 4.2, o resultado do processo de geração de ID está apresentado na Figura 4.3, onde são exibidas as novas versões dos documentos de entrada e o documento gerado com os valores de similaridade entre os conteúdos.

Como adicionais do sistema, nesta etapa os documentos gerados com identificadores são salvos no diretório onde se encontram os documentos XML de entrada. O usuário também possui a opção "*Save Results*", caso queira salvar o documento XML gerado pelo Phoenix (CAMPELLO et al., 2012).

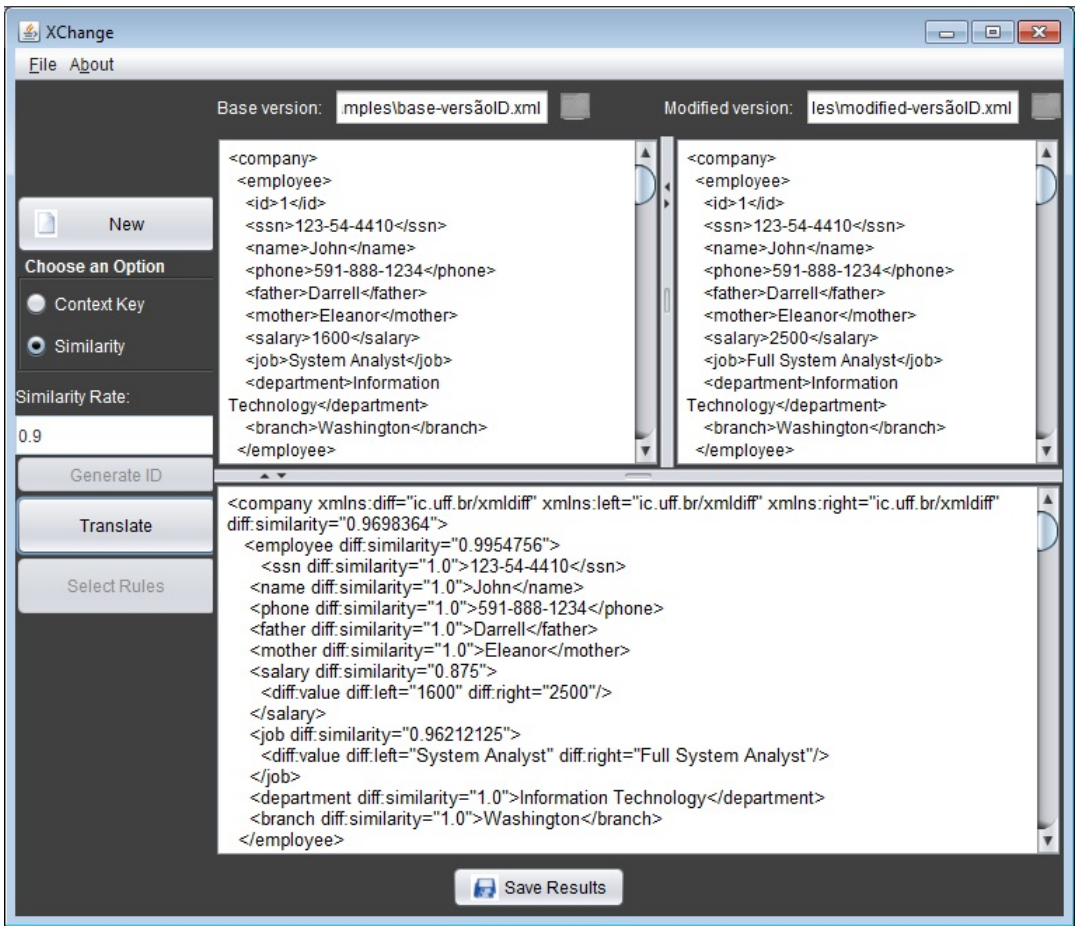


Figura 4.3: Geração de ID

4.2.3 Tradução dos documentos XML em fatos Prolog

A partir dos documentos com o novo atributo ID, a tradução em fatos Prolog é executada através do botão "*Translate*". Cada elemento do documento XML origina um fato correspondente que descreve o seu conteúdo, sua posição hierárquica no documento e um identificador sequencial. A Listagem A.6, apresentada na Apêndice, corresponde aos fatos Prolog gerados a partir das versões com identificadores do documento XML.

No XChange, o resultado desta etapa é apresentado no painel inferior, substituindo o documento XML gerado pelo algoritmo Phoenix (CAMPELLO et al., 2012). A Figura 4.4 apresenta o resultado após o processo de tradução.

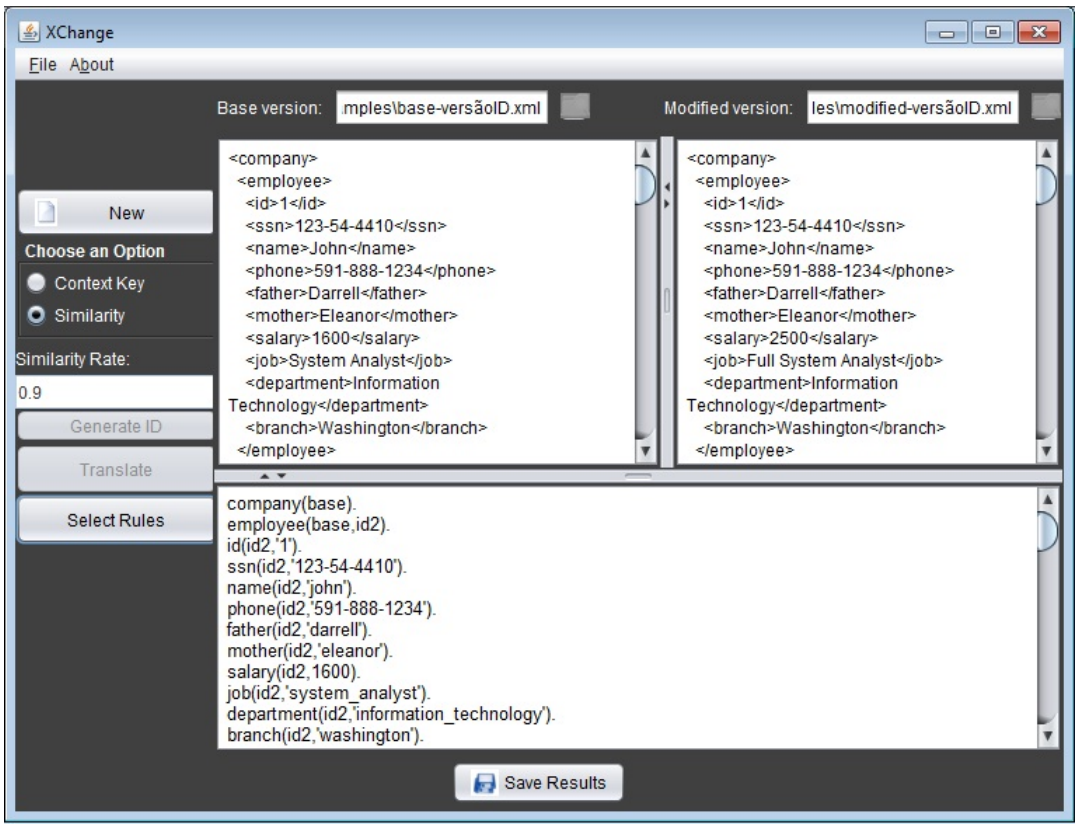


Figura 4.4: Tradução dos documentos XML em fatos Prolog

4.2.4 Definição das regras de inferência

O processo de definição das regras de inferência é iniciado através do botão "*Select Rules*". O usuário pode criar as regras através de uma interface intuitiva, ou para o caso de um especialista em linguagem Prolog, pode construir as regras manualmente em um arquivo e posteriormente, carregá-las na aplicação. Em ambos os casos, o sistema cria automaticamente uma regra, denominada regra-base, para identificar se os elementos são os mesmos em ambas versões. Em outras palavras, a principal função de uma regra-base é garantir que as comparações serão feitas entre um mesmo elemento em ambas as versões (GARCIA, 2012).

Na abordagem "*Context Key*" do XChange, ao decidir por criar novas regras, é apresentada ao usuário, uma tela onde deve-se escolher qual é o atributo chave do seu contexto de negócio (GARCIA, 2012). Entretanto na abordagem proposta, este atributo chave é o subelemento ID, pois este caracteriza um identificador único do elemento e trata-se de um nodo pertencente ao elemento que se deseja realizar a inferência. Desta

forma, quando o usuário opta por criar novas regras, uma janela similar a apresentada na Figura 4.5 é aberta.

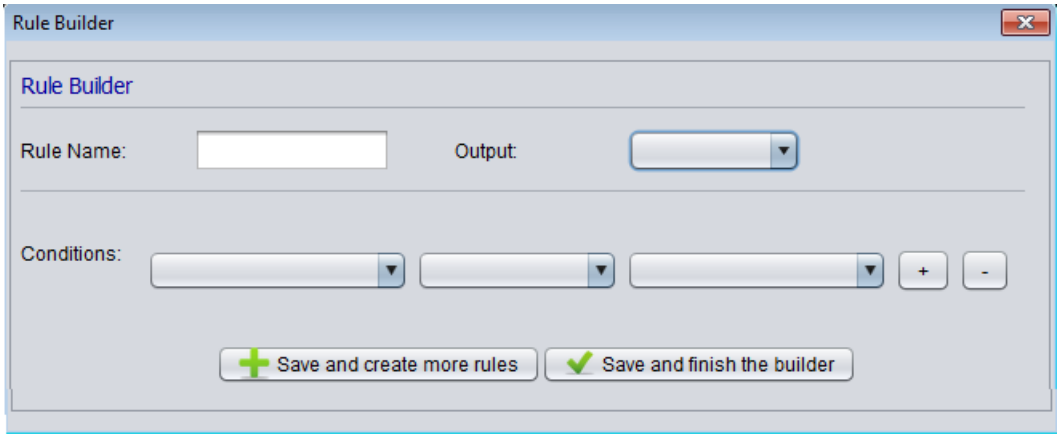


Figura 4.5: Criação de regras Prolog

Os campos da Figura 4.5 definem a estrutura da regra criada. No campo "*Rule Name*" deve-se definir o nome da regra a ser criada. A opção "*Output*" lista os atributos dos documentos XML e está responsável por definir qual atributo, dentre estes, é apresentado como saída para a regra. O campo "*Conditions*" define as condições para que a regra seja válida. A regra construída é validada se, e somente se, as condições estabelecidas nos campos seguintes também forem válidas (GARCIA, 2012). Os botões "+" e "-" possibilitam respectivamente, a adição e remoção de regras. Por fim, pode-se salvar e criar novas regras através da opção "*Save and create more rules*" ou salvar as regras e finalizar o construtor através do botão "*Save and finish the builder*".

Ao finalizar o construtor de regras, o XChange exibe a interface da Figura 4.6, que também é mostrada quando o usuário carrega as regras de um arquivo. A Figura 4.6 mostra a estrutura das regras existentes. O botão "*Identify Rules*" permite ao usuário identificar quais das regras exibidas serão usadas para o processo de inferência.

A Figura 4.7 representa as opções de escolha para que o usuário defina as regras utilizadas para inferência. As regras apresentadas encontram-se na Listagem A.7 da Apêndice.

Figura 4.6: Estrutura das regras para inferência.

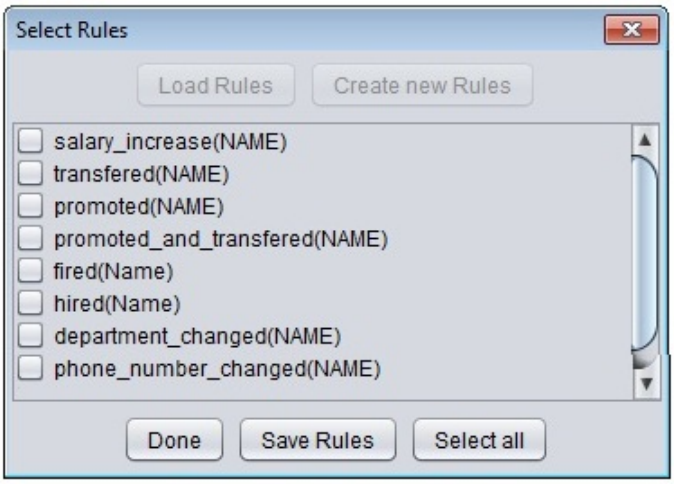


Figura 4.7: Seleção de regras.

4.2.5 Execução da inferência e compreensão dos resultados

Após a construção das regras, a base de conhecimento necessária para a realização da inferência está formada, ou seja, os fatos juntamente com a regra-base e as regras geradas pelo usuário constituem a teoria que alimenta o processo de inferência (GARCIA, 2012). O XChange exibe como saída o nome das regras, seguido pelos valores dos elementos para os quais as regras aplicadas são válidas. O resultado do processo de inferência é apresentado na parte inferior da interface do XChange e está representado na Figura 4.8.

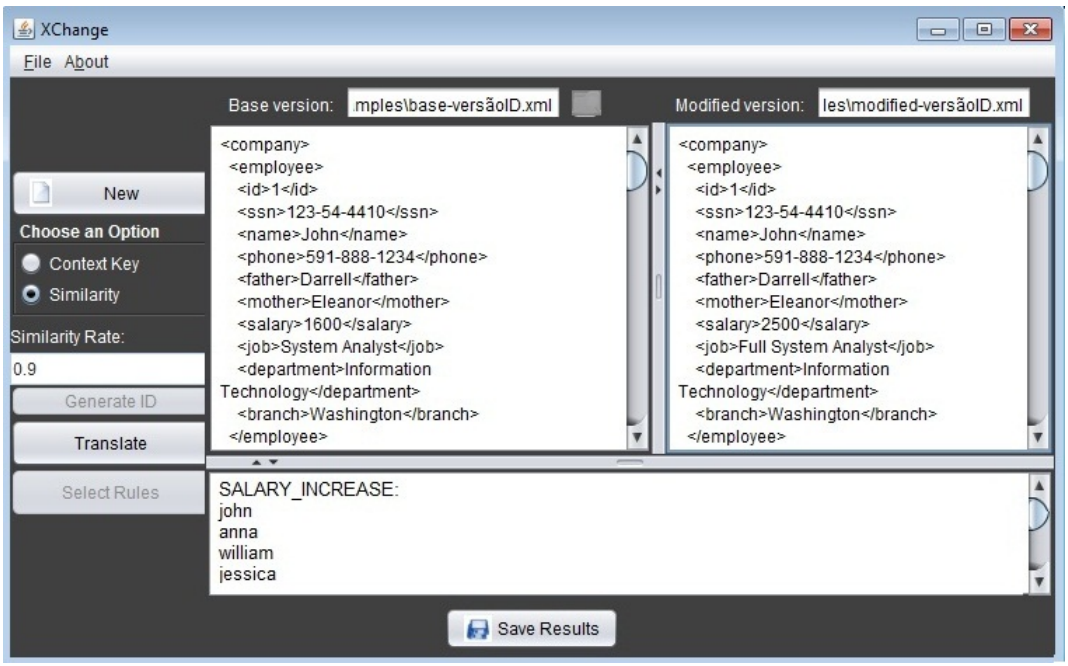


Figura 4.8: Resultado do processo de inferência.

A Listagem A.8, em anexo na Apêndice deste trabalho, apresenta todos os resultados inferidos através dos documentos de entrada. Desta forma, ao realizar inferência com base em fatos e regras, é possível identificar as mudanças ocorridas entre as versões e compreendê-las dentro do contexto do conteúdo que o documento XML representa.

4.3 Conclusão

Este capítulo apresentou os passos para realização de inferência baseada em similaridade. A abordagem proposta foi apresentada de forma prática na descrição de funcionamento do XChangeSim. Através das figuras apresentadas acompanhou-se um exemplo de utilização, assim como o passo-a-passo para manusear o XChange.

É possível observar que com o uso de similaridade, as versões são comparadas a nível estrutural com mais exatidão, em relação à abordagem de chave de contexto. Ao obter um grau de equivalência mais preciso, melhora-se também os resultados inferidos. Estes resultados trazem informações que ajudam a compreender as alterações ocorridas no processo de evolução do documento XML. Desta forma, tem-se também resultados a nível semântico que contribuem para o controle de mudanças em documentos XML.

5 Considerações Finais

O gerenciamento de mudanças em documentos XML tem se tornado cada vez mais importante. Entretanto, muitas abordagens existentes buscam caracterizar apenas as alterações estruturais dos documentos analisados, sem apresentar uma interpretação para as modificações encontradas.

Ao analisar os trabalhos relacionados a esta pesquisa, pode-se constatar que a maior parte dos estudos buscam resultados sintáticos. Porém, em documentos XML, compreender as informações resultantes do controle de mudanças entre versões de um documento é uma tarefa importante dentro dos inúmeros cenários que este formato pode representar. Visto isso, este trabalho apresentou uma abordagem de análise semântico-estrutural, a fim de encontrar significados implícitos nos resultados obtidos pela gerência de mudanças em documentos XML.

Este trabalho apresentou a abordagem XChangeSim que é baseada em similaridade e visa apoiar o contexto do projeto XChange. A análise por similaridade entre os elementos das versões garante maior precisão na equivalência de conteúdo e, através disto, melhores resultados ao se aplicar as regras que definem o contexto das modificações.

Foi apresentado um exemplo de utilização onde mostrou-se o funcionamento do XChangeSim e através das ações descritas pode-se obter resultados significativos para as alterações encontradas entre os documentos de entrada.

Para tanto, tornou-se notável a necessidade de escolher uma taxa de similaridade aceitável na abordagem XChangeSim, ou seja, um valor coerente. Trabalhos futuros podem melhorar a definição para a taxa de similaridade da aplicação, de forma que a interação do usuário com a aplicação seja cada vez mais intuitiva. Também espera-se que o projeto possa analisar vários documentos XML, visto que a versão atual do XChange trabalha com duas versões de um documento. Por fim, é possível realizar também um comparativo entre as abordagens de chave de contexto e similaridade presentes no XChange.

A Apêndice

As Listagens A.1 e A.2 apresentam os documentos XML presentes no exemplo de utilização desta pesquisa.

Listagem A.1: Exemplo de um documento XML - versão base

```
<?xml version="1.0" encoding="UTF-8"?>
<company>
   <employee>
      <ssn>123-54-4410</ssn>
      <name>John</name>
      <phone>591-888-1234</phone>
      <father>Darrell</father>
      <mother>Eleanor</mother>
      <salary>1600</salary>
      <job>System Analyst</job>
      <department>Information Technology</department>
      <branch>Washington</branch>
   </employee>
   <employee>
      <ssn>567-80-2314</ssn>
      <name>Harry</name>
      <phone>591-888-4334</phone>
      <father>Dick</father>
      <mother>Glenda</mother>
      <salary>1900</salary>
      <job>Support Help Desk</job>
      <department>Information Technology</department>
      <branch>Washington</branch>
   </employee>
   <employee>
      <ssn>234-65-6524</ssn>
      <name>Paul</name>
      <phone>531-888-6734</phone>
      <father>Clark</father>
      <mother>Evelyn</mother>
      <salary>2000</salary>
      <job>Programmer</job>
      <department>Information Technology</department>
      <branch>New York</branch>
   </employee>
   <employee>
```

```
37        <ssn>456-98-4734</ssn>
38        <name>Elizabeth</name>
39        <phone>431-888-6124</phone>
40        <father>Nathaniel</father>
41        <mother>Fiona</mother>
42        <salary>3900</salary>
43        <job>Manager</job>
44        <department>Sales</department>
45        <branch>Los Angeles</branch>
46     </employee>
47     <employee>
48        <ssn>146-02-6844</ssn>
49        <name>Anna</name>
50        <phone>351-888-9867</phone>
51        <father>Philip</father>
52        <mother>Tracy</mother>
53        <salary>3000</salary>
54        <job>Seller</job>
55        <department>Sales</department>
56        <branch>Chicago</branch>
57     </employee>
58     <employee>
59        <ssn>086-31-5948</ssn>
60        <name>William</name>
61        <phone>351-888-2217</phone>
62        <father>Norman</father>
63        <mother>Violet</mother>
64        <salary>3000</salary>
65        <job>Seller</job>
66        <department>Sales</department>
67        <branch>Chicago</branch>
68     </employee>
69     <employee>
70        <ssn>346-54-8921</ssn>
71        <name>Jessica</name>
72        <phone>351-888-2347</phone>
73        <father>Dean</father>
74        <mother>Wendy</mother>
75        <salary>2800</salary>
76        <job>System Analyst</job>
77        <department>Information Technology</department>
78        <branch>Chicago</branch>
79     </employee>
80     <employee>
81        <ssn>111-25-4811</ssn>
82        <name>Morris</name>
```

```
83        <phone>431-888-7125</phone>
84        <father>Raymond</father>
85        <mother>Zoe</mother>
86        <salary>3200</salary>
87        <job>Analyst</job>
88        <department>Information Technology</department>
89        <branch>Los Angeles</branch>
90     </employee>
91     <employee>
92        <ssn>149-74-3018</ssn>
93        <name>Bradley</name>
94        <phone>351-888-3288</phone>
95        <father>Nicholas</father>
96        <mother>Susanna</mother>
97        <salary>3000</salary>
98        <job>Test Analyst</job>
99        <department>Information Technology</department>
100       <branch>Chicago</branch>
101    </employee>
102 </company>
```

Listagem A.2: Exemplo de um documento XML - versão modificada

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <company>
3     <employee>
4        <ssn>123-54-4410</ssn>
5        <name>John</name>
6        <phone>591-888-1234</phone>
7        <father>Darrell</father>
8        <mother>Eleanor</mother>
9        <salary>2500</salary>
10       <job>Full System Analyst</job>
11       <department>Information Technology</department>
12       <branch>Washington</branch>
13    </employee>
14    <employee>
15       <ssn>567-80-2314</ssn>
16       <name>Harry</name>
17       <phone>531-888-8934</phone>
18       <father>Dick</father>
19       <mother>Glenda</mother>
20       <salary>1900</salary>
21       <job>Support Help Desk</job>
22       <department>Information Technology</department>
23       <branch>New York</branch>
```

```
24    </employee>
25    <employee>
26       <ssn>234-65-6524</ssn>
27       <name>Paul</name>
28       <phone>531-888-6734</phone>
29       <father>Clark</father>
30       <mother>Evelyn</mother>
31       <salary>2000</salary>
32       <job>Developer</job>
33       <department>Information Technology</department>
34       <branch>New York</branch>
35    </employee>
36    <employee>
37       <ssn>456-98-4734</ssn>
38       <name>Elizabeth</name>
39       <phone>431-888-6124</phone>
40       <father>Nathaniel</father>
41       <mother>Fiona</mother>
42       <salary>3900</salary>
43       <job>Manager</job>
44       <department>Marketing</department>
45       <branch>Los Angeles</branch>
46    </employee>
47    <employee>
48       <ssn>146-02-6844</ssn>
49       <name>Anna</name>
50       <phone>351-888-9867</phone>
51       <father>Philip</father>
52       <mother>Tracy</mother>
53       <salary>3900</salary>
54       <job>Seller</job>
55       <department>Sales</department>
56       <branch>Chicago</branch>
57    </employee>
58    <employee>
59       <ssn>086-31-5948</ssn>
60       <name>William</name>
61       <phone>431-888-6119</phone>
62       <father>Norman</father>
63       <mother>Violet</mother>
64       <salary>4000</salary>
65       <job>Manager</job>
66       <department>Sales</department>
67       <branch>Los Angeles</branch>
68    </employee>
69    <employee>
```

```
70        <ssn>076-42-1181</ssn>
71        <name>Leonard</name>
72        <phone>271-888-9127</phone>
73        <father>Simon</father>
74        <mother>Ursula</mother>
75        <salary>1450</salary>
76        <job>Seller</job>
77        <department>Sales</department>
78        <branch>Dallas</branch>
79     </employee>
80     <employee>
81        <ssn>346-54-8921</ssn>
82        <name>Jessica</name>
83        <phone>351-888-2347</phone>
84        <father>Dean</father>
85        <mother>Wendy</mother>
86        <salary>3800</salary>
87        <job>Test Analyst</job>
88        <department>Information Technology</department>
89        <branch>Chicago</branch>
90     </employee>
91  </company>
```

A Listagem A.3 apresenta o documento gerado pelo algoritmo Phoenix para os documentos de entrada do XChange. Na Listagem pode-se observar o grau de similaridade entre os atributos dos documentos XML, assim como os valores dos elementos de cada versão, quando estes não tiverem coincidência exata, isto é, o valor "diff:similarity" igual a 1.

Listagem A.3: Resultado da execução do Phoenix.

```
1  <company xmlns:diff="ic.uff.br/xmldiff" xmlns:left="ic.uff.br/xmldiff"
2  xmlns:right="ic.uff.br/xmldiff" diff:similarity="0.9698364">
3   <employee diff:similarity="0.9954756">
4     <ssn diff:similarity="1.0">123-54-4410</ssn>
5     <name diff:similarity="1.0">John</name>
6     <phone diff:similarity="1.0">591-888-1234</phone>
7     <father diff:similarity="1.0">Darrell</father>
8     <mother diff:similarity="1.0">Eleanor</mother>
9     <salary diff:similarity="0.875">
10      <diff:value diff:left="1600" diff:right="2500"/>
11    </salary>
12    <job diff:similarity="0.96212125">
13      <diff:value diff:left="System␣Analyst" diff:right="Full␣System␣Analyst"/>
14    </job>
```

```
15    <department diff:similarity="1.0">Information Technology</department>
16    <branch diff:similarity="1.0">Washington</branch>
17  </employee>
18  <employee diff:similarity="0.99209106">
19    <ssn diff:similarity="1.0">567-80-2314</ssn>
20    <name diff:similarity="1.0">Harry</name>
21    <phone diff:similarity="0.9375">
22      <diff:value diff:left="591-888-4334" diff:right="531-888-8934"/>
23    </phone>
24    <father diff:similarity="1.0">Dick</father>
25    <mother diff:similarity="1.0">Glenda</mother>
26    <salary diff:similarity="1.0">1900</salary>
27    <job diff:similarity="1.0">Support Help Desk</job>
28    <department diff:similarity="1.0">Information Technology</department>
29    <branch diff:similarity="0.7777778">
30      <diff:value diff:left="Washington" diff:right="New␣York"/>
31    </branch>
32  </employee>
33  <employee diff:similarity="0.99524856">
34    <ssn diff:similarity="1.0">234-65-6524</ssn>
35    <name diff:similarity="1.0">Paul</name>
36    <phone diff:similarity="1.0">531-888-6734</phone>
37    <father diff:similarity="1.0">Clark</father>
38    <mother diff:similarity="1.0">Evelyn</mother>
39    <salary diff:similarity="1.0">2000</salary>
40    <job diff:similarity="0.82894737">
41      <diff:value diff:left="Programmer" diff:right="Developer"/>
42    </job>
43    <department diff:similarity="1.0">Information Technology</department>
44    <branch diff:similarity="1.0">New York</branch>
45  </employee>
46  <employee diff:similarity="0.9950397">
47    <ssn diff:similarity="1.0">456-98-4734</ssn>
48    <name diff:similarity="1.0">Elizabeth</name>
49    <phone diff:similarity="1.0">431-888-6124</phone>
50    <father diff:similarity="1.0">Nathaniel</father>
51    <mother diff:similarity="1.0">Fiona</mother>
52    <salary diff:similarity="1.0">3900</salary>
53    <job diff:similarity="1.0">Manager</job>
54    <department diff:similarity="0.82142854">
55      <diff:value diff:left="Sales" diff:right="Marketing"/>
56    </department>
57    <branch diff:similarity="1.0">Los Angeles</branch>
58  </employee>
59  <employee diff:similarity="0.9982639">
60    <ssn diff:similarity="1.0">146-02-6844</ssn>
```

```
61    <name diff:similarity="1.0">Anna</name>
62    <phone diff:similarity="1.0">351-888-9867</phone>
63    <father diff:similarity="1.0">Philip</father>
64    <mother diff:similarity="1.0">Tracy</mother>
65    <salary diff:similarity="0.9375">
66      <diff:value diff:left="3000" diff:right="3900"/>
67    </salary>
68    <job diff:similarity="1.0">Seller</job>
69    <department diff:similarity="1.0">Sales</department>
70    <branch diff:similarity="1.0">Chicago</branch>
71  </employee>
72  <employee diff:similarity="0.98496854">
73    <ssn diff:similarity="1.0">086-31-5948</ssn>
74    <name diff:similarity="1.0">William</name>
75    <phone diff:similarity="0.9166667">
76      <diff:value diff:left="351-888-2217" diff:right="431-888-6119"/>
77    </phone>
78    <father diff:similarity="1.0">Norman</father>
79    <mother diff:similarity="1.0">Violet</mother>
80    <salary diff:similarity="0.9375">
81      <diff:value diff:left="3000" diff:right="4000"/>
82    </salary>
83    <job diff:similarity="0.8269231">
84      <diff:value diff:left="Seller" diff:right="Manager"/>
85    </job>
86    <department diff:similarity="1.0">Sales</department>
87    <branch diff:similarity="0.7777778">
88      <diff:value diff:left="Chicago" diff:right="Los␣Angeles"/>
89    </branch>
90  </employee>
91  <employee diff:similarity="0.9563625">
92    <ssn diff:similarity="0.8636364">
93      <diff:value diff:left="111-25-4811" diff:right="076-42-1181"/>
94    </ssn>
95    <name diff:similarity="0.8269231">
96      <diff:value diff:left="Morris" diff:right="Leonard"/>
97    </name>
98    <phone diff:similarity="0.9166667">
99      <diff:value diff:left="431-888-7125" diff:right="271-888-9127"/>
100   </phone>
101   <father diff:similarity="0.875">
102     <diff:value diff:left="Raymond" diff:right="Simon"/>
103   </father>
104   <mother diff:similarity="0.75">
105     <diff:value diff:left="Zoe" diff:right="Ursula"/>
106   </mother>
```

```
107     <salary diff:similarity="0.8125">
108       <diff:value diff:left="3200" diff:right="1450"/>
109     </salary>
110     <job diff:similarity="0.78846157">
111       <diff:value diff:left="Analyst" diff:right="Seller"/>
112     </job>
113     <department diff:similarity="0.787037">
114       <diff:value diff:left="Information␣Technology" diff:right="Sales"/>
115     </department>
116     <branch diff:similarity="0.8088235">
117       <diff:value diff:left="Los␣Angeles" diff:right="Dallas"/>
118     </branch>
119   </employee>
120   <employee diff:similarity="0.9966613">
121     <ssn diff:similarity="1.0">346-54-8921</ssn>
122     <name diff:similarity="1.0">Jessica</name>
123     <phone diff:similarity="1.0">351-888-2347</phone>
124     <father diff:similarity="1.0">Dean</father>
125     <mother diff:similarity="1.0">Wendy</mother>
126     <salary diff:similarity="0.9375">
127       <diff:value diff:left="2800" diff:right="3800"/>
128     </salary>
129     <job diff:similarity="0.9423077">
130       <diff:value diff:left="System␣Analyst" diff:right="Test␣Analyst"/>
131     </job>
132     <department diff:similarity="1.0">Information Technology</department>
133     <branch diff:similarity="1.0">Chicago</branch>
134   </employee>
135   <employee diff:side="left" diff:similarity="0.0">
136     <ssn diff:side="left" diff:similarity="0.0">
137       <diff:value diff:left="149-74-3018"/>
138     </ssn>
139     <name diff:side="left" diff:similarity="0.0">
140       <diff:value diff:left="Bradley"/>
141     </name>
142     <phone diff:side="left" diff:similarity="0.0">
143       <diff:value diff:left="351-888-3288"/>
144     </phone>
145     <father diff:side="left" diff:similarity="0.0">
146       <diff:value diff:left="Nicholas"/>
147     </father>
148     <mother diff:side="left" diff:similarity="0.0">
149       <diff:value diff:left="Susanna"/>
150     </mother>
151     <salary diff:side="left" diff:similarity="0.0">
152       <diff:value diff:left="3000"/>
```

```
153     </salary>
154     <job diff:side="left" diff:similarity="0.0">
155       <diff:value diff:left="Test␣Analyst"/>
156     </job>
157     <department diff:side="left" diff:similarity="0.0">
158       <diff:value diff:left="Information␣Technology"/>
159     </department>
160     <branch diff:side="left" diff:similarity="0.0">
161       <diff:value diff:left="Chicago"/>
162     </branch>
163   </employee>
164 </company>
```

As Listagens A.4 e A.5 representam os documentos XML com os novos atributos identificadores. Pode-se notar a equivalência dos elementos cujo atributo ID é o mesmo entre as listagens.

Listagem A.4: Exemplo do documento XML com ID - versão base

```
1  <company>
2    <employee>
3      <id>1</id>
4      <ssn>123-54-4410</ssn>
5      <name>John</name>
6      <phone>591-888-1234</phone>
7      <father>Darrell</father>
8      <mother>Eleanor</mother>
9      <salary>1600</salary>
10     <job>System Analyst</job>
11     <department>Information Technology</department>
12     <branch>Washington</branch>
13   </employee>
14   <employee>
15     <id>2</id>
16     <ssn>567-80-2314</ssn>
17     <name>Harry</name>
18     <phone>591-888-4334</phone>
19     <father>Dick</father>
20     <mother>Glenda</mother>
21     <salary>1900</salary>
22     <job>Support Help Desk</job>
23     <department>Information Technology</department>
24     <branch>Washington</branch>
25   </employee>
26   <employee>
27     <id>3</id>
```

```
28      <ssn>234-65-6524</ssn>
29      <name>Paul</name>
30      <phone>531-888-6734</phone>
31      <father>Clark</father>
32      <mother>Evelyn</mother>
33      <salary>2000</salary>
34      <job>Programmer</job>
35      <department>Information Technology</department>
36      <branch>New York</branch>
37    </employee>
38    <employee>
39      <id>4</id>
40      <ssn>456-98-4734</ssn>
41      <name>Elizabeth</name>
42      <phone>431-888-6124</phone>
43      <father>Nathaniel</father>
44      <mother>Fiona</mother>
45      <salary>3900</salary>
46      <job>Manager</job>
47      <department>Sales</department>
48      <branch>Los Angeles</branch>
49    </employee>
50    <employee>
51      <id>5</id>
52      <ssn>146-02-6844</ssn>
53      <name>Anna</name>
54      <phone>351-888-9867</phone>
55      <father>Philip</father>
56      <mother>Tracy</mother>
57      <salary>3000</salary>
58      <job>Seller</job>
59      <department>Sales</department>
60      <branch>Chicago</branch>
61    </employee>
62    <employee>
63      <id>6</id>
64      <ssn>086-31-5948</ssn>
65      <name>William</name>
66      <phone>351-888-2217</phone>
67      <father>Norman</father>
68      <mother>Violet</mother>
69      <salary>3000</salary>
70      <job>Seller</job>
71      <department>Sales</department>
72      <branch>Chicago</branch>
73    </employee>
```

```
  <employee>
    <id>7</id>
    <ssn>111-25-4811</ssn>
    <name>Morris</name>
    <phone>431-888-7125</phone>
    <father>Raymond</father>
    <mother>Zoe</mother>
    <salary>3200</salary>
    <job>Analyst</job>
    <department>Information Technology</department>
    <branch>Los Angeles</branch>
  </employee>
  <employee>
    <id>8</id>
    <ssn>346-54-8921</ssn>
    <name>Jessica</name>
    <phone>351-888-2347</phone>
    <father>Dean</father>
    <mother>Wendy</mother>
    <salary>2800</salary>
    <job>System Analyst</job>
    <department>Information Technology</department>
    <branch>Chicago</branch>
  </employee>
  <employee>
    <id>9</id>
    <ssn>149-74-3018</ssn>
    <name>Bradley</name>
    <phone>351-888-3288</phone>
    <father>Nicholas</father>
    <mother>Susanna</mother>
    <salary>3000</salary>
    <job>Test Analyst</job>
    <department>Information Technology</department>
    <branch>Chicago</branch>
  </employee>
</company>
```

Listagem A.5: Exemplo do documento XML com ID - versão modificada

```
<company>
  <employee>
    <id>1</id>
    <ssn>123-54-4410</ssn>
    <name>John</name>
    <phone>591-888-1234</phone>
```

```
7     <father>Darrell</father>
8     <mother>Eleanor</mother>
9     <salary>2500</salary>
10    <job>Full System Analyst</job>
11    <department>Information Technology</department>
12    <branch>Washington</branch>
13  </employee>
14  <employee>
15    <id>2</id>
16    <ssn>567-80-2314</ssn>
17    <name>Harry</name>
18    <phone>531-888-8934</phone>
19    <father>Dick</father>
20    <mother>Glenda</mother>
21    <salary>1900</salary>
22    <job>Support Help Desk</job>
23    <department>Information Technology</department>
24    <branch>New York</branch>
25  </employee>
26  <employee>
27    <id>3</id>
28    <ssn>234-65-6524</ssn>
29    <name>Paul</name>
30    <phone>531-888-6734</phone>
31    <father>Clark</father>
32    <mother>Evelyn</mother>
33    <salary>2000</salary>
34    <job>Developer</job>
35    <department>Information Technology</department>
36    <branch>New York</branch>
37  </employee>
38  <employee>
39    <id>4</id>
40    <ssn>456-98-4734</ssn>
41    <name>Elizabeth</name>
42    <phone>431-888-6124</phone>
43    <father>Nathaniel</father>
44    <mother>Fiona</mother>
45    <salary>3900</salary>
46    <job>Manager</job>
47    <department>Marketing</department>
48    <branch>Los Angeles</branch>
49  </employee>
50  <employee>
51    <id>5</id>
52    <ssn>146-02-6844</ssn>
```

```
53      <name>Anna</name>
54      <phone>351-888-9867</phone>
55      <father>Philip</father>
56      <mother>Tracy</mother>
57      <salary>3900</salary>
58      <job>Seller</job>
59      <department>Sales</department>
60      <branch>Chicago</branch>
61    </employee>
62    <employee>
63      <id>6</id>
64      <ssn>086-31-5948</ssn>
65      <name>William</name>
66      <phone>431-888-6119</phone>
67      <father>Norman</father>
68      <mother>Violet</mother>
69      <salary>4000</salary>
70      <job>Manager</job>
71      <department>Sales</department>
72      <branch>Los Angeles</branch>
73    </employee>
74    <employee>
75      <id>7</id>
76      <ssn>076-42-1181</ssn>
77      <name>Leonard</name>
78      <phone>271-888-9127</phone>
79      <father>Simon</father>
80      <mother>Ursula</mother>
81      <salary>1450</salary>
82      <job>Seller</job>
83      <department>Sales</department>
84      <branch>Dallas</branch>
85    </employee>
86    <employee>
87      <id>8</id>
88      <ssn>346-54-8921</ssn>
89      <name>Jessica</name>
90      <phone>351-888-2347</phone>
91      <father>Dean</father>
92      <mother>Wendy</mother>
93      <salary>3800</salary>
94      <job>Test Analyst</job>
95      <department>Information Technology</department>
96      <branch>Chicago</branch>
97    </employee>
98    <employee>
```

```
99      <id>10</id>
100   </employee>
101 </company>
```

A Listagem A.6 exibe os fatos obtidos após a tradução dos documentos XML com identificadores.

Listagem A.6: Fatos traduzidos

```
1  company(base).
2  employee(base,id2).
3  id(id2,'1').
4  ssn(id2,'123-54-4410').
5  name(id2,'john').
6  phone(id2,'591-888-1234').
7  father(id2,'darrell').
8  mother(id2,'eleanor').
9  salary(id2,1600).
10 job(id2,'system_analyst').
11 department(id2,'information_technology').
12 branch(id2,'washington').
13 employee(base,id3).
14 id(id3,'2').
15 ssn(id3,'567-80-2314').
16 name(id3,'harry').
17 phone(id3,'591-888-4334').
18 father(id3,'dick').
19 mother(id3,'glenda').
20 salary(id3,1900).
21 job(id3,'support_help_desk').
22 department(id3,'information_technology').
23 branch(id3,'washington').
24 employee(base,id4).
25 id(id4,'3').
26 ssn(id4,'234-65-6524').
27 name(id4,'paul').
28 phone(id4,'531-888-6734').
29 father(id4,'clark').
30 mother(id4,'evelyn').
31 salary(id4,2000).
32 job(id4,'programmer').
33 department(id4,'information_technology').
34 branch(id4,'new_york').
35 employee(base,id5).
36 id(id5,'4').
37 ssn(id5,'456-98-4734').
```

```
38 name(id5,'elizabeth').
39 phone(id5,'431-888-6124').
40 father(id5,'nathaniel').
41 mother(id5,'fiona').
42 salary(id5,3900).
43 job(id5,'manager').
44 department(id5,'sales').
45 branch(id5,'los_angeles').
46 employee(base,id6).
47 id(id6,'5').
48 ssn(id6,'146-02-6844').
49 name(id6,'anna').
50 phone(id6,'351-888-9867').
51 father(id6,'philip').
52 mother(id6,'tracy').
53 salary(id6,3000).
54 job(id6,'seller').
55 department(id6,'sales').
56 branch(id6,'chicago').
57 employee(base,id7).
58 id(id7,'6').
59 ssn(id7,'086-31-5948').
60 name(id7,'william').
61 phone(id7,'351-888-2217').
62 father(id7,'norman').
63 mother(id7,'violet').
64 salary(id7,3000).
65 job(id7,'seller').
66 department(id7,'sales').
67 branch(id7,'chicago').
68 employee(base,id8).
69 id(id8,'7').
70 ssn(id8,'111-25-4811').
71 name(id8,'morris').
72 phone(id8,'431-888-7125').
73 father(id8,'raymond').
74 mother(id8,'zoe').
75 salary(id8,3200).
76 job(id8,'analyst').
77 department(id8,'information_technology').
78 branch(id8,'los_angeles').
79 employee(base,id9).
80 id(id9,'8').
81 ssn(id9,'346-54-8921').
82 name(id9,'jessica').
83 phone(id9,'351-888-2347').
```

```
84  father(id9,'dean').
85  mother(id9,'wendy').
86  salary(id9,2800).
87  job(id9,'system_analyst').
88  department(id9,'information_technology').
89  branch(id9,'chicago').
90  employee(base,id10).
91  id(id10,'9').
92  ssn(id10,'149-74-3018').
93  name(id10,'bradley').
94  phone(id10,'351-888-3288').
95  father(id10,'nicholas').
96  mother(id10,'susanna').
97  salary(id10,3000).
98  job(id10,'test_analyst').
99  department(id10,'information_technology').
100 branch(id10,'chicago').
101
102 company(modified).
103 employee(modified,id11).
104 id(id11,'1').
105 ssn(id11,'123-54-4410').
106 name(id11,'john').
107 phone(id11,'591-888-1234').
108 father(id11,'darrell').
109 mother(id11,'eleanor').
110 salary(id11,2500).
111 job(id11,'full_system_analyst').
112 department(id11,'information_technology').
113 branch(id11,'washington').
114 employee(modified,id12).
115 id(id12,'2').
116 ssn(id12,'567-80-2314').
117 name(id12,'harry').
118 phone(id12,'531-888-8934').
119 father(id12,'dick').
120 mother(id12,'glenda').
121 salary(id12,1900).
122 job(id12,'support_help_desk').
123 department(id12,'information_technology').
124 branch(id12,'new_york').
125 employee(modified,id13).
126 id(id13,'3').
127 ssn(id13,'234-65-6524').
128 name(id13,'paul').
129 phone(id13,'531-888-6734').
```

```
130 father(id13,'clark').
131 mother(id13,'evelyn').
132 salary(id13,2000).
133 job(id13,'developer').
134 department(id13,'information_technology').
135 branch(id13,'new_york').
136 employee(modified,id14).
137 id(id14,'4').
138 ssn(id14,'456-98-4734').
139 name(id14,'elizabeth').
140 phone(id14,'431-888-6124').
141 father(id14,'nathaniel').
142 mother(id14,'fiona').
143 salary(id14,3900).
144 job(id14,'manager').
145 department(id14,'marketing').
146 branch(id14,'los_angeles').
147 employee(modified,id15).
148 id(id15,'5').
149 ssn(id15,'146-02-6844').
150 name(id15,'anna').
151 phone(id15,'351-888-9867').
152 father(id15,'philip').
153 mother(id15,'tracy').
154 salary(id15,3900).
155 job(id15,'seller').
156 department(id15,'sales').
157 branch(id15,'chicago').
158 employee(modified,id16).
159 id(id16,'6').
160 ssn(id16,'086-31-5948').
161 name(id16,'william').
162 phone(id16,'431-888-6119').
163 father(id16,'norman').
164 mother(id16,'violet').
165 salary(id16,4000).
166 job(id16,'manager').
167 department(id16,'sales').
168 branch(id16,'los_angeles').
169 employee(modified,id17).
170 id(id17,'7').
171 ssn(id17,'076-42-1181').
172 name(id17,'leonard').
173 phone(id17,'271-888-9127').
174 father(id17,'simon').
175 mother(id17,'ursula').
```

```
176 salary(id17,1450).
177 job(id17,'seller').
178 department(id17,'sales').
179 branch(id17,'dallas').
180 employee(modified,id18).
181 id(id18,'8').
182 ssn(id18,'346-54-8921').
183 name(id18,'jessica').
184 phone(id18,'351-888-2347').
185 father(id18,'dean').
186 mother(id18,'wendy').
187 salary(id18,3800).
188 job(id18,'test_analyst').
189 department(id18,'information_technology').
190 branch(id18,'chicago').
191 employee(modified,id19).
192 id(id19,'10').
```

A Listagem A.7 apresenta as regras utilizadas no exemplo desta pesquisa. Estas regras serviram como base para o processo de inferência do projeto.

Listagem A.7: Regras utilizadas

```
1  salary_increase(NAME) :- same_element(Fb,Fm,ID), salary(Fb,SALARYBase),
2  salary(Fm,SALARYModified), SALARYBase<SALARYModified, name(Fb,NAME).
3
4  transfered(NAME) :- same_element(Fb,Fm,ID), branch(Fb,BRANCHBase),
5  branch(Fm,BRANCHModified), BRANCHBase\=BRANCHModified,name(Fb,NAME).
6
7  promoted(NAME) :- same_element(Fb,Fm,ID), job(Fb,JOBBase),
8  job(Fm,JOBModified), JOBBase\=JOBModified, name(Fb,NAME),
9  salary_increase(NAME).
10
11 promoted_and_transfered(NAME) :- promoted(NAME), transfered(NAME).
12
13 fired(Name) :-  exists_in_base(ID), not(exists_in_modified(ID)),
14 id(Fb,ID),  name(Fb,Name).
15
16 hired(Name) :-  exists_in_modified(ID), not(exists_in_base(ID)),
17 id(Fm,ID),  name(Fm,Name).
18
19 department_changed(NAME) :- same_element(Fb,Fm,ID),
20 department(Fb,DEPARTMENTBase), department(Fm,DEPARTMENTModified),
21 DEPARTMENTBase\=DEPARTMENTModified, name(Fb,NAME).
22
23 phone_number_changed(NAME) :- same_element(Fb,Fm,ID),
```

```
24 phone(Fb,PHONEBase), phone(Fm,PHONEModified),
25 PHONEBase\=PHONEModified, name(Fb,NAME).
```

A Listagem A.8 expõe os resultados obtidos após o processo de inferência. Os resultados mostram as mudanças ocorridas entre as versões, dentro do contexto de negócio do documento XML, baseando-se nas regras de inferência.

Listagem A.8: Resultados da inferência

```
1  SALARY_INCREASE:
2  john
3  anna
4  william
5  jessica
6  TRANSFERED:
7  harry
8  william
9  morris
10 PROMOTED:
11 john
12 william
13 jessica
14 PROMOTED_AND_TRANSFERED:
15 william
16 FIRED:
17 bradley
18 HIRED:
19 DEPARTMENT_CHANGED:
20 elizabeth
21 morris
22 PHONE_NUMBER_CHANGED:
23 harry
24 william
25 morris
```

Referências Bibliográficas

ABITEBOUL, S.; CLUET, S.; FERRAN, G. ; ROUSSET, M.-C. The Xyleme Project. **Computer Networks**, 2002.

BRAY, T.; PAOLI, J.; SPERBERG-MCQUEEN, C. M.; MALER, E. ; YERGEAU, F. Extensible markup language (XML) 1.0 (fifth edition). **World Wide Web Consortium (W3C), November**, 2008.

CAMPELLO, B.; PINTO, F. **Phoenix: uma abordagem para cálculo de similaridade de documentos XML.** Trabalho de Conclusão de Curso - Universidade Federal Fluminense, 2012.

COBÉNA, G.; ABITEBOUL, S. ; MARIAN, A. **Detecting changes in XML documents**. In: Data Engineering, 2002. Proceedings. 18th International Conference on, p. 41-52, 2002.

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L. ; STEIN, C. **Introduction to Algorithms, Third Edition**. 3rd. ed., The MIT Press, 2009.

DING, Y.; FOO, S. A review of ontology mapping and evolving. **Journal of Information Science**, 2002.

GARCIA, P. A. EDX: Uma Abordagem de Apoio ao Controle de Mudanças de Documentos XML. **Trabalho de Conclusão de Curso - Universidade Federal de Juiz de Fora**, 2012.

HARA, C.; SANTOS, R. C. Utilização de chaves em algoritmos de detecção de diferenças para documentos XML. **In: XXII Simpósio Brasileiro de Banco de Dados**, 2007.

Institute of Electrical and Electronics Engineers. **IEEE Standard for Software Configuration Management Plans**. IEEE, 2005.

LEVENSHTEIN, V. I. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. **Soviet Physics Doklady.**, v.10, n.8, p. 707–710, February 1966.

MADRIA, S. K.; VIYANON, W. ; BHOWMICK, S. S. **XML data integration based on content and structure similarity using keys**. In: Proceedings of the OTM 2008 Confederated International Conferences, CoopIS, DOA, GADA, IS, and ODBASE 2008. Part I on On the Move to Meaningful Internet Systems: p. 484 - 493. 2008.

MADRIA, S. K.; VIYANON, W. **XDI-CSSK: a system for detecting XML similarity on content and structure using relational database.** Technical report, Technical Report, Dept of Missouri University of Science and Technology., 2009.

MADRIA, S. K.; VIYANON, W. **XML-SIM: Structure and content semantic similarity detection using keys**. In: On the Move to Meaningful Internet Systems, OTM. Lecture Notes in Computer Science v. 6427, 2010, p. 1061-1078. 2010.

NOLL, R. P.; SACCOL, D. D. B. ; EDELWEISS, N. **Uma proposta para análise de similaridade entre documentos XML ontologias definidas em OWL**. 2007. Dissertação de Mestrado - Universidade Federal do Rio Grande do Sul.

OLIVEIRA, A. M. D.; MURTA, L. ; BRAGANHOLO, V. Uso de inferência na evolução de documentos semi-estruturados no contexto de controle de modificações. **In: Simpósio Brasileiro de Banco de Dados (SBBD), 2012, São Paulo, SP: SBC**, 2012.

PORTELA, M. M.; NOGUEIRA, J. A. B. **Integrando evolução semântica e similaridade de documentos XML.** Trabalho Final da Disciplina de Gerência de Dados Semi-estruturados. Universidade Federal Fluminense, 2012.

PRESSMAN, R. S. **Engenharia de Software - Uma Abordagem Profissional**. 7. ed., Rio de Janeiro: ARTMED - McGraw-Hill, 2011.

SOUZA, A. B. D.; SILVA, J. P. D.; OLIVEIRA, W. C. C. D.; KUMA, T. H. ; SILVEIRA, I. F. Recuperação semântica de objetos de aprendizagem: Uma abordagem baseada em tesauros de propósito genérico. **In: Simpósio Brasileiro de Informática na Educação (SBIE), Florianópolis**, 2008.

VINSON, A. R. **PathSim: um algoritmo para calcular a similaridade entre caminhos XML**. Dissertação de Mestrado em Ciência da Computação, Universidade Federal do Rio Grande do Sul, 2007.

WANG, Y.; DEWITT, D. J. ; CAI, J.-Y. **X-Diff: An effective change detection algorithm for XML documents**. In: Data Engineering, 2003. Proceedings. 19th International Conference on, 2003.