Geração de Dados para Testes de Desempenho e Estresse a Partir de Testes Funcionais

Iure Fé, Alcemir Santos, Ismayle Santos, Pedro Santos Neto, Ricardo Britto

[1]Departamento de Informática e Estatística – Universidade Federal do Piauí
{iure,ismayle,alcemir,pasn,rbritto}@ufpi.edu.br

Resumo. *Embora a atividade de teste esteja bem difundida no desenvolvimento de software, a automação por completo desta atividade ainda é um grande desafio, especialmente em softwares que possuem diversas regras e manipulam muitos dados provenientes de bancos de dados. A geração de dados para o teste torna essa prática muito onerosa. Quando o assunto é a geração de dados para teste de desempenho e estresse, esse problema é ainda maior, chegando a inviabilizar o teste em alguns casos. Neste trabalho é apresentada uma ferramenta para a geração de dados para testes de desempenho e estresse reutilizando testes funcionais. Essa abordagem reduz o esforço para criação de tais testes.*

Abstract. *Nowadays, the software testing activity is used in software development, but the whole automation of this activity still a great challenge, especially in software based on complex data models. The complex data generation procedure makes software testing activity impracticable, especially in stress and performance software testing. In this work we present a tool to generate data useful to stress and performance software testing, reusing a previous functional testing to data generation procedure. The presented approach reduces the software testing creation procedure effort.*

1. Introdução

A dependência dos seres humanos por sistemas de software aumenta a cada dia. A vida humana está fortemente baseada em sistemas de computação, incluindo computadores e programas de controle. Por conta disso, a garantia da qualidade de tais sistemas teve sua importância ressaltada, uma vez que problemas nos sistemas podem representar problemas na vida humana.

O teste de software constitui um elemento crítico na garantia de qualidade, representando a revisão final da especificação, projeto e geração de código [Pressman 2002]. A realização dessa atividade é geralmente bastante onerosa durante o desenvolvimento de um sistema de software. Dependendo do tipo de sistema a ser desenvolvido, ela pode ser responsável por mais de 50% dos custos. Isso ocorre porque uma parte significativa dessas atividades ainda é executada de forma manual, com a geração de casos de teste baseada nos documentos de requisitos ou em outros documentos produzidos durante o processo de desenvolvimento.

Boa parte dos custos relacionados à atividade de teste está relacionada com a geração de dados para a realização do teste. Em alguns casos, um teste simples como

verificar a mudança de um estado de um objeto pode exigir uma quantidade grande de dados, com diferentes restrições. Por conta disso, a automação da geração de tais dados é fundamental, particularmente falando no contexto de sistemas de informações existentes no mundo real, como sistemas bancários, normalmente repletos de regras e com muitos dados envolvidos.

Os problemas citados motivaram o desenvolvimento deste trabalho que busca aumentar a eficácia das atividades de teste. Nos estudos preliminares realizados, foi possível notar que a geração de dados de teste é um dos maiores problemas para a automação da atividade. Se for levado em consideração testes de desempenho e estresse, que exigem a execução de 1.000 ou 10.000 testes ao mesmo tempo, esse problema é bem maior.

Neste trabalho é apresentada uma ferramenta para a geração de dados para testes de desempenho e estresse reutilizando testes funcionais. É importante salientar que a abordagem apresentada é inovadora e não foi explorada em trabalhos similares. A idéia explorada utiliza o teste funcional como base para tal geração, permitindo a criação de um novo conceito: geração de dados dirigida por teste.

Este trabalho é estruturado da seguinte forma: na Seção 2 serão descritos alguns trabalhos relacionados; na Seção 3 será descrita a FERRARE [Santos 2008], uma ferramenta que constrói *scripts* para testes de desempenho e estresse a partir de testes funcionais previamente executados; na Seção 4 será apresentada a ferramenta desenvolvida para a geração de dados para testes; na Seção 5 será apresentada uma aplicação da ferramenta desenvolvida; a Seção 6 conclui o trabalho, discutindo, entre outros aspectos, os trabalhos futuros.

2. Trabalhos Relacionados

Na literatura não existem trabalhos que apresentem uma solução eficiente para a geração de dados para testes de desempenho e estresse. Foram encontrados alguns trabalhos de geração de dados para testes de unidade e algumas ferramentas que geram dados aleatórios, sem levar em consideração as regras de negócio do sistema sob teste.

Dentre os trabalhos relacionados à geração de dados para testes de unidade, o mais relevante apresenta a ferramenta *Korat* [Boyapati *et al* 2009]. Essa ferramenta gera dados a partir de um arquivo de configuração baseado no formalismo *Java Modeling Language* (JML).

Dentre as ferramentas que geram dados para testes de desempenho e estresse, as mais importantes são a ferramenta *open-source generatedata* [GENERATE DATA 2009] e a ferramenta proprietária *EMS data generator* [SQLMANAGER 2009]. Ambas automatizam o processo de geração de dados, mas exigem que o modelo de dados do software sob teste não possua relacionamentos e restrições complexas. Neste caso, o usuário da ferramenta teria que resolver as dependências que existirem e criar manualmente os dados das tabelas necessários.

Boa parte dos trabalhos que envolvem automação dos testes de desempenho e estresse faz isso a partir de modelos descrevendo o software. Um artefato muito utilizado como base para essa automação são as máquinas de estados finitos. O projeto AGEDIS [Hartman e Nagin 2004], inclui uma ferramenta que pode automatizar testes

de desempenho e estresse a partir da descrição de um modelo comportamental do software, utilizando máquinas de estado. Shams, Krishnamurty e Far (2006) descrevem uma abordagem baseada em modelos para o teste de desempenho em aplicações Web, utilizando máquinas de estado para modelar as dependências entre requisições e as dependências de dados de um sistema. O grande problema das abordagens utilizando máquinas de estado, descrito no relatório final do projeto AGEDIS, é que tal forma de modelagem não parece ser apropriada para representar interações entre diversos elementos, gerando um esforço muito grande e baixa inteligibilidade. Isso normalmente torna as abordagens tão complexas que não são utilizadas na práxis industrial. Além disso, tais abordagens pouco comentam sobre a geração de dados para o teste, o que leva a crer que os exemplos praticados não refletem o que existe no mundo real.

Outras abordagens para automação utilizam modelos baseados em padrões de mercado, como a UML. Garousi, Briand e Labiche (2006) desenvolveram uma metodologia de teste de estresse em sistemas distribuídos baseada em modelos UML estendidos com informações sobre tempo e análise de controle de fluxo em diagramas de seqüência. O problema nessa abordagem é propor a criação de mais artefatos durante o desenvolvimento, gerando ainda mais trabalho e "burocracia", uma vez que a tarefa de mantê-los consistentes e atualizados não é trivial. Pouco é discutido sobre a geração de dados no trabalho, de forma similar aos trabalhos citados anteriormente.

3. FERRARE

A FERRARE - **FERR**amenta de **A**utomação de testes de **R**equisitos de desempenho e **E**stresse – [Santos *et al.* 2008] foi desenvolvida para a geração de scripts de testes de desempenho e estresse a partir de scripts de testes funcionais. Ela é dividida em dois módulos: *Extractor* e *Generator*. Um esboço do funcionamento da FERRARE pode ser vista na Figura 1.

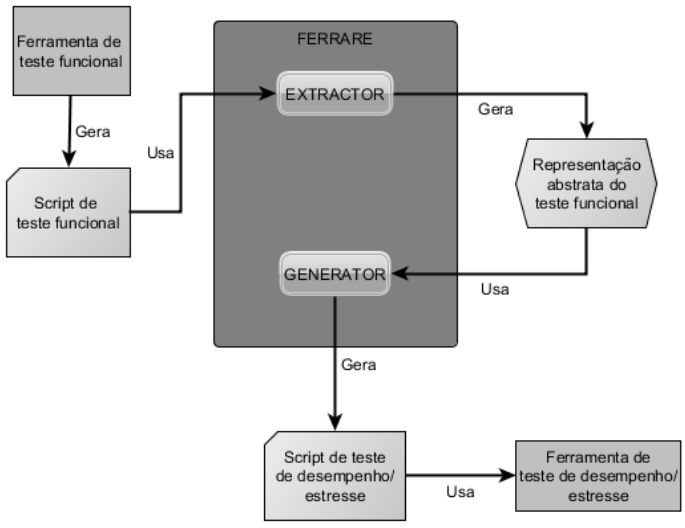

Figura 1. Esboço de funcionamento da FERRARE

A FERRARE foi concebida para trabalhar com qualquer ferramenta de teste funcional e qualquer ferramenta de teste de desempenho e estresse, desde que seja criado o extrator e gerador para as ferramentas desejadas, conforme será discutido mais adiante.

Atualmente, a FERRARE trabalha com as ferramentas de testes funcionais Selenium IDE e Canoo Web Test , além das ferramentas de teste de desempenho e estresse Apache JMeter e WebLoad . Isso significa que ela gera testes de desempenho e estresse para serem executados no JMeter ou WebLoad, a partir de testes funcionais feitos com o Selenium IDE ou Canoo WebTest (ferramentas de entrada). Uma descrição completa do funcionamento da FERRARE, incluindo detalhamento sobre testes funcionais, de desempenho e estresse pode ser consultada em Santos *et al.* (2008).

O módulo *Extractor* é responsável pela extração das informações contidas no *script* de teste funcional. Isso inclui a identificação das ações relacionadas ao teste (procedimento de teste) e dos dados de entrada, saídas esperadas e demais condições do teste (caso de teste). A extração gera uma representação abstrata do teste funcional, independente de tecnologia.

O módulo *Generator* é responsável pela geração do teste de desempenho com base nas informações fornecidas pelo *Extractor*. Essa geração envolve a especificação de diferentes parâmetros tais como a quantidade de usuários simultâneos, tempos limites e número de máquinas utilizadas para execução dos testes.

A FERRARE gera os testes de desempenho e estresse a partir da criação de "cópias" do teste funcional, levando em consideração as restrições associadas às entradas utilizadas neste teste. Caso tenha sido utilizado um teste funcional que realiza o cadastro de um livro em uma aplicação, a FERRARE pode gerar 100 "cópias" desse teste, no formato exigido para testes de desempenho e estresse, respeitando as características dos campos, como a obrigatoriedade, tamanhos e formatos. É importante a ferramenta não realiza uma simples "cópia", uma vez que também são realizadas diversas outras ações associadas, para permitir sua execução simultânea [Santos *et al.* 2008].

Embora essa geração feita pela FERRARE auxilie bastante a geração de testes de desempenho e estresse, uma limitação atual da ferramenta é o fato de não gerar dados no mecanismo de persistência. A versão atual da ferramenta só permite a correta execução do teste se os dados necessários já estiverem no mecanismo de persistência utilizado pelo sistema sob teste. Se a execução de um teste muda o estado desse mecanismo de persistência, provavelmente todos os outros testes não serão executados com sucesso.

A necessidade de povoar o mecanismo de persistência de uma aplicação pode ser uma tarefa muito onerosa. Como exemplo, pode-se citar o caso de um sistema de empréstimo de livros. Para a realização de um empréstimo em um sistema como esse, são necessários pelo menos um usuário e um exemplar de um livro. Para a realização de um teste de desempenho e estresse que simule a realização de empréstimos de livros para 1000 usuários, exigirá muito mais dados. Se esses dados tiverem que ser gerados de forma manual, a própria realização do teste fica comprometida.

4. GENESIS

Para a maioria dos sistemas de informação, a execução de testes de desempenho e estresse exige um banco de dados devidamente povoado e com todas as dependências resolvidas, o que pode tornar complexo a preparação do ambiente de teste.

A GENESIS é uma extensão da FERRARE que tem como objetivo gerar dados para o teste de desempenho e estresse. O funcionamento da ferramenta é baseado na

replicação de dados provenientes de um teste funcional. A idéia central implementada é que a replicação dos dados de um teste funcional pode servir de base para a execução de diversos testes funcionais simultâneos, resultando em um teste de desempenho ou estresse.

É importante frisar que essa abordagem elimina a necessidade de se ter conhecimento sobre todas as restrições relacionadas ao modelo de dados e à lógica de negócio da aplicação. Como os dados são replicados a partir de uma instância do banco apta a executar um teste funcional, as réplicas também deverão manter a mesma propriedade. Essa abordagem é inovadora, pois reduz a complexidade para a geração de dados. A inovação está justamente no fato de utilizar um teste funcional, e o estado do banco de dados antes da sua execução, para tal replicação.

A GENESIS é composta por dois módulos, o *Mapper* e *DataReplicator*. Um esboço do funcionamento da ferramenta é apresentado na Figura 2 e detalhado nas próximas subseções.

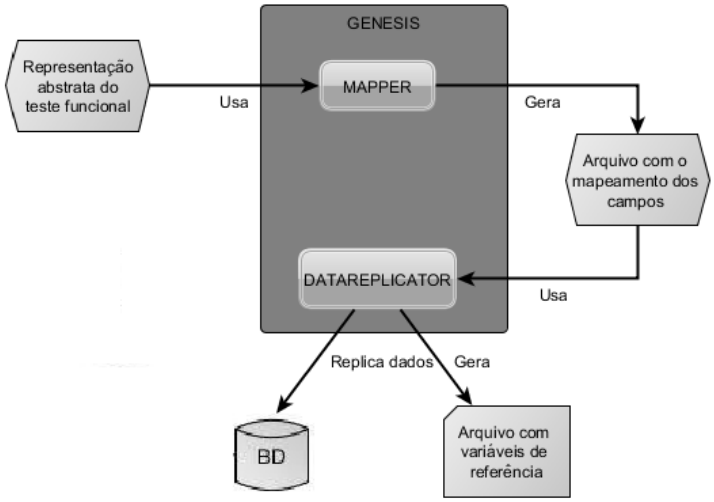

Figura 2. Esboço de funcionamento da GENESIS.

Para um melhor entendimento do funcionamento da GENESIS será utilizado um exemplo de um sistema de empréstimo de livro, cujo diagrama de classes simplificado é exibido na Figura 3. Nesse sistema hipotético, será considerado que os usuários podem tomar emprestados até dois livros, desde que eles não sejam cativos. Além disso, cada livro pode ter diversos exemplares. Cada empréstimo representa o relacionamento entre o usuário e um exemplar de um determinado livro.

Para a realização de um empréstimo o sistema exige por parte do usuário a informação de sua matrícula e o código do exemplar do livro desejado. Existe uma interface de usuário específica para realização dessa tarefa (Figura 4). Um dos requisitos da aplicação é que o tempo de resposta para empréstimo seja de até 5s, suportando até 1.000 usuários simultâneos. Para a verificação desse requisito é necessária a criação de um teste de desempenho simulando as condições especificadas. Isso seria exatamente um teste de desempenho. A verificação do comportamento do sistema, quando elevamos o número de usuários para 10.000, seria um teste de estresse, uma vez que ele analisa o comportamento do sistema sob situações anormais de uso.

É fácil notar que a execução de testes desse tipo exige uma grande quantidade de dados, o que dificulta a geração dos mesmos de forma manual. Vale ressaltar que o modelo de dados desse exemplo é bastante modesto em relação a sistemas reais, no entanto, deixa claro o problema a ser tratado.

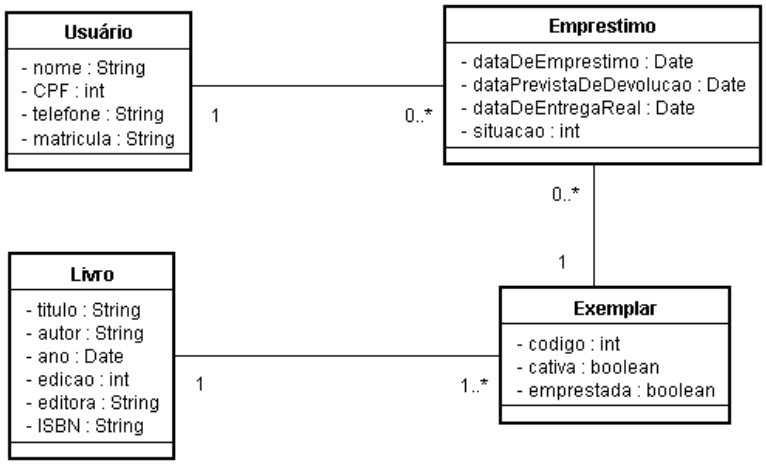

Figura 3. Diagrama de classes do sistema de empréstimo de livros.

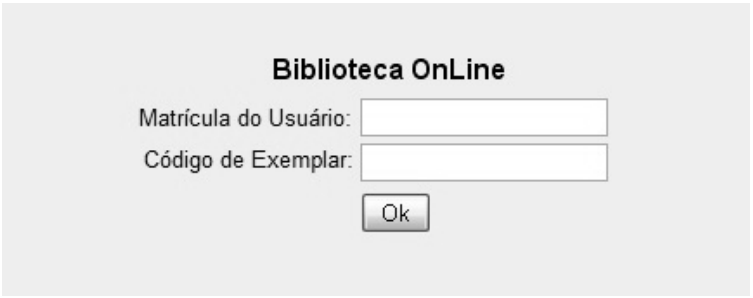

Figura 4. Tela de Empréstimo.

4.1. *Mapper*

O ponto inicial para a geração de dados a partir de um teste funcional é identificar o mapeamento entre os campos de uma interface de usuário com os respectivos atributos das classes de entidade. O módulo *Mapper* auxilia o testador a realizar o mapeamento, que é feito em duas etapas:

1. Análise da representação abstrata de um teste funcional, criada pelo módulo *Extractor* da FERRARE em busca dos nomes dos campos utilizados no teste;
2. Ligação do nome dos campos encontrados na primeira etapa com suas respectivas colunas nas tabelas do banco de dados utilizado pelo sistema sob teste (Figura 5). É importante salientar que o nome do banco de dados e demais parâmetros para conexão ao mesmo são fornecidos à ferramenta pelo testador.

Figura 5. Associação dos campos das telas às colunas das tabelas.

O *Mapper* inicia sua execução requisitando informações gerais sobre a base de dados. Essa informação está geralmente contida no *schema*, que é um conjunto de metadados do próprio banco. Depois, utilizando os nomes dos campos usados como entradas na interface de usuário, o *Mapper* busca no *schema* as tabelas existentes e suas colunas.

É importante ressaltar que, de posse das informações coletadas pelo *Mapper*, cabe ao testador especificar a correta ligação entre os nomes dos campos e as respectivas colunas nas tabelas do banco de dados.

Ao final do processo de mapeamento é gerado um arquivo contendo tal informação, que será utilizada pelo *DataReplicator* para continuidade do processo de geração de dados.

4.2. *DataReplicator*

Utilizando a informação gerada pelo *Mapper*, o módulo *DataReplicator* inicia o processo de replicação, interagindo diretamente com o banco de dados do sistema sob teste. É importante salientar que para o processo de replicação ser bem sucedido, nas tabelas mapeadas pelo *Mapper* deve existir um registro matriz (o registro existente no banco, utilizado previamente no teste funcional). Logicamente, se não houver nenhum registro em uma tabela mapeada, certamente isso indica que para a execução do teste não é necessário haver registros na tabela do banco e dados.

O processo de replicação é executado de acordo com a seguinte sequência de ações:

1. Obter um mapeamento contendo campo, coluna do banco de dados e valor utilizado no teste funcional;
2. Acessar o registro contendo o valor especificado para o campo em questão. Caso o registro já tenha sido visitado, ir para o passo 4;
3. Criar uma réplica do registro, com todos os dados iguais, caso isso seja possível, ou gerando-se outros valores, de acordo com as regras associadas. Se o registro em questão possuir relacionamento com algum outro registro de outra tabela,

seguir para o passo 2, marcando o registro atual como já visitado. Se não existir relacionamento, seguir para o passo 4;

4. Adicionar em 1 (um) a quantidade de registros gerados e verificar se já foi gerado o número de cópias exigido para o teste. Se o número de cópias ainda não foi alcançado, voltar para o passo 3, utilizando como registro matriz para a nova cópia o registro criado na iteração atual.
5. Verificar se todos os mapeamentos existentes no arquivo gerado pela *Mapper* já tiveram dados gerados. Se ainda falta algum campo e entidade a ter valores gerados, seguir para o passo 2, utilizando como base o mapeamento em questão.

Conforme mencionado no passo 3, o processo de replicação cria valores diferentes apenas para os campos que possuem restrições quanto ao seu formato. O restante será sempre igual ao dado original quando possível. Para isso foi utilizado o *schema* do banco de dados (Figura 6). Ele possui tabelas sobre todos os bancos de dados do SGBD, permitindo a extração dos nomes de tabelas, colunas, privilégios e restrições sobre as colunas contidas no banco. A Figura 6 exibe algumas tabelas do *schema*; a tabela TABLES possui informações como nome e estrutura de cada tabela do banco; *COLUMNS* possui informações gerais sobre as colunas existentes no banco; *KEY_COLUMN_USAGE* contém informações sobre os relacionamentos de uma coluna com outra tabela; *CONTRAINTS* traz informações sobre campos únicos e chaves compostas. De um modo geral, todas as informações necessárias para realização da replicação foram obtidas a partir do uso do *schema*.

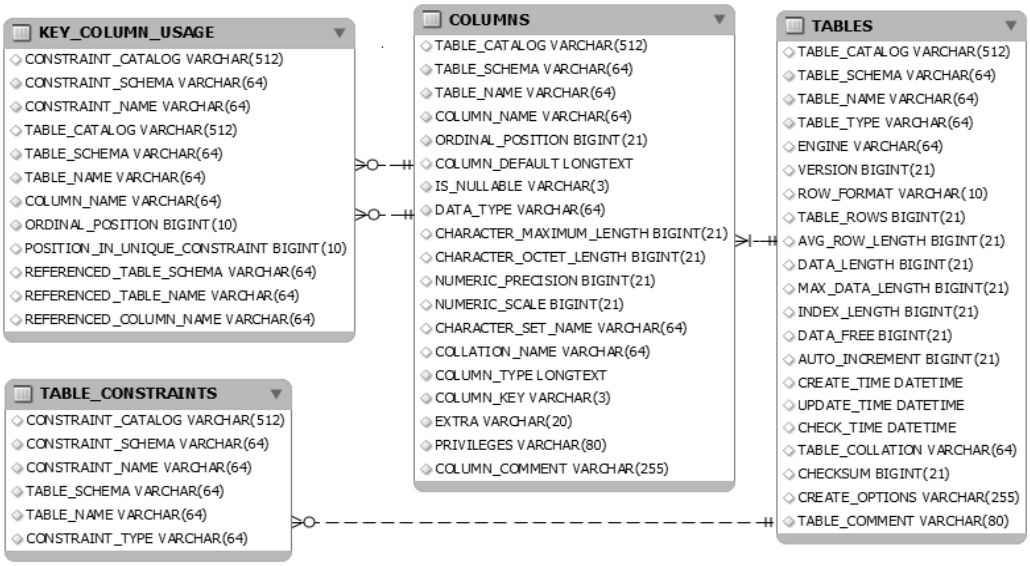

Figura 6. Exemplo de *schema*.

Eventualmente, registros de uma tabela só podem ser criados após a resolução de suas dependências. No exemplo da Figura 3, isso poderia acontecer com livro e exemplar, portanto, livro deveria ser criado antes de exemplar. A GENESIS resolve tal problema criando sempre a dependência antes. Porém essa estratégia encontra problemas

no caso de haver dependências reflexivas, que são auto-relacionamentos ou dependências cíclicas, conforme exibido na Figura 7.

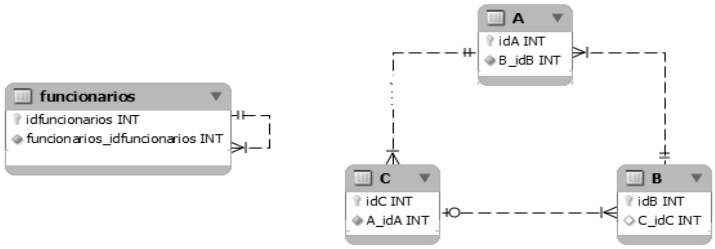

Figura 7: Exemplo de Relacionamento reflexivo à esquerda, e cíclico à direita.

Nas associações cíclicas deve haver uma das chaves estrangeiras que permita o cadastro de valor nulo. No exemplo da Figura 7, essa chave é o atributo "C_idC" na tabela B. Para resolver esse caso, o gerador guarda o valor que essa chave receberia, que é o "id" de "C" e atribui inicialmente valor nulo a "C_idC". Ao término do cadastro das tabelas A, B e C, como já existe a tabela que é a referência, o elemento que possui valor nulo para a chave estrangeira será atualizado recebendo o valor do id de C. Isto imita o procedimento que seria feito manualmente por um usuário do banco de dados ao cadastrar este tipo de relacionamento e é a forma utilizada pela GENESIS para resolver tal problema.

Um ponto importante existente no processo de replicação de dados executado pelo *DataReplicator* é a verificação que visa garantir que todos os dados relacionados às entradas do teste funcional foram replicados (passo 5). Para que isso seja possível, é necessário verificar se na replicação foram gerados dados para cada mapeamento gerado pelo *Mapper*. Em alguns casos, o modelo de dados da aplicação pode conter *loops*, de forma que o passo 5 não precise ser acionado, pois um único mapeamento pode ter originado a geração de todos os dados necessários para o teste funcional.

O módulo *DataReplicator*, e por conseguinte a GENESIS, finaliza sua execução criando um arquivo que contém todos os valores a serem utilizados pela FERRARE. Esses valores correspondem aos valores que devem ser inseridos nos campos do formulário do software sob teste.

4.3. Integração entre FERRARE e GENESIS

A Figura 8 exibe a integração entre FERRARE e GENESIS. Percebe-se na mesma figura o fluxo de dados que existe entre as ferramentas, bem como o papel de cada uma delas na configuração dos testes de desempenho e estresse.

O primeiro ponto para a automação do teste de desempenho e estresse, na abordagem utilizada por este trabalho, é a criação dos testes funcionais. Uma vez que existam tais testes, é necessário usar a FERRARE para realizar a criação do *script* para os testes de desempenho e estresse.

Antes da criação do *script*, a FERRARE cria uma representação abstrata do teste funcional, que é utilizada pela GENESIS para a replicação dos dados no banco de dados e para a geração de um arquivo de referência, que permite o acesso a esses dados. Este arquivo de referência é utilizado pela FERRARE para a finalização do *script* de teste de

desempenho e estresse, *script* esse que deverá ser utilizado por uma ferramenta que de fato execute os testes.

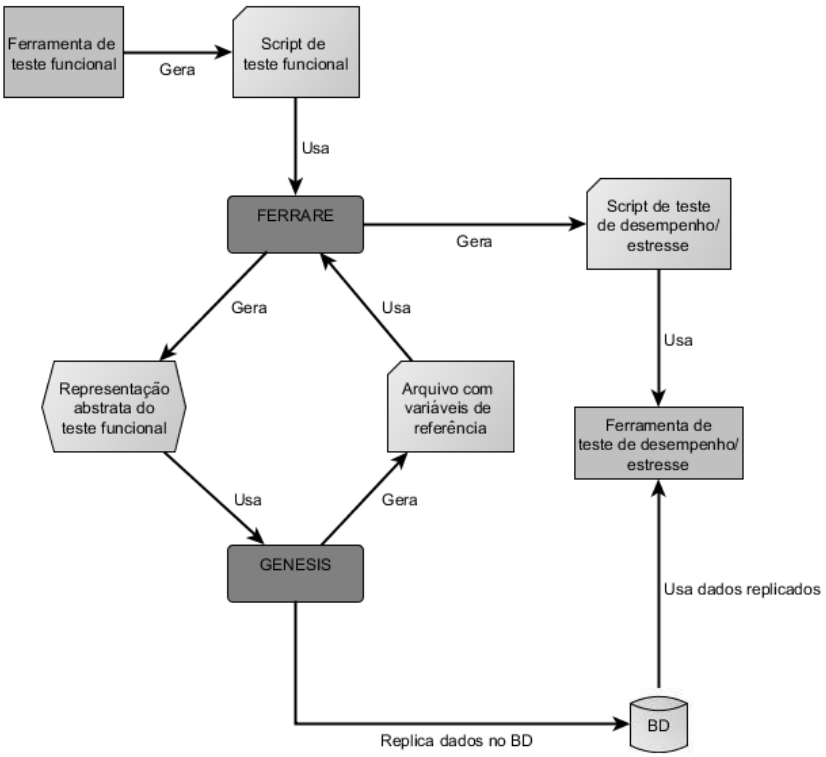

Figura 8. Integração entre GENESIS e FERRARE

4.4. Limitações da Ferramenta

No atual estado de desenvolvimento da ferramenta existem algumas limitações:

- A ferramenta funciona apenas com o sistema gerenciador de banco de dados *MySQL* [MySQL 2009], no entanto, sua extensão para outros bancos é simples;
- O mapeamento dos campos de interface de usuário e tabelas no banco de dados, que é necessário para a replicação dos dados, ainda exige interferência de um operador humano para ser executado;
- A replicação de certos tipos de dados como dados criptografados e dados geográficos não é suportada pela ferramenta.

5. Uma Aplicação da GENESIS

Para melhor demonstrar o funcionamento da ferramenta GENESIS, foi desenvolvida uma aplicação de empréstimo de livros. Tal aplicação possui o modelo de dados que pode ser visto na Figura 3.

Foi utilizado o conjunto de dados previamente cadastrado conforme exibido na Figura 9. Esses dados iniciais são usados como molde para a replicação dos registros usados no teste de desempenho e estresse.

Usuário

matricula	CPF	nome	telefone
05n10169	42086364083	John Hunter	9999999999

Exemplar

código	emprestada	tipoDaCopia
1	false	false
2	false	false

Livro

ISBN	ano	autor	edicao	editora	titulo
8571949972	2004-01-01	Alves, William Pereira	1	Érica	Fundamentos de Bancos de Dados

Figura 9: Dados pré-cadastrados no banco de dados para o teste funcional.

Supondo a tela exibida na Figura 4, foi utilizado o *Mapper* conforme exibido na Figura 5.

Possuindo o mapeamento da estrutura do banco e o modelo de dados da Figura 9, pode-se iniciar a replicação dos dados. A replicação dos dados depende do tipo de dado a ser replicado. Isso é necessário por que alguns dados não podem ser repetidos. Dessa forma, dependendo do tipo de dado a ser gerado, o algoritmo a ser utilizado pode variar.

Supondo que o CPF, matrícula do leitor, o ISBN do livro e código de exemplar exibidos na Figura 9 são únicos, devem ser criados novos valores para essas colunas e as demais podem ser replicadas com valores idênticos. Atualmente na ferramenta é possível especificar classes para a geração de dados com formato específico, como é o caso do CPF. Neste trabalho vamos ignorar isso para facilitar o entendimento.

A Figura 10 exibe exemplos de dados gerados a partir do uso da GENESIS, seguindo as prescrições mencionadas. Os dados que devem ser únicos foram gerados de forma aleatória. Os demais dados foram simplesmente replicados, o que pode ser verificado analisando conjuntamente a Figura 9 (registro matriz) e a Figura 10 (réplica).

Usuário

matricula	CPF	nome	telefone
05n10169	42086364083	John Hunter	9999999999
ueizuapVT	42086364084	John Hunter	9999999999

Exemplar

código	emprestada	tipoDaCopia
1	false	false
2	false	false
3	false	false
4	false	false

Livro

ISBN	ano	autor	edicao	editora	titulo
8571949972	2004-01-01	Alves, William Pereira	1	Érica	Fundamentos de Bancos de Dados
8571949973	2004-01-01	Alves, William Pereira	1	Érica	Fundamentos de Bancos de Dados

Figura 10: Tabelas do Banco de Dados após replicação.

6. Conclusões e Trabalhos Futuros

Neste trabalho foi apresentada uma ferramenta de geração de dados para testes de desempenho e estresse. A idéia explorada utiliza o teste funcional como base para tal

geração, permitindo a criação de um novo conceito: geração de dados dirigida por teste. A ferramenta desenvolvida, denominada GENESIS, é uma extensão da FERRARE, outra ferramenta idealizada para auxiliar a automação de testes de desempenho e estresse.

O processo de geração de dados para teste normalmente não é uma tarefa complexa. Grande parte do trabalho está em entender o modelo de dados envolvido e povoar o banco de dados seguindo tal modelo. Isso pode ser feito de forma manual, a partir do uso da aplicação, ou de forma semi-automatizada, via programas criados com base no entendimento das restrições existentes. A abordagem implementada pela GENESIS elimina a necessidade de conhecimento do modelo de dados. Isso acontece por que os dados gerados são baseados nos dados utilizados para a execução do teste funcional, dados esses que respeitam todas as restrições existentes para a aplicação.

O funcionamento da GENESIS é baseado na replicação de dados provenientes de um teste funcional. A idéia central é que a replicação dos dados de um teste funcional pode servir de base para a execução de diversos testes funcionais simultâneos. Como os dados são replicados a partir de uma instância do banco apta a executar um teste funcional, as réplicas também deverão manter a mesma propriedade.

Uma vez que o uso de teste funcional está bem difundido nas organizações, o uso da FERRARE, para criação do *script* a ser utilizado por uma ferramenta de teste de desempenho e/ou estresse, e o da GENESIS, para replicar os dados, pode contribuir para a difusão do uso de testes desempenho e/ou estresse nas empresas que desenvolvem software. Essa é outra contribuição deste trabalho.

Existem alguns trabalhos em andamento, no intuito de aprimorar as ferramentas descritas neste trabalho:

- O desenvolvimento de um mecanismo simples para permitir a geração de dados com formatos específicos;
- Introdução de inteligência na *Mapper*, automatizando completamente o mapeamento entre campos de interface de usuário e tabelas no banco de dados;
- Avaliação da ferramenta em exemplos reais mais complexos, através da realização de experimento formal, de modo a demonstrar a avaliar a produtividade do seu uso, quando comparado com a abordagem tradicional.

Agradecimentos

Este trabalho recebeu apoio financeiro do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e da empresa Infoway Consultoria e Soluções em Tecnologia para Gestão de Saúde.

Bibliografia

Boyapati, C., Khurshid, S. & Marinov, D . (2002) *Korat: Automated Testing Based on Java Predicate.* ACM International Symposium on Software Testing and Analysis (ISSTA), July 2002.

Garousi, V., Briand, L., e Labiche, Y. (2006). Traffic-aware stress testing of distributed systems based on UML models. In Proceedings of the 28th International Conference on Software Engineering (ICSE), pages 391-400, Shangai, China.

GENERATE DATA. (2009) Disponível em <http://www.generatedata.com/>.

Hartman, A. e Nagin, K. (2004). The AGEDIS tools for model based testing. In Proceedings of the International Symposium on Software Testing and Analysis (ISSTA 2004), Boston, Massachusetts, USA.

MySQL. MySQL 5.0 Reference Manual. (2009) Disponível em: <http://dev.mysql.com/doc/refman/5.0/en/information-schema.html>

Pressman, R. (2002) *Engenharia de Software*, McGraw-Hill, 5ª Edição.

Santos, I. S., Araujo, F. F. B., Bezerra, R. S. e Santos-Neto, P. (2008) *FERRARE-FERRamenta de Automação dos testes de Requisitos de desempenho e Estresse.* Anais da ERCEMAPI 2008, São Luiz.

Shams, M.; Krishnamurthy, D.; e Far, B. (2006). A model-based approach for testing the performance of web applications. In Proceedings of the 3rd International Workshop on Software Quality Assurance, pages 54-61, Portland, Oregon.

SQLMANAGER. (2009) *EMS Data Generator for MySQL*. Disponível em: <http://sqlmanager.net/products/mysql/datagenerator>.