CASEPRO: UMA FERRAMENTA DE APOIO AO REUSO EM BANCO DE DADOS.

REIS, Alison Carlos da Paixão
MOURA, Leandro Barros de
RIBEIRO, Vagner Cavalcanti
PARANÁ, Jayrson Sousa
CIPRIANO, Cássio
FERNANDES, Leonardo Monteiro

Resumo

O presente artigo tem por objetivo ilustrar a importância do reuso na grande área de banco de dados, conceito este outrora bastante difundido em engenharia de software; abordar o que são componentes reutilizáveis dentro de um banco de dados e como eles podem facilitar o desenvolvimento de aplicações em geral; e apresentar o desenvolvimento de uma ferramenta CASE chamada CASE PRO. Esta ferramenta tem como principal característica a criação de código de objetos automatizados, ou seja, de acordo com as tabelas dispostas em um determinado banco de dados dentro de um SGBD, é possível criar código fonte de componentes como, por exemplo, *stored procedures*, com sintaxe pré-definida. Como contribuição, demonstra-se, por meio da ferramenta geradora de código fonte, os benefícios advindos pela prática de reutilização de componentes na área de banco de dados, além de propor meios viáveis para sua disseminação de forma prática e concisa.

Palavras chave: Banco de dados, Reuso, *Stored procedures*.

CASEPRO: A TOOL TO SUPPORT THE REUSED OF DATABASE

Abstract

This article aims to illustrate the importance of reuse in the large database area, a concept that was once widely used in software engineering; Address what reusable components are within a database and how they can facilitate the development of applications in general; And present the development of a CASE tool called CASE PRO. This tool has as main characteristic the creation of code of automated objects, that is to say, according to the tables arranged in a given database inside a DBMS, it is possible to create source code of components like, for example, stored procedures, with Syntax. As a contribution, it is demonstrated, through the source code generator tool, the benefits derived from the practice of reusing components in the database area, in addition to proposing viable means for its dissemination in a practical and concise manner.

Keywords: Databases. Reuse. Stored procedures.

Introdução

Nos dias atuais, com a evolução no mundo tecnológico, a busca por soluções cada vez mais rápidas e precisas, fez com que o desenvolvimento de frameworks para auxiliar nas árduas tarefas impostas pelo mercado de trabalho se tornasse cada vez mais comum e disseminado. Tais alternativas têm contribuído de maneira promissora a fim de almejar resultados cada vez mais satisfatórios para as companhias.

Uma abordagem comumente observada na área de engenharia de software, trata do reuso de software, que tem por premissa a reutilização de artefatos de forma a contribuir para o bom aproveitamento de tempo e pode ser, da mesma forma, conceituada na área de banco de dados. A reutilização de software busca aumentar a qualidade e produtividade no desenvolvimento de software, evitando a duplicação do esforço e reaproveitando o máximo possível das experiências de projetos passados (LUCRÉDIO, 2009).

Este conceito utilizado na engenharia de software pode ser discutido em banco de dados com o uso de mecanismos disponibilizados pela linguagem SQL (*Structured Query Language*), como, por exemplo: a utilização de procedimentos armazenados, os quais são programados para executar determinadas operações que são executadas no momento em que se desejar realizar uma transação dentro do banco de dados. Portanto, a reutilização de componentes em banco de dados se configura como uma das formas de reuso em banco de dados, capaz de facilitar o desenvolvimento de aplicações em banco de dados, além de tornar o conceito mais preciso e harmonioso.

Este artigo dá um enfoque maior às *stored procedures*, justamente por ser o componente a ser explorado no aplicativo. Porém, *trigges*, *views e functions* serão conceituados, por se tratar de componentes de reuso além de que, posteriormente, poderão ser utilizados como objeto de implementação em trabalhos futuros.

O surgimento de ferramentas CASE (*Computer-Aided Software Engineering*) para o auxílio no desenvolvimento de software, tem colaborado para o sucesso de boa parte dos projetos de software. Portanto, a ferramenta CASE, Case Pro, proposta para o desenvolvimento deste trabalho, irá identificar todas as tabelas existentes nos bancos de dados dentro do SGBD gerando, por meio do dialeto SQL, as estruturas das *stored procedures*.

Com isso, a proposta da ferramenta é apresentar ao programador uma maneira simples para a criação e gerenciamento de *stored procedures*, visto que é uma funcionalidade que não fora explorada em SGBD's, tornando a ferramenta uma alternativa

para o auxílio no desenvolvimento em banco de dados.

Nesse sentido, esse artigo tem por objetivo geral debater a respeito dos principais componentes de reuso por meio da plataforma de trabalho o SGBD SQL Server, além de propor por meio de uma ferramenta desenvolvida na linguagem de programação Java® criada para a geração de *stored procedures* denominada Case Pro, uma forma automatizada de criação de componentes reutilizáveis em banco de dados.

Os objetivos listados abaixo representam as principais atividades desenvolvidas neste artigo, são elas: 1) Utilizar técnicas de reuso em banco de dados; 2) Descrever quais são os principais componentes de reuso em banco de dados; 3) Demonstrar, por meio de uma ferramenta CASE, a geração automática de componentes reutilizáveis em banco de dados.

O artigo está organizado da seguinte forma. O capítulo um aborda a respeito da revisão de literatura, bem como, sobre reuso de software, reuso em banco de dados e dos seus componentes; o capítulo dois introduz a ferramenta resultante do desenvolvimento deste artigo e aborda sua forma de uso. Em seguida apresentamos as considerações finais.

1. Revisão de Literatura

1.1. Reuso de Software

O reuso de software é um conceito antigo que perpassa através das décadas. O conceito de reuso formal surgiu na década de setenta, a aproximadamente trinta anos, e, desde então, várias tecnologias e linhas de reuso foram apresentadas. Sametinger (a*pud* SOUZA, 1997).

Após a disseminação de soluções de tecnologia da informação, tanto no contexto empresarial, quanto no cotidiano da sociedade ter-se tornado cada vez mais presente e essencial, a busca incessante de técnicas que agilizam os processos de criação de software se tornaram paralelamente, ou equivalentemente necessários.

O contexto de reuso de software engloba a grande área de engenharia de software, na qual se difunde que a criação de métodos, diagramas e demais componentes podem ser reutilizáveis no decorrer do desenvolvimento de determinado sistema computacional. Por outro lado, apesar de ter predominância na área de engenharia de software, não se pode descartar que esta técnica pode, perfeitamente, ser contextualizada no desenvolvimento de software propriamente dito.

Segundo Ezran, Morisio e Tully (2002), reuso de software é o uso de qualquer informação que um desenvolvedor pode precisar no processo de criação de um software. Portanto, é possível que linhas de código de determinada linguagem de programação possam vir a se tornar paradigmas, ou seguir determinados padrões, de forma que tais componentes possam se tornar reutilizáveis pelo desenvolvedor, diminuindo o retrabalho e colaborando na conclusão de um sistema.

2.2. Reuso em banco de Dados

Em se tratando de banco de dados, pode-se afirmar que esta perspectiva é viável, uma vez que os SGBD's modernos possuem, de forma nativa, componentes que podem ser criados pelo desenvolvedor e reutilizados em momento oportuno pelo sistema. O desenvolvimento de software baseado em componentes (DBC) é definido como uma abordagem que permite simultaneamente reduzir custos no desenvolvimento, diminuir o período de implementação necessário e aumentar a qualidade (SZYPERSKI *apud* SOUZA, 2002).

Deste modo, é evidente que boa parte da produtividade pode ser alcançada com o reuso de componentes. Ademais, a tecnologia de componentes baseia-se em unidades construídas para serem reusadas em diferentes contextos sem que haja, de preferência, alteração em sua forma (SOUZA, 2004). Em relação ao banco de dados, pode- se evidenciar, por exemplo, os seguintes componentes: *stored procedures*, *Views* e *Triggers*.

Ainda de acordo com Souza (2002), um componente é uma unidade de decomposição com suas interfaces definidas em contratos e com dependências explicitas relativas ao contexto. Para esse autor, um componente de software pode ser desenvolvido separadamente, independentemente, e, posteriormente, participar da composição.

Os próximos tópicos tratam dos componentes de reuso em banco de dados *stored procedures*, *Views* e *Triggers.*

2.3. *Stored procedures*

Stored procedures são procedimentos que ficam armazenados no SGBD a espera para que sejam chamados a fim de executar alguma tarefa especifica. São comumente usados e disseminados na indústria de software, pois propicia uma desvinculação da aplicação, de seu banco de dados; a aplicação responde por todas as regras de negócio e o SGBD se torna gerenciador das transações advindas do software principal.

Uma *stored procedure* é composta por um ou mais comandos SQL agrupados e armazenados no próprio SGBD. Normalmente cada *stored procedure* é criada para suportar a execução de determinada função ou tarefa repetitiva, conforme Huth (2002).

A sintaxe básica para criação de uma *stored procedure* para inserção de dados em uma determinada tabela no banco de dados é descrito a seguir. Onde:

‘SP_NOMETABELA_X’: será o nome da *stored procedure*;
@VARIAVEL: será o nome da variável determinada pelo autor da *stored procedure*;
‘TIPO DE DADOS’: será o tipo de dados da variável;
‘TABLE’: será a tabela atingida pelas transações;

```
Create procedure ‘sp_nometabela_insere’
@variavel1 ‘tipo de dados’
@variavel2 ‘tipo de dados’
@variavel3 ‘tipo de dados’ as insert into‘table’values
(@variável1; @variável2; @variável3
```

Ao ser executado o procedimento dentro do SGBD, será inserido na tabela todos os campos designados pelo operador do sistema por meio de uma única linha:

```
sp_nometabela_insere @variável1,
@variável2, @variável3
```

Nesse sentido, ao executar a linha acima, esta será chamada a *stored procedure* dentro do SGBD que, por sua vez, identificará qual será o procedimento a ser executado. Portanto, são identificados os campos declarados, a fim de que se possa armazenar as variáveis existentes no corpo da *stored procedure*, que, por fim, valida a inserção. O mesmo ocorre com a *stored procedure* para atualizar uma informação em uma determinada tabela do banco de dados, como pode ser observado no código para gerar a *stored procedure* a seguir:

```
Create procedure ‘sp_nometabela_atualiza’
@variavel1 ‘tipo de dados’,
@variavel2 ‘tipo de dados’,
```

@variavel3 ‘tipo de dados’ as
Update ‘table’ set ‘campo2’= @variável2, ‘campo3’=@variável3 where ‘campo1’ = ‘@variavel1’

Agora, ao ser executado o procedimento, por sua vez o SGBD irá atualizar o item dentro da tabela determinada no escopo do procedimento:

sp_nometabela_atualiza @variavel1,
@variavel2, @variavel3

O item determinado pela coluna ‘CAMPO1’ será imediatamente atualizado de acordo com as informações advindas de ‘VARIAVEL1’, ‘VARIAVEL2’ e ‘VARIAVEL3’.

É perfeitamente possível a criação de procedimentos armazenados para a exclusão de um dado em uma tabela do banco de dados.

Create procedure ‘sp_nometabela_deleta’
@variavel1 ‘tipo de dados’ as
delete ‘table’ where campo1 =
@variável1

Em linhas gerais, este procedimento fará com que seja eliminado um item na tabela em questão no momento em que é chamado pelo SGBD:

sp_nometabela_deleta ‘campo1’
Será deletado o campo do código indicado pelo usuário.

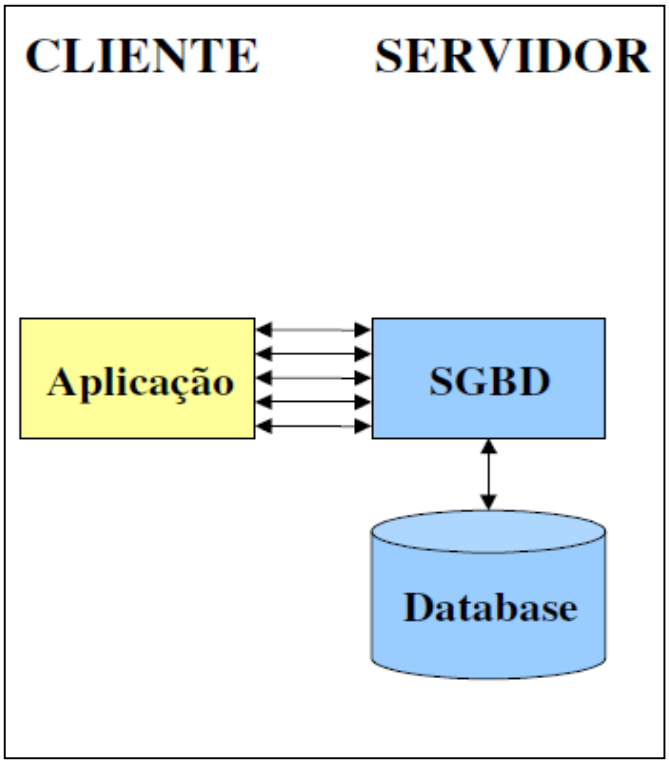


Figura 1: Aplicação convencional. Fonte: Huth (2002).

A figura 1 representa a forma como é dada a interação entre cliente e servidor sem o auxílio de *stored procedure*. A consulta que gera o produto cartesiano será armazenado no SGBD, que por sua vez, o reproduzirá ao ser chamado por uma aplicação. Por este motivo pode ser considerado um componente reutilizável.

2.3*Trigger*

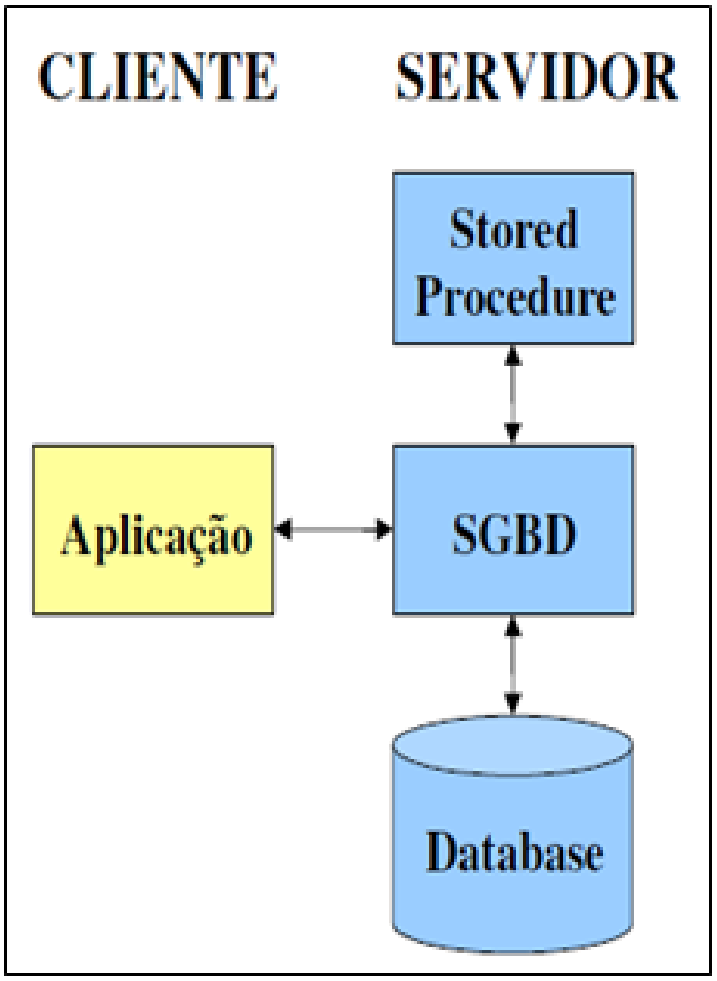


Figura 2: Utilização de stored procedure. Huth (2002)

Por sua vez, a figura 2 apresenta a forma como é realizada a interação entre servidor e cliente com o auxílio de uma stored procedure.

Outro comando importante em SQL é o CREATE TRIGGER. Em muitos casos, é conveniente especificar um tipo de ação a ser tomada quando certos eventos ocorrem e quando certas condições são satisfeitas, conforme Elmasri e Navathe (2010).

2.1 Views

Uma view em terminologia SQL, segundo Elmasri e Navathe (2010), é uma única tabela que é derivada de outras tabelas. Uma view necessariamente não existe em sua forma física; ela é considerada uma tabela virtual, ao contrário das tabelas da base, cujas tuplas sempre estão armazenadas fisicamente no banco de dados. Assim como os procedimentos armazenados, as tabelas virtuais ficam armazenadas no SGBD sendo utilizadas por conveniência da aplicação.

No dialeto SQL o comando para criar uma view é CREATE VIEW, conforme a seguir:

```
create view ‘nome_view’ as
selectc.cpf,c.nome, c.end  from cliente c, vendas v where c.cpf=v.cpf
```

Uma *Trigger* pode ser considerada um dos componentes de reuso em banco de dados, por que está diretamente vinculada a uma determinada tabela do banco, e executada ao efetuar alguma operação. Geralmente é utilizada, ou disparada, quando alguma operação de inserção, atualização ou exclusão é realizada em determinada tabela do BD, por este motivo dar-se o nome de *trigger* com a tradução para gatilho.

É bastante comum implementar *triggers* para a criação de tabelas de auditoria em banco de dados, uma vez que, ao ser implementado, o gatilho é disparado no momento em que o usuário efetua alguma inserção, atualização ou exclusão na tabela, a fim de coletar informações como: dados do usuário que efetuou a operação bem como data e hora da operação utilizada. Toma-se como exemplo o código para criação de uma hipotética tabela cliente com os campos, id, nome, sobrenome e sexo a seguir:

```
create table cliente( id int,
nome varchar(50), sobrenome varchar(50), sexo varchar(10)
```

Esta outra tabela denominada auditoria cliente que armazenará os dados do usuário, cod_clim, hora e operação:

```
create table auditoria_cliente ( usuario varchar(30),
cod_cli int, hora datetime,
operação varchar (30)
```

Será criada uma *trigger* para que, em todo momento em que ocorrer uma transação de inserção na tabela “cliente” será armazenado na tabela “auditoria_cliente”: o nome do usuário que efetuou a operação, o “id” do cliente inserido, a data e hora da inserção e, por fim, o tipo da operação, que neste caso foi uma inserção. A sua sintaxe fica da seguinte forma:

```
1.create trigger auditoria 2.on
```

3.cliente 4.for insert 5.as 6.begin

7.insert into auditoria_cliente 8.values

9.(system_user ,(select id from 10.inserted), getdate(), 'insert') 11.end

Basicamente ocorre o seguinte: A linha 1 cria a *trigger*, as linhas 2 e 3 estabelecem qual tabela será afetada, a linha 4 diz respeito a operação em que a *trigger* será tratada, a linha 5 indica que pode dar inicio ao processo que irá executá-la, a linha 6 da início a operação, e as linhas 7, 8, 9 e 10:

insert into auditoria_cliente values (system_user), (select id from inserted), (getdate, 'insert')

está o código que irá inserir na tabela auditoria_cliente, onde:

system_user: é o usuário atual do sistema;

select id from inserted: sub consulta que coleta o id do cliente existente na tabela inserted, que é uma tabela criada pelo SGBD para armazenar temporariamente os dados da tabela física afetada para que se possa ser tratada pela *trigger*; getdate(): data e hora atual;

insert: operação executada, no caso, um insert;

3. CASEPRO: Uma Ferramenta para criação de Componentes Reutilizáveis

Apesar de ter tratado dos componentes: *view* e *trigger*, neste artigo, a ferramenta Case Pro implementará apenas o conceito de*stored procedure*. Nas engenharias tradicionais os produtos são criados a partir da reutilização de produtos ou sistemas já existentes, experimentados e testados em outros sistemas (LANNA, 2009). Case Pro irá convergir para a criação de componentes reutilizáveis, artefatos estes já bastante testados e difundidos na indústria de banco de dados, como é o caso das *stored procedures*.

A ferramenta foi desenvolvida na plataforma Java®, a fim de automatizar a criação de componentes reutilizáveis em banco de dados, mais especificamente, no SGBD SQL Server, ao qual fora testado nas versões 2008 e 2012 obtendo sucesso em ambas as versões. O objetivo deste aplicativo é gerar, de acordo com os objetos existentes dentro do SGBD, um dos componentes integrantes do escopo de reuso em banco de dados, mais especificamente as *strored procedures*.

A ferramenta trabalha de forma cíclica: é necessária a autenticação no sistema, figura 3, na qual é utilizada as mesmas credenciais pertinentes aos usuários do SGBD, depois é

preciso selecionar o banco de dados, selecionar a tabela a ser manipulada, só então os metadados serão apresentados e por último, é realizado o procedimento para a geração de *stored procedure*. A Figura 3 ilustra este procedimento.

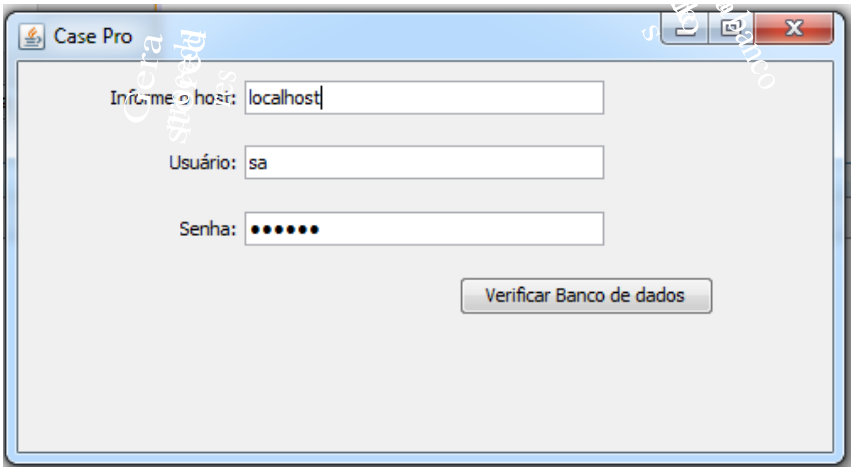


Figura 3: Tela de *login* do Case Pro com o usuário SA do Sql Server.

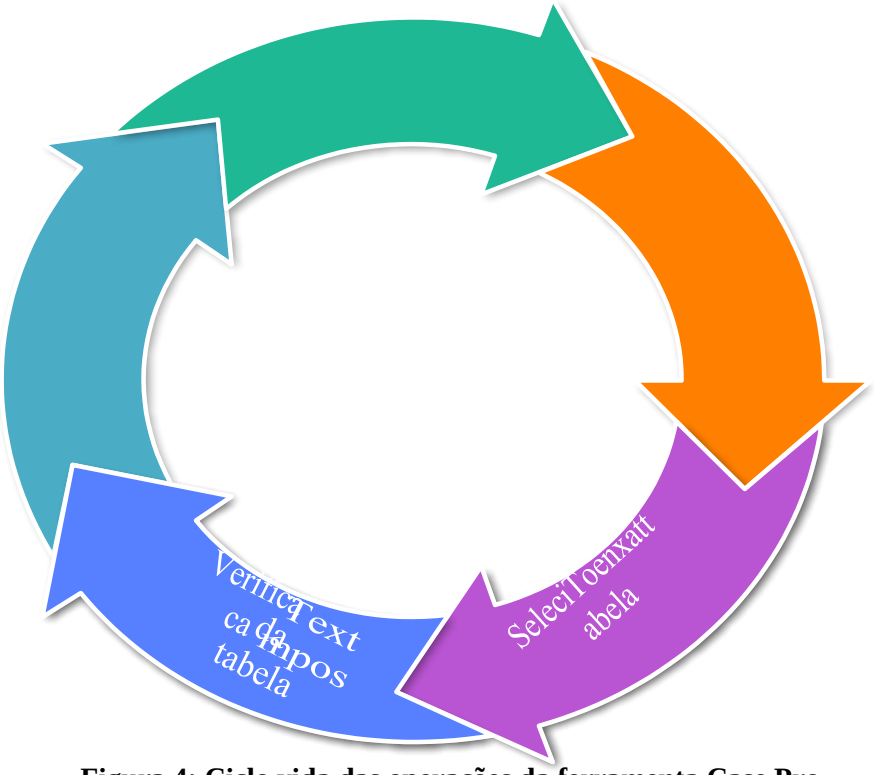


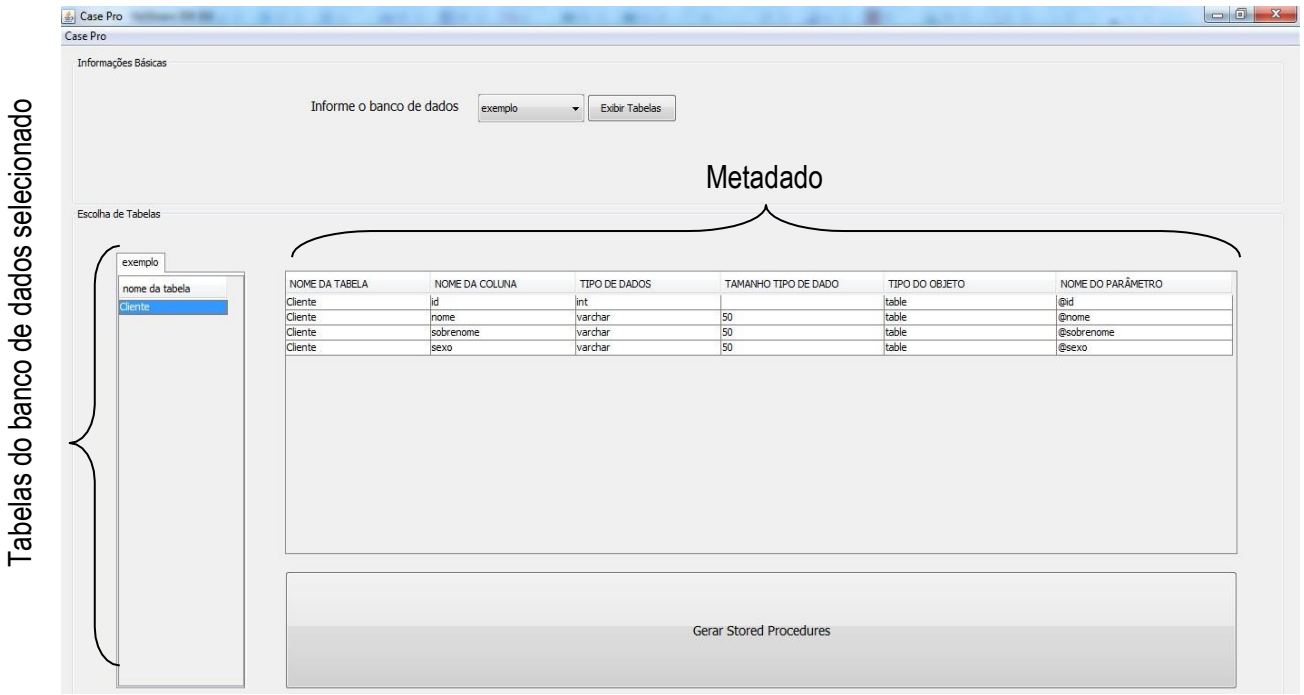
Figura 4: Ciclo vida das operações da ferramenta Case Pro.

3.1 Contexto de uso

Na autenticação, o usuário utilizará as mesmas credenciais utilizadas pelos os usuários cadastrados no SGBD, ao passo que a ferramenta valida as informações, a fim de efetivar,

desta forma, o acesso ao aplicativo. Uma vez autenticado no sistema, o usuário terá a tela principal do Case Pro, assim como é apresentado na figura 5, a qual poderá trabalhar para a implementação das *stored procedure*s. O ambiente foi desenvolvido deforma bem simples e direta, a fim de que os usuários cheguem ao seu objetivo: a geração dos componentes reutilizáveis, neste caso as *stored procedure*s.

Por meio de uma consulta SQL, são coletados os metadados da tabela selecionada a fim de que se possa preencher a *grid*, de modo que é armazenado o nome da tabela, o nome da coluna, o tipo de dados, o tamanho do tipo de dado, o tipo de objeto e o nome do parâmetro. O aplicativo irá trabalhar com esta tabela que servirá como matéria prima para a geração das *stored procedure*s. Na parte superior há um quadro para que o usuário possa selecionar o banco de dados de sua preferência, ao qual listará todos os bancos de dados existentes no SGBD, e, ao se clicar no botão "exibir tabelas", será exibido todas as tabelas existentes no banco de dados na parte esquerda da tela, bem como será preenchida a tabela de metadados, localizada no centro da aplicação. O grande botão "Gerar *stored procedure*" é o responsável por, como o próprio nome diz, gerar o código executável de criação de *stored procedure* de acordo com a sintaxe do SGBD Microsoft SQL Server.



3.2 Resultados obtidos

A Reutilização de Processos de Software vêm se destacando como uma das principais

práticas para prover a melhoria contínua de processos por aproveitar as informações (tecnológicas e gerencias) produzidas durante desenvolvimentos de software passados, com o objetivo de reduzir o esforço necessário para o desenvolvimento de um novo projeto (CARVALHO, et al, 2011).

Por trás da ferramenta existe um algoritmo que é responsável por manipular os elementos existentes na tabela de metadados e, posteriormente, através da janela "gerador de *stored procedure*", figura 6, alocar o código gerado para a criação de *stored procedure*s em abas dentro da janela, denominadas: *Insert*, *update*, *delet e select*, ambas instantaneamente preenchidas, com respectivo código necessário para gerar os procedimentos armazenados, com a sintaxe do SGBD SQL Server. É possível que, a qualquer momento, o usuário retorne a tela de *login*, para isto, é preciso que se clique no menu superior esquerdo denominado "case pro", o qual haverá um botão específico para o retorno da mesma.

Um componente de software pode ser definido como uma unidade de composição que encapsula seu projeto e implementação e oferece seus serviços ao ambiente externo por meio de interfaces bem definidas, conforme Gimenes; Huzita, 2005; Szyperski, (2002) *apud* Lanna (2009).

Os componentes são gerados a partir do algoritmo que interpreta a tabela como se fosse uma matriz com os campos dispostos em um plano cartesiano. Suponha-se seja a tabela descrita na figura 7, a mesma está disposta com 4 linhas e sempre, em todos os casos, 6 colunas, com a contagem que se inicia por 0, ou seja, da coluna 0 a coluna 5, da linha 0 a linha 3. Desta forma, o algoritmo fará a varredura por meio de uma estrutura de repetição, levando-se em consideração a tabela da seguinte forma: tabela (linha, coluna).

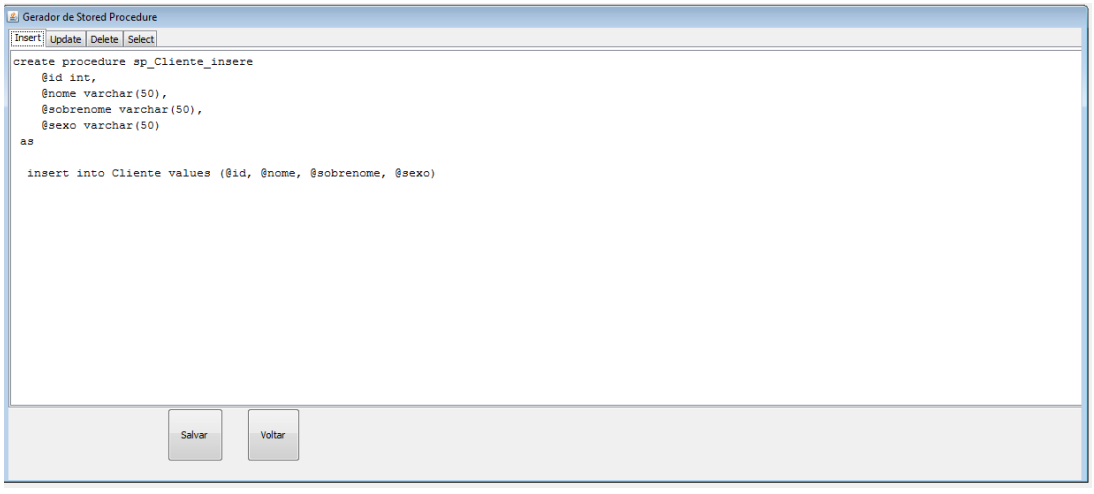


Figura 6: Tela de gerador de *stored procedure*.

A partir do código gerado, assim como ilustra a figura 6, em que os componente estão dispostos em abas de maneira que se possa editá-los, é apresentado as *stored procedures* de inserção, atualização, exclusão e seleção.

NOME DA TABELA	NOME DA COLUNA	TIPO DE DADOS	TAMANHO TIPO DE DADO	TIPO DO OBJETO	NOME DO PARÂMETRO
Cliente	id	int		table	@id
Cliente	nome	varchar	50	table	@nome
Cliente	sobrenome	varchar	50	table	@sobrenome
Cliente	sexo	varchar	50	table	@sexo

Figura 7. Tabela com os metadados.

Os dados serão dispostos de maneira que:

Tabela (0,4)	Table
Tabela (1,4)	Table
Tabela (2,4)	Table
Tabela (3,4)	Table
Tabela (0,5)	@id
Tabela (1,5)	@nome
Tabela (2,5)	@sobrenome
Tabela (3,5)	@sexo

Posição	**Valor**
Tabela (0,0)	Cliente
Tabela (1,0)	Cliente
Tabela (2,0)	Cliente
Tabela (3,0)	Cliente
Tabela (0,1)	Id
Tabela (1,1)	Nome
Tabela (2,1)	Sobrenome
Tabela (3,1)	Sexo
Tabela (0,2)	Int
Tabela (1,2)	Varchar
Tabela (2,2)	Varchar
Tabela (3,2)	Varchar
Tabela (0,3)	Nulo
Tabela (1,3)	50
Tabela (2,3)	50
Tabela (3,3)	50

De posse destes dados, o algoritmo concatena as informações da maneira conveniente para estruturar as *stored procedures.* O pseudocódigo da figura 8 apresenta o algoritmo necessário para criar uma *stored Procedure* de inserção.

É declarada uma variável i, que servirá como contador da estrutura de repetição, ela

receberá o valor de 0; a estrutura entrará em *loop* de repetição enquanto a variável for menor do que a quantidade de linhas da tabela. Por conseguinte, o campo de texto recebe as primeiras palavras chaves para criação da *stored procedure: campo_texto<-("createproceduresp_" + tabela(0, 0) + "_insere \n").*

O padrão de nomenclatura para criação das *stored procedures* foi definido da seguinte maneira:

Sp_nomDatTabelaSelect

É perceptível que o algoritmo efetua a busca do nome da tabela por meio do conteúdo da célula na posição Tabela(0,0), a qual define a coluna que contém o nome da tabela selecionada pelo usuário.

Sp_nomDatTabelaInsert Sp_nomDatTabelaUpdate Sp_nomDatTabelaDelete

```
algoritmo "Insert_procedure"
início
var i: inteiro
var pos: inteiro
var coluna: inteiro
var linha: inteiro
tabela(linha, coluna)
pos <- tabela.pegaQtdLinhas: valor total de linhas da tabela

campo_texto <- ("create procedure sp_" + tabela(0, 0) + "_insere \n");
para i <- 0, enquanto i < pos
início
    se  ( tabela(i, 3) = nulo ) então
    início
        se (i < (pos - 1)) então
          início
            imprima (campo_texto +" "  + tabela(i, 5) + " "+ tabela(i, 2) + ", \n ");
          fim
        se não, se (i = (pos - 1)) então
          início
            imprima (campo_texto +" "  + tabela(i, 5) + " "+ tabela(i, 2) + " \n ");
          fim
      fim
    se não, se  (i < (pos - 1)) então
        início
          imprima(campo_texto  + "    " + tabela(i, 5) + " " + tabela(i, 2) + "(" + tabela(i, 3) + "), \n ");
      fim
    se não, se (i = (pos - 1) E tabela(i, 3) <> nulo)
          início
          imprima (campo_texto + "    " + tabela(i, 5) + " " + tabela(i, 2) + "(" + tabela(i, 3) + ") \n ");
          fim
     incrementa o contador i++

    fim
fim
```

Figura 8. Pseudocódigo do algoritmo utilizado para gerar a sintaxes de *stored procedure* de inserção.

Para facilitar o entendimento do pseudocódigo, o mesmo foi esboçado nos

fluxogramas das figuras 9, 10 e 11 a seguir, sendo que ambas se complementam. A primeira parte do fluxograma apresenta o início do algoritmo. São definidas duas variáveis: i, que receberá o valor zero, e servirá como contador, e *pos*, que recebe a quantidade de linhas existentes na tabela atual. Após isso, é dado início à estrutura de repetição.

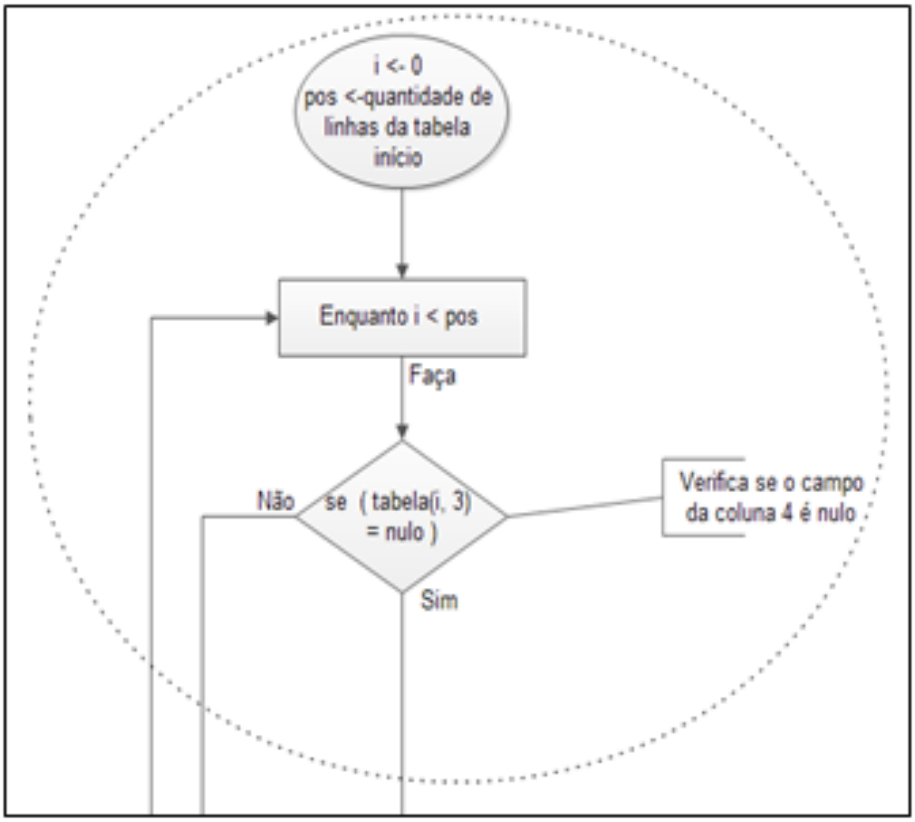


Figura 9. Estruturas de decisões fundamentais

A figura 9 apresenta as estruturas de decisões fundamentais para a formulação do corpo da *stored procedure.* Este cenário é necessário para que se possa alcançar a declaração dos atributos sem que seus respectivos tipos fiquem com o valor *null* entre parênteses, como por exemplo @id int(null), sem esta estrutura de decisão seria possível esta notação, o que causaria erro de sintaxe.

A figura 10 explica a continuação do contexto para a formulação da *stored procedure*, e a Figura 11 estabelece o fim do algoritmo.

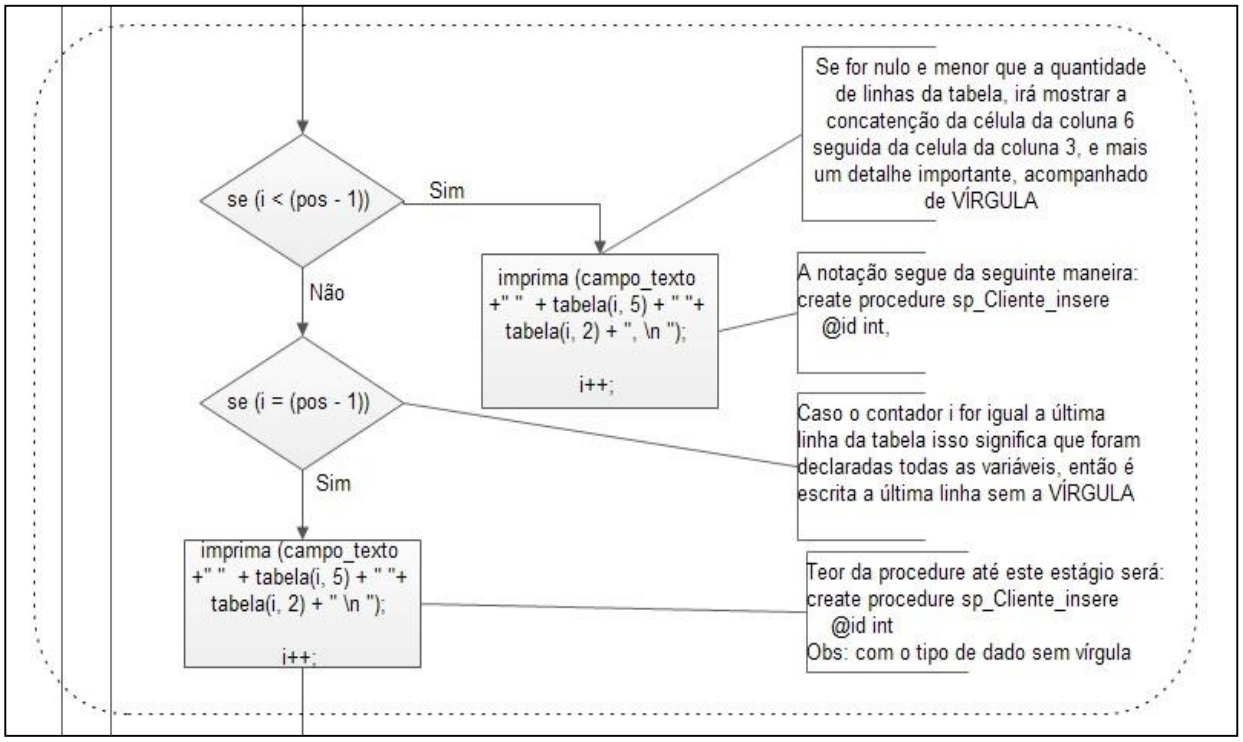


Figura 10. Meio do algoritmo.

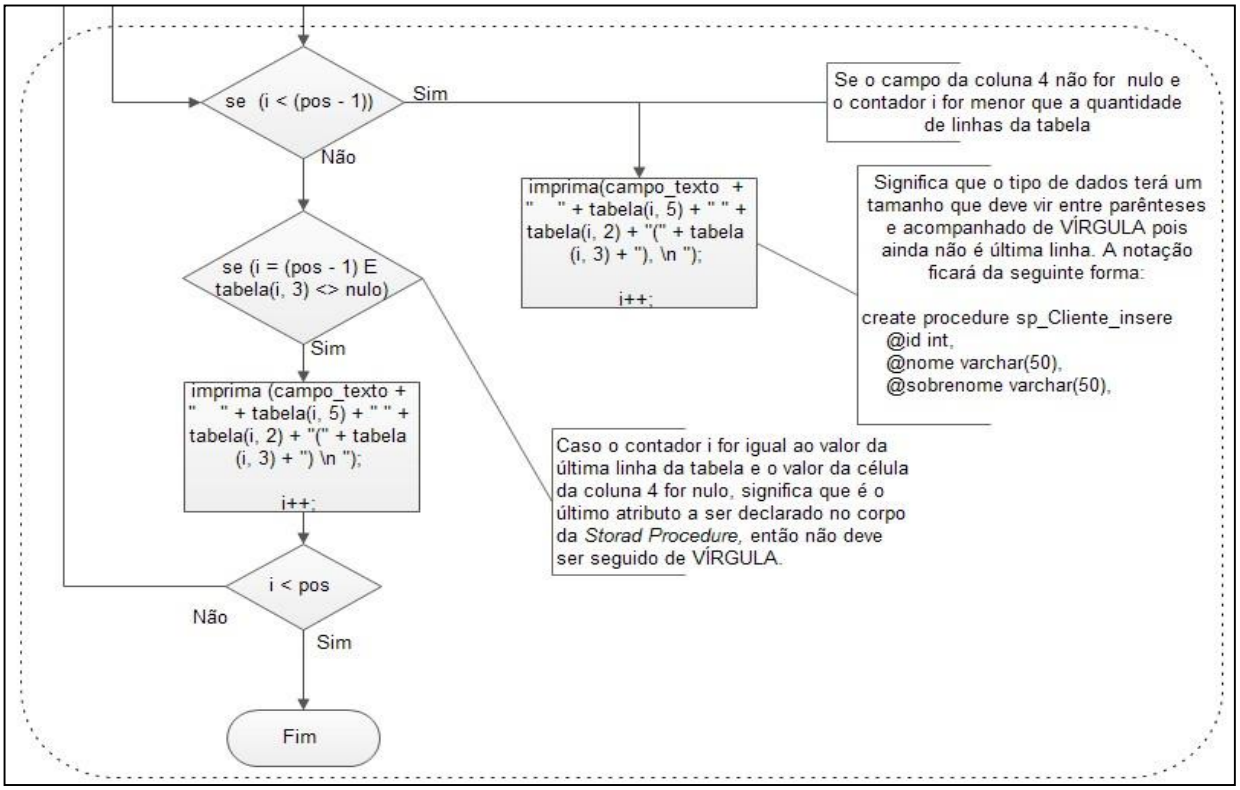


Figura 11. Fim do algoritmo.

Para efetuar a construção das demais *stored procedures*, o algoritmo segue a mesma tendência, apenas se reajusta para a formulação da sintaxe de cada uma.

Considerações Finais

Reuso em banco de dados é uma linha pouca explorada dentro do meio científico. A ferramenta vem ao encontro para sanar este percalço, por meio da produção de código reutilizável, tornando-a uma verdadeira alternativa frente aos demais produtos disponíveis no mercado atual, haja vista ser ainda pouco explorado nesta área.

Infere-se na ferramenta que ela é capaz de satisfazer ao problema levantado no decorrer deste artigo: a produção de componentes de reuso em banco de dados outrora definidos, mais especificamente *stored procedures*, bem como, produzir, por meio de uma interface gráfica, soluções para que o programador edite tais componentes, tornando, desta maneira, viável a utilização da aplicação em um ambiente de produção de software.

A proposta de fornecer um aplicativo que gerencie a criação e edição de *stored procedures* foi satisfeita pela implementação da ferramenta Case Pro, porém, o enriquecimento do aplicativo se faz necessário, visto que, somente um dos componentes foi tratado: as *stored procedures*.

Trabalhos Futuros: A implementação dos demais componentes de reuso em banco de dados, são uma das alternativas para trabalhos futuros, já que a ferramenta Case Pro contemplou apenas o conceito de *stored procedure*, pode-se explorar ainda os conceitos de *trigger* e e*view*. Outra proposta seria a mensuração do impacto que a introdução desta ferramenta traria no ambiente didático/pedagógico, como ferramenta de apoio ao ensino das disciplinas de banco de dados nas universidades.

Por tratar de forma simples, clara e objetiva, há a possibilidade de se aplicar o uso desta ferramenta no apoio pedagógico no ensino das disciplinas de banco de dados nos cursos de tecnologia da informação. Professores de tais disciplinas podem utilizar como ferramenta didática ao ministrar o conteúdo de *Storage Procedures*, e tornar o aprendizado mais harmonioso e eficaz.

Referências

CARLI, Erivaldo de. **Frameworks e reuso de software: implementação de um sistema de efetividade - RH utilizando o framework casca.**124f. (Monografia de Bacharelado em Engenharia de Computação). Rio Grande, Universidade Federal do Rio Grande, 2008. Disponível em: <http://www.nti.furg.br/images/stories/producao_cientifica/tcc_erivaldo_de_carli.pdf>. Acesso em 25/07/2014.

CARVALHO, Daniel Dias de; COSTA, Anderson J. Serra da; LIMA, Adailton Magalhães; REIS, Rodrigo Quites Reis; SALES, Ernani de Oliveira. **Apoio à reutilização de processos de software em um ambiente de engenharia de software centrado em processo.** Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/sbqs/20 11/SBQS2011-TT16_82927_1.pdf> Acesso em 15/08/2014.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de Banco de Dados**. São Paulo: Editora Pearson, 2010. 87-88p

HUTH, G. **Um modelo para gerenciamento de banco de dados SQL através de *Strored Procedures.*** Florianópolis: Universidade Federal de Santa Catarina, 2002. 7p. Dissertação (mestrado) – Federal de Santa Catarina, Florianópolis.

LANNA, André Luiz Peron Martins. **Reuso de processos de software baseado na componentização de processos e conhecimento.** Belo Horizonte: Pontifícia Universidade Católica de Minas Gerais, 2009.40p. Dissertação (mestrado). - Pontifícia Universidade Católica de Minas Gerais, Minas Gerais.

LUCRÉDIO, D. **Uma abordagem orientada a modelos para reutilização de Software**. São Carlos: Instituto de ciências matemáticas e de computação universidade de São Paulo, 2009. 6p. Tese (doutorado) – Universidade São Paulo, São Carlos.

SOUZA, Marcelo Gurgel. **Estudo do desenvolvimento de software baseado em reuso no contexto do ministério da defesa e de seus comandos subordinados**. 148f. (Dissertação de Mestrado em Sistemas de Computação) Rio de Janeiro, IME, 2004. Disponível em: <http://www.comp.ime.eb.br/dissertacoes/2004- Marcelo_Souza.pdf>. Acesso em 25/07/2014.