Um Experimento em um Ambiente de Business Intelligence Industrial para melhoria da manutenção de cargas de dados

Alternative Title: An Experiment in an Industrial Business Intelligence environment to improve data loads maintenance

Juli Kelle Góis Costa[a], Igor Peterson Oliveira Santos[a], Methanias Colaço Júnior [a,b], André Vinícius R. P. Nascimento[b]

[a]Postgraduate Program in Computer Science - PROCC.
UFS – Federal University of Sergipe
São Cristóvão/SE - Brasil.
julikelle@hotmail.com, igorp.ita@hotmail.com

[b] Competitive Intelligence Research and Practice Group – NUPIC
Information Systems Departament - DSI
UFS – Federal University of Sergipe
Itabaiana/SE - Brasil.
mjrse@hotmail.com,
andreviniciusnascimento@gmail.com

RESUMO

Aplicações *Business Intelligence* (BI) e *Data Analytics* efetivas dependem de um *Data Warehouse* (DW), um repositório histórico de dados projetado para dar suporte a processos de tomada de decisão. Sem um DW eficiente, as organizações não podem extrair, em um tempo aceitável, os dados que viabilizam ações estratégicas, táticas e operacionais mais eficazes. Este artigo apresenta uma abordagem e uma ferramenta de desenvolvimento rápido de aplicações (RAD) para aumentar a eficiência e a eficácia do desenvolvimento e manutenção de programas ETL *(Extract, Transform and Load)*. Além disto, também é descrito um experimento controlado feito na indústria para analisar a eficiência e eficácia da ferramenta na manutenção de procedimentos ETL. Os resultados do experimento específico aqui apresentado indicam que a nossa abordagem pode de fato ser usada como método para acelerar e melhorar a manutenção do processo de ETL.

Palavras-Chave

Engenharia de Software Experimental, RAD, *Data Warehouse*, ETL, *Data Analytics*, *Data Warehouse*, Manutenção.

ABSTRACT

Business Intelligence (BI) and Data Analytics applications depend on an effective ETL (Extract, Transform and Load) process . This paper presents an approach and a Rapid Application Development (RAD) tool to increase efficiency and effectiveness of ETL programs development and maintenance. Furthermore, it is also described a controlled experiment conducted in industry to carefully evaluated the efficiency and effectiveness of the tool. The results indicate that our approach can indeed be used as method aimed at improving and speed up ETL process maintenance.

Categories and Subject Descriptors

D.2.3 [**Software Engineering**]: Coding Tools and Techniques – *Standards*.
D.2.9 [**Software Engineering**]: Management – *Productivity, Time estimation.*
H.2.7 [**Database Management]:** Database Administration – *Data Warehouse and repository, Security, integrity, and protection.*
H.6.3 [**Management Of Computing And Information Systems**]: Software Management - *Software development.*

General Terms

Experimentation, Management and Standardization.

Keywords

Experimental Software Engineering, RAD, Data Warehouse, ETL, Data Analytics, Data Warehouse, Maintenance.

1. INTRODUÇÃO

A importância de um ambiente de *análise de dados* está diretamente relacionada com a qualidade dos dados que são armazenados no seu banco de dados histórico, também chamado de *Data Warehouse (DW)*. Desse modo, identificar e solucionar problemas com validade, consistência e integridade dos dados representam preocupações constantes das empresas no processo de utilização de ambientes de *Data Analytics* [1].

Os problemas com qualidade de dados podem surgir em várias fases do processo de carga de dados dos Sistemas Transacionais para o DW, conhecido como ETL *(Extract, Transform and Load)*, especialmente no estágio de povoamento [2]. Uma das alternativas usadas para o povoamento de dimensões[1], (perspectivas de análise de um negócio) [3] é a criação de rotinas codificadas manualmente em linguagens procedurais que estendem a SQL [2, 3]. O objetivo é capturar os tratamentos

[1] São tabelas que qualificam os indicadores de desempenho, formadas por atributos de negócio, em sua maioria descritivos, que servem como títulos das colunas em relatórios gerenciais [3].

específicos que devem ser dados a esses artefatos e usufruir de procedimentos executando dentro do banco. Essa codificação manual, assim como os erros nas estratégias de implementação do tratamento de histórico, são apontadas, dentre outras, como principais causas para a má qualidade de dados gerados em um *Data Warehouse* [1].

O objetivo deste artigo é apresentar um dos resultados da construção e experimentação de uma ferramenta de Desenvolvimento Rápido de Aplicações (RAD) para manutenção e geração automática de código ETL, avaliando a relação existente entre sua utilização e a qualidade dos dados que são movidos, gerados e atualizados durante o processo de povoamento de um *Data Warehouse*. Este é um ambiente ímpar que necessita de maior integração e interdisciplinaridade entre as áreas de Engenharia de Software (ES) e Banco de Dados. Neste tipo de ambiente, os dados fontes já estão claramente definidos e o objetivo principal é documentar como extrair e disponibilizar esses dados dos diversos sistemas legados para os usuários finais [4].

Em trabalhos anteriores, foi realizada uma revisão, com abordagens sistemáticas, sobre as características dos procedimentos de povoamento de dimensões em um ambiente de *Data Warehouse*. A partir do levantamento dessas características, foi idealizado um esquema de dados que pudesse capturar a semântica necessária para geração de rotinas de forma automática [5]. Em seguida, foi construída a ferramenta, nomeada FGCod, para manutenção e geração automática de código em extensões SQL.

A ferramenta é um *framework* que pretende fornecer características de RAD para seus usuários programadores de processos ETL, gerando procedimentos para o SQL Server ou Oracle. Estes dois Sistemas Gerenciadores de Banco de Dados (SGBDs) são duas extensões implementadas e liberadas para uso, ou seja, é possível estender o *framework* para geração de procedimentos para outros bancos. A orquestração da execução dos procedimentos é tarefa de outro módulo da ferramenta, não abordado neste artigo. Os resultados indicaram que a FGCod pode aumentar a efetividade do processo de manutenção.

O restante do trabalho está estruturado como segue. A seção 2 apresenta a metodologia. A seção 3 a ferramenta FGCod e a breve descrição da mesma. Na seção 4, encontra-se a definição e o planejamento do experimento. Na Seção 5, é apresentada a operação do experimento. A Seção 6 contém os resultados do experimento. Na Seção 7, são apresentados os trabalhos relacionados. Por fim, a Seção 8 apresenta a conclusão e os trabalhos futuros.

2. METODOLOGIA

A metodologia utilizada para o trabalho em questão consiste, em termos de classificação, de uma pesquisa exploratória [21], de laboratório e experimental. Exploratória, porque foi realizada uma revisão de literatura, revendo conteúdos necessários para definição da abordagem. Após esta revisão, foram definidos os metadados necessários para capturar a semântica necessária para o povoamento de cada subsistema. Definidos os metadados, foi desenvolvido o módulo de manutenção da ferramenta, de forma iterativa e incremental.

Também foi classificada como de laboratório e experimental devido à execução de um experimento controlado, após o desenvolvimento do módulo, para realização da pesquisa e coleta de dados, seguindo as diretrizes de Wohlin [6], a fim de analisar a comparação da codificação na manutenção manual e automática em ambientes de apoio à decisão.

Sumarizando, o experimento foi executado em 6 etapas: Criação do ambiente de Data Warehouse; Definição dos Casos de Uso para as rotinas de povoamento; Revisão de conceitos básicos sobre rotinas de povoamento para o grupo de programadores; Treinamento da ferramenta de geração automática de código; Solicitação de manutenção nos Casos de Uso de forma manual; Solicitação de manutenção nos Casos de Uso de forma automática.

Ao término do experimento, foram analisadas as métricas que serviram de base para avaliar as hipóteses. Neste momento, foram analisadas duas métricas: a) tempo de desenvolvimento – verificar o tempo gasto para a codificação de manutenção de cada procedimento; b) número de erros de codificação – número de erros de codificação encontrados na fase de manutenção dos procedimentos. O experimento em questão está detalhado nas seções 4, 5 e 6.

3. FERRAMENTA FGCOD

A FGCod é uma ferramenta de RAD para aumentar a efetividade do desenvolvimento e manutenção de programas ETL. A mesma foi desenvolvida na linguagem Java, de forma iterativa e incremental. Ela está disponível para download em <http://fgcod.wordpress.com/>. Neste link, também se encontra o tutorial e pacote experimental para replicação. Extensões da ferramenta podem gerar códigos para outros SGBDs.

Ainda neste link, estão a definição dos metadados e a descrição detalhada de como usar a Ferramenta FGCod. Estas definições podem ser úteis para nortear abordagens semelhantes que pretendam automatizar e melhorar a geração de código, bem como para Sistemas de Informação que fazem uso de procedimentos armazenados em Bancos de Dados.

3.1 Funcionalidades da FGCod

Segue, abaixo, descrição das principais funcionalidades da FGCod.

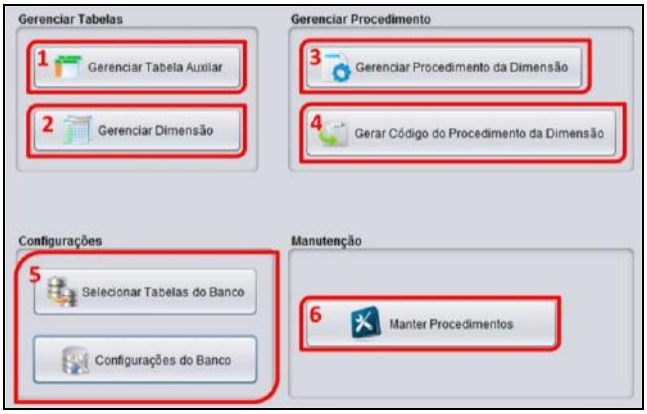

Figura 1: Tela inicial da FGCod

A Figura 1 ilustra a parte principal da tela inicial da FGCod. Em (1), pode ser realizado o gerenciamento das tabelas auxiliares (tabelas intermediárias que recebem dados dos sistemas transacionais), ou seja, são configurados o nome e esquema da tabela, assim como os atributos que a compõem, suas características e comportamentos. Em (2), acontece o gerenciamento das tabelas de dimensão, onde são configurados os

metadados que compõem uma dimensão e seus tratamentos para histórico de dados. No gerenciamento do procedimento, localizado em (3), deve ser informada a tabela auxiliar e a dimensão que irá compor o procedimento, além de criar o relacionamento entre os atributos dessas tabelas. Para finalizar, em (4), o usuário gera código SQL para o procedimento definido em (3).

Em (5) são feitas as configurações de conexão com o banco de dados e também a seleção das bases de dados, que possuem as tabelas necessárias para captura dos metadados que auxiliarão na criação do procedimento. A opção (6) é responsável por manter os procedimentos já criados anteriormente, então, nesta opção será possível selecionar um procedimento acrescentar algum dado para o mesmo, excluir ou até mesmo alterar.

4. DEFINIÇÃO E PLANEJAMENTO DO EXPERIMENTO

Nesta e nas duas próximas seções, nosso trabalho é apresentado como um processo experimental. O mesmo segue as diretrizes de Wohlin et al. em [6]. Esta seção irá focar na definição do objetivo e no planejamento do experimento.

4.1 Definição do Objetivo

O objetivo deste experimento é avaliar, por meio de um experimento controlado, o uso da geração automática de código na manutenção de rotinas de povoamento em um ambiente de Data Warehouse, através de uma ferramenta de Desenvolvimento Rápido de Aplicações (RAD), a FGCod.

O experimento terá como alvo programadores de processos ETL para ambientes de *Business Intelligence*, com alguma experiência no mercado. O objetivo foi formalizado utilizando o modelo GQM proposto por Basili [7]: **Analisar** o uso de uma ferramenta de geração de código automático, na manutenção de procedimentos SQL, para povoar um ambiente DW, **com a finalidade de** avaliar se a utilização de uma ferramenta de geração automática de código pode substituir a codificação da manutenção manual, com **respeito à** eficiência e eficácia do processo de geração automática de código, **do ponto de vista de** programadores e gestores de suporte à decisão, **no contexto de** programadores de uma empresa de *BI*.

4.2 Planejamento

4.2.1 Formulação de Hipóteses

As questões de pesquisa para o experimento que precisam ser respondidas são as seguintes: 1ª) A geração automática de código pode reduzir o tempo dos programadores no processo de manutenção de procedimentos para povoamento de um DW?; 2ª) A geração automática de código pode reduzir ou eliminar erros na codificação para manutenção de procedimentos e nos dados povoados em um DW?

Para avaliar estas questões, serão utilizadas duas métricas como variáveis dependentes: 1ª) Média de tempo, para Codificação da manutenção Manual e para Codificação da manutenção Automática; 2ª) Média de grau de criticidade de erros de povoamento, calculada considerando: (a) Registros carregados incorretamente para o DW: (1) Erro e falta de tratamento histórico para atributos das dimensões; (2) Registros duplicados; (3) Não atualização dos dados; (b) Utilização de matriz GUT (sigla para Gravidade, Urgência e Tendência) [9] para gerenciamento de problemas, através dos erros citados no item (a), bem como cálculo do grau de criticidade.

Tendo os objetivos e métricas definidas, serão consideradas as hipóteses:

HIPÓTESE 1

- H_{0tempo}: A codificação para manutenção automática e manual tem a mesma eficiência. ($\mu_{tempoCodificaçãoManual} = \mu_{tempoCodificaçãoAutomática}$).
- H_{1tempo}: A codificação para manutenção automática é mais eficiente que a codificação manual. ($\mu_{tempoCodificaçãoManual} > \mu_{tempoCodificaçãoAutomática}$).

HIPÓTESE 2

- H_{0erros}: A codificação para manutenção automática e manual tem a mesma eficácia. ($\mu_{grauErrosCodificaçãoManual} = \mu_{grauErrosCodificaçãoAutomática}$).
- H_{1erros}: A codificação para manutenção automática é mais eficaz que a codificação manual. ($\mu_{grauErrosCdificaçãoManual} > \mu_{grauErrosCodificaçãoAutomática}$).

Variáveis Independentes. As variáveis independentes do experimento são a descrição dos Casos de Uso e Processos ETL para carga de dados históricos. Para os Casos de Uso, serão consideradas as situações da prática e os tipos de tratamento histórico dos dados efetivamente usados pelos programadores selecionados, relatados como os utilizados no mercado. São os tipos 1 e 2. O tipo 1 é a abordagem de reescrita, sem salvaguardar os dados passados. No tipo 2, cria-se um novo registro para dado alterado e o registro antigo é mantido como histórico.

4.2.2 Descrição de Casos de Uso Utilizados no Experimento

4.2.2.1 UC01 – Manter Clientes

O objetivo, neste Caso de Uso, é dar manutenção em um procedimento que realiza carga de uma tabela de Clientes, para uma Dimensão do DW, nomeada DIM_Cliente. Inicialmente o procedimento só trata de atributos que não armazenam histórico, ou seja, do Tipo 1. Após manutenção, o procedimento atenderá ao armazenamento de histórico Tipo 2, para alguns atributos. Alguns atributos da dimensão não armazenarão histórico, ou seja, serão do tipo 1, outros atributos armazenarão histórico de acordo com os critérios apresentados no armazenamento de histórico do tipo 2.

O Quadro 1 ilustra qual o tipo de armazenamento de histórico de cada atributo antes da manutenção e após manutenção. As especificações do que será feito na manutenção estão descritas no Quadro 2.

4.2.2.2 UC02 – Manter Produtos

O objetivo, neste Caso de Uso, é dar manutenção em um procedimento que realiza carga da tabela de Produtos, para uma Dimensão do DW, nomeada Dim_Produto. A dimensão terá armazenamento histórico, Tipo 2, para alguns atributos. Os demais atributos serão do Tipo 1.

Quadro 1: Característica da dimensão dw.DIM_Cliente e seus atributos quanto ao armazenamento de histórico.

Nome da dimensão: dw.DIM_Cliente		
Características antes da	**Características após**	

manutenção		manutenção		
Atributos	**Tipo de Histórico**	**Atributos**	**Tipo de Histórico**	**Papel do Atributo**
data_carga		data_carga		Data de Carrega-mento
cod_cliente		cod_cliente		Chave natural
Cpf	1	Cpf	1	
nome_cliente	1	nome_cliente	1	
data_nascimento	1	data_nascimento	1	
Endereco	1	Endereco	2	
Bairro	1	Bairro	2	
Cidade	1	Cidade	2	
Estado	1	Telefone	1	
dt_inicio		dt_inicio		Data Inicial
dt_fim		dt_fim		Data Final
fl_corrente		fl_corrente		Flag Corrente

Quadro 2. Especificação do Caso de Uso 01 – Manter Clientes

[UC 01] Manter Atributos Cliente	
Descrição:	Alteração e Inclusão de Atributos no procedimento "CarregaCliente".
Ator:	DBA (Administrador de banco de dados) e Programadores
Prioridade:	Essencial
Entradas e Pré-condições:	Ter o Procedimento já criado para carregar dados dos Clientes da área de *Staging* para o DW.
Saídas e pós-condições:	- Atributos referentes à Endereço (endereço, bairro e cidade) alterados o Tipo de histórico, no procedimento. - Atributo referente à Telefone Incluso no procedimento. - Atributo referente à Estado Excluído do procedimento.
Fluxo básico:	1. O procedimento "CarregaCliente" necessita de manutenção, para suprir necessidades de Negócio. 2. Será necessário executar os seguintes Subfluxos: a. Subfluxo Alterar b. Subfluxo Incluir c. Subfluxo Excluir
Subfluxo Alterar	1. Ao ser feita carga de dados da área de Staging (tabela auxiliar) para a Dimensão Cliente (**dw.DIM_Cliente)** será necessário armazenar histórico do endereço do Cliente. 2. O Tipo de histórico necessário é o Tipo 2. 3. Os Seguintes atributos devem ser alterados para armazenamento do Tipo 2: - "endereco" - "bairro" - "cidade"
Subfluxo Incluir	1. Uma nova Coluna foi adicionada para Clientes. 2. A inclusão de um Atributo para armazenar telefone, foi incluso na Tabela de **"staging.TB_Aux_Cliente" e "dw.DIM_Cliente".** 3. O atributo referente ao telefone, tanto na **"staging.TB_Aux_Cliente"** como na **"dw.DIM_Cliente"** é "telefone". 4. O Atributo "telefone" não tem armazenamento de histórico, ou seja, é do Tipo 1.
Subfluxo Excluir	1. Uma nova Coluna foi excluída para Clientes. 2. A exclusão do Atributo para armazenar o Estado do Cliente, foi excluído da Tabela de **"staging.TB_Aux_Cliente"** e **"dw.DIM_Cliente"**. 3. O atributo referente à Estado do Cliente, tanto na **"staging.TB_Aux_Cliente"** como na **"dw.DIM_Cliente"** era "estado".

Quadro 3. Característica da dimensão dw.DIM_Produto e seus atributos quanto ao armazenamento de histórico.

Nome da dimensão: dw.DIM_Produto				
Características antes da manutenção		**Características após manutenção**		
Atributos	**Tipo de Histórico**	**Atributos**	**Tipo de Histórico**	**Papel do Atributo**
id_produto		id_produto		Surrogate Key
codigo_produto		codigo_produto		Chave Natural
codigo_barra	1	codigo_barra	1	
Descricao	2	Descricao	2	
Linha	2	Linha	1	
Valor	2	Valor	2	
Composicao	2	Fornecedor	2	
dt_inicio		dt_inicio		Data Inicial
dt_fim		dt_fim		Data Final
fl_corrente		fl_corrente		Flag Corrente

Quadro 4. Especificação do Caso de Uso 02 – Manter Produtos.

[UC 02] Manter Atributos Produto	
Descrição:	Alteração, Inclusão e Exclusão de Atributos no procedimento "CarregaProduto".
Ator:	DBA (Administrador de banco de dados) e Programadores
Prioridade:	Essencial
Entradas e Pré-condições:	Ter o Procedimento já criado para carregar dados dos Produtos da área de *Staging* para o DW.
Saídas e pós-condições:	- Atributo referente a Linha do Produto alterado o Tipo de histórico. - Atributo referente à Fornecedor Incluso. - Atributo referente à Composição do Produto Excluído.
Fluxo básico:	O procedimento "CarregaProduto" necessita de manutenção, para suprir necessidades de Negócio. Será necessário executar os seguintes Subfluxos: Subfluxo Alterar Subfluxo Incluir Subfluxo Excluir
Subfluxo Alterar	Ao ser feita carga de dados da área de Staging (**staging.TB_Aux_Produto**) para a Dimensão Produto (**dw.DIM_Produto)** não será mais necessário armazenar histórico para a Linha do Produto. O Tipo de histórico necessário é o Tipo 1. O Seguinte atributo deve ser alterado para armazenamento do Tipo 1: - "linha".
Subfluxo Incluir	Uma nova Coluna foi adicionada para Produtos. A inclusão de um Atributo para armazenar Fornecedor do Produto, foi incluído na Tabela de **"staging.TB_Aux_ Produto" e "dw.DIM_ Produto".** O atributo referente ao Fornecedor, tanto na **"staging.TB_Aux_Produto"** como na **"dw.DIM_Produto"** é "fornecedor". O Atributo "fornecedor" terá armazenamento de histórico, Tipo 2.
Subfluxo Excluir	Uma nova Coluna foi excluída para Produtos. A exclusão do Atributo para armazenar a Composição do Produto, foi excluído da Tabela de **"staging.TB_Aux_Produto" e "dw.DIM_Produto".** O atributo referente à Composição do Produto, tanto na **"staging.TB_Aux_Produto"** como na **"dw.DIM_Produto"** era "composicao".

O Quadro 3 ilustra qual o tipo de armazenamento de histórico de cada atributo antes da manutenção e após manutenção. As especificações do que será feito na manutenção estão descritas no Quadro 4.

Os processos ETL, utilizados neste trabalho, possuem dois tipos de tratamentos para a realização do experimento: Codificação Manual e Automática (utilizando a FGCod).

4.2.3 Seleção de Participantes

O processo de seleção dos participantes ocorrerá por conveniência. O colaborador escolhido será a Qualycred (www.qualycred.com), uma empresa que fornece consultoria em soluções de *Business Intelligence* para Indústria. A mesma fornecerá 8 programadores com quatro anos de experiência em outras áreas e dois anos de experiência, atuando especificamente com ETL para DW, no SGBD SQL SERVER.

4.2.4 Projeto do Experimento

Projetou-se o experimento num contexto pareado, em que um grupo avaliará ambas as abordagens: Codificação para manutenção Manual e Automática. Para entendimento da manutenção a ser feita, serão modificados Casos de Uso, os quais serão apresentados de forma bem detalhada aos programadores. O experimento será separado em dois momentos, sorteando metade da quantidade de programadores para iniciar com a opção 1 e os restantes com a opção 2, seguida da inversão das opções entre os grupos, na sequência. As duas opções são: 1º) Codificação da manutenção feita para as regras alteradas no primeiro Caso de Uso, com a Geração do Código Manual e, logo após, a Geração do Código Automático para o segundo Caso de Uso, ambas seguidas da execução do procedimento produzido; 2º) Codificação feita para regras alteradas no segundo Caso de Uso, com a Geração do Código Manual e, logo após, a Geração do Código Automático para o primeiro Caso de Uso, ambas seguidas da execução do procedimento. Desta forma, será favorecida a aleatoriedade, não priorizando o aprendizado manual ou automático. Os programadores poderão copiar e colar seus códigos já escritos na indústria.

4.2.5 Instrumentação

Foi utilizado um ambiente com as mesmas condições de trabalho dos programadores, perfazendo uma infraestrutura de DW já conhecida por todos.

Ferramentas. As ferramentas a serem utilizadas são o SQL Server 2008 e a FGCod, descrita na seção 2 deste trabalho. Será utilizado o módulo de manutenção. A versão, devido aos participantes, gera códigos SQL, na linguagem T-SQL (Transact-SQL), envolvendo comportamentos do Tipo 1, 2, para o tratamento de histórico em dimensões.

5. OPERAÇÃO DO EXPERIMENTO

5.1 Preparação

A seguir, são enumeradas as etapas de preparação para a execução do experimento.

1. *Alocação de programadores a Casos de Uso:* foi apresentado, a cada programador, um documento impresso, contendo uma descrição detalhada dos Casos de Uso que seriam utilizados por eles, em caso de eventuais dúvidas;
2. *Revisão de conceitos básicos sobre rotinas de povoamento para o grupo de programadores*: nessa fase, foi realizada uma revisão sobre as rotinas de povoamento para ambientes de DW, com os programadores selecionados;
3. *Treinamento da ferramenta FGCod – módulo manutenção*: pela facilidade de uso, foi realizado um treinamento de 30 minutos com os programadores, ministrado por uma pessoa não envolvida com o experimento, a fim de que eles pudessem se familiarizar com a ferramenta de manutenção automática de código.

5.2 Execução

Ao final das etapas anteriores, deu-se início ao experimento, com a realização do sorteio dos dois grupos e posterior execução, seguindo o que foi definido no planejamento.

A avaliação da ferramenta, ao final do experimento, por parte dos profissionais, foi positiva, visto que os mesmos comentaram que a usabilidade da ferramenta contribui para a redução do tempo na manutenção dos procedimentos.

5.2.1 Coleta de Dados

Foi calculado o tempo gasto por cada programador, com uso de um cronômetro, para manutenção manual e para manutenção automática, de cada Caso de Uso, levando em consideração o tempo para manutenção e execução. Ao término de todo o experimento, foram armazenados os procedimentos manuais e automáticos para análise de erros.

Os procedimentos mantidos pelos programadores foram avaliados individualmente, comparando-os a um procedimento padrão, considerados como um "modelo" a ser seguido, o qual já possui uma estrutura bem testada e aprovada por empresas. Para execução de cada procedimento, foi analisada a qualidade dos dados carregados para o DW, por um profissional conhecedor do negócio, identificando possíveis inconsistências existentes. Após a detecção de inconsistências, foram atribuídos pesos às mesmas, a depender do grau de criticidade apresentado. Os resultados desses dados coletados serão apresentados na seção 6.

5.3 Validação dos Dados

Foram computadas as médias do tempo de manutenção e médias de grau de criticidade de erros encontrados nos dados povoados no DW, após execução dos procedimentos.

Como auxílio para análise, interpretação e validação, foram utilizados três tipos de testes estatísticos, Teste Shapiro-Wilk, Teste T e Teste de Wilcoxon. O Teste Shapiro-Wilk foi utilizado para verificar normalidade das amostras. O Teste T foi utilizado para comparar a média de duas amostras pareadas [6] e, por fim, o Teste de Wilcoxon para comparar as médias das amostras pareadas que não obtiveram normalidade nos dados, verificando a magnitude da diferença [6].

Todos os testes estatísticos foram feitos utilizando a Ferramenta SPSS – IBM [8].

6. RESULTADOS

6.1 Análise e Interpretação de Dados

Para responder às questões, foram analisadas as seguintes variáveis dependentes: a) o tempo para manutenção de cada procedimento e; b) os erros encontrados no povoamento.

6.1.1 Tempo de Manutenção dos Procedimentos

Para responder à Questão de Pesquisa 01, os resultados relacionados ao tempo de manutenção dos procedimentos por cada participante são apresentados na Tabela 1.

Para o Caso de Uso 01 (UC01), os resultados mostram que a média de tempo dos programadores para realizarem manutenção no procedimento de forma manual foi de 52,38 minutos e de forma automática 10,38 minutos. No Caso de Uso 02 (UC02), para manutenção manual, os programadores obtiveram uma média de 11,5 minutos e, de forma automática, 5,38 minutos.

Tabela 1: Tempos de manutenções individuais por programador

	UC01		UC02	
	Manual (minutos)	**Automática (minutos)**	**Manual (minutos)**	**Automática (minutos)**
Programador1	82	13	12	6
Programador 2	59	17	6	5
Programador 3	88	21	10	9
Programador 4	34	5	22	3
Programador 5	16	3	9	3
Programador 6	22	6	10	8
Programador 7	66	8	13	4
Programador 8	52	10	12	5

Esses resultados sugerem que a manutenção de procedimentos de forma automática possui, em média, menor tempo de implementação, quando comparada com a manutenção dos mesmos procedimentos de forma manual por programadores com experiência na área. Todavia, não é possível fazer tal afirmação sem evidências estatísticas suficientemente conclusivas.

Para isto, primeiramente, definimos um nível de significância de 0.05 em todo o experimento e foi aplicado o Teste de Shapiro-Wilk, para análise da distribuição normal das amostras. O valor da variável Sig., leia-se *p-value*, para manutenção manual, foi 0.683, e 0.657, na manutenção automática. Estes valores são maiores que o nível de significância adotado. Assume-se que a distribuição dos dados, para ambas as implementações, é normal.

Sendo assim, o Teste de hipótese aplicado neste contexto será o Teste T, caracterizado como paramétrico, para amostras emparelhadas. Para isso, verificou-se um Sig. de 0.001, fortemente menor que o nível de significância adotado. Desta forma, confirmou-se a evidência de diferença entre as médias de 42,0, para o UC01.

Posteriormente, foi aplicado o Teste de Shapiro-Wilk, para análise da distribuição normal das amostras, no UC02. O valor da variável Sig., para manutenção manual, foi 0.091, e 0.378, na manutenção automática. Estes valores são maiores que o nível de significância adotado. Assume-se que a distribuição dos dados, para ambas as implementações, é normal.

Desta forma, no UC02, também foi utilizado o Teste T, para o Teste de hipótese com amostras emparelhadas. Para isso, verificou-se um Sig. de 0.018 , menor que o nível de significância de 0.05. Com isso, confirmou-se a evidência de diferença entre as médias, para o UC02.

Diante do apresentado, confirmou-se a evidência de diferença entre as médias, para o UC01 e UC02. O tempo médio de manutenção é significativamente diferente, ou seja, a hipótese (H_0), de que a manutenção automática e manual têm a mesma eficiência foi rejeitada para ambos Casos de Uso.

6.1.2 Erros de Povoamento dos Dados

Para responder à Questão de Pesquisa 02, foram comparados os erros, contabilizados levando em conta o esforço através da matriz GUT (Gravidade, Urgência e Tendência) [9] (Tabela 2), encontrados após execução dos Procedimentos SQL, para os dois Casos de Uso diferentes. Os resultados relacionados ao grau de criticidade de erros dos procedimentos por cada participante são apresentados na

Tabela 3. Estes erros foram encontrados com abordagem *blind*, com o auxílio de um usuário de negócio com conhecimento do sistema OLTP, o qual não estava a par do experimento.

Na matriz GUT, Tabela 2, são definidos os problemas que foram encontrados nos procedimentos gerados, a partir da manutenção. Além disso, é estabelecido o valor para os índices de gravidade, urgência e tendência de erros para povoamento de dados em um DW.

O primeiro problema detectado, Tabela 2, foi a presença de código no script sem estar sendo utilizado, como, por exemplo, a declaração de uma variável sem futura utilização. Isso acarreta na poluição do código, dificultando ainda mais o entendimento e manutenção do mesmo.

No segundo caso, linha 2 da Tabela 2, é identificado o erro para o tratamento de histórico do atributo de data final, com comportamento tipo 2 (armazenamento de histórico). Esse erro infringe a regra de atualização e armazenamento correto da data final para os registros que não são mais correntes ou válidos atualmente. O pior caso envolve o problema 3. A não atualização de dados, com o tratamento correto de histórico na dimensão, é uma falta grave e deve ser diagnosticada o quanto antes no processo de povoamento. Por fim, a duplicação de dados, problema 4, está atrelado à inconsistência e redundância em um ambiente de DW. Esta redundância leva a altos custos de armazenamento e acesso aos dados.

Para o Caso de Uso 01 (UC01), os resultados mostram que a média de grau de criticidade de erros dos programadores, para implementarem o procedimento de forma manual, foi 57,75, e, de forma automática, 0 (zero), ou seja, ao serem executados os procedimentos mantidos por cada programador através da ferramenta FGCod, não foram encontrados erros. No Caso de Uso 02 (UC02), para manutenção manual, foi obtida uma média de grau de criticidade de erros de 40,75. Para manutenção automática, a média também foi 0 (zero).

Pela constância da ausência de erros na manutenção automática e consequente não normalidade dos dados, já que as diferenças destes erros não se distribuem normalmente, foi aplicado o Teste de Wilcoxon como Teste de Hipótese. Esse se caracteriza como não paramétrico, para amostras emparelhadas, levando em consideração que a amostra tem um comportamento contínuo e simétrico. O teste, além de comparar a diferença das amostras, também verifica a magnitude dessa diferença.

Tabela 2: Matriz GUT

	Problemas	**G**	**U**	**T**	**Grau Crítico**
1	Código desnecessário no Script	2	1	2	4
2	Erro na atualização do atributo que representa a data final para uma dimensão tipo 2.	3	2	2	12
3	Não atualização de dados.	5	5	5	125
4	Duplicação de dados.	4	4	4	64

Tabela 3: Grau de criticidade de erros contabilizados para cada programador

	UC01 – Clientes			UC02 – Produtos		
	Problemas	Total de Grau Crítico		Problemas	Total de Grau Crítico	
		Manual	Automático		Manual	Automático
Programador1	-	0	0	-	0	0
Programador 2	-	0	0	2	12	0
Programador 3	-	0	0	2; 3; 4	173	0
Programador 4	1	27	0	2; 3	48	0
Programador 5	2	36	0	3	12	0
Programador 6	1	27	0	2; 3; 4	173	0
Programador 7	-	0	0	2; 3; 4	173	0
Programador 8	4	125	0	3; 5	76	0

Com aplicação do Teste de Wilcoxon, para o UC01, verificou-se que o Sig., com valor 0.017, é menor que o nível de significância de 0.05, ou seja, confirmou-se a evidência de diferença entre as médias de grau de criticidade de erros, para o UC01.

Diante dos resultados, confirmam-se evidências de que os erros de implementação de forma manual e automática são significativamente diferentes. Ou seja, a hipótese (H_0), de que a manutenção automática e manual têm a mesma eficácia também foi rejeitada.

6.2 Ameaças à Validade

Embora os resultados do experimento tenham se mostrado satisfatórios, o mesmo apresenta ameaças à sua validade que não podem ser desconsideradas.

Ameaças à validade interna: Como há coincidências nas declarações e funcionalidades dos procedimentos a serem criados para os dois casos de uso, o código produzido no UC01 pode ter sido utilizado como base para criação manual do procedimento no UC02 e vice-versa. Isso pode ter ajudado a diminuir o tempo de construção e erros para o UC02, mais complexo. No entanto, a ferramenta mostrou-se mais eficaz também para este caso. De fato, são casos como este que pretendemos beneficiar, dado que na prática, conforme relatado pela empresa colaboradora, são os casos que apresentam mais problemas.

Ainda que os participantes tenham sido treinados a utilizarem a ferramenta, os mesmos não a utilizam diariamente. Essa falta de contato constante com a mesma pode ter afetado os resultados, os quais poderiam ser ainda melhores, pró-ferramenta. O treinamento da ferramenta foi realizado logo no início do experimento, considerando um fenômeno estudado pela psicologia denominado *Demand Characterization*, o qual considera que um artefato experimental pode ter uma interpretação, pelos participantes, do propósito do experimento. Isto pode levar à mudança de comportamento inconsciente, para se adaptar a esta interpretação [10]. De acordo com este conceito, este treinamento poderia ter prejudicado o andamento do experimento, mas, para mitigar este fator, pode-se dizer que foram utilizadas pelo menos duas abordagens diferentes: *The More The Merrier* e *Unobtrusive Manipulations and Measures* [10]. Respectivamente, na primeira, para evitar o viés, com um único experimentador, o experimento contou com mais um pesquisador para conduzir o experimento e um instrutor para a ferramenta, não envolvido com a pesquisa. A segunda nos norteou a não informar quais fatores e métricas seriam avaliados, de modo que os participantes não tivessem pistas sobre a hipótese de pesquisa.

Ameaças à validade externa: o baixo número de participantes é uma ameaça, visto que pode influenciar os resultados do experimento. Mas, em Engenharia de Software, pelo custo de experimentos controlados com programadores que possuem experiência com processos do mercado, há uma exponencial dificuldade em conseguir disponibilidade empresarial para liberação destes profissionais. Para este experimento, foi proposta a utilização, privilegiada e cobiçada pela comunidade científica de Engenharia de Software, de pessoas do mercado, mitigando esta ameaça com uma cota estratificada de programadores com experiência na área de ETL para BI. Assim, assume-se que houve uma razoável representatividade perante a população de desenvolvedores de procedimentos ETL.

Ameaças à validade de construção: As Especificações dos Casos de Uso podem não ter ficado muito claras ao entendimento de algum programador. Esta ameaça foi mitigada com a leitura prévia e análise do entendimento de 3 programadores ETL.

7. TRABALHOS RELACIONADOS

Existem, atualmente, muitas ferramentas de ETL *open source* que usam um interpretador para procedimentos em XML. O *Pentaho Data Integration (Kettle)* [11], por exemplo, é uma ferramenta gráfica a qual salva seus procedimentos em XML. Alguns de seus componentes são exclusivos para cada tipo de banco de dados, dificultando alguns tratamentos de histórico. O *Talend* [12] é outra ferramenta muito utilizada, a qual gera código Java ou Perl. O *CloverETL Data Integration* [13] também tem seus recursos baseados e armazenados em XML. Além das ferramentas citadas anteriormente, existem várias outras que também utilizam o XML, como por exemplo, *DataCleaner* [14], *RedHat* [15] e *Apatar* [16].

Algumas ferramentas proprietárias também utilizam XML, é o exemplo da *IBM Information Server (Data Stage)* [17] e do *Oracle Warehouse Builder* (OWB) [18]. Ambas usam XML com suporte OLAP (*On-line Analytical Processing*) para criar o *workspace* analítico e os metadados necessários nos catálogos de banco de dados.

Em [19], é apresentada uma abordagem dirigida a modelos para a geração automática de processos ETL. Essa abordagem difere da nossa, uma vez que o objetivo é gerar processos baseados em modelos de arquitetura de ferramentas ETL já existentes. Nesse sentido, o tratamento dado a dimensões é limitado às ferramentas que se integram ao *framework*.

Em [20], é apresentado um *framework* para programação de rotinas ETL. Para avaliação, os participantes do estudo foram os próprios autores, invalidando o experimento. A falta de definição de metadados também é uma fraqueza em relação à nossa abordagem. Os metadados que definem a extração e povoamento são utilizados diretamente no código, dificultando a reutilização.

Em [22], é apresentado um experimento controlado, para verificar a efetividade da automatização de geração de código ETL.

Em resumo, até o momento, não foram encontrados trabalhos fortemente relacionados com a manutenção do tipo de carga aqui apresentado. Uma vez que, diferente das abordagens anteriores, nossa ferramenta gera efetivamente código em uma extensão da linguagem SQL, a fim de dar suporte às empresas que optam por fazer carga de dados para o DW, utilizando procedimentos em SQL.

Existem motivos pelos quais empresas optam por este tipo de carga, pois há diversos benefícios em utilizar *Stored Procedures* neste contexto, pois além de melhorar a performance e criar mecanismos de segurança entre a manipulação dos dados, pode reduzir o tráfego na rede, visto que os comandos são executados diretamente no servidor, que é isolado e dedicado somente para execução dos procedimentos. Há críticas, em sites sem embasamento científico, quanto ao uso de *stored procedures* para extrações e cargas, contudo, as críticas não se aplicam ao estado da arte em SGBDs e ao contexto de cargas para um DW aqui apresentado, no qual as cargas acontecem utilizando dados já presentes no banco, o que favorece a execução de procedimentos diretamente no SGBD.

8. CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho apresentou importantes contribuições para redução do tempo de desenvolvimento e qualidade na ES para carga de DWs, bem como fomenta a experimentação em ambiente industrial. A ferramenta passou a ser usada pelas empresas atendidas pelo parceiro escolhido, considerando experimentação e análise dos resultados realizadas, bem como perfazendo uma inovação para quem adota esta abordagem.

Vale ressaltar que a execução segura e eficiente de procedimentos em SQL diretamente no Banco de Dados é uma opção considerada por grande parte da indústria, necessitando de ferramentas de apoio a este tipo de abordagem em Engenharia de Software.

Dado este contexto e a inovação da ferramenta, a apresentação deste experimento, embasará a adoção da mesma ou a criação de uma abordagem similar por empresas que utilizam este tipo de estratégia.

Na análise dos resultados, apesar de terem sido usados programadores acostumados a escrever processos de carga, houve evidencias da redução do tempo e eliminação dos erros de codificação. Isto denota um benefício ainda maior para programadores iniciantes em ETL.

Por fim, como trabalhos futuros, pretende-se estender a abordagem para diversos idiomas SQL, já que os experimentos realizados até o momento foram somente para o T-SQL.

9. REFERÊNCIAS

[1] Singh, R. and Singh, K. A Descriptive Classification of Causes of Data Quality Problems in Data Warehouse. In: IJCSI INTERNATIONAL JOURNAL OF COMPUTER SCIENCE ISSUES. Vol. 7, Issue 3, No 2, 2010.

[2] Kimball, R., Ross, M. and Thomthwaite, W. The data warehouse lifecycle toolkit. 2. ed. Indianapolis, Indiana: Wiley Publishing Inc., 2008.

[3] Santos, V. and Belo, O. Slowly Changing Dimensions Specification a Relational Algebra Approach. In: PROC. Of Int. Conf. on Advances in Communication and Information Technology, 2011.

[4] Colaço Jr., M. Projetando sistemas de apoio à decisão baseados em data warehouse. Rio de Janeiro: Axcel Books, 2004.

[5] Santos, I. P. O., Costa, J. K. G., Nascimento, A. V. R. P. and Colaço Jr., M.. 2012 Desenvolvimento e Avaliação de uma Ferramenta de Geração Automática de Código para Ambientes de Apoio à Decisão. In: XII WTICG, 2012, Juazeiro. XII ERBASE.

[6] Wohlin, C. et al. Experimentation in Software Engineering: An introduction. USA : Kluwer Academic Publishers, 2000.

[7] Basili, V. and Weiss, D. A Methodology for Collecting Valid Software Engineering Data. In: IEEE Transactions on Software Engineering. vol.10(3): 728-738. November, 1984.

[8] SPSS, IBM software. Disponível em: <http://goo.gl/eXfcT3>. Acesso em dezembro de 2014.

[9] Marshall, Jr., I. et al. Gestão de qualidade. Rio de Janeiro: FGV, 2006.

[10] Orne, M. T. Sobre a psicologia social da experiência psicológica: Com referência particular para exigir características e suas implicações. 1962.

[11] PENTAHO, Open Source Business Intelligence. Disponível em: <http://goo.gl/aRzFVl>. Acesso em novembro de 2014.

[12] TALEND, Open Integration Solutions. Disponível em: <www.talend.com>. Acesso em novembro de 2014.

[13] CLOVERETL, Data Integration Software. Disponível em: <www.cloveretl.com>. Acesso em novembro de 2014.

[14] DATACLEANER, The premier data quality solution. Disponível em: <www.datacleaner.org/>. Acesso em novembro de 2014.

[15] REDHAT, Application Development and Integration. Disponível em: < http://www.redhat.com>. Acesso em novembro de 2014.

[16] APATAR, Connecting Data. Disponível em: <www.apatar.com/>. Acesso em novembro de 2014.

[17] IBM, Information Server (Data Stage). Disponível em: <http://goo.gl/H4j8v1>. Acesso em dezembro de 2014.

[18] ORACLE, Oracle Warehouse Builder. Disponível em: <http://goo.gl/Md4hjn>. Acesso em dezembro de 2014.

[19] Munoz, L., Mazon, J. and Trujillo, J. Automatic generation of ETL processes from conceptual models. In: Proceedings of the ACM Twelfth International Workshop on Data Warehousing and Olap. Hong Kong, China, 2009.

[20] Thomsen, C. and Pedersen, T. B. Pygrametl: a powerful programming framework for extract-transform-load programmers. In: Proceedings of the ACM Twelfth International Workshop on Data Warehousing and Olap. Hong Kong, China, 2009.

[21] Severino, A. J. Metodologia do trabalho científico. 23. ed. [s.l.] Editora, Cortez, 2008.

[22] Costa, J. K. G., Santos, I. P. O., Colaço Jr., M.. and Nascimento, A. V. R. P.. Experimentação na Indústria para Aumento da Efetividade da Construção de Procedimentos ETL em um Ambiente de Business Intelligence. SBSI – Simpósio Brasileiro de Sistemas de Informação, 2015.