Experimentação na Indústria para Aumento da Efetividade da Construção de Procedimentos ETL em um Ambiente de *Business Intelligence*

Alternative Title: Experimentation at Industrial Setting to Improve the Effectiveness of the ETL Procedures Implementation in a Business Intelligence Environment

Juli Kelle Góis Costa[a], Igor Peterson Oliveira Santos[a], André Vinícius R. P. Nascimento[b], Methanias Colaço Júnior [a,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.
andreviniciusnascimento@gmail.com, mjrse@hotmail.com

RESUMO

Aplicações de Business Intelligence (BI) 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. Ambientes de BI possuem um processo de Engenharia de Software particular, baseado em dados, para desenvolver programas de Extração, Transformação e Carga (ETL) de informações para o DW. Este artigo apresenta um método e uma ferramenta de Desenvolvimento Rápido de Aplicações (RAD) para aumentar a eficiência do desenvolvimento de programas ETL. A avaliação experimental da abordagem foi realizada em um experimento controlado feito na indústria para analisar a efetividade da ferramenta neste tipo de ambiente. Os resultados indicaram que a nossa abordagem pode ser usada como método para acelerar e melhorar o desenvolvimento de processos ETL.

Palavras-Chave

Engenharia de Software Experimental, RAD, Data Warehouse, ETL, BI.

ABSTRACT

Business Intelligence (BI) relies on Data Warehouse (DW), a historical data repository designed to support the decision making process. Without an effective Data Warehouse, organizations cannot extract the data required for information analysis in time to enable more effective strategic, tactical, and operational insights. This paper presents an approach and a Rapid Application Development (RAD) tool to increase efficiency and effectiveness of ETL (Extract, Transform and Load) programs development. An experimental evaluation of the approach is carried out in a controlled experiment that carefully evaluated the efficiency and effectiveness of the tool in an industrial setting. The results indicate that our approach can indeed be used as method aimed at improving ETL process development.

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, BI.

1. INTRODUÇÃO

A importância de um ambiente de Business Intelligence (BI) está diretamente relacionada com a qualidade dos dados que são armazenados no 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 dos sistemas de suporte à decisão [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]. As rotinas de povoamento de dimensões[1] (perspectivas de análise de um

[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].

negócio) [3], desenvolvidas na última fase do processo de carga de um DW, são, em diversas empresas, como as já atendidas pela empresa que colaborou com este artigo (vide seleção de participantes), codificadas manualmente em linguagens procedurais que estendem a SQL [2, 3]. O objetivo é capturar os tratamentos 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 o resultado da construção e experimentação de uma ferramenta de Desenvolvimento Rápido de Aplicações (RAD) para 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 Sistemas de Informação, Engenharia de Software (ES) e Banco de Dados. Neste tipo de ambiente, a preocupação principal da ES não é a lógica do processo e sim com as formas de integração e apresentação dos Sistemas de Informação. Em outras palavras, 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, para este trabalho, foi construída a ferramenta, nomeada FGCod, para 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. O processo de carga de dados se dá através da execução dos procedimentos, gerados pela ferramenta, no SQL Server ou no 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.

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

2. FERRAMENTA FGCOD

A FGCod é uma ferramenta de RAD para aumentar a efetividade do desenvolvimento 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/>. Extensões da ferramenta podem gerar códigos para outros SGBDs.

As subseções 2.1 e 2.2 apresentam a definição dos metadados e a descrição da Ferramenta FGCod, respectivamente. 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.

2.1 Definição de Metadados

Foi feito um levantamento sobre os metadados necessários para capturar a semântica utilizada na criação de procedimentos de carga de dados históricos, ou seja, quais os dados que descrevem este tipo de procedimento e podem permitir geração automática para diversas cargas, levando em consideração os diferentes tipos de armazenamento de histórico nas Dimensões de um DW. A identificação desses metadados foi realizada através de revisão sistemática e, principalmente, através da análise de procedimentos efetivos utilizados por empresas.

Após a identificação dos metadados, foram concebidos os projetos conceitual e lógico de banco de dados. Durante o projeto conceitual, foi possível criar um conjunto de abstrações, independente de tecnologia, que é capaz de capturar a semântica envolvida durante o processo de povoamento de dimensões em um ambiente de DW. As próximas seções apresentam os modelos conceituais, exibidos em partes separadas, de acordo com cada tipo de tratamento de histórico dos atributos de uma Dimensão.

2.1.1 Tipo 1

O comportamento do atributo de Tipo 1 sobrescreve o valor antigo de uma linha da Dimensão pelo valor atual da Área de *Staging*[2]. Dessa forma, o atributo sempre irá refletir o valor mais recente proveniente do ambiente operacional [3]. A área de *Staging* possui tabelas auxiliares, as quais recebem os dados provenientes do ambiente Operacional.

Em uma tabela auxiliar e para sua entidade no modelo conceitual (vide Figura 1), os metadados fundamentais são o nome e esquema, assim como os atributos que a constituem. Esses atributos são compostos por: chave natural[3]; nome do atributo; tipo dos dados; tamanho; precisão; escala; restrição de nulidade; e atributo que representa data de carregamento. Para a dimensão, os metadados são: *Surrogate Key*[4], chave natural; nome do atributo; tipo dos dados; tamanho; precisão; escala e restrição de nulidade. Este último, contido em ambas as tabelas, serve para indicar se o atributo pode ou não ser um registro nulo.

A dimensão e a tabela auxiliar são cruciais para o funcionamento do procedimento. Dessa forma, este irá conter essas duas tabelas, assim como seus relacionamentos. Afinal, é necessário identificar qual atributo da tabela auxiliar corresponde ao atributo da dimensão, para realizar as atualizações dos registros.

Para contemplar esses metadados, foi criado um esquema conceitual que pode ser visto, na Figura 1. Nesta, em (a), estão contidos os metadados para o procedimento. Por fim, em (b) e (c), estão os metadados da tabela auxiliar e dimensão, respectivamente.

2.1.2 Tipo 2

No comportamento do atributo de Tipo 2, há o tratamento e armazenamento de histórico, através da criação de novos registros [3]. Neste tipo, todos os valores de um atributo ao longo do tempo são armazenados no DW. Os metadados da dimensão que contemplam o povoamento tipo 2 são – além dos identificados no

[2] Área intermediária, entre o ambiente transacional e o DW, onde os dados precisam ser preparados para povoar, em seguida, o DW [3].

[3] Chave (identificador) das tabelas do Ambiente Operacional.

[4] Identificador das Dimensões no DW.

tipo 1 – os seguintes: data inicial, data final e flag corrente[5] (vistos em (a) da Figura 2); e comportamento do atributo. Este último surgiu a partir do momento que houve a necessidade de coletar os metadados para os diversos tipos de povoamento em uma dimensão. Logo, tornou-se necessário criar uma entidade que armazenasse o comportamento do atributo, que corresponde aos tipos 1, 2 ou 3. Essa entidade pode ser vista em (b), da Figura 2.

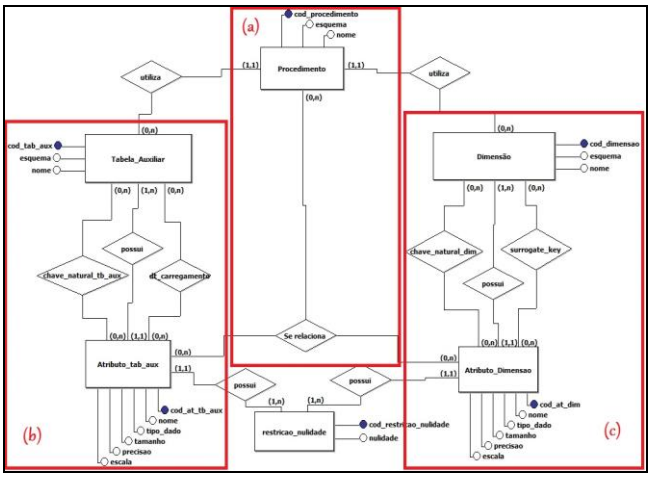

Figura 1: Esquema Conceitual dos metadados – para o procedimento, tabela auxiliar e dimensão – identificados para o povoamento de atributo tipo 1.

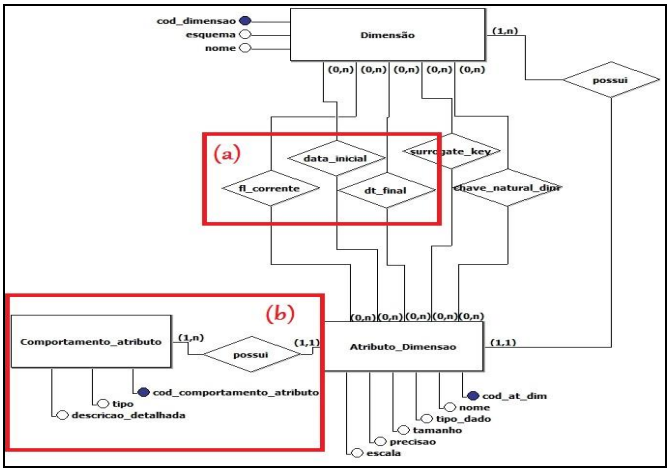

Figura 2: Esquema Conceitual dos metadados identificados, que contemplam a dimensão, para o povoamento de atributo tipo 2.

2.1.3 Tipo 3

O tratamento de histórico do Tipo 3 é utilizado quando há uma mudança e existe a necessidade de manter o histórico sem a criação de um novo registro. Essa característica é alcançada através do armazenamento do histórico em vários atributos de um mesmo registro de dados. Logo, a partir da necessidade, podem ser criadas quantas colunas forem desejadas para a dimensão[3].

Para conceber os metadados para o atributo tipo 3, seria necessário adicionar uma entidade que servisse para armazenar os atributos atual, penúltimo e antepenúltimo de uma dimensão. O nome dessa entidade (visto em vermelho na Figura 3) é a especificação do atributo tipo 3.

[5] Atributo que identifica se o registro é o mais atual ou não.

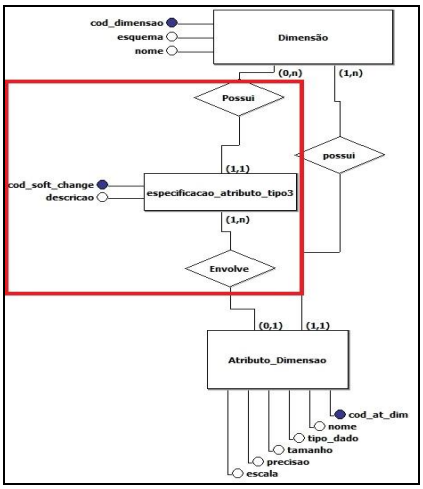

Figura 3: Esquema Conceitual dos metadados identificados, que contemplam a dimensão, para o povoamento de atributo do tipo 3.

2.2 FGCod

A Figura 4 ilustra a parte principal da tela inicial da FGCod. Em (1), pode ser realizado o gerenciamento das tabelas auxiliares, ou seja, são configurados o nome e esquema da tabela, assim como os atributos que a compõe, 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, em (3), deve ser informada a tabela auxiliar e a dimensão para compor o procedimento, e criar o relacionamento entre os atributos dessas tabelas. Para finalizar, em (4), o usuário gera código SQL para o procedimento definido em (3).

O código gerado pela FGCod oferece maior flexibilidade para tratar casos específicos de transformações durante as rotinas de povoamento. Este código pode ser facilmente modificado por um programador que conheça uma extensão da linguagem SQL, como T-SQL, PL/SQL, pgSql, etc. Além disso, conforme larga experiência do colaborador escolhido na indústria (12 anos fornecendo soluções BI), para o experimento apresentado a seguir, e [1], muitas empresas codificam manualmente suas rotinas de povoamento, gerando problemas de qualidade pela falta de uma padronização nessa codificação.

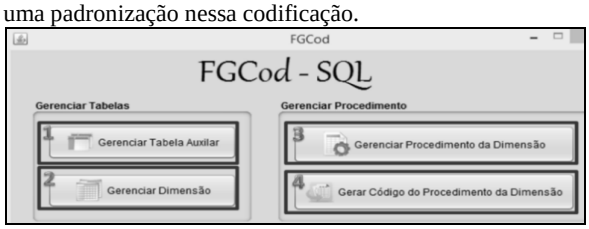

Figura 4: Tela inicial da FGCod

Vale ressaltar que, o módulo em questão da FGCod, trata apenas do processo de carga de dados. Módulos tais como modelagem gráfica do *workflow* de carga (orquestração), modelagem textual [8] e *engine* de execução não estão sendo discutidos neste artigo.

3. 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.

3.1 Definição do Objetivo

O objetivo deste experimento é avaliar, por meio de um experimento controlado, o uso da geração automática de código para 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, no mínimo, 2 anos de experiência no mercado e um ano de experiência em programação ETL. O objetivo foi formalizado utilizando o modelo GQM proposto por Basili [7]: **Analisar** o uso de uma ferramenta de geração de código automático 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 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*.

3.2 Planejamento

3.2.1 Formulação de Hipóteses

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

Para avaliar estas questões, serão utilizadas duas métricas: 1ª) Média de tempo, para Codificação Manual e para Codificação Automática; 2ª) Média de grau de criticidade de erros de povoamento.

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

HIPÓTESE 1

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

HIPÓTESE 2

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

Para ambas as hipóteses, a H_0, é a hipótese que se deseja rejeitar. Para averiguar quais das hipóteses são rejeitadas, serão consideradas as variáveis dependentes e independentes que se encontram na Figura 5.

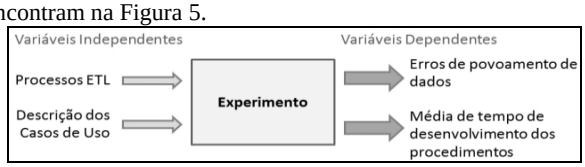

Figura 5: Variáveis Dependentes e Independentes do Experimento

3.2.1.1 Variáveis Independentes

A seguir, são descritas as variáveis independentes do experimento. Para os Casos de Uso, consideraremos as situações da prática e os tipos de tratamento histórico dos dados efetivamente usados pelos programadores selecionados e relatados pelos mesmos como os utilizados no mercado. São os tipos 1 e 2, abordados a seguir.

3.2.1.1.1 Descrição de Casos de Uso Utilizados no Experimento

UC01 – Clientes

O objetivo, neste Caso de Uso, será gerar procedimento para realizar carga da Área de *Staging*, de uma tabela de Clientes, para uma Dimensão [3], nomeada Cliente. Para este caso, a dimensão terá todos os atributos com comportamento do Tipo 1.

A Tabela 1 apresenta as características da dimensão Cliente no ambiente de DW.

Tabela 1 - Característica da dimensão dw.DIM_Cliente e seus atributos quanto ao armazenamento de histórico

Nome da dimensão	dw.DIM_Cliente	
Tipo de tratamento de histórico	Tipo 1	
Atributo	**Tipo**	**Papel do Atributo**
id_cliente		Surrogate Key
cod_cliente		Chave natural
Cpf	1	
nome_cliente	1	
data_nascimento	1	
Endereco	1	
Bairro	1	
Cidade	1	

UC02 - Produtos

Para este Caso de Uso, o objetivo será gerar procedimento para realizar cargas da Área de *Staging*, de uma tabela de Produtos, para a Dimensão Produto. Neste momento, a dimensão terá armazenamento histórico, Tipo 2, para alguns atributos. Os demais atributos serão do Tipo 1.

A Tabela 2 apresenta as características da dimensão Produto no ambiente de DW.

Tabela 2 - Característica da dimensão dw.DIM_Produto e seus atributos quanto ao armazenamento de histórico

Nome da dimensão	dw.DIM_Produto	
Tipo de tratamento de histórico	Tipo 1 e 2	
Atributo	**Tipo de Histórico**	**Papel do Atributo**
id_produto		Surrogate Key
codigo_produto		Chave Natural
codigo_barra	1	
Descricao	2	
Linha	2	
Valor	2	
dt_inicio		Data Inicial
dt_fim		Data Final
fl_corrente		Flag Corrente

3.2.1.1.2 Processos ETL

Os processos ETL, utilizados neste trabalho, possuem dois tipos de tratamentos para a realização do experimento:

1) Codificação Manual: desenvolvimento manual dos procedimentos ETL, em SQL, para carga de dados definida nos dois Casos de Uso já apresentados anteriormente.

2) Codificação Automática: geração de código SQL para os mesmos Casos de Uso, utilizando a ferramenta proposta nesse trabalho, a FGCod, que será descrita mais adiante na seção de Instrumentação.

3.2.1.2 Variáveis dependentes

Serão utilizadas duas métricas como variáveis dependentes:

- Média de tempo, para Codificação Manual e para Codificação Automática, calculada utilizando um cronômetro, considerando a média de tempo gasto para Codificação dos procedimentos.
- Média de grau de criticidade de erros de povoamento, calculada considerando:
  - 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.
  - Utilização de matriz GUT (sigla para Gravidade, Urgência e Tendência) [10] para gerenciamento de problemas através dos erros citados acima e cálculo do grau de criticidade.

3.2.2 Seleção de Participantes

O processo de seleção dos participantes ocorrerá por conveniência, perfazendo o tipo de amostragem por cota, na qual serão preservadas as mesmas características de interesse presentes na população como um todo. O colaborador a ser escolhido será a Qualycred (www.qualycred.com), uma empresa que fornece consultoria em soluções de *Business Intelligence* para Indústria. A mesma fornecerá, para a execução do experimento, oito programadores com quatro anos de experiência em outras áreas e um ano de experiência atuando especificamente com ETL para DW, no SGBD SQL SERVER.

3.2.3 Projeto do Experimento

Projetou-se o experimento num contexto pareado, em que um grupo avaliará ambas as abordagens: Codificação Manual e Automática. Para entendimento da codificação a ser feita, foram elaborados dois Casos de Uso (vistos na seção 3.2.1.1.1), os quais serão apresentados de forma bem detalhada aos programadores.

O experimento será separado em dois momentos. Contendo duas opções, sendo elas: 1º) Codificação feita para as regras apresentadas no Caso de Uso 01, com a Geração do Código Manual e, logo após, a Geração do Código Automática, ambas seguidas da execução do procedimento produzido, para povoar o *DW*, a partir dos dados provenientes da área de *Staging*; 2º) Codificação feita para regras apresentadas no Caso de Uso 02, com a Geração do Código Automático e, logo após, a Geração do Código Manual, ambas seguidas da execução do procedimento.

No primeiro momento serão sorteados 4 programadores (subgrupo 1) para iniciar com a opção 1 e os restantes (subgrupo 2), paralelamente, com a opção 2. No segundo momento, será feito o inverso, subgrupo 1 executa a opção 2 e, paralelamente, subgrupo 2 executa a opção 1. Desta forma, será favorecida a aleatoriedade, não priorizando o aprendizado manual ou automático. Contudo, para simular um ambiente real e averiguar se o uso da ferramenta fará diferença, os programadores poderão copiar e colar seus códigos já escritos na indústria. Isto inclusive mitigará o aprendizado sobre o caso de uso, em favor da ferramenta, considerando o momento que um grupo usa a ferramenta depois de ter codificado manualmente.

3.2.4 Instrumentação

O processo de instrumentação se dará inicialmente com a configuração do ambiente para o experimento e planejamento de coleta dos dados. Este será realizado em um laboratório de informática da Universidade Federal de Sergipe – UFS. Para que os participantes do experimento tenham as mesmas condições de trabalho, os computadores serão ajustados a fim de possuírem as mesmas configurações.

Seguem abaixo a tecnologia, as ferramentas a serem instaladas e artefatos.

3.2.4.1 SQL Server 2008

O Sistema Gerenciador de Banco de Dados SQL SERVER, na versão 2008, servirá como base para o armazenamento dos metadados identificados, e, consequentemente, será utilizado para armazenar os metadados da FGCod, descrita na seção 2.

3.2.4.2 Ferramenta de Geração Automática de Código Sql para ETL (FGCod)

A FGCod foi descrita na seção 2 deste trabalho. A versão a ser utilizada no experimento, devido aos participantes, gera códigos *SQL*, na linguagem T-SQL (*Transact-SQL*), envolvendo comportamentos do Tipo 1, 2 e 3 para o tratamento de histórico em dimensões.

3.2.4.3 Ambiente Criado e Artefatos Produzidos

Será definido e criado um ambiente de *Data Warehouse* com os esquemas dimensionais e áreas de *Staging* para cada programador.

4. OPERAÇÃO DO EXPERIMENTO

4.1 Preparação

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

1) *Alocação de programadores a Casos de Uso*: foi decidido, nesse ponto, que todos os programadores selecionados deveriam implementar os dois casos de uso propostos.
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*: 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 geração automática de código.

Em resumo, todos os computadores foram preparados, com as mesmas configurações, para que os programadores estivessem sobre as mesmas condições de trabalho. Além disso, 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.

4.2 Execução

Ao final das etapas anteriores, deu-se início ao experimento, seguido de acordo com o planejamento, descrito na seção 3.2.3.

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 no desenvolvimento dos procedimentos.

4.2.1 Coleta de Dados

Foi calculado o tempo gasto por cada programador, para codificação manual e para codificação Automática, de cada Caso de Uso, levando em consideração o tempo para implementação e execução. Sob supervisão, cada programador informava o término e era registrado o tempo de um cronômetro, utilizado para este fim. Ao término de todo o experimento, foram armazenados os procedimentos manuais e automáticos para análise de erros.

Em outro momento, e após o experimento, foram analisados os erros para o povoamento de dados de cada procedimento SQL (manual e automático) criado pelos programadores.

A execução dos códigos SQL gerados ocasionou o povoamento de dados. Este momento refere-se à fase de carregamento dos dados provenientes de uma área intermediária (Área de *Staging*) para o repositório central, o *DW*.

Os procedimentos criados pelos programadores foram testados individualmente, comparando-os a um procedimento padrão considerado 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, identificando possíveis inconsistências existentes, quando comparados aos dados que foram carregados pelo procedimento "modelo". Após a detecção de inconsistências, foram atribuídos pesos às mesmas, a depender do grau de criticidade apresentado.

O resultado desses dados coletados será apresentado na seção 5 deste trabalho.

4.3 Validação dos Dados

Para realização do experimento, foi considerado um fator, implementação do procedimento *SQL*, e dois tratamentos, implementação de forma manual e implementação de forma automática. Diante deste contexto, foram computadas as médias do tempo de implementação e médias de grau de criticidade de erros encontrados nos dados povoados no DW após execução dos procedimentos.

Como auxilio para análise, interpretação e validação, foram utilizados três tipos de testes estatísticos, Teste Kolmogorov-Smirnov, Teste T e Teste de Wilcoxon. O Teste Kolmogorov-Smirnov 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 [9].

5. RESULTADOS

5.1 Análise e Interpretação de Dados

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

5.1.1 Tempo de Criação dos Procedimentos

Para responder à Questão de Pesquisa 01, os resultados relacionados ao tempo de implementação dos procedimentos por cada participante são apresentados na Tabela 3.

Tabela 3 - Tempos de implementações individuais por programador

	UC01		UC02	
	Manual (minutos)	**Automática (minutos)**	**Manual (minutos)**	**Automática (minutos)**
Programador1	29	7	26	7
Programador 2	39	10	30	4
Programador 3	52	10	54	15
Programador 4	29	13	26	6
Programador 5	35	7	43	12
Programador 6	35	9	54	9
Programador 7	61	10	54	10
Programador 8	26	7	29	5

Para o Caso de Uso 01 (UC01), a média de tempo dos programadores para implementarem o procedimento de forma manual foi de 38,5 minutos e de forma automática 9,12 minutos. No Caso de Uso 02 (UC02), para implementação manual, uma média de 39,5 minutos e, de forma automática, 8,5 minutos.

Esses resultados sugerem que a criação de procedimentos de forma automática possui, em média, menor tempo de implementação, quando comparado com a criação dos mesmos procedimentos de forma manual. Com isso, a partir desta análise prévia dos dados, supõe-se que a resposta para a Questão 01 de Pesquisa seria "sim", que a geração automática de código pode aumentar a produtividade dos programadores no processo de povoamento de um DW. Mas, 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 Kolmogorov-Smirnov, para análise da distribuição normal. No UC01, em ambas amostras, implementação manual e automática, o valor da variável Sig. 0.200, leia-se *p-value*, associados ao teste de Kolmogorov-Smirnov, são maiores que o nível de significância adotado. No UC02, foi aplicado mesmo Teste obtendo valor de Sig. 0.090, para implementação manual, e 0.200, na implementação automática. Assume-se, então, que a distribuição dos dados, para ambas as implementações, é normal.

Sendo assim, o Teste de hipótese aplicado neste contexto será o Teste T. No UC01, verificou-se que o Sig. de 0.000, obtido, é fortemente menor que o nível de significância adotado. Desta forma, confirmou-se a evidência de diferença entre as médias de 29,125, para o UC01. No UC02, verificou-se um Sig. de 0.000, menor que o nível de significância de 0.05. Com isso, confirmou-se a evidência de diferença entre as médias, também para o UC02.

Diante do apresentado, confirmou-se a evidência de diferença entre as médias, para o UC01 e UC02. O tempo médio de implementação é significativamente diferente, ou seja, a hipótese (H_0), de que a codificação automática e manual têm a mesma eficiência é rejeitada, pois o teste de significância é <0.05 para ambos Casos de Uso. Consequentemente, não é rejeitada a hipótese alternativa de que a codificação automática é mais eficiente que a codificação manual.

5.1.2 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) [10] (Tabela 4), encontrados após execução dos Procedimentos SQL criados de forma manual e automática, para dois Casos de Uso diferentes. Os resultados relacionados ao grau de criticidade de erros dos procedimentos por cada participante são apresentados na **Tabela 5**. 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 4, são definidos os problemas que foram encontrados nos procedimentos gerados no experimento. Além disso, é estabelecido o valor para os índices de gravidade, urgência e tendência de erros para povoamento de dados em um *DW*.

Tabela 4 - Matriz GUT

	Problemas	**G**	**U**	**T**	**Grau Crítico**
1	Eliminação e inserção de registros (com comportamento tipo 1) já encontrados na dimensão.	3	3	3	27
2	Falta de atualização para alguns atributos com comportamento do tipo 1.	3	3	4	36
3	Erro na atualização do atributo que representa a data final para uma dimensão tipo 2.	3	2	2	12
4	Não atualização de dados.	5	5	5	125
5	Duplicação de dados.	4	4	4	64

O primeiro problema detectado, Tabela 4, foi a eliminação de antigos registros e inserção dos mesmos para os dados mais atuais provenientes da área de *Staging*, para atributos do tipo 1 (sem necessidade de guardar histórico). Esse acontecimento se torna um problema por violar os valores das *Surrogate Keys* das dimensões, visto que, se os registros já estiverem vinculados à tabela de fatos (indicadores de desempenho do negócio), estes não conseguirão relacionar-se com os registros da dimensão, por haver alterações nos valores das chaves estrangeiras.

No segundo caso, linha 2 da Tabela 4, ocorre a falha na atualização para atributos com comportamento do tipo 1. No terceiro problema, 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 4. 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 5, está atrelado à inconsistência e redundância em um ambiente de DW. Esta

redundância leva a altos custos de armazenamento e acesso aos dados.

Tabela 5 - 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

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 26,88, e, de forma automática, 0 (zero), ou seja, ao serem executados os procedimentos criados por cada programador através da ferramenta FGCod, não foram encontrados erros. No Caso de Uso 02 (UC02), para implementação manual, foi obtida uma média de grau de criticidade de erros de 83,38. Para geração automática, a média também foi 0 (zero).

Pela constância da ausência de erros na geração 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.

Com aplicação do Teste de Wilcoxon, para o UC01, verificou-se que o Sig. de 0.066 é maior que o nível de significância de 0.05, ou seja, não é possível afirmar que as médias são estatisticamente diferentes. Neste UC01, para metade dos procedimentos desenvolvidos de forma manual, não foram apresentados erros. Acredita-se que este resultado se deu pelo fato de ser um Caso de Uso simples, que apresentava poucas restrições, ficando mais perceptível para o programador detectar e implementar todas as especificações. Diferentemente do UC02, o qual apresenta maior grau de dificuldade para implementação.

Para o UC02, também foi aplicado o Teste de Wilcoxon. Verificou-se que o valor de Sig. é 0.017, menor que o nível de significância adotado. Assim, confirmou-se a evidência de diferença entre as médias de grau de criticidade de erros, para o UC02.

De forma geral, diante dos resultados, confirmam-se evidências de que os erros de implementação de forma manual e automática são significativamente diferentes, para casos de usos mais complexos. Desta forma, a hipótese (H_0) de que a codificação automática e manual têm a mesma eficácia é rejeitada apenas para o UC02.

5.2 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 [11]. 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* [11]. 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 pode ser uma ameaça, visto que pode influenciar negativamente os resultados do experimento. Esta ameaça foi mitigada com a conveniência de seleção de programadores com experiência na área de ETL para BI.

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.

6. TRABALHOS RELACIONADOS

Existem, atualmente, muitas ferramentas de ETL *open source* que usam um interpretador para procedimentos em XML. O Pentaho Data Integration (Kettle) [12], 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 [13] é outra ferramenta muito utilizada, a qual gera código Java ou Perl. O CloverETL Data Integration [14] 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 [15], RedHat [16] e Apatar [17].

Algumas ferramentas proprietárias também utilizam XML, é o exemplo da IBM Information Server (Data Stage) [18] e do Oracle Warehouse Builder (OWB) [19]. Ambas usam XML com suporte OLAP (*On-line Analytical Processing*) para criar o *workspace* analítico e os metadados necessários.

Em [20], 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 [21], é apresentado um *framework* para programação de rotinas ETL. Para avaliação, os participantes do estudo foram os próprios autores, invalidando o experimento, bem como a abordagem para o tratamento de dimensões é limitada aos Tipos 1 e 2. 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 resumo, até o momento, não foram encontrados trabalhos fortemente relacionados com o 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. No experimento realizado, muitos erros foram detectados com a execução do Procedimento SQL, gerado de forma manual (maneira utilizada atualmente por parte da Indústria), como, por exemplo: eliminação e inserção de registros já encontrados na dimensão; falta de atualização para alguns atributos; não atualização de dados; e duplicação de dados.

Existem motivos pelos quais empresas optam por este tipo de carga, pois há diversos benefícios em utilizar *Stored Procedures (SP)* 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 *SP* para extrações e cargas, contudo, elas 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.

7. CONCLUSÃO E TRABALHOS FUTUROS

Ambientes de Inteligência Aplicada aos Negócios (*Business Intelligence)* exigem dados organizacionais válidos, consistentes e íntegros. Esses itens de qualidade representam preocupações constantes para as empresas no processo de utilização dos sistemas de suporte à decisão.

Nesse contexto, o presente trabalho apresenta importantes contribuições para aumento da produtividade e qualidade na ES para carga de DWs, bem como fomenta a experimentação em ambiente industrial. A ferramenta RAD encapsula um método para acelerar e melhorar a qualidade do desenvolvimento de processos ETL baseados em SQL e passou a ser usada pelas empresas atendidas pelo parceiro escolhido, considerando experimentação e análise dos resultados, 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 ES.

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 evidências do aumento da produtividade e redução ou eliminação dos erros de codificação durante a fase de construção de ETL mais complexos para ambientes de BI. Isto denota um benefício ainda maior para programadores iniciantes em ETL.

Por fim, como trabalhos futuros, novas implementações e experimentos estão sendo feitos com o objetivo de encontrar evidências de que a utilização da ferramenta também pode melhorar a produtividade e reduzir erros de codificação durante a fase de manutenção das rotinas de povoamento.

8. REFERÊNCIAS

[1] Singh, R. and Singh, K.. 2010. *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.

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

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

[4] Colaço Jr., M.. 2004. *Projetando Sistemas de Apoio à Decisão Baseados em Data Warehouse*. Rio de Janeiro: Axcel Books.

[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*.2000. *Experimentation in Software Engineering: An introduction*. USA : Kluwer Academic Publishers.

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

[8] Mazanec, M. and Macek, O. 2012. *On General-purpose Textual Modeling Languages.* In: 12th Annual International Workshop - DATESO. April.

[9] SPSS, IBM software. Retrieved December 5, 2014 from http://goo.gl/eXfcT3.

[10] Marshall, Jr., I. *et al*. 2006. *Gestão de qualidade*. R.J.: FGV.

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

[12] PENTAHO, Open Source Business Intelligence. Retrieved November 6, 2014 from http://goo.gl/aRzFVl.

[13] TALEND, Open Integration Solutions. Retrieved November 6, 2014 from www.talend.com.

[14] CLOVERETL, Data Integration Software. Retrieved November 6, 2014 from www.cloveretl.com.

[15] DATACLEANER, The premier data quality solution. Retrieved November 7, 2014 from www.datacleaner.org.

[16] REDHAT, Application Development and Integration. Retrieved November 6, 2014 from http://www.redhat.com.

[17] APATAR, Connecting Data. Retrieved November 6, 2014 from www.apatar.com/.

[18] IBM, Information Server (Data Stage). Retrieved November 6, 2014 from http://goo.gl/H4j8v1.

[19] ORACLE, Oracle Warehouse Builder. Retrieved November 10, 2014 from http://goo.gl/Md4hjn.

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

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