REFATORAÇÃO: APERFEIÇOANDO UM CÓDIGO EXISTENTE

BARROZO, Graccielle Castilho[1] *VINHAS, Hingriddi Marques[1]*

REIS, José Cláudio de Sousa[2]

(1) Acadêmicos do 8º período do Curso de Bacharelado em Ciência da Computação, UNIFENAS, Campus de Alfenas.

(2) Orientador.

Resumo

Este projeto apresenta um estudo sobre as várias técnicas de refatoração existentes, suas vantagens e desvantagens. Para isto, foi desenvolvido um Sistema de Controle de Monografia sem o uso de nenhuma técnica de refatoração. O sistema foi desenvolvido utilizando duas Arquiteturas: a Arquitetura em Camadas e a Arquitetura Orientada a Objetos. As tecnologias utilizadas para o desenvolvimento do sistema foram a Plataforma .NET da Microsoft, a linguagem de programada C# e o Banco de Dados utilizado foi o SqlServer 2008. Após o desenvolvimento do sistema, foram aplicadas diversas refatorações e foi comprovado que a utilização de tais técnicas melhora a qualidade do código, deixando-o mais limpo e de fácil manutenabilidade. Pôde-se observar também que com o uso de técnicas de refatoração houve uma redução de aproximadamente 9% no total de linhas de códigos do sistema.

Abstract

This project presents a study on the various techniques of refactoring existing, its advantages and disadvantages. For this, we developed a System Control Monograph without the use of any technique of refactoring. The system was developed using two architectures: a Layered Architecture and Object Oriented Architecture. The technologies used for the development of the system were the Platform. NET Microsoft's C # language and programmed Database was used

SqlServer 2008. After the development of the system were implemented various refactorings and it was proven that the use of such techniques improves code quality, leaving it cleaner and easier maintainability. It might also be noted that with the use of refactoring techniques decreased by approximately 9% of the total lines of code in the system.

Palavras-chaves: refatoração, código fonte, manutenabilidade,entendimento, aperfeiçoamento.

1 INTRODUÇÃO

Nos tempos de hoje os projetistas de sistemas estão cada vez mais sendo procurados para o desenvolvimento de softwares que facilitam a realização de tarefas diárias, pela falta de tempo que a maioria das pessoas apresenta. Devido a esta grande demanda muitos projetistas não se preocupam em desenvolver softwares com qualidade.

Para se atingir um software de qualidade não basta apenas ele estar funcionando é necessário que internamente esteja bem estruturado, compreensível e de fácil manutenibilidade. Com a finalidade de se alcançar esses objetivos, várias técnicas precisam ser aplicadas dentre elas a refatoração.

Refatoração é a alteração de um código fonte, visando melhorar o entendimento e a manutenibilidade sem alterar suas funções externas. Apesar de trazer benefícios para um código fonte existente, a refatoração pode apresentar riscos se não aplicada da forma correta, tais como: atraso do projeto, introdução de falhas no sistema, tornar o código ilegível e não modificável.

1.1 Objetivos

Este projeto tem como objetivos:

- Verificar as vantagens e as desvantagens da refatoração.

- Estudar as diversas técnicas de refatoração aplicá-las a um software e fazer uma comparação analítica das duas versões.

2. REFERENCIAL TEÓRICO

2.1 Engenharia de Software

Segundo Pressman (1995) devido ao crescimento eminente dos softwares, o mesmo se tornou uma preocupação administrativa, mais tarde a sua confiabilidade foi colocada em questionamento. Então surgiu a Engenharia de Software para tentar contornar a crise do software, dando um tratamento mais controlado ao desenvolvimento de sistemas.

Na construção de sistemas, para obter um software final de alta qualidade, é preciso que seja seguido um conjunto de passos pré-definidos, chamado de processo de software, este é um esboço para as atividades que são necessárias para construir um software de qualidade.

É aplicável à grande maioria dos projetos de software independente de seu tamanho o arcabouço de processo genérico, tal ele, Comunicação, Planejamento, Modelagem, Construção e Implementação, respectivamente.

Nos tempos de hoje existem vários modelos de processo de software entre eles podemos citar os mais usados, o Modelo em Cascata e os modelos evolucionários: Modelo em Espiral e o Processo Unificado.

2.2 Refatoração

Quando os desenvolvedores começaram a analisar seus códigos para incluirem novas funcionalidades ou modificarem as já existentes, eles observaram que os códigos estavam em sua maioria desestruturados, de dificil

compreensão, manutenção e com trechos duplicados. A refatoração surgiu através dessa observação.

Algumas pessoas pensam que Refatoração é apenas uma limpeza de código, mas ela vai além disso, porque fornece técnicas específicas para cada tipo de alteração. Então se forem usadas da forma correta deixa-o menos propenso a erros.

“Refatoração é uma alteração feita na estrutura interna do software para torná-lo mais fácil de ser entendido e menos custoso de ser modificado sem alterar seu comportamento observável.” (FOWLER, 2004, p. 52).

3 MATERIAL E MÉTODOS

3.1 Tecnologias utilizadas

No desenvolvimento deste projeto utilizaram-se as seguintes tecnologias: C#, ASP. NET, SQL Server 2008, Visual Studio. NET.

No desenvolvimento do sistema aplicaram-se os seguintes métodos: PU, UML, Arquitetura Orientada a Objetos e Camadas, Modelagem Relacional para o banco de dados.

3.2 Etapas de Desenvolvimento do projeto

- Primeira etapa: Pesquisa para que se pudesse entender a refatoração, suas principais técnicas e como aplicá-las;
- Segunda etapa: Desenvolvimento de um sistema sem utilizar nenhuma técnica de refatoração;
- Terceira etapa: Refatoração de grande porte, gerando assim uma nova versão do sistema;

- Quarta etapa: Análise comparativa entre as versões do sistema.

4 REFATORAÇÃO: ANÁLISE PRÁTICA

4.1 Definição do Sistema

O Sistema de Controle de Monografias – SCM tem por objetivo facilitar efetivamente o controle de entrada e empréstimos de monografias.

Espera-se que com o uso desse sistema, as atividades básicas do controle de monografias sejam realizadas de forma ágil, tornando o trabalho do secretário mais prático. As atividades de cadastro, edição e exclusão de alunos, professores, monografias e empréstimos serão controlados pelo Sistema de Controle de Monografias.

4.2 Sistema sem Classe

Na primeira etapa do projeto foi desenvolvido o Sistema de Controle de Monografia sem a utilização de classes. Apesar de estar utilizando uma plataforma totalmente orientada a objetos o código estava escrito de forma procedural, ou seja, todas as funções, tais como consultas, inserções e atualizações no banco de dados, foram desenvolvidas nos próprios forms, não aproveitando assim alguns benefícios da orientação a objetos, como a reutilização de códigos.

O sistema contém vários “Maus Cheiros”, mas com certeza o código duplicado ocorre com maior frequência, podendo ser encontrado com facilidade nessa versão do sistema. Na sequência um exemplo de código duplicado no SCM.

Temos dois Subsistemas, o Subsistema Aluno e o Subsistema Empréstimo, em ambos há uma necessidade de utilizar uma função que selecione o aluno pela sua matrícula.

No Subsistema Aluno, essa função é utilizada na página de alteração de Alunos, onde existe a opção de consulta pela matrícula do aluno.

```
protected void Button3_Click(object sender, EventArgs e)
{
    if (TextBox2.Text != "")
    {
        Label3.Text = "";

        SqlConnection conn = new SqlConnection(ConnectionStr);
        conn.Open();

        SqlCommand comando = new SqlCommand("ConsultaAlunoEspecifico", conn);

        //a instrucao SQL vira de uma store procedure
        comando.CommandType = CommandType.StoredProcedure;

        comando.Parameters.AddWithValue("@matricula", TextBox2.Text);

        //executa a instrucao SQL e armazena o resultado no obj SqlDataReader
        SqlDataReader reader = comando.ExecuteReader();
```

FIGURA 1 – Procedimento Consulta da página Alterar Alunos.

Já no Subsistema Empréstimo, na página de inclusão de Empréstimo, é necessária a utilização da mesma função para selecionar o aluno que irá fazer o empréstimo.

```
protected void Button1_Click(object sender, EventArgs e)
{
    if (TextBox2.Text != "" && TextBox3.Text != "" && TextBox4.Text != "" && TextBox5.Text != "")
    {
        Label12.Text = "";
        SqlConnection conn = new SqlConnection(ConnectionStr);
        conn.Open();

        SqlCommand comando = new SqlCommand("ConsultaAlunoEspecifico", conn);

        //a instrucao SQL vira de uma store procedure
        comando.CommandType = CommandType.StoredProcedure;

        comando.Parameters.AddWithValue("@matricula", TextBox2.Text);

        //executa a instrucao SQL e armazena o resultado no obj SqlDataReader
        SqlDataReader reader = comando.ExecuteReader();
```

FIGURA 2 – Procedimento Consulta da página Incluir Empréstimo.

Como se pode observar, lugares diferentes do sistema necessitam utilizar uma mesma função (procedimento). Para diminuir esse problema foi observada a necessidade de remodelar o sistema com uma estrutura de classes, capaz de unir esses trechos de códigos.

4.3 Sistema com Classe

Devido aos vários problemas apresentados pela versão sem classe do SCM, a primeira refatoração realizada no sistema foi a **Converter Projeto Procedural em Objetos**, pois além de ser uma refatoração considerada de grande porte, ela deixou o sistema de SCM com uma estrutura de classes capaz de unificar a maioria dos problemas de códigos duplicados do sistema.

Os passos para executar essa refatoração, são:

- ✓ Analisar cada tabela do banco de dados, e verificar quais podem ser transformadas em classes.
- ✓ Percorre-se o código e movem-se as devidas funções para uma classe de acordo com sua funcionalidade.

Tomando o exemplo dado de código duplicado no sistema sem classe, depois de criada a Classe **Aluno**, o próximo passo é criar um método que retorne uma

consulta pela matrícula do aluno. Na sequência o código da estrutura da classe **Aluno** já com o método **ConsultaAlunoPorMatricula**:

```
public partial class Aluno
{
    public string ConnectionStr = @"Data Source=.\SQLEXPRESS;AttachDbFilename=C:\Users\Hingriddi\Desktop\TCC\Controle de Monografia\App_Data\ControleMonografia.mdf;
                                    Integrated Security=True;Connect Timeout=30;User Instance=True";

    public SqlDataReader ConsultaAlunoPorMatricula(string matricula)
    {
        SqlConnection conexaoAlunoPorMatricula = new SqlConnection(ConnectionStr);
        conexaoAlunoPorMatricula.Open();
        SqlCommand comandoConsultaAlunoPorMonografia = new SqlCommand("ConsultaAlunoEspecifico", conexaoAlunoPorMatricula);
        comandoConsultaAlunoPorMonografia.CommandType = CommandType.StoredProcedure;
        comandoConsultaAlunoPorMonografia.Parameters.AddWithValue("@matricula", matricula);
        SqlDataReader readerConsultaAlunoPorMonografia = comandoConsultaAlunoPorMonografia.ExecuteReader();
        return readerConsultaAlunoPorMonografia;
    }
```

FIGURA 3 – Classe Aluno

A grande vantagem é que este método(ConsultaAlunoPorMatricula) será chamado de vários lugares no sistema, acabando com o grande problema de código duplicado para essa função, reduzindo o trabalho em futuras manutenções do sistema e economizando linhas de código.

Com o método **ConsultaAlunoPorMatricula** criado, apenas troca-se o código que estava no form e chama-se o novo método criado na classe Aluno.

O código do form, ficou assim:

```
protected void btPesquisarAluno_Click(object sender, EventArgs e)
{
    if (TextBox2.Text == "")
    {
        habilitarCampos("DESATIVAR");
        Label3.Text = "Digite o campo para pesquisa.";
        TextBox2.Focus();
    }
    else {
        Label3.Text = "";
        SqlDataReader reader = aluno.ConsultaAlunoPorMatricula(TextBox2.Text);
```

FIGURA 4 – Utilizando o método ConsultaAlunoPorMatricula da classe Aluno.

4.4 Sistema Refatorado

Mesmo aplicando a refatoração **Converter Projeto Procedural em Objetos**, o sistema ainda apresentava alguns “Maus Cheiros”, então foram aplicadas algumas técnicas de refatoração para melhorar a estrutura interna do código. Na sequência, a descrição de cada uma das técnicas aplicadas:

Renomear Métodos:

Essa refatoração é utilizada quando o nome de um método não revela seu propósito, deve-se então alterar o nome deste método.

O SCM apresentava esse problema, então foi aplicada esta refatoração. Na sequência, como era a declaração do método e como ficou após a refatoração:

```
protected void Button1_Click(object sender, EventArgs e)
```

FIGURA 5 – Declaração do método sem refatoração

```
protected void btIncluirEmprestimo_Click(object sender, EventArgs e)
```

FIGURA 6 – Declaração do método após a refatoração

Extrair Método:

Essa refatoração é utilizada quando um método está muito longo ou é necessário comentá-lo para ter seu propósito compreendido, esse trecho precisa ser extraído para um novo método.

O SCM apresentava esse problema, então foi aplicada esta refatoração. Na sequência, como era o método e como ficou após a refatoração:

```
protected void Button3_Click(object sender, EventArgs e)
{
    if (TextBox3.Text != "")
    {
        Label3.Text = "";

        SqlConnection conn = new SqlConnection(ConnectionStr);
        conn.Open();

        SqlCommand comando = new SqlCommand("ConsultaProfessorEspecifico", conn);

        //a instrucao SQL vira de uma store procedure
        comando.CommandType = CommandType.StoredProcedure;

        comando.Parameters.AddWithValue("@codigo", TextBox3.Text);

        //executa a instrucao SQL e armazena o resultado no obj SqlDataReader
        SqlDataReader reader = comando.ExecuteReader();
        if (reader.HasRows == true)
        {
            Label4.Visible = true;
            TextBox1.Visible = true;
            Label7.Visible = true;
            Label5.Visible = true;
            TextBox2.Visible = true;
            Label8.Visible = true;
            Label6.Visible = true;
            DropDownList2.Visible = true;
            Label9.Visible = true;
            Button1.Visible = true;
```

FIGURA 7 – Procedimento que pesquisa um Professor e habilita os campos para edição antes da refatoração.

```
protected void btPesquisarProfessor_Click(object sender, EventArgs e)
{
    if (TextBox3.Text == "")
    {
        habilitarCampos("DESATIVAR");
        Label3.Text = "Digite o campo para pesquisa.";
        TextBox3.Focus();
    }
```

FIGURA 8 – Procedimento que pesquisa um Professor e habilita os campos para edição após a refatoração.

Reverter Condicional:

Essa refatoração é utilizada quando se tem uma condicional que seria mais fácil de entender se o seu sentido estivesse invertido. Então inverta o sentido da condicional e reordene as cláusulas.

O SCM apresentava esse problema, então foi aplicada esta refatoração. Na sequência, como era a condicional e como ficou após a refatoração:

```
protected void Button1_Click(object sender, EventArgs e)
{
    if (TextBox1.Text != "" && TextBox6.Text != "")
    {
        Label12.Text = "";
        SqlConnection conn = new SqlConnection(ConnectionStr);
        conn.Open();
        SqlCommand comando = new SqlCommand("DevolverEmprestimo", conn);

        //a instrucao SQL vira de uma store procedure
        comando.CommandType = CommandType.StoredProcedure;

        DateTime datadev = Convert.ToDateTime(TextBox1.Text);
        string datadevolucao = datadev.ToString("yyyy-MM-dd");

        comando.Parameters.AddWithValue("@codigo", TextBox6.Text);
        comando.Parameters.AddWithValue("@dataentrega", datadev);
        comando.ExecuteNonQuery();
        Response.Redirect("Emprestimo.aspx?acao=devolver&id=" + TextBox6.Text);
    }
    else {
        Label12.Text = "Preencha os campos obrigatórios(*).";
    }
}
```

FIGURA 9 – Procedimento que realiza a devolução da Monografia antes da refatoração

```
protected void btDevolverEmprestimo_Click(object sender, EventArgs e)
{
    if (TextBox1.Text == "" || TextBox6.Text == "")
    {
        Label12.Text = "Preencha os campos obrigatórios(*).";
    }
    else {
        Label12.Text = "";
        SqlDataReader reader = emprestimo.ConsultaEmprestimoPorCodigo(TextBox6.Text);
        if (reader.HasRows)
        {
            emprestimo.DevolverEmprestimo(TextBox6.Text, TextBox1.Text);
            Response.Redirect("Emprestimo.aspx?acao=devolver&id=" + TextBox6.Text);
        }
        else
        {
            Label12.Text = "Código do empréstimo incorreto.";
        }
    }
}
```

FIGURA 10 – Procedimento que realiza a devolução da Monografia após a refatoração

Todas essas refatorações utilizadas no SCM contribuirão para a diminuição de linhas de código, bem como a facilitação no entendimento do código e manutenções futuras.

5 RESULTADOS E DISCUSSÕES

As várias refatorações aplicadas ao SCM possibilitaram uma análise efetiva na melhoria da qualidade do software, além de uma redução considerável no número de linhas de código.

Na sequência, o gráfico demonstra a redução das linhas de código da versão sem refatoração do SCM para a versão refatorada do SCM.

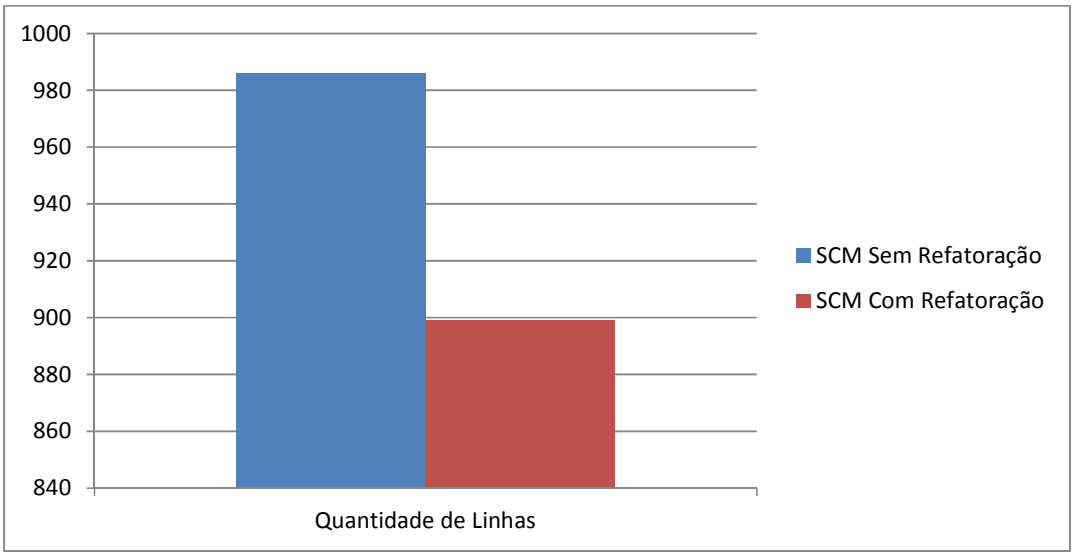

FIGURA 11 – Gráfico comparativo entre as versões sem Refatoração e com Refatoração do SCM.

As melhorias que foram identificadas referem-se à fatores de qualidade, tais como: Eficiência, Manutenabilidade, Reutilização, Concisão, Modularidade e Simplicidade.

6 CONCLUSÃO

Com base nos resultados adquiridos, pôde-se verificar que é possível o uso de técnicas que orientem a refatoração, pois possuem passos simples que se seguidos corretamente, levam ao sucesso da refatoração. Observou-se que o código quando refatorado torna-se mais legível e eficiente, facilitando a

manutenabilidade do software. Pôde-se verificar também que o uso da refatoração contribui grandiosamente para a reutilização de código, o que colabora para um processo de desenvolvimento de software mais ágil e eficaz.

A refatoração não pode ser aplicada em todo código fonte, é necessário analisar cada trecho separadamente e verificar a possibilidade de refatorá-lo, pois aplicar a refatoração de forma equivocada poderá trazer problemas ao software.

Como indicação para trabalhos futuros, seria interessante a realização de um estudo sobre Técnicas de Refatoração para Banco de Dados, e aplicação de **novas** técnicas ligadas a refatoração.

REFERÊNCIAS

ABREU, Luiz. **Asp.Net 4.0: Curso Completo**. São Paulo: Editora de Informática FCA, 2011.

BATTISTI, Júlio**. ASP. NET: Uma nova revolução da construção de sites e aplicações web**. Rio de Janeiro: Axcel Books, 2001.

BOOCH, Grandy; RUMBAUGH, James; JACOBSON, Ivar.**UML – GUIA DO USUÁRIO.**São Paulo:Editora Campos, 2000.

FOWLER, Martin. **Refatoração: Aperfeiçoando o projeto de código existente**. Porto Alegre: Bookman, 2004.

FURLAN, José Davi. **Modelagem de Objetos através da UML**. São Paulo: Makron Books, 1998.

HOTEK, Mike. **SQL Server 2008 – Passo a Passo.** São Paulo: Bookman, 2010.

MARQUES, Davi Azevedo, **Refatoração: Aperfeiçoando um software existente.** Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) – Universidade José do Rosário Vellano, Alfenas – MG, 2006.

PRESSMAN, Roger. **Engenharia de Software**. São Paulo: Makron Books, 1995.

PRESSMAN, Roger. **Engenharia de Software**. São Paulo: Makron Books, 2006.

REFACTORING. Disponível em: <www.refactoring.com>. Acesso em: 14 Nov. 2012.

SCOTT, Kendall. **Processo Unificado Explicado**. Porto Alegre: Bookman, 2003.

SHARP, John. **Microsoft Visual C# 2010 Passo a Passo**. São Paulo: Bookman, 2008.