UNIVERSIDADE FEDERAL DO PARANÁ – UFPR
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
MESTRADO EM INFORMÁTICA

JOÃO CARLOS GARCIA ÁRIAS

TESTE DE APLICAÇÕES BASEADO EM ANÁLISE DE INSTÂNCIAS DE DADOS ALTERNATIVAS

Curitiba / 2011

JOÃO CARLOS GARCIA ÁRIAS

TESTE DE APLICAÇÕES BASEADO EM ANÁLISE DE INSTÂNCIAS DE DADOS ALTERNATIVAS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.

Orientadora: Prof.ª Dr.ª Silvia Regina Vergilio
Co-orientadora: Prof.ª Dr.ª Maria Cláudia Figueiredo Pereira Emer

Curitiba / 2011

Agradecimentos

Agradeço à professora Silvia Regina Vergilio, pela orientação, apoio e dedicação. Agradeço também à professora Maria Claudia Figueiredo Pereira Emer pelos ensinamentos e apoio.

Agradeço à equipe de trabalho da Celepar, particularmente ao Rafael Demetrio Benvenutti pelo incentivo, ao Marco Aurélio Cordeiro, Danillo Bazello e Debora Ruedell pelo apoio e disponibilização de flexibilidade no meu horário de trabalho, o que tornou possível a minha dedicação ao curso de mestrado.

Agradeço também a minha esposa Andreza, meus pais e demais familiares e amigos, pelo apoio, incentivo e paciência nos momentos de dificuldade.

Agradeço à Universidade Federal do Paraná por me dar a oportunidade de alcançar tamanho crescimento pessoal e profissional.

Sumário

Resumo

No desenvolvimento de software frequentemente é necessário validar a especificação dos dados do sistema, geralmente descrita por esquemas. O esquema de dados define a estrutura lógica e os relacionamentos entre os dados manipulados e armazenados por aplicações de software. Para auxiliar a detectar defeitos em esquemas e garantir a integridade dos dados por eles definidos, foi proposta uma abordagem de teste baseada em defeitos, denominada Análise de Instâncias de Dados Alternativas (AIDA). Na abordagem AIDA, uma instância de dados associada ao esquema em teste sofre alterações simples gerando instâncias de dados alternativas. A Análise de Instâncias de Dados Alternativas (AIDA) pode ser aplicada no teste de diferentes tipos de esquema, desde que o mesmo seja representado de acordo com um modelo formal. Apesar de existirem trabalhos na literatura que utilizam informações do esquema para testar as aplicações, a maioria não considera os possíveis defeitos presentes no esquema para gerar os dados de teste. Esses defeitos podem ocasionar falhas na aplicação e, considerando esse fato, este trabalho explora a utilização da AIDA no teste de aplicações que utilizam esquemas de dados, introduzindo estratégias de uso em dois contextos de aplicação: aplicações de banco de dados relacional e aplicações que utilizam Serviços Web. Para validar as estratégias em ambos contextos foram realizados experimentos em aplicações reais. Os resultados obtidos são analisados e verifica-se que a abordagem utilizada foi eficaz em revelar não somente defeitos de esquema, mas também defeitos relacionados à própria aplicação.

Palavras-chave: Teste de esquemas de dados, Teste baseado em defeitos, Teste de base de dados relacional, Serviço Web.

Abstract

In the proccess of software development, there is frequently a need to validate the application's data specification, which is usually described by schemas. Data schemas define the logical structure and the relationships among data that are handled and stored by applications. To support the activity of detecting faults in schemas and to assure the integrity of the data being manipulated by it, a fault based approach called Alternative Data Instance Analysis (ADIA) was proposed. In the ADIA approach, the data instance associated with the schema under test is slightly changed, generating alternative data instances. The ADIA approach can be applied to test several kinds of data schemas, if the schema can be represented according to a formal model. Even though works that make use of information from the schema to test the application exist in the literature, most of these works do not take under consideration possible faults that are present in the schema to generate the test data. These schema faults may generate failures on the application, and considering this fact, this work explores the use of the ADIA approach to test applications that use data schemas, introducing practical strategies in two application contexts: relational database applications and applications that use Web Services. To validate these strategies in the two contexts explored, experiments were conducted in real applications. The results are then analysed, and we conclude that the approach is efficacious to reveal faults related to the schema as well as related to applications.

Key-words: Data schema testing, Fault based testing, Relational database testing, Web Services.

Lista de Figuras

Lista de Tabelas

1 INTRODUÇÃO

O teste de software é a atividade de investigação que busca fornecer informações sobre a qualidade de um software em relação ao contexto em que ele deve operar, incluindo os processos de análise e de utilização do produto para encontrar seus defeitos [46]. A atividade de teste de software é de fundamental importância, pois a não detecção de defeitos no produto pode gerar a construção de um software de forma diferente do que foi especificado. Um componente fundamental na atividade de teste de software é o caso de teste, que descreve uma condição particular a ser testada, e é composto por valores de entrada, restrições para a sua execução e um resultado ou comportamento esperado.

Para gerar dados de teste com alta probabilidade de revelar defeitos, técnicas de teste de software são utilizadas: a técnica funcional, a técnica estrutural e a técnica baseada em defeitos [46]. Na técnica funcional, não há preocupação com os detalhes de implementação do software, observando-se apenas se os dados de entrada são apropriadamente aceitos e se a resposta a esses dados, ou saída, é a esperada. A técnica estrutural foca na análise da implementação (procedimentos e dados) de um software, onde os caminhos lógicos são geralmente testados por casos de teste que exercitam conjuntos específicos de condições, laços e caminhos associados a um grafo de fluxo de controle do software. A técnica baseada em defeitos procura revelar a presença de defeitos que comumente ocorrem no desenvolvimento de software, produzindo um conjunto de casos de teste que diferenciem o software original de suas alternativas, que são geradas por modificações no software original de acordo com os defeitos previamente conhecidos. A abordagem de teste baseada em defeitos depende da definição de classes de defeitos que comumente ocorrem no desenvolvimento de software.

Durante o desenvolvimento de software, frequentemente é necessário validar a especificação dos dados do sistema, ou seja, a informação que o software manipula. Os dados manipulados e armazenados por aplicações são geralmente descritos por esquemas. O esquema de dados define a estrutura lógica e os relacionamentos entre os dados. Alguns exemplos de esquemas de dados são: esquema XML [62], esquema de banco de dados relacional geralmente descritos por Diagramas Entidade-Relacionamento (DER) [52] etc. O teste de esquema por meio de abordagens, critérios e ferramentas de teste é uma forma de assegurar a qualidade dos dados manipulados por uma aplicação de software. Se o esquema estiver incorreto, ou seja, se alguma definição referente aos dados contiver algum defeito em relação à especificação dos dados, informações inválidas podem

ser aceitas pela aplicação, podendo causar falhas, ou ainda, dados válidos podem não ser aceitos, também gerando resultados inesperados no processamento da aplicação.

O teste de um esquema compreende a validação das definições dos dados que serão manipulados pela aplicação, buscando-se verificar se o esquema foi implementado de acordo com a sua especificação. Alguns trabalhos são encontrados na literatura para o teste de esquemas de dados [22, 24, 25, 30, 42]. O teste de aplicação compreende a validação das regras de negócio, validação e apresentação implementadas a nível de aplicação, em relação às definições contidas na especificação do sistema. As diferentes técnicas de teste mencionadas anteriormente têm sido utilizadas para o teste de aplicação e, também, algumas estratégias de teste têm sido propostas para o teste da aplicação utilizando informações do esquema. Por exemplo, no contexto de banco de dados relacional, destacam-se os trabalhos de Árias et al. [4], Tongrak et al. [57] e Zhang et al. [64]. No contexto de aplicações Web, destacam-se os trabalhos de Bertolino et al. [6], Dong et al. [20], Solino [53] e Tsai et al. [58].

Para auxiliar a detectar defeitos em esquemas e para garantir a integridade dos dados por eles definidos, foi proposta uma abordagem de teste baseada em defeitos, denominada Análise de Instâncias de Dados Alternativas (AIDA) [25]. Na abordagem AIDA, uma instância de dados associada ao esquema em teste sofre alterações simples gerando instâncias de dados alternativas. Cada alternativa representa possíveis defeitos que podem estar presentes no esquema, e é gerada a partir de classes de defeitos previamente definidas a partir de um modelo genérico capaz de representar os mais variados tipos de esquema de dados. Consultas às instâncias geradas são executadas e o resultado destas consultas, se diferente do resultado esperado, aponta que existem defeitos no esquema, pois a instância consultada foi considerada válida para esse esquema.

A abordagem AIDA pode ser aplicada no teste de diferentes tipos de esquema, desde que o mesmo seja representado de acordo com o modelo formal. Algumas ferramentas de suporte estão disponíveis para AIDA. A ferramenta XTool (*XML and Relational Database Schema Testing Tool*) [41] que permite o teste de dois tipos de esquemas: esquemas XML escritos em XML Schema [62], e esquemas de banco de dados relacional para o SGBD *PostGreSQL* [45]. Pode-se também aplicar a abordagem AIDA no contexto de Serviços Web (Web Services – WS), com o auxílio da ferramenta WSeTT (Web Service Testing Tool) [53], que adapta a abordagem ao contexto de documentos WSDL (Web Service Description Language) [59], refinando as classes de defeitos para documentos XML. Utilizando estas ferramentas, a abordagem AIDA foi avaliada no teste de diversos tipos de esquema [21, 22] e os resultados mostram sua eficácia em revelar defeitos.

Entretanto, observa-se que as instâncias de dados alternativas geradas pela abordagem e que descrevem defeitos nos esquemas são entradas para as aplicações que utilizam o esquema em teste, e podem ser também utilizadas como dados de teste para estas aplicações. Como mencionado, na literatura existem alguns trabalhos que utilizam os esquemas de dados para gerar dados de teste para as aplicações, entretanto nenhum destes trabalhos utiliza dados que descrevem defeitos nos esquemas. Estes defeitos podem estar refletidos na aplicação e os dados gerados com a AIDA podem ser utilizados de maneira complementar para revelar outros tipos de defeitos além daqueles revelados pelas abordagens existentes.

1.1 OBJETIVOS

O último parágrafo da seção anterior descreve a principal motivação para o presente trabalho, que tem como objetivo explorar o uso da abordagem AIDA no teste de aplicações que utilizam esquemas de dados. O trabalho propõe estratégias de utilização da AIDA em dois contextos: o teste de aplicações de banco de dados relacional e o teste de aplicações de Serviços Web. Para avaliar as estratégias propostas foram conduzidos experimentos de validação com aplicações reais.

No contexto de banco de dados relacional, o experimento foi conduzido para testar três aplicações de banco de dados reais. A ferramenta XTool foi utilizada para aplicar a abordagem de teste AIDA nos esquemas das aplicações de banco de dados. As bases de dados alternativas geradas pela XTool foram utilizadas como dados de teste. Os resultados deste experimento são apresentados resumidamente e já foram publicados [4]. Eles mostram que as instâncias de dados alternativas são capazes de revelar defeitos na aplicação além de defeitos nos esquemas.

No contexto de aplicações que utilizam Serviço Web, o experimento foi conduzido para testar três aplicações reais que utilizam Serviço Web. Neste contexto, o esquema no formato XML utilizado é o documento WSDL referente a cada Serviço Web. Com o auxílio da ferramenta WSeTT realizaram-se consultas ao documento WSDL e instâncias de dados alternativas de documentos WSDL foram criadas. A partir dessas instâncias, mensagens SOAP alternativas são geradas e enviadas para o teste do Serviço Web, e para a geração de dados de teste para as aplicações clientes, seguindo a mesma metodologia do primeiro experimento.

Com a condução dos experimentos foi possível validar o uso das estratégias propostas. Elas foram capazes de revelar todos os defeitos de esquema e a maioria dos defeitos presentes nas aplicações. Além disso, observou-se que a proposta contribuiu para o processo de teste existente na empresa que forneceu as aplicações para teste.

1.2 ORGANIZAÇÃO DO TRABALHO

Este trabalho está organizado da seguinte forma: o Capítulo 2 apresenta algumas definições sobre teste de software, descrevendo técnicas de teste, e trabalhos relacionados sobre teste de esquemas, teste de aplicações de banco de dados relacional e teste de aplicações de Serviços Web. O Capítulo 3 descreve a abordagem AIDA, e apresenta seus contextos de aplicação e ferramentas de apoio. O Capítulo 4 introduz uma estratégia de uso da AIDA para o teste de aplicações que utilizam banco de dados relacionais. O Capítulo 5 apresenta uma estratégia de uso da AIDA para o teste de aplicações que utilizam Serviços Web. O Capítulo 6 apresenta a análise final dos resultados, contribuições do trabalho e possíveis desdobramentos para o mesmo. O trabalho também contém um apêndice (Apêndice A) que apresenta tabelas que detalham os defeitos revelados no teste de aplicações de Serviços Web, segundo a estratégia apresentada no Capítulo 5.

2 TESTE DE SOFTWARE

O processo de desenvolvimento de software envolve uma série de atividades. No entanto, apesar das técnicas, métodos e ferramentas empregados estarem em um contínuo processo de evolução, este processo está ainda sujeito a erros. Atividades agregadas sob o nome de Garantia de Qualidade de Software têm sido introduzidas ao longo de todo o processo de desenvolvimento, entre elas atividades de verificação e validação, que têm por objetivo minimizar a ocorrência de erros inerentes ao processo de desenvolvimento de software. Dentre as técnicas de verificação e validação, a atividade de teste é uma das mais utilizadas, constituindo-se em uma forma de fornecer evidências da confiabilidade do software em complemento a outras atividades. A atividade de teste consiste de uma análise dinâmica do produto em desenvolvimento e é uma atividade relevante para a identificação e eliminação de erros que persistem [46].

O teste de software envolve basicamente quatro etapas: planejamento, projeto de casos de teste, execução e avaliação dos resultados. Essas atividades devem ser desenvolvidas ao longo do processo de desenvolvimento do software e, geralmente, consistem em três fases: teste de unidade, teste de integração e teste de sistema [46]. O teste de unidade foca na menor unidade do projeto de software e procura identificar erros de lógica e de implementação isolados em cada módulo. O teste de integração é feito durante a integração dos módulos que compõem a estrutura do programa, visando descobrir erros associados às interfaces entre os módulos, buscando validar se após a integração dos módulos a estrutura obtida está de acordo com o que foi determinado no projeto. O teste de sistema é realizado após a integração dos módulos que compõem o sistema, e procura identificar erros de funções e características de desempenho que não estejam de acordo com a especificação.

Um ponto muito importante na atividade de teste, independentemente da fase, é o projeto e a avaliação da qualidade de um determinado conjunto de casos de teste T utilizado para o teste de um programa P, visto que geralmente é impraticável utilizar todo o domínio de dados de entrada para avaliar os aspectos funcionais e operacionais de um programa em teste. Um caso de teste é composto por uma entrada para P (dado de teste) e da saída esperada para a execução da entrada dada. O objetivo é, portanto, utilizar casos de teste que tenham alta probabilidade de revelar a maioria dos defeitos com um mínimo de tempo e esforço. Para selecionar estes “bons” casos de

teste, alguns critérios a serem satisfeitos foram propostos [46].

O teste bem sucedido é aquele que consegue determinar casos de teste para os quais o programa em teste falhe. Na prática observa-se que a própria atividade de projeto de casos de teste é bastante efetiva em evidenciar a presença de defeitos de software.

2.1 TÉCNICAS DE TESTE

Em geral, os critérios de teste de software são estabelecidos a partir de três técnicas: a funcional, a estrutural e a baseada em defeitos [46]. Na técnica funcional, os critérios e requisitos de teste são estabelecidos a partir das funções do software estabelecidas na sua especificação. Na técnica estrutural, os critérios e requisitos são derivados essencialmente a partir das características da estrutura do software. Na técnica baseada em defeitos, os critérios e requisitos de teste são oriundos do conhecimento sobre erros típicos cometidos no processo de desenvolvimento de software.

É importante observar que as técnicas de teste de software são consideradas complementares [46], sendo utilizadas em diversos contextos, no teste de: modelos, orientação a objetos, orientação a aspectos, componentes e em outros tipos de software em variados domínios de aplicação. Dentre estes domínios destacam-se dois tipos de teste: o teste do esquemas de dados, e o teste da aplicação que manipula o esquema de dados. Ambos são de particular interesse para o presente trabalho.

O esquema de dados (esquema XML, esquema de banco de dados, esquema de Serviço Web etc) define a estrutura lógica e os relacionamentos entre os dados manipulados e armazenados por aplicações. O teste de esquema é uma forma de assegurar a qualidade dos dados manipulados por uma aplicação de software. Se o esquema estiver incorreto, informações inválidas podem ser aceitas pela aplicação, ou dados válidos podem não ser aceitos, gerando resultados inesperados no processamento da aplicação.

O teste da aplicação que manipula o esquema de dados também é de extrema relevância, pois é fundamental avaliar se a forma como a aplicação manipula os dados está de acordo com sua especificação, e também se defeitos contidos no esquema de dados podem estar refletidos na aplicação.

Nas próximas seções apresenta-se uma visão geral dos trabalhos que aplicam estas técnicas nos contextos relacionados ao trabalho descrito nesta dissertação. São dois contextos: o contexto de aplicações Web e o de banco de dados relacional.

2.2 TESTE NO CONTEXTO DE APLICAÇÕES WEB

O nome Web vem do termo mais amplo World Wide Web (WWW), que representa um sistema de informações mundial distribuído, onde as informações são "ligadas" umas às outras por *links* (ligações) de hipertexto. Desenvolvimentos mais recentes deste tecnologia têm permitido distribuir, junto com as informações, objetos e programas que realizam atividades junto ao cliente consumidor da informação. Pode-se definir uma aplicação Web como uma aplicação de software que utiliza a Web como ambiente de execução, permitindo o acesso à informação e produtos através da Internet.

Uma aplicação Web é heterogênea, ou seja, utiliza diferentes componentes de software e diversas tecnologias. Dentre tais tecnologias, destaca-se a XML (eXtensible Markup Language) [26] que é uma recomendação da W3C para gerar linguagens de marcação, capaz de descrever diversos tipos de dados, tendo como propósito principal a facilidade de compartilhamento de informações através da Internet. Outro componente muito utilizado em aplicações Web é o Serviço Web, que é uma solução utilizada na integração de sistemas e na comunicação entre aplicações diferentes, tornando possível que novas aplicações possam interagir com aquelas que já existem, e que sistemas desenvolvidos em plataformas diferentes sejam compatíveis.

As aplicações Web são empregadas em muitas atividades e precisam de garantia de qualidade e confiabilidade, o que pode ser obtido por meio de metodologias e ferramentas de teste. Existem alguns trabalhos que têm adaptado ou modificado abordagens de teste de software tradicionais para o teste de aplicações Web [13, 14, 15, 34, 38, 39, 49]. Outros trabalhos focam somente no teste de alguns aspectos da aplicação Web como, por exemplo, a interação entre componentes Web por meio de mensagens XML [6, 29, 35]. No contexto deste trabalho, o teste de Serviços Web é de particular interesse e será abordado a seguir.

2.2.1 Teste de Serviços Web

Bloomberg [7] expõe alguns aspectos que envolvem o teste de Serviços Web, entre eles: teste do mecanismo de requisição e resposta do Serviço Web através de mensagens SOAP; utilização das informações contidas nos documentos WSDL para o planejamento dos casos de teste; teste da publicação, busca e associação de Serviço Web, que se trata de uma nova característica sob a perspectiva de teste de software; emulação do cliente de um Serviço Web, assim como emulação de um Serviço Web para um cliente.

Os trabalhos existentes visam focalizar alguns destes aspectos. Tsai et al [58] propõem a utilização de um *framework* chamado Coyote para converter especificações WSDL em cenários de teste para Serviço Web. Guangquan, Mei e Jun [31] desenvolveram um processo de negócio para definir um método para o teste de Serviços Web, baseado no diagrama de atividades definido na UML 2.0. Dong [20] propõe um método para a geração automática de casos de teste para Serviços Web. A partir do documento WSDL, são gerados casos de teste de duas perspectivas: dados de teste e operações de teste. Os dados de teste são gerados analisando-se os tipos de dados das mensagens e as operações de teste são geradas analisando-se dependências de operações.

O teste de Serviços Web baseado em perturbação de mensagens SOAP tem sido investigado por muitos trabalhos na literatura, a maioria com teste baseado em defeitos [2, 18, 44]. Operadores de perturbação modificam mensagens SOAP que serão enviadas como dados de teste para os Serviços Web. Dentre estes trabalhos, o único que perturba as mensagens SOAP a partir de informações de um esquema escrito em XML é o de Offut e Xu [44], porém a perturbação não considera defeitos que podem estar presentes no WSDL.

2.2.2 Teste de Esquemas XML

Um esquema XML (XML Schema) é uma linguagem, baseada no formato XML, para definição de regras de validação em documentos no formato XML.

Características de um esquema XML:

- define elementos que podem aparecer em um documento.
- define atributos que podem aparecer em um documento.
- define que elementos são elementos filhos.
- define a ordem dos elementos filhos.
- define o número de elementos filhos.
- define se um elemento é vazio ou pode incluir texto.
- define tipos de dados para elementos e atributos.
- define valores padrão e fixos para elementos e atributos.

O teste de esquemas XML é de fundamental importância em aplicações que manipulam dados descritos no formato XML.

Li e Miller [37] propõem operadores de mutação para esquemas XML. Esses operadores fazem alterações simples no esquema, alterando um valor ou um atributo. Os autores mostram que muitos defeitos descritos pelos operadores de mutação não são detectados pelos analisadores comumente usados para validar documentos XML *Schema*. Entretanto, eles não apresentam um experimento no qual os operadores de mutação são aplicados no processo de teste de um esquema XML e não mencionam como os operadores devem ser usados para gerar casos de teste para a execução do teste de esquema.

Franzotte e Vergilio [27] apresentam uma aplicação da análise de mutantes para o teste de esquemas XML. Genevès et al. [28] desenvolveram um sistema para analisar defeitos em esquemas XML em evolução, através da análise de possíveis incompatibilidades em suas *queries*.

A abordagem AIDA pode ser utilizada para testar esquemas XML, sendo que alguns trabalhos já foram conduzidos com este objetivo [21, 23]. Para auxiliar a aplicação da abordagem, foi desenvolvida a ferramenta XTool [41].

2.3 TESTE NO CONTEXTO DE BANCO DE DADOS RELACIONAL

O teste de aplicações de banco de dados relacional avalia aspectos relacionados a:

conformidade da aplicação em relação a sua especificação; ao esquema do banco de dados em relação aos dados reais modelados; a acurácia no banco de dados, segurança e privacidade; e a execução correta de operações de inserção, atualização e exclusão de dados. Nesse contexto, existem vários estudos sendo conduzidos para o teste de aplicações de banco de dados relacional. Esses estudos envolvem a geração de dados, teste do projeto e da aplicação de banco de dados [9, 10, 11, 12, 33, 36, 54, 56, 64].

2.3.1 Teste em aplicações de banco de dados utilizando informações do esquema

No contexto de banco de dados relacional muitos trabalhos abordam técnicas que utilizam informações do esquema de dados para testar o banco de dados.

Robbert e Maryanski [50] usam informações obtidas do esquema relacional de banco de dados para gerar um plano de teste para a aplicação de banco de dados.

Chan et al. [9] propõem uma abordagem baseada em defeitos para testar sentenças SQL de uma aplicação de banco de dados, usando informações capturadas do modelo de dados conceitual para gerar sentenças SQL mutantes.

Chays et al. [10] desenvolveram um *framework* para auxiliar o teste de banco de dados relacional, que analisa o estado do banco de dados antes e depois das operações efetuadas pelo usuário e, também, as respostas obtidas a partir dos dados de entrada do usuário. Neste trabalho foi também desenvolvida uma ferramenta para popular a base de dados com dados que satisfaçam as restrições definidas, e sua aplicação em um sistema de grande porte é avaliada.

Chays e Deng [11] desenvolveram uma ferramenta, a AGENDA, para auxiliar o teste de banco de dados relacional, que popula a base de dados, gera dados de entrada e valida aspectos da saída produzida e do novo estado da base de dados.

Grolinger e Capretz [30] propõem uma abordagem de teste unitário para aplicações que utilizam banco de dados, com o objetivo de avaliar o código da aplicação a partir das alterações feitas no banco de dados durante o seu ciclo normal de evolução.

2.3.2 Teste do esquema de dados associado ao banco de dados relacional

No contexto de banco de dados relacional, alguns trabalhos investigam o teste do esquema associado à base de dados.

Aranha et al. [3] desenvolveram uma ferramenta de apoio ao teste de bancos de dados relacionais, a RDBTool, que utiliza como critério de teste a definição e o uso de atributos.

Tongrak e Suwannasart [57] apresentam uma ferramenta para testar as restrições definidas para um banco de dados, automatizando a geração de casos de teste de acordo com o critério definido pelo testador, que compreende o esquema do banco de dados, as restrições de integridade das entidades, as restrições de integridade referencial e as restrições de domínio.

A abordagem AIDA também pode ser utilizada para testar esquemas de dados associados a banco de dados relacional, sendo que alguns trabalhos já foram conduzidos com este objetivo [22, 24]. Para auxiliar a aplicação da abordagem, foi desenvolvida a ferramenta XTool [41].

2.4 CONSIDERAÇÕES FINAIS

Observa-se que alguns dos trabalhos citados procuram utilizar o esquema para gerar dados de teste. No entanto, em muitas dessas abordagens não se considera o teste baseado em defeitos, ou seja, não são avaliados defeitos provenientes do esquema, que podem estar refletidos na aplicação que utiliza o esquema.

Considerando que a abordagem AIDA pode ser aplicada em diferentes contextos, neste trabalho propõe-se que ela seja utilizada no teste de aplicações de banco de dados e de Serviços Web. Sendo assim, no próximo capítulo a abordagem AIDA é descrita em detalhes. Nos Capítulos 4 e 5 são apresentadas estratégias que permitem o uso da AIDA para o teste de aplicações de banco de dados relacional e de Serviços Web, respectivamente.

3 ANÁLISE DE INSTÂNCIAS DE DADOS ALTERNATIVAS

Como mencionado no capítulo anterior, o esquema de dados de uma aplicação de software contém a definição dos dados por ela manipulados. Se o esquema estiver incorreto, ou seja, se alguma definição referente aos dados contiver algum defeito em relação à especificação dos dados, informações inválidas podem ser aceitas pela aplicação, podendo causar falhas, ou ainda, dados válidos podem não ser aceitos, também gerando resultados inesperados no processamento da aplicação.

A Análise de Instâncias de Dados Alternativas (AIDA) [25] é uma abordagem de teste baseada em defeito que auxilia o teste de esquemas de dados, buscando garantir a integridade dos dados por eles definidos e que são manipulados na aplicação. Um diagrama de como funciona a abordagem AIDA é apresentado na Figura 3.1.

Na AIDA, defeitos que são comumente introduzidos no esquema durante o seu desenvolvimento são identificados e organizados em classes de defeitos. As classes de defeito direcionam a geração de instâncias de dados alternativas e, também, consultas ao esquema em teste. As instâncias de dados representam os possíveis defeitos existentes no esquema, de acordo com as classes de defeito definidas. Através das consultas é possível revelar esses defeitos no esquema que está sendo testado. Tanto instâncias quanto consultas são geradas a partir de padrões definidos para cada classe de defeito. Estes defeitos estão relacionados a definições incorretas, ou ausentes, de restrições que deveriam ter sido definidas para os dados no esquema.

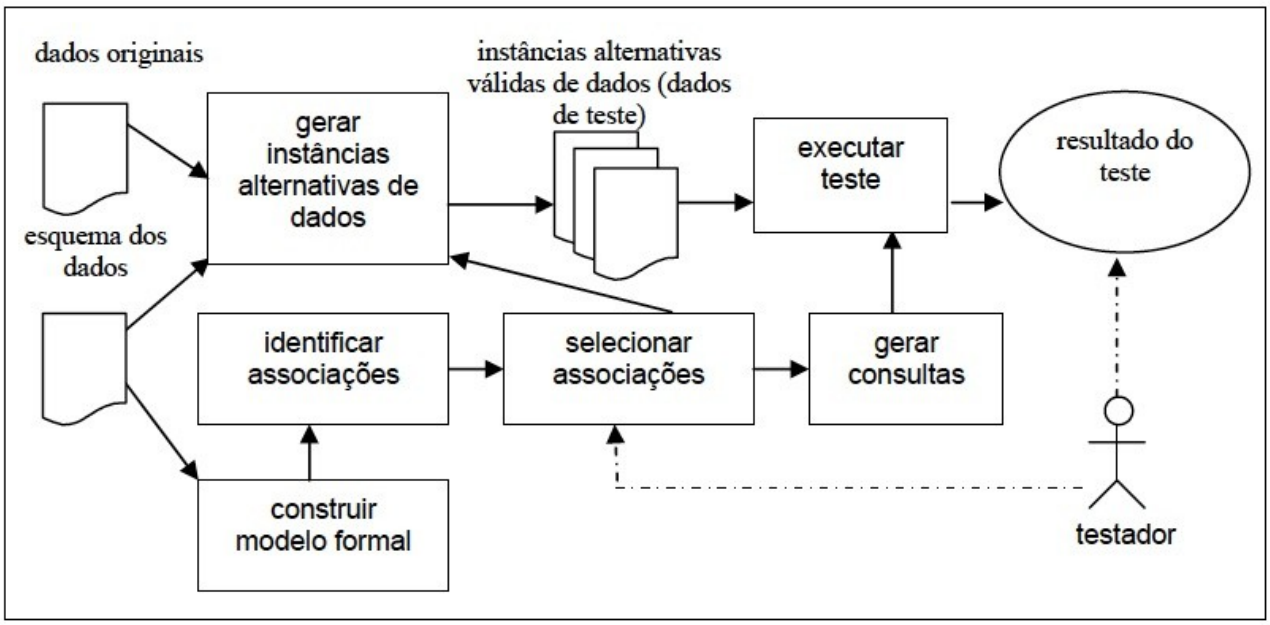

Figura 3.1 - Processo de Teste da Abordagem AIDA (extraída de [25])

3.1 DESCRIÇÃO DA ABORDAGEM

A abordagem AIDA e todos os seus passos são descritos nas subseções a seguir.

3.1.1 Construção do Modelo Formal

Na abordagem AIDA o modelo de dados é definido por um metamodelo *MM*, apresentado na Figura 3.2.

O metamodelo *MM* consiste das classes: Elemento (entidades ou objetos), Atributo (propriedades dos elementos), Restrição (restrições associadas aos elementos e atributos) e, da classe associativa: Associação (propriedades da associação reflexiva da classe Elemento). Um elemento possui atributos e está associado a outros elementos; um elemento ou um atributo possui zero ou mais restrições; uma associação entre elementos é definida pelo seu tipo.

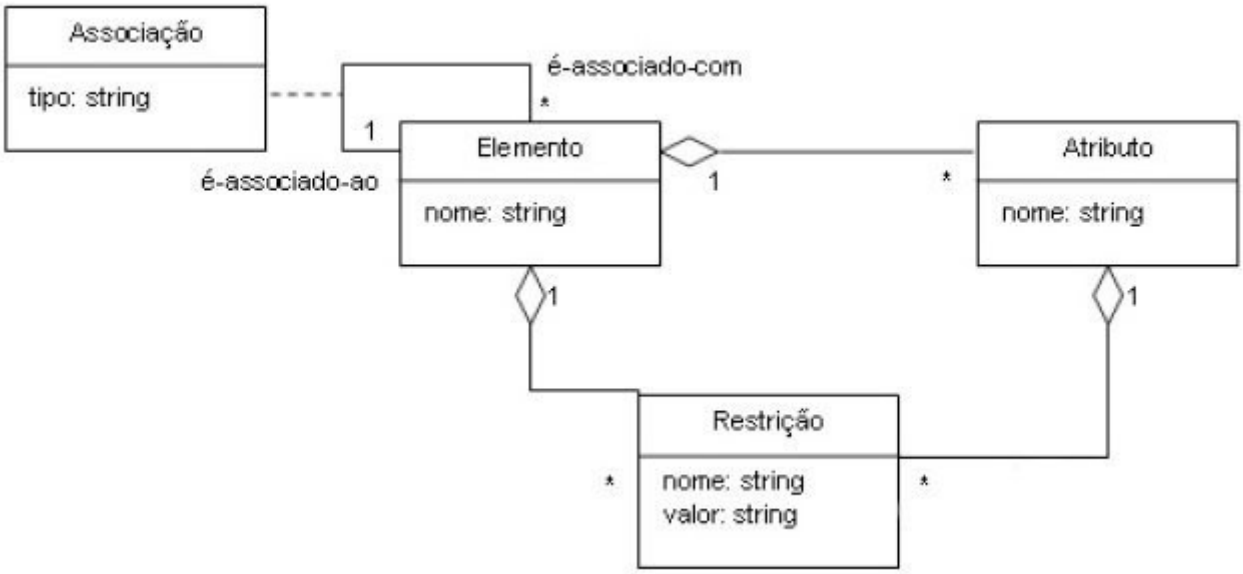

Figura 3.2 - Metamodelo MM (extraída de [25])

As restrições identificadas no metamodelo *MM* para esquemas de dados, definidas usando *OCL* (*Object Constraint Language*) [43], são apresentadas na Tabela 3.1 de forma intuitiva.

Com base no metamodelo *MM*, modelos de dados representando domínios específicos são definidos, de modo que o metamodelo *MM* permite que sejam instanciados e interpretados os componentes do modelo de dados.

A representação formal do modelo de dados de um esquema de dados define os elementos, atributos, restrições e relacionamentos entre eles. No contexto de teste, a representação formal é necessária para que sejam geradas as instâncias de dados e as consultas a essas instâncias.

Considerando-se um esquema de dados *S*, sua representação segundo um modelo de dados *M* é dada por $S = (E, A, R, P)$, onde:

- *E* é um conjunto finito de elementos (ou entidades);
- *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.

Considerando $U = E \cup A$. As regras de associações são representadas por :

- $p(x, y) \mid x, y \in E \wedge x \neq y;$

- $p(x,r) \mid x \in E \wedge r \in R$;
- $p(x,r,SU) \mid x \in U \wedge r \in R \wedge SU = \{u_1, u_2, ..., u_m\} \subset U, \ \forall \ u_i \neq x, 1 \leq i \leq m, m \geq 1,$ onde m é o número de elementos e atributos de SU.

Tabela 3.1 – Restrições identificadas no metamodelo MM (extraído de [25])

Restrição	**Definição Intuitiva**
Tipo	Tipos de dados primitivos (*string*, inteiro, real, booleano, data, hora) e tipos de dados definidos pelo usuário, a partir dos tipos primitivos, que podem ser atribuídos aos elementos e atributos.
Valor	Elemento ou atributo definido com um valor padrão ou fixo definido pelo usuário.
Enumeração	Lista de valores enumerados que podem ser atribuídos aos elementos ou atributos.
Limite inferior ou limite superior	Limites superior e inferior atribuídos aos valores numéricos de elementos ou atributos.
Tamanho, tamanho máximo ou mínimo	Quantidade de caracteres permitida para um valor do tipo *string* de um elemento ou atributo.
Dígito ou dígito decimal	Quantidade de dígitos ou dígitos decimais permitida para um valor numérico de um elemento ou atributo.
Seqüência	Seqüência de caracteres ou números permitidos para o conteúdo de um elemento ou atributo.
Espaço	Tratamento dado aos caracteres de espaço no conteúdo de um elemento ou atributo.
Uso	Um atributo pode ser definido como opcional ou obrigatório.
Único	O conteúdo de um atributo pode ser definido como único ou não.
Identificador	Um atributo pode ser definido como identificador.
Ocorrência máxima ou mínima	Número de vezes, máximo ou mínimo, que um determinado elemento pode ocorrer.
Ordem	Ordem que elementos-filhos de um determinado elemento devem seguir.
Associação	Dois ou mais elementos podem estar relacionados por uma associação do tipo cardinalidade, generalização/especialização, agregação e elemento associativo.
Condição	Uma condição semântica é um predicado que deve ser satisfeito pelo conteúdo de determinado atributo ou elemento.

A Figura 3.3 apresenta um exemplo de uso da representação formal para descrever um modelo de dados segundo o metamodelo *MM*.

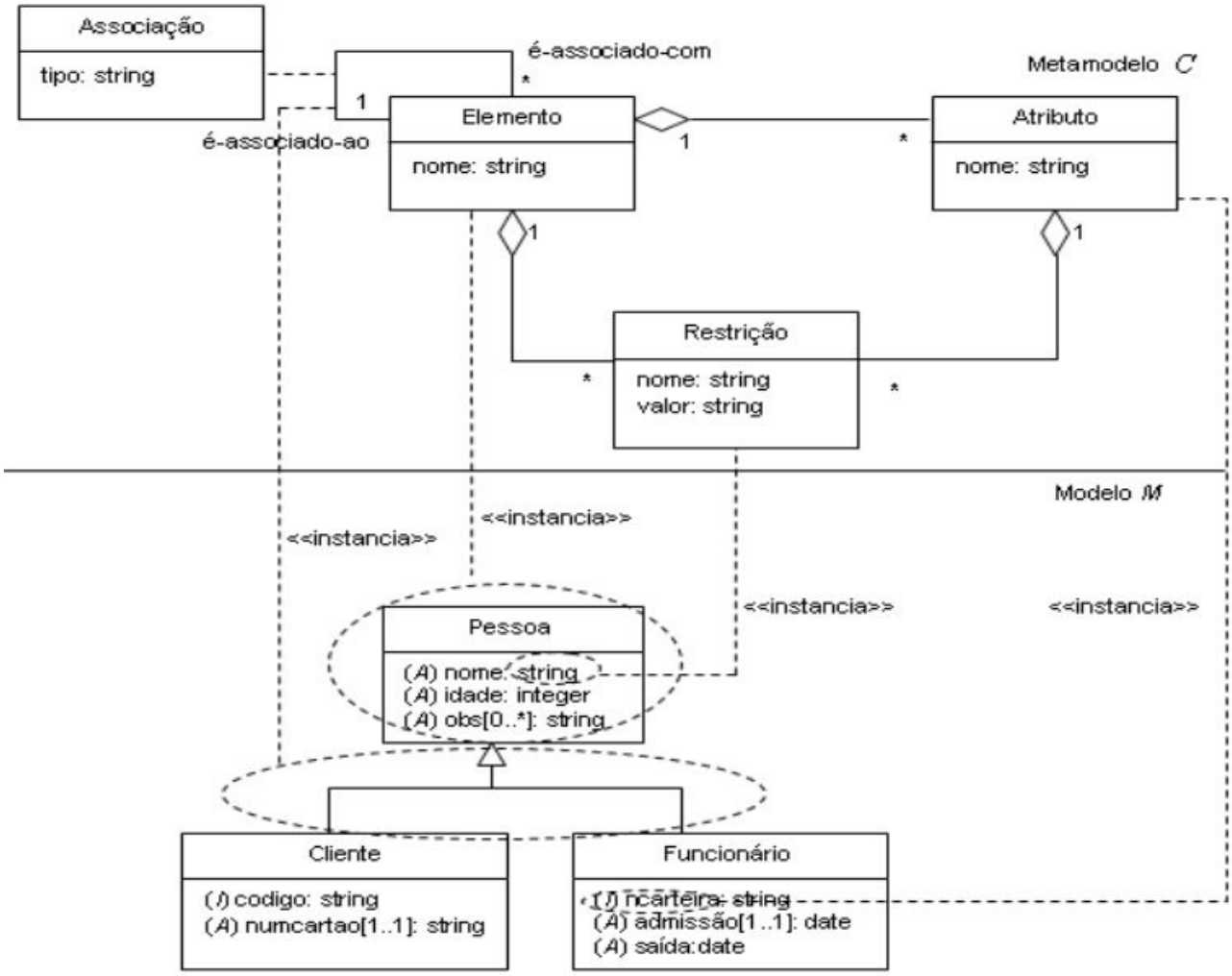

Figura 3.3 - Exemplo de um modelo de dados M descrito pelo metamodelo MM (extraído de [25])

O modelo de dados *M* ilustrado na Figura 3.3 representa dados de clientes e funcionários da base de dados de uma empresa. Para exemplificar que os componentes do modelo *M* são instâncias do metamodelo *MM*, deve-se observar que: as classes *Cliente* e *Funcionário* são especializações da classe *Pessoa*, a associação generalização/especialização é uma instância da associação entre elementos prevista no *MM*; a classe *Pessoa* é uma instância da classe Elemento; o tipo de dado *string* do atributo nome da classe *Pessoa* é uma instância da classe Restrição e o atributo *ncarteira* da classe *Funcionário* é uma instância da classe *Atributo* no *MM*.

Assim, a representação do modelo *M* , *S = (E, A, R, P)*, é apresentada na Figura 3.4.

E = {Pessoa, Cliente, Funcionario}
A = {nome, idade, obs, codigo, numcartao, ncarteira, admissao, saida}
R = {tipo, relacionamento, ocorrência, identificador, uso}
P = {Pessoa(nome, idade, obs, relacionamento : Cliente Funcionario),
nome(tipo),
idade(tipo),
obs(tipo, ocorrência),
Cliente(codigo, numcartao, relacionamento : Pessoa),
codigo(tipo, identificador),
numcartao(tipo, uso),
Funcionario(ncarteira, admissao, saida, relacionamento : Pessoa),
ncarteira(tipo, identificador),
admissao(tipo, uso),
saida(tipo)}

Figura 3.4 – Representação formal do modelo de dados M

3.1.2 Classes de Defeitos

As classes de defeito na abordagem AIDA direcionam a detecção de defeitos em esquemas. Essas classes foram definidas através da análise de definições de esquemas de dados existentes e de acordo com as restrições para esquemas de dados identificadas no metamodelo *MM*, definido por Emer [25]. As classes de defeito representam defeitos típicos, relacionados a definições incorretas ou ausentes de restrições aos dados, que podem ter sido introduzidos durante o projeto de um esquema ou em algum processo de atualização ou de manutenção do mesmo. Os defeitos identificados no metamodelo *MM* são representados em um modelo de dados, que especifica um esquema de dados específico para uma aplicação de software. As classes de defeito são instanciadas para um esquema de dados, de acordo com as restrições aos dados previstas para o esquema e para a aplicação correspondente. Os defeitos são classificados em quatro grupos de restrições: domínio, definição, relacionamento e semântica. Os grupos e as respectivas classes de defeitos são apresentados na Tabela 3.2.

3.1.3 Associações de defeito

Uma associação de defeito é formada pela associação de um elemento (entidade), ou atributo, com restrições aos dados associadas a uma classe de defeito.

As associações de defeitos estão relacionadas a padrões de alteração, que são aplicados a uma instância original de dados, que quando modificada gera uma instância de dados alternativa.

As associações de defeito também estão associadas a padrões de consultas que serão executadas nas instâncias alternativas e cujos resultados poderão indicar um defeito no esquema.

Como exemplo de uma associação de defeito, podemos citar a associação entre um atributo definido com o tipo de dado Inteiro e a classe de defeito *Tipo de Dado Incorreto*, que orienta a geração de uma instância de dados alternativa com um tipo diferente de Inteiro, que pode revelar um defeito de *Tipo de Dado Incorreto*.

Tabela 3.2 – Classes de defeito da abordagem AIDA (extraída de [25])

Identificador	Classe de defeito	Descrição
Grupo 1 – Restrições de domínio		
TDI	Tipo de Dado Incorreto	Definição incorreta do tipo de dado
VI	Valor Incorreto	Valor padrão ou fixo definido incorretamente
VEI	Valores Enumerados Incorretos	Definição incorreta do conjunto de valores permitidos
VMMI	Valores Máximos e Mínimos Incorretos	Definição incorreta de valores limites
TI	Tamanho Incorreto	Número de caracteres máximos e mínimos permitidos definidos incorretamente
DI	Dígito Incorreto	Número de dígitos máximos e mínimos permitidos definidos incorretamente
SVI	Seqüência de valores incorretos	Definição incorreta da seqüência de valores permitidos.
CEBI	Caracteres de Espaço em Branco Incorretos	Definição incorreta de tipos especiais de espaços em branco permitidos
Grupo 2 - Restrições de definição		
UI	Uso Incorreto	Definição incorreta de uso (opcional ou obrigatório)
UNI	Unicidade Incorreta	Definição incorreta do número de ocorrências permitidas
II	Identificador Incorreto	Definição incorreta de um identificador
Grupo 3 - Restrições de relacionamento		
OI	Ocorrência Incorreta	Número máximo e mínimo de repetições definido incorretamente
ORI	Ordem Incorreta	Definição incorreta de ordem
AI	Associação Incorreta	Definição incorreta de cardinalidade (número de ocorrências de um elemento em relação a outro elemento em um relacionamento), generalização/especialização (um ou mais elementos herdam propriedades de um outro elemento), agregação (um elemento é parte ou está contido em outro elemento), elemento associativo (relacionamento que possui atributos e sua existência depende da existência do relacionamento)
Grupo 4 - Restrições semânticas		
COI	Condição Incorreta	Definição incorreta de uma condição, que deve ser satisfeita pelos dados

3.1.4 Instâncias de Dados Alternativas

As instâncias de dados alternativas são geradas através de modificações simples na instância de dados original, sendo que tais alterações devem ser validadas para o esquema em teste. As alterações são feitas pela inclusão, modificação e exclusão de elementos ou atributos da instância original, de acordo com os padrões definidos para cada classe de defeito. Cada alteração gera uma instância de dados diferente. As alterações são orientadas pelo conjunto de associações de defeito identificadas no esquema em teste e, desta forma, as instâncias representam os possíveis defeitos que podem ser revelados no esquema que está sendo testado. As instâncias geradas são validadas com relação às definições do esquema e são, então, separadas em válidas e inválidas.

3.1.5 Consultas

O conjunto de consultas é gerado utilizando-se uma linguagem de consulta específica para o modelo de dados do esquema que está sendo testado. Cada classe de defeito direciona um padrão de consulta, definindo o resultado que deve ser retornado pela consulta para que o defeito possa ser revelado. As consultas padrão são instanciadas para cada instância de dados alternativa válida, de acordo com as associações de defeito identificadas no esquema em teste. O resultado da consulta é geral, o defeito é revelado pela análise do testador que compara o resultado obtido com o esperado.

3.1.6 Casos de Teste

Um caso de teste é formado pelo dado de teste e o resultado esperado. Os dados de teste para o esquema de dados em teste são compostos por uma consulta a um determinado elemento ou atributo, e uma instância de dados alternativa válida. Os defeitos no esquema em teste são revelados por meio da análise dos resultados obtidos nas consultas às instâncias de dados alternativas, comparando-os com a especificação dos resultados esperados para o dado de teste, de acordo com a especificação dos dados do esquema ou da aplicação. Além dos resultados das consultas, o testador também pode analisar as instâncias de dados alternativas inválidas geradas. Isso pode auxiliar na

detecção de defeitos, porque dados que deveriam ser válidos se o esquema estivesse correto, conforme a especificação dos dados, podem ter sido considerados inválidos para o esquema em teste, ou dados válidos podem ter sido considerados inválidos.

3.2 CRITÉRIOS DE TESTE BASEADOS EM ASSOCIAÇÕES DE DEFEITO

Uma associação de defeito é constituída por elementos ou atributos de um esquema, associados a uma classe de defeito. Com base nas associações de defeito presentes no esquema de dados e nas classes de defeitos identificadas, definem-se os critérios de teste da abordagem AIDA. Tais critérios exigem o exercício das associações de defeito através da execução de consultas às instâncias de dados alternativas válidas geradas por elas.

Considerando *Z* um elemento ou um atributo, definem-se os critérios:

• Todas Restrições - exige que todas as associações de defeito com relação a *Z* relacionadas às classes de defeitos dos grupos de restrições de domínio, definição, relacionamento e semântica sejam exercitadas, executando cada consulta às instâncias de dados alternativas relacionadas a essas associações.

• Todas Restrições de Domínio - requer que todas as associações de defeito com relação a *Z* relacionadas às classes de defeitos do grupo de restrições de domínio sejam exercitadas, exigindo que todas as consultas às instâncias de dados alternativas relacionadas a essas associações sejam executadas.

• Todas Restrições de Definição - requer que todas as associações de defeito com relação a *Z* relacionadas às classes de defeitos do grupo de restrições de definição sejam exercitadas, exigindo que todas as consultas às instâncias de dados alternativas relacionadas a essas associações sejam exercitadas.

• Todas Restrições de Relacionamento - requer que todas as associações de defeito com relação a *Z* relacionadas às classes de defeitos do grupo de restrições de relacionamento sejam exercitadas,

exigindo que todas as consultas às instâncias de dados alternativas relacionadas a essas associações sejam exercitadas.

• Todas Restrições Semânticas - requer que todas as associações de defeito com relação a *Z* relacionadas à classe de defeito do grupo de restrições semânticas sejam exercitadas, exigindo que todas as consultas às instâncias de dados alternativas relacionadas a essas associações sejam executadas.

• Todas Restrições/CDS - exige que pelo menos uma associação de defeito com relação a *Z* relacionada às classes de defeitos hierarquicamente superiores seja exercitada, caso a associação exista, executando a consulta às instâncias de dados alternativas relacionadas a essa associação.

• Todos Grupos de Restrições - exige que pelo menos uma associação de defeito com relação a *Z* relacionada a cada grupo de restrições de domínio, definição, relacionamento e semântica seja exercitada, caso a associação exista, executando a consulta às instâncias de dados alternativas relacionadas a essa associação.

A partir da análise do padrão de alterações e de consultas estabelecidos para as classes de defeitos definidas na abordagem AIDA, conclui-se que existe uma relação entre as classes de defeito identificadas. Essas relações permitem que se estabeleça uma hierarquia entre as classes de defeitos, que possibilite a redução do custo de aplicação da abordagem, sem que haja perda de eficácia na tarefa de revelar defeitos. Uma classe de defeito X é dita hierarquicamente superior (CDS) a uma classe de defeito Y com respeito a um elemento ou atributo x se ambas as condições forem atendidas: o conjunto de instâncias de dados alternativas gerado por alterações na classe X, possui instâncias alternativas que representam os mesmos defeitos representados por instâncias alternativas geradas através de alterações especificadas na classe Y; as consultas geradas a partir da classe X, têm a mesma probabilidade de revelar os mesmos defeitos revelados pelas consultas geradas a partir da classe Y. Esta relação também é verificada entre os critérios derivados a partir das classes.

3.3 PROCESSO DE TESTE

O processo de teste da AIDA é realizado através de uma sequência de passos:

1) O esquema em teste *S* e a instância de dados original *I0*, válida para *S*, são disponibilizados pelo testador;
2) O modelo formal *M* que representa *S* é construído;
3) A partir do modelo *M*, identifica-se o conjunto de associações de defeito;
4) O testador seleciona as associações que serão exercitadas no teste segundo um critério de teste pré-estabelecido;
5) A instância de dados original *I0* válida para o esquema *S* é usada para gerar as instâncias de dados alternativas $(I_1, I_2, ..., I_i, ..., I_n)$, sofrendo alterações em seus elementos ou atributos;
6) As instâncias de dados alternativas geradas $(I_1, I_2, ..., I_i, ..., I_n)$, são separadas em válidas e inválidas para o esquema em teste;
7) As consultas *Q* são geradas para cada instância de dados alternativa válida *Ii*, para as associações selecionadas;
8) Os dados de teste *(q, I)* são executados. q é a consulta, tal que $q \in Q$, e *I* é uma instância de dados alternativa válida para o esquema em teste, tal que $I \in I0$;
9) O testador compara os resultados obtidos com a execução das consultas com os resultados esperados, para verificar se foi revelado algum defeito no esquema para o conjunto de dados de teste executado;
10) Como resultado do processo, defeitos no esquema podem ser revelados, apontando correções necessárias no esquema de dados que está sendo testado.

3.4 CONTEXTOS DE USO DA AIDA

A abordagem AIDA pode ser aplicada no teste de diferentes tipos de esquema desde que estes sejam representados formalmente de acordo com o modelo *M* descrito pelo metamodelo *MM*. A abordagem foi explorada para o teste de esquemas XML [21, 23], no teste de esquemas de banco de dados relacional [22, 24] e no teste de esquemas WSDL no contexto de Serviços Web [53].

Nestes cenários, pode-se validar o esquema implementado, com relação a sua especificação, comparando-se os resultados esperados com os resultados obtidos por meio das consultas.

A seguir são apresentados exemplos de uso da AIDA nestes contextos mencionados: no teste de esquemas XML, esquemas relacionais, e esquemas WSDL no contexto de serviços Web.

3.4.1 Teste de Esquemas XML

A estratégia utilizada pela AIDA para o teste de esquemas XML é descrita a seguir.

Considere o fragmento do esquema de um sistema de uma loja de CDs a ser testado e o fragmento da instância de dados original (documento XML original) apresentado na Figura 3.5. O esquema contém a definição dos dados referentes a CDs 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>
```

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>
```

Fragmento da instância de dados original

Figura 3.5 - Fragmento de um esquema e de um documento XML associado ao esquema (extraída de [41])

O modelo formal construído é apresentado na Figura 3.6.

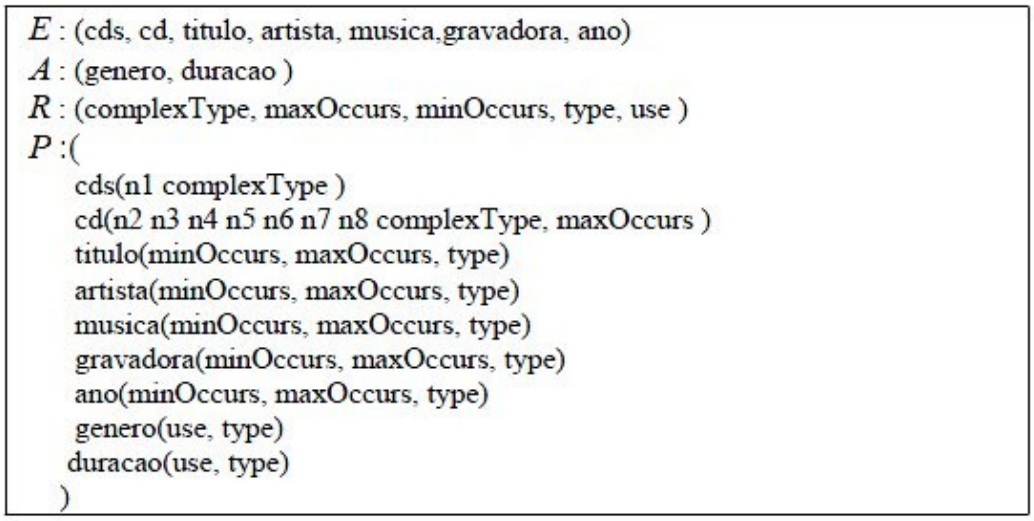

Figura 3.6 – Modelo Formal do esquema de uma loja de CDs (extraída de [41])

Para o esquema XML apresentado na Figura 3.5, uma possível associação de defeito (TDI, ano) relaciona o elemento "ano" à classe de defeito *Tipo de Dado Incorreto* (Tabela 3.2) por meio da restrição "Tipo" (ver Tabela 3.1) definida para o elemento "ano".

Considere a instância de dado original (documento XML):

<cds>...<ano>1996</ano>....</cds>

Com base na associação de defeito, as instâncias de dados alternativas são criadas por meio de uma modificação simples na instância de dados original. Alguns exemplos de alterações que podem ser feitas na instância de dados original de acordo com a associação de defeito são:

<cds>...<ano>abc1234</ano>....</cds>
<cds>...<ano>-1234</ano>....</cds>
<cds>...<ano>1234.56</ano>....</cds>

A associação de defeito direciona a geração de consultas em *XQuery* [61]. Essas consultas e os documentos XML alternativos válidos formam os dados de teste. Os dados de teste são executados e o resultado dessa execução é avaliado pelo testador de acordo com a especificação. Um exemplo de uma consulta gerada pela associação de defeito é:

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

Essa consulta retorna o conteúdo de cada elemento “ano” da instância alternativa (documento XML alternativo) válida que está sendo consultada. O resultado obtido após a execução do dado de teste formado por esta consulta e pelos documentos XML alternativos válidos, gerados por meio da mesma associação de defeito que produziu essa consulta, é:

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

O conteúdo “-1234” obtido através da consulta ao elemento “ano” é um valor numérico inválido para o elemento que representa “ano”. Assim, pode-se concluir que o esquema XML em teste permitiu que um documento XML com um dado incorreto fosse considerado válido. Portanto, conclui-se que o esquema possui um defeito na definição do tipo do elemento “ano”.

Assim, outras associações de defeitos são possíveis de serem estabelecidas no esquema em teste, a partir das quais outras instâncias alternativas e consultas são geradas e poderão também revelar outros defeitos.

3.4.2 Teste de Esquemas de Banco de Dados Relacional

Conforme mencionado anteriormente, a AIDA pode ser aplicada no contexto de teste de esquemas de bases relacionais. Um exemplo deste uso é descrito a seguir.

Considere o esquema definido pelo DER (Diagrama Entidade-Relacionamento) da Figura 3.7, e sua correspondente DDL (*Definition Description Language*) apresentada na Figura 3.8.

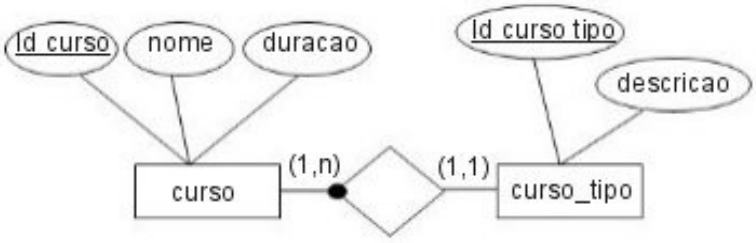

Figura 3.7 – Diagrama Entidade-Relacionamento (adaptado de [22])

Considerando as classes de defeitos apresentadas na Tabela 3.2 e o esquema definido na Figura 3.8, e que na especificação do esquema foi prevista uma restrição de tamanho para o atributo *name*, uma possível associação de defeito é a relação entre o atributo *name* e a classe de defeito TI (Tamanho Incorreto), indicando que o elemento *name* pode conter um defeito de definição incorreta de tamanho.

```
CREATE TABLE curso (
        id_curso        int IDENTITY,
        nome            varchar(40) NOT NULL,
        id_curso_tipo   int NOT NULL,
        duracao         int NOT NULL
);

ALTER TABLE curso
        ADD PRIMARY KEY NONCLUSTERED (id_curso);
```

Figura 3.8 - Exemplo de representação de um esquema de dados (adaptado de [22])

O modelo formal construído é apresentado na Figura 3.9.

S = (E, A, R, P)

E = {curso, curso_tipo}

A = {id_curso, nome, duracao, id_curso_tipo, descricao}

R = {tipo, chave, uso, tamanho, unicidade, associação}

P = { P1(curso, id_curso), P2(curso, nome), P3(curso, duracao),
P4(curso, associacao, curso_tipo), P5(curso, chave, curso_tipo),
P6(id_curso, tipo), P7(id_curso, chave), P8(nome, tipo), P9(nome, uso),
P10(nome, tamanho), P11(duracao, tipo), P12(duracao, uso),
P13(curso_tipo, id_curso_tipo), P14(curso_tipo, descricao),
P15(curso_tipo, associacao, curso), P16(id_curso_tipo, tipo),
P17(id_curso_tipo, chave), P18(descricao, tipo), P19(descricao, uso),
P20(descricao, tamanho), P21(descricao, unicidade) }

Figura 3.9 – Modelo Formal (adaptado de [22])

Considere a seguinte parte da definição de uma base de dados, referente a atribuição de um valor considerado válido para o atributo *nome*:

nome = "Ciência da Computação"

Um exemplo de instância alternativa para a instância de dados original seria por exemplo trocar o valor do atributo *nome* para "Ciência da Computação xxxxxxxxxxxxxxxxxxxxxxxxx". Um exemplo de consulta em SQL [55] a ser realizada nesta instância de dados alternativa é:

select nome from curso;

Esta consulta produzirá como resultado um elemento cujo número de caracteres é maior que o número máximo constante em sua especificação que, quando analisado, indicará que o esquema possui um defeito, pois validou um nome com um tamanho muito superior ao desejado, de acordo com a especificação do esquema.

3.4.3 Teste de Documentos WSDL

A AIDA pode também ser utilizada para o teste de serviços Web, considerando o documento WSDL [48] como um esquema que descreve o serviço, o qual poderá conter defeitos devido a uma especificação incorreta de seu esquema WSDL.

O objetivo, portanto, é o teste do documento WSDL, que por sua vez é um documento XML que está sujeito a defeitos tais quais os descritos na Tabela 3.2, e para o qual diversas instâncias alternativas podem ser geradas e consultadas.

Portanto, para a aplicação da AIDA no teste de Serviços Web são realizadas consultas aos documentos WSDL alternativos utilizando a linguagem de consulta XQuery. O retorno das consultas é verificado para ver se o mesmo coincide com a especificação do serviço. Desta forma, espera-se revelar os defeitos sem a necessidade de ter que gerar os respectivos clientes do serviço Web, reduzindo-se o custo operacional da atividade de teste. As consultas XQuery são apresentadas na Tabela 3.3 que, além do nome e descrição, apresenta em qual operador ou classe de defeito a consulta foi baseada.

Para ilustrar o procedimento para realizar os testes considera-se um exemplo que consiste em verificar se um cliente é o dono de determinada conta. Para este exemplo, o Serviço Web deve retornar *true* para os valores apresentados na Tabela 3.4 e *false* para todas as outras combinações.

A Figura 3.10 mostra o trecho de um arquivo WSDL que apresenta a ordem dos parâmetros do serviço *verificaCliente*.

```
1  <message name="verificaClienteMessage">
2      <part name="idCliente" nillable="true" type="xs:int" />
3      <part name="idConta" nillable="true" type="xs:int" />
4  </message>
```

Figura 3.10 – Trecho de um arquivo WSDL

Tabela 3.3 – Consultas XQuery (adaptada de 53])

Operador	**Descrição**	**Operador/Classe de defeito**
consultaAssinaturaServiço	consulta os elementos modificados pelo operador de mutação mudaAssinaturaServiço.	operador mudaAssinaturaServiço
consultaTipoAssintaturaServiço	retorna os tipos dos parâmetros definidos na assinatura dos serviços.	operador mudaTipoAssinaturaServiço
consultaTipoElementoMensagem	retorna os tipos das mensagens definidos no Web Service.	operador mudaTipoElementoMensagem
consultaInputOutput	retorna a mensagem de input e output de uma dada operação.	operador mudaInputOutput
Incorrect Order	retorna a ordem definida nos elementos de tipo complexo (complexType).	Classe de defeito Ordem Incorreta (AIDA)
Incorrect Value	busca por valores default e fixos (fixed) de um elemento	Classe de defeito Valor Incorreto (AIDA)
Incorrect Occurrence	busca o número mínimo e máximo de ocorrências de um elemento.	Classe de defeito Ocorrência Incorreta (AIDA)
Incorrect Use	consulta pelo tipo de uso de um atributo.	Classe de defeito Uso Incorreto (AIDA)
Incorrect Value (atributos)	consulta se um atributo possui um valor default ou fixed.	Classe de defeito Valor Incorreto (AIDA)
Incorrect Data Types	consulta pelos tipos de dados mais comuns no XML esquema	Classe de defeito Tipo de Dado Incorreto (AIDA)
Incorrect Enumerated Values	consulta os valores aceitáveis para um elemento.	Classe de defeito Valores Enumerados Incorretos (AIDA)
Incorrect Maximum and Minimum Values	consulta os valores máximo e mínimo de valores numéricos.	Classe de defeito Valores Máximos e Mínimos Incorretos (AIDA)
Incorrect Pattern	consulta o padrão definido para um elemento.	Classe de defeito Sequência de Valores Incorreta (AIDA)
Incorrect White Spaces Characters	consulta como o processador XML deve processar os espaços em branco.	Classe de defeito Caracteres de Espaço em Branco Incorretos (AIDA)
Incorrect Length	consulta a restrição de tamanho de um elemento.	Classe de defeito Tamanho Incorreto (AIDA)
Incorrect Digits	consulta a quantidade total de dígitos que um tipo numérico pode conter.	Classe de defeito Dígito Incorreto (AIDA)

Tabela 3.4 – Valores de retorno do Serviço Web (extraído de [53])

idCliente	**idConta**	**Retorno do WS**
1	2	true
1	3	true
2	1	true

Como pode ser observado na Figura 3.10, o documento WSDL apresenta os parâmetros idCliente e idConta ordenados. A consulta XQuery *Assinatura Serviço* irá retornar a ordem especificada no WSDL. Com o resultado da consulta, o testador deve comparar e verificar se o documento WSDL está descrito de acordo com a especificação. Caso não esteja, um defeito no documento WSDL é apontado.

Suponha que o programador tenha invertido a ordem nos parâmetros no método que implementa o WS. Ao invés da assinatura *boolean verificaCliente(int idCliente, int idConta)* teríamos a seguinte assinatura: *boolean verificaCliente(int idConta, int idCliente)*. Dessa forma, quando o testador configurar uma mensagem a ser enviada com os valores *idCliente* igual a 1 e *idConta* igual a 3, que segundo a Tabela 3.5 deveria retornar *true*, irá retornar o valor *false*.

Avaliando a consulta XQuery realizada previamente o testador consegue verificar que o documento WSDL está ou não de acordo com a especificação.

3.5 FERRAMENTAS DE APOIO

Para implementar e avaliar em aplicações reais os resultados obtidos pela abordagem AIDA, foram desenvolvidas em outros trabalhos duas ferramentas de apoio: a XTool [41] e a WSeTT [53], descritas nesta seção.

3.5.1 A Ferramenta XTool

A ferramenta XTool foi desenvolvida utilizando a linguagem Java para testar esquemas XML, utilizando o modelo DOM (Document Object Model) [19] para processar o esquema XML, manipular e validar os documentos XML alternativos. A XTool também pode ser usada no teste de esquemas de banco de dados PostGreSQL [45]. A ferramenta XTool implementa todos os passos descritos no processo de teste da abordagem AIDA. A Figura 3.11 apresenta a arquitetura da XTool.

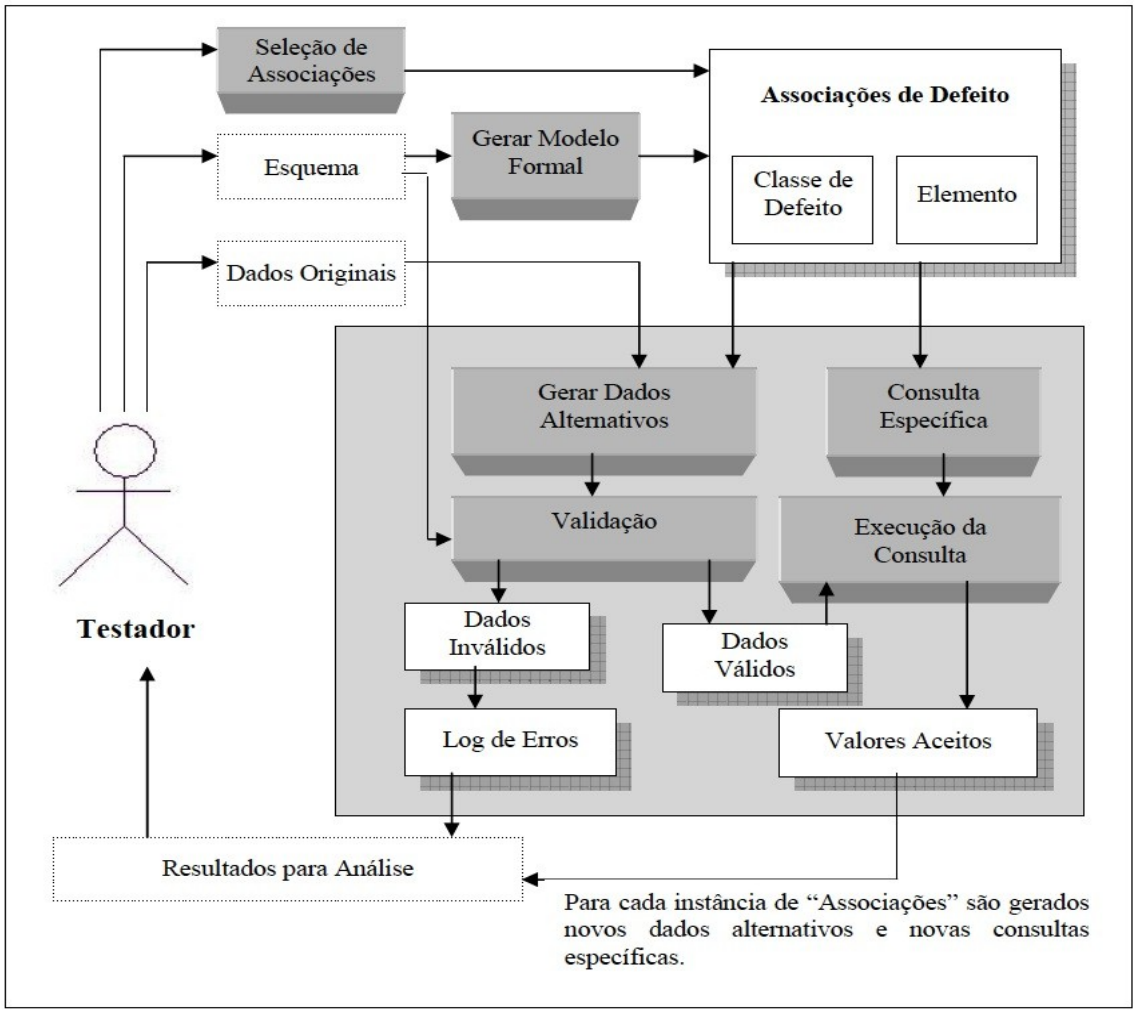

Figura 3.11 - Arquitetura da Ferramenta XTool (extraído de [41])

A ferramenta XTool provê uma interface gráfica, por meio da qual o testador informa como parâmetro de entrada o esquema (XML Schema ou Schema PostGreSQL) a ser testado e quais testes deverão ser realizados a partir dos elementos do esquema. Os testes se referem às classes de defeito que estão separadas pelo tipo de restrição.

A ferramenta transforma, então, o esquema informado para o seu modelo formal correspondente, e a partir desse modelo associa os elementos com as respectivas classes de defeitos, gerando as associações de defeito. O testador poderá indicar outras associações, que não foram selecionadas automaticamente pela ferramenta, analisando a especificação do esquema e selecionando as classes de defeito mais pertinentes ao processo de teste, e que têm maiores possibilidades de revelar defeitos.

Os resultados obtidos através da execução da ferramenta compõem um relatório, que é

utilizado pelo testador para comparar os resultados esperados segundo a especificação do esquema com os resultados obtidos, a fim de localizar definições incorretas ou a ausência de restrições no esquema.

3.5.2 A Ferramenta WSeTT

Com o propósito de auxiliar a aplicação da abordagem AIDA no teste de WS, foi desenvolvida a ferramenta WSeTT (Web Service Auxiliary Testing Tool) [53]. A ferramenta foi desenvolvida utilizando a linguagem Java. A interface gráfica com o usuário foi desenvolvida utilizando o pacote Swing, e as mensagens SOAP são construídas com o auxílio do Velocity. Uma vez criadas, as mensagens SOAP são enviadas, recebidas e processadas com o auxílio do Axis2. Para realizar as consultas XQuery é utilizado o framework Qexo [47].

A ferramenta implementa alguns operadores de mutação que foram desenvolvidos levando em consideração as restrições presentes nos documentos WSDL. Os operadores propostos são descritos na Tabela 3.5.

Tabela 3.5 – Operadores de mutação da WSeTT (extraída de [53])

Operador	**Descrição**
mudaAssinatura Serviço	Troca a ordem dos parâmetros de um serviço disponibilizado no WSDL. O operador troca somente a ordem dos elementos responsáveis por definir a assinatura de um serviço disponibilizado pelo Serviço Web.
mudaTipoAssinatura Serviço	Ao invés de modificar a ordem dos parâmetros na assinatura de um serviço, esse operador modificará apenas o tipo entre dois parâmetros.
mudaTipoElemento Mensagem	Modifica o tipo de um elemento definido em uma mensagem por um elemento de outro tipo definido em outra mensagem também presente no mesmo documento WSDL.
mudaInputOutput	Modifica o tipo da mensagem de input pelo tipo da mensagem de output de uma mesma operação definida no documento WSDL.

Com a ferramenta WSeTT é possível realizar consultas, baseadas nos operadores propostos e nas classes de defeitos propostas por Emer [25], aos documentos WSDL com a finalidade de detectar erros na especificação dos mesmos. Estas consultas foram previamente sumarizadas na Tabela 3.3.

Através da ferramenta WSeTT, o testador pode visualizar o resultado de cada consulta, e posteriormente confrontar este resultado com a especificação do serviço, analisando se os mesmos estão em conformidade com a especificação. A Figura 3.12 ilustra de maneira simplificada esse processo, exemplificado na Seção 3.4.3.

A WSeTT implementa também uma estratégia de teste que consiste em gerar mensagens SOAP modificadas de acordo com os operadores propostos. Essas mensagens são enviadas ao Serviço Web em teste e as mensagens de retorno são analisadas.

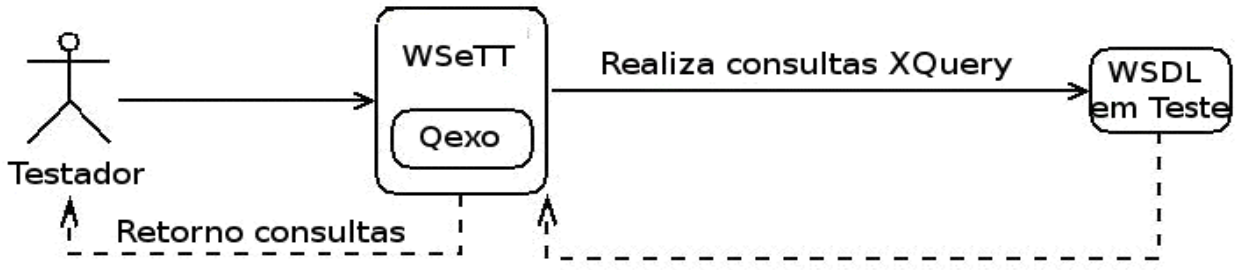

Figura 3.12 - Arquitetura do procedimento para realizar consultas XQuery (extraída de [53])

3.6 CONSIDERAÇÕES FINAIS

A abordagem AIDA visa o teste de esquemas de dados. Na literatura são poucos os trabalhos com este objetivo, e estes trabalhos focalizam em apenas um tipo de esquema. A AIDA é uma abordagem de teste baseada em defeitos, que pode ser aplicada em esquemas de diferentes contextos. Através da AIDA obtém-se a geração automática do conjunto de teste, o que não acontece com muitas das abordagens existentes.

A AIDA é a única abordagem genérica que pode ser aplicada para testar diferentes tipos de esquema e que auxilia no processo de revelar outros tipos de defeitos nas aplicações de software

que manipulam dados descritos por um esquema.

Para utilização da AIDA pode-se ainda contar com ferramentas, com interface gráfica, que auxiliam o processo de geração das instâncias alternativas, facilidade esta que nem sempre está disponível nas outras abordagens existentes.

Neste trabalho, optou-se por utilizar a AIDA para o teste de aplicações devido aos fatores apresentados acima, e para que se possa avaliar a sua eficiência no teste de aplicações reais, o que revela um estudo inédito na literatura.

A ideia é utilizar as instâncias de dados alternativas que são geradas pela AIDA, considerando defeitos presentes no esquema, para realizar o teste das aplicações que utilizam estes esquemas.

Dois contextos foram escolhidos: aplicações reais de banco de dados relacional, e aplicações de Serviços Web. Nos próximos capítulos (respectivamente os Capítulos 4 e 5) são introduzidas abordagens de aplicação da AIDA nestes contextos.

4 UTILIZAÇÃO DA ABORDAGEM AIDA PARA O TESTE DE APLICAÇÕES DE BANCO DE DADOS RELACIONAL

Como mencionado anteriormente, a abordagem AIDA permite o teste de diferentes tipos de esquema, através de instâncias de dados alternativas. Neste capítulo é descrito como a abordagem AIDA pode ser utilizada no teste de aplicações que utilizam banco de dados, a partir de esquemas de banco de dados relacionais. A estratégia proposta para o teste de tais aplicações é esquematizada na Figura 4.1.

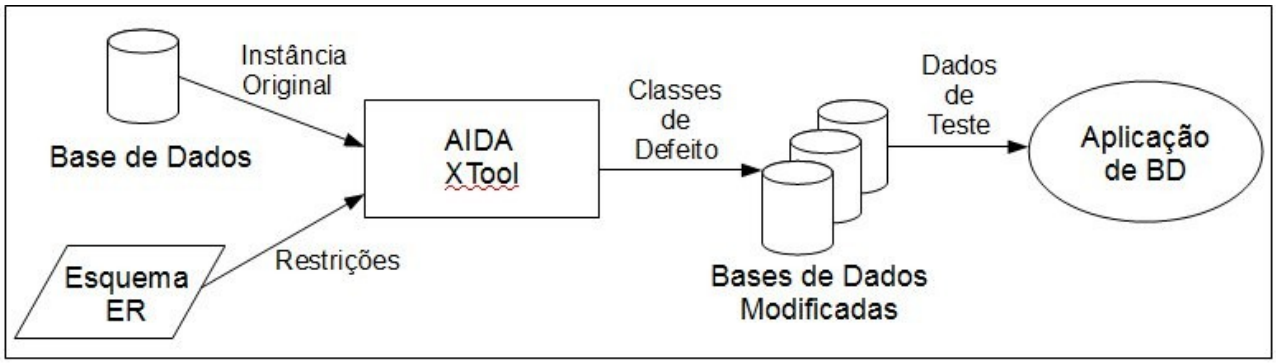

Figura 4.1 - Teste de Aplicações que usam banco de dados relacional usando a AIDA

A abordagem AIDA, que tem como entrada o esquema em teste, produz diferentes bases de dados modificadas a partir de uma instância original. As instâncias da base de dados alternativas geradas a partir da classe de defeito, em conjunto com consultas SQL, permitirão o teste do esquema ER.

Entretanto as instâncias alternativas podem ser vistas como dados de teste (entrada) para a aplicação que utiliza o esquema ER. Elas poderão revelar defeitos na aplicação que têm como causa uma especificação incorreta do esquema, ou uma parte do código que não foi implementada de acordo com o esquema especificado.

A estratégia portanto permite o teste da aplicação considerando defeitos no esquema. Para avaliar esta proposta foi conduzido um experimento com aplicações reais, descrito nas próximas seções.

4.1 DESCRIÇÃO DOS SISTEMAS UTILIZADOS

Nesta seção são apresentados três sistemas reais, acessados através de ambiente Web, que por estarem atualmente em utilização são identificados por Sistema 1, Sistema 2 e Sistema 3. Os sistemas foram desenvolvidos por uma empresa governamental.

O Sistema 1 (S1) é uma solução para registro das informações referentes às realizações governamentais, integrada a sistemas de informações operacionais de planejamento orçamentário e execução financeira, sendo estes registros consolidados em uma base de dados única, objetivando a visualização integrada e consistente das realizações de governo através de ferramentas de Inteligência de Negócio e de Georeferenciamento.

O Sistema 2 (S2) é uma solução para o controle de atendimentos (solicitações, sugestões, reclamações, elogios e denúncias a respeito de ações e programas dos órgãos ligados ao Poder Executivo Estadual), permitindo o acompanhamento das solicitações abertas e comunicação direta com os atendentes cadastrados no sistema. O sistema possibilita ainda a consulta, de forma quantitativa, dos atendimentos realizados de acordo com vários critérios.

O Sistema 3 (S3) tem por objetivo permitir o acompanhamento e controle financeiro de programas governamentais financiados com recursos externos. O sistema é uma solução para o registro e acompanhamento das informações financeiras dos programas cadastrados. O sistema permite também a consolidação das informações em demonstrativos com propósito de auditoria externa.

4.2 DESCRIÇÃO DOS ESQUEMAS UTILIZADOS

Na Tabela 4.1 são apresentadas algumas características dos esquemas testados. É também apresentado o número de registros associado a cada entidade da instância de dados original, utilizada para a realização dos testes.

Tabela 4.1: Características dos esquemas testados.

Aplicação	Esquema	Entidade	Número de atributos	Número de restrições	Número de registros
S1	Esquema 1	tb_empreendimento	22	3	5
		tb_orgaounidade	20	2	7
		tb_agrupador2	5	3	6
		tb_realizacao	10	4	7
		tb_fonterecurso	8	1	7
		tb_empenho	19	1	7
		tb_categoriabempublico	5	2	7
S2	Esquema 2	tb_orgao	34	2	8
		tb_suborgao	19	2	8
		tb_atendimento	41	6	7
		tb_encaminhamento	17	5	7
		tb_usuario	11	3	7
S3	Esquema 3	tb_conta_bancaria	7	1	11
		tb_item_prog_fin_executor	4	1	9
		tb_cambio	7	1	11
		tb_item_estrutura_poa_executor	4	0	15
		tb_item_estrutura_poa	5	0	8

4.3 METODOLOGIA

O experimento foi realizado para testar aplicações de bases de dados relacionais em PostGreSQL, sendo que o desenvolvimento das aplicações já tinha sido finalizado. Os defeitos que foram detectados durante o desenvolvimento foram resgatados e semeados nas aplicações e nos esquemas, de modo que as aplicações foram testadas contendo defeitos reais.

Com o objetivo de testar as aplicações, inicialmente a XTool foi usada para testar os esquemas de dados e obter os dados de teste (instâncias alternativas) que foram utilizados no teste da aplicação. O testador fez uso das especificações dos dados das aplicações, para que os resultados obtidos com a execução da XTool fossem analisados e para que possíveis defeitos no esquema fossem detectados.

As etapas do experimento foram as seguintes:

1) Os defeitos detectados durante o desenvolvimento das aplicações foram resgatados e semeados nas respectivas aplicações e esquemas;
2) A XTool foi executada para testar os esquemas, por meio de associações de defeito

selecionadas automaticamente pela ferramenta;

3) O relatório gerado pela XTool foi analisado, comparando-se os resultados obtidos através das consultas SQL geradas pela XTool, com os resultados esperados de acordo com a especificação de cada esquema, para verificar se foram revelados defeitos no esquema;
4) Os dados de teste obtidos com a execução da XTool foram aplicados no teste das aplicações correspondentes, buscando-se detectar os defeitos semeados na aplicação;
5) Os resultados obtidos nos testes dos esquemas e das aplicações foram analisados, e as conclusões com relação a eficácia dos testes foram obtidas.

As etapas de 1 a 3 estão associadas a aplicação tradicional da AIDA para teste de esquema, e as etapas 4 e 5 dizem respeito a estratégia proposta na Figura 4.1.

4.4 DEFEITOS SEMEADOS

Através da análise de relatórios gerados em uma aplicação Web (Bugzilla), que contém um histórico dos defeitos revelados durante o desenvolvimento de sistemas, bem como de suas respectivas resoluções, selecionou-se um grupo de defeitos para cada aplicação. Estes defeitos foram inseridos novamente no código representando defeitos reais encontrados durante o teste das aplicações. Alguns defeitos dizem respeito ao esquema, e alguns à aplicação. As informações sobre os esquemas, entidades e tipos de defeitos inseridos são apresentadas na Tabela 4.2, podendo-se ter também uma noção da complexidade de cada esquema. Uma descrição mais detalhada dos defeitos semeados é apresentada no Apêndice B.

Tabela 4.2 - Aplicações/Esquemas testados

Sistema	Esquemas Testados/Total	Entidades Testadas/Total	Atributos	Restrições	Registros	Defeitos Inseridos	
						Esquema	Aplicação
S1	1/2	7/70	89	16	46	3	8
S2	1/1	5/40	122	18	37	1	7
S3	1/1	5/55	27	3	54	0	5

Foi realizado o teste de apenas um esquema de cada aplicação, de modo que o esquema selecionado foi o esquema considerado principal com relação à importância dos dados descritos. Para simplificar o processo de teste, nos sistemas S1 e S2 foram testadas apenas as entidades de cada esquema relacionadas aos defeitos semeados. No sistema S3 foi realizado o teste considerando todas as entidades relacionadas ao único esquema contido na aplicação, tendo em vista que as entidades deste esquema possuem uma complexidade relativamente baixa.

4.5 EXECUÇÃO DA XTOOL

A XTool foi executada para testar os esquemas, por meio de associações de defeito selecionadas automaticamente pela ferramenta. Os dados referentes às associações de defeito selecionadas pela XTool, obtidos na execução da ferramenta são apresentados nas Tabelas 4.3, 4.4 e 4.5.

Nas Tabelas 4.3, 4.4 e 4.5 são exibidos o número de associações de defeito, o número de instâncias de dados alternativas e o número de consultas geradas pela XTool para cada entidade, que representa uma tabela, de cada esquema. Estas entidades foram selecionadas pois são as relacionadas com os defeitos semeados.

O número de associações de defeito mostrado para cada entidade nas Tabelas 4.3, 4.4 e 4.5, representa o número de associações selecionadas automaticamente pela ferramenta. A partir destas associações selecionadas foram geradas as instâncias alternativas que foram validadas automaticamente. A ferramenta gerou um número de consultas a estas instâncias alternativas, que auxilam o testador na tarefa de detectar defeitos através da comparação dos resultados obtidos com os resultados esperados.

Tabela 4.3: Número de associações de defeito, instâncias de dados e consultas geradas para o Esquema 1.

Entidade	**Número de associações de defeito**	**Número de instâncias de dados alternativas**		**Número de consultas**
		Válidas	**Inválidas**	
tb_empreendimento	20	23	15	43
tb_orgaounidade	12	13	28	25
tb_agrupador2	7	12	11	19
tb_realizacao	18	22	17	40
tb_fonterecurso	1	0	2	1
tb_empenho	1	0	2	1
tb_categoriabempublico	12	18	4	30

Tabela 4.4: Número de associações de defeito, instâncias de dados e consultas geradas para o Esquema 2.

Entidade	**Número de associações de defeito**	**Número de instâncias de dados alternativas**		**Número de consultas**
		Válidas	**Inválidas**	
tb_orgao	17	15	53	32
tb_suborgao	19	5	63	24
tb_atendimento	30	55	48	85
tb_encaminhamento	77	42	44	77
tb_usuario	22	14	56	36

Tabela 4.5: Número de associações de defeito, instâncias de dados e consultas geradas para o Esquema 3.

Entidade	Número de associações de defeito	Número de instâncias de dados alternativas		Número de consultas
		Válidas	Inválidas	
tb_carga_siaf_despesa_origem	1	0	2	1
tb_conta_bancaria	3	8	4	11
tb_carga_siaf	1	0	2	1
tb_carga_siaf_credor	1	0	2	1
tb_continente	2	1	14	3
tb_item_estrutura_executor	15	85	4	100
tb_ativo	8	48	4	56
tb_endereco	9	49	5	58
tb_despesa_origem	1	0	2	1
tb_despesa_origem_item_estrutura	17	85	2	102
tb_documento	9	43	17	52
tb_despesa	2	1	9	3
tb_executor_programa	18	101	4	119
tb_idioma	1	0	2	1
tb_idioma_pessoa	1	0	2	1
tb_fonte_executor	9	57	3	66
tb_modalidade_desembolso	3	0	10	3
tb_item_estrutura_sub_programa	10	60	2	70
tb_item_estrutura_poa_executor	4	23	4	27
tb_item_prog_fin_executor	5	2	30	7
tb_item_programacao_financeira	10	35	15	45
tb_controle_carga_siaf	10	42	3	52
tb_localidade	4	8	16	12
tb_desembolso	3	1	16	4
tb_item_estrutura	16	46	36	62
tb_moeda	3	2	22	5
tb_moeda_pais	7	20	30	27
tb_movimento_financeiro_origem	27	167	16	194
tb_nacionalidade	10	62	4	72
tb_projeto_atividade_programa	12	44	13	56
tb_pessoa	7	0	45	7
tb_nota_explicativa	4	18	3	22
tb_executor	9	0	73	9
tb_pais	6	6	34	12
tb_item_estrutura_poa	4	15	20	19
tb_plano_operativo_anual	2	1	11	3
tb_pedido_desembolso	10	45	2	55
tb_movimento_financeiro_desdobrado	34	166	30	200
tb_programacao_financeira	12	44	11	56
tb_programa	10	45	20	55
tb_teste_banco	3	6	3	9
tb_teste_fornecedor	3	6	3	9
tb_vinculo_localidade_regional	4	5	16	9
tb_tipo_endereco	2	5	4	7
tb_sub_programa	4	3	18	7
tb_tipo_documento	4	3	24	7
tb_tipo_lancamento_contabil	1	2	0	3
tb_teste_pessoa	3	2	6	5
tb_unidade_federativa	6	30	11	36
tb_situacao_origem	3	1	10	4
tb_regional	3	5	12	8
tb_tipo_regional	2	1	6	3
tb_cambio	4	23	3	27
tb_tipo_cambio	2	1	14	3
tb_localizacao	2	1	8	3
tb_localizacao	2	1	8	3

4.6 DEFEITOS REVELADOS

Nas próximas seções são apresentados os defeitos revelados nos testes do esquema e da aplicação.

4.6.1 Defeitos Revelados no Teste do Esquema

O relatório gerado pela XTool foi analisado e comparado com os resultados esperados, de acordo com a especificação dos dados, para verificar se foram revelados defeitos. Os dados resultantes são apresentados nas Tabelas 4.6 e 4.7. Nesta seção não são apresentados defeitos encontrados no Sistema 3, pois para este sistema não foram semeados defeitos de esquema.

Tabela 4.6: Defeitos de esquema revelados no Sistema 1.

Sistema 1			
Esquema	**Entidade**	**Atributo**	**Defeito revelado**
Esquema 1	tb_agrupador2	codagrupador2	Constraint violation exception de chave primária. As sequences do banco de dados estavam “zeradas”.
Esquema 1	tb_orgao_unidade	ind_administracao	Foi possível inserir valores nulos para o campo ind_administracao, que segundo a especificação deveria ser. NOT NULL
Esquema 1	tb_empreendimento	data_ultima_alteracao	Foi possível inserir valores nulos para o campo data_ultima_alteracao que segundo a especificação deveria ser NOT NULL.
Número de defeitos revelados *(por esquema)*			**3**
Total de defeitos semeados *(por esquema)*			**3**

Tabela 4.7: Defeitos de esquema revelados no Sistema 2.

Sistema 2			
Esquema	**Entidade**	**Atributo**	**Defeito revelado**
Esquema 2	tb_suborgao	cepsuborgao	O campo cepsuborgao deveria ser char(9) ao invés de char(8).
Número de defeitos revelados *(por esquema)*			**1**
Total de defeitos semeados *(por esquema)*			**1**

As Tabelas 4.8 e 4.9 apresentam os defeitos revelados por esquema, agrupados por critério de teste.

Tabela 4.8: Defeitos revelados por critério de teste para o Sistema 1.

Sistema 1							
Esquema	**Critério de teste**	**Classe de defeito**	**Número de associações de defeito**	**Número de consultas**	**Número de instâncias de dados alternativas**		**Número de defeitos revelados**
					Válidas	**Inválidas**	
Esquema 1	Restrição de Definição	Uso Incorreto	2	5	3	11	2
	Restrição de Domínio	Tipo de Dado Incorreto	1	6	5	0	1
Número de defeitos revelados *(por esquema)*							**3**
Total de defeitos semeados *(por esquema)*							**3**

Tabela 4.9: Defeitos revelados por critério de teste para o Sistema 2.

Sistema 2							
Esquema	**Critério de teste**	**Classe de defeito**	**Número de associações de defeito**	**Número de consultas**	**Número de instâncias de dados alternativas**		**Número de defeitos revelados**
					Válidas	**Inválidas**	
Esquema 2	Restrição de Domínio	Tipo de Dado Incorreto	1	8	7	0	1
Número de defeitos revelados *(por esquema)*							**1**
Total de defeitos semeados *(por esquema)*							**1**

No teste dos esquemas referentes aos Sistemas 1 e 2 as classes de defeito Uso Incorreto e Tipo de Dado Incorreto produziram a maior quantidade de associações de defeito e consultas e, conforme pode-se observar nas Tabelas 4.8 e 4.9, revelaram mais defeitos.

A Tabela 4.10 mostra um resumo dos resultados obtidos com relação aos defeitos nos esquemas das aplicações, comparando as quantidades de defeitos semeados e revelados. Conforme mencionado anteriormente, o Sistema 3 não apresentou defeitos na especificação do esquema pois para este sistema todos os defeitos semeados foram oriundos da aplicação.

Tabela 4.10: Resumo dos resultados do teste de esquema.

Aplicação	Número de defeitos semeados no esquema	Número de defeitos revelados	Defeitos não revelados
Sistema 1	3	3	0
Sistema 2	1	1	0
Sistema 3	0	0	0

4.6.2 Defeitos Revelados no Teste da Aplicação

Os dados de teste obtidos com a execução da XTool nos esquemas de dados foram aplicados no teste das aplicações correspondentes.

Cada dado de teste gerado pela XTool foi usado na aplicação como parâmetro de entrada para cada respectivo atributo. Desta forma, para cada funcionalidade da aplicação testada, executou-se uma operação de inclusão para cada dado de teste gerado pela XTool. Assim sendo, a cada execução utilizou-se como parâmetro de entrada em cada funcionalidade o dado de teste gerado pela XTool, e para os demais parâmetros de entrada foram utilizados dados válidos segundo a especificação do esquema.

Para cada operação de inclusão, os parâmetros de entrada foram inseridos: através da aplicação, de acordo com os campos disponíveis na interface da aplicação; ou através de scripts de inserção direta no banco de dados, quando a interface da aplicação não possibilita a aplicação direta do caso de teste.

Em cada execução comparou-se o resultado obtido com o resultado esperado, de acordo com a especificação. No Apêndice C são apresentados os defeitos de aplicação revelados no processo de teste, assim como os casos de teste utilizados para revelá-los. Alguns defeitos de aplicação não foram revelados através da execução dos casos de teste gerados automaticamente pela XTool.

No Sistema 1, para os defeitos: “Problemas no cadastramento de Empenho” e “Tela em branco na manutenção de Categoria”, os dados de teste gerados não produziram nenhuma mensagem de erro; e para o defeito “Adm Direta deve ter pelo menos pelo menos 1 empenho”, a XTool não produziu nenhum dado de teste que testasse a condição que produzia defeito.

No Sistema 2: para o defeito “Bug na consulta de Atendimentos” não foi gerado nenhum dado de teste com número de atendimento igual ou superior a 10000, para que fosse possível testar a necessidade de colocar "/ano" para que a consulta fosse realizada com sucesso; para o defeito “Erro na exclusão de atendimentos” não foi gerado nenhum dado de teste que produzisse algum registro que gerasse algum erro na sua exclusão; para o defeito “Bug na consulta de encaminhamentos” não foi gerado nenhum dado de teste que produziu algum defeito na consulta de registros de encaminhamento; no defeito “Resposta via email suprimindo informações” não foi gerado um dado de teste contendo aspas que poderia produzir um defeito na resposta via email; e no defeito “Login de usuário aceitando espaços” não foi gerado nenhum dado de teste contendo o caractere espaço em branco.

No Sistema 3: a ferramenta não produziu dados de teste com o número da agência bancária igual a zero, nem com o número da conta bancária igual a zero para revelar os dois defeitos semeados relacionados à inclusão de Conta Bancária; para o defeito “Erro na exibição de valores da Programação Financeira” não foram produzidos dados de teste da Programação Financeira que ao serem exibidos os valores aparecessem zerados; e para o defeito “Erro na pesquisa de câmbio” não foram gerados dados de teste referentes ao período de 09/07/2008 a 10/07/2008 que ao serem exibidos mostrariam os valores dos dias 08/07 e 10/07.

Esses defeitos não revelados, para os três sistemas, poderiam ter sido revelados através da introdução manual de associações de defeito pelo testador. Um ponto a ser investigado é a influência da instância de dados original fornecida para o teste. Instâncias com uma maior quantidade e variedade de registros levam a geração de instâncias alternativas mais variadas, e isto poderia aumentar a eficácia do teste e possivelmente contribuir para revelar estes defeitos. A XTool possui um mecanismo para possibilitar a introdução de associações de defeito pelo testador, entretanto, seu desempenho pode ser reduzido ao manipular instâncias originais com uma grande quantidade de registros.

A Tabela 4.11 mostra um resumo dos resultados obtidos na busca de defeitos através dos testes executados na aplicação, comparando as quantidades de defeitos semeados e revelados.

Tabela 4.11: Resumo dos resultados do teste da aplicação.

Aplicação	Número de defeitos semeados na aplicação	Número de defeitos revelados	Defeitos não revelados
Sistema 1	8	5	3
Sistema 2	7	2	5
Sistema 3	5	1	4

As Tabelas 4.12, 4.13 e 4.14 mostram as quantidades de dados de teste gerados automaticamente pela XTool, por critério de teste, para cada aplicação.

Tabela 4.12: Resultados por critério de teste para o sistema 1.

Critério de teste	Classe de defeito	Dados de teste
Restrição de Definição	Uso Incorreto	73
Restrição de Domínio	Tipo de Dado Incorreto	945

Tabela 4.13: Resultados por critério de teste para o sistema 2.

Critério de teste	Classe de defeito	Dados de teste
Restrição de Definição	Uso Incorreto	71
Restrição de Domínio	Tipo de Dado Incorreto	1091

Tabela 4.14: Resultados por critério de teste para o sistema 3.

Critério de teste	Classe de defeito	Dados de teste
Restrição de Definição	Uso Incorreto	210
Restrição de Domínio	Tipo de Dado Incorreto	3520

4.7 DISCUSSÃO

O número de instâncias alternativas (dados de teste) e de consultas geradas pela XTool cresce com relação ao número de associações de defeito estabelecidas, como esperado. Entretanto, parece não haver uma relação observável entre este número e o número de elementos no esquema. A semântica do mesmo parece influenciar além da estrutura. Isto também ocorre no caso do teste de programas durante a aplicação de alguns critérios de teste, no qual LOC ou mesmo a complexidade

ciclomática V(G) parecem não influenciar tanto no número de elementos requeridos e/ou dados de teste necessários. Isto deverá ser explorado em trabalhos futuros. Similarmente ao teste de mutação utilizando-se operadores essenciais [5], poderá ser introduzido o conceito de associações de defeito essenciais.

Vale a pena ressaltar que a ferramenta XTool apresenta um mecanismo para que o usuário selecione as associações identificadas automaticamente, ou que estabeleça outras, caso ache necessário. Isto poderá diminuir o custo da aplicação da AIDA, permitindo a seleção de algumas associações específicas para o tipo de aplicação sendo testada.

Com relação à eficácia, observa-se que a estratégia foi capaz de revelar todos os defeitos de esquema em teste. Com relação ao teste das aplicações, vale a pena observar que a estratégia foi capaz de revelar defeitos relacionados exclusivamente ao funcionamento da aplicação, apesar de não ser este o seu objetivo, sendo 62,5% dos defeitos foram revelados em S1, 28,6% em S2 e 20% em S3. Isto mostra que os dados podem ser utilizados de maneira complementar aos dados gerados por outros tipos de teste nestas aplicações. É interessante observar que a maior eficácia está no sistema S1, para o qual foram gerados menos testes. Este sistema tem uma característica diferente dos demais que é o fato de lidar com dois esquemas de dados, manipular diferentes tipos de dados em diferentes funcionalidades. Este fato parece apontar que a utilização de dados no teste da aplicação, usando uma abordagem de teste de esquema, se justifica quando o sistema não manipula uma grande quantidade de dados, mas quando os tipos dos dados manipulados forem bastante diversificados.

4.8 CONSIDERAÇÕES FINAIS

Neste capítulo foram apresentados os resultados da aplicação da abordagem AIDA no teste de aplicações que utilizam banco de dados relacional, com o auxílio da ferramenta XTool, que constitui-se no primeiro contexto de aplicação da AIDA proposto neste trabalho. A estratégia proposta foi capaz de revelar todos os defeitos dos esquemas testados, e no teste das aplicações a estratégia revelou 62,5% dos defeitos em S1, 28,6% em S2 e 20% em S3.

No próximo capítulo será apresentado o segundo contexto de aplicação da estratégia proposta, que consiste na aplicação da estratégia no teste de aplicações que utilizam Serviços Web.

5 UTILIZAÇÃO DA ABORDAGEM AIDA PARA O TESTE DE APLICAÇÕES DE SERVIÇOS WEB

Neste capítulo é introduzida uma estratégia de aplicação da AIDA no contexto de teste de Serviços Web. A estratégia é ilustrada na Figura 5.1.

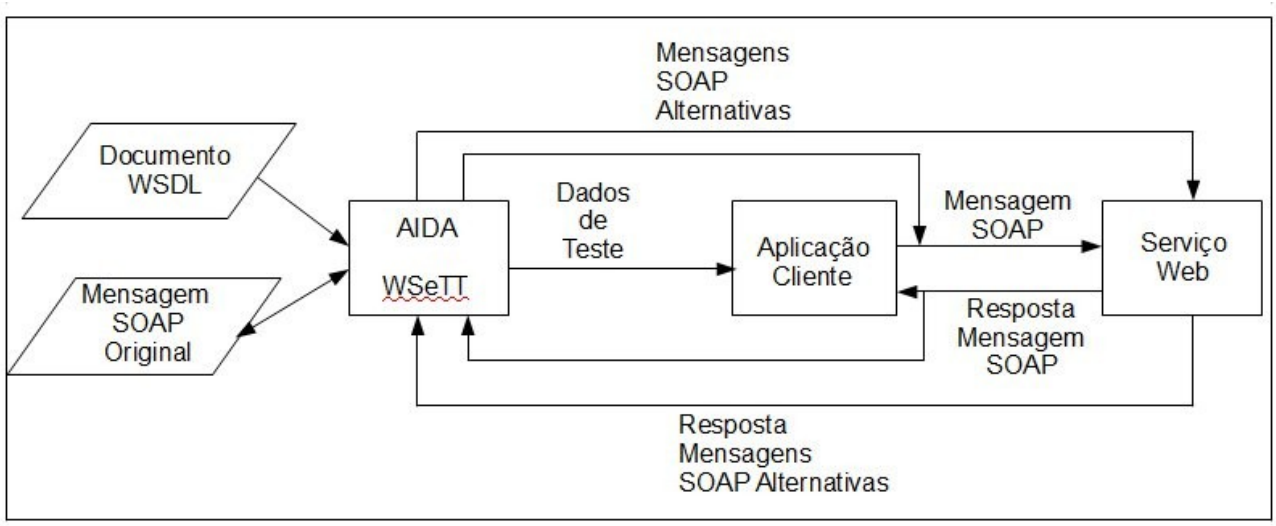

Figura 5.1 – Diagrama que ilustra o uso da WSeTT para auxiliar o teste de aplicações que usam WS

Além de permitir o teste do próprio documento WSDL através de consultas XQuery, propõe-se que as instâncias WSDL modificadas sejam utilizadas para gerar mensagens SOAP alternativas a partir de uma mensagem original. Estas mensagens SOAP alternativas podem ser utilizadas para o teste do Serviço Web diretamente ou serem utilizadas para a geração de dados de teste para a aplicação que utiliza o Serviço Web (aplicação cliente). Esta, por sua vez, enviará mensagens SOAP ao Serviço Web como resultado da entrada recebida. Em ambas as situações o resultado (resposta, mensagem SOAP retornada) é analisado pelo testador para verificar a existência ou não de um defeito.

Semelhantemente ao que foi realizado no contexto de aplicação de bases de dados relacionais (Capítulo 4), para avaliar a estratégia foi realizado um experimento com três aplicações reais.

5.1 DESCRIÇÃO DOS SISTEMAS UTILIZADOS

Três sistemas desenvolvidos pela mesma empresa pública que desenvolveu os sistemas descritos no experimento do Capítulo 4 foram utilizados, que por questões de sigilo são identificados por Sistema 1, Sistema 2 e Sistema 3.

A seguir é apresentada uma breve descrição da funcionalidade de cada sistema utilizado, que na verdade é a aplicação cliente na estratégia proposta.

Sistema 1 (S1): tem como função o controle das disponibilidades de vagas ofertadas e contratações de trabalho aprendiz no âmbito estadual.

Sistema 2 (S2): registra os processos e suas tramitações, auxilia os procuradores do Estado na realização de suas atividades fim, propiciando um incremento na arrecadação da Dívida Ativa através de um melhor gerenciamento dos executivos fiscais e dados dos contribuintes, além de cruzamento estratégico de informações.

Sistema 3 (S3): ferramenta de apoio aos processos produtivos definidos pela metodologia de uma empresa, na forma de registro do planejamento, da execução e do controle dos processos produtivos em relação aos produtos (artefatos) gerados no desenvolvimento de software.

5.2 DESCRIÇÃO DOS SERVIÇOS WEB UTILIZADOS

As aplicações descritas no item anterior utilizam Serviços Web, cujas funcionalidades são brevemente descritas a seguir. Da mesma forma como foi definido para os sistemas utilizados, por questões de sigilo os Serviços Web são identificados como WS 1, WS 2 e WS 3.

WS 1: disponibilização de dados de endereçamento, municípios e estados do Brasil, que é utilizado pelo Sistema 1.

WS 2: serviço que tem como função a disponibilização de dados sobre dívidas ativas provenientes de um sistema *mainframe*, através de um módulo Web. O Sistema 2 acessa tais informações sobre dívidas ativas e também atualiza informações no módulo Web.

WS 3: atualização de dados de projetos cadastrados em um sistema para controle e acompanhamento de resultados, através do Sistema 3.

Algumas características destes Serviços Web, bem como a relação com as aplicações clientes são mostradas na Tabela 5.1.

Tabela 5.1: Características dos Serviços Web

Serviço Web	Número de Operações	Aplicação Cliente
WS 1	17	Sistema 1
WS 2	7	Sistema 2
WS 3	15	Sistema 3

5.3 METODOLOGIA

O experimento foi realizado para testar aplicações de Serviços Web que utilizam restrições definidas por um esquema, representado por um arquivo WSDL, sendo que o desenvolvimento das aplicações já tinha sido finalizado. Os defeitos que foram detectados durante o desenvolvimento foram resgatados, analisando-se as ocorrências registradas em uma ferramenta Bugzilla, e semeados nas aplicações, nos esquemas e nos Serviços Web, de modo que as aplicações foram testadas contendo defeitos reais.

Com o objetivo de testar as aplicações, inicialmente a WSeTT foi usada para testar o esquema e o Serviço Web, e obter os dados de teste (instâncias alternativas) que foram utilizados no teste da aplicação. O testador fez uso das especificações dos dados das aplicações, para que os resultados obtidos com a execução da WSeTT fossem analisados e para que possíveis defeitos no esquema e no Serviço Web fossem detectados.

As etapas do experimento foram as seguintes:

1) Os defeitos detectados durante o desenvolvimento das aplicações foram resgatados e semeados nas respectivas aplicações, esquemas e Serviços Web;
2) A WSeTT foi utilizada para realizar consultas XQuery ao documento WSDL, relacionadas às associações de defeito selecionadas pelo testador;
3) O resultado das consultas XQuery foi analisado, comparando-se os resultados obtidos com os resultados esperados de acordo com a especificação de cada esquema, para verificar se foram revelados defeitos nos esquemas;
4) Para cada operação de Serviço Web, gerou-se uma mensagem SOAP original, válida de acordo com a especificação do Serviço Web, e a partir desta mensagem original a ferramenta gerou mensagens SOAP alternativas para operador de mutação selecionado pelo testador, sendo estas mensagens alternativas enviadas ao Serviço Web pela própria ferramenta;
5) As mensagens SOAP retornadas pelo Serviço Web em resposta a cada mensagem alternativa enviada foram analisadas, comparando-se os resultados obtidos com os resultados esperados de acordo com a especificação de cada Serviço Web, para verificar se foram revelados defeitos nos Serviços Web;
6) Para cada mensagem SOAP alternativa gerada pela ferramenta, gerou-se um dado de teste que foi aplicado diretamente nas aplicações correspondentes, buscando-se detectar os defeitos semeados na aplicação;
7) Os resultados obtidos nos testes dos esquemas, dos Serviços Web e das aplicações foram analisados e as conclusões com relação a eficácia dos testes foram obtidas.

As etapas de 1 a 3 estão associadas a aplicação tradicional da AIDA para teste de esquema e de Serviço Web, e as etapas de 4 a 7 dizem respeito a estratégia proposta na Figura 5.1.

5.4 DEFEITOS SEMEADOS

Defeitos detectados durante o desenvolvimento das aplicações, relacionados ao uso dos Serviços Web foram identificados e reimplantados. Esses defeitos são identificados e descritos na Tabela 5.2. Eles são classificados em defeitos de esquema, referentes ao WSDL, da aplicação cliente e do Serviço Web.

Tabela 5.2: Defeitos semeados

Aplicação	Tipo Defeito	Localização	Descrição
S1	Aplicação	Cadastrar Empresa	A aplicação aceita cep que não seja do Paraná.
	Aplicação	Cadastrar Instituição Formadora	A aplicação não valida o número de dígitos válidos para um cep.
	Aplicação	Cadastrar Instituição Formadora	A aplicação não testa se o cep informado é nulo.
	Aplicação	Cadastrar Instituição Formadora	A aplicação não testa se o logradouro informado é nulo.
	Aplicação	Cadastrar Empresa	A aplicação não retorna um endereço ao ser informado um cep válido.
	Serviço Web	buscarEnderecoPorCep	O serviço retorna nulo quando recebe um cep válido.
	Esquema	buscarEnderecoPorCep	A ordem dos dois parâmetros de entrada foi invertida em relação à especificação.
	Serviço Web	buscarLogradouroPorNome	O serviço não retorna nenhuma opção ao ser informado um logradouro existente.
	Serviço Web	buscarLogradouroPorNome	O serviço não retorna o bairro correspondente ao logradouro informado.
S2	Aplicação	Obter lista de Certidão de Dívida Ativa (CDA)	Selecionando-se uma Procuradoria, Comarca e a lista de Municípios, não é retornada a correspondente lista de CDAs.
	Aplicação	Obter lista de Certidão de Dívida Ativa (CDA)	O sistema não valida se foi selecionado pelo menos um Município, e retorna uma mensagem de erro ao tentar obter a lista de CDAs.
	Aplicação	Pegar para Ajuizar	A aplicação mostra uma mensagem de erro ao tentar pegar para ajuizar uma lista de CDAs.
	Aplicação	Ajuizar	A aplicação mostra uma mensagem de erro ao tentar ajuizar uma lista de CDAs.
	Serviço Web	listarDividasPorMunicipio	Passando uma lista de Municípios válidos, não é retornada a correspondente lista de CDAs.
	Serviço Web	gerarCertidoes	O sistema não faz o processo de ajuizamento da lista de CDAs, e retorna uma mensagem de erro.
S3	Aplicação	Pesquisar Projeto	O sistema não retorna nenhum projeto como resultado da pesquisa.
	Aplicação	Incluir Projeto	O sistema exibe uma mensagem de erro ao tentar incluir um projeto.
	Aplicação	Incluir/Alterar Projeto	O sistema não valida o número máximo de caracteres para o nome de um projeto.
	Aplicação	Incluir/Alterar Projeto	O sistema não valida o formato das datas de início e término previsto.
	Aplicação	Incluir/Alterar Projeto	O sistema não valida se campos obrigatórios foram preenchidos.
	Aplicação	Alterar Projeto	O sistema gera uma mensagem de erro ao tentar alterar informações de um projeto.
	Serviço Web	atualizaProjeto	Um novo projeto não é incluído.
	Esquema	atualizaProjeto	Não é gravada a Data de Término do projeto informado pelo usuário.
	Serviço Web	listaProjetos	O Serviço Web não retorna nenhum projeto como resultado da pesquisa.

Tabela 5.3: Totais de defeitos semeados

Sistema	Defeitos de Aplicação	Defeitos de Esquema	Defeitos de WS	Total
S1	5	1	3	9
S2	4	0	2	6
S3	6	1	2	9

Observando-se os defeitos que foram semeados, mostrados na Tabela 5.3, percebe-se que a estratégia proposta pretende detectar defeitos relacionados a aplicação, ao esquema e ao Serviço Web.

5.5 EXECUÇÃO DA WSETT

Conforme descrição da metodologia contida na Seção 5.3, a ferramenta WSeTT foi executada para testar os Serviços Web, por meio de consultas XQuery, e de mensagens SOAP alternativas, geradas a partir dos operadores de mutação da Tabela 3.5, que são enviadas ao Serviço Web pela simulação de clientes feita pela própria ferramenta.

Os resultados obtidos com as consultas XQuery e as mensagens SOAP geradas, são apresentados nas Tabelas 5.4 e 5.5.

Tabela 5.4: Número de operadores de mutação e consultas geradas pela WSeTT

Serviço Web	**Operador de mutação**	**Número de consultas**
WS 1	Assinatura Serviços	17
WS 1	Tipo Assinatura Serviços	17
WS 1	Input Output	21
WS 2	Assinatura Serviços	7
WS 2	Tipo Assinatura Serviços	7
WS 2	Input Output	7
WS 3	Assinatura Serviços	15
WS 3	Tipo Assinatura Serviços	15
WS 3	Input Output	15
WS 3	Incorrect Occurrence	1
WS 3	Incorrect Data Types	12

Na Tabela 5.5 são apresentados os números de mensagens SOAP consideradas inválidas. Tais mensagens SOAP foram consideradas inválidas por serem mensagens, geradas automaticamente pela ferramenta, que contém o número de parâmetros diferente da correspondente operação do Serviço Web, e portanto não puderam ser enviadas ao Serviço Web pela ferramenta.

Tabela 5.5: Operadores de mutação e mensagens SOAP geradas pela WSeTT

Serviço Web	**Operação**	**Operador de mutação**	**Número de mensagens SOAP**	
			Válidas	Inválidas
WS 1	buscaEnderecoPorCep	Muda Assinatura Serviço	1	0
	buscaEnderecoPorCep	Muda Tipo Assinatura Serviço	1	0
	buscaEnderecoPorCep	Muda Tipo Elemento Mensagem	130	992
	buscarLogradouroPorNome	Muda Assinatura Serviço	1	0
	buscarLogradouroPorNome	Muda Tipo Assinatura Serviço	1	0
	buscarLogradouroPorNome	Muda Tipo Elemento Mensagem	133	989
WS 2	listarDividasPorMunicipio	Muda Tipo Elemento Mensagem	108	102
	listarDividasPorMunicipio	Muda Input Output	1	0
	alterarDividasSelecionadas	Muda Tipo Elemento Mensagem	112	98
	alterarDividasSelecionadas	Muda Input Output	1	0
	geraCertidoes	Muda Tipo Elemento Mensagem	85	125
	geraCertidoes	Muda Input Output	1	0
WS 3	listaProjetos	Muda Assinatura Serviço	23	0
	listaProjetos	Muda Tipo Assinatura Serviço	23	0
	listaProjetos	Muda Tipo Elemento Mensagem	61	869
	listaProjetos	Muda Input Output	0	1
	atualizaProjeto	Muda Assinatura Serviço	23	0
	atualizaProjeto	Muda Tipo Assinatura Serviço	23	0
	atualizaProjeto	Muda Tipo Elemento Mensagem	60	870
	atualizaProjeto	Muda Input Output	0	1

As mensagens geradas a partir dos operadores de mutação Muda Tipo Elemento Mensagem e Muda Input Output, disponíveis na ferramenta, produzem apenas a estrutura da mensagem SOAP, não contendo os respectivos dados de teste na mensagem, devido ao fato de que não foi implementada na ferramenta uma heurística que automatizasse o processo de geração automática destes dados de teste. Assim, analisou-se o padrão implementado na ferramenta XTool para geração de dados de teste, e foram identificados os padrões de dados de teste apresentados na Tabela 5.6.

Tabela 5.6: Conjunto de dados de teste baseado no padrão definido na XTool

Tipo Atributo	Dado de Teste
byte	12345
	null
datetime	2005-10-22
	null
double	12345
	null
float	1234.56
	null
integer	12345
	null
string	abc1234
	null
string[]	abc1234
	null

Assim como na metodologia utilizada pela ferramenta XTool, utilizaram-se como dados de teste, além do conjunto definido na Tabela 5.6, dados válidos para cada atributo, ou seja, foram geradas mensagens com parâmetros válidos, definidas na ferramenta WseTT como “Mensagem Original”.

Percebe-se pela análise da Tabela 5.5 que a ferramenta gerou automaticamente um grande número de mensagens SOAP mutantes. A partir das mensagens mutantes geradas, mensagens adicionais foram geradas, para que fosse possível aplicar os dados de teste apresentados na Tabela 5.6.

O cálculo do número de mensagens adicionais geradas é detalhado na Tabela 5.7.

Tabela 5.7: Cálculo das Mensagens Adicionadas

Serviço Web	Operação	Operador de Mutação	Mensagens geradas pela WSeTT	Combinações (Casos de Teste * Número de Parâmetros – 1 mensagem original gerada pelo operador)			Total de Mensagens Adicionadas (Mensagens geradas pela WseTT * Total de Combinações)
				Casos de Teste	Número de Parâmetros	Total	
WS 1	buscaEnderecoPorCep	Muda Tipo Elemento Mensagem	130	2	2	3	390
WS 1	buscaLogradouroPorNome	Muda Tipo Elemento Mensagem	133	2	2	3	399
Total de Mensagens Adicionadas para WS 1							789
WS 2	listarDividasPorMunicipio	Muda Tipo Elemento Mensagem	108	2	1	1	108
WS 2	alterarDividasSelecionadas	Muda Tipo Elemento Mensagem	112	2	1	1	112
Total de Mensagens Adicionadas para WS 2							220
WS 3	listaProjetos	Muda Tipo Elemento Mensagem	61	2	4	7	427
WS 3	atualizaProjetos	Muda Tipo Elemento Mensagem	60	2	4	7	420
Total de Mensagens Adicionadas para WS 3							847
Total de Mensagens Adicionadas							1856

5.6 DEFEITOS REVELADOS

Nas próximas seções são apresentados os defeitos revelados nos testes do esquema, do Serviço Web e da aplicação.

5.6.1 Defeitos Revelados no Teste do Esquema

Os defeitos de esquema (documento WSDL) revelados através da execução das consultas XQuery são apresentados na Tabela 5.8.

Tabela 5.8: Defeitos revelados nas consultas XQuery

Serviço Web	**Operação**	**Operador de mutação**	**Consulta**	**Defeito revelado**
WS 1	buscarEnderecoPorCep	Incorrect Order	<operation name="buscarEnderecoPorCep" parameterOrder="tipoCodificacaoLocalidadecep">	A ordem dos parâmetros está incorreta.
WS 3	atualizaProjeto	Incorrect Data Types	<element name="dataFinal" nillable="true" type="xsd:dateTime" />	Não é gravada a Data de Término do projeto informado pelo usuário.
Número de defeitos de esquema (documento WSDL) inseridos: 2				
Número de defeitos revelados: 2				

5.6.2 Defeitos Revelados no Teste do Serviço Web

Com o objetivo de revelar defeitos nos Serviços Web , utilizou-se a ferramenta WSeTT para gerar mensagens SOAP, e enviar estas mensagens para o Serviço Web através da própria ferramenta.

Os dados de teste mostrados na Tabela 5.7 foram utilizados no teste dos Serviços Web (Web Services - WS) WS 1, WS 2 e WS 3, e os resultados da execução dos testes são apresentados nas Tabelas 5.9, 5.10 e 5.11. No Apêndice A são apresentadas as mensagens SOAP que revelaram os defeitos correspondentes a estas tabelas.

Tabela 5.9: Defeitos revelados através de mensagens SOAP em WS 1

Serviço Web	Operação	Operador de mutação	Defeito revelado
WS 1	buscarEnderecoPorCep	Mensagem Original	O serviço retorna nulo quando recebe um cep válido.
	buscarEnderecoPorCep	Muda Tipo Assinatura Serviço	O serviço retorna nulo quando recebe um cep válido.
	buscarEnderecoPorCep	Muda Assinatura Serviço	A ordem dos parâmetros está incorreta, pois foi retornada uma mensagem de erro ao enviar para o Serviço Web uma mensagem SOAP com a ordem dos parâmetros de acordo com a especificação do Serviço Web.
	buscarLogradouroPorNome	Mensagem Original	O serviço não retorna nenhuma opção ao ser informado um logradouro existente.
	buscarLogradouroPorNome	Mensagem Original	O serviço não retorna o bairro correspondente ao logradouro informado.

Tabela 5.10: Defeitos revelados através de mensagens SOAP em WS 2

Serviço Web	Operação	Operador de mutação	Defeito revelado
WS 2	listarDividasPorMunicipio	Mensagem Original	Selecionando-se Municípios válidos, não é retornada a correspondente lista de CDAs.
	gerarCertidoes	Mensagem Original	O sistema não faz o processo de ajuizamento da lista de CDAs, e retorna uma mensagem de erro.
	gerarCertidoes	Muda Tipo Elemento Mensagem	O sistema não faz o processo de ajuizamento da lista de CDAs, e retorna uma mensagem de erro.
	gerarCertidoes	Muda Input Output	O sistema não faz o processo de ajuizamento da lista de CDAs, e retorna uma mensagem de erro.

Tabela 5.11: Defeitos revelados através de mensagens SOAP em WS 3

Serviço Web	Operação	Operador de mutação	Defeito revelado
WS 3	atualizaProjeto	Mensagem Original	Um novo projeto não foi incluído pois não foi possível atualizar o banco de dados.
	atualizaProjeto	Muda Assinatura Serviço	Um novo projeto não foi incluído pois não foi possível atualizar o banco de dados.
	atualizaProjeto	Muda Tipo Elemento Mensagem	Um novo projeto não foi incluído pois não foi possível atualizar o banco de dados.
	atualizaProjeto	Mensagem Original	Não foi gravada a Data de Término do projeto informado pelo usuário.
	atualizaProjeto	Muda Assinatura Serviço	Não foi gravada a Data de Término do projeto informado pelo usuário.
	atualizaProjeto	Muda Tipo Elemento Mensagem	Não foi gravada a Data de Término do projeto informado pelo usuário.
	listaProjetos	Mensagem Original	O Serviço Web não retorna nenhum projeto como resultado da pesquisa.
	listaProjetos	Muda Tipo Elemento Mensagem	O Serviço Web não retorna nenhum projeto como resultado da pesquisa.

Os defeitos detectados pelas mensagens geradas a partir dos operadores de mutação, mostrados na Tabelas 5.9, 5.10 e 5.11 representam a primeira mensagem de cada operador que foi capaz de revelar o defeito, não sendo necessariamente a única mensagem que foi capaz de revelar o defeito em questão.

A Tabela 5.12 apresenta um resumo dos resultados obtidos nos testes dos três Serviços Web.

Tabela 5.12: Resultado do teste de Serviço Web

Serviço Web	Número de Consultas	Número de Mensagens SOAP			Número de defeitos semeados	Número de defeitos revelados	Defeitos não revelados
		Geradas	Adicionadas	Total			
WS 1	55	267	789	1056	3	3	0
WS 2	21	308	220	528	2	2	0
WS 3	58	213	847	1060	2	2	0
TOTAL	**134**	**788**	**1856**	**2644**	**7**	**7**	**0**

Das 134 consultas geradas para os três Serviços Web, apenas duas efetivamente revelaram defeitos.

Para WS 1: das 267 mensagens geradas 2 revelaram defeitos, das 789 mensagens adicionadas nenhuma revelou defeitos, e 3 mensagens originais revelaram defeitos. Para WS 2: das 308 mensagens geradas 1 revelou defeito, das 220 mensagens adicionadas 108 revelaram defeitos e 2 mensagens originais revelaram defeitos. Para WS 3: das 213 mensagens geradas 6 revelaram defeitos, das 847 mensagens adicionadas 2 revelaram defeitos, e 3 mensagens originais revelaram defeitos.

5.6.3 Defeitos Revelados no Teste da Aplicação

Os dados de teste (mensagens SOAP) obtidos com a execução da WSeTT foram utilizados no teste das aplicações clientes.

Da mesma forma como os dados de teste mostrados na Tabela 5.6 foram utilizados no teste do WS 1, do WS 2 e do WS 3, os mesmos dados de teste foram utilizados no teste do S1, do S2 e do S3, sendo os resultados da execução dos testes apresentados nas Tabelas 5.13, 5.14 e 5.15.

Os defeitos detectados pelos casos de teste obtidos a partir das mensagens SOAP, geradas através dos operadores de mutação, mostrados na Tabelas 5.13, 5.14 e 5.15 representam o primeiro caso de teste de cada operador que foi capaz de revelar o defeito, não sendo necessariamente o único caso de teste que foi capaz de revelar o defeito em questão.

Tabela 5.13: Defeitos revelados no teste do Sistema 1

Aplicação	Funcionalidade	Operador de Mutação	Dado de teste	Resultado esperado	Resultado obtido
Sistema 1	Cadastrar Instituição Formadora	Muda Assinatura Serviço	cep:2005-10-22 tipoCodificacaoLocalidade: 1	Mensagem de validação do numero de dígitos do CEP.	A aplicação não valida o número de dígitos válidos para um CEP e gera um erro inesperado.
Sistema 1	Cadastrar Instituição Formadora	Muda Assinatura Serviço	cep:null tipoCodificacaoLocalidade: 1	Mensagem de validação informando que o valor do CEP não pode ser nulo.	A aplicação não testa se o CEP informado é nulo e gera um erro inesperado.
Sistema 1	Cadastrar Empresa	Mensagem Original	cep:80810280 tipoCodificacaoLocalidade: 1	Endereço correspondente ao cep informado.	A aplicação não retorna um endereço ao ser informado um CEP válido.
Sistema 1	Cadastrar Instituição Formadora	Muda Assinatura Serviço	nomeLogradouro: null chaveBairro: 36492	Mensagem de validação informando que o nome do logradouro não pode ser nulo.	A aplicação não testa se o logradouro informado é nulo.

Tabela 5.14: Defeitos revelados no teste do Sistema 2

Aplicação	Funcionalidade	Operador de Mutação	Dado de teste	Resultado esperado	Resultado obtido
Sistema 2	Obter lista de Certidão de Dívida Ativa (CDA)	Mensagem Original	procuradoria: 1 comarca: 1 municipios: 7535, 7537, 7539	Lista de CDAs correspondente.	Não foi retornada a correspondente lista de CDAs.
Sistema 2	Obter lista de Certidão de Dívida Ativa (CDA)	Muda Tipo Elemento Mensagem	procuradoria: 1 comarca: 1 municipios: null	Mensagem informando que o usuário deve selecionar pelo menos um Município.	O sistema não validou se foi selecionado pelo menos um Município, e retornou uma mensagem de erro ao tentar obter a lista de CDAs.
Sistema 2	Pegar para ajuizar	Mensagem Original	listaDividasSelecionadas: 32, 34, 35	Exibir a tela de sucesso ao pegar para ajuizar e para confirmar o processo de ajuizar as CDAs.	Mensagem de erro inesperado.
Sistema 2	Ajuizar	Mensagem Original	listaDividasSelecionadas: 32, 34, 35	Mensagem de sucesso ao ajuizar as CDAs.	Mensagem de erro inesperado.

Tabela 5.15: Defeitos revelados no teste do Sistema 3

Aplicação	**Funcionalidade**	**Operador de Mutação**	**Dado de teste**	**Resultado esperado**	**Resultado obtido**
Sistema 3	Pesquisar Projeto	Mensagem Original	Nome do Projeto: saff Nome do Cliente: sepl registroInicial: 1 qtde: 10	Exibir como resultado da pesquisa o projeto saff.	O sistema não retornou nenhum projeto como resultado da pesquisa.
Sistema 3	Pesquisar Projeto	Muda Tipo Elemento Mensagem	Nome do Projeto: null Nome do Cliente: sepl registroInicial: 1 qtde: 10	Exibir como resultado da pesquisa todos os projetos do cliente sepl.	O sistema não retornou nenhum projeto como resultado da pesquisa.
Sistema 3	Incluir/Alterar Projeto	Mensagem Original	Nome do Projeto: saff Data de Início: 01/01/2008 Data de Término: 31/12/2009	Mensagem de sucesso de inclusão ou alteração do projeto.	O sistema exibiu uma mensagem de erro ao tentar incluir ou alterar um projeto.
Sistema 3	Incluir/Alterar Projeto	Muda Tipo Elemento Mensagem	Nome do Projeto: abc1234 Data de Início: 01/01/2008 Data de Término: 31/12/2009	Mensagem de sucesso de inclusão ou alteração do projeto.	O sistema exibiu uma mensagem de erro ao tentar incluir ou alterar um projeto.
Sistema 3	Incluir/Alterar Projeto	Muda Assinatura Serviço	Nome do Projeto: saff Data de Início: 31/12/2009 Data de Término: 01/01/2008	Mensagem de alerta informando que a data de término deve ser maior que a de início.	O sistema não validou que a data de término é menor que a data de início e procedeu com o cadastro do projeto.
Sistema 3	Incluir/Alterar Projeto	Muda Tipo Elemento Mensagem	Nome do Projeto: saff Data de Início: 2005-10-22 Data de Término: 31/12/2009	Mensagem de alerta informando que os formatos das datas estão incorretos.	O sistema não validou o formato da data de início.

A Tabela 5.16 apresenta um resumo do teste das três aplicações. O número de dados de teste apresentado para cada aplicação representa a soma do número de mensagens SOAP mutantes geradas automaticamente pela WSeTT, com o número de mensagens SOAP adicionais geradas para executar os casos de teste mostrados na Tabela 5.6.

Tabela 5.16: Resultados do teste da aplicação

Aplicação	**Número de dados de teste**	**Número de defeitos semeados**	**Número de defeitos revelados**	**Defeitos não revelados**
Sistema 1	4212	5	4	1
Sistema 2	528	4	4	0
Sistema 3	1060	6	3	3
TOTAL		**15**	**11**	**4**

5.7 DISCUSSÃO

Através deste experimento pode-se verificar a eficiência da ferramenta WSeTT no suporte à detecção de defeitos em Serviços Web. A ferramenta foi capaz de gerar automaticamente uma grande quantidade de consultas XQuery e também de mensagens SOAP mutantes, que em muito auxiliaram o processo de teste. Todos os defeitos de esquema semeados foram revelados pela estratégia aplicada no experimento.

Os dados de teste obtidos com a execução da WSeTT, a partir das mensagens SOAP mutantes, foram aplicados no teste das aplicações clientes obtendo-se bons resultados, sendo que dentre os 15 defeitos semeados menos de 27% dos defeitos de aplicação não foram revelados pela estratégia.

O esforço gasto no processo de teste, principalmente na geração das mensagens SOAP mutantes com os respectivos dados de teste, foi consideravelmente grande, pois foi necessário gerar manualmente os dados de teste para grande parte das mensagens mutantes geradas. Esta tarefa seria em grande parte facilitada se houvesse uma heurística na ferramenta WSeTT que fosse capaz de gerar os dados de teste automaticamente nas mensagens SOAP mutantes. Apesar disso, a ferramenta facilitou o teste pois ajudou a estruturar possíveis combinações dos parâmetros de entrada e saída dos Serviços Web, considerando possíveis defeitos presentes no Serviço Web e no documento WSDL, cabendo ao testador apenas fornecer os dados de teste.

5.8 CONSIDERAÇÕES FINAIS

Neste capítulo foi apresentada uma estratégia de aplicação da AIDA no teste de aplicações que utilizam Serviços Web. Foi também descrito um experimento realizado em aplicações reais que serviu de base para avaliar a eficácia da estratégia aplicada.

Pode-se concluir pela análise dos resultados obtidos que a estratégia aplicada no experimento é bastante válida, tanto para o teste de esquemas, dos Serviços Web e das aplicações clientes, pois o custo/benefício da estratégia foi bastante favorável tendo-se em vista os resultados finais obtidos, e a geração automática dos dados de teste.

6 CONCLUSÕES

Este trabalho teve como objetivo explorar o uso da abordagem de teste de esquemas AIDA no teste de aplicações. Assim, duas estratégias de uso da AIDA foram propostas: uma no contexto de aplicações de banco de dados relacional, e outra no contexto de Serviços Web. Nessas estratégias, as instâncias alternativas, obtidas a partir de classes de defeito que representam defeitos contidos no esquema utilizado pela aplicação, foram associadas a dados de teste na aplicação. Desta forma espera-se identificar defeitos presentes no esquema que podem estar refletidos na aplicação.

Para validar as estratégias propostas foram conduzidos dois experimentos relativos a ambos os contextos, utilizando aplicações reais (Capítulos 4 e 5). Nestes experimentos buscou-se revelar defeitos detectados durante o desenvolvimento das aplicações, que foram reimplantados para avaliar a eficácia das abordagens.

Pela análise dos resultados obtidos na utilização da abordagem AIDA para o teste de aplicações de banco de dados relacional e de Serviços Web, conforme apresentado nos Capítulos 4 e 5, respectivamente, pode-se concluir que o uso da estratégia permitiu revelar todos os defeitos de esquema semeados. Além disso, produziu resultados significativos na detecção dos defeitos semeados nas aplicações testadas, revelando 40% dos defeitos semeados no contexto de banco de dados relacional, e 80% no contexto de Serviços Web. O contexto de Serviços Web parece ser um contexto mais propício ao uso da estratégia, mas isto deverá ser investigado em trabalhos futuros.

A utilização das ferramentas XTool e WSeTT auxiliou na geração das instâncias alternativas, que foram utilizadas como dados de teste para testar as aplicações. No entanto, a falta de um processo automatizado que pudesse extrair os dados de teste para uso direto nas aplicações que foram testadas, gerou um grande esforço por parte do testador, tendo em vista que em ambos os contextos foi gerado um grande número de instâncias alternativas.

Tem-se então as principais contribuições deste trabalho:

- Introdução de estratégias de uso da AIDA para o teste de aplicações, utilizando as instâncias alternativas geradas pela abordagem como dados de teste para as aplicações. Desta forma é

utilizada uma abordagem baseada em defeitos, que considera que defeitos existentes no esquema podem estar refletidos na aplicação, o que representa uma contribuição com relação aos trabalhos existentes na literatura;

- Avaliação das abordagens propostas através de experimentos realizados em aplicações reais, onde buscou-se detectar defeitos identificados durante o desenvolvimento das aplicações.

6.1 TRABALHOS FUTUROS

Algumas possibilidades de extensão do presente trabalho incluem:

- Geração automática de dados de teste através da WSeTT: implementar na ferramenta WSeTT um mecanismo para gerar automaticamente os dados de teste para a aplicação, a partir das mensagens SOAP geradas pela ferramenta;
- Conduzir novos experimentos em novas aplicações, nos mesmos contextos utilizados neste trabalho, buscando-se avaliar a eficácia dos critérios de teste associados a AIDA, e a partir de um maior escopo definir a melhor forma de aplicação da estratégia para que se obtenha resultados que melhor contribuirão para a atividade de teste, e também reduzir o custo de aplicação da estratégia;
- Comparação com outros tipos de técnicas e estratégias, tais como as descritas no Capítulo 2;
- Utilização da estratégia em aplicações que usam outros tipos de esquema, como por exemplo de banco de dados orientado a objetos e esquemas de dado objeto-relacional, podendo-se gerar as instâncias alternativas (dados de teste) a partir de diagramas de classe.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Almeida, L. F. Explorando Teste Baseado em Perturbação no Contexto de Web Services. Dissertação de Mestrado, UFPR - Universidade Federal do Paraná, Curitiba - PR, 2006.

[2] Almeida, L. F.; Vergilio, S. R. Exploring Perturbation Based Testing for Web Services. In ICWS'06: Proceedings of the IEEE International Conference on Web Services, pages 717-726, Washington, DC, USA, 2006. IEEE Computer Society.

[3] Aranha, M. C. L. F .M.; Mendes, N. C.; Jino, M.; Toledo, C. M. T. RDBTool: Uma Ferramenta de Apoio ao Teste de Bases de Dados Relacionais. In XI CITS: Conferência Internacional de Tecnologia de Software. Agosto, 2000.

[4] Árias, J. C. G..; Emer, M. C. P.; Vergilio, S. R. Testando Aplicações de Banco de Dados com Análise de Instâncias de Dados Alternativas. Conferência Latinoamericana de Informática (CLEI 2010). Setembro, 2010.

[5] Barbosa, E.F.; Vincenzi, A.M.R.; Maldonado, J.C. “Uma Contribuição para a Determinação de um Conjunto Essencial de Operadores de Mutação no Teste de Programas C”. Anais do XII Simpósio Brasileiro de Engenharia de Software, Maringá, PR, Brasil, Outubro, 1998.

[6] Bertolino, A.; Gao, J.; Marchetti, E.; Polini, A. Automatic Test Data Generation for XML Schema-based Partition Testing. AST '07: Proceedings of the Second International Workshop on Automation of Software Test. May, 2007.

[7] Bloomberg, J. Report: Testing Web Services, 2004. Disponível em "http://www.parasoft.com/jsp/templates/misc/soap/web_services_excerpt.pdf". Acessado em Julho de 2010.

[8] Bultan, B.; Fu, X.; Hull, R.; and Su, J. Conversation specification: A new approach to design and analysis of e-service composition. In WWW '03: Proceedings of the 12th international conference on World Wide Web, pages 403-410, New York, NY, USA, 2003. ACM.

[9] Chan, M.; Cheung, S. Testing Database Applications with SQL Semantics. In Proc. of the 2nd Intl. Symp. on Cooperative Database Systems for Advanced Applications, pp 364-375, March 1999.

[10] Chays, D.; Dan, S.; Frankl, P. G.; Vokolos, Filippos I.; Weyuker, Elaine J. A Framework for Testing Database Applications. In Proc. of the 2000 ACM SIGSOFT Intl. Symp. on Software Testing and Analysis, Vol. 25 Issue 5, August 2000.

[11] Chays, D.; Deng, Y. Demonstration of AGENDA Tool Set for Testing Relational Database Applications. In Proc. of the 25th Intl. Software Engineering Conference, 2003. IEEE Computer Society, pp 802 – 803, May 2003.

[12] Cruz Filho, Paulo N.; Vergilio, S. R. Teste de Software Baseado em Perturbação de Dados Dirigida por Padrões. Dissertação de Mestrado. Universidade Federal do Paraná – UFPR, 2007.

[13] Cruz Filho, P. N. ; Vergilio, S. R. Uma Ferramenta para o Teste de Web Services Baseado em Perturbação de Dados Dirigida por Padrões. In: XXI Simpósio Brasileiro de Engenharia de Software, XIV Sessão de Ferramentas. João Pessoa, 2007. v. 1. p. 63-69.

[14] Cruz Filho, P. N. ; Vergilio, S. R. . Using XML Patterns to Guide Perturbation Based Testing of Web Services. In: International Conference on Software Engineering and Knowledge Engineering, 2008, San Francisco.. v. 1. p. 197-202.

[15] Deng, Yuetang; Frankl, Phyllis; Chays, David. Testing Database Transactions with AGENDA. In Proc. of the 27th Intl. Conference on Software engineering. ACM Press, May 2005.

[16] Di Lucca, G.A.; Di Penta, M. Considering Browser Interaction in Web Application Testing. In Proceedings of 5th IEEE Intl. Workshop on Web Site Evolution. IEEE Computer Society Press, 2003.

[17] Di Lucca, G.A.; Fasolino, A.R.; Faralli, F.; De Carlini, U. Testing Web applications. In Proceedings of International Conference on Software Maintenance, pages 3–6. IEEE Press, outubro 2002.

[18] Di Lucca, G.A.; Fasolino, A.R.; Pace, F.; Tramontana, P.; De Carlini, U. WARE: A tool for the Reverse Engineering of Web Applications. In Proceedings of VI Eur. Conf. on Software Maintenance and Reengineering. IEEE Computer Society Press, março 2002.

[19] Document Object Model (DOM), disponível em: http://www.w3c.org/DOM/, Acessado em Dezembro de 2010.

[20] Dong, W. Testing WSDL Based Web Service Automatically. WCSE '09: Proceedings of the 2009 WRI World Congress on Software Engineering - Volume 03. May, 2009.

[21] Emer, M. C. F. P.; Vergilio, S. R.; Jino, M. A Testing Approach for XML Schemas. Proceedings of the 29th Annual International Computer Software and Applications Conference - COMPSAC. Workshop on Quality Assurance and Testing of Web-Based Applications. 2005.

[22] Emer, M. C. F. P.; Vergilio, S. R.; Jino, M. Testing Relational Database Schemas with Alternative Instance Analysis. In Proc. Of the 20th International Conference on Software Engineering and Knowledge Engineering (SEKE 2008), 2008.

[23] Emer, M. C. P.; Nazar, I. F.; Vergilio, S. R. ; Jino, M. . Evaluating a Fault-Based Testing Approach for XML Schemas. In: IEEE Latin-American Test Workshop, 2007, Cuzco. 8th IEEE Latin-American Test Workshop, 2007. v. 1.

[24] Emer, M. C. P. ; Vergilio, S. R. ; Jino, M. ; Nazar, I. F. ; Caxeiro, P. V. . Uma Avaliação do Teste Estrutural Baseado em Defeitos em Esquemas de Banco de Dados Relacional. In: I Brazilian Workshop on Systematic and Automated Software Testing, João Pessoa, 2007. v. 1. p. 29-36.

[25] Emer, M. C. F. P. Abordagem de Teste Baseada em Defeitos para Esquemas de Dados. Tese de Doutorado, UNICAMP, Campinas, SP, 2007.

[26] Extensible Markup Language (XML), disponível em: http://www.w3c.org/XML/, Acessado em 2010.

[27] Franzotte, L; Vergilio, S. R. Applying Mutation Testing to XML Schemas. In Proc. of the 8th Inl. Conference on Software Engineering and Knowledge Engineering (SEKE2006), 2006.

[28] Genevès, P.; Layaïda, N.; Quint, V. Identifying query incompatibilities with evolving XML schemas. ICFP'09: Proceedings of the 14th ACM SIGPLAN International Conference on Functional Programming. August, 2009.

[29] Grechanik, M. Finding errors in components that exchange XML data. ASE '07: Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering. November, 2007.

[30] Grolinger, K.; Capretz, M. A. M. A unit test approach for database schema evolution. Information and Software Technology, Volume 53 Issue 2. February, 2011.

[31] Guangquan, Z.; Mei, R.; Jun, Z. A Business Process of Web Services Testing Method Based on UML2.0 Activity Diagram. IITA '07: Proceedings of the Workshop on Intelligent Information Technology Application. December, 2007.

[32] Heckel, Reiko and Lohmann, Marc. Towards Contract-based Testing of Web Services. Electronic Notes in Theorical Computer Science 82 No. 6, 2006.

[33] Kapfhammer, G. M.; Soffa, M. L. A Family of Test Adequacy Criteria for Database-driven Applications. In Proc. of the 9th European Software Engineering Conference, held jointly with 11th ACM SIGSOFT Intl. Symp. on Foundations of Software Engineering, Vol. 28 Issue 5, September 2003.

[34] Kung, D. C.; Liu, C.H.; Hsia, P. An Object-Oriented Web Test Model for Testing Web Applications. In The 24th Annual International Computer Software and Applications Conference, COMPSAC 2000, pages 537–542. IEEE Press, 2000.

[35] Lee, S. C.; Offutt, J. Generating test cases for XML-based web component interactions using mutation analysis. In 12th International Symposium on Software Reliability Engineering, pages 200–209. IEEE Press, novembro 2001.

[36] Leitão, P. S. J.; Vilela, P. R. S.; Jino, M. Mapping Faults to Failures in SQL Manipulation Commands. In Proc. of the 3rd ACS/IEEE International Conference on Computer Systems and Applications (AICCSA-05), janeiro 2005.

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

[38] Liu, C.H.; Kung, D. C.; Hsia, P. Object-based Data Flow Testing of Web Applications. In First Asia-Pacific Conference on Quality Software, pages 7–16. IEEE Press, 2000.

[39] Liu, C. H.; Kung, D. C.; Hsia, P.; Hsu, C. T. Structural Testing of Web Applications. In 11th International Symposium on Software Reliability Engineering, pages 84–96. IEEE Press, 2000.

[40] Maldonado, J. C. Critérios Potenciais Usos: Uma Contribuição ao Teste Estrutural de Software. Tese de doutorado, Departamento de Engenharia de Computação e Automação Industrial, Faculdade de Engenharia Elétrica e de Computação, UNICAMP, julho 1991.

[41] Nazar, I. F. XTool: Uma Ferramenta para Teste de Esquemas. Dissertação de Mestrado, Universidade Federal do Paraná - UFPR, Curitiba, PR, 2007.

[42] Nazar, I. F.; Emer, M. C. F. P.; Vergilio, S. R.; Jino, M. XTool: Uma Ferramenta de Teste Baseado em Defeitos para Esquemas de Dados. In Proc. 26º Simpósio Brasileiro de Redes de Computadores (SBRC 2008), Workshop de Testes e Tolerância a Falhas (WTF 2008), 2008.

[43] OMG. Object Constraint Language – OCL. Available Specification version 2.0. http://www.omg.org/spec/OCL/2.0/PDF/, acessado em Agosto de 2010.

[44] Offutt, J. and Xu, W. Generating Test Cases for Web Services Using Data Perturbation. ACM SIGSOFT, 2004.

[45] PostGreSQL disponível em: http://www.postgresql.org/docs/ . Acessado em Dezembro de 2010.

[46] Pressman, Roger. Engenharia de software. 6ª ed. São Paulo: Makron Books, 2006.

[47] Qexo - The GNU Kawa implementation of XQuery, 2006. Disponível em: http://www.gnu.org/software/qexo/. Acessado em Agosto de 2010.

[48] Rapps, S; Weyuker, E.J. Selecting software test data using data flow information. IEEE Transations on Software Engineering, 1985.

[49] Ricca, F.; Tonella, P. Analysis and Testing of Web Applications. In Proceedings of the 23rd International Conference on Software Maintenance, pages 25–34. IEEE Press, maio 2002.

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

[51] Siblini, R. and Mansour, N. Testing Web Services. In AICCSA'05: Proceedings of the ACS/IEEE 2005 International Conference on Computer Systems and Applications, pages 135-vii, Washington, DC, USA, 2005. IEEE Computer Society.

[52] Silberschatz, Abraham; Korth, Henry F.; Sudarshan, S. Sistema de Banco de Dados. Editora Campus, 2006.

[53] Solino, A. L. S. Teste Baseado em Defeitos para Web Services. Dissertação de Mestrado, Universidade Federal do Paraná - UFPR, Curitiba, PR, 2008.

[54] Spoto, E. S. Critério de Teste Estrutural para Programas de Aplicação de Base de Dados Relacional. Tese de doutorado, Departamento de Engenharia de Computação e Automação Industrial, Faculdade de Engenharia Elétrica e de Computação, UNICAMP, dezembro 2000.

[55] Structured Query Language (SQL), disponível em: http://en.wikipedia.org/wiki/SQL , Acessado em Dezembro de 2010.

[56] Suárez-Cabal, M. J.; Tuya, J. Using an SQL Coverage Measurement for Testing Database Applications. In Proc. of the 12th Intl. Symp. on the Foundations of Engineering, November 2004.

[57] Tongrak, P.; Suwannasart, T. “A tool for generating test case from relational database constraints testing”. Computer Science and Information Technology, 2009. ICCSIT 2009. Aug. 2009.

[58] Tsai, W. T.; Ray, P.; Song, W. and Cao, Z. Coyote: An XML Based Framework for Web Services Testing. In HASE '02: Proceedings of the 7th IEEE International Symposium on High As-

surance Systems Engineering (HASE'02), page 173, Washington, DC, USA, 2002. IEEE Computer Society.

[59] Web Services Description Language (WSDL) 1.1, disponível em: http://www.w3.org/TR/wsdl

[60] XML, disponível em: http://www.w3.org/XML/. Acessado em Fevereiro de 2011.

[61] XML Query Language. (XQuery), disponível em: http://www.w3c.org/XML/Query/. Acessado em Dezembro de 2010.

[62] XML Schema, disponível em: http://www.w3c.org/XML/Schema/ , Acessado em Dezembro de 2010.

[63] Xu, W.; Offutt, J. and Luo, J. Testing Web Services by XML Perturbation. In Proceeding of the 16th IEEE International Symposium on Software Reliability Engineering (ISSRE'05). IEEE, 2005.

[64] Zhang, J.; Xu, C.; Cheung, S. C. Automatic Generation of Database Instances for White-box Testing. In Proc. of the 25th Annual Intl. Computer Software and Applications Conference, pp 161 – 165, October 2001.

APÊNDICE A – MENSAGENS SOAP QUE REVELARAM DEFEITOS

Este apêndice apresenta as mensagens SOAP geradas no experimento descrito no Capítulo 5 que revelaram defeitos para WS 1, WS 2 e WS 3.

Tabela A1: Defeitos revelados através de mensagens SOAP em WS 1

Serviço Web	Operação	Operador de mutação	Mensagem SOAP	Defeito revelado
WS 1	buscarEnderecoPorCep	Mensagem Original	<ns1:buscarEnderecoPorCep xmlns:ns1="http://www.gic.desenvolvimento.eparana.parana:8080/cep/services/EnderecamentoInterface" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> <cep xsi:type="int">80810280</cep> <tipoCodificacaoLocalidade xsi:type="byte">1</tipoCodificacaoLocalidade> </ns1:buscarEnderecoPorCep>	O serviço retorna nulo quando recebe um cep válido.
WS 1	buscarEnderecoPorCep	Muda Tipo Assinatura Serviço	<ns1:buscarEnderecoPorCep xmlns:ns1="http://www.gic.desenvolvimento.eparana.parana:8080/cep/services/EnderecamentoInterface" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> <cep si:type="byte">80810280</cep> <tipoCodificacaoLocalidade xsi:type="int">1</tipoCodificacaoLocalidade> </ns1:buscarEnderecoPorCep>	O serviço retorna nulo quando recebe um cep válido.
WS 1	buscarEnderecoPorCep	Muda Assinatura Serviço	<ns1:buscarEnderecoPorCep xmlns:ns1="http://www.gic.desenvolvimento.eparana.parana:8080/cep/services/EnderecamentoInterface" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> <cep xsi:type="int">80810280</cep> <tipoCodificacaoLocalidade xsi:type="byte">1</tipoCodificacaoLocalidade> </ns1:buscarEnderecoPorCep>	A ordem dos parâmetros está incorreta, pois foi retornada uma mensagem de erro ao enviar para o Serviço Web uma mensagem SOAP com a ordem dos parâmetros de acordo com a especificação do Serviço Web.

WS 1	buscarLogradouroPorNome	Mensagem Original	<ns1:buscarLogradouroPorNome xmlns:ns1="http://www.gic.desenvolvimento.eparana.parana:8080/cep/services/EnderecamentoInterface" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> <nomeLogradouro xsi:type="string">Principal</nomeLogradouro> <chaveBairro xsi:type="int">36492</chaveBairro> </ns1:buscarLogradouroPorNome>	O Serviço Web não retorna nenhuma opção ao ser informado um logradouro existente.
WS 1	buscarLogradouroPorNome	Mensagem Original	<ns1:buscarLogradouroPorNome xmlns:ns1="http://www.gic.desenvolvimento.eparana.parana:8080/cep/services/EnderecamentoInterface" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> <nomeLogradouro xsi:type="string">Principal</nomeLogradouro> <chaveBairro xsi:type="int">36492</chaveBairro> </ns1:buscarLogradouroPorNome>	O serviço não retorna o bairro correspondente ao logradouro informado.

Tabela A2: Defeitos revelados através de mensagens SOAP em WS 2

Serviço Web	**Operação**	**Operador de mutação**	**Mensagem SOAP**	**Defeito revelado**
WS 2	listarDividasPorMunicipio	Mensagem Original	<ns1:listarDividasPorMunicipio xmlns:ns1="http://exception.framework.celepar.pr.gov" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> <listaCodMunicipios xsi:type="String[]">31,32,33</listaCodMunicipios> </ns1:listarDividasPorMunicipio>	Selecionando-se Municípios válidos, não é retornada a correspondente lista de CDAs.

WS 2	gerarCertidoes	Mensagem Original	<ns1:gerarCertidoes xmlns:ns1="http://exception.framework.celepar.pr.gov" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> </ns1:gerarCertidoes>	O sistema não faz o processo de ajuizamento da lista de CDAs, e retorna uma mensagem de erro.
WS 2	gerarCertidoes	Muda Tipo Elemento Mensagem	<ns1:gerarCertidoes xmlns:ns1="http://exception.framework.celepar.pr.gov" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> </ns1:gerarCertidoes>	O sistema não faz o processo de ajuizamento da lista de CDAs, e retorna uma mensagem de erro.
WS 2	gerarCertidoes	Muda Input Output	<ns1:gerarCertidoes xmlns:ns1="http://exception.framework.celepar.pr.gov" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> </ns1:gerarCertidoes>	O sistema não faz o processo de ajuizamento da lista de CDAs, e retorna uma mensagem de erro.

Tabela A3: Defeitos revelados através de mensagens SOAP em WS 3

Serviço Web	**Operação**	**Operador de mutação**	**Mensagem SOAP**	**Defeito revelado**
WS 3	atualizaProjeto	Mensagem Original	<ns1:atualizaProjeto xmlns:ns1="https://swebdesenv01.celepar.parana:8443/ecar/services/ItemWebService" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> <codigoProjeto xsi:type="int">10</codigoProjeto> <nomeProjeto xsi:type="string">saff</nomeProjeto> <dataInicio xsi:type="date">01/01/2008</dataInicio> <dataTermino xsi:type="date">31/12/2009</dataTermino>	Um novo projeto não foi incluído.

			</ns1:atualizaProjeto>	
WS 3	atualizaProjeto	Muda Assinatura Serviço	<ns1:atualizaProjeto xmlns:ns1="https://swebdesenv01.celepar.parana:8443/ecar/services/ItemWebService" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> <codigoProjeto xsi:type="int">10</codigoProjeto> <nomeProjeto xsi:type="string">saff</nomeProjeto> <dataTermino xsi:type="date">31/12/2009</dataTermino> <dataInicio xsi:type="date">01/01/2008</dataInicio> </ns1:atualizaProjeto>	Um novo projeto não foi incluído pois não foi possível atualizar o banco do e-CAR.
WS 3	atualizaProjeto	Muda Tipo Elemento Mensagem	<ns1:atualizaProjeto xmlns:ns1="https://swebdesenv01.celepar.parana:8443/ecar/services/ItemWebService" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> <codigoProjeto xsi:type="soapenc:int">10</codigoProjeto> <nomeProjeto xsi:type="soapenc:string"> abc1234 </nomeProjeto> <dataInicio xsi:type="soapenc:date">01/01/2008</dataInicio> <dataTermino xsi:type="soapenc:date">31/12/2009</dataTermino> </ns1:atualizaProjeto>	Um novo projeto não foi incluído pois não foi possível atualizar o banco do e-CAR.
WS 3	atualizaProjeto	Mensagem	<ns1:atualizaProjeto	Não foi gravada

		Original	xmlns:ns1="https://swebdesenv01.celepar.parana:8443/ecar/services/ItemWebService" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> <codigoProjeto xsi:type="int">10</codigoProjeto> <nomeProjeto xsi:type="string">saff</nomeProjeto> <dataInicio xsi:type="date">01/01/2008</dataInicio> <dataTermino xsi:type="date">31/12/2009</dataTermino> </ns1:atualizaProjeto>	a Data de Término do projeto informado pelo usuário.
WS 3	atualizaProjeto	Muda Assinatura Serviço	<ns1:atualizaProjeto xmlns:ns1="https://swebdesenv01.celepar.parana:8443/ecar/services/ItemWebService" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> <codigoProjeto xsi:type="int">10</codigoProjeto> <nomeProjeto xsi:type="string">saff</nomeProjeto> <dataTermino xsi:type="date">31/12/2009</dataTermino> <dataInicio xsi:type="date">01/01/2008</dataInicio> </ns1:atualizaProjeto>	Não foi gravada a Data de Término do projeto informado pelo usuário.
WS 3	atualizaProjeto	Muda Tipo Elemento Mensagem	<ns1:atualizaProjeto xmlns:ns1="https://swebdesenv01.celepar.parana:8443/ecar/services/ItemWebService" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">	Não foi gravada a Data de Término do projeto informado pelo usuário.

			<codigoProjeto xsi:type="soapenc:int">10</codigoProjeto> <nomeProjeto xsi:type="soapenc:string"> abc1234 </nomeProjeto> <dataInicio xsi:type="soapenc:date">01/01/2008</dataInicio> <dataTermino xsi:type="soapenc:date">31/12/2009</dataTermino> </ns1:atualizaProjeto>	
WS 3	listaProjetos	Mensagem Original	<ns1:listaProjetos xmlns:ns1="https://swebdesenv01.celepar.parana:8443/ecar/services/ItemWebService" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> <nomeProjeto xsi:type="string">saff</nomeProjeto> <nomeCli xsi:type="string">sepl</nomeCli> <registroInicial xsi:type="int">1</registroInicial> <qtde xsi:type="int">10</qtde>	O Serviço Web não retorna nenhum projeto como resultado da pesquisa.
WS 3	listaProjetos	Muda Tipo Elemento Mensagem	<ns1:listaProjetos xmlns:ns1="https://swebdesenv01.celepar.parana:8443/ecar/services/ItemWebService" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> <nomeProjeto xsi:type="string">null</nomeProjeto> <nomeCli xsi:type="string">sepl</nomeCli>	O Serviço Web não retorna nenhum projeto como resultado da pesquisa.

			<registroInicial xsi:type="int">1</registroInicial> <qtde xsi:type="int">10</qtde>	

APÊNDICE B – DEFEITOS IDENTIFICADOS DURANTE O DESENVOLVIMENTO DAS APLICAÇÕES QUE UTILIZAM BANCO DE DADOS RELACIONAL

Através da análise de relatórios gerados em uma aplicação Bugzilla, que contém um histórico dos defeitos revelados durante o desenvolvimento de sistemas, bem como de suas respectivas resoluções, obteve-se um grupo de defeitos para cada aplicação testada no experimento descrito no Capítulo 4. Estes defeitos são apresentados na Tabela B.1.

Tabela B.1: Defeitos identificados durante o desenvolvimento das aplicações que utilizam banco de dados relacional

Aplicação	Defeito	Espécie Defeito	Localização	Descrição
Sistema 1	Erro na quantidade da mensuração, na funcionalidade Manter Empreendimento.	Aplicação	Funcionalidade Manter Empreendimento .	Na função Manter Empreendimento, quando o usuário coloca algum dado não numérico na quantidade do item de mensuração e clica em alterar, a função apresenta tela em branco. Deveria ser testada a quantidade não numérica, apresentando mensagem de erro.
Sistema 1	Aparece tela branca quando é solicitada a manutenção de alguns órgãos.	Aplicação	Funcionalidade Manter Órgão.	A tela em branco aparecia ao exibir ou excluir um órgão/unidade cujo indicativo de administração estava “null” no banco de dados (o campo não é obrigatório.
Sistema 1	Não é possível cadastrar agrupadores (não inclui agrupadores na função manter agrupador de empreendimento) e vinculação com as realizações/empreendimentos (não permite vincular agrupador a realização na função manter realização)	Esquema	Funcionalidades: Manter Agrupador de Empreendimento e Manter Realização).	O erro era uma constraint violation exception de chave primária. As sequences do banco de dados estavam “zeradas”, ou seja, em seu estado inicial de uso – o valor de start era 1.
Sistema 1	Ajuste necessário na função Manter Fonte de Recurso.	Aplicação	Funcionalidade Manter Fonte de Recurso.	Na inclusão de um registro na tabela de fonte de recurso, com exercício=2009, código=999 e nome=”RECURSO EXTRA-ORÇAMENTÁRIO”, apareceu a mensagem: “Sistema temporariamente indisponível. Aguarde alguns instantes e tente novamente”. O problema parece estar no tamanho do conteúdo dos campos, pois quando foram reduzidos, o sistema executou a inclusão. Pode ser que o campo da tela tenha um tamanho maior que no banco de dados.
Sistema 1	Problemas no cadastramento de Empenho.	Aplicação	Funcionalidade Manter Empenho do	O usuário relatou que vinha tentando manter empreendimento e o sistema apresentou tela de erro.

			Empreendimento .	
Sistema 1	Manutenção de categoria – tela em branco.	Aplicação	Funcionalidade Manter Categoria de Bem Público.	Quando solicitada a inclusão de uma Categoria de Bem Público o sistema está apresentando tela em branco.
Sistema 1	Adm Direta deve ter pelo menos pelo menos 1 empenho.	Aplicação	Funcionalidade Manter Empreendimento .	Para Empreendimentos da Administração Direta, quando a situação “Andamento” ou “Concluída” ou “Paralisada”, solicitar obrigatoriamente pelo menos 1 Empenho.
Sistema 1	Erro na consulta de Atualidade de Empreendimentos.	Esquema	Funcionalidade Manter Empreendimento .	Existem empreendimentos cadastrados que não possuem período de atualização. O problema pode estar relacionado ao mecanismo de consulta ou de inclusão de Empreendimento que não preenche a data de atualização. A data de atualização deveria estar preenchida em todos os empreendimentos.
Sistema 2	Bug na inclusão de órgãos.	Aplicação	Funcionalidade: Manter Órgão.	Ao incluir um novo órgão, o sistema não permite a confirmação da inclusão, e direciona para a tela de consulta do último órgão cadastrado. O erro ocorre pois não é feita a verificação de preenchimento de campos obrigatórios.
Sistema 2	Bug na manutenção de subórgãos.	Esquema	Funcionalidade: Manter Subórgão.	Na tela de inclusão, ao preencher os campos CEP e telefone, a inclusão não é realizada. Foram realizados testes com os demais campos, e observou-se que o bug ocorre somente quando o CEP é preenchido.
Sistema 2	Bug na consulta de Atendimentos.	Aplicação	Funcionalidade: Pesquisar Atendimento.	Ao consultar um atendimento por meio do número do atendimento, caso o número seja igual ou superior a 10000, é necessário colocar "/ano", para que a consulta seja realizada. É necessário correção para que a pesquisa funcione sem o ano para o ano corrente.
Sistema 2	Erro na exclusão de atendimentos.	Aplicação	Funcionalidade: Manter Atendimento.	A exclusão não é permitida, pois é solicitada a criação de uma carta de resposta. Ao criar a carta de resposta, a mesma não está sendo gravada, não permitindo assim a exclusão.
Sistema 2	Bug na consulta de encaminhamentos.	Aplicação	Funcionalidade: Consultar Encaminhamento .	Na consulta Atendimento -> Registro de Encaminhamento -> Consulta/Manutenção, ao pesquisar o status TODOS ou PENDENTE é exibida uma tela para salvar o arquivo php, e não exibe a tela de resultado da consulta.
Sistema 2	Resposta via email suprimindo informações.	Aplicação	Funcionalidade: Manter Atendimento.	Ao responder a um atendimento via e-mail ocorre que se tem aspas " " no texto da resposta corta o texto a partir das aspas.
Sistema 2	Login de usuário aceitando espaços	Aplicação	Funcionalidade: Manter Usuário.	Ao incluir ou alterar um usuário, o sistema está permitindo espaços em branco no login. Deve-se inserir validação para que isso não ocorra.
Sistema 3	Erro na inclusão de Conta Bancária.	Aplicação	Funcionalidade: Manter Conta Bancária.	Não deve-se aceitar número da agência bancária igual a 0 (zero).
Sistema 3	Erro na inclusão de Conta Bancária.	Aplicação	Funcionalidade: Manter Conta	Não deve-se aceitar número da conta bancária igual a 0 (zero).

			Bancária.	
Sistema 3	Erro na exibição de valores da Programação Financeira.	Aplicação	Funcionalidade: Manter Programação Financeira.	Em alguns níveis da estrutura (ex.: 4.2 Programa de Apoio aos Arranjos Produtivos Locais no Estado do Paraná Avaliado e Auditado -> iii.Auditoria Externa), ao clicar em exibir todos os valores aparecem zerados.
Sistema 3	Erro na pesquisa de câmbio.	Aplicação	Funcionalidade: Manter Câmbio.	Ao pesquisar o período de 09/07/2008 a 10/07/2008, foram exibidos os valores dos dias 08/07 e 10/07.
Sistema 3	Erro ao alterar valores do POA (Plano Operativo Anual).	Aplicação	Funcionalidade: Manter Plano Operativo Anual.	Erro: 2008-07-12 17:14:51,776 ERROR [CadastroBasicoFacade] alterarValorCategoria org.hibernate.PropertyValueException: not-null property references a null or transient value: gov.pr.sepl.saff.pojo.ItemEstruturaPoa.itemEstrutura

APÊNDICE C – DADOS DE TESTE QUE REVELARAM DEFEITOS NAS APLICAÇÕES QUE UTILIZAM BANCO DE DADOS RELACIONAL

A Tabela C.1 apresenta um resumo dos defeitos revelados no teste das aplicações utilizadas no experimento descrito no Capítulo 4, bem como os respectivos casos de teste responsáveis por revelar cada defeito.

Tabela C.1: Dados de teste que revelaram defeitos nas aplicações que utilizam banco de dados relacional

Aplicação	Defeito	Espécie de Defeito	Localização	Defeito Revelado		Dado de Teste que Revelou o Defeito
				Teste Aplicação	Teste Esquema	
Sistema 1	Erro na quantidade da mensuração, na funcionalidade Manter Empreendimento.	Aplicação	Funcionalidade de Manter Empreendimento	sim	não	A aplicação permitiu a inserção de valores não numéricos para o campo valor_pago (tb_empreendimento).
Sistema 1	Aparece tela branca quando é solicitada a manutenção de alguns órgãos.	Esquema	tb_orgao_unidade / ind_administracao	sim	sim	A aplicação permitiu que valores nulos fossem inseridos para o campo ind_administracao.
Sistema 1	Não é possível cadastrar agrupadores (não inclui agrupadores na função manter agrupador de empreendimento) e vinculação com as realizações/empreendimentos (não permite vincular agrupador a realização na função manter realização)	Esquema	tb_agrupador2 / cod_agrupador2	sim	sim	Na inserção de novos registros na aplicação, mesmo todos sendo válidos, ocorreu erro de constraint violation exception de chave primária, pois as sequences do banco de dados estavam “zeradas”
Sistema 1	Ajuste necessário na função Manter Fonte de Recurso.	Aplicação	Funcionalidade de Manter Fonte de Recurso.	sim	não	A aplicação permitiu que valores maiores que tipo character(1) fossem inseridos para o campo tipo_fonte (tb_fonte_recurso).
Sistema 1	Problemas no cadastramento de Empenho.	Aplicação	Funcionalidade de Manter Empenho do Empreendimento.	não	não	-
Sistema 1	Manutenção de	Aplicação	Funcionalida	não	não	-

	categoria – tela em branco.		de Manter Categoria de Bem Público.			
Sistema 1	Adm Direta deve ter pelo menos pelo menos 1 empenho.	Aplicação	Funcionalida de Manter Empreendim ento.	não	não	-
Sistema 1	Erro na consulta de Atualidade de Empreendimento s.	Esquema	tb_empreendi mento / data_ultima_ alteracao	sim	sim	A aplicação permitiu que valores nulos fossem inseridos para o campo data_ultima_alteracao.
Sistema 2	Bug na inclusão de órgãos.	Aplicação	Funcionalida de: Manter Órgão.	sim	não	O sistema permitiu a execução da operação de inclusão sem a seleção de um município (valor nulo para cod_municipio de tb_orgao).
Sistema 2	Bug na manutenção de subórgãos.	Esquema	tb_suborgao / cep_sub_org ao	sim	sim	Obteve-se sucesso na inclusão de um valor com 10 caracteres (2005-10-22), sendo que na especificação do esquema o tamanho máximo deveria ser de 9 caracteres.
Sistema 2	Bug na consulta de Atendimentos.	Aplicação	Funcionalida de: Pesquisar Atendimento.	não	não	-
Sistema 2	Erro na exclusão de atendimentos.	Aplicação	Funcionalida de: Manter Atendimento.	não	não	-
Sistema 2	Bug na consulta de encaminhamentos .	Aplicação	Funcionalida de: Consultar Encaminham ento.	não	não	-
Sistema 2	Resposta via email suprimindo informações.	Aplicação	Funcionalida de: Manter Atendimento.	não	não	-
Sistema 2	Login de usuário aceitando espaços	Aplicação	Funcionalida de: Manter Usuário.	não	não	-
Sistema 3	Erro na inclusão de Conta Bancária.	Aplicação	Funcionalida de: Manter Conta Bancária.	não	não	-
Sistema 3	Erro na inclusão de Conta Bancária.	Aplicação	Funcionalida de: Manter Conta Bancária.	não	não	-
Sistema 3	Erro na exibição de valores da Programação Financeira.	Aplicação	Funcionalida de: Manter Programação Financeira.	não	não	-
Sistema 3	Erro na pesquisa de câmbio.	Aplicação	Funcionalida de: Manter Câmbio.	não	não	-
Sistema 3	Erro ao alterar valores do POA (Plano Operativo Anual).	Aplicação	Funcionalida de: Manter Plano Operativo Anual.	sim	não	A aplicação permitiu a inserção de valor nulo para o campo valor_financiado (tb_item_estrutura_poa).