XTool: Uma Ferramenta de Teste Baseado em Defeitos para Esquemas de Dados

Igor F. Nazar[1], Maria Claudia F. P. Emer[2], Silvia R. Vergilio[1], Mario Jino[2]

[1]DInf –UFPR, CP: 19081, CEP: 81.531-970 – Curitiba – PR – Brasil

[2]DCA – FEEC – UNICAMP, CP:6101, CEP: 13.083-970 – Campinas – SP - Brasil

{igor,silvia}@inf.ufpr.br, {mcemer,jino}@dca.fee.unicamp.Br

Abstract *This work describes a tool, named XTool, which tests data schemas considering previously defined classes of faults. XTool automatically generates test data composed by data instances and queries to these instances. The implemented approach allows the test of different kind of schemas. XTool contributes to improve the quality of the software applications, by ensuring the integrity of the data that it manipulates and that is defined by the schema under test.*

Resumo. *Este trabalho descreve a ferramenta, denominada XTool, que testa esquemas de dados considerando classes de defeitos previamente definidas.* A *XTool gera automaticamente dados de teste formados por instâncias de dados e consultas a essas instâncias. A abordagem implementada pela ferramenta pode ser utilizada em diferentes tipos de esquemas. O objetivo do teste realizado pela XTool é contribuir para melhorar a qualidade das aplicações de software, assegurando a integridade dos dados que ela manipula e que são definidos pelo esquema em teste.*

1. Introdução

O uso de sistemas baseados em computação exige o desenvolvimento de software com alto padrão de qualidade. Um aspecto importante para garantir a qualidade de um sistema está relacionado à integridade das informações que a aplicação manipula, tanto para a integração entre aplicações quanto para armazenamento de dados. Esses dados, em geral, são definidos por um esquema, que contém regras referentes à estrutura lógica dos dados permitida. No contexto de integração de sistemas, destaca-se a linguagem XML (*eXtensible Markup Language* (W3C 2006a)) utilizada para padronização de documentos. Um exemplo de esquema XML bastante utilizado é o XML *Schema* (W3C 2006b). No contexto de armazenamento de dados, destacam-se bases de dados relacionais associadas a esquemas descritos por meio do Modelo Entidade-Relacionamento (MER) (Chen 1976).

Geralmente os dados são validados por *parsers* de acordo com o esquema. Entretanto, para garantir a qualidade dos dados isso não é suficiente, pois um esquema pode estar incorreto com relação à especificação e escrito de maneira sintaticamente correta, ou seja, ser válido. Um esquema incorreto pode validar dados incorretos que quando recebidos pela aplicação produzem uma falha, ou seja, o esquema incorreto permite que dados incorretos sejam considerados válidos ou o contrário.

Uma forma de evitar isto é realizar o teste do esquema, após a sua especificação ou atualização. Na literatura, a maioria dos trabalhos existentes se dedica somente a testar as aplicações. No contexto de esquemas de dados relacionais, eles visam à

geração de dados, ao teste do projeto e da aplicação de base de dados (Chan e Cheung 1999, Chays et al. 2000, Chays e Deng 2003, Deng et al. 2005, Kapfhammer e Soffa 2003, Suárez-Cabal e Tuya 2004, Zhang et al. 2001) e quando consideram o esquema é somente como fonte de informações para a obtenção de dados de teste. (Chan et al. 2005, Robbert e Maryanski 1991). No contexto de esquemas XML, a maioria dos trabalhos tem como objetivo testar a interação de componentes que se comunicam utilizando a linguagem XML (Lee e Offutt 2001) e serviços Web (Offutt e Xu 2004, Xu et al. 2005).

Todos esses trabalhos não têm como foco o teste de esquema. São poucos os trabalhos que se dedicam a esse tema. Outro aspecto a ser ressaltado é que é mais raro ainda encontrar ferramentas que auxiliem na realização dessa tarefa. Nesse contexto, destacam-se as ferramentas *RDBTool* (*Relational Data-Base Testing Tool*) (Aranha et al. 2000) e *XTM* (*Tool for XML Schema Testing Based on Mutation*) (Franzotte e Vergilio 2006). A primeira ferramenta permite o teste estrutural em esquemas de base de dados relacional, que visa testar associações entre definições e usos de atributos em um esquema. A *XTM* apóia o teste de mutação em esquemas escritos em XML *Schema*. Essa ferramenta implementa operadores de mutação que provocam ligeiras modificações no esquema em teste, gerando esquemas mutantes. Os dados de teste são gerados para provocar um comportamento diferente entre os esquemas mutantes e o esquema original. Esse comportamento diferente está em validar ou não os dados.

Ambas as ferramentas são úteis e facilitam a aplicação de uma abordagem de teste para os respectivos contextos focalizados. Entretanto, elas apresentam algumas limitações tais como: 1) a abordagem implementada por estas ferramentas não pode ser diretamente aplicada a outros tipos de esquemas, tais como objeto-relacional, esquemas DTD (*Document Type Definition*), etc. 2) elas não permitem a geração automática de dados de teste, uma das mais custosas tarefas e consumidora de esforço para aplicação de uma abordagem ou critério de teste.

Para lidar com essas limitações, este artigo descreve a ferramenta *XTool* (*XML and Relational Database Schema Testing Tool*), que implementa uma abordagem baseada em defeitos genérica denominada Análise de Instâncias de Dados Alternativas (AIDA). Essa abordagem é baseada em defeitos porque considera classes de defeitos comumente presentes em esquemas. É genérica porque está baseada em um modelo formal que permite a representação de diferentes tipos de esquemas (Emer et al. 2007a). Na abordagem de teste AIDA, uma instância de dados associada ao esquema em teste sofre alterações simples gerando instâncias de dados alternativas. Essas instâncias de dados são geradas com base em padrões especificados nas classes de defeitos previamente identificadas. Consultas às instâncias de dados alternativas também são geradas a partir de padrões definidos nas classes de defeitos. As instâncias de dados representam os possíveis defeitos no esquema e as consultas são capazes de revelar esses defeitos. Portanto, a abordagem é denominada Análise de Instâncias de Dados Alternativas devido ao uso de instâncias de dados alternativas para detectar defeitos em esquemas de dados.

O artigo é organizado como segue: a Seção 2 descreve alguns trabalhos relacionados; a Seção 3 descreve a ferramenta *XTool*, a abordagem implementada e suas principais funcionalidades; a Seção 4 apresenta um exemplo de utilização da *XTool* no teste de um esquema XML; a Seção 5 contém resultados de estudos de caso realizados com o apoio da *XTool*; a Seção 6 apresenta as conclusões e trabalhos futuros.

2. Trabalhos Relacionados

O teste em aplicações de base de dados envolve aspectos relacionados à correção, tais como: comportamento da aplicação em relação à especificação; esquema da base de dados em relação aos dados do mundo real modelados; acurácia na base de dados, segurança e privacidade; e execução correta de operações de inserção, atualização e exclusão de dados (Chays et al. 2000). Nesse contexto, existem vários estudos sendo conduzidos para o teste dessas aplicações. Esses estudos envolvem geração de dados, teste do projeto e da aplicação de base de dados (Chan e Cheung 1999, Chays et al. 2000, Chays e Deng 2003, Deng et al. 2005, Kapfhammer e Soffa 2003, Suárez-Cabal e Tuya 2004, Zhang et al. 2001); alguns estudos exploram o uso de informações do esquema para o teste da aplicação de base de dados (Chan et al. 2005, Robbert e Maryanski 1991). Nesse contexto de base de dados relacional, destaca-se o trabalho de Aranha et al. (2000) que aborda o teste de esquema.

Aranha et al. (2000) descrevem a ferramenta *RDBTool* (*Relational Data-Base Testing Tool*), que realiza teste em base de dados relacional para focalizar a estrutura lógica dos dados e os próprios dados. O teste é executado por meio de operações (sentenças SQL) na base de dados para revelar defeitos na definição de atributos e relacionamentos especificados no esquema, empregando critério de teste estrutural. A ferramenta *RDBTool* executa a análise estática do esquema da base de dados em teste, extraindo dados sobre a definição de atributos e relacionamentos e determinando os elementos requeridos com o auxílio do testador, que deve escolher os elementos que serão testados e os critérios de teste que serão empregados. Um exemplo de critério de teste proposto nesse trabalho é o critério "todos os tipos de definição". A partir destas informações são geradas as sentenças SQL para testar cada elemento requerido e executar o teste. O testador analisa os resultados do teste para avaliar a cobertura dos critérios e possíveis defeitos no esquema. A *RDBTool* pode ser aplicada somente no teste de esquemas de base de dados relacional e não gera automaticamente as sentenças SQL.

No contexto de esquemas XML, vários trabalhos abordam o teste de aplicações Web (Di Lucca e Di Penta 2003, Di Lucca et al. 2002a, Di Lucca et al. 2002b, Kung et al. 2000, Liu et al. 2000a, Liu et al. 2000b, Ricca e Tonella 2002); e outros trabalhos testam somente alguns aspectos dessas aplicações; por exemplo, a interação entre componentes Web por meio de mensagens XML (Lee e Offutt 2001) e serviços Web (Offutt e Xu 2004, Xu et al. 2005). Nesse contexto, destacam-se os trabalhos de Li e Muller (2005) e Franzotte e Vergilio (2006) que visam o teste de esquemas.

Li e Muller (2005) propõem operadores de mutação para XML *Schema* que fazem alterações simples no esquema, como troca de um valor ou atributo. Os autores mostram que alguns desses operadores geram modificações que não são descobertas apenas validando os documentos XML, mas não apresentam um processo de teste nem uma ferramenta de aplicação.

Franzotte e Vergilio (2006) introduzem um conjunto de operadores de mutação e aplicam o teste de mutação em esquemas XML com o apoio de uma ferramenta denominada *XTM* (*Tool for XML Schema Testing Based on Mutation*). Os operadores de mutação são subdivididos em operadores de mutação elementar, que alteram valores de atributos e elementos modificando a ordem, o uso e o tipo de dado, por exemplo; e operadores de mutação estrutural, que modificam a estrutura da árvore de um esquema, invertendo, adicionando e removendo nós. Resultados promissores são apresentados, visto ser a técnica baseada em defeitos bastante eficaz em termos de defeitos revelados.

Entretanto, a utilização da *XTM* e de seus operadores é limitada a esquemas XML escritos em XML *Schema*. Além disso, a geração de dados de teste precisa ser feita manualmente pelo testador, que deverá também manualmente identificar esquemas equivalentes.

Ambas as ferramentas (*RDBTool* e *XTM*) só podem ser aplicadas em determinados contextos e não oferecem apoio automatizado à difícil tarefa de geração de dados de teste. Para lidar com essas limitações a ferramenta *XTool* foi implementada e será descrita na próxima seção.

3. Ferramenta *XTool*

A *XTool* (*XML and Relational Database Schema Testing Tool*) apóia o teste de esquemas de dados implementando uma abordagem baseada em defeitos, denominada Análise de Instâncias de Dados Alternativas (AIDA) (Emer et al. 2007a). Essa abordagem tem como objetivo detectar defeitos em esquemas de dados para garantir a integridade dos dados definidos por meio do esquema e manipulados por uma aplicação de software.

A AIDA é genérica, ou seja, pode ser aplicada em esquemas de dados de diferentes contextos, pois emprega um modelo de dados definido por um metamodelo *MM* baseado em MOF (*Meta Object Facility*) (OMG 2006) que representa os esquemas de dados, instanciando e interpretando os componentes do esquema. A Figura 1 apresenta o metamodelo *MM*, que consiste das classes: Elemento (entidades ou objetos), Atributo (características dos elementos), Restrição (restrições associadas aos elementos e atributos) e, da classe associativa: Associação (características referentes à classe Elemento).

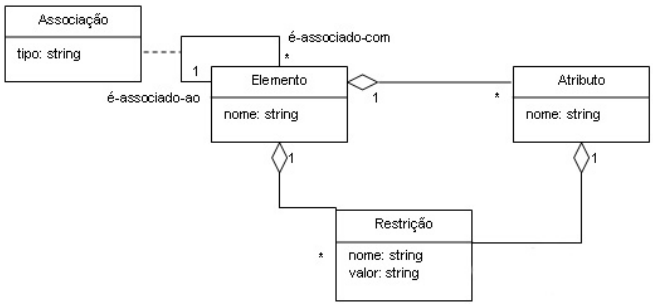

Figura 1. Metamodelo *MM*

Além disso, a AIDA considera defeitos comuns, que podem ser introduzidos no esquema durante o seu desenvolvimento, e a representação formal, que permite a identificação automática de possíveis defeitos. Os defeitos são organizados em classes de defeitos (Tabela 1). As classes de defeitos foram determinadas a partir de inspeções realizadas em esquemas de dados e com base em trabalhos da literatura, tais como: Lee e Offutt (2001) e Offutt e Xu (2004). Um exemplo de classe de defeito relacionada à restrição de domínio é a classe denominada "Valores Enumerados Incorretos", que diz respeito à definição incorreta de um conjunto de valores permitidos para elementos ou atributos. A representação formal providencia a identificação dos elementos, atributos, restrições e relacionamentos entre eles. Desse modo, um esquema de dados é denotado por $S = (E, A, R, P)$, onde:

- E é um conjunto finito de elementos.
- A é um conjunto finito de atributos.

- R é um conjunto finito de restrições referentes ao domínio, definição, relacionamento e conteúdo de elementos ou de atributos.
- P é um conjunto de regras de associação que relacionam elementos, atributos e restrições. Uma regra de P pode ter um dos seguintes formatos. Considere $U = E \cup A$.
  - $p(x, y) \mid x, y \in E \wedge x \neq y;$
  - $p(x, r) \mid x \in E \wedge r \in R;$
  - $p(x, r, SU) \mid x \in U \wedge r \in R \wedge SU = \{u_1, u_2, ..., u_m\} \subset U, \quad \forall u_i \neq x,\ 1 \leq i \leq m, m \geq 1,$ onde m é o número de elementos e atributos de SU.

Tabela 1. Restrições e classes de defeitos

Classe de defeito	Descrição
Grupo 1 - Restrições de domínio	
Tipo de dado incorreto	Definição incorreta de tipo de dado de um elemento
Valor incorreto	Valor padrão ou fixo definido incorretamente para o elemento
Valores enumerados incorretos	Definição incorreta do conjunto de valores permitidos para o elemento
Valores máximos e mínimos incorretos	Definição incorreta de valores limites definidos para o elemento
Tamanho incorreto	Número de caracteres máximos e mínimos permitidos para um elemento definidos incorretamente
Limite de dígitos incorreto	Número de dígitos máximos e mínimos permitidos para um elemento definidos incorretamente
Modelo de valores incorreto	Definição incorreta da seqüência de valores permitidos para um elemento
Tratamento de espaço em branco incorreto	Definição incorreta do tratamento para espaço em branco
Grupo 2 - Restrições de definição	
Uso incorreto	Definição incorreta de um elemento como opcional ou obrigatório
Unicidade incorreta	Definição incorreta do número de ocorrências permitidas para um elemento
Identificador incorreto	Definição incorreta de uso e unicidade para elementos que compõem o identificador de um elemento complexo
Grupo 3 - Restrições de relacionamento	
Ocorrência incorreta	Definição incorreta do número máximo e mínimo de repetições de um elemento
Ordem incorreta	Definição incorreta da ordem de ocorrência para os elementos-filho de um determinado elemento
Associação Incorreta	Definição incorreta de uma associação: cardinalidade, generalização/especialização, agregação, elemento associativo
Grupo 4 - Restrições semânticas	
Condição incorreta	Definição incorreta de restrições do conteúdo de um elemento em relação ao conteúdo de outro elemento

3.1 Funcionalidades da *XTool*

A Figura 2 ilustra as principais funcionalidades da ferramenta *XTool* que são descritas a seguir. Na Figura 2 pode ser observado que o teste é iniciado pelo testador, que disponibiliza o esquema em teste e uma instância de dado associada a esse esquema para a ferramenta.

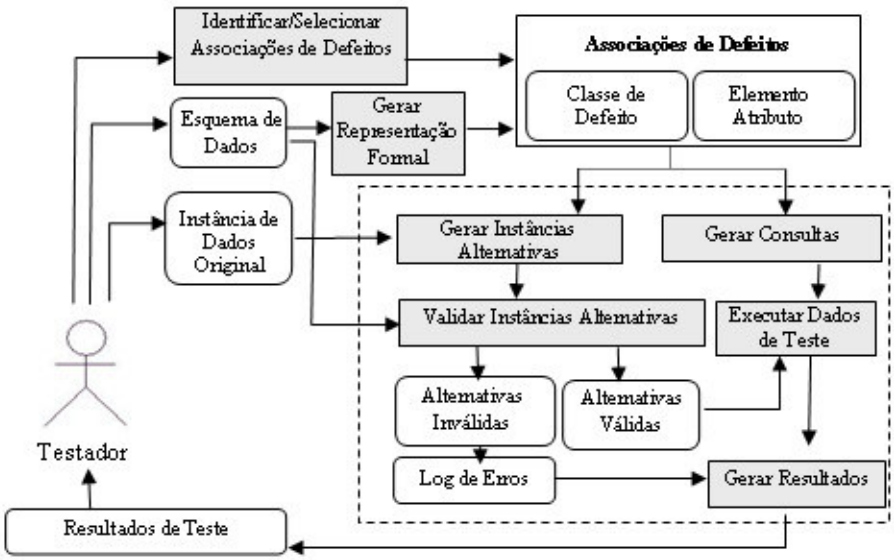

Figura 2. Funcionalidades da *XTool*

A ferramenta gera a representação formal para o esquema. Essa representação formal é utilizada para estabelecer automaticamente associações de defeitos presentes no esquema. Uma associação de defeito é um elemento (entidade) ou atributo e restrições aos dados do esquema associadas a uma classe de defeito. Após a geração das associações de defeitos, elas podem ser selecionadas pelo testador, que poderá também introduzir outras associações de defeitos, utilizando uma caixa de diálogo da ferramenta, se alguma outra associação puder ser estabelecida. Essas outras associações, inseridas pelo testador, podem representar restrições aos dados que podem estar ausentes no esquema, mas que o testador gostaria também de testar.

As associações de defeitos selecionadas norteiam a geração das instâncias de dados alternativas que podem ser documentos XML ou instâncias de base de dados relacional, de acordo com o esquema em teste. As instâncias de dados alternativas são geradas automaticamente por pequenas alterações nos dados da instância de dados original (disponibilizada pelo testador). As instâncias de dados alternativas representam os defeitos descritos pelas classes de defeitos. Essas instâncias passam por um processo de validação e são separadas em válidas e inválidas com relação às definições do esquema em teste.

As consultas também são geradas automaticamente, de acordo com as associações de defeitos selecionadas, e utilizam padrões de consulta previamente estabelecidos em uma linguagem de consulta adequada ao tipo de dado associado ao esquema em teste.

Na abordagem implementada pela *XTool*, os dados de teste são formados por uma instância de dados alternativa válida e por uma consulta que deve ser executada nessa instância de dados alternativa, de acordo com uma associação de defeito. O resultado dessa consulta deve ser avaliado pelo testador e pode indicar um defeito no esquema em teste.

As instâncias de dados alternativas inválidas também são resultados de teste e podem ser analisadas pelo testador. O testador verifica se os resultados obtidos com o teste estão de acordo com os resultados esperados, sempre que esses resultados diferem, um defeito foi revelado.

É interessante observar que o teste do esquema é realizado independentemente da aplicação que utiliza os esquemas, porém, é necessário que a especificação dos dados da aplicação esteja disponível para que os resultados de teste sejam analisados pelo testador. Nada impede que as instâncias de dados alternativas geradas possam também ser utilizadas em uma outra etapa como dados de teste para a aplicação.

3.2 Aspectos de Implementação

A *XTool* realiza atualmente o teste de esquemas XML escritos em XML *Schema* e o teste de esquemas de base de dados relacionais para o SGBD *PostGreSQL* (PostGre 2006).

As funcionalidades da *XTool* foram implementadas de acordo com o diagrama de classes apresentado na Figura 3. Essas classes são brevemente descritas a seguir:

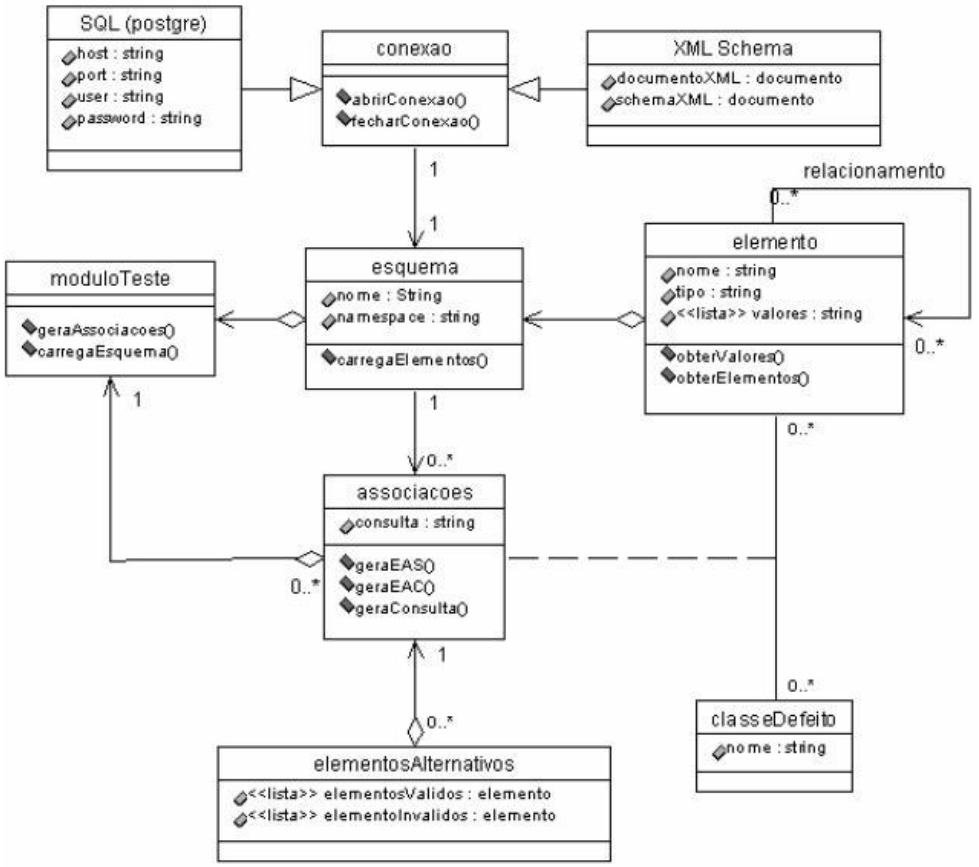

Figura 3. Diagrama de classes da XTool

- moduloTeste → classe principal, responsável pela interação entre a interface do usuário e as demais classes;
- esquema → classe responsável por armazenar o esquema em teste e algumas outras informações relevantes sobre o esquema;
- elemento →classe que representa os elementos de um esquema;
- classeDefeito → classe que representa as classes de defeitos previamente estabelecidas;
- associações → classe que representa as associações de defeitos;

- elementosAlternativos → classe responsável por armazenar as instâncias de dados alternativas;
- conexao → classe responsável por gerenciar a conexão com o esquema. Essa classe atualmente contém duas subclasses que permitem o teste de esquemas *PostGreSQL* e XML *Schema*. Para configurar a ferramenta para o teste de um outro tipo de esquema, uma nova sub-classe precisa ser criada.
    - conexao SQL (postgre) → subclasse da classe conexão, responsável pela conexão com uma base de dados *PostGreSQL*;
    - conexao XMLSchema → subclasse da classe conexão, responsável pelo acesso a documentos XML.

A *XTool* foi implementada na linguagem Java com o emprego do DOM (*Document Object Model*) (W3C 2006c) para manipular e validar documentos XML; Qexo (Bothner 2006) para consultar os documentos XML por meio da linguagem *XQuery* (W3C 2006d); JDBC (*Java Database Connectivity*) (SUN 2006) para manipular e consultar informações em bases de dados *PostGreSQL* (PostGre 2006) por meio das linguagens DQL (Linguagem de Consulta de Dados) e DML (Linguagem de Manipulação de Dados) que são subconjuntos da SQL (*Structured Query Language*).

4. Testando um Esquema com a *XTool*

Para ilustrar o procedimento de uso da ferramenta *XTool*, bem como as informações por ela produzidas, essa seção mostra como é realizado o teste de um esquema XML. Para isso, considere o fragmento do esquema de uma loja de cd's a ser testado e, o fragmento da instância de dados original (documento XML original, fornecido pelo usuário) apresentado na Figura 4. O esquema contém a definição dos dados referentes a cd's de música que podem ser armazenados em documentos XML associados a esse esquema.

```
<schema>
 <element
name="cds"><complexType><sequence>
 <element name="cd" maxOccurs="unbounded">
 <complexType> <sequence>
 <element name="titulo" type="string"
minOccurs="1"  maxOccurs="1" />
 <element name="artista" type="string"
minOccurs="1"  maxOccurs="unbounded" />
…
 <element name="ano" type="integer"
minOccurs="1"  maxOccurs="1"/>
…
 <attribute name="duracao" type="string"
use="optional"/>
 </complexType> </element>
 </sequence> </complexType> </element>
</schema>
```

(a) Exemplo 1: Fragmento do esquema XML para dados sobre cd's disponíveis em uma loja de música

```
<?xml version="1.0" ?>
<cds
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
  xsi:schemaLocation="cd.xsd">
  <cd genero="instrumental" duracao="00:46:32">
    <titulo>Sucessos do Cinema</titulo>
    <artista>Richard Clayderman</artista>
    <musica>I just called to say I love
      you</musica>
    <musica>I will always love you</musica>
    <musica>Unchained melody</musica>
    <musica>Theme from love story</musica>
    <musica>How deep is your love</musica>
    <gravadora>Polygram</gravadora>
    <ano>1996</ano>
  </cd>
...
</cds>
```

(b) Fragmento da instância de dados original

Figura 4: Fragmento de um esquema e de um documento XML associado ao esquema

A Figura 5 apresenta um fragmento da representação formal gerada pela *XTool* para descrever os elementos, atributos, restrições e regras de associação entre elementos, atributos e restrições do esquema em teste apresentado na Figura 4.

E = {cds, cd, titulo, artista, musica, gravadora, ano}
A = {genero, duracao}
R = {ordem, ocorrência, tipo, uso}
P = {p_1(cds, cd), p_2(cds, ordem, cd),
p_3(cd, titulo), p_4(cd, artista),
p_5(cd, musica), p_6(cd, gravadora),
p_7(cd, ano), p_8(cd, genero),
p_9(cd, duracao), p_{10}(cd, ordem, titulo,
artista, musica, gravadora, ano, genero,
duracao), p_{11}(cd, ocorrência),
p_{12}(titulo, ocorrência), p_{13}(titulo, tipo)
p_{14}(artista, ocorrência), ...
p_{20}(ano, tipo), ...
p_{24}(duração, uso), p_{25}(duracao, tipo)}

Figura 5: Fragmento de representação formal

No Exemplo 1 é ilustrada uma associação de defeito identificada na representação formal do esquema XML. Essa associação de defeito relaciona o elemento "ano" à classe de defeito "Tipo de Dado Incorreto" por meio da restrição "tipo" (*type*) definida para o elemento "ano".

Exemplo 1: Exemplo de associação de defeito identificada no esquema XML em teste
Elemento : ano
Classe de Defeito : Tipo de dado Incorreto
Restrição : tipo

O Exemplo 2 ilustra o conteúdo do elemento "ano" na instância de dados original associada ao esquema do Exemplo 1

Exemplo 2: Exemplo de conteúdo do elemento "ano"
<cds><cd> ... <ano>1996</ano>....</cd></cds>

Com base na associação de defeito do Exemplo 1 e das outras associações identificadas no esquema XML, as instâncias de dados alternativas são criadas por meio de uma modificação simples na instância de dados original (documento XML). Exemplos de alterações que podem ser feitas na instância de dados original de acordo com associação descrita anteriormente podem ser vistos no Exemplo 3. Cada uma dessas alterações gera uma instância de dados alternativa ou um documento XML alternativo.

Exemplo 3: Exemplos de possíveis alterações no conteúdo do elemento "ano"
<cds>...<ano>1996</ano>....</cds>
<cds>...<ano>abc1234</ano>....</cds>
<cds>...<ano>-1234</ano>....</cds>
<cds>...<ano>1234.56</ano>....</cds>

As instâncias de dados alternativas são validadas em relação ao esquema XML em teste e separadas em válidas e inválidas. No Exemplo 4, as alterações propostas no Exemplo 3 são classificadas em válidas ou inválidas.

Exemplo 4: Alterações válidas ou inválidas
Alterações válidas

```
<cds>...<ano>1996</ano>....</cds>
<cds>...<ano>-1234</ano>....</cds>
```

Alterações inválidas

```
<cds>...<ano>abc1234</ano>....</cds>
<cds>...<ano>1234.56</ano>....</cds>
```

As associações também guiam a geração de consultas em *XQuery*. Essas consultas e os documentos XML alternativos válidos formam dados de teste. Os dados de teste são executados e o resultado dessa execução é avaliado pelo testador de acordo com a especificação. Por exemplo, a associação de defeito do Exemplo 1 gera a consulta ilustrada no Exemplo 5. Essa consulta retorna o conteúdo de cada elemento "ano" do documento XML alternativo válido que está sendo consultado.

Exemplo 5: Consulta para a associação de defeito do Exemplo 1

```
let $doc := document("cd.xml")
for $i in $doc\\cds\cd\ano
return <resultado>{$i}</resultado>
```

O resultado obtido após a execução do dado de teste formado pela consulta do Exemplo 5 e pelos documentos XML alternativos válidos, gerados por meio da mesma associação de defeito que produziu essa consulta, é apresentado no Exemplo 6.

Exemplo 6: Resultado da consulta do Exemplo 5

```
<resultado>
  <ano>1996</ano>
  <ano>-1234</ano>
<resultado>
```

O resultado obtido é comparado com o resultado esperado. O conteúdo "-1234" obtido da consulta ao elemento "ano" é um valor numérico inválido para o conteúdo de um elemento que representa "ano". Dessa forma, pode ser concluído que o esquema XML sob teste permitiu que um documento XML com um dado incorreto fosse considerado válido. Portanto, o esquema possui um defeito na definição do elemento "ano". Esse defeito pode ser revelado porque é coberto pela classe de defeito "Restrições de Domínio - Tipo de Dado Incorreto".

5. Resultados de Estudos de Caso

A fim de validar a implementação da ferramenta *XTool* e a aplicabilidade da abordagem, foram efetuados estudos de caso para realizar o teste de esquemas XML e de esquema de base de dados relacional. O objetivo foi verificar o custo, em termos do número de dados de teste (consultas e associações) necessários, e a eficácia, em termos do número de defeitos revelados. Para realizar essa análise, os defeitos foram revelados durante o teste da aplicação em desenvolvimento, portanto os defeitos são naturais e não semeados.

Os resultados obtidos em ambos os estudos de caso foram bastante animadores e são resumidos a seguir. No contexto de XML, os esquemas utilizados definem dados de um sistema de matrícula e de um sistema de biblioteca. Esses esquemas contêm dados de estudantes ("aluno.xsd"); disciplinas disponíveis no curso ("disciplina.xsd"); usuários registrados na biblioteca ("usuario.xsd"); e títulos disponíveis na coleção da

biblioteca (“obras.xsd”). No contexto de base de dados relacional, o esquema define dados referentes ao histórico de alunos egressos de uma Universidade.

Na Tabela 2 são apresentadas características dos esquemas, como: número de elementos ou entidades e atributos; e número de registros armazenados na instância de dados original do esquema correspondente.

Tabela 2. Características dos esquemas

Tipo de esquema	Esquema	Elementos/Entidades	Atributos	Registros
XML	Aluno	10	0	5
	Disciplina	10	0	6
	Obras	11	0	7
	Usuário	8	0	6
Base de dados relacional	Universidade	19	73	126

A Tabela 3 apresenta os resultados de teste da *XTool* para cada esquema. Nessa tabela, o número de associações de defeitos identificadas e selecionadas; o número de instâncias alternativas geradas; o número de consultas produzidas pela *XTool*; e, o número de defeitos revelados no teste de cada esquema são apresentados. Nessa tabela pode ser observado, por exemplo, que a partir das 16 associações de defeitos identificadas para o esquema “aluno.xsd” foram revelados 4 defeitos no esquema.

Tabela 3. Resultados de teste da *XTool*

Tipo de esquema	Esquema	Associações de defeitos	Alternativas válidas/inválidas	Consultas geradas	Defeitos revelados
XML	Aluno	16	241/42	16	4
	Disciplina	16	283/43	16	4
	Obras	16	252/138	16	2
	Usuário	12	187/73	12	3
Base de dados relacional	universidade	297	942/1720	1240	27

Nestes estudos de caso, a classe de defeito mais eficaz nos esquemas XML foi a classe de defeito “Restrições de Domínio – Tipo de Dado Incorreto” e no esquema da base de dados relacional foi a classe de defeito “Restrições de Definição – Uso Incorreto”. Além dessas classes de defeitos, as classes “Restrições de Relacionamento – Ocorrência Incorreta”; “Restrições de Domínio – Tamanho Incorreto”; “Restrições de Domínio – Dígito Incorreto”; “Restrições de Domínio – Valores Enumerados Incorretos”; e “Restrições de Relacionamento – Associação Incorreta” também revelaram defeitos nos esquemas testados.

Os estudos de caso mostraram que a ferramenta apóia a aplicação da AIDA, auxiliando para a detecção de defeitos em esquemas XML e esquemas de base de dados relacional. Além disso, os resultados obtidos com os estudos de caso permitem afirmar que o maior custo da aplicação da abordagem com o apoio da *XTool* está na análise dos resultados de teste; dado que, essa análise é feita manualmente e envolve uma grande quantidade de dados devido ao número de instâncias de dados alternativas e de consultas geradas no processo de teste. No entanto, essa é uma limitação inerente à atividade de teste relativa à automatização do oráculo.

Além desses estudos de caso foram realizados outros dois, nos quais os defeitos foram semeados nos esquemas em teste, gerando esquemas mutantes. Um dos estudos

de caso utilizou a ferramenta *XTM* para gerar diferentes esquemas XML mutantes incorretos (Emer et al. 2007b). Esses esquemas foram testados com a *XTool* e todos tiveram seus defeitos revelados, mostrando que as classes de defeitos da *XTool* cobrem os defeitos descritos pelos operadores de mutação da *XTM*. O outro estudo de caso foi realizado por meio da inserção ad hoc de defeitos em esquemas de base de dados relacional gerando esquemas mutantes (Emer et al. 2007c). Esses esquemas mutantes foram testados pela *XTool* e todos os defeitos inseridos foram cobertos pelas classes de defeitos da *XTool* e, portanto, revelados. É importante salientar que os defeitos semeados foram considerados revelados com o uso da *XTool*, se os resultados de teste dos mutantes fossem diferentes dos resultados de teste do esquema original correspondente.

6. Conclusões e Trabalhos Futuros

Este artigo apresentou a ferramenta *XTool* que apóia a abordagem de teste denominada AIDA. Essa abordagem de teste é baseada em classes de defeitos que descrevem possíveis defeitos inseridos durante a definição ou evolução de um esquema de dados. Dessa forma, essa abordagem de teste pode revelar os defeitos cobertos pelas classes de defeitos e que estejam relacionados à definição incorreta ou ausente de restrições aos dados no esquema. Diferentemente das abordagens de teste implementadas pelas ferramentas encontradas na literatura (*RDBTool e XTM*), a abordagem de teste implementada é genérica, podendo ser aplicada em diferentes tipos de esquemas. Atualmente, a ferramenta *XTool* está configurada para o teste de dois tipos de esquemas: *XML Schema* e *PostGreSQL Schema*, utilizados respectivamente para definir dados em formato XML e dados de base de dados relacional.

A *XTool* foi empregada em estudos de caso, nos quais esquemas XML e um esquema de base de dados relacional foram testados. Esses estudos de caso mostraram que o uso da *XTool* é imprescindível para apoiar a aplicação da abordagem de teste AIDA, possibilitando a descoberta de defeitos, em esquemas de dados, cobertos pelas classes de defeito definidas. Os estudos de caso também permitiram a observação da necessidade de refinar a *XTool*, por exemplo, implementando um suporte ao oráculo para facilitar a comparação dos resultados obtidos com o teste com os resultados esperados, mesmo que não seja possível automatizar completamente essa tarefa. A grande vantagem da ferramenta comparada com as demais é que a tarefa de geração de dados de teste (consultas e instâncias de dados) é realizada automaticamente, sem a participação do usuário. Isso contribui para reduzir o esforço de teste. Além disso, o teste do esquema pode ser efetuado independentemente da disponibilidade da aplicação. As instâncias de dados alternativas geradas podem ser, em um segundo momento, utilizadas como dados de teste para o teste das aplicações. Isso deverá ser avaliado em trabalhos futuros.

A *XTool* pode ser estendida para testar esquemas em outros contextos, tais como: base de dados XML e serviços Web. Novos estudos de caso devem ser realizados para explorar outros esquemas e contextos. Os resultados desses novos estudos de caso poderão contribuir para o refinamento das classes de defeitos definidas na abordagem implementada.

Referências

Aranha, M. C. L. F. M.; Mendes, N.C.; Jino, M.; Toledo, C.M.T. (2000) "RDBTool: A Support Tool for Testing Relational Database". XI ICST, Int. Conference of Software Technology.

Bothner, Per. (2006) “Qexo: The GNU Kawa implementation of XQuery”, http://www.gnu.org/software/qexo/, 2005. Acessado em 2006.

Chan, M.; Cheung, S.. (1999) “Testing Database Applications with SQL Semantics”. In Proceedings of the 2nd International Symposium on Cooperative Database Systems for Advanced Applications, pp 364-375, March.

Chan, W. K.; Cheung, S. C.; Tse, T. H.. (2005) “Fault-Based Testing of Database Application Programs with Conceptual Data Model”. In Proceedings of the Fifth International Conference on Quality Software, pp 187-196.

Chays, D.; Deng, Yuetang. (2003) “Demonstration of AGENDA Tool Set for Testing Relational Database Applications”. In Proceedings of the 25th International Software Engineering Conference. IEEE Computer Society, pp 802 – 803. May.

Chays, David; Dan, Saikat; Frankl, Phyllis G.; Vokolos, Filippos I.; Weyuker, Elaine J.. (2000) “A Framework for Testing Database Applications”. In Proceedings of the ACM SIGSOFT International Symposium on Software Testing and Analysis, Vol. 25 Issue 5, August.

Chen, P. P.. (1976) “The Entity-Relationship Model – Toward a Unified View of Data”. ACM Transactions on Database Systems (TODS), Vol. 1, No 1, pp 9-36.

Deng, Yuetang; Frankl, Phyllis; CHAYS, David. (2005) “Testing Database Transactions with AGENDA”. In Proceedings of the 27th International Conference on Software Engineering. ACM Press, May.

Di Lucca, G.A. e Di Penta, M.. (2003) “Considering Browser Interaction in Web Application Testing”. In Proceedings of the 5th IEEE Intl. Workshop on Web Site Evolution. IEEE Computer Society Press.

Di Lucca, G.A.; Fasolino, A.R.; Faralli, F. e De Carlini, U.. (2002a) “Testing Web applications”. In Proceedings of the International Conference on Software Maintenance, pages 3–6. IEEE Press, October.

Di Lucca, G.A.; Fasolino, A.R.; Pace, F.; Tramontana, P. e De Carlini, U.. (2002b) “WARE: A tool for the Reverse Engineering of Web Applications”. In Proceedings of the VI Eur. Conf. on Software Maintenance and Reengineering. IEEE Computer Society Press, March.

Emer, M. C. F. P.; Vergilio, S. R.; Jino, M. (2007a) “A Fault-Based Testing of Data Schemas”. In Proc. of the 19th Intl. Conference on Software Engineering and Knowledge Engineering (SEKE 2007), July.

Emer, M. C. F. P.; Nazar, I. F.; Vergilio, S. R.; Jino, M. (2007b) “Evaluating a Fault-Based Testing Approach for XML Schemas”. VIII Latin-American Test Workshop. Cuzco, Peru, March.

Emer, M. C. F. P.; Vergilio, S. R.; Jino, M.; Nazar, I. F.; Caxeiro, P. V. (2007c) “Uma Avaliação do Teste Baseado em Defeitos em Esquemas de Banco de Dados Relacional”. In Proc. of the 1st Brazilian Workshop on Systematic and Automated Software Testing (SAST 2007) em conjunto com o XXI Simpósio Brasileiro de Engenharia de Software (SBES 2007), outubro.

Franzotte, L; Vergilio, S.R. (2006) “Applying Mutation Testing to XML Schemas”. In Proc. of the 18th Inl. Conference on Software Engineering and Knowledge Engineering (SEKE2006).

Kapfhammer, Gregory M.; Soffa, Mary Lou. (2003) “A Family of Test Adequacy Criteria for Database-driven Applications”. In Proceedings of the 9th European Software Engineering Conference held jointly with 11th ACM SIGSOFT International Symposium on Foundations of Software Engineering ESEC/FSE-11, Volume 28 Issue 5. ACM Press, September.

Kung, D.C.;. Liu, C.H e Hsia, P.. (2000) “An Object-Oriented Web Test Model for Testing Web Applications”. In Proceedings of the 24th Annual International Computer Software and Applications Conference, COMPSAC 2000, pages 537–542.

Lee, S.C. e Offutt, J.. (2001) “Generating test cases for XML-based web component interactions using mutation analysis”. In Proceedings of the 12th International Symposium on Software Reliability Engineering, pages 200–209. IEEE Press, November.

Li, J. B.; Miller, J. (2005) “Testing the Semantics of W3C XML Schema”. In Proc. of the 29th Annual Intl. Computer Software and Applications Conference (COMPSAC-2005), Julho.

Liu, C.H.; Kung, D.C. e Hsia, P.. (2000a) “Object-based Data Flow Testing of Web Applications”. In Proceedings of the 1st Asia-Pacific Conference on Quality Software, pages 7–16. IEEE Press.

Liu, C.H.; Kung, D.C.; Hsia, P. e Hsu, C.T.. (2000b) “Structural Testing of Web Applications”. In Proceedings of the 11th International Symposium on Software Reliability Engineering, pages 84–96. IEEE Press.

Offutt, J.; Xu, W. (2004) “Generating Test Cases for Web Services Using Data Perturbation”. In Proceedings of the TAV-WEB, volume 29. ACM SIGSOFT SEN, September.

OMG (2006). “Meta-Object Facility Core Specification Version 2.0”. http://www.omg.org/ cgi-bin/doc?formal/ 2006- 01-01, January 2006. (acessado em 2006).

PostGre (2006) “PostGreSQL”, http://www.postgresql.org/docs/, Acessado em 2006.

Ricca, F. e Tonella, P.. (2002) “Analysis and Testing of Web Applications”. In Proceedings of the 23rd International Conference on Software Maintenance, pages 25–34. IEEE Press, May.

Robbert, M. A.; Maryanski, F. J.. (1991) “Automated Test Plan Generator for Database Application Systems”. In Proceedings of the ACM SIGSAMLL/PC Symposium on Small Systems, pp 100-106.

Suárez-Cabal, M. J.; Tuya, J.. (2004) “Using a SQL Coverage Measurement for Testing Database Applications”. In Proceedings of the 12th International Symposium on the Foundations of Engineering. November.

SUN (2006) “Java Database Connectivity (JDBC)”, http://java.sun.com /javase/technologies/database/, Acessado em 2006.

W3C (2006a) “Extensible Markup Language (XML)”, http://www.w3c.org/XML/, Acessado em 2006.

W3C (2006b) “XML Schema”, http://www.w3c.org/XML/Schema/, Acessado em 2006.

W3C (2006c) “Document Object Model (DOM)”, http://www.w3c.org/DOM/, Acessado em 2006.

W3C (2006d) “XML Query Language. (XQuery)”, http://www.w3c.org/XML/Query/. Acessado em 2006.

Xu, W.; Offutt, J. e Luo, J.. (2005) “Testing Web Services by XML Pertubation”. In Proceedings of the 16th IEEE International Symposium on Software Reliability Engineering. IEEE.

Zhang, Jian; Xu, Chen; Cheung, S.-C.. (2001) “Automatic Generation of Database Instances for White-box Testing”. In Proceedings of the 25th Annual International Computer Software and Applications Conference, 2001, pp 161 – 165, October.