DOI:10.5748/9788599693131-14CONTECSI/PS-4703

MAINTENANCE METRICS APPLIED IN SOFTWARE REFACTORING

Bruno Nascimento Ferreira (Instituto de Pesquisas Tecnológicas, São Paulo, Brasil) - bruno.n.ferreira@ieee.org
Júlio Arakaki (Pontifícia Univeridade Católica de São Paulo, São Paulo, Brasil) - jarakaki@pucsp.br

Most of the activities to be performed in software maintenance usually consume about 70% of the total cost of the project. To reduce the cost of these maintenance, re-factoring becomes an important strategy that can be applied to the software being analyzed. Identifying the point at which refactoring is recommended may require the same effort as anticipating future changes in that code. The techniques that perform this identification depend on the inspection of the code by specialists. In this study, we develop an approach to support the decision to refactor code snippets in object-oriented systems, reducing the subjectivity of expert evaluation. In this work, we study the relationship between code snippets of changes in the system and the decision of the experts about the need to refactor the altered code, using some maintenance metrics such as the Maintenance Index, Cyclomatic Complexity and Metrics Of Chidamber and Kemerer Thus, the most representative metrics for the evaluation of refactoring are identified and applying regression methods, a equation is elaborated that indicates when the values of the metrics indicate the moment in which the experts recommend the refactoring. When applying the equation obtained in these research, we obtained as a result, a possible prediction about the need for refactoring in at least 86.67% of the analyzed cases.

Keywords: Refactoring; Maintenance metrics; Maintenance assessment.

APLICAÇÃO DE MÉTRICAS DE MANUTEBILIDADE NA RE-FATORAÇÃO DE SOFTWARES

A maioria das atividades a serem realizadas nas manutenções do software geralmente consomem cerca de 70% do custo total do projeto. Para reduzir o custo destas manutenções, a re-fatoração torna-se uma estratégia que importante que poderá ser aplicada no software em análise. Identificar o momento no qual a refatoração é recomendada pode demandar um mesmo esforço do que prever as futuras alterações no referido código. As técnicas que realizam esta identificação dependem da inspeção do código por especialistas. Neste estudo, desenvolve-se uma abordagem para apoiar a decisão da refatoração de trechos de código em sistemas orientados a objetos, diminuindo a subjetividade da avaliação de especialistas. Estuda-se neste trabalho, a relação entre trechos de código de alterações no sistema e a decisão dos especialistas sobre a necessidade de refatoração do código alterado, utilizando-se de algumas métricas de manutenabilidade, como o Índice de Manutenabilidade, Complexidade Ciclomática e as Métricas de Chidamber e Kemerer Desse modo, são identificadas as métricas mais representativas para a avaliação da refatoração e aplicando-se métodos de regressão, é elaborada uma fórmula que aponta quando os valores das métricas indicam o momento no qual os especialistas recomendam a refatoração. Ao aplicar-se a fórmula obtida nestas pesquisa, obteve-se como resultado, uma possível previsão sobre a necessidade de refatoração em pelo menos 86,67% dos casos analisados.

Palavras-chaves: refatoração; métricas de manutenibilidade; avaliação da manutenibilidade

INTRODUÇÃO

No ciclo de vida de um software típico, o desenvolvimento de um projeto é estimado em 30% do custo total, enquanto sua manutenção, em 70% (Boehm & Basili, 2001). Por consequência, existe a necessidade do gerenciamento do esforço de manutenção e a sua medição, pois, não se consegue controlar aquilo que não se mede (Chaparro, 2014).

Para Pressman, a mudança no software pode ocorrer a qualquer momento por qualquer motivo, com o agravante de que alguns tipos de sistemas, como os aplicativos Web, possuem funcionalidades que devem ser disponibilizadas para os usuários em questão de dias ou semanas, para garantir que o sistema esteja disponível na data adequada. Por consequência se faz necessário que o esforço de manutenção para a inclusão de uma nova funcionalidade seja adequado.

O esforço de manutenção é definido pela ISO\IEEE 25010 como um atributo de qualidade chamado manutenibilidade. Que é definida, como "o grau de eficácia e eficiência com que o produto ou o sistema pode ser modificado pelos responsáveis pela manutenção"..

Alguns autores, como Anda, Fowler e McCabe, desenvolveram métodos para medir a manutenibilidade do software criando métricas ou realizando a busca de características que indicam que a qualidade do código está inadequada. Para eles, os métodos de identificação de trechos de código com falhas de manutenibilidade têm etapas em comum: (1) a busca de classes ou estruturas que estão com os indicadores de manutenibilidade negativos, (2) a necessidade de uma avaliação manual e (3) a correção da falha, que corresponde à reescrita de todo ou parte do software.

Fowler descreve uma técnica usada para reescrever trechos de código, em sistemas orientados a objeto, chamada de "refatoração", que é "a mudança feita na estrutura interna do software para torná-lo fácil de entender e barato de ser modificado, sem alterar seu comportamento observável". A refatoração pode manter a manutenibilidade em níveis adequados se aplicada continuamente no processo de desenvolvimento (Beck & Date, 2002).

Para Fowler, a avaliação da qualidade do código produzido é realizada pela busca de indicadores, que apontam para problemas chamados de Code Smells ou "Cheiro do Código". Code Smells expressam a metáfora do mal cheiro e, aplicadas ao contexto da engenharia de software, indicam que o código difícil de manter deixa indícios de que precisa ser refatorado.

Chidamber e Kemerer criaram uma série de métricas estruturais, que são chamadas de "métricas de CK". Em seus experimentos, as métricas foram aplicadas a uma série de sistemas e os resultados foram analisados usando modelos estatísticos em busca de valores fora da distribuição normal, pois estes possivelmente representariam problemas de projeto.

Anda avalia a manutenibilidade de um software de forma combinada utilizando métricas bem definidas e a vaga definição de especialistas.

Nestes estudos, a avaliação dos indicadores sempre dependendo de avaliadores e seus critérios subjetivos para decidir se o código precisa ou não ser refatorado.

A dependência de especialistas leva a problemas como a dificuldade de depender de pessoas qualificadas para realizar esta avaliação e à divergência de opinião entre especialistas. (Anda, 2007). Além disso, existem cenários onde a dependência de critérios subjetivos não é adequada, como em casos de exigência legal ou de departamentos de qualidade interna que exigem a documentação de processos e procedimentos de acordo com a normal NBR/ISO 9001. Logo, o processo de decisão sobre a necessidade de refatoração precisa de critérios objetivos passíveis de serem documentados e reproduzidos.

OBJETIVO

O objetivo deste estudo é desenvolver uma abordagem para avaliar se a refatoração poderá ser recomendada para determinados trechos de código fonte. E, dessa forma, torná-lo mais eficiente para futuras alterações, utilizando-se de métricas de manutenibilidade.

O objetivo é que a abordagem identifique os trechos de código recomendados para a refatoração em projetos dentro de um ambiente de desenvolvimento de software, como uma empresa ou um conjunto de empresas, permitindo uma avaliação adequada para cada ambiente e possibilitando a identificação de trechos que tenham a refatoração recomendada, sem a necessidade da inspeção do código pelos especialistas do ambiente em estudo e, a cada nova implementação. E desta maneira, proporcionando melhor eficiência em comparação com uma revisão do código produzido manualmente.

A classificação de trechos para a qual a refatoração é recomendada deve diminuir a necessidade da inspeção de código. Porém, para definir os valores de manutenibilidade adequados para o ambiente da aplicação utilizado nesta abordagem, o trabalho utiliza-se da avaliação de especialistas sobre uma amostra de trechos de código. Documentações auxiliares como documentos de arquitetura e guias de desenvolvimento estão fora do escopo do trabalho, a abordagem se restringe a avaliar a manutenibilidade apenas pela avaliação do código fonte.

MÉTRICAS DE MANUTENIBILIDADE

O processo de medição pode ser definido como o processo em que os números e símbolos são atribuídos a entidades do mundo real para descrevê-los apropriadamente (Fowler, 1999). A medição é utilizada para realizar a comparação entre as entidades do mundo real.

No contexto da engenharia de software, a medição é usada para entender o que ocorre nos processos de desenvolvimento e manutenção do software (Fowler, 1999).

Pressman discutiu a eficácia da utilização de métricas de software afirmando a existência de centenas de métricas propostas para avaliar sistemas, mas ressaltando que nem todas fornecem apoio prático para a engenharia de software. Isto porque algumas delas são muito complexas, outras são esotéricas, e algumas não expressam a noção intuitiva sobre qualidade de software.

A seleção de métricas utilizadas neste estudo utilizou os critérios citados por Pressman e incluiu novos critérios de acordo com a necessidade do estudo: (1) busca de um apoio prático, (2) popularidade em artigos, (3) disponibilidade de ferramentas para a medição na linguagem C#, e (4) crítica destas métricas na literatura.

Algumas das métricas que buscam avaliar a manutenibilidade do software estão destacadas nos itens a seguir:

A. Complexidade Ciclomática

A complexidade ciclomática é a métrica elaborada no estudo de McCabe com o objetivo de medir o esforço de teste e depuração de um programa estruturado.

A métrica se baseia na interpretação do fluxo de execução de um programa, entendendo o fluxo como um grafo, o que permite a utilização das propriedades dos grafos para atribuir um valor de complexidade.

Ela surgiu em contraposição a métricas de volumes como linhas de código, pois, um programa com múltiplos caminhos, como, por exemplo, com múltiplas instruções IF-ELSE, é mais complexo de ser depurado do que programas com muitas linhas de código, que seguem um fluxo único.

Para calcular a complexidade ciclomática, cada instrução de decisão é interpretada como um nó do grafo e cada bloco de instruções sequenciais como uma aresta.

A complexidade é calculada pela formula de Berge, conforme a fórmula 1: “O número ciclomático V(G) do grafo G com n vértices, e Nós, e p componentes conectados é: ”(Oman & Hagemeister, 1992).

$$v(G)=e-n+p(1)$$

A. *Índice de manutenibilidade*

O índice de manutenibilidade, ou MI (do inglês - Maintainability Index), combina diversas métricas já existentes em uma única métrica, com a finalidade de representar o esforço sobre a manutenção em um único valor (Sjøberg et al, 2013).

Para construção da métrica foram utilizadas algumas das métricas existentes que expressam a noção de manutenibilidade. Para isto o autor definiu três grupos de métricas que deveriam ser observados. São elas:

Métricas de atributos de maturidade: definidos a partir de características físicas e medidas pala idade e pelo uso do sistema;

Métricas de código fonte: características do código de fonte do sistema; são classificadas nas seguintes subcategorias: estrutura de controle, estrutura de informação e tipografia, nome e comentários;

Métricas de documentação de apoio: características de documentação do sistema, sendo documentação abstrata ou atributos físicos.

Estas métricas foram observadas em 16 grandes sistemas da empresa HP, a fim de se encontrar uma relação que expressava a manutenibilidade para os engenheiros desta empresa (Yamashita & Moonen, 2012).

Após a coleta destas métricas foram utilizados cinco métodos para quantificar o esforço de manutenção. São eles; (1) modelos hierárquicos e multidimensionais, (2) modelos de regressão polinomiais; (3) medida da complexidade para previsão de erros do código; (4) análise estatística para considerar somente os componentes principais e (5) análise de fatores.

Algumas métricas se destacaram para a previsão do esforço de manutenção, resultando em uma função que expressa a manutenibilidade, e resultando na equação 2:

$$MI = 171 - 5{,}2\ln(HV) - 0{,}23CC - 16{,}2\ln(LOC) + 50\sin\sqrt{2.46 * COM} \quad (2)$$

Onde: HV é o Volume de Halstead, CC é Complexidade Ciclomática, LOC são as linhas de código e COM é a porcentagem de comentários do sistema.

Para validação, a métrica foi aplicada em 11 sistemas das empresas HP e do Departamento de Defesa Americano, e o seu resultado foi comparado com a opinião dos engenheiros da HP sobre a dificuldade de manutenção dos sistemas avaliados.

Os resultados classificaram os sistemas com o valor de índice de manutenibilidade maior que 85 como sistemas de alta manutenibilidade, e, aqueles com menos de 65, como sistemas de baixa manutenibilidade (Yamashita & Moonen, 2012)

Porém, o estudo realizado por Bray et al. recomenda cautela na sua utilização. É recomendada a calibração dos valores, a fim de adequar de forma mais precisa os coeficientes que classificam o código como bom ou ruim. Além disso, o estudo recomenda que os usuários da métrica adequem o peso da porcentagem de comentários de acordo com a quantidade necessária para cada situação.

B. *Métricas de Chidamber e Kemerer*

As métricas de manutenibilidade apresentadas anteriormente foram criadas para medição de sistemas procedurais e adaptadas para sistemas orientados a objetos. Um conjunto de métricas criadas para medição do projeto de sistemas orientados a objetos são as métricas de Chidamber e Kemerer.

As métricas de Chidamber e Kemerer se baseiam no processo de projeto orientado a objetos, OOD (do inglês – Object Oriented Design), proposto por Booch, com a proposta de expressar os conceitos deste

paradigma como acoplamento e coesão.

Do processo OOD são derivados seis métricas que expressam os conceitos da OOD. São elas:

- Métodos ponderados por classe – WMC (do inglês - Weighted Methods Per Class):
- Métrica calculada pelo número de métodos de cada classe ponderada pela complexidade de cada método.
- Profundidade da árvore de herança – DIT (do inglês - Depth of Inheritance Tree):
- Mede o número de filhos de uma determinada classe. Caso uma classe não tenha filhos, o seu valor de DIT é zero. Caso a classe possua herança múltipla, o valor do DIT é o valor da sua maior profundidade.
- Número de filhos – NOC (do inglês – Number of Children):
- Número de subclasses de uma classe na hierarquia.
- Acoplamento entre objetos – CBO (do inglês – Coupling between object classes):
- Número de classes com a qual a classe medida está acoplada.
- Repostas para a classe – RFC (do inglês – Response For a Class):
- O número de métodos que podem ser potencialmente chamados em resposta a uma mensagem recebida por um objeto.
- Falta de coesão nos métodos – LCOM (do inglês – Lack of Cohesion in Methods):

  LCOM é a contagem do número de métodos semelhantes subtraída da contagem de número de métodos não semelhantes. A semelhança é obtida pelo número de atributos utilizados pelos métodos.

Segundo Anda, as métricas de CK são provavelmente as métricas mais usadas. Para Barbosa Junior, elas são o conjunto de métricas estáticas mais referenciais na literatura de TI. Além disso, as métricas de CK são uma das primeiras métricas baseadas no modelo de programação orientado a objetos, o que justifica o uso destas métricas neste artigo.

Uma das características das métricas de CK é que as suas métricas podem indicar valores conflitantes; enquanto uma métrica indica um resultado positivo para manutenibilidade, outra pode indicar o resultado negativo. Assim como as métricas NOC e DIT, que avaliam diferentes atributos de qualidade da manutenibilidade, quanto maior o número de filhos de uma classe (NOC), melhor a sua reutilização; mas sua complexidade também aumenta (DIT). Logo, existe a necessidade de que as equipes de desenvolvimento e/ou arquitetura avaliem e definam o valor ideal para cada atributo de qualidade considerando os prós e contras de cada um, esta definição está fora do escopo deste trabalho.

AVALIAÇÃO DE MANUTENIBILIDADE POR ESPECIALISTAS

Existem duas formas de se avaliar a manutenibilidade pela avaliação de especialistas: a avaliação não guiada e a guiada (Anda, 2007).

Ambas as estratégias possuem dificuldades relacionadas à confiabilidade dos avaliadores. O estudo de Schneiderman encontrou divergência de opinião na avaliação de especialistas, sendo que os especialistas deste estudo não participaram do processo de desenvolvimento dos sistemas em análise. Já Shepperd encontrou alta confiabilidade na avaliação de especialistas que participam do time de desenvolvimento do sistema.

Outra dificuldade é a de que a avaliação por especialistas depende de pessoas qualificadas, com um alto nível de conhecimento nas ferramentas utilizadas para desenvolver o sistema em análise(Anda, 2007).

Mas, apesar destas dificuldades, a avaliação por especialistas é uma das técnicas mais utilizadas para a revisão de código. Fowler afirma que cabe aos especialistas e desenvolvedores do sistema quantificar qual é o momento de refatorar o sistema, definindo quanto é o número de linhas de código que caracteriza um método longo demais e quanto é o número de parâmetros que caracteriza um excesso.

C. Avaliação Não Guiada

A avaliação de software não guiada é a revisão do código de fonte por um ou mais desenvolvedores, normalmente com um alto grau de conhecimento, em busca de falhas no projeto ou na implementação.
O seu processo é demorado devido à natureza manual desta atividade, o que justifica o uso de técnicas e abordagem para a redução deste tempo, como a técnica proposta por este trabalho.

D. Avaliação Guiada

A avaliação de software guiada é realizada por especialistas utilizando algum tipo de documentação de apoio, como, por exemplo, checklist, panfletos e outros guias formais.

Uma das formas de avaliação guiada é a busca de Code Smells (cheiro do código) no código fonte.

Code Smells é definido por Fowler como a metáfora que expressa que o código difícil de manter deixa indícios de que precisa ser refatorado. Porém, avaliar e tomar a decisão sobre esses indícios depende dos critérios do avaliador.

Segundo Yamashita e Moonen, uma grande vantagem da utilização do Code Smells é a sua associação com uma série de estratégias de refatoração, mas sua utilização requer abordagens alternativas, como a inspeção do código, para avaliar a manutenibilidade do sistema.

Sjøberg avalia a utilização de code smells como complemento para quantificar o esforço de manutenção dos sistemas, pois não refletem todos os aspectos da manutenibilidade. Sugerindo utilizar os Code Smells como um guia para os especialistas que avaliam determinado trecho de código.

REFATORAÇÃO

O termo refatoração (do inglês Refactoring) surgiu na tese de Ph.D. de Opdyke como a alteração da técnica da reestruturação, que é aplicada em funções e módulos em sistemas estruturados, para aplicação em sistemas orientados a objetos.

O objetivo da refatoração é a melhora da manutenibilidade em estruturas internas, e ocorre durante o processo de desenvolvimento, durante os processos de desenvolvimento e operação.

A aplicação da refatoração deve ocorrer dentro do processo de desenvolvimento de sistemas, sendo responsabilidade do desenvolvedor a divisão do tempo entre as atividades de desenvolvimento e refatoração.

Fowler et al. listou um serie de benefícios da aplicação da refatoração, são eles:

- Melhoria do projeto do software;
- Melhoria da capacidade de compreensão do código;
- Melhoria no processo de localização de defeitos;
- Redução no tempo de alterações.
- Além dos benefícios também são listadas algumas dificuldades e problemas que ocorrem durante o processo de aplicação da refatoração. São eles:
- Banco de dados são difíceis de serem refatorados;
- Troca de interfaces publicadas resulta na perda da compatibilidade com versões anteriores.
- Mudanças no projeto da aplicação são resistentes à refatoração;
- Aplicação da refatoração em casos da necessidade de reescrita: A aplicação da refatoração em sistemas com alto número de defeitos pré-existentes não é efetiva;
- Aplicação da refatoração em sistemas com exigência de desempenho não é recomendada.

Fowler et al. também associa cada code smell a uma serie de estratégias de refatoração, proporcionando um guia de passos a serem realizados quando um problema é encontrado. O catálogo

possui 22 code smells que podem ser eliminados por um catálogo de 72 operações de refatoração.

E. Críticas a utilização de Code Smells para avaliar a manutenibilidade

A pesquisa de Chaparro et al. alerta que a refatoração pode eliminar os code smells, mas, em contrapartida, pode ocasionar efeitos colaterais, como a remoção de padrões de projeto desejados ou a introdução de defeitos.

Outra característica é que segundo Fard et al. é que as ferramentas que realizam a medição dos code smells tipicamente definem métricas e valores máximos a serem observado para cada Code Smells. Porém, esses valores são definidos a critério de cada ferramenta.

Gomes da Silva demonstra que as refatorações propostas por Fowler et al. causam efeitos positivos e negativos nas propriedades de baixo nível do sistema.

O trabalho de Gomes da Silva conclui que a refatoração traz benéficos em sua aplicação; porém, ela pode não ser recomendada em alguns casos devido a impactos negativos para manutenibilidade, sendo necessária uma avaliação do ganho da refatoração para a manutenibilidade antes de sua aplicação. A pesquisa de Chaparro et al. afirma que avaliar os prós e contras das operações de refatoração, antes de aplicá-las, é muito difícil para os desenvolvedores, pois refatorações podem acarretar em muitas alterações no código, o que dificulta a visualização mental de todas as operações necessárias. As dificuldades de avaliação se agravam se considerados os atributos de qualidade que tem características conflitantes (trade off), como acoplamento e coesão.

A fim de facilitar o processo de previsão de vantagens, ou desvantagens, de cada refatoração, a pesquisa de Chaparro et al. acarretou no desenvolvimento da ferramenta RIPE (do Inglês Refactoring Impact PrEdiction), que utiliza o catálogo de refatorações e code smells, proposto por Fowler et al., para previsão dos benefícios da aplicação de 12 refatorações em 11 métricas de manutenibilidade

A vantagem do uso de code smells para avaliação da manutenibilidade sobre as métricas tradicionais, como complexidade ciclomática, é a sua associação com uma série de operações de refatoração. Mas os code smells não descrevem nem identificam de forma clara a manutenibilidade do sistema, o que dificulta a sua utilização e a avaliação do custo benefício da refatoração.

Já Sjøberg et al. concluiu que nenhum dos 12 code smells analisados em seu estudo pode ser associado com o aumento de esforço na manutenção, e que o tamanho dos arquivos e classes e o número de mudanças nos arquivos são indicativos mais relevantes de esforço na manutenção de sistemas.

ABORDAGEM PROPOSTA

A abordagem proposta neste estudo tem como objetivo ser adaptável ao ambiente no qual a aplicação é desejada. Para isso são propostas algumas etapas para atingir o objetivo de previsão da necessidade de refatoração. São elas:

- Definição do ambiente; é definido qual será o ambiente, por ambiente entende-se o conjunto de sistemas e especialistas que irão participar da abordagem de apoio a decisão de refatoração. Após a definição dos sistemas são eleitos trechos de código de fonte que implementam alguma funcionalidade, para a avaliação da necessidade de refatoração.
- Coleta das métricas; são coletadas as métricas de manutenibilidade para cada trecho proposto.
- Coleta da opinião dos especialistas; são coletadas as opiniões dos especialistas sobre a necessidade de refatoração.
- Correlação entre a opinião e as métricas; são aplicados modelos de regressão linear para correlacionar a necessidade de refatoração com as métricas de manutenabilidade afim de obter a equação de previsão.
- Uso do modelo de previsão: é utilizada a equação de previsão para prever a necessidade de refatoração antes da análise dos especialistas.

Neste artigo a abordagem é realizada em uma empresa como experimento e são observados os resultados na sessão de análise de dados.

F. Ambiente para a medição

Definição do ambiente no qual a abordagem será aplicada é o primeiro passo para aplicação; este passo deve-se ao questionamento ao uso da métrica Índice de Manutenibilidade, realizado por BRAY, M. et al., pois segundo este estudo a métrica deve ter os seus valores limites calibrados de acordo com a necessidade de cada empresa.

Neste artigo à abordagem de apoio a decisão sobre refatoração ocorrera correlacionando as métricas de manutenibilidade com a decisão da refatoração em uma empresa. A empresa alvo deste estudo tem é do segmento de desenvolvimento de sistemas e tem um porte médio e fornece soluções sistêmicas a clientes do governo. Atualmente a empresa possui 15 principais produtos implementados por diferentes equipes de desenvolvimento, sendo que, destes, dez são sistemas desenvolvidos na linguagem C# e os demais nas linguagens Java, Python e Scala.

Todos os projetos desenvolvidos na linguagem C# utilizam algumas bibliotecas utilitárias em comum. As bibliotecas possuem a licença de código aberto e estão disponíveis para acesso via internet no serviço Github, o que facilita a sua utilização como objeto de análise neste trabalho.

Neste experimento, as bibliotecas utilizadas pelos sistemas da empresa e que a manutenibilidade será avaliada são:

- Dapper dot net: Ferramenta de Micro ORM (Object Relational Mapping), que facilita o mapeamento do modelo relacional do banco de dados em objetos ou a operação inversa.
- Quartz.Net: Ferramenta que realiza o agendamento e a execução de tarefas e permite o seu monitoramento.
- Autofac: Ferramenta que funciona como um contêiner de inversão de dependência, auxiliando na atribuição de propriedades ou variáveis no qual o seu tipo são interfaces previamente registradas para seus tipos concretos.
- Umbraco CMS: Um sistema de gerenciamento de conteúdo, que permite editar gerenciar e publicar páginas web, dentro de seu ambiente.

O processo de manutenção destas bibliotecas utilitárias ocorre pela proposta de alteração em um sistema de código, esta proposta ocorre pela utilização de Pull Requests (ou PR – em português, solicitação de recebimento), e é usada tipicamente no modelo de desenvolvimento colaborativo Fork e Pull.

Um Pull Request é um conjunto de alterações do código fonte, proposta por algum desenvolvedor, no repositório de controle de versões. Os PRs normalmente contêm uma funcionalidade, e são comparáveis com a versão anterior do código, permitindo a avaliação da qualidade do código produzido em relação ao código anterior.

Durante o processo de desenvolvimento no modelo colaborativo Fork e Pull os desenvolvedores implementam cada nova funcionalidade em um novo PR. Neste cenário a necessidade de refatoração pode ser avaliada para cada PR.

G. Métricas selecionadas

Para a avaliação da necessidade de refatoração são utilizadas algumas das métricas, selecionadas durante a seção III deste artigo.

As métricas foram sumarizadas na tabela I, assim como a lista de problemas que cada uma das métricas identifica.

Após a execução da análise é esperado que as métricas mais representativas para a avaliação da necessidade de refatoração sejam identificadas.

TABLE I. TABELA DE MÉTRICAS X PROBLEMAS QUE CADA MÉTRICA IDENTIFICA

Métrica		**Problemas identificados**
Complexidade ciclomática		Complexidade dos métodos\funções
Índice de Manutenibilidade		Complexidade dos métodos\funções
		Volume total do sistema
		Utilização de comentários no código
Métricas de CK	Métodos ponderados por classe	Quantidade de métodos de cada classe
		Complexidade dos métodos\funções
	Profundidade da árvore de herança	Uso excessivo de herança
	Número de filhos	Uso excessivo de herança
	Acoplamento entre objetos	Acoplamento excessivo
	Repostas para a classe	Muitos métodos são encadeados
	Falta de coesão nos métodos	Baixa coesão
Linhas de código por classe		Classes muito grandes
Linhas de código por método		Métodos\funções muito grandes

H. Ferramentas para medição

Existem diversas ferramentas que realizam a medição das métricas selecionadas para o experimento, o estudo de Novak e Rakic listou algumas das ferramentas de mercado existentes que realizam a medição da manutenibilidade para as linguagens da plataforma .NET e as relacionou com as métricas. A listagem de foi estendida e atualizada pela busca nos websites dos fornecedores resultando em uma nova listagem que pode ser verificada na tabela II. Onde CC -Complexidade Ciclomática, LOC Linhas de Código, DIT - Profundidade da Árvore de Herança, CBO-Acoplamento entre Objetos, RFC- Respostas para a classe, LCOM - Falta de coesão nos métodos, A - Abstração, TMS -Total de métodos Sobrecarregados, - TC - Total de Classes, TM -Total de Métodos, MI- Índice de Manutenibilidade e COM - Porcentagem de comentários.

TABLE II. TABELA DE MÉTRICAS X PROBLEMAS QUE CADA MÉTRICA IDENTIFICA

	CC	LOC	DIT	CBO	RFC	LCOM	A	TMS	TC	TM	MI
VS Code Metrics	X	X	X								
Ndepend	X	X	X	X	X	X		X	X	X	
Source Monitor	X	X	X	X					X	X	
Nitriq	X	X		X			X		X		
Visual Studio	X		X	X							X

A ferramenta que oferece suporte ao maior número de métricas e foi selecionada para uso no estudo é a NDepend, que é uma ferramenta comercial para realização das medições. O NDepend tem suporte a uma licença acadêmica que viabiliza a sua utilização nesta pesquisa, porém ela não fornece o valor do Índice de manutenibilidade, que pode ser medido pela IDE Visual Studio.

I. Questionário para os especialistas

Com as medidas de cada PR, o próximo procedimento é à coleta da opinião dos especialistas sobre a necessidade de refatoração.

Para a obtenção de um maior número de opiniões neste experimento, foram selecionados membros de vários times de desenvolvimento que ocupam cargos distintos. No total, foram selecionados 4 voluntários que ocupam posições de liderança e que representam as suas equipes: Um arquiteto de sistemas, um líder técnico, um gerente de qualidade e um desenvolvedor sênior.

Todos os selecionados possuem experiência com o desenvolvimento de sistemas de software e com as bibliotecas selecionadas. Neste caso, todos aceitaram participar desta pesquisa.

Para a obtenção da opinião destes especialistas sobre a necessidade de refatoração de cada trecho e de forma padronizada, será utilizado um questionário que deverá ser preenchido para cada PR.

Os principais objetivos do questionário são:

- Classificar o código como um candidato à refatoração pela atribuição de uma nota de zero a três, sendo que:
  a. Sem nenhuma refatoração indicada; neste caso, os especialistas classificam o código com o valor “0” (zero).
  b. Existe a possibilidade de refatoração de itens de pouco impacto, tornando a refatoração desnecessária; neste caso, os especialistas classificam o código com o valor “1” (um).
  c. A refatoração é recomendada para o trecho de código apresentado, proporcionando ganho significativo na manutenibilidade; neste caso, os especialistas classificam o código com o valor "2" (dois).
  d. A refatoração é altamente recomendada para o trecho de código apresentado: o código terá problemas de manutenção no futuro; neste caso, os especialistas classificam o código com o valor “3” (três).
- Justificar a não recomendação da refatoração em caso de problemas de manutenibilidade no código, devido a algum atributo de qualidade conflitante.
- Obter a percepção dos especialistas sobre, no caso de existência, os problemas de manutenção que o código apresenta; a fim de relacionar os problemas com as métricas de manutenibilidade existentes, confirmando a precisão das métricas com as suas propostas originais.
- Comentários com a opinião dos especialistas sobre a manutenibilidade do código apresentado.

O questionário proposto foi disponibilizado em: https://docs.google.com/forms/d/1TIUICnOiUe-eBSZg8Vv6f7xppPFxFwtSuO-O7AyOlJw/viewform.

J. Correlação das métricas utilizando análise estatística

Para correlacionar as métricas com as opiniões de especialistas, será utilizado o método de regressão linear, conforme descrito por DEVORE, L. J.

A técnica de regressão linear e correlação explora a relação entre duas ou mais variáveis, de modo que se possa obter informações sobre uma delas por meio dos valores conhecidos das outras.

O objetivo da regressão pode ser investigar as variáveis que estão relacionadas deterministicamente; neste caso, ao conhecer o valor de x, implica-se no conhecimento exato de y, ou de maneira não determinística (ou estatística), na qual existe a tendência dos valores de y se comportarem de acordo com os valores de x.

No modelo de regressão proposto pelo trabalho, o valor de “y” será a opinião dos especialistas sobre a refatoração, que é medida pela nota atribuída pelos especialistas, e os valores de “x” serão as métricas selecionadas pelo estudo.

Permitindo, assim, a análise dos seguintes valores:

1. Coeficiente de correlação, ou R^2, que indica até que ponto as variáveis estão relacionadas. No estudo, o quanto é a relação das métricas com a necessidade de refatoração.

2. Nível descritivo, ou valor-p, que indica o nível de significância de cada variável dentro do modelo de regressão, permitindo a análise das métricas que representam um maior impacto para a avaliação da refatoração.
3. Coeficientes da equação, que são utilizados para formar a equação que relaciona as variáveis, valores de "x", a fim de se chegar a um valor de "y". A soma das multiplicações dos coeficientes de cada variável somada pelo coeficiente e intersecção resulta na equação de previsão.

Para o cálculo da regressão neste trabalho utiliza-se R, que é um ambiente de desenvolvimento e uma linguagem para computação estatística e gráfica. A escolha da ferramenta R se deve ao seu uso no estudo de Sjøberg et al. e sua licença de código aberto.

Na aplicação dos modelos de regressão não é esperado um ajuste determinístico, e sim um ajuste aproximado.

Em alguns casos, as variáveis de "x" podem se comportar como curva ou como um polinômio de segundo grau ou maior em relação à variável "y"(Devore, 2014). Nestes casos, métodos de regressão não linear podem ser aplicados para a análise. Mas, devido à relação esperada entre as variáveis de manutenibilidade e a necessidade de refatoração, se espera uma relação polinomial de primeiro grau. Porque quanto maior o valor de uma variável "x" (ou menor, dependendo da variável em estudo) maior o valor de "y", sendo essa uma relação direta.

EXPERIMENTO E ANALISE DOS DADOS

Nesta seção são apresentados os trechos de código selecionados e o processo de medição das métricas dos trechos de código selecionados até a sua medição. Nesta seção também é descrito o processo de aplicação da regressão linear múltipla e a análise dos dados.

K. Pull Requests selecionados

Os trechos de código selecionados para este estudo são Pull Requests reais realizados nos sistemas Dapper.net, Quartz.net, Autofac e Umbraco.

Os Pull Requests foram selecionados na lista de PRs do Github de cada ferramenta. Foram escolhidos os PRs contendo alterações de classes e métodos de C#, que implementam alguma funcionalidade ou implementam alguma alteração significativa para o software. São consideradas mudanças não significativas: mudanças em código de teste, mudança em parâmetros e mudanças de grafia.

Foram selecionados 18 Pull Requests, sendo 3 PRs para a ferramenta Autofac e 5 para as demais. Foi selecionado um número menor de PRs para o Autofac devido à falta de PRs com mudanças significativas. A tabela III lista todos os PRs selecionados.

TABLE III. TABELA COM OS PULL REQUESTS AVALIADOS

Sist.	PR	Link
Dapper.net	308	https://github.com/StackExchange/dapper-dot-net/pull/308
	194	https://github.com/StackExchange/dapper-dot-net/pull/194
	188	https://github.com/StackExchange/dapper-dot-net/pull/188
	169	https://github.com/StackExchange/dapper-dot-net/pull/169
	235	https://github.com/StackExchange/dapper-dot-net/pull/235
Quar	270	https://github.com/quartznet/quartznet/pull/270
	263	https://github.com/quartznet/quartznet/pull/263

quartz.net	261	https://github.com/quartznet/quartznet/pull/261
	209	https://github.com/quartznet/quartznet/pull/209
	260	https://github.com/quartznet/quartznet/pull/260
Autofac	622	https://github.com/autofac/Autofac/pull/622
	542	https://github.com/autofac/Autofac/pull/542
	606	https://github.com/autofac/Autofac/pull/606/
Umbraco	875	https://github.com/umbraco/Umbraco-CMS/pull/875
	873	https://github.com/umbraco/Umbraco-CMS/pull/873
	870	https://github.com/umbraco/Umbraco-CMS/pull/870
	865	https://github.com/umbraco/Umbraco-CMS/pull/865
	861	https://github.com/umbraco/Umbraco-CMS/pull/861

L. Processo de medição

Para realizar a medição das métricas de Chidamber & Kemerer, será utilizada a ferramenta NDepend.

O NDepend é uma ferramenta comercial de análise de código para a plataforma .NET que extrai diversas métricas de código com possibilidade de extensão pela utilização de consultas LINQ ou pela utilização de sua API.

A análise do NDepend por padrão não fornece os valores das métricas de CK; para isso, é necessária a utilização de consultas LINQ personalizadas. Para simplificar a análise dos dados, foi construída uma ferramenta denominada MetricUtil, que usa a API do NDepend para analisar o código, executando consultas LINQ e salvando as medidas de cada trecho de código em uma planilha xls.

A ferramenta está disponível no link sob licença de código aberto: https://github.com/brunonascimento/metricsUtil

A fim de realizar a medição das demais métricas do estudo, como a Complexidade Ciclomática, Índice de manutenibilidade, Linhas de Código da Classe e Linhas de Código do Método, foi utilizada a ferramenta Code Metrics disponível no próprio IDE da plataforma .NET, o Visual Studio.

Após colher e tabular os dados das métricas para os 18 Pull Requests nas duas ferramentas, Visual Studio Code Metrics e MetricsUtil, foram excluídas da análise as classes e métodos que não foram alterados pelo PRs, gerando a tabela IV com as medidas de cada PR por classe.

TABLE IV. MÉTRICAS POR PULL REQUEST

	PR	NOC	CBO	DIT	LCOM	WMC	RFC	CC	IM	LOC Class	LOC Meth
Dapper.Net	308	0	32	0	0	96	3	572	56	1281	3
	194	0	33	0	0	95	1	551	56	1227	1
	188	0	30	0	0	92	3	548	56	1208	53
	169	0	30	0	0	119	1	577	58	1258	3
	235	0	32	0	0	131	2	604	58	1373	7

Quartz.Net	263	5	47	2	0,9675	112	20	388	60	1086	21
	263	0	20	1	0,7272	10	24	38	60	117	69
	261	0	48	0	0,9411	102	12	346	68	679	15
	261	0	19	1	0,7670	10	24	81	58	195	104
	261	2	60	1	0,9803	125	3	573	65	1182	4
	209	0	48	0	0,943	102	7	344	68	674	28
	209	0	51	0	0,7647	11	85	130	48	417	324
	209	0	32	0	0,8529	57	1	337	60	665	1
	260	0	48	0	0,9438	102	2	346	68	679	1
	260	0	5	0	0,7407	8	0	55	70	121	8
Autofac	622	0	23	0	0,9354	27	8	57	75	115	11
	622	0	3	0	0	2	4	21	57	24	23
	622	0	9	0	0	53	1	69	80	130	3
	542	0	12	0	0,3333	1	13	4	83	7	3
	606	0	25	1	0,889	18	4	61	72	117	7
	606	1	9	1	0,65	15	5	64	67	118	21
	606	0	14	0	0,3333	1	11	4	83	7	3
	606	0	4	0	0,7176	10	1	28	81	48	6
Umbraco	861	0	70	6	0	15	33	60	62	151	4
	875	0	20	3	0	5	14	20	62	53	28
	873	0	6	1	0,6666	5	2	20	82	37	2
	870	0	18	2	0,6428	3	44	11	56	40	33
	865	0	90	6	0,8809	32	22	135	55	464	17
	865	0	78	1	0,8921	133	38	415	61	1231	50
	865	0	70	6	0	21	12	109	57	258	13

M. Aplicação da Regressão Linear

Como primeira análise, foi aplicada uma regressão com todas as métricas colhidas e a média da opinião dos especialistas.

O modelo apresentou um coeficiente de correlação ajustado (ou R^2 ajustado) de 0,5952, o que indica que as variáveis de manutenibilidade impactam em 59,52% o valor da necessidade de refatoração. Observam-se também os valores-P, que indicam o quanto cada variável impacta a necessidade de refatoração. A fim de simplificar a regressão, foram gerados mais dois modelos, excluindo as métricas menos significativas.

Considerando que a regressão aplicada tem nível de significância de 5%, padrão para regressões na ferramenta R, são selecionadas as variáveis cujo valor de P representa um valor menor que 5%, pois estas

são as mais significativas. Assim, conforme pode ser observado na tabela V, gera-se o modelo 03. No Modelo 02, foram incluídas as métricas de NOC e CBO, pois apresentam um p-valor mais próximo ao nível de significância.

TABLE V. VALOR-P PARA O MODELO DE REGRESSÃO 1

	valor-P
NOC	**11,31%**
CBO	**9,89%**
DIT	33,20%
LCOM	32,85%
WMC	**3,02%**
RFC	83,32%
CC	44,55%
IM	45,08%
LOC_Class	**4,83%**
LOC_Meth	26,93%

O modelo 02, contendo as métricas NOC, CBO, WMC, LOC Classe, e o Modelo 03, contendo as métricas WMC e LOC Classe.

Aplicando as regressões para o Modelo 02 com as métricas NOC, CBO, WMC e LOC Classe e para o Modelo 03 – com as métricas WMC e LOC Classe.

Observa-se que o modelo 02 apresenta um coeficiente de correlação ajustado de 0,6235, enquanto o modelo 03 mostra um coeficiente de correlação ajustado de 0,43, o que indica que o modelo que mais se adequa é o modelo 2. Indicando, assim, que as métricas mais significativas para os especialistas selecionados nos PRs analisados foram o NOC, CBO, WMC e LOC Classe.

N. *Avaliação da concordancia entre especialistas*

Para avaliar a concordância entre especialistas foi utilizado o coeficiente kappa. Kappa é uma medida de concordância entre observadores, e mede o grau de concordância além do que seria esperado somente pelo acaso.

No caso de dois observadores o índice Kappa de Cohen pode ser aplicado; para múltiplos observadores, a medida de Fleiss Kappa.

Os valores de Kappa, Cohen e Fleiss, podem ser calculados pela ferramenta R a partir da opinião dos especialistas. Os valores de kappa de Fleiss resultam num coeficiente de 0.208.

Segundo LANDIS, J. R. e KOCH, G. G. (1977), que sugerem a interpretação do valor de Kappa de acordo com a tabela VII, conclui-se que, enquanto alguns especialistas não apresentam concordância nenhuma, outros apresentam uma concordância quase perfeita. Mas, avaliando os especialistas em conjunto, eles apresentam uma concordância razoável.

TABLE VI. INTEPRETAÇÃO PARA OS VALORES DE KAPPA DE FLEISS

<0	Sem concordância
0-0.19	Concordância baixa
0.20-0.39	**Concordância razoável**
0.40-0.59	Concordância moderada
0.60-0.79	Concordância substancial
0.80-1.00	Concordância quase perfeita

ANÁLISE DOS DADOS

Conforme se pode observar pela coleta da opinião dos especialistas, estes, em conjunto,

apresentaram uma concordância razoável de opinião, mas não apresentaram um valor de concordância muito alto, colocando em dúvida a qualidade da avaliação destes especialistas, assim como foi apontado por trabalhos anteriores(Anda, 2007). Por utilizar especialistas da mesma empresa, esperava-se inicialmente que a concordância entre os especialistas fosse maior.

A análise dos dados dos especialistas não excluiu nenhum dos PRs avaliados, nem mesmo aqueles que apresentaram uma maior divergência de opinião. Um dos fatores que podem ter afetado na divergência de opinião foi a avaliação ter sido feita pelos especialistas de modo independente. Algumas técnicas de avaliação conjunta podem ser estudadas em trabalhos futuros com o intuito de diminuir este desvio.

A análise de regressão no modelo 02 foi a que apresentou o maior coeficiente de correlação, apresentando um ajuste em 62,35%. Observa-se que as métricas mais significativas foram NOC, CBO, WMC e LOC Classe, sendo que as métricas WMC e LOC Classe estão associadas claramente com o tamanho de cada classe.

O CBO representa o número de classes acopladas com outra classe, que tem uma relação direta com o tamanho da classe; e o NOC representa o número de filhos de uma classe que está associado com a implementação de herança do modelo de orientação a objetos.

Observa se que, na questão 3 do questionário, identificam-se os problemas apontados pelos especialistas. Os problemas apontados pelos especialistas foram:

- Classes muito grandes, com 28 PRs apontados (59,6%)
- Muitos métodos por classe, com 16 PRs apontados (34%)
- O código possui tamanho\volume muito alto, com 15 PRs apontados (31,9%)
- Métodos funções muito grandes, com 13 PRs apontados (27,7%)
- Outros, com 12 PRs apontados (12,8%)
- Falta de comentários no código, com 3 PRs apontados (6,4%)
- Muitos métodos encadeados, com 3 PRs apontados (6,4%)

Os problemas a, b, c e d podem ser associados diretamente ao tamanho das classes e métodos, assim como as métricas LOC Classe e WMC (métodos ponderados por classe), que também foram as mais significativas para os modelos de regressão.

Indiretamente, também podemos associar a métrica CBO (acoplamento entre objetos) com o tamanho das classes, pois, quanto maior uma classe, maior o seu acoplamento.

Apenas a métrica NOC (número de filhos) foi apontada como significativa pela regressão, mas não foi identificada pelos especialistas.

Na questão 4 do questionário, na qual os especialistas poderiam comentar sobre a manutenibilidade, apareceram duas menções à herança que podem ser associadas à métrica NOC.

O resultado da aplicação da regressão linear identificou o modelo mais adequado, o modelo 2, com coeficiente de correlação ajustado de 0,6235, ou seja, capaz de prever a refatoração em 62,35% dos casos (para estes especialistas, nestes PRs avaliados), tornando possível a escrita da equação de previsão da necessidade de refatoração, através da multiplicação dos coeficientes pelas métricas, resultando na equação.

$$ref = -0{,}23707 * NOC + 0{,}009129 * CBO - 0{,}01251 * WMC + 0{,}001821 * LOC\ Classe + 0{,}244452 \quad (1)$$

No total das 30 avaliações, 30 classes e 18 PRs, 4 avaliações (ou 13,33%) mudaram a categoria da refatoração.

CONCLUSÃO

Neste trabalho, propôs-se o desenvolvimento de uma abordagem para avaliar se a refatoração é recomendada para trechos de código de fonte utilizando métricas de manutenibilidade. Foram apresentados

os principais conceitos envolvidos sobre a manutenibilidade e sobre cada uma das métricas listadas na seção III- Métricas de Manutenabilidade.

A abordagem proposta se se baseia em trabalhos pré-existentes e em métricas de manutenibilidade que podem ser medidas por uma variedade de ferramentas e algoritmos. Porém, o benefício ao se utilizar a abordagem desenvolvida é a definição de valores máximos para as métricas, permitindo automatização na tarefa dos especialistas na hora de avaliar a necessidade de refatoração.

O esforço de coleta de métricas foi maior que o esperado, mas a análise dos valores utilizando regressões lineares com a ferramenta R se mostrou eficaz.

Para os especialistas selecionados, o trabalho mostrou que existe um grau de concordância razoável entre os especialistas de uma mesma empresa, mas este grau de concordância é muito próximo a baixa concordância o que é um valor menor que o esperado para especialistas que trabalham dentro da mesma empresa. Para isto, técnicas de análise de código em conjunto podem ser estudadas em trabalhos futuros.

A análise dos dados utilizando as regressões lineares mostrou que as métricas que mais impactam na necessidade de refatoração estão relacionadas ao tamanho das classes, WMC e LOC Classe, seguidos pelo CBO e NOC, o que é um resultado que condiz com os resultados do trabalho de Sjøberg et. at..

A equação encontrada pela análise das regressões lineares indicou que a fórmula pode prever a necessidade de refatoração em 62,35% dos casos, sendo que, aplicada aos 18 PRs em 4 sistemas, esta fórmula foi capaz de prever a necessidade em 86,67% dos casos, o que é um resultado promissor.

Alguns itens importantes que não foram abordados neste trabalho poderão ser temas de outros trabalhos de pesquisas, como:

- Técnicas de avaliação de especialistas em conjunto;
  Utilizando a equação de kappa de Fleiss, e segundo a interpretação de Ladis, conclui-se que a concordância entre os especialistas é razoável, abaixo de boa e superior a baixa, mas com um valor muito próximo à concordância baixa.
  Logo, faz-se necessário o estudo de técnicas para aumentar a concordância entre especialistas sobre a manutenibilidade e sobre a avaliação da necessidade de refatoração.
- Análise da necessidade de refatoração utilizando aprendizado de máquina; A aplicação da abordagem resultou em uma equação capaz de prever a necessidade de refatoração para 62,35% dos casos utilizando apenas análises estatísticas pela aplicação de regressões lineares. Porém, outra abordagem de análise que poderia ser aplicada em trabalhos futuros seria a utilização de algoritmos de aprendizados de máquina para prever a necessidade de refatoração.
- Aplicação da abordagem em outras empresas e linguagens;
  Como experimento deste estudo, a abordagem foi aplicada em uma determinada empresa com o auxílio de quatro especialistas, que avaliaram quatro sistemas desenvolvidos na linguagem C#.
  Porém, resultados de ambientes mais diversos com a finalidade de comparação e ainda com a possibilidade de encontrar uma equação com valores mais gerais, de modo que seja possível a aplicabilidade independentemente da empresa.

REFERENCIAS BIBLIOGRÁFICAS

ANDA,B. Assessing Software System Maintainability Using Structural Measures and Expert Assessments, Proc IEEE Int'l Conf. Software Maintenance, 2007, p. 204-213.

BARBOSA JUNIOR, N. Avaliação da Manutenibilidade de Software Através das Métricas de C&K. 2011. 108 f. Dissertação (Mestrado), IPT - Instituto de Pesquisas Tecnológicas do Estado de São Paulo - IPT, São Paulo, Agosto,2011.

BECK, B. K.; DATE, P. Test-Driven Development By Example Test-Driven Development By Example. 2002, p. 138.

BOEHM, B.; BASILI, V. R. Software Defect Reduction Top 10 List. Computer, v. 34, 2001, p. 135–137.

BRAY, M. et al. C4 Software Technology Reference Guide — A Prototype. Software Engineering Institute - Carnegie Mellon University, Jan. 1997, p. 426.

CHAPARRO, O. et al. On the Impact of Refactoring Operations on Code Quality Metrics. IEEE International Conference on Software Maintenance and Evolution 2014, 2014, p. 456–460.

CHIDAMBER, S.R.; KEMERER C. F., C.F. A Metrics Suite for Object Oriented Design. IEEE Transactions on Software Engineering, 20: 1994, p. 476-493.

DEMARCO, T. Controlling Software Projects – Management, Mensurament and Estimates, Englewood Cliffs: [s.n.], 1986.

DEVORE, L. J. Probabilidade e estatística para engenharia e ciências, 8. ed. São Paulo, Cengage Learning, p. 633, 2014.

FENTON, N. E.; PFLEEGER, S. L. Software Metrics: A Rigorous and Practical Approach, 2. ed. Boston, PWS Publishing Company, p. 638, 1997.

FOWLER, M. et al. Refactoring Improving the Design of Existing Code, 1. ed. Boston, The Addison-Wesley Object Technology Series, p. 464, 1999.

GOMES DA SILVA, J. Uma técnica de avaliação da aplicabilidade da Refatoração em Software Orientado a Objetos com base nas métricas de Chidamber e Kemerer. São Paulo, 2012. 97 f. Dissertação (Mestrado Profissional Engenharia de Computação – Engenharia de Software) - Coordenadoria de Ensino Tecnológico, Instituto de Pesquisas Tecnológicas do Estado de São Paulo, São Paulo, 2012.

ISO/IEC/IEEE 25010-2011 Systems and software engineering , ANSI/IEEE Std 25010-2011 - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models. 2011, p. 34.

MCCABE, T. J.; A Complexity Measure, IEEE Transactions on Software Engineering , VOL. SE-2, NO. 4, 1976, p. 308–320.

NBR/ISO 9001-2000 - Sistemas de gestão da qualidade – Requisitos. 2000, p. 30.

NOVAK, J.; RAKIC, G. COMPARISON OF SOFTWARE METRICS TOOLS FOR . NET, 13th International Multiconference Information Society (IS), Ljubljana, Slovenia, out. 2010, p. 231–234.

OMAN, P.; COLEMAN, D.; et al. Using metrics to evaluate software system maintainability, IEEE Computer, v. 27, 1994, p. 44 - 49.

OMAN, P.; HAGEMEISTER, J. Metrics for assessing a software system's maintainability, Proceedings Conference on Software Maintenance 1992, 1992, p. 337 - 344.

OPDYKE, F. W. Refactoring object-oriented frameworks. 1992. Tese (Doutorado) - Graduate College of the University of Illinois, Urbana-Champaign,1992, p. 197.

PRESSMAN, R. S. Software Enginnering: a practitioner's approach. McGraw-Hill Higher Education, 7 ed., p. 928, 2009.

SJØBERG, D. I. K., YAMASHITA, A., ANDA, B. C. D., MOCKUS, A., DYBÅ, T. Quantifying the Effect of Code Smells on Maintenance Effort, IEEE Transactions on Software Engineering, ago.2013, p. 1144–1156.

YAMASHITA, A.; MOONEN, L. Do code smells reflect important maintainability aspects?, Software Maintenance (ICSM), 2012 28th IEEE International Conference on,2012, p. 306 – 315.

YAMASHITA, A. How good are code smells for evaluating software maintainability? Results from a

comparative case study. IEEE International Conference on Software Maintenance, ICSM,2013, p. 566–571

GITHUB - Using pull requests - User Documentation. Disponível em: <https://help.github.com/articles/using-pull-requests/>. Acesso em: 20 jun.2015.

R - The R Project for Statistical Computing. Disponível em: <https://www.r-project.org/>. Acesso em: 27 jul 2001

LANDIS, J. R.; KOCH, G. G. The measurement of observer agreement for categorical data. Biometrics, v. 33, n. 1, p. 159–174, 1977.