Pontifícia Universidade Católica
do Rio de Janeiro

Alexandre Leite Silva

Reuso de Estratégias Sensíveis a Domínio para Detecção de Anomalias de Código

Um Estudo de Múltiplos Casos

Dissertação de Mestrado

Dissertação apresentada ao Programa de Pós-Graduação em Informática da PUC-Rio como requisito parcial para obtenção do título de Mestre em Informática.

Orientador: Prof. Alessandro Fabrício Garcia
Co-Orientador: Elder José Reioli Cirilo

Rio de Janeiro
Agosto de 2013

PUC-Rio - Certificação Digital Nº 1112618/CA

PONTIFÍCIA UNIVERSIDADE CATÓLICA
DO RIO DE JANEIRO

Alexandre Leite Silva

Reuso de Estratégias Sensíveis a Domínio para Detecção de Anomalias de Código

Um Estudo de Múltiplos Casos

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós-Graduação em Informática da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Alessandro Fabrício Garcia
Orientador
Departamento de Informática - PUC-Rio

DSc. Elder José Reioli Cirilo
Co-Orientador
Departamento de Informática - PUC-Rio

Prof.ª Simone Diniz Junqueira Barbosa
Departamento de Informática - PUC-Rio

Prof. Eduardo Magno Lages Figueiredo
UFMG

Prof. José Eugenio Leal
Coordenador Setorial do Centro
Técnico Científico - PUC-Rio

Rio de Janeiro, 8 de agosto de 2013

PUC-Rio - Certificação Digital Nº 1112618/CA

Todos os direitos reservados. É proibida a reprodução total ou parcial do trabalho sem autorização da universidade, do autor e do orientador.

Alexandre Leite Silva

Alexandre iniciou sua carreira em 1998, em Dom Pedrito-RS, como Técnico em Processamento de Dados. Em 2002, foi bolsista de iniciação científica na UFPel-RS. Em 2004, mudou-se para o Rio de Janeiro, onde desenvolveu-se profissionalmente no mercado de desenvolvimento de sistemas, trabalhando em fábricas de software, empresa do ramo varejista e de serviços. Também trabalhou no Laboratório de Engenharia de Software - LES, na PUC-Rio, onde concluiu a graduação (2011) e o mestrado (2013). Atualmente, lidera uma equipe de desenvolvimento de sistemas de análise de risco financeiro, trabalhando com métodos ágeis, desenvolvimento em Java para Web e *gamification*.

Ficha Catalográfica

Silva, Alexandre Leite

Reuso de estratégias sensíveis a domínio para detecção de anomalias de código Um estudo de múltiplos casos / Alexandre Leite Silva ; orientadores: Alessandro Fabrício Garcia, Elder Reioli Cirilo. – 2013.

84 f. : il. ; 30 cm

Dissertação (mestrado)–Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2013.

CDD: 004

PUC-Rio - Certificação Digital Nº 1112618/CA

Agradecimentos

Agradeço à Dani, pelo incentivo e apoio incondicional, e à Sofia, pela vida de contos de fadas, príncipes e Peter Pan.

À PUC-Rio, pelos auxílios concedidos, sem os quais este trabalho não poderia ter sido realizado.

Aos amigos, que sempre ajudaram quando eu precisei.

Ao prof. Alessandro, pela generosidade em apoiar esse trabalho, e ao Elder, pela cumplicidade.

Ao prof. Lucena, pela oportunidade de estudar e trabalhar no LES durante o mestrado, e à Vera, que é uma grande amiga.

Aos meus pais e familiares, que, mesmo longe, sempre estiveram por perto. E, finalmente, à Deus, pela oportunidade de realizar esse trabalho.

PUC-Rio - Certificação Digital Nº 1112618/CA

Resumo

Silva, Alexandre Leite; Garcia, Alessandro Fabrício (Orientador); Cirilo, Elder José Reioli (Co-Orientador). **Reuso de Estratégias Sensíveis a Domínio para Detecção de Anomalias de Código Um Estudo de Múltiplos Casos.** Rio de Janeiro, 2013. 84p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Para promover a longevidade de sistemas de software, estratégias de detecção são reutilizadas para identificar anomalias relacionadas a problemas de manutenção, tais como classes grandes, métodos longos ou mudanças espalhadas. Uma estratégia de detecção é uma heurística composta por métricas de software e limiares, combinados por operadores lógicos, cujo objetivo é detectar um tipo de anomalia. Estratégias pré-definidas são usualmente aplicadas globalmente no programa na tentativa de revelar onde se encontram os problemas críticos de manutenção. A eficiência de uma estratégia de detecção está relacionada ao seu reuso, dado o conjunto de projetos de uma organização. Caso haja necessidade de definir limiares e métricas para cada projeto, o uso das estratégias consumirá muito tempo e será negligenciado. Estudos recentes sugerem que o reuso das estratégias convencionais de detecção não é usualmente possível se aplicadas de forma universal a programas de diferentes domínios. Dessa forma, conduzimos um estudo exploratório em vários projetos de um domínio comum para avaliar o reuso de estratégias de detecção. Também avaliamos o reuso de estratégias conhecidas, com calibragem inicial de limiares a partir do conhecimento e análise de especialistas do domínio. O estudo revelou que, mesmo que o reuso de estratégias aumente quando definidas e aplicadas para um domínio específico, em alguns casos o reuso é limitado pela variação das características dos elementos identificados por uma estratégia de detecção. No entanto, o estudo também revelou que o reuso pode ser significativamente melhorado quando as estratégias consideram peculiaridades dos interesses recorrentes no domínio ao invés de serem aplicadas no programa como um todo.

Palavras-chave

Anomalias; detecção; reúso; acurácia

PUC-Rio - Certificação Digital Nº 1112618/CA

Abstract

Silva, Alexandre Leite; Garcia, Alessandro Fabrício (Advisor); Cirilo, Elder José Reioli (Co-advisor). **Reuse of Domain-Sensitive Strategies for Detecting Code Anomalies A Multi-Case Study.** Rio de Janeiro, 2013. 84p. MSc. Dissertation – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

To prevent the quality decay, detection strategies are reused to identify symptoms of maintainability problems in the entire program. A detection strategy is a heuristic composed by the following elements: software metrics, thresholds, and logical operators combining them. The adoption of detection strategies is largely dependent on their reuse across the portfolio of the organizations software projects. If developers need to define or tailor those strategy elements to each project, their use will become time-consuming and neglected. Nevertheless, there is no evidence about efficient reuse of detection strategies across multiple software projects. Therefore, we conduct an industry multi-project study to evaluate the reusability of detection strategies in a critical domain. We assessed the degree of accurate reuse of previously-proposed detection strategies based on the judgment of domain specialists. The study revealed that even though the reuse of strategies in a specific domain should be encouraged, their accuracy is still limited when holistically applied to all the modules of a program. However, the accuracy and reuse were both significantly improved when the metrics, thresholds and logical operators were tailored to each recurring concern of the domain.

Keywords

Anomalies; detection; reuse; accuracy

PUC-Rio - Certificação Digital Nº 1112618/CA

Sumário

PUC-Rio - Certificação Digital Nº 1112618/CA

PUC-Rio - Certificação Digital Nº 1112618/CA

Lista de figuras

PUC-Rio - Certificação Digital Nº 1112618/CA

Lista de tabelas

PUC-Rio - Certificação Digital Nº 1112618/CA

Lista de métricas de software

AM: Accessed Methods

ATFD: Access to Foreign Data

CBO: Coupling Between Objects

LOC: Lines of Code

MaxNesting: Maximun Nesting Level

CC: McCabe's Cyclomatic Number

NOAV: Number of Accessed Variables

NOM: Number of Methods

TCC: Tight Class Cohesion

WMC: Weighted Method Count

PUC-Rio - Certificação Digital Nº 1112618/CA

1
Introdução

Na medida em que sistemas de software sofrem mudanças, problemas estruturais podem ser introduzidos no código fonte. Em geral, as atividades de manutenção que não são realizadas de maneira disciplinada, acabam por trazer algumas consequências negativas para as atividades subsequentes de manutenção e evolução de um programa. Os problemas estruturais que decorrem dessa falta de cuidado são chamados de anomalias de código, ou popularmente de *bad smells* (Fowler, 1999). Segundo estudos empíricos, módulos de programas que apresentam tipos de anomalias recorrentes, tais como métodos longos e classes que promovem mudanças em cascata[1], são geralmente as fontes de declínio da qualidade do sistema.

Exemplos de consequências negativas em tais módulos anômalos são: (i) introdução de falhas por programadores durante manutenção destes (Macía et al., 2011), (Olbrich et al., 2009), (Khomh et al., 2009), e (ii) sintomas de degeneração de projeto, ou seja, violações de princípios de projeto (Macía et al., 2011), (Macía et al, 2012), (Lozano et al., 2007). No primeiro caso, a introdução de falhas é promovida devido à complexidade inerente ao módulo anômalo. Devido à dificuldade de compreensão do código, programadores acabam introduzindo falhas de forma inadvertida (Macía et al., 2011). No segundo caso, as anomalias de código são frequentemente materializações de violações de princípios básicos de projeto de software, tais como interface simples, baixo acoplamento e alta coesão (Macía et al, 2011), (Macía et al, 2012).

Quando essas anomalias não são devidamente identificadas e removidas do código, frequentemente ocorre a degradação parcial ou total do sistema (Hochstein e Lindvall, 2005). Existe um catálogo com vários tipos de anomalias (Fowler, 1999) e, portanto, a detecção destas depende fortemente de uma abordagem automatizada, para identificação de instâncias de cada tipo de

[1] Do inglês: *Shogun Surgery (Fowler, 1999)*

PUC-Rio - Certificação Digital Nº 1112618/CA

anomalia. Além disso, à medida que um sistema cresce durante o seu ciclo de vida, identificar anomalias de código manualmente fica ainda mais difícil ou impeditivo (Macía et al., 2012). A partir das anomalias propostas por (Fowler, 1999), surgiram abordagens para apoiar a automação do processo de detecção de anomalias de código. Essa automação é, em geral, baseada na coleta e observação de métricas estáticas dos elementos de um programa (Fenton e Neil, 2000), (Lanza e Marinescu, 2006). A partir das métricas, é possível quantificar os atributos dos elementos do código fonte, como, por exemplo, acoplamento (Chidamber e Kemerer, 1994), coesão (Bieman e Kang, 1995) e complexidade ciclomática (McCabe, 1976). A partir dos valores medidos, é possível identificar uma relação entre características dos elementos de código e um sintoma de uma anomalia. Por exemplo, classes grandes[2] podem concentrar muito conhecimento sobre o sistema, o que dificulta o entendimento e, consequentemente, a sua manutenção. Da mesma forma, métodos muito grandes e complexos podem dificultar a atividade de manutenção e a evolução dos sistemas. Foi observado em um estudo recente que cerca de 70% das instâncias de classes grandes e métodos longos, detectados automaticamente, estão relacionados com sintomas de degeneração de projeto (Macía et al., 2012).

Através da relação entre múltiplas métricas e cada tipo de anomalia, é possível definir uma estratégia de detecção para apoiar a descoberta de instâncias de anomalias automaticamente em um programa (Marinescu, 2004) e (Lanza e Marinescu, 2006). Uma estratégia de detecção é uma condição composta por métricas e limiares, combinados através de operadores lógicos. Através desta condição é possível filtrar um conjunto específico de elementos do programa. Este conjunto de elementos de código representa os candidatos a anomalias de código nocivas a manutenibilidade do sistema (Lanza e Marinescu, 2006). Considerando que, à medida que um sistema cresce, identificar anomalias manualmente fica ainda mais difícil ou impeditivo, surgiram as ferramentas de detecção automática de anomalias. Algumas delas são: *JDeodorant* (Tsantalis et al., 2008), PMD, *iPlasma*, *inFusion* e *Stench Blossom* (Murphy-Hill e Black, 2010), e grande parte das ferramentas propostas é baseada nas estratégias de detecção propostas por (Lanza e Marinescu, 2006).

[2] Do inglês: *God Classes (Fowler, 1999)*

PUC-Rio - Certificação Digital Nº 1112618/CA

Mesmo com o apoio de ferramentas, o caráter subjetivo das definições de anomalias pode levar "vários caminhos" para a definição de estratégias de detecção. As metáforas propostas para identificar as anomalias podem ser compreendidas de várias maneiras e, consequentemente, diferentes combinações de métricas podem ser escolhidas. Por exemplo, para detectar uma classe grande, que envolve muitas responsabilidades, uma estratégia pode ser baseada em atributos de tamanho, como, por exemplo, número de linhas da classe e o número de métodos, e atributos de coesão entre os métodos da classe, além do acoplamento com outras classes, por exemplo. Ademais, pode ser definido que a violação do limite associado com apenas uma métrica ou com um subconjunto arbitrário destas métricas pode ser necessários para que uma anomalia seja detectada.

A situação é ainda mais complicada por fatores adicionais na definição de uma estratégia. Primeiro, existe mais de uma métrica para cada um destes atributos. Acoplamento, em particular, pode ser computado através de métricas de acoplamento aferente (*fan-in*) (Henry e Kafura, 1981), acoplamento eferente (*fan-out*) (Henry e Kafura, 1981) ou acoplamento por herança (*depth of inheritance* - DIT) (Chidamber e Kemerer, 1994), dentre várias outras (Harrison et al., 1998), (Poshyvanyk e Marcus, 2006). Segundo, a escolha de limiares também permite várias possibilidades. Qualquer mudança sutil nos limiares das métricas de uma estratégia representa um grande impacto no conjunto de elementos candidatos a anomalias. Dessa forma, definir limiares apropriados para as métricas de uma estratégia de detecção é determinante para encontrar um conjunto de elementos que estejam relacionados de forma coerente com as anomalias que se deseja evitar (Moha et al, 2009), (Fontana et al., 2011).

Finalmente, apesar de todo o cuidado para definir uma estratégia de detecção, não são todos os candidatos a anomalias que precisam ser removidos. Alguns destes candidatos detectados não são necessariamente sintomas de problemas consideráveis para futuras manutenções ou evoluções do sistema. Por exemplo, uma classe relacionada a uma biblioteca gráfica pode ser identificada como uma *God Class* devido à grande quantidade de métodos. Nesse sentido, apesar da classe identificada possuir muitos métodos, é possível que os métodos

PUC-Rio - Certificação Digital Nº 1112618/CA

desta classe estejam implementando interesses[3] (funcionalidades) periféricos ao sistema. Dessa forma, mesmo que essa classe seja detectada por uma estratégia de detecção, ao considerar os interesses do domínio do projeto envolvido, remover essa anomalia pode ser custoso e não ser benéfico o suficiente para a longevidade do projeto de software (Fontana et al., 2011), (Fontana et al., 2012).

1.1.Motivação e Problema

Apesar do apoio de ferramentas, definir estratégias para detecção eficaz de anomalias de código é difícil e custoso (Fontana et al., 2011). Em particular, escolher métricas e definir limiares apropriados de uma estratégia de detecção é determinante para encontrar um conjunto de elementos que estejam relacionados, de fato, com as anomalias que se deseja evitar. Dessa forma, podemos considerar que o resultado desejável de uma estratégia de detecção é formado por muitos elementos que representam instâncias de anomalias e poucos elementos que não são necessariamente sintomas de problemas.

De forma a ser viável, a utilidade de uma estratégia de detecção não pode ser restrita para cada sistema de software em avaliação. A definição de uma estratégia de detecção deveria ser útil para um conjunto de projetos. Caso contrário, os desenvolvedores precisariam definir uma estratégia de detecção para cada tipo possível de anomalia, em cada projeto. Para isso, seria preciso rever as métricas e limiares apropriados, além das ocorrências identificadas pelas ferramentas que não representam necessariamente problemas no código. Para atingir um resultado satisfatório, essa tarefa, ao ser executada especificamente para cada projeto e cada estratégia de detecção, vai demorar bastante tempo. Muito provavelmente, o tempo gasto com essa tarefa pode atrapalhar outras atividades da própria manutenção do código. Dessa forma, definir uma estratégia de detecção para cada sistema de um portfólio de projetos é uma tarefa que fatalmente seria negligenciada. A partir desse cenário, percebemos que a eficiência de uma estratégia de detecção de anomalias está relacionada: primeiro, à baixa ocorrência de elementos que não são necessariamente sintomas de problemas e, segundo, à facilidade do seu reúso. Por outro lado, é irrealista

[3] Do inglês: *concerns*.

PUC-Rio - Certificação Digital Nº 1112618/CA

esperar que estratégias de detecção de anomalias sejam úteis para todos os projetos de software. Portanto, é importante estudar quais fatores influenciam no reúso e eficácia de estratégias de detecção. De posse desse conhecimento, desenvolvedores poderiam definir estratégias de detecção que são eficazes para um conjunto de projetos de software. Porém, existe pouco conhecimento empírico sobre os fatores que devem ser considerados por desenvolvedores na definição de estratégias que sejam eficazes e que possam ser reutilizadas em vários projetos (Fontana et al., 2011) (Fontana et al., 2012). Em particular, pouco se sabe se o domínio do projeto de software permite a definição de estratégias de detecção que sejam reutilizáveis; além disso, pouco se sabe como as informações relativas ao conjunto de interesses do domínio de um sistema podem influenciar a definição das estratégias de detecção. Exemplos de domínios de sistemas de software são: análise do mercado financeiro, clusterização, sistemas de gerência de bancos de dados, entre outros. Exemplos de interesses são: interface, persistência, regras de negócio, segurança e funcionalidades auxiliares, entre outros.

Para definir estratégias de detecção que sejam eficazes e que possam ser reutilizadas em vários projetos, alguns fatores podem ser considerados. Por exemplo, para promover o reúso de estratégias de detecção para vários projetos, podemos considerar um ambiente de desenvolvimento, em que: (i) os programadores seguem práticas comuns e bem definidas, (ii) além de considerar características que definem similaridades estruturais entre os projetos de um mesmo domínio. Quando consideramos fatores como esses durante a definição de uma estratégia de detecção, podemos esperar que as estruturas de código que representam uma instância de uma anomalia possam se repetir, nos vários projetos do mesmo domínio de aplicações. Dessa forma, ao considerar e estudar fatores relacionados ao domínio dos sistemas em que se busca encontrar anomalias, podemos analisar a possibilidade de promoção de maior reúso das estratégias de detecção em sistemas do mesmo domínio.

1.2. Limitações de Trabalhos Relacionados

Em 2006, Lanza e Marinescu apresentaram um processo para a formação de estratégias de detecção de anomalias. além de estratégias de detecção para onze anomalias de código. Nesse trabalho, são descritas em detalhes como os autores

PUC-Rio - Certificação Digital Nº 1112618/CA

escolheram as métricas e limiares para cada estratégia de detecção apresentada. Além disso, também são apresentados os detalhes de cada sintoma de problema que se deseja evitar, relacionados com as métricas escolhidas. Mesmo que as onze estratégias de detecção apresentadas sejam resultado de uma ampla análise estatística, não existe uma avaliação sobre o grau de reúso dessas estratégias quando aplicadas em vários sistemas. Além disso, não há o indicativo sobre as características específicas de um grupo de sistemas que possam influenciar a definição das métricas de cada estratégia de detecção. O mesmo pode ser dito também para a definição dos limiares associados a cada métrica.

Em 2010, Guo, Seaman, Zazworka e Shull, relataram um estudo de caso sobre a adaptação de estratégias de detecção de anomalias a partir da opinião de especialistas do domínio. Nesse estudo, a partir da lista de ocorrências identificadas pelas estratégias de detecção, um grupo de especialistas do domínio foi reunido para analisar os resultados. Dessa forma, os especialistas classificaram cada uma das ocorrências entre: um sintoma de problema que realmente pode resultar em uma falha no sistema ou uma ocorrência que pertence a uma parte estável do código e, portanto, não representa um sintoma de problema. A partir da opinião dos especialistas, os limiares das estratégias foram alterados para que as estratégias de detecção pudessem ser novamente aplicadas aos sistemas, com o objetivo de diminuir o número de ocorrências que não representam sintomas de problemas. Esse estudo, apesar de apresentar uma boa sugestão de protocolo para o uso da opinião de especialistas, apresenta o resultado da avaliação das estratégias de detecção para apenas um sistema. Além disso, o trabalho não descreve quais as características do sistema escolhido e não propõe qualquer análise sobre o reúso da estratégia adaptada em outros sistemas.

PUC-Rio - Certificação Digital Nº 1112618/CA

1.3. Objetivos e Questões de Pesquisa

A partir do que foi apresentado nas subseções anteriores, motivamos a condução de um estudo sobre eficácia e reúso de estratégias de detecção, levando em consideração características de um domínio. Uma estratégia de detecção será útil somente se puder ser reutilizada em vários projetos. Considerando um ambiente de desenvolvimento homogêneo, em cada um dos projetos em que a estratégia for aplicada, deve haver uma baixa ocorrência de elementos detectados

que não representam necessariamente problemas no código. Para isso, investigamos:

- Se é possível definir estratégias de detecção com alto grau de reúso para sistemas da mesma família de aplicações, onde é possível que estas foram desenvolvidas com os mesmos frameworks, e, portanto, possuam estruturas semelhantes;
- Quais fatores podem encorajar ou desencorajar o reúso das estratégias de detecção em uma mesma família de sistemas; e
- Se é possível promover ainda mais o reúso de estratégias de detecção ao considerar conjuntos de elementos de código com estrutura e responsabilidade semelhantes e bem definidas.

Para conduzir a investigação desses três itens, definimos as seguintes questões de pesquisa:

1. É possível reusar estratégias de detecção de anomalias de forma eficaz em um conjunto de sistemas de um mesmo domínio? [QP1]
2. É possível diminuir a ocorrência de detecções erradas (falsos positivos) das estratégias de detecção de anomalias, ao considerar conjuntos de elementos de código que possuam características semelhantes e responsabilidades bem definidas? [QP2]

Para responder essas duas questões de pesquisa, em uma primeira etapa, busca-se calibrar ou definir estratégias de detecção para sistemas de um domínio específico, a partir de características conhecidas e observadas pelos desenvolvedores de um subconjunto de sistemas neste domínio. Esta fase tem o objetivo de calibrar estratégias existentes ou definir novas estratégias para serem utilizadas em sistemas do domínio alvo. Portanto, o conhecimento dos especialistas do domínio sobre o código fonte foi utilizado primeiro para calibrar os limiares de métricas usadas em estratégias convencionais existentes (ex. [Lanza e Marinescu, 2006] [Marinescu, 2004] e [Macía et al., 2012]). Tal conhecimento do especialista sobre o código foi também usado para definir novas estratégias com métricas não exploradas em tais estratégias convencionais. Os especialistas do domínio são mestres em informática e possuem amplo conhecimento de engenharia de software, o que é fundamental para identificar os atributos de código que podem indicar a ocorrência de problemas. Em uma segunda etapa, avalia-se o reúso e a acurácia das estratégias em uma família de outros sistemas

PUC-Rio - Certificação Digital Nº 1112618/CA

do mesmo domínio. Além do grau de reúso, a acurácia das estratégias é avaliada através da quantidade de elementos encontrados que não representam sintomas de problemas.

Para investigar o reúso das estratégias de detecção, foram considerados seis sistemas web de apoio à tomada de decisão. Esse conjunto de sistemas foi escolhido, pois operam em um domínio crítico através da geração de indicadores para apoiar a avaliação de riscos no mercado financeiro. Nesse domínio, o tempo de resposta e a precisão dos dados são importantes, pois a apresentação de indicadores errados pode gerar uma análise errada. Ao favorecer uma decisão errada, pode acontecer a consequente perda de valores financeiros (seção 3.2).

1.4. Estrutura da Dissertação

Os próximos capítulos estão estruturados com os seguintes objetivos:

Capítulo 2 – *Terminologia e Trabalhos Relacionados*: apresenta a terminologia usada aqui, além dos trabalhos relacionados à definição e ao uso de estratégias de detecção de anomalias.

Capítulo 3 – *Definição do Estudo Empírico*: descreve em detalhes o projeto do estudo empírico desenvolvido nessa dissertação.

Capítulo 4 – *Resultados e Discussões*: apresenta os resultados do estudo empírico proposto e debate sobre: o reúso das estratégias de detecção de anomalias em uma família de sistemas do mesmo domínio, os fatores que podem encorajar ou desencorajar o reúso das estratégias de detecção de anomalias em sistemas do mesmo domínio e o reúso de estratégias de detecção de anomalias em conjuntos de elementos com responsabilidades bem definidas.

Capítulo 5 – *Conclusão*: apresenta as conclusões e contribuições dessa dissertação, além de descrever trabalhos futuros.

PUC-Rio - Certificação Digital Nº 1112618/CA

2
Terminologia e Trabalhos Relacionados

Estratégias de detecção permitem a identificação de anomalias de código em um programa (Marinescu, 2004). A definição de limiares apropriados para as métricas de uma estratégia de detecção é determinante para encontrar um conjunto de elementos que estejam relacionados com as anomalias que se deseja evitar em um sistema. De fato, a eficiência de uma estratégia está diretamente relacionada à baixa ocorrência de erros na lista de anomalias detectadas em um programa. A estratégia de detecção também deveria ser reutilizável em um conjunto de projetos de software de forma a justificar o esforço associado com sua definição.

Portanto, investigamos nessa dissertação se é possível definir estratégias de detecção com alto grau de reúso em uma família de sistemas do mesmo domínio. A definição desse estudo empírico é descrita em detalhes no Capítulo 3. Antes, porém, apresentamos a terminologia usada para o desenvolvimento do nosso estudo empírico, além de trabalhos relacionados à definição e uso de estratégias de detecção de anomalias.

Primeiramente, é feita uma breve exposição sobre o que são anomalias de código (seção 2.1). Na seção 2.2, mostramos o que é e como é formada uma estratégia de detecção de anomalias, a partir da proposta apresentada por Lanza e Marinescu (2006). Neste trabalho, as estratégias de detecção não são aplicadas somente aos programas como um todo; elas são aplicadas também aos módulos que implementam interesses recorrentes de um domínio particular do programa. Portanto, na seção 2.3, apresentamos brevemente o que é mapeamento de interesses na implementação de um sistema. Para finalizar esse capítulo, descrevemos alguns trabalhos relacionados da literatura (seção 2.4).

2.1.
Anomalias de Código

Segundo (Fowler, 1999), uma anomalia representa um sintoma de problema em um trecho do código-fonte de um programa. Em particular, é um problema

PUC-Rio - Certificação Digital Nº 1112618/CA

estrutural que dificulta a manutenção no código-fonte de uma aplicação. Esse sintoma, ou anomalia, não está diretamente relacionado a uma falha (*bug*) no sistema. Porém, devido à complexidade de código associado com uma anomalia, esta pode facilitar a introdução de falhas por desenvolvedores em versões futuras do sistema (Fowler, 1999).

Não existem critérios precisos e únicos para indicar onde está uma anomalia no programa. Em geral, à medida que o programador desenvolve sua capacidade de identificar problemas no código, ele pode desenvolver também seus próprios critérios para avaliar se um trecho de código pode representar, ou não, um sintoma de declínio de qualidade.

Por exemplo, em um conjunto de projetos de uma organização, seus desenvolvedores podem começar a notar que classes tendem a apresentar dificuldades de manutenção quando estas excedem um valor limite no seu número de métodos. Desta forma, é possível determinar quantos métodos, em geral, é aconselhável manter em uma classe, sem comprometer o entendimento e a modificação dessa própria classe. Este valor limite representa um alarme lançado para os desenvolvedores e revisores de qualidade do código. Quando um desenvolvedor encontra uma classe que possui um número de métodos que excede a quantidade determinada, existe uma probabilidade de esta conter problemas no entendimento e outros atributos de qualidade do código fonte.

Da mesma forma, desenvolvedores podem determinar a quantidade máxima de linhas de código que um método usualmente deve conter. Logo, cada método que excede esse número de linhas é um candidato à fonte de problemas de manutenção ou outros atributos de qualidade.

Embora não existam critérios universais para indicar anomalias em todos os projetos de software existentes, certos tipos de anomalias são recorrentes e conhecidos. Para ajudar na caracterização destes tipos de anomalias, Fowler (1999) propôs 22 metáforas de problemas associados com cada tipo de anomalia. Devido ao declínio de qualidade associado com estes tipos de anomalia, Fowler denominou-os de *bad smells*. O uso dessas metáforas facilita a comunicação e identificação destes sintomas recorrentes de problemas que comprometem a qualidade do código fonte de sistemas de software.

Por exemplo, através da anomalia chamada *God Class*, podemos identificar uma classe que acumula muitas responsabilidades e concentra muito

PUC-Rio - Certificação Digital Nº 1112618/CA

conhecimento sobre o sistema (Fowler, 1999). Assim, a existência de uma *God Class* pode representar um impacto negativo na manutenção e evolução do sistema. Da mesma forma que uma *God Class*, podemos encontrar métodos que também reúnem muito conhecimento. Esses métodos são caracterizados no catálogo de Fowler como instâncias de anomalias do tipo *Long Method*. Um *Long Method* é caracterizado quando um método possui muitas linhas de código e, em geral, um código muito complexo. Igualmente a uma *God Class*, um *Long Method* também tende a dificultar a manutenção do sistema (Fowler, 1999).

Além desses dois tipos de anomalias associados com um módulo particular, existem também anomalias atreladas à relação entre elementos. Por exemplo, através da anomalia *Shotgun Surgery* é possível identificar um método que, a partir de uma mudança sofrida, resulta em uma série de pequenas mudanças em elementos de outras classes. Dentre outros problemas, a partir de um método que sofre de *Shotgun Surgery*, é possível que durante as várias mudanças recorrentes nos outros elementos, algum elemento seja esquecido durante a série de mudanças. A partir desse esquecimento, podemos facilitar a introdução de falhas nos sistemas.

Portanto, os tipos de anomalias catalogados por Fowler (1999) apresentam uma grande variedade de complexidade. A partir dessas metáforas é possível identificar sintomas de problemas em diferentes níveis de um programa: no relacionamento entre múltiplos métodos ou classes, no nível de cada classe ou método, ou ainda problemas mais sutis, como métodos com muitos parâmetros (Emden e Moonen, 2002).

Dessa forma, identificar anomalias periodicamente em um projeto de software é importante por vários motivos. À medida que os programadores alteram o código – para atingir metas em prazos curtos ou realizar mudanças com foco estrito na incorporação da nova funcionalidade – a qualidade do código tende naturalmente a diminuir (Lehman, 1980). Através da busca por anomalias, é possível encontrar trechos de código que não estão no lugar correto e irão, consequentemente, dificultar atividades futuras de manutenção e evolução (Emden e Moonen, 2002). Se anomalias não são encontradas e removidas, o impacto das mudanças não planejadas na estrutura do código pode representar um efeito cumulativo. Quanto mais difícil fica entender a estrutura de um código,

PUC-Rio - Certificação Digital Nº 1112618/CA

mais difícil é manter e evoluir esse código, podendo resultar em estágios de degradação que inviabilizam a continuação do projeto.

As subseções seguintes apresentam exemplos de tipos de anomalias que serão abordados no estudo empírico desta dissertação.

2.1.1. God Class

De maneira ampla, uma classe candidata à anomalia *God Class* é identificada como uma classe que reúne muito conhecimento sobre o sistema. Dessa forma, uma *God Class* pode concentrar muito conhecimento em uma única classe, o que pode representar um impacto negativo na evolução do sistema como um todo. De qualquer forma, algumas classes que possuem uma estrutura semelhante à *God Class* podem pertencer a uma parte estável do sistema, sem causar maiores problemas (Riel, 1996), (Rapu et al., 2004), (Marinescu, 2005), (Lanza e Marinescu, 2006). Essa anomalia é comparável à *Large Class*, proposta por (Fowler, 1999).

Na Figura 1 é possível ver uma parte de um diagrama de classes. Nesse diagrama, mesmo que não seja possível identificar os atributos das classes, é possível ver que existe uma classe que é bem maior que as demais. Assim, ao avaliar as métricas dessa classe, possivelmente ela será uma classe candidata à *God Class*. De qualquer forma, pela diferença entre o tamanho das classes desse diagrama, é provável que o entendimento e a consequente manutenção dessa classe seja mais difícil que as demais.

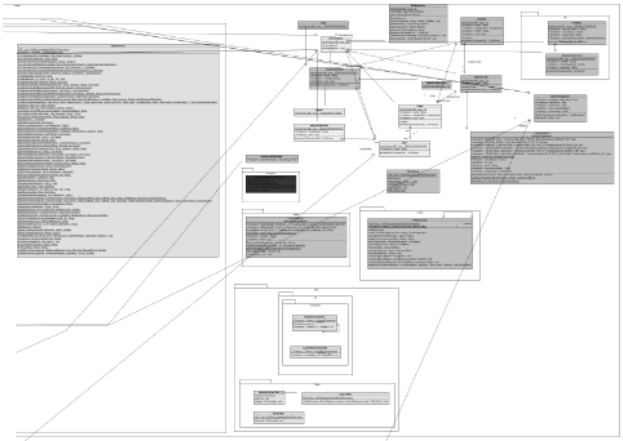


Figura 1 - Diagrama de classes com candidato à anomalia God Class.

PUC-Rio - Certificação Digital Nº 1112618/CA

2.1.2.
Long Method

Da mesma forma que a *God Class* está relacionada às classes que concentram muito conhecimento, existem métodos que acabam concentrando muita lógica de um programa. Também são frequentes cenários onde métodos são criados originalmente com uma única responsabilidade bem definida. Porém, com o passar do tempo, mais conhecimento é acrescentado ao método, até que fica difícil entender e manter esse código. Os métodos com essas características são identificados pela anomalia *Long Method* (Lanza e Marinescu, 2006).

2.1.3.
Shotgun Surgery

A anomalia *Shotgun Surgery* identifica um método que, constantemente, a cada mudança sofrida, implica pequenas mudanças em vários outros métodos. Dessa forma, a partir de um método que sofre de *Shotgun Surgery*, é possível que durante as várias mudanças recorrentes nos outros elementos, algum elemento seja esquecido durante a série de mudanças. A partir desse esquecimento, podemos facilitar a introdução de falhas nos sistemas (Fowler, 1999) (Lanza e Marinescu, 2006).

Segundo (Lanza e Marinescu, 2006), um método infectado por *Shotgun Surgery* possui muitas entidades dependentes dele (Figura 2).

PUC-Rio - Certificação Digital Nº 1112618/CA

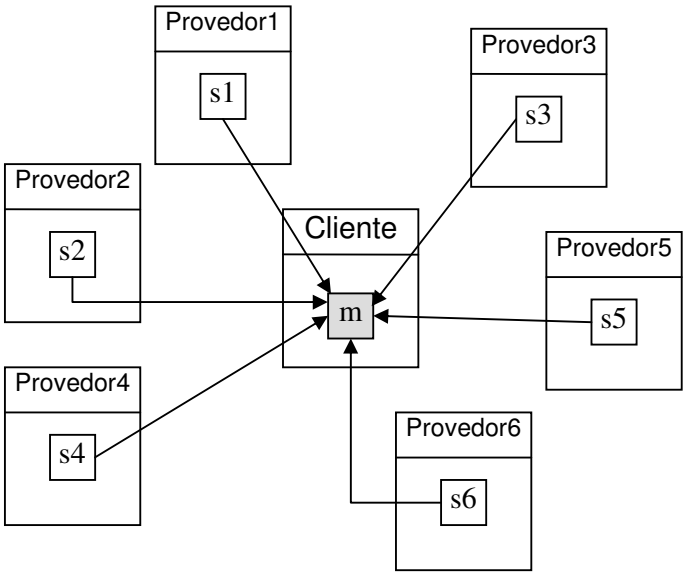


Figura 2 - Ilustração de um método infectado por Shotgun Surgery – Adaptada de (Lanza e Marinescu, 2006).

PUC-Rio - Certificação Digital Nº 1112618/CA

2.2. Estratégias de Detecção

Segundo Marinescu (2005), uma estratégia de detecção é um mecanismo usado para analisar os elementos de código através de métricas. Uma estratégia é composta pela combinação de métricas e limiares de forma a possibilitar a identificação de elementos de código que refletem as anomalias propostas por (Fowler, 1999). A partir de uma estratégia de detecção, um conjunto de elementos de código é selecionado e relacionado a um tipo de anomalia específica. Esse conjunto de elementos será examinado para que melhorias sejam feitas no código, evitando o declínio da qualidade do sistema (Marinescu, 2004), (Marinescu, 2005) e (Lanza e Marinescu, 2006).

2.2.1. Processo de Definição de Estratégias

A partir de metáforas propostas por (Fowler, 1999), Marinescu (2001) propôs uma estratégia de detecção de problemas de código, baseada em métricas, para sistemas orientados a objetos. Mais tarde, para facilitar a definição de estratégias de detecção de anomalias, Lanza e Marinescu (2006) descreveram o

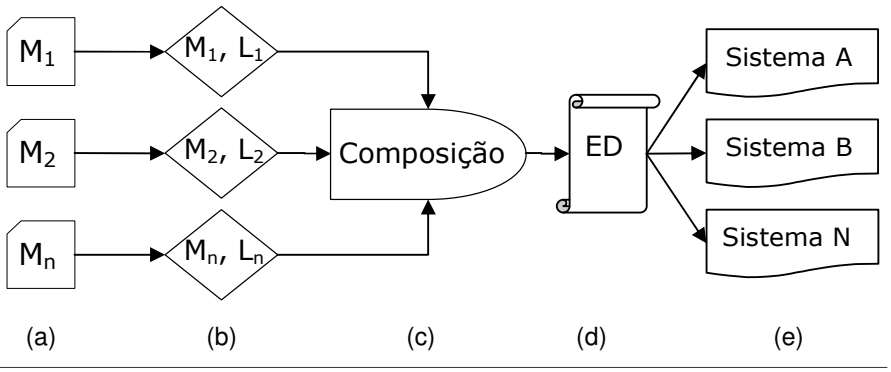
processo de formação de uma estratégia. A Figura 3 apresenta visualmente os passos principais deste processo. O processo é formado basicamente por quatro etapas, até a geração de uma estratégia de detecção de anomalias.

Para formar uma nova estratégia de detecção, o primeiro passo é identificar as características estruturais que representam sintomas de um tipo de anomalia no código-fonte. Deve-se, então, escolher métricas apropriadas para que seja possível identificar esses sintomas (Figura 3 – a). A seleção de métricas apropriadas requer o entendimento da finalidade de cada métrica utilizada. Depois, é necessário escolher limiares apropriados para as métricas escolhidas, para que seja possível filtrar um conjunto de elementos de código (Figura 3 – b). Dessa forma, é muito importante identificar os valores que farão a referência correta entre valores das métricas que podem ser muito baixos ou muito altos. Esses valores são a referência para saber se o elemento medido está com um sintoma de problema. A seguir, essas métricas e filtros são, então, combinados entre si, através de operadores lógicos (e.g., AND, OR) (Figura 3 – c), para, finalmente, formar uma estratégia de detecção (Figura 3 – d).

Como se pode observar, através de uma estratégia de detecção é possível codificar o conhecimento a respeito das características que devem ser evitadas no código fonte, com a finalidade de identificar uma determinada anomalia. Logo, escolher métricas e limiares apropriados é determinante para definir uma estratégia de detecção de anomalias que possa apoiar a descoberta de sintomas de problemas no código fonte dos sistemas (Lanza e Marinescu, 2006), (Fontana et al., 2011) e (Moha et al., 2009).

A partir do processo de formação de estratégias de detecção de anomalias apresentado na Figura 3, a principal intenção dessa abordagem é permitir que uma estratégia de detecção possa ser posteriormente aplicada em diversos sistemas (Figura 3 – e). Dessa forma, espera-se que as características escolhidas para identificar uma anomalia se mantenham dentre diferentes sistemas. No entanto, observa-se que, a partir do domínio em que as estratégias de detecção são aplicadas, algumas ocorrências de sintomas não estão necessariamente relacionadas a problemas de manutenção do código e indicam, na realidade, falsos positivos (Fontana et al., 2011) (Fontana et al., 2012).

PUC-Rio - Certificação Digital Nº 1112618/CA



Legenda:

M_i: Resultado da métrica i ED: Estratégia de detecção

L_i: Limiar associado à métrica i

Figura 3 - Processo de formação de uma estratégia de detecção (Lanza e Marinescu, 2006) e seu uso em diversos sistemas – Adaptado de (Olbrich et al., 2009).

PUC-Rio - Certificação Digital Nº 1112618/CA

Segundo (Lanza e Marinescu, 2006), uma *God Class* é baseada nas seguintes características:

1. A classe acessa muitos dados de outras classes mais simples, diretamente ou através de métodos de acesso;
2. A classe é grande e complexa;
3. Existe uma baixa coesão entre os métodos dessa classe.

Cada uma dessas características é medida através de uma métrica e possui um limiar associado. Essas métricas são descritas em detalhes na seção 3.5. Mesmo assim, as características que compõem a estratégia de detecção para a *God Class* podem ser vistas na Figura 4.

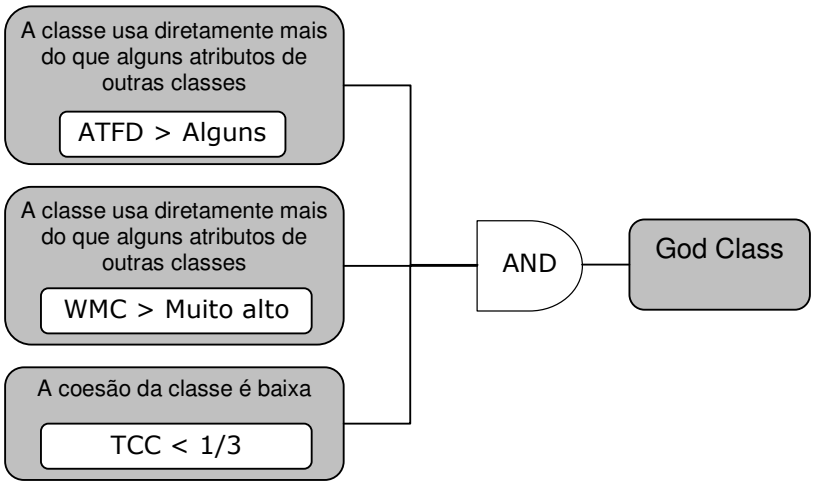


Figura 4 - Adaptação da estratégia de detecção para uma God Class segundo (Lanza e Marinescu, 2006).

PUC-Rio - Certificação Digital Nº 1112618/CA

Segundo (Lanza e Marinescu, 2006), uma *Long Method* é baseada nas seguintes características:

1. O método é muito grande;
2. O método possui muita complexidade;
3. O método possui muitos níveis de endentação;
4. O método usa muitas variáveis.

Da mesma forma, cada uma dessas características é medida através de uma métrica e possui um limiar associado. Essas métricas são descritas em detalhes na seção 3.5. Mesmo assim, as características que compõem a estratégia de detecção para a *Long Method* podem ser vistas na Figura 5.

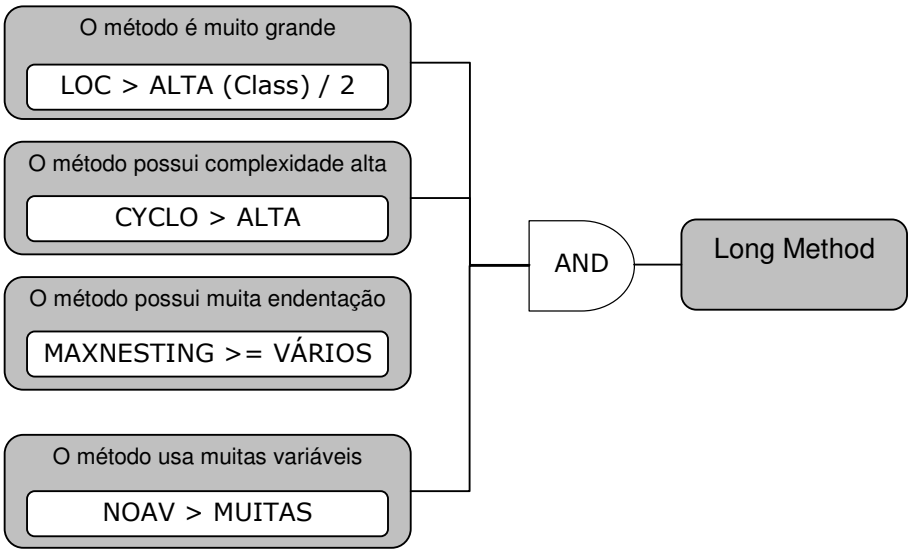


Figura 5 - Adaptação da estratégia de detecção para um Long Method segundo (Lanza e Marinescu, 2006).

PUC-Rio - Certificação Digital Nº 1112618/CA

2.2.2. Análise da Lista de Elementos Detectados

A partir da combinação das métricas e limiares, forma-se uma estratégia única de detecção de anomalias. A partir de cada estratégia de detecção de anomalias, é possível identificar candidatos de elementos no código que representam ocorrências de um tipo de anomalia específica. Ao analisar o resultado da lista de elementos identificados por uma estratégia, cada candidato pode ser definido de duas formas: o candidato pode ser acerto ou um falso positivo. Um acerto é aquele candidato que realmente representa um sintoma de problemas no código e é classificado como uma incidência da anomalia investigada. Um falso positivo é aquele candidato detectado pela estratégia que não representa um problema no código e, portanto, deveria ser desconsiderado tendo em vista o tipo de anomalia investigada. Ainda, existem elementos que podem ser classificados como falsos negativos: esses elementos existem no código e representam uma incidência do tipo de anomalia investigada, porém não foram identificados pela estratégia de detecção.

Um conjunto de elementos de código candidatos a uma anomalia é identificado quando as métricas coletadas dos atributos desses elementos excedem os limares propostos para as métricas em cada estratégia de detecção. Para formar

esse conjunto, a condição completa definida em uma estratégia de detecção deve ser satisfeita. Por exemplo, se somente operadores *AND* são utilizados, então todos os limiares devem ser violados para que um candidato seja revelado. Ainda, no caso de somente operadores *OR* serem utilizados, apenas um dos limiares precisa ser violado para que um candidato seja encontrado. As demais combinações de operadores seguem a lógica.

De qualquer forma, não existe um limiar perfeito. O que existe são limiares aceitáveis, a partir de uma justificativa coerente, de acordo com o que está sendo medido. Na prática, os limiares são úteis para definir um conjunto de elementos candidatos a uma anomalia, para o propósito de uma estratégia de detecção (Lanza e Marinescu, 2006). Além disso, qualquer mudança sutil nos limiares das métricas de uma estratégia representa um grande impacto no conjunto de elementos candidatos a anomalias (Fontana et al., 2011).

Como ilustração, na Figura 6, temos o exemplo hipotético de uma estratégia de detecção para a anomalia "Anomalia". E, na Figura 7, um diagrama de classes composto por três classes. Nesse diagrama, as classes que possuem sintomas da anomalia e que possuem, de fato, a anomalia, estão identificadas pela linha pontilhada.

Assim, supomos que a estratégia de detecção (Figura 6) seja aplicada nas classes representadas no diagrama de classes (Figura 7). Considerando que a métrica ATFD (*Access To Foreign Data* - seção 3.5.2) resulte no mesmo valor para as três classes (excedendo o limiar proposto para as três classes), o resultado da estratégia de detecção será determinado pela métrica NOM (*Number of Methods* – seção 3.5.9). Essa situação é resultado do uso do operador "AND" para essa estratégia, em que ambas as métricas devem ser satisfeitas. Nesse caso, a lista de elementos detectados para a estratégia de detecção é formada pelas classes *Class1* e *Class2*. Dessa forma, para a análise dos elementos detectados pela estratégia, temos a seguinte situação:

1. A classe *Class1* foi detectada corretamente: essa classe possui os sintomas da anomalia e possui, de fato, a anomalia investigada;
2. A classe *Class2* é um falso positivo: ela foi identificada pela estratégia de detecção, dados os sintomas identificados, porém não possui a anomalia que está sendo investigada;

PUC-Rio - Certificação Digital Nº 1112618/CA

3. A classe *Class3* é um falso negativo: apesar de possuir a anomalia "Anomalia", ela não foi identificada pela estratégia de detecção.

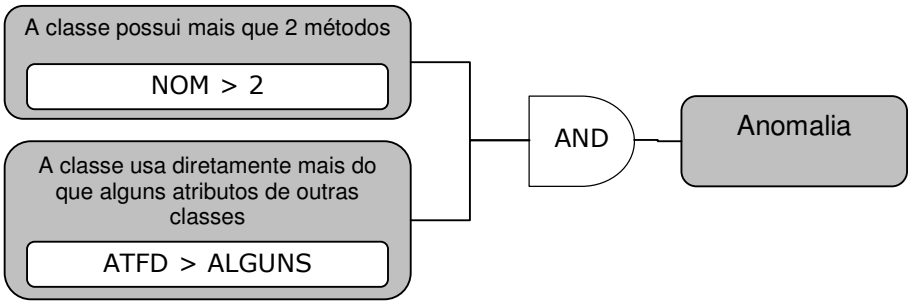


Figura 6 - Exemplo hipotético de uma estratégia de detecção.

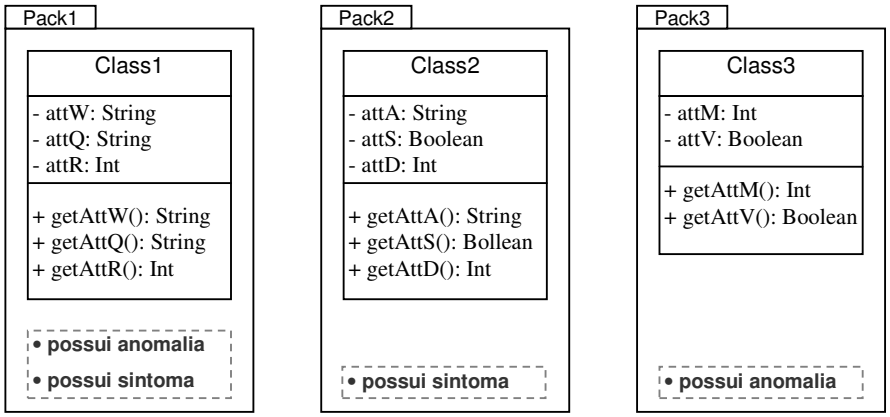


Figura 7 - Exemplo de um diagrama de classes com incidência de anomalia.

De qualquer forma, a partir da lista de candidatos, a classificação entre uma incidência de anomalia e um falso positivo pode ser subjetiva. Além disso, através de uma mudança nos limiares é possível adaptar uma estratégia de detecção para que a lista de candidatos de elementos no código que representam ocorrências de um tipo de anomalia específica tenha uma quantidade baixa de falsos positivos.

2.2.3. Reúso de Estratégias de Detecção

De acordo com o que foi exposto na seção 1.1, a eficiência de uma estratégia de detecção de anomalias está relacionada: primeiro, à baixa ocorrência

PUC-Rio - Certificação Digital Nº 1112618/CA

de elementos que não são necessariamente sintomas de problemas e, segundo, à facilidade do seu reúso. Nesse estudo, consideramos o reúso de uma estratégia de detecção como a capacidade que uma estratégia possui de ser aplicada novamente a um sistema (tal qual foi definida), de forma a gerar uma lista de candidatos com baixa ocorrência de falsos positivos. Uma estratégia de detecção pode ser reutilizada na íntegra ou através da especialização de seus limiares para um contexto particular. No próximo capítulo, ao descrever os procedimentos experimentais do estudo desta dissertação, definimos um protocolo para se avaliar diretamente diferentes níveis de reúso de uma estratégia de detecção.

2.3. Mapeamento de Interesses

Segundo (Robillard e Murphy, 2007), um interesse de uma aplicação pode ser definido como qualquer coisa que possa ser considerada como uma unidade conceitual, do ponto de vista dos interessados[4] no projeto. Dessa forma, essas unidades conceituais podem ser funcionalidades específicas do domínio, requisitos não funcionais, padrões de projetos, entre outros. Alguns exemplos de interesses que podem ser encontrados em programas são: emissão de relatório, persistência, gerenciamento de transações, segurança, fachada, entre outros.

Na Figura 8 é possível ver uma parte de um diagrama de classes. Nesse diagrama, mesmo que não seja possível identificar os atributos das classes, as classes estão identificadas por um símbolo, conforme o interesse que implementam. Dessa forma, é possível verificar que, mesmo em três pacotes distintos, as classes que estão com o símbolo "+" pertencem ao mesmo interesse. Da mesma forma, existe uma classe com o símbolo "*", representando um segundo interesse, e mais duas classes com o símbolo "#", representando um terceiro interesse.

[4] Do inglês: *stakeholders*

PUC-Rio - Certificação Digital Nº 1112618/CA

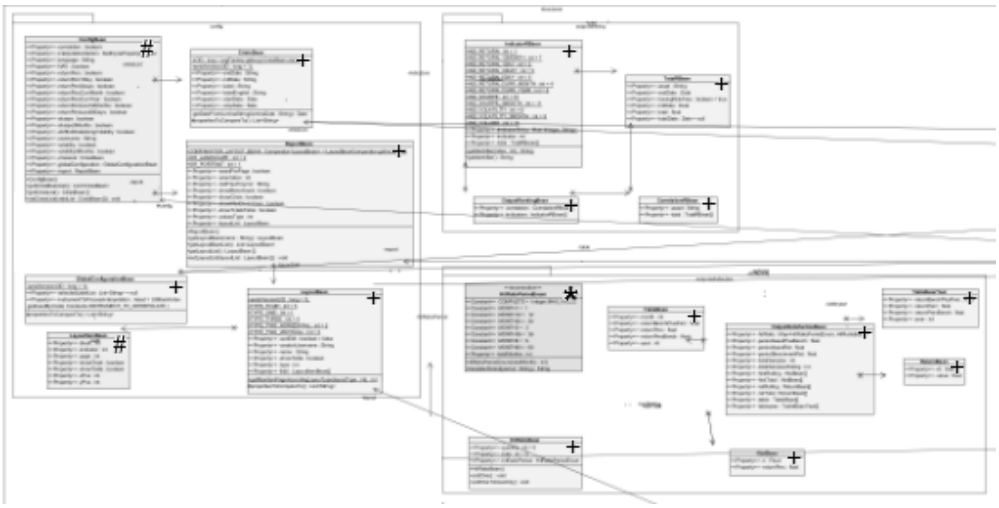


Figura 8 - Exemplo de diagrama de classes com mapeamento de três interesses.

A partir da classificação das classes segundo o interesse que implementam, podemos realizar o mapeamento lógico dos interesses presentes em um sistema orientado a objetos. Através desse mapeamento é possível agrupar os elementos de código, de forma a estabelecer uma relação entre um interesse específico e os vários elementos de código que implementam esse interesse. Em outras palavras, através do mapeamento de interesses é possível organizar os elementos de código de uma forma que não pode ser representada pela estrutura hierárquica, como a estrutura de pacotes. Além disso, vários elementos de código podem estar relacionados a um ou mais interesses. Dessa forma, é possível facilitar a rastreabilidade dos elementos, a partir do interesse que implementam, de forma a minimizar o impacto de mudanças.

Para facilitar a manutenção do mapeamento de interesses, algumas ferramentas foram propostas, como, por exemplo, *Concern Mapper* (Robillard e Weigand-Warr, 2005) e FEAT (Robillard e Murphy, 2007). Na Figura 9 é possível ver um exemplo de uso da ferramenta *Concern Mapper*. Nessa figura, podemos ser o mapeamento de três interesses para uma determinada aplicação: "Count Views", "Favourite" e "Photo_Label". Além disso, podemos ver as classes relacionadas a cada um dos interesses mapeados.

PUC-Rio - Certificação Digital Nº 1112618/CA

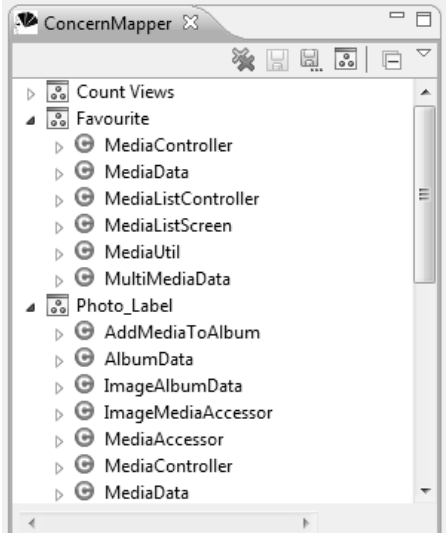


Figura 9 - Representação do mapeamento de interesses através da ferramenta ConcernMapper (Robillard e Weigand-Warr, 2005).

PUC-Rio - Certificação Digital Nº 1112618/CA

2.4.Trabalhos Relacionados

Em 2004, Marinescu propôs uma abordagem baseada em métricas para detectar anomalias, através de estratégias de detecção, apresentada na ferramenta iPlasma. Segundo o que foi proposto, uma estratégia de detecção é usada para identificar desvios dos princípios e heurísticas de bom design orientado a objetos. Além disso, o trabalho descreve que uma estratégia de detecção é formada por um conjunto de métricas, sendo cada métrica associada a um limiar específico, combinadas através de operadores lógicos. Mesmo assim, esse estudo não apresenta detalhes sobre como é feita a definição das estratégias de detecção. Ainda, o trabalho apresenta alguns poucos resultados para um conjunto pequeno de anomalias, avaliadas em poucos sistemas proprietários. Já em 2006, Lanza e Marinescu apresentaram um processo detalhado para a formação de estratégias de detecção de anomalias, além da definição de estratégias de detecção para onze anomalias de código. Nesse trabalho, mesmo que as estratégias de detecção apresentadas sejam resultado de uma ampla análise estatística, as sugestões de métricas e limiares propostas são apenas um indicativo de estratégias que podem detectar anomalias. Além disso, não há uma avaliação sobre o grau de reúso dessas estratégias de detecção quando aplicadas em outros sistemas. No trabalho também não existe um indicativo de quais características específicas de um

conjunto de sistemas pode influenciar a definição de estratégias de detecção, sejam suas métricas ou limiares.

Em 2010, Guo et al. relataram um estudo de caso em que a adaptação das estratégias de detecção foi realizada através da opinião de um grupo de especialistas do domínio. Nesse estudo, os especialistas classificaram as ocorrências de anomalias, indicadas como resultado das estratégias, entre sintomas de problemas e falsos positivos. No estudo, a avaliação dos especialistas era feita até que o grupo entrasse em consenso. Esse estudo inicia o uso da opinião de especialistas de um domínio específico para realizar a adaptação de estratégias de detecção de anomalias. Mesmo assim, os próprios autores relatam que, como o considerou apenas um sistema, os resultados ainda são muito restritos. De qualquer forma, o roteiro usado pelos autores motiva trabalhos de adaptação de estratégias de detecção, envolvendo pessoas que têm conhecimento sobre o código fonte dos sistemas e que participam ativamente do desenvolvimento de sistemas de um domínio específico.

Ainda em 2010, Schumacher et al. realizaram um estudo empírico em um ambiente comercial, o qual compararam os resultados de programadores identificando anomalias manualmente, com os resultados de uma ferramenta automática de detecção de anomalias. Segundo o estudo, o conhecimento dos programadores sobre as anomalias ainda é muito diferente do que é identificado pelas estratégias de detecção. Além disso, as estratégias de detecção precisam sofrer adaptações para que possam resultar em uma boa lista de candidatos às anomalias de código. Mesmo que esse estudo tenha sido desenvolvido em um ambiente real, foi restrito a apenas dois sistemas. De qualquer forma, o estudo motiva a adaptação das estratégias conhecidas, para que seja possível gerar uma lista de candidatos à anomalias que possam, de fato, indicar sintomas de problemas. Dessa forma, quando a adaptação das estratégias é apoiada por uma pessoa envolvida no desenvolvimento dos sistemas, é possível melhorar os resultados da estratégia de detecção.

Em 2011, Zhang et al. realizaram uma revisão sistemática da literatura, para identificar o estado da arte sobre anomalias de código. Segundo o estudo, o conhecimento sobre anomalias de código ainda é insuficiente. Ainda, os autores relatam que existem muito poucos estudos que relatam o impacto do uso de estratégias de detecção de anomalias, sendo a grande maioria dos trabalhos

PUC-Rio - Certificação Digital Nº 1112618/CA

relacionados com o desenvolvimento de novas ferramentas e métodos que detectam anomalias automaticamente. Segundo os autores, esse é o indicativo de uma grande lacuna no conhecimento atual sobre anomalias de código. Além disso, a revisão realizada nesse estudo apresenta que existem poucas evidências que justificam o uso de estratégias de detecção de anomalias.

Em 2011, Fontana et al. relataram suas experiências ao usar algumas ferramentas de detecção de anomalias. Segundo os autores, as definições das anomalias informais, o que pode favorecer resultados incertos. Dessa forma, definir limiares apropriados é fundamental para uma boa estratégia de detecção. Além disso, ainda segundo os autores, mesmo com o apoio das ferramentas, é preciso considerar um tempo razoável, em atividade manual, para adaptar as estratégias de detecção, a fim de diminuir a ocorrência de sintomas que não representam necessariamente problemas no código. A partir desse trabalho, corroboramos a ideia de que uma estratégia de detecção de anomalias eficiente está relacionada à baixa ocorrência de elementos que não representam necessariamente problemas no código e à facilidade do seu reúso. Dessa forma, motivamos a investigação de estratégias de detecção de anomalias adaptadas a partir de características do domínio em estudo, com alto grau de reúso.

Segundo esses trabalhos relacionados, existem poucos estudos empíricos relacionados à detecção de anomalias. Além disso, alguns trabalhos reforçam a importância de considerar características do domínio, para realizar adaptações nas estratégias de detecção. Mesmo assim, é importante considerar um tempo razoável, em atividade manual, para adaptar as estratégias de detecção.

Apesar de motivarem o envolvimento de características do domínio para a adaptação de estratégias de detecção, os trabalhos relacionados não avaliam o reúso das estratégias de detecção. Pelo que foi visto, a eficiência de uma estratégia de detecção de anomalias está relacionada à baixa ocorrência de elementos que não são necessariamente sintomas de problemas e à facilidade do seu reúso. Dessa forma, o estudo empírico conduzido nessa dissertação é motivado pela investigação de estratégias de detecção com alto grau de reúso e baixa ocorrência de falsos positivos, para uma mesma família de sistemas.

PUC-Rio - Certificação Digital Nº 1112618/CA

3
Definição do Estudo Empírico

O objetivo central deste trabalho é investigar se é possível aumentar o reúso de estratégias de detecção quando estas são aplicadas em projetos de software do mesmo domínio. Para que fosse possível alcançar este objetivo, esta dissertação apresenta um estudo empírico em múltiplos casos de uma organização da indústria. O estudo empírico investigou o reúso de sete estratégias de detecção, relacionadas a três anomalias. Estas estratégias foram aplicadas em seis projetos de um domínio específico.

O reúso das estratégias de detecção de anomalias foi avaliado segundo o percentual de acertos e falsos positivos das ocorrências identificadas para cada anomalia em estudo. Cada estratégia foi aplicada nos programas de forma a identificar ocorrências de anomalias. A classificação das ocorrências entre como acertos e falsos positivos é decorrente da opinião de três especialistas do domínio investigado.

A hipótese geral a ser testada durante o estudo é que é possível atingir um alto grau de reúso das estratégias de detecção, na medida em que as mesmas são definidas em função de características de sistemas do mesmo domínio. O domínio escolhido para este estudo foi de sistemas de apoio à tomada de decisão (seção 3.2). Neste domínio, foi possível identificar alguns interesses que ocorrem em vários projetos. Na implementação de cada um desses interesses, foi observado que geralmente as classes compartilham responsabilidades semelhantes e bem definidas. A partir dos interesses identificados nos projetos do domínio em estudo, podemos esperar que ocorra uma similaridade estrutural entre os sistemas. Dessa forma, a estrutura definida para cada anomalia em estudo (por exemplo, *God Class*), pode se repetir nos sistemas do domínio como um todo.

A partir da proximidade estrutural entre os sistemas do mesmo domínio, motivamos o estudo de estratégias de detecção com alto grau de reúso e baixa ocorrência de falsos positivos, para uma mesma família de sistemas. Para isso, investigamos:

PUC-Rio - Certificação Digital Nº 1112618/CA

- Se é possível definir estratégias de detecção com alto grau de reúso para sistemas da mesma família de aplicações, onde é possível que estas foram desenvolvidas com os mesmos frameworks, e, portanto, possuam estruturas semelhantes;
- Quais fatores podem encorajar ou desencorajar o reúso das estratégias de detecção em uma mesma família de sistemas; e
- Se é possível promover ainda mais o reúso de estratégias de detecção ao considerar conjuntos de elementos de código com estrutura e responsabilidade semelhantes e bem definidas.

Para conduzir a investigação desses três itens, definimos as seguintes questões de pesquisa:

1. É possível reusar estratégias de detecção de anomalias de forma eficaz em um conjunto de sistemas de um mesmo domínio? [QP1]
2. É possível diminuir a ocorrência de detecções erradas (falsos positivos) das estratégias de detecção de anomalias, ao considerar conjuntos de elementos de código que possuam características semelhantes e responsabilidades bem definidas? [QP2]

Para responder essas duas questões de pesquisa, as próximas seções deste capítulo apresentam detalhadamente a definição do estudo empírico conduzido no contexto dessa dissertação. A seção 3.1 apresenta o objetivo desse estudo empírico a partir da abordagem GQM (*goal*, *question*, *metric*) (Basili et al., 1994), utilizado por Wohlin (1999). A seguir, na seção 3.2, fazemos a exposição do contexto em que o estudo empírico foi conduzido. Depois, apresentamos o projeto desse estudo empírico de maneira detalhada (seção 3.3). Na seção 3.4, apresentamos as anomalias que foram investigadas; depois, as métricas que compõem as estratégias de detecção definidas inteiramente pelos especialistas (seção 3.5) e as estratégias de detecção escolhidas na literatura (seção 3.6), para o desenvolvimento das três fases propostas para esse estudo empírico. Ainda, apresentamos brevemente a ferramenta de detecção de anomalias usada no estudo (seção 3.7) e, finalmente, os interesses mapeados nos sistemas escolhidos para representar o domínio em estudo (seção 3.8).

PUC-Rio - Certificação Digital Nº 1112618/CA

3.1.
Objetivo do Estudo Empírico

De acordo com o formato proposto por Wohlin (1999), o objetivo do estudo empírico pode ser caracterizado da seguinte forma:

Analisar: a generalidade das estratégias de detecção de anomalias de código

Para o propósito de: reúso dessas estratégias

Com respeito à: diminuição da ocorrência de falsos positivos

Do ponto de vista: de mantenedores de software

No contexto de: sistemas web de apoio à tomada de decisão.

Para isso, o reúso das estratégias de detecção foi investigado em seis sistemas web de apoio à tomada de decisão. Esse conjunto de sistemas foi escolhido, pois operam neste domínio crítico, através da geração de indicadores para apoiar a avaliação de riscos no mercado financeiro (seção 3.2).

Em uma primeira etapa, procurou-se definir e calibrar as estratégias de detecção para sistemas desse domínio, a partir de características conhecidas e observadas por três especialistas no domínio. Assim, a primeira etapa do estudo empírico proposto aqui tem o objetivo de calibrar estratégias conhecidas na literatura ou definir novas estratégias para a investigação do seu reúso em sistemas do domínio alvo. As novas estratégias são definidas a partir de métricas não exploradas em estratégias convencionais. Para isso, o conhecimento dos especialistas do domínio sobre o código fonte foi utilizado primeiro para calibrar os limiares de métricas usadas em estratégias convencionais existentes (Lanza e Marinescu, 2006), (Marinescu, 2004) e (Macía et al., 2012). Em outras palavras, foram reutilizadas estratégias da literatura com adaptação somente dos limiares.

Depois, ainda nessa etapa, o conhecimento dos especialistas foi usado também para definir novas estratégias, a partir de métricas não exploradas nas estratégias convencionais da literatura. Para isso, as métricas e seus respectivos limiares foram combinados através de operadores lógicos (conforme a formação de estratégias de detecção, seção 2.2). Dessa forma, o conhecimento dos especialistas foi a base para determinar quais atributos deveriam ser medidos, com a finalidade de indicar um sintoma de problema relacionado a cada uma das anomalias investigadas.

PUC-Rio - Certificação Digital Nº 1112618/CA

A partir da definição das estratégias de detecção, realizamos um treinamento dessas estratégias para que fosse possível realizar ajustes nos seus limiares. Para isso, avaliamos o grau de reúso e a acurácia das estratégias de detecção para um conjunto pequeno de sistemas do domínio. Em uma segunda etapa, avaliamos o reúso e a acurácia das estratégias definidas na primeira etapa, em outro conjunto de sistemas do mesmo domínio.

Nesse estudo, o grau de reúso e a acurácia das estratégias de detecção foram avaliados através da quantidade de falsos positivos encontrados, a partir do conjunto de ocorrências identificados pelas estratégias de detecção. Consideramos como falsos positivos, todas as ocorrências indicadas pelas estratégias de detecção de anomalias, quando aplicadas a um dos sistemas do domínio em estudo, que não representam necessariamente um sintoma de problema no código.

A definição das ocorrências que são falsos positivos é feita através da avaliação qualitativa dos especialistas do domínio, durante as sessões de investigação realizadas nesse estudo (seção 3.3.1). Os falsos negativos não são alvo do nosso estudo, uma vez que estamos avaliando sistemas muito grandes com centenas de classes. É impeditiva a análise por desenvolvedores de cada ocorrência dos três tipos de anomalias em cada classe dos sistemas envolvidos. De fato, análise de falsos negativos não é geralmente alvo de estudos desta natureza (Marinescu, 2001) (Marinescu, 2004).

PUC-Rio - Certificação Digital Nº 1112618/CA

3.2. Contexto de Aplicação

O estudo empírico proposto para essa dissertação foi conduzido em uma empresa de consultoria e desenvolvimento em sistemas de missão crítica. A empresa é dirigida por doutores e mestres em informática, e foi fundada em 2000. Em 2010, a empresa absorveu um conjunto de sistemas web de apoio à tomada de decisão, originalmente desenvolvido por outra empresa. Conforme já mencionado, esse conjunto de sistemas opera em um domínio crítico, pois realiza a análise de indicadores para o mercado financeiro. Nesse domínio, o tempo de resposta e a precisão dos dados são importantes, pois a apresentação de indicadores errados pode gerar uma análise errada. Ao favorecer uma decisão errada, pode acontecer a consequente perda de valores financeiros. Logo, como forma de promover a

confiabilidade destes sistemas em longo prazo, é importante que os sistemas permaneçam manuteníveis. Caso contrário, as dificuldades de manutenção facilitarão a introdução de faltas nos programas ao longo do histórico do projeto. Além disso, a baixa manutenibilidade dificulta que a empresa se adapte a mudanças nas regras de negócio ou incorpore inovações, perdendo, assim, competitividade no mercado. A seguir, apresentamos várias características destes programas, algumas delas sinalizando a importância de manter a manutenibilidade dos mesmos através, por exemplo, de detecção de anomalias de código.

Segundo a Tabela 1, cada equipe é responsável por dois sistemas avaliados nesse estudo e é representada por um líder. Assim, os seis sistemas escolhidos para o estudo (A, B, C, D, E e F) estão divididos entre três equipes distintas. No total, as equipes são formadas por oito programadores distintos (P1, P2, P3, P4, P5, P6, P7 e P8). Além disso, o líder de cada uma dessas três equipes participa do estudo como especialista do domínio (E1, E2 e E3). Os três especialistas do domínio que participam desse estudo possuem experiência de mais de dois anos sobre o desenvolvimento de sistemas para o domínio em estudo. Além disso, todos os especialistas são mestres em informática e, durante a pós-graduação, aprofundaram seus conhecimentos sobre engenharia de software. Dessa forma, devemos esclarecer que os especialistas do domínio já possuíam conhecimento prévio sobre: (i) boas práticas de desenvolvimento de software, (ii) métricas de software relacionadas à orientação a objetos, e (iii) características que devem ser observadas em sistemas orientados a objetos, como alta coesão e baixo acoplamento, entre outras características. Ainda, os especialistas já tinham conhecimento de anomalias mais simples, como *God Class* e *Long Method*.

Como vários fatores de processo e produtos podem influenciar a introdução de anomalias de código, estudamos o reúso de estratégias no contexto de uma organização. Mesmo que o conjunto de sistemas seja legado de outra empresa, os sistemas vêm sofrendo refatorações constantes, o que colabora para que o código seja modificado para uma estrutura semelhante. Além disso, apesar das equipes dos projetos serem distintas, sempre procuramos considerar um ambiente homogêneo de desenvolvimento e manutenção de sistemas. Assim, o contexto de aplicação desse estudo é formado por desenvolvedores que pertencem à mesma organização e seguem práticas, ferramentas, biblioteca e *frameworks* semelhantes.

PUC-Rio - Certificação Digital Nº 1112618/CA

Tabela 1 - Composição das equipes que mantêm os sistemas usados no estudo

Sistemas	Especialistas	Programadores
A e B	E1	P1, P2 e P3
C e D	E2	P4 e P5
E e F	E3	P6, P7 e P8

Os sistemas que fazem parte desse estudo possuem uma estrutura direcionada à operação de grande quantidade de dados. A partir desses dados é possível gerar indicadores para a tomada de decisão no mercado financeiro. Os dados estão relacionados, por exemplo, com informações históricas de ativos financeiros e informações relacionadas à configuração e armazenamento de estruturas utilizadas pelos usuários. A partir dessas estruturas, é possível controlar: carteiras de ativos financeiros, tipos de relatório, variáveis utilizadas nos cálculos dos indicadores, modos de interpolação de dados, entre outras informações.

Nesses sistemas, a interface com o usuário é importante. Deve exibir os indicadores de modo a facilitar o entendimento das informações e a consequente tomada de decisões. Assim, existem muitas classes que compõem os elementos visuais. Esses elementos também recebem as requisições do usuário e dão início à geração de informações e processamento de dados. Ao final das operações necessárias, os dados são mostrados na interface e o usuário pode analisá-los através de gráficos e relatórios em diferentes formatos.

No domínio em estudo, o tempo de resposta das solicitações também é fundamental para apoiar a tomada de decisões. Desse modo, algumas operações realizadas por esses sistemas utilizam tecnologias assíncronas com processamento nos clientes (*client-side*); isto é, operações são executadas diretamente no navegador do cliente como, por exemplo, *javascript* e *JQuery*. Além disso, a manutenibilidade dessas classes também é importante para não acarretar potenciais efeitos colaterais ao desempenho.

Nesses sistemas, também existe um conjunto de classes que garante o controle de acesso às informações por meio de autenticação. A autenticação é necessária, pois existem restrições para os diferentes perfis de usuários. Além disso, um grande conjunto de classes é usado para refletir o modelo do banco de dados. Da mesma forma que em sistemas de outros domínios, essas classes são necessárias para garantir a integridade das informações. Ainda, nesses sistemas, é

PUC-Rio - Certificação Digital Nº 1112618/CA

importante garantir a frequente comunicação com serviços de terceiros. Esses serviços fornecem dados provenientes de fontes de dados financeiros como, por exemplo, Bloomberg (www.bloomberg.com).

Outro ponto importante para a escolha destes sistemas é a recorrência de conjuntos de elementos com responsabilidades bem definidas (seção 3.8). Dessa forma, é possível garantir a proximidade estrutural dos conjuntos de classes dos sistemas em estudo, o que é fundamental para responder nossas duas questões de pesquisa (Capítulo 3). Além disso, através da recorrência de interesses nos sistemas é possível avaliar o percentual de falsos positivos das estratégias considerando as características específicas dos conjuntos de elementos com estruturas e responsabilidades bem definidas (especificamente nossa segunda questão de pesquisa – QP2, Capítulo 3).

Um exemplo de interesse identificado no domínio é chamado de *Ação*. Nesse interesse, um conjunto de elementos de código do domínio é formado por classes que recebem as requisições do usuário e iniciam a geração de indicadores financeiros. Essas classes têm as seguintes responsabilidades: recebem os parâmetros necessários, calculam uma grande quantidade de informações e geram os resultados para serem exibidos na interface do sistema. Mesmo que essas classes tenham um papel intermediário entre a interface do sistema e as classes de negócio, é preciso evitar que essas classes fiquem muito grandes. Segundo (Riel, 1996), em um bom projeto orientado a objetos a inteligência do sistema deve ser distribuída uniformemente entre as classes de alto nível. Além disso, também é importante evitar que o acoplamento dessas classes fique muito disperso na aplicação. Um comportamento como esse pode levar a uma situação indesejável, porque uma mudança na classe que possui acoplamento disperso pode gerar uma série de mudanças em todas as outras classes acopladas, o que pode facilitar a introdução de falhas.

Outro exemplo de conjunto de elementos de código do domínio em estudo (entre outros domínios) é formado por classes responsáveis pela persistência dos dados. Essas classes são formadas geralmente por muitos métodos de atribuição e leitura de valores de atributos (*getters* e *setters*). Dessa forma, nesse e em outros domínios, as classes de persistência frequentemente possuem métodos bem simples. Não devem possuir métodos muito longos; caso contrário, isso é provavelmente um indicador que estejam incorporando algum processamento

PUC-Rio - Certificação Digital Nº 1112618/CA

adicional dos dados, incluindo parte da lógica da aplicação de forma indesejável. Assim, uma classe da camada de persistência com essas características pode indicar também a existência de acoplamentos nocivos à manutenção do programa.

Desde a absorção desse conjunto de sistemas, em 2010, já foram registrados mais de 160 *bugs*, relacionados ao código herdado. Mesmo assim, os sistemas vêm sofrendo fatorações constantes para que a estrutura do código seja semelhante.

3.3. Projeto do Estudo Empírico

Segundo (Marinescu, 2004), um bom índice de acurácia de uma estratégia de detecção deve estar acima de 60%. De qualquer forma, o índice usado nesse estudo foi um pouco mais rigoroso e está um pouco acima deste índice sugerido na literatura: 66%, isto é, dois terços de acertos nas detecções feitas por cada estratégia. A escolha do índice de acurácia de 66% também se deu pelo fato de que, dessa forma, é possível garantir que a cada três ocorrências identificadas pelas estratégias de detecção, apenas uma é classificada como um falso positivo. Imaginamos que, se um desenvolvedor está procurando sintomas de problemas no código através de uma estratégia de detecção e encontra um número de falsos positivos maior que dois terços, muito provavelmente ele vai ficar desmotivado a reusar a mesma estratégia em outro programa. Além disso, se for necessário definir uma estratégia de detecção para cada sistema de um conjunto de sistemas, essa tarefa vai custar muito tempo e fatalmente será negligenciada. Dessa forma, para avaliar se as estratégias de detecção de anomalias escolhidas possuem um alto grau de reúso, definimos esse critério: as estratégias de detecção de anomalias, quando aplicadas aos sistemas em estudo, devem resultar em, no máximo, 33% de ocorrências de falsos positivos. Sendo assim, para avaliar o reúso de estratégias de detecção de anomalias no domínio escolhido, foi definido um estudo empírico formado por três fases.

O objetivo da primeira fase, chamada de fase de ajustes dos limiares, é definir estratégias de detecção de anomalias que tenham percentual de falsos positivos abaixo de 33%, para duas aplicações do domínio em estudo. A segunda fase, chamada de fase de reúso das estratégias de detecção, tem por objetivo

PUC-Rio - Certificação Digital Nº 1112618/CA

avaliar se as estratégias definidas na fase de ajustes possuem alto grau de reúso em outros quatro sistemas do mesmo domínio. Da mesma forma que na primeira fase, espera-se que na segunda fase o resultado de falsos positivos das estratégias se mantenha abaixo de 33%. Finalmente, a última etapa é chamada de fase de estratégias orientadas a interesses. Essa fase tem como objetivo verificar se o percentual de falsos positivos das estratégias pode ser melhorado, tendo em vista a aplicação das estratégias de detecção apenas em classes de um mesmo interesse, sendo esse interesse recorrente nos programas do domínio em estudo.

3.3.1. Protocolo para a avaliação as ocorrências identificadas pela ferramenta de detecção

Nesse estudo, o percentual de falsos positivos é definido através da avaliação qualitativa dos especialistas do domínio. Essa avaliação foi realizada durante cada uma das seis sessões de investigação ocorridas nesse estudo. Uma sessão de investigação é caracterizada pela aplicação das estratégias de detecção de anomalias a cada um dos sistemas escolhidos, para a avaliação qualitativa das ocorrências identificadas, por um dos especialistas do domínio. Para isso, foi definido o seguinte protocolo: (i) Antes de cada sessão com o especialista, as estratégias de detecção de anomalias foram aplicadas a um sistema e o resultado das ocorrências identificadas pelas estratégias de detecção foi coletado e colocado em planilhas. Dessa forma ficou mais fácil registrar a classificação do especialista para cada ocorrência; (ii) Depois, no início de cada sessão de investigação, era apresentada uma breve descrição do estudo empírico e seus objetivos. Depois, alguns conceitos técnicos relacionados ao estudo eram retomados, como: o que são anomalias de código, o que são métricas e o que são estratégias de detecção. Além disso, também foi observada a necessidade de que o especialista pudesse colaborar com o estudo de forma transparente, sem se deixar influenciar pelos problemas que poderiam aparecer no código fonte. A seguir, eram mostradas as definições das anomalias investigadas nesse estudo, as estratégias de detecção de anomalias definidas para identificar essas anomalias, além das métricas usadas para formar as estratégias de detecção. Seguinte, era comentado que haveria o uso de uma planilha pra registrar a classificação das ocorrências. (iii) A ferramenta era executada novamente, para gerar a lista de ocorrências identificadas por cada

PUC-Rio - Certificação Digital Nº 1112618/CA

estratégia de detecção e, a cada ocorrência identificada pelas estratégias de detecção, o especialista fazia a sua classificação qualitativa, para indicar se a ocorrência era um falso positivo ou se realmente representava um sintoma de problema para o domínio das aplicações em estudo. Dessa forma, a partir da avaliação qualitativa dos especialistas do domínio, foi possível definir o percentual de falsos positivos de cada estratégia de detecção de anomalias. O percentual de falsos positivos é o resultado do número de ocorrências classificadas como falso positivo, em relação ao número de ocorrências identificadas por cada estratégia de detecção de anomalias.

3.3.2. Fase de Ajustes dos Limiares

Como mencionado, o objetivo da fase de ajustes dos limiares, é definir estratégias de detecção de anomalias que tenham percentual de falsos positivos abaixo de 33%, para duas aplicações do domínio em estudo. No início da fase de ajustes dos limiares, o apoio dos especialistas do domínio foi fundamental, para que fosse possível definir: (i) as características do domínio dos sistemas envolvidos nesse estudo empírico, além dos sistemas escolhidos para representar o domínio em estudo (seção 3.2), (ii) as anomalias que são relevantes investigar, para o domínio em estudo (seção 3.4), e (iii) as estratégias de detecção de anomalias usadas para investigar as anomalias definidas anteriormente (seção 3.6).

Para isso, as definições de anomalias que são recorrentes na literatura (Zhang, Hall e Baddoo, 2011) foram apresentadas aos especialistas do domínio. Isso foi feito para que os especialistas pudessem avaliar as anomalias que seriam interessantes investigar no domínio alvo, do ponto de vista de quem participa do dia-a-dia do desenvolvimento de sistemas do domínio em estudo. A partir da escolha das anomalias, foram definidas as estratégias de detecção de anomalias que seriam utilizadas. Conforme mencionado anteriormente, foram utilizadas estratégias definidas inteiramente a partir da sugestão dos especialistas do domínio, além de estratégias conhecidas da literatura (Lanza e Marinescu, 2006), (Marinescu, 2004) e (Macía et al., 2012). No caso das estratégias escolhidas a partir da literatura, os especialistas sugeriram refinamentos nos limiares das

PUC-Rio - Certificação Digital Nº 1112618/CA

métricas, de acordo com suas experiências e observações sobre as características do código-fonte dos sistemas escolhidos para representar o domínio em estudo.

Ainda na fase de ajustes dos limiares, foi escolhida uma ferramenta de detecção de anomalias de código. A escolha dessa ferramenta foi determinada para que fosse possível avaliar as ocorrências identificadas pelas estratégias de detecção de anomalias, considerando o mapeamento de interesses das classes de cada sistema escolhido para o estudo (seção 3.7).

Para avaliar o percentual de falsos positivos de cada estratégia de detecção escolhida, na fase de ajustes dos limiares, foram realizadas duas sessões de investigação. Cada sessão de investigação teve a participação de um especialista do domínio, considerando os dois sistemas escolhidos para essa fase. A partir da avaliação qualitativa do especialista do domínio, para cada ocorrência identificada pelas estratégias de detecção de anomalias definidas para essa fase, foi possível definir o percentual de falsos positivos para cada uma das estratégias de detecção escolhidas. Ao final da fase de ajustes, verificamos as estratégias que resultaram em no máximo 33% de falsos positivos. Dessa forma, as estratégias que não excederam esse critério foram aplicadas na fase seguinte, chamada fase de avaliação do reúso das estratégias de detecção.

3.3.3. Fase de Avaliação do Reúso das Estratégias de Detecção

Como mencionado, o objetivo da fase de avaliação do reúso das estratégias de detecção é avaliar se as estratégias definidas na fase de ajustes dos limiares podem ser reusadas em outros quatro sistemas do mesmo domínio, de forma que cada estratégia não resulte em mais do que 33% de falsos positivos, considerando a média entre os sistemas avaliados. Nessa etapa, o grau de reúso de cada estratégia de detecção de anomalias aplicada nos sistemas em estudo é definido através da seguinte classificação:

- Reúso total: a estratégia foi aplicada nos sistemas em estudo e resultou diretamente em no máximo 33% de falsos positivos, em todos os sistemas;
- Reúso parcial: a estratégia foi aplicada nos sistemas em estudo, porém o percentual de falsos positivos excedeu 33% em um ou dois sistemas; isto

PUC-Rio - Certificação Digital Nº 1112618/CA

é, as estratégias foram reusadas de forma eficaz em, pelo menos, a metade dos sistemas;

- Nenhum reúso: nesse caso, a estratégia foi aplicada nos sistemas do domínio e o percentual de falsos positivos excedeu 33% em mais de dois sistemas; isto é, as estratégias foram reusadas de forma eficaz em menos da metade dos sistemas.

Da mesma forma que na fase anterior, na fase de reúso, o percentual de falsos positivos para todas as estratégias de detecção é determinado pela avaliação qualitativa do especialista do domínio. Assim, o percentual de falsos positivos para cada estratégia de detecção de anomalias é definido pelo número de ocorrências classificadas como falso positivo pelo especialista do domínio, em relação ao número total de ocorrências identificadas pela ferramenta de detecção.

Na fase de reúso, foram realizadas quatro sessões de investigação, sendo uma para cada um dos quatro sistemas escolhidos para essa etapa. A partir dos resultados das quatro sessões de investigação, procurou-se identificar quais estratégias tiveram um reúso total. Dessa forma, essa etapa procura gerar indícios das estratégias que tiveram bons resultados, considerando o domínio das aplicações em estudo. Depois, como segunda atividade dessa fase, investigamos os casos em que o percentual de falsos positivos esteve acima de 33%. Nesses casos, procuramos entender quais fatores influenciaram a alta ocorrência de falsos positivos. Para isso, investigamos os valores das métricas de cada conjunto de elementos identificados pelas estratégias que excederam 33% de falsos positivos. Assim, foi possível observar quais fatores podem encorajar ou desencorajar o reúso dessas estratégias de detecção de anomalias, para os seis sistemas que representam o domínio em estudo.

3.3.4. Fase de Estratégias Orientadas a Interesses

A última fase proposta para o estudo empírico conduzido por essa dissertação é chamada de fase de estratégias orientadas a interesses. Essa fase tem como objetivo verificar se o percentual de falsos positivos das estratégias diminui ao considerar a aplicação das estratégias de detecção apenas em elementos de cada interesse recorrente nos sistemas do mesmo domínio. Através da última fase,

PUC-Rio - Certificação Digital Nº 1112618/CA

damos início ao estudo de estratégias de detecção orientadas a conjuntos de elementos com características e responsabilidades semelhantes e bem definidas.

3.4. Anomalias Investigadas

No estudo empírico conduzido aqui, as anomalias investigadas foram definidas juntamente com os especialistas do domínio. Dessa forma, foi possível investigar anomalias que são interessantes do ponto de vista de quem acompanha o dia a dia do desenvolvimento dos sistemas que representam o domínio em estudo. Dessa forma, foram investigadas: uma anomalia em nível de classe, uma anomalia em nível de método e uma anomalia relacionada a mudanças. São elas: *God Class* (seção 2.1.1), *Long Method* (seção 2.1.2) e *Shotgun Surgery* (seção2.1.3).

3.5. Métricas que compõem as estratégias de detecção definidas para o estudo

Nesse estudo, investigamos o grau de reúso de estratégias conhecidas da literatura e estratégias formadas inteiramente pelos especialistas, a partir do processo de formação de estratégias de detecção (seção 2.2). Antes de apresentar as estratégias que foram utilizadas, apresentamos as métricas que compõem essas estratégias de detecção. As métricas descritas aqui foram coletadas através das ferramentas SCOOP, *Together* e *Understand* (justificamos a escolha dessas ferramentas na seção 3.7).

3.5.1. Accessed Methods

Accessed Methods (AM) é uma métrica de acoplamento e está relacionada à quantidade de métodos externos utilizados por um método (Lanza e Marinescu, 2006). Através dessa métrica foi possível formar a estratégia de detecção para a anomalia *Shotgun Surgery*.

PUC-Rio - Certificação Digital Nº 1112618/CA

3.5.2. Access to Foreign Data

Access to Foreign Data (ATFD) é uma métrica de acoplamento e está relacionada ao número de atributos de classes externas, que são acessados diretamente ou através de métodos de acesso (Marinescu, 2005). Essa métrica é usada na estratégia de detecção da anomalia *God Class* (Lanza e Marinescu, 2006).

3.5.3. Coupling Between Objects

Coupling Between Objects (CBO) é uma métrica de acoplamento e está relacionada ao número de classes a que a classe avaliada está acoplada. CBO é definida dado que uma classe está acoplada à outra classe sempre que uma classe usa métodos ou variáveis definidas em outra classe (Chidamber e Kemerer, 1994). Além disso, o acoplamento excessivo de classes representa uma degradação de um projeto modular.

Essa métrica foi usada em duas estratégias de detecção formadas inteiramente pelos especialistas, para identificar a anomalia *God Class*.

3.5.4. FanOut

FanOut é uma métrica que representa o número de fluxos locais de um método, somado ao número de estruturas de dados que o método altera (Henry e Kafura, 1981). Essa métrica foi usada na estratégia *Shotgun Surgery* (Fowler, 1999).

3.5.5. Lines of Code

Lines of Code (LOC) é uma métrica que representa o número total de linhas de código de um método (Lorenz e Kidd, 1994). Essa métrica foi usada nas estratégias de detecção formadas inteiramente por especialistas para detectar as anomalias *God Class* e *Long Method*, além de ser usada na estratégia para detectar *Long Method* a partir de (Lanza e Marinescu, 2006).

PUC-Rio - Certificação Digital Nº 1112618/CA

3.5.6.
Maximum Nesting Level

Maximum Nesting Level (MaxNesting) é uma métrica que está relacionada ao maior nível de endentação das estruturas de controle presentes em um método (Lanza e Marinescu, 2006). Essa métrica também é usada na estratégia de detecção *Long Method*, a partir de (Lanza e Marinescu, 2006).

3.5.7.
McCabe's Cyclomatic Number

McCabe's Cyclomatic Number (CC) é uma métrica que representa o número de caminhos linearmente independentes através do código de um método (McCabe, 1976). Essa métrica foi usada nas estratégias definidas inteiramente por especialistas para as estratégias *Long Method* e *Shotgun Surgery*, além de ser usada na estratégia de detecção *Long Method*, a partir de (Lanza e Marinescu, 2006).

3.5.8.
Number of Accessed Variables

Number of Accessed Variables (NOAV) é uma métrica que representa o número total de variáveis acessadas diretamente pelo método avaliado (Lanza e Marinescu, 2006). Essa métrica é resultante da soma do número de variáveis locais, do número de parâmetros do método e do número de variáveis globais usadas no método avaliado. Essa métrica também é usada na estratégia de detecção *Long Method*, a partir de (Lanza e Marinescu, 2006).

3.5.9.
Number of Methods

Number of Methods (NOM) é uma métrica que representa o número de métodos de uma classe. Essa métrica foi usada na estratégia de detecção formada inteiramente pelos especialistas para a estratégia de detecção *God Class*.

PUC-Rio - Certificação Digital Nº 1112618/CA

3.5.10.
Tight Class Cohesion

Tight Class Cohesion (TCC) é uma métrica de coesão e está relacionada ao número relativo de pares de métodos de uma classe que acessam em comum pelo menos um atributo da classe avaliada (Bieman e Kang, 1995). Essa é uma métrica normalizada, o que representa que os seus valores variam entre zero e um. Quanto mais baixo o valor do TCC, mais baixa a coesão da classe. Essa métrica é usada na estratégia *God Class*, a partir de (Lanza e Marinescu, 2006).

3.5.11.
Weighted Method Count

Weighted Method Count (WMC) é uma métrica que representa a soma da complexidade ciclomática de todos os métodos de uma classe. Uma classe que resulta em alto valor para a métrica WMC indica uma classe de difícil manutenção (Chidamber e Kemerer, 1994) (McCabe, 1976). Essa métrica também é usada na estratégia de detecção da anomalia *God Class* (Lanza e Marinescu, 2006).

3.6.
Estratégias de Detecção Escolhidas

A partir das anomalias escolhidas, as estratégias de detecção avaliadas nesse estudo foram definidas em conjunto com os especialistas do domínio. Assim, com o apoio dos especialistas, foram escolhidas e calibradas manualmente estratégias de detecção conhecidas da literatura. Depois, foram formadas novas estratégias de detecção, tendo como orientação o processo de formação de estratégias de detecção proposto por (Lanza e Marinescu, 2006) (seção 2.2).

Para definir as estratégias em conjunto com os especialistas, foi necessário decidir quais métricas identificam os sintomas que devem ser evitados, tendo em vista as características do domínio em estudo. Dessa forma, para definir as estratégias que avaliam *God Class*, os especialistas sugeriram uma métrica relacionada ao tamanho e uma métrica relacionada ao acoplamento. Além disso, os especialistas sugeriram que fosse possível variar a métrica de tamanho para avaliar qual estratégia poderia apresentar melhores resultados, tendo em vista os sistemas do domínio em estudo. Depois, para identificar *Long Method*, os

PUC-Rio - Certificação Digital Nº 1112618/CA

especialistas do domínio sugeriram que fossem usadas uma métrica de tamanho e uma métrica de complexidade. Por último, para identificar *Shotgun Surgery*, os especialistas sugeriram uma métrica de complexidade e uma métrica de acoplamento.

Depois de definir estratégias de detecção em conjunto com os especialistas do domínio, foram escolhidas três estratégias de detecção, a partir da literatura. Dessa forma, cada uma das estratégias da literatura está relacionada a uma das anomalias escolhidas para o estudo. Além disso, as estratégias escolhidas a partir da literatura foram descritas no Capítulo 2 (seção 2.2.1). De qualquer forma, os limiares usados para todas as estratégias escolhidas para o estudo foram definidos segundo as opiniões dos três especialistas. A partir da absorção dos seis sistemas, as equipes têm trabalhado constantemente em refatorações do código. Durante as refatorações, os especialistas puderam melhorar o seu conhecimento sobre o código-fonte, no que se refere às estruturas de código que representam sintomas de problemas. Dessa forma, era esperado que os especialistas tivessem uma opinião semelhante para o que fosse definido como um sintoma de anomalia no código.

Nesse estudo, identificamos as estratégias de detecção definidas inteiramente pelos especialistas pelo sufixo "Esp" e identificamos as estratégias de detecção originadas a partir da literatura através do sufixo "Lit". Em especial, o sufixo "EspLoc" está relacionado à estratégia de detecção definida inteiramente pelos especialistas que possui a métrica LOC (Lines of Code) e o sufixo "EspNom" está relacionado à estratégia de detecção definida inteiramente pelos especialistas que possui a métrica NOM (Number of Methods). Assim, as estratégias de detecção escolhidas na fase de ajustes, para detecção de anomalias definidas pelos especialistas, a partir de características do domínio em estudo, são apresentadas na Tabela 2, e as estratégias de detecção definidas a partir da literatura, com os limiares definidos pelos especialistas, são apresentadas na Tabela 3.

Tabela 2 - Estratégias de detecção sugeridas inteiramente pelos especialistas

Anomalia	**Estratégia**
God Class EspLoc	(LOC > 150) e (CBO > 6)
God Class EspNom	(NOM > 15) e (CBO > 6)
Long Method Esp	(LOC > 50) e (CC > 5)
Shotgun Surgery Esp	(CC > 7) e (AM > 7)

PUC-Rio - Certificação Digital Nº 1112618/CA

Tabela 3 - Estratégias de detecção sugeridas na literatura com limiares ajustados pelos especialistas

Anomalia	Estratégia
God Class Lit	(ATFD > 5) e (WMC > 46) e (TCC < 33)
Long Method Lit	(LOC > 50) e (CC > 6) e (MaxNesting > 5) e (NOAV > 3)
Shotgun Surgery Lit	(FanOut > 16)

3.7. Ferramenta de Detecção Escolhida

À medida que um sistema cresce, identificar anomalias manualmente fica ainda mais difícil ou impeditivo. Dessa forma, várias ferramentas de detecção automática de anomalias foram propostas. Algumas delas são: *JDeodorant* (Tsantalis, Chaikalis e Chatzigeorgiou, 2008), PMD, *iPlasma*, *inFusion* e *Stench Blossom* (Murphy-Hill e Black, 2010), e grande parte das ferramentas propostas é baseada nas estratégias de detecção propostas por (Lanza e Marinescu, 2006). Mesmo assim, muitas ferramentas não fornecem acesso às regras (métricas e limiares) usadas nas estratégias de detecção (Fontana et al., 2011). Sem essa condição, não existe a possibilidade de refinamento das estratégias. Assim, a adaptação e o reúso das estratégias ficam prejudicados.

Para tornar viável a investigação da primeira questão de pesquisa (Questão de Pesquisa 1, Capítulo 3), era preciso escolher uma ferramenta de detecção de anomalias em que fosse possível definir as próprias estratégias de detecção. Além disso, para que fosse possível investigar a segunda questão de pesquisa (Questão de Pesquisa 2, Capítulo 3), era necessário escolher uma ferramenta em que fosse possível realizar o mapeamento dos interesses dos sistemas, para avaliar o reúso das estratégias de detecção de anomalias segundo um interesse específico. Sendo assim, a partir das necessidades de: ter acesso às estratégias de detecção de anomalias, através da adaptação das métricas e limiares, e a possibilidade de avaliar as ocorrências de anomalias tendo em vista o mapeamento de interesses dos sistemas em estudo, a ferramenta escolhida foi SCOOP[5].

SCOOP é uma ferramenta que utiliza estratégias de detecção a partir da composição de métricas e limiares, segundo a proposta de (Lanza e Marinescu, 2006), e já foi usada com sucesso em estudos empíricos anteriores, tais como

[5] Disponível em: http://www.inf.puc-rio.br/~ibertran/SCOOP/

PUC-Rio - Certificação Digital Nº 1112618/CA

aqueles reportados em (Macía et al., 2012a), (Macía et al., 2012b) e (Macía et al., 2011). Atualmente, o SCOOP está disponível como um plug-in para o Eclipse, integrado com ferramentas de coleta de métricas. Nesse estudo, SCOOP foi usado com as ferramentas de coletas de métricas: Together[6] e Understand[7].

3.8. Interesses Mapeados nos Sistemas em Estudo

Para avaliar se é possível diminuir a ocorrência de falsos positivos, ao considerar as características de conjuntos de elementos com responsabilidades bem definidas (Questão de Pesquisa 2, Capítulo 3), foi necessário realizar o mapeamento dos interesses nas classes dos seis sistemas que representam o domínio em estudo. Essa atividade também foi realizada com o apoio dos especialistas do domínio, tendo em vista que o mapeamento adequado dos interesses para os sistemas que representam o domínio em estudo, somente poderia ser realizado por quem participa no dia a dia do desenvolvimento e manutenção desses sistemas.

Durante o mapeamento dos interesses, primeiro foram observados os interesses mais gerais, como, por exemplo, interface, persistência e recursos auxiliares. Os interesses mais gerais são aqueles que estão presentes no domínio em estudo e em outros domínios. Em seguida, foi realizado o mapeamento dos interesses relacionados especificamente ao domínio das aplicações. Através do acompanhamento dos especialistas do domínio pode-se garantir a identificação dos elementos de código para cada um dos interesses mapeados para o domínio. Os interesses mais gerais são:

- Autenticação/Segurança (Tabela 4 - a): os elementos desse interesse são responsáveis por garantir o controle de acesso ao sistema. O funcionamento dos elementos desse interesse segue protocolo *Lightweight Directory Access Protocol*, ou LDAP[8].

[6] Disponível em: http://www.borland.com/products/together/try/

[7] Disponível em: http://www.scitools.com/

[8] Mais informações sobre esse protocolo em: http://support.microsoft.com/kb/196455/en-us

PUC-Rio - Certificação Digital Nº 1112618/CA

- Auxiliar (Tabela 4 - b): os elementos desse interesse são bibliotecas para apoio aos métodos, sendo bibliotecas desenvolvidas por outras empresas e bibliotecas desenvolvidas pela própria equipe.
- Exceção (Tabela 4 - c): são elementos que garantem o controle das exceções que possam ocorrer durante a execução do sistema.
- Interface (Tabela 4 - d): são elementos relacionados estritamente com os objetos que compõem a interface do sistema.
- Persistência (Tabela 4 - e): elementos relacionados à integridade dos dados de negócio e controle da configuração do usuário;

Os interesses específicos do domínio são:

- Ação (Tabela 4 - f): os elementos desse interesse obtêm os parâmetros da interface e dão início aos cálculos para geração de indicadores financeiros. Além disso, esses elementos dão início às operações de alteração de parâmetros relacionados com a geração dos indicadores. Elementos de Ação também são responsáveis por carregar a página inicialmente ou preparar a interface através de dados das carteiras de investimento armazenadas previamente. Além disso, alguns elementos desse interesse funcionam de forma assíncrona.
- Engine (Tabela 4 - g): elementos que possuem operações de negócio, através da geração ou cálculo de indicadores financeiros. Recebe os dados, calcula os indicadores e gera resultados de saídas para serem exibidos em gráficos na interface ou relatórios;
- Indicadores (Tabela 4 - h): são elementos de código que fornecem dados para a interface. Esses dados são indicadores financeiros, tais como: retorno, volatilidade, entre outros;
- Serviços (Tabela 4 - i): são elementos que armazenam os parâmetros usados nas análises dos dados, além de classes mais complexas, diretamente relacionadas com a manipulação de arquivos XML e armazenamento de dados da sessão.
- Tarefas (Tabela 4 - j): São elementos que implementam *threads* (ou linhas de execução), executadas de acordo com uma programação. Esses

PUC-Rio - Certificação Digital Nº 1112618/CA

elementos verificam informações relacionadas à integridade de dados, como, por exemplo, verificações são limpeza e preenchimento de controle de armazenamento temporário de dados históricos, de informações de ativos, frequência de cotações de ativos, etc.

Apesar dos interesses possuírem estruturas diferentes, na fase de estratégias orientadas a interesses (seção 3.3.4), os interesses são vistos de maneira isolada. Essa decisão é determinante para a avaliação das estratégias orientadas a interesses. Mesmo assim, pelo acompanhamento dos especialistas do domínio, verificou-se que, de fato, existia um conjunto razoável de interesses recorrentes nos sistemas escolhidos para representar o domínio em estudo.

A partir da Tabela 4 é possível observar a recorrência dos interesses mapeados nos sistemas que representam o domínio em estudo. Os interesses escolhidos são representados por letras minúsculas na tabela, mas nomeados nas próximas seções. Nessa mesma tabela, é possível ver que três interesses estão associados apenas ao sistema A. São eles: *Engine*, Indicadores e Tarefas. Mesmo que não haja a recorrência desses interesses, eles são mantidos aqui para que possam ser revistos em estudos futuros.

A Tabela 5 descreve o tamanho dos sistemas escolhidos para o estudo, em número de linhas de código (NLOC) e número de classes. Mesmo que os sistemas variem em tamanho, segundo os especialistas do domínio, a proximidade estrutural dos sistemas é observada através da recorrência de interesses mapeados nos sistemas.

Tabela 4 - Mapeamento dos interesses recorrentes nos sistemas em estudo

Sistema	Interesses mapeados									
	a	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*
A	X	X	X	X	X	X	X	X	X	X
B	X	X	X	X	X	X			X	
C	X	X	X	X	X	X			X	
D	X	X	X	X	X	X			X	
E	X	X	X	X	X	X			X	
F	X	X	X	X	X	X			X	

PUC-Rio - Certificação Digital Nº 1112618/CA

Tabela 5 - Descrição do tamanho dos sistemas usados no estudo

Sistema	Nº de linhas de código	Nº de classes
A	21599	161
B	10011	81
C	12504	130
D	5935	41
E	31766	150
F	21602	149

Apesar de estruturalmente semelhante, o compartilhamento de código não é visto entre os sistemas escolhidos para representar o domínio em estudo. Esses sistemas compartilham apenas algumas bibliotecas de cálculo de indicadores financeiros. Essas bibliotecas também foram desenvolvidas por outra empresa, antes que a empresa do contexto em estudo absorvesse o conjunto de aplicações de análise de risco financeiro.

Além disso, os sistemas escolhidos para o estudo possuem finalidades distintas, apesar de estruturalmente semelhantes. Em um dos sistemas é possível ter acesso aos dados históricos dos ativos, adquiridos através de serviços de terceiros, além de permitir a manutenção de estruturas de dados relacionadas aos mais variados cálculos para a geração de indicadores financeiros. Em outro sistema é possível realizar a confecção de uma carteira de ativos, a partir dos ativos cadastrados no banco de dados. Dessa forma, os riscos de uma carteira de investimentos podem ser avaliados, em outro sistema desse mesmo conjunto de sistemas, a partir de análises tradicionais, que estão pré-definidas. Em outro sistema, o mais complexo dos sistemas escolhidos, também é possível realizar a análise de riscos financeiros através da escolha de um conjunto de ativos, porém, sendo possível criar novos tipos de análises, que vão além das análises tradicionais já apresentadas no outro sistema. Em um quinto sistema desse conjunto de sistemas, é possível realizar a criação e análise de fundos de investimentos, para a posterior análise de riscos através de análises tradicionais e, por fim, o último sistema permite a análise de ativos financeiros, considerando a disponibilidade geográfica desses ativos. A partir desse sistema é possível gerar análises que podem ser verificadas em outro sistema desse conjunto de sistemas.

Nos sistemas escolhidos para o estudo também é possível observar a ocorrência de muitos cálculos complexos. Esses cálculos fazem a análise de dados dos ativos através da análise histórica dos dados. Além disso, algumas simulações

PUC-Rio - Certificação Digital Nº 1112618/CA

são realizadas para avaliar um risco financeiro e, durante a geração desses indicadores, também são realizados cálculos mais complexos.

PUC-Rio - Certificação Digital Nº 1112618/CA

4
Resultados e Discussões

No capítulo 3, detalhamos o estudo empírico conduzido nessa dissertação. Esse estudo é guiado por duas questões de pesquisa (Capítulo 3) e está definido em três fases. Através dessas três fases, investigamos o grau de reúso de estratégias de detecção definidas a partir de características do domínio em estudo.

O objetivo da primeira etapa, chamada de fase de ajustes de limiares, é definir estratégias de detecção de anomalias que tenham percentual de falsos positivos abaixo de 33%, para duas aplicações do domínio em estudo. A segunda etapa, chamada de fase de avaliação do reúso das estratégias de detecção, tem por objetivo avaliar se as estratégias definidas na fase de ajustes de limiares possuem alto grau de reúso em outros quatro sistemas do mesmo domínio. Finalmente, a última etapa é chamada de fase de interesses do domínio. Essa fase tem como objetivo verificar se o percentual de falsos positivos das estratégias pode ser melhorado, tendo em vista a aplicação das estratégias de detecção apenas em classes de um mesmo interesse, sendo esse interesse recorrente nos programas do domínio em estudo.

Antes de descrever os resultados desse estudo, devemos esclarecer que, devido às nossas questões de pesquisa (Capítulo 3), não poderíamos usar outros métodos empíricos (por exemplo, experimentos). Dessa forma, os resultados descritos aqui estão relacionados a um estudo de caso e restritos para fins de generalização. Além disso, nesse estudo empírico, as ocorrências consideradas como falsos negativos são as ocorrências que já foram classificadas pelos especialistas como uma anomalia, porém, caso ocorra uma alteração nos limiares da estratégia de detecção de anomalias em questão, essa ocorrência deixa de ser identificada (mesmo sabendo-se que é uma ocorrência de anomalia conhecida).

Neste capítulo, cada seção está relacionada a uma das três fases realizadas no estudo empírico. Assim, a seção 4.1 apresenta os resultados da primeira fase, fase de ajustes dos limiares; a seção 4.2 apresenta os resultados da segunda fase, fase de avaliação do reúso das estratégias de detecção; a seção 4.3 apresenta os

PUC-Rio - Certificação Digital Nº 1112618/CA

resultados da terceira fase, chamada de fase de estratégias orientadas a interesses. Ainda neste capítulo, apresentamos o resultado de uma avaliação das estratégias de detecção de anomalias definidas nesse estudo empírico em várias versões dos sistemas usados no estudo. Essa avaliação tem o objetivo de verificar se as estratégias de detecção de anomalias definidas para o domínio em estudo poderiam ser úteis, ao identificar anomalias mesmo durante a evolução histórica dos sistemas. Assim, os resultados dessa avaliação de versões são apresentados na seção 4.4. Na seção 4.5 são apresentadas as lições aprendidas e, finalmente, na seção 4.6 apresentamos algumas ameaças à validade desse estudo e descrevemos o que foi feito para amenizar essas ameaças.

Neste capítulo, tabelas são usadas para apresentar os resultados das três fases propostas para esse estudo empírico. Dessa forma, as colunas das tabelas seguem uma nomenclatura específica e abreviações padronizadas. Assim, a coluna "NO/FP" indica os valores de: números de ocorrências identificadas pelas estratégias de detecção de anomalias (antes do símbolo "/") e número de falsos positivos identificados pelo especialista do domínio (após o símbolo "/"); a coluna "%FP" indica o percentual de falsos positivos identificados pelo especialista do domínio, em relação ao número total de ocorrências de identificadas pelas estratégias de detecção de anomalias. Além disso, destacamos em negrito os resultados em que o percentual de falsos positivos ficou acima de 33%. Dessa forma é mais fácil identificar os casos em que o percentual de falsos positivos de uma estratégia de detecção de anomalias, quando aplicada a um determinado sistema, gerou um resultado pior do que o mínimo de precisão aceitável.

Além disso, é necessário lembrar novamente que nesse estudo foram usados sufixos específicos nos nomes das estratégias usadas, para que fosse possível identificar as estratégias definidas inteiramente pelos especialistas e as estratégias de detecção em que os limiares foram ajustados a partir de estratégias de detecção escolhidas na literatura. Dessa forma, identificamos as estratégias de detecção definidas inteiramente pelos especialistas pelo sufixo "Esp" e identificamos as estratégias de detecção originadas a partir da literatura através do sufixo "Lit". Em especial, o sufixo "EspLoc" está relacionado à estratégia de detecção definida inteiramente pelos especialistas que possui a métrica LOC (Lines of Code) e o sufixo "EspNom" está relacionado à estratégia de detecção definida inteiramente pelos especialistas que possui a métrica NOM (Number of Methods).

PUC-Rio - Certificação Digital Nº 1112618/CA

4.1. Resultados da fase de ajustes dos limiares

Como mencionado, o objetivo da fase de ajustes dos limiares foi refinar os limiares das estratégias definidas juntamente com os especialistas do domínio, a partir do percentual de falsos positivos encontrado. Assim, os sistemas A e B foram escolhidos para a primeira avaliação das estratégias de detecção definidas juntamente com os especialistas. Essa escolha se deve especificamente à disponibilidade imediata do especialista E1 (seção 3.2). Os resultados mostrados a seguir foram obtidos a partir de duas sessões de investigação (uma para cada sistema).

A Tabela 6 apresenta o percentual de falsos positivos para as sete estratégias da primeira fase, quando aplicadas aos sistemas A e B. Nessa tabela, é possível identificar que duas estratégias - *God Class EspLoc* e *God Class Lit* - resultaram em mais do que 33% de falsos positivos. Em um primeiro momento, isso significa que, para a estratégia de detecção *God Class Lit*, mesmo que as métricas utilizadas sejam recorrentes da literatura, os limiares definidos juntamente com os especialistas não foram muito bons. A partir desse resultado, como um exercício, investigamos se havia a possibilidade de reduzir o percentual de falsos positivos para as duas estratégias de detecção, *God Class EspLoc* e *God Class Lit*, apenas alterando os limiares dos seus componentes. Para isso, avaliamos os valores das métricas dos elementos de cada conjunto de falsos positivos, considerando que cada conjunto de falsos positivos contém elementos dos dois sistemas.

Para a estratégia *God Class EspLoc*, não foi possível identificar um limiar que pudesse eliminar elementos do conjunto de falsos positivos. Isso aconteceu porque os elementos identificados pela estratégia são bastante diferentes. Por exemplo, os falsos positivos gerados a partir da *God Class EspLoc* estão relacionados a quatro interesses diferentes: Ação, Auxiliar, Persistência e Serviços. Também verificamos o caso em que dois elementos que pertencem ao mesmo interesse, com valores muito próximos para as métricas dessa estratégia, foram classificados de maneira distinta. Nesse caso, o elemento que possuía os maiores valores para as métricas foi considerado um falso positivo, enquanto o elemento que possuía os valores mais baixos foi considerado uma anomalia.

PUC-Rio - Certificação Digital Nº 1112618/CA

Assim, percebemos que essa mistura de elementos, muito parecidos e classificados de maneira diferente, acaba impedindo que um ajuste nos limiares elimine alguma parte dos falsos positivos. De qualquer forma, considerando que o resultado para o sistema A foi de 22% de falsos positivos, ficou decidido manter essa estratégia para a fase seguinte – fase de avaliação de reúso das estratégias de detecção.

Depois, para a estratégia *God Class Lit*, ao contrário da estratégia de detecção anterior, foi possível modificar os limiares para duas métricas dessa estratégia de detecção, de forma a eliminar todos os elementos do conjunto de falsos positivos. Dessa forma, alteramos dois limiares da estratégia *God Class Lit* e reduzimos o percentual de falsos positivos dessa estratégia para 0%, sem criar falsos negativos.

Assim, a estratégia *God Class Lit* foi refinada segundo os limiares dos elementos identificados como falsos positivos, a partir das duas sessões de investigação dessa fase. Dessa forma, segundo os critérios descritos anteriormente, as sete estratégias de detecção definidas na fase de ajustes foram mantidas para a fase de reúso.

PUC-Rio - Certificação Digital Nº 1112618/CA

Tabela 6 - Resultado de nº de ocorrências e nº de falsos positivos da primeira fase

Estratégia	Sistema A		Sistema B	
	NO/FP	%FP	NO/FP	%FP
God Class EspLoc	27/6	22%	17/7	**41%**
God Class EspNom	15/3	20%	5/1	20%
God Class Lit	4/3	**75%**	2/0	0%
Long Method Esp	30/0	0%	19/3	16%
Long Method Lit	1/0	0%	0/0	0%
Shotgun Surgery Lit	61/12	20%	25/6	24%
Shotgun Surgery Esp	21/0	0%	7/1	14%

Na fase de avaliação de reúso das estratégias de detecção, como veremos a seguir, essas sete estratégias foram então aplicadas a outros quatro sistemas do mesmo domínio.

4.2. Resultados da fase de avaliação do reúso de estratégias de detecção

Na fase de avaliação do reúso de estratégias de detecção, objetivamos observar se é possível reusar as estratégias diretamente (tal qual definidas na fase

de ajustes) em outros sistemas do mesmo domínio. A apresenta as estratégias de detecção definidas para a fase de reúso.

Tabela 7 - Estratégias de detecção definidas para a segunda fase do estudo

Anomalia	Estratégia
God Class EspLoc	(LOC > 150) e (CBO > 6)
God Class EspNom	(NOM > 15) e (CBO > 6)
God Class Lit	(ATFD > 6) e (WMC > 46) e (TCC < 11)
Long Method Esp	(LOC > 50) e (CC > 5)
Long Method Lit	(LOC > 50) e (CC > 6) e (MaxNesting > 5) e (NOAV > 3)
Shotgun Surgery Lit	(FanOut > 16)
Shotgun Surgery Esp	(CC > 7) e (AM > 7)

Assim, depois de quatro sessões de investigação, uma para cada sistema, os resultados das sete estratégias são apresentados na Tabela 8 e na Tabela 9.

De acordo com a Tabela 8 e a Tabela 9, podemos observar que a estratégia *God Class EspLoc* manteve um comportamento semelhante à fase de reúso: mesmo tendo o resultado de falsos positivos abaixo de 33% para dois sistemas, apresentou resultados acima de 33% de falsos positivos em outros dois sistemas. Da mesma forma, a estratégia *God Class EspNom* apresentou resultados acima de 33% de falsos positivos em dois sistemas, além de um sistema em que essa estratégia resultou exatamente em 33% de falsos positivos. Nessa situação, a estratégia *God Class EspNom* mostrou-se adequada na fase de ajustes, porém teve um resultado ruim na fase de avaliação do reúso. Nessa fase, o conjunto de elementos classificados como falso positivo para a estratégia *God Class EspNom* estava composto principalmente (46% e 38%) por elementos dos interesses Interface e Persistência. Em resumo, as duas estratégias definidas juntamente com os especialistas, para a anomalia *God Class*, tiveram os piores resultados na fase de avaliação do reúso.

Além dessas, outras duas estratégias resultaram em mais do que 33% de falsos positivos. Primeiro, a estratégia *Shotgun Surgery Lit* teve este desempenho ruim quando aplicada ao sistema C. Esse caso resultou em 76% de falsos positivos, sendo que 62% dos falsos positivos pertencem ao interesse Ação. Nesse caso, os métodos indicados como falsos positivos são usados para controle dos dados da sessão do usuário. Segundo, a estratégia *Long Method Esp*, quando aplicada ao sistema D, também não obteve bom desempenho. Essa situação resultou em 40% de falsos positivos, sendo que os falsos positivos pertencem aos

PUC-Rio - Certificação Digital Nº 1112618/CA

interesses Auxiliar e Persistência, sendo 50% para cada um. Aqui, as classes identificadas como falsos positivos estão relacionadas ao registro de *log* de requisições ao servidor, no caso do interesse Auxiliar, e ao cálculo de um indicador financeiro, a partir de uma estrutura de dados de um *Bean*, no caso do interesse Persistência.

A partir do resultado da fase de ajustes de limiares e da fase de avaliação de reúso, temos apenas oito situações em que as estratégias resultaram em um número de falsos positivos aquém do limite aceitável, isto é, com taxa de erros acima de 33%. De forma geral, é possível perceber que houve um reúso satisfatório, já que o reúso com sucesso das estratégias ocorreu em 81% dos casos: considerando seis sistemas e sete estratégias, temos 42 situações de investigação. Cada situação de investigação é representada pela investigação do reúso de uma estratégia de detecção, quando aplicada a um sistema do domínio em estudo. Dessa forma, das 42 situações de investigação, em apenas oito situações o resultado foi aquém do limite aceitável. Este alto reúso foi alcançado tanto com as estratégias definidas em conjunto com o especialista (*God Class EspLoc*, *God Class EspNom*, *Long Method Esp* e *Shotgun Surgery Esp*), quanto com as estratégias com limiares definidos na literatura (*God Class Lit*, *Long Method Lit* e *Shotgun Surgery Lit*).

Considerando apenas as situações de investigação das estratégias com limiares definidos na literatura, em 5% dos casos temos resultados aquém do limite aceitável. Por outro lado, considerando apenas as situações de investigação das estratégias definidas inteiramente pelos especialistas, temos resultados aquém do limite aceitável em 25% dos casos. Apesar de o resultado ser melhor nos casos das estratégias com limiares definidos na literatura, percebeu-se que duas estratégias de detecção de anomalias escolhidas a partir da literatura, resultaram em 0% de falsos positivos em todos os casos em que encontraram ocorrências. Essas duas estratégias foram usadas como propostas em (Lanza e Marinescu, 2006), apenas com alterações nos limiares. Mesmo que essas duas estratégias resultassem em 100% de anomalias identificadas, essas estratégias acabaram não identificando ocorrências que foram identificadas pelas estratégias definidas inteiramente pelos especialistas e que foram classificadas como anomalias, a partir da opinião qualitativa dos especialistas do domínio. Essa evidência mostra que, apesar de serem estratégias conhecidas da literatura, essas estratégias de detecção

PUC-Rio - Certificação Digital Nº 1112618/CA

não colaboraram muito para a identificação de sintomas de problemas de código nos sistemas escolhidos para representar o domínio em estudo. Considerando o resultado dessas duas estratégias, decidimos flexibilizar os limiares associados à cada métrica, para que fosse possível avaliar se as estratégias poderiam identificar os elementos que já haviam sido classificados como anomalias. Mesmo assim, não foi possível identificá-los. Isso nos leva a crer que as métricas definidas por essas estratégias não resultam em boas estratégias para o domínio em estudo.

Os elementos identificados como falsos positivos pela estratégia *God Class EspLoc* (na maioria, nos sistemas B, C e D), na sua maioria, estão relacionados a interesses como auxiliar e persistência. Os outros interesses identificados por essa estratégia são Interface, Ação, Serviços e Tarefas. A estratégia *God Class EspNom* identificou falsos positivos principalmente nos sistemas D e F, na maioria relacionados aos interesses Interface e Persistência. Depois, os interesses Auxiliar, Ação e Indicadores. A estratégia *Long Method Esp* resultou em falsos positivos, na sua maioria, nos interesses Auxiliar e Persistência. Depois, no interesse Ação. Por fim, a estratégia *Shotgun Surgery Lit* identificou falsos positivos principalmente nos interesses Ação e Auxiliar. Além disso, identificou falsos positivos nos interesses Interface, Serviços, Engine, Indicadores e Tarefas. Dessa forma, nessa fase, os interesses Auxiliar, Persistência e Ação foram os interesses que mais indicaram falsos positivos.

A partir dos resultados da fase de avaliação de reúso, podemos concluir que existe certa tendência de comportamento padrão entre sistemas que pertencem a um mesmo domínio e em ambientes semelhantes: os desenvolvedores estão na mesma organização, e seguindo práticas, ferramentas e *frameworks* semelhantes. Mesmo assim, uns poucos casos peculiares podem encorajar e desencorajar as adaptações nos limiares das estratégias. Considerando as duas estratégias que geraram mais casos de falsos positivos acima de 33%, *God Class EspLoc* e *God Class EspNom*, decidimos, novamente, investigar se era possível ajustar os limiares das estratégias, para diminuir o número de falsos positivos. Além disso, observamos que certas características comuns entre os sistemas certamente podem influenciar positivamente o grau de reúso das estratégias.

Assim, considerando a estratégia *God Class EspNom*, quando aplicada ao sistema F, observamos o conjunto dos elementos classificados como falsos positivos (50% do resultado). Nesse conjunto, 75% dos elementos possuem o

PUC-Rio - Certificação Digital Nº 1112618/CA

valor do componente CBO igual a 10. Assim, olhando especificamente para esse caso, podemos alterar o valor da componente CBO para 11 (resultando na estratégia [NOM > 15 e CBO > 11]) e o número de falsos positivos cai para 20% no sistema F (era 50%). Em contrapartida, a partir dessa alteração na estratégia *God Class EspNom*, o número de falsos positivos aumenta para 27% e 40%, nos sistema A e C (era 20% e 33%), e caem para 0%, 12% e 50% nos sistemas B, E e D (eram 20%, 30 e 80%). Mesmo que o número de falsos positivos aumente no sistema C, o resultado está pouco acima do limite aceitável.

Mesmo assim, outro exemplo pode ser mais criterioso. Consideramos agora, a mesma estratégia *God Class EspNom*, porém no caso em que é aplicada ao sistema D. Aqui, temos 80% de falsos positivos. Da mesma forma, ao investigar os elementos classificados como falsos positivos, verificamos que em 100% dos casos o valor do componente CBO é menor que 18. Assim, considerando especificamente esse caso, podemos alterar o valor da componente CBO para 18 (resultando na estratégia [NOM > 15 e CBO > 18]) e o percentual de falsos positivos cai para 0%, nos sistemas B, D, E e F. Em contrapartida, o percentual de falsos positivos se mantém em 33% no sistema C e aumenta para 25% no sistema A. A partir desse último exemplo, percebemos que, ao realizar um ajuste mais criterioso, é possível diminuir para 0% o percentual de falsos positivos em quatro dos seis sistemas.

A partir dos exemplos citados acima, considerando um pequeno ajuste no limiar de uma estratégia de detecção de anomalias, mostramos que é possível perceber um melhor equilíbrio dentro dos sistemas escolhidos para representar o domínio em estudo. Nessa situação, foi possível perceber um comportamento semelhante entre os sistemas. Em todos os sistemas, a estratégia *God Class EspNom* identificou elementos no interesses Auxiliar, que é mais geral. Além disso, identificou elementos no interesse Interface (que é mais específico) em cinco dos seis sistemas.

Apesar de existirem casos em que uma alteração no limiar pode ser considerada para eliminar falsos positivos, existem casos em que a distribuição dos elementos identificados pelas estratégias se torna impeditivo para a eliminação de falsos positivos. Por exemplo, a estratégia *God Class EspLoc*, quando aplicada aos sistemas B, C e D, apresentou um número de falsos positivos acima do esperado. Assim, da mesma forma que na fase de ajustes dos limiares,

PUC-Rio - Certificação Digital Nº 1112618/CA

decidimos investigar os elementos identificados por essa estratégia, com o objetivo de eliminar falsos positivos. Igualmente ao que ocorreu naquela fase, a partir dos sistemas C e D foi possível verificar uma grande mistura de falsos positivos e anomalias que ocorrem simultaneamente entre os limiares impostos pela estratégia de detecção.

Por exemplo, as classes *AbstractBackgroundTask* e *UtilitiesBackgroundTask* possuem LOC igual a 195 e CBO igual a 13, e LOC igual a 183 e CBO igual 24, respectivamente, e pertencem ao mesmo interesse - Tarefas. Apesar de estruturalmente similares e pertencerem ao mesmo interesse, a classe com menos acoplamento foi classificada pelo especialista como uma anomalia *God Class* e a classe com mais acoplamento foi classificada pelo especialista como um falso positivo. Outros exemplos desse comportamento heterogêneo foram verificados ainda ao observar as estratégias *Long Method Esp*, quando aplicada ao sistema D, e a estratégia *Shotgun Surgery Lit*, quando aplicada sistema C. A partir desses exemplos, percebemos que, quando existe uma grande mistura de falsos positivos e anomalias que ocorrem simultaneamente entre os elementos identificados por uma estratégia de detecção de anomalias, uma alteração nos limiares que possa diminuir o número de falsos positivos é desencorajada.

De maneira geral, olhando para cada caso específico, verificamos que o reúso pode ser melhorado em alguns casos, sem gerar um efeito colateral. Mesmo assim, em alguns outros casos, para realizar uma melhoria no reúso das estratégias de detecção de anomalias, é possível que sejam criados falsos negativos. Dessa forma, pode ser verificado um limite para o grau de reúso das estratégias de detecção. Pelo que foi verificado, nos casos onde existe uma distribuição heterogenia entre os elementos identificados por uma estratégia de detecção, o reúso dessa estratégia é limitado.

No total, dos sete casos que excederam o limiar de 33% (mais um que esteve exatamente nesse limiar), em quatro casos existe pelo menos um cenário em que duas classes com estruturas similares foram classificadas uma como anomalia e outra como um falso positivo. Isso mostrou que existem casos em que é impossível definir um limiar que elimine boa parte dos falsos positivos sem gerar novos falsos negativos. Como uma consequência direta, pode-se afirmar que existe um limite no grau de reúso das estratégias, isto é, uma nova adaptação na

PUC-Rio - Certificação Digital Nº 1112618/CA

tentativa de diminuir o percentual de falsos positivos pode aumentar o número de falsos negativos.

Tabela 8 - Ocorrências de falsos positivos e anomalias dos sistemas A, B e C

Estratégia	Sistemas					
	A		B		C	
	NO/FP	%FP	NO/FP	%FP	NO/FP	%FP
God Class EspLoc	27/6	22%	17/7	**41%**	17/7	**41%**
God Class EspNom	15/3	20%	5/1	20%	6/2	**33%**
God Class Lit	1/0	0%	2/0	0%	0/0	0%
Long Method Esp	30/0	0%	19/3	16%	6/1	17%
Long Method Lit	1/0	0%	0/0	0%	0/0	0%
Shotgun Surgery Lit	61/12	20%	25/6	24%	17/13	**76%**
Shotgun Surgery Esp	21/0	0%	7/2	28%	0/0	0%

Tabela 9 - Ocorrências de falsos positivos e anomalias dos sistemas D, E e F

Estratégia	Sistemas					
	D		E		F	
	NO/FP	%FP	NO/FP	%FP	NO/FP	%FP
God Class EspLoc	10/8	**80%**	30/5	17%	24/7	29%
God Class EspNom	5/4	**80%**	10/3	30%	8/4	**50%**
God Class Lit	0/0	0%	3/0	0%	2/0	0%
Long Method Esp	5/2	**40%**	40/2	5%	26/3	12%
Long Method Lit	0/0	0%	4/0	0%	0/0	0%
Shotgun Surgery Lit	13/1	8%	48/1	2%	44/2	5%
Shotgun Surgery Esp	1/0	0%	12/0	0%	9/0	0%

Nos sistemas A e E, foi possível observar que as estratégias tiveram reúso total. O sistema E é formado basicamente por elementos do interesse Auxiliar (60%) e elementos do interesse Ação (30%). Dessa forma, mesmo que os falsos positivos encontrados pertencessem a esses interesses, o percentual de falsos positivos ficou abaixo do limite esperado. Já o sistema A é formado por elementos do interesse Ação (40%), elementos do interesse Serviços (21%), Auxiliar (18%) e outros interesses com menos elementos, como Engine, Persistência, Indicadores e Autenticação. Da mesma forma, para um sistema com essa distribuição de interesses o número de falsos positivos para as estratégias definidas não excedeu os 33%.

4.3.Resultados da fase de estratégias orientadas a interesses

Na terceira fase definida para esse estudo empírico, fase de estratégias orientadas a interesses, investigou-se a possibilidade de diminuir a ocorrência de falsos positivos ao observar estritamente os elementos de cada um dos interesses

PUC-Rio - Certificação Digital Nº 1112618/CA

mapeados nos seis sistemas que representam o domínio em estudo. Nessa fase, são consideradas as mesmas estratégias de detecção aplicadas aos sistemas como um todo, durante a fase avaliação de reúso (). De qualquer forma, é necessário esclarecer que nessa fase são observados apenas os interesses que possuem recorrência significativa nos seis sistemas avaliados. Assim, os interesses *Engine*, Indicadores e Tarefas, que estão relacionados a apenas um sistema, não foram considerados.

Ao considerar estritamente os elementos de cada interesse, observamos se: (i) haveria potencial benefício em utilizar estratégias de detecção de anomalias específicas para cada interesse do domínio – este caso foi observado quando as estratégias tiveram um número maior do que 33% falsos positivos, e (ii) foi suficiente o uso de estratégias de detecção de anomalias no programa como um todo – este caso foi observado quando as estratégias de detecção tiveram um número menor do que 33% falsos positivos.

As tabelas a seguir apresentam o número de ocorrências (NO) e percentagem de falsos positivos (FP) quando são observados estritamente os elementos em cada interesse mapeado nos sistemas. A partir dos resultados apresentados nas tabelas a seguir (Tabela 10, Tabela 11, Tabela 12, Tabela 13 e Tabela 14), percebe-se que não haveria necessidade de especialização das estratégias de detecção de anomalias para os interesses: (i) tanto para os casos de interesses Autenticação/Segurança e Auxiliar, que são mais gerais (isto é, podem ocorrer frequentemente em aplicações de outros domínios), (ii) como para os interesses Ação e Serviços, que são interesses característicos dos sistemas que representam o domínio em estudo. Nesse sentido, ajustar os limiares para as estratégias de detecção de anomalias, considerando o mapeamento de interesses, não seria benéfico para reduzir significativamente o percentual de falsos positivos nos casos acima. Por outro lado, note que o contrário pode ser dito para o caso dos interesses Interface e Persistência. Nesses casos, nota-se nas tabelas que os números de falsos positivos, independentemente da anomalia analisada, estão bem acima do resultado esperado em vários casos.

Ao investigar as características dos elementos classificados nos interesses Interface e Persistência, foi possível observar alguns comportamentos peculiares. Por exemplo, os elementos identificados pelas estratégias de detecção de anomalias, a partir do interesse Interface, se resumem a classes que se repetem em

PUC-Rio - Certificação Digital Nº 1112618/CA

cinco dos seis sistemas em estudo. Essas classes estão repetidas entre os sistemas em estudo e não fazem compartilhamento de código. São elas *AppPageContext* e *FormBeanProperty*. Essas duas classes são usadas em cinco, dos seis sistemas analisados, e foram detectadas pelas estratégias *God Class EspLoc*, *God Class EspNom* e *Shotgun Surgery Lit*. Dessa forma, pode-se concluir que o percentual de falsos positivos acima do esperado é resultado especificamente de uma classificação divergente entre os especialistas. Já no interesse Persistência, 33 elementos, entre classes e métodos, foram identificados pelas estratégias de detecção de anomalias. Desse conjunto, 64% são falsos positivos. Ao observar especificamente os falsos positivos, resultantes desse interesse específico, são observados dois tipos de elementos. O primeiro conjunto (52%) desses elementos está relacionado a objetos do banco de dados, ou seja, elementos com pouca lógica e muitos métodos *getters* e *setters*. O segundo conjunto (48%) é formado por elementos relacionados à configuração do *framework* Struts[10].

Tabela 10 - Percentual de falsos positivos por interesse – Shotgun Surgery Esp

Interesse	NO/FP	% FP
Ação	24/0	0%
Auxiliar	16/2	13%
Persistência	2/0	0%
Serviços	5/0	0%

Tabela 11 - Percentual de falsos positivos por interesse – God Class EspLoc

Interesse	NO/FP	% FP
Ação	30/7	23%
Autenticação/segurança	2/0	0%
Auxiliar	52/12	23%
Interface	10/6	**60%**
Persistência	13/11	**84%**
Serviços	10/3	30%

Tabela 12 - Percentual de falsos positivos por interesse – God Class EspNom

Interesse	NO/FP	% FP
Ação	6/2	33%
Auxiliar	19/3	16%
Interface	10/6	**60%**
Persistência	7/6	**86%**
Serviços	4/0	0%

[10] Mais detalhes em: http://struts.apache.org/

PUC-Rio - Certificação Digital Nº 1112618/CA

Tabela 13 - Percentual de falsos positivos por interesse – Long Method Esp

Interesse	NO/FP	% FP
Ação	46/3	7%
Autenticação/segurança	2/0	0%
Auxiliar	52/5	10%
Persistência	13/5	**38%**
Serviços	7/0	0%

Tabela 14 - Percentual de falsos positivos por interesse – Shotgun Surgery Lit

Interesse	NO/FP	% FP
Ação	83/15	18%
Autenticação/segurança	1/0	0%
Auxiliar	81/12	15%
Exceção	1/0	0%
Interface	5/2	**40%**
Persistência	16/5	31%
Serviços	11/2	18%

4.4.Avaliação histórica das estratégias de detecção

Como forma de ampliar a validação das estratégias de detecção formadas na fase de ajustes de limiares, avaliamos versões posteriores dos mesmos sistemas escolhidos para representar o domínio em estudo. Dessa forma, esse exercício tem como objetivo realizar uma avaliação histórica das estratégias de detecção de anomalias, através de mais duas versões de cada sistema escolhido para representar o domínio em estudo. Todas as três versões de cada sistema foram geradas já na empresa em que o estudo foi feito.

Nas tabelas a seguir, são apresentadas as três versões avaliadas para cada sistema em estudo. Nessas tabelas, a segunda linha (primeira versão detalhada dos sistemas) representa os sistemas que foram avaliados durante as três fases definidas para o estudo empírico dessa dissertação.

A partir dos resultados é possível verificar que em pelo menos 83% dos casos (35 vezes) a quantidade de ocorrências, identificadas pelas estratégias de detecção de anomalias, cresce com o decorrer do desenvolvimento dos sistemas. No restante dos casos não há esse crescimento no decorrer das versões. Dessa forma, podemos concluir que o apoio de especialistas do domínio em estudo é uma alternativa viável para a definição de estratégias de detecção de anomalias um bom grau de reúso mesmo no decorrer das versões dos sistemas em estudo.

PUC-Rio - Certificação Digital Nº 1112618/CA

Tabela 15 - Avaliação histórica no sistema A

Sistema	Versão	Data	GC EspL	GC EspN	GC Lit	LM Esp	LM Lit	SS Lit	SS Esp
A	1-10-0	2011-07-04	27	15	1	30	1	61	21
A	1-22-5	2012-05-07	38	33	3	43	3	80	27
A	1-35-1	2013-01-17	42	38	4	56	5	101	31

Tabela 16 - Avaliação histórica no sistema B

Sistema	Versão	Data	GC EspL	GC EspN	GC Lit	LM Esp	LM Lit	SS Lit	SS Esp
B	1-5	2010-12-27	17	5	0	19	0	25	7
B	1-11-0	2011-04-28	18	15	2	25	1	25	7
B	1-24-0	2010-05-31	25	20	1	28	2	30	13

Tabela 17 - Avaliação histórica no sistema C

Sistema	Versão	Data	GC EspL	GC EspN	GC Lit	LM Esp	LM Lit	SS Lit	SS Esp
C	2-29-0	2012-07-27	17	6	0	6	0	17	0
C	2-36-0	2013-02-27	17	12	1	5	0	15	0
C	2-37-0-2	2013-03-21	17	14	1	5	0	12	0

Tabela 18 - Avaliação histórica no sistema D

Sistema	Versão	Data	GC EspL	GC EspN	GC Lit	LM Esp	LM Lit	SS Lit	SS Esp
D	1-1	2011-03-20	10	5	0	5	0	13	1
D	1-9-9	2011-09-23	10	7	0	9	0	15	2
D	1-35-1	2013-02-18	9	8	1	13	1	25	6

Tabela 19 - Avaliação histórica no sistema E

Sistema	Versão	Data	GC EspL	GC EspN	GC Lit	LM Esp	LM Lit	SS Lit	SS Esp
E	2-25-0	2012-01-16	30	10	3	40	4	48	12
E	2-36-0	2012-08-23	43	25	5	66	6	53	19
E	2-44-0	2013-04-16	48	26	5	80	9	62	24

Tabela 20 - Avaliação histórica no sistema F

Sistema	Versão	Data	GC EspL	GC EspN	GC Lit	LM Esp	LM Lit	SS Lit	SS Esp
F	2-13-0	2012-01-12	24	8	2	26	0	44	9
F	2-24-0	2012-10-02	34	23	5	33	2	50	15
F	2-30-2	2013-04-15	42	25	7	47	4	60	18

4.5. Lições aprendidas

Essa seção apresenta algumas lições aprendidas a partir do estudo de caso proposto nesta dissertação.

PUC-Rio - Certificação Digital Nº 1112618/CA

Experiência dos especialistas do domínio. O sucesso do reúso das estratégias de detecção requer a participação de desenvolvedores com experiência em um conjunto de projetos de software do mesmo domínio. A experiência dos especialistas, tanto no domínio, quanto em boas práticas de desenvolvimento de software, foi muito importante para que eles entendessem o objetivo do estudo e participassem ativamente das decisões envolvidas. Mesmo assim, a tomada de algumas decisões eventualmente ficou atrasada, porque dessa forma criou-se uma dependência da opinião dos especialistas do domínio. Em alguns momentos, algumas sessões de investigação foram adiadas por conta da necessidade de uma versão emergencial do sistema, ou por outros problemas. Nesses casos, seria necessário ter à disposição um programador muito experiente para colaborar com o estudo, fosse para apoiar alguma decisão ou para conduzir uma sessão de investigação. Dessa forma, vemos como fundamental a participação de um especialista do domínio: (i) nas próximas execuções de estudos semelhantes, ou (ii) em projetos de software que definam estratégias de detecção para serem aplicadas de forma recorrente e com sucesso.

Protocolo das sessões de investigação. Nesse estudo, o protocolo executado em cada sessão de investigação foi baseado em (Guo et al., 2010). Mesmo que naquele trabalho fosse descrito que o poder de repetição do estudo era baixo, no estudo proposto por essa dissertação se mostrou adequado. Além disso, o estudo pôde gerar indícios do grau de reúso de algumas estratégias de detecção de anomalias. Nos próximos estudos o mesmo protocolo pode ser usado e adaptado, considerando as características do ambiente em que o estudo seja desenvolvido.

Estratégias de detecção conhecidas da literatura. Para o estudo proposto nessa dissertação, a maioria das estratégias usadas a partir da literatura encontrou muito poucas ocorrências de anomalias. Apesar de essas estratégias resultarem em 0% de falsos positivos, acabaram por não identificar várias ocorrências, identificadas por outras estratégias, que já haviam sido classificadas pelos especialistas como anomalias. Essa diferença nos resultados das estratégias pode ser vista no capítulo dos resultados (Capítulo 4). Nesse sentido, pode ser interessante afrouxar os limiares associados às métricas dessas estratégias, para que elas possam identificar mais anomalias e não esconder tantas ocorrências de

PUC-Rio - Certificação Digital Nº 1112618/CA

problemas. Logo, um futuro estudo deveria abordar de forma sistemática também uma avaliação de falsos negativos das estratégias.

Mapeamento de interesses. Nos próximos estudos, podemos propor uma análise anterior ao mapeamento de interesses, para desconsiderar os elementos que não representam necessariamente problemas no código. Dessa forma, podemos reduzir o número de elementos que serão avaliados pelas estratégias de detecção, enquanto mantemos a atenção voltada para um conjunto menor de elementos de código. Assim, pode-se diminuir o tempo em cada sessão de investigação, facilitando o trabalho, sem concorrer com outras atividades de manutenção.

4.6. Ameaças à Validade do Estudo Empírico

Nesta seção descrevemos algumas ameaças à validade desse estudo e o que foi feito para mitigar essas ameaças.

Escolha da empresa e dos sistemas que representam o domínio em estudo. A escolha do contexto em que o estudo foi aplicado está relacionada à ameaça de validade externa. Como em todo estudo empírico, os resultados apresentados nesse estudo são limitados ao domínio dos sistemas escolhidos. Além disso, devido às nossas questões de pesquisa (Capítulo 3), não poderíamos usar outros métodos empíricos (por exemplo, experimentos), em que generalizações são esperadas. De qualquer forma, para que fosse possível conduzir um estudo de caso em um ambiente real, considerando um domínio crítico, não poderíamos deixar de contar com o apoio de parceiros da indústria. Outra questão importante, da forma como esse estudo foi proposto, existia a necessidade de se avaliar uma quantidade considerável de sistemas do mesmo domínio, para um domínio real, o que acaba por restringir um pouco o nosso leque de alternativas. Dessa forma, para avaliar seis sistemas do mesmo domínio, considerando a proximidade estrutural desses sistemas, em um domínio crítico, a empresa escolhida para o estudo demonstrou disponibilidade imediata, tanto para acesso ao código fonte, quanto para a disponibilidade de apoio dos especialistas do domínio.

PUC-Rio - Certificação Digital Nº 1112618/CA

Ambiente homogêneo de desenvolvimento. Para amenizar as ameaças à validade interna desse estudo, sempre procuramos considerar um ambiente homogêneo de desenvolvimento. A partir da absorção dos seis sistemas, as equipes têm trabalhado constantemente em refatorações do código. Durante as refatorações, os especialistas puderam melhorar o seu conhecimento sobre o código-fonte, no que se refere às estruturas de código que representam sintomas de problemas. Da mesma forma, com as várias refatorações do código, era esperado que os programadores já participassem ativamente de práticas comuns como, por exemplo, o uso de ferramentas e *frameworks* semelhantes. Além disso, a proximidade estrutural entre os sistemas avaliados foi uma característica decisiva para esperar que, a partir das características de próprios sistemas desse domínio e o conhecimento dos especialistas do domínio sobre o código fonte, os problemas estruturais que evidenciam as ocorrências geradas a partir das estratégias de detecção de anomalias se repetissem entre os sistemas.

Classificação das ocorrências identificadas pelas estratégias de detecção. A classificação das ocorrências identificadas pelas estratégias de detecção de anomalias, entre falsos positivos e anomalias, foi restrita a cada líder da equipe de manutenção dos sistemas em questão, devido à disponibilidade dos especialistas. Dado que esse é um ambiente real, com prazos e problemas que podem surgir de forma inesperada, era do nosso conhecimento que os interesses desse estudo nem sempre seriam tratados como prioridade no ambiente de desenvolvimento. Para isso, foi preciso contar com a disponibilidade dos especialistas para a realização das sessões de investigação. Além disso, para classificar as ocorrências identificadas pelas estratégias de detecção de anomalias era preciso considerar a opinião de quem vive o dia a dia do desenvolvimento de sistemas do domínio em estudo.

Mesmo que um especialista independente pudesse fazer a avaliação das ocorrências identificadas pelas estratégias de detecção, não haveria o conhecimento do código fonte. Esse conhecimento é essencial para definir elementos do sistema que pertencem a uma parte estável do código e que, apesar de apresentarem sintomas de problemas, não são considerados anomalias.

PUC-Rio - Certificação Digital Nº 1112618/CA

5
Conclusões e Trabalhos Futuros

Detectar anomalias é difícil e custoso. Além disso, a eficiência de uma estratégia de detecção de anomalias está relacionada à baixa ocorrência de falsos positivos e à facilidade do seu reúso. Dessa forma, motivamos o estudo de estratégias de detecção com alto grau de reúso e baixa ocorrência de falsos positivos, para uma mesma família de sistemas (desde que possuam estruturas semelhantes).

Segundo o nosso estudo, em alguns casos, o reúso das estratégias de detecção pode ser melhorado se aplicados aos programas do mesmo domínio, sem gerar um efeito colateral. Mesmo assim, em outros casos, para realizar uma melhoria no reúso das estratégias, é possível que sejam criados falsos negativos. Nas subseções seguintes descrevemos como as contribuições dessa dissertação vão ao encontro do problema mencionado acima.

5.1.
Contribuições da Dissertação

O estudo conduzido por essa dissertação investigou o reúso de sete estratégias de detecção, relacionadas a três anomalias, em seis projetos de um domínio específico. O reúso das estratégias foi avaliado a partir do percentual de falsos positivos, classificados segundo a análise qualitativa de três especialistas do domínio, a partir das ocorrências encontradas pelas estratégias de detecção de anomalias. A partir do grau de reúso das estratégias, foram investigadas as situações em que fosse possível aumentar o grau de reúso, tendo em vista os sistemas escolhidos para o estudo. Nesse contexto, descrevemos abaixo as nossas contribuições.

Avaliação do reúso de estratégias de detecção em sistemas do mesmo domínio. De maneira geral, percebemos um reúso satisfatório, já que o reúso com sucesso das estratégias ocorreu em 81% dos casos. Mesmo assim, olhando para cada caso específico, verificamos que o reúso pode ser melhorado em alguns

PUC-Rio - Certificação Digital Nº 1112618/CA

casos, sem gerar um efeito colateral. Em outros casos, é impossível definir um limiar que elimine boa parte dos falsos positivos sem gerar novos falsos negativos, devido ao comportamento heterogêneo dos elementos de código. Como uma consequência direta, pode-se afirmar que existe um limite no grau de reúso das estratégias, isto é, uma nova adaptação na tentativa de diminuir o percentual de falsos positivos pode aumentar o número de falsos negativos. Essa contribuição está relacionada diretamente à nossa primeira questão de pesquisa (QP1)

Investigar estratégias de detecção orientadas a interesses. A presente dissertação inicia o estudo de estratégias de detecção tendo em vista conjuntos de elementos com características e responsabilidades semelhantes e bem definidas. Nesse estudo, percebeu-se que tanto em interesses como Autenticação/Segurança e Auxiliar, que são mais gerais, quanto os interesses Ações, Engine e Serviços, que são características mais específicas deste domínio, não há a necessidade de especialização das estratégias. Nesse sentido, ajustar os limiares para as estratégias considerando o mapeamento de interesses não seria benéfico para reduzir significativamente o percentual de falsos positivos nos casos acima. Por outro lado, o contrário pode ser dito para o caso dos interesses Persistência, Interface, Indicadores e Tarefas. Os interesses Persistência e Interface são mais gerais, e Indicadores e Tarefas são específicos do domínio em estudo. Esses resultados estão relacionados à nossa segunda questão de pesquisa (QP2).

PUC-Rio - Certificação Digital Nº 1112618/CA

5.2. Trabalhos Futuros

Os resultados obtidos, além das contribuições expostas acima, representam uma iniciativa na investigação de estratégias de detecção que possam ser usadas com alto grau de reúso. Nessa seção, descrevemos alguns trabalhos futuros que podem ampliar o estudo proposto aqui.

1. *Investigar a variação de complexidade das estratégias de detecção*

A partir dos resultados da fase se investigação de reúso, percebeu-se que duas estratégias de detecção de anomalias escolhidas a partir da literatura resultaram em 0% de falsos positivos em todos os casos em que encontraram ocorrências. Mesmo que essas duas estratégias resultassem em 100% de anomalias identificadas, essas estratégias não detectaram ocorrências que foram

identificadas pelas estratégias definidas inteiramente com o apoio dos especialistas e classificadas como anomalias. Essa evidência mostra que, apesar dessas estratégias de detecção de anomalias serem conhecidas da literatura e os seus limiares terem sido ajustados pelos especialistas, essas estratégias de detecção não colaboraram muito para a identificação de problemas de código nos sistemas definidos para representar o domínio em estudo. Além disso, a diferença entre as anomalias identificadas a partir das estratégias definidas completamente pelos especialistas e pelas estratégias a partir da literatura (com limiares definidos pelos especialistas) motiva uma investigação mais abrangente sobre a variedade da complexidade das estratégias de detecção de anomalias.

2. *Investigar o reúso de outras estratégias de detecção nesse mesmo domínio*

Nesse estudo, a partir das características do domínio em estudo e do apoio dos especialistas do domínio, investigamos uma anomalia em nível de classe, uma anomalia em nível de método e uma anomalia relacionada a mudanças. Considerando as 22 metáforas propostas por (Fowler, 1999), como trabalho futuro, podemos considerar a investigação do reúso de outras estratégias de detecção nesse mesmo domínio.

3. *Investigar o reúso de estratégias de detecção em outros domínios*

Como foi mencionado, a eficiência de uma estratégia de detecção de anomalias está relacionada à baixa ocorrência de falsos positivos e à facilidade do seu reúso. Dessa forma, motivamos o estudo de estratégias de detecção com alto grau de reúso e baixa ocorrência de falsos positivos, para famílias de sistemas de outros domínios. Assim, através do reúso eficiente de estratégias de detecção de anomalias, pode-se evitar a perda de qualidade do código em sistemas de outros domínios, além de evitar que o uso de estratégias de detecção de anomalias venha a ser uma tarefa negligenciada.

4. *Investigar a acurácia das estratégias de detecção a partir de sistemas de rastreamento de mudanças*

O presente estudo, sendo um estudo de caso, tem como expectativa gerar indícios de que é possível definir estratégias com alto grau de reúso. A partir desse trabalho, se espera desenvolver outros estudos em que seja possível avaliar as ocorrências identificadas pelas estratégias de detecção de anomalias, entre

PUC-Rio - Certificação Digital Nº 1112618/CA

anomalias e falsos positivos, a partir de outras fontes, como por exemplo, a partir de sistemas de rastreamento de mudanças.

PUC-Rio - Certificação Digital Nº 1112618/CA

6
Referências bibliográficas

BASILI, V.; CALDIERA, G.; ROMBACH, H. **The Goal Question Metric approach.** The Encyclopedia of Software Engineering, Vol. 2, pp. 528-532, John Wiley & Sons, 1994.

BASILI, V.; ROMBACH, H. **The TAME project:** Towards improvement-oriented software environments. IEEE Transactions on Software Engineering, Vol. 14, No.6, pp. 758-773, 1988.

BASILI, R.; WEISS, D. **A Methodology for Collecting Valid Software Engineering Data.** IEEE Transactions on Software Engineering, Vol. 10, No.6, pp. 728-738, 1984.

BIEMAN, J.; KANG, B. **Cohesion and reuse in an object-oriented system.** Proceedings of ACM SIGSOFT Software Engineering Notes. Vol. 20. No. SI. ACM, 1995.

CHIDAMBER, S. R.; KEMERER, C. F. **A metrics suite for object oriented design**. IEEE Transactions on Software Engineering, V. 20, No. 6, p. 476-493, 1994.

DANPHITSANUPHAN, P.; SUWANTADA, T. **Code Smell Detecting Tool and Code Smell-Structure Bug Relationship**. Spring Congress on Engineering and Technology (S-CET), IEEE, 2012. p. 1-5.

DELIGIANNIS, I. et al. **A controlled experiment investigation of an object-oriented design heuristic for maintainability**. Journal of Systems and Software 72.2, pp. 129-143, 2004.

DELIGIANNIS, I. et al. **An empirical investigation of an object-oriented design heuristic for maintainability.** Journal of Systems and Software 65.2, pp. 127-139, 2003.

EMDEN, E.; MOONEN, L. **Java quality assurance by detecting code smells**. Proceedings of the 9th Working Conference on Reverse Engineering. IEEE, pp. 97–107, 2002.

FENTON, N.; PFLEEGER, S. **Software metrics:** a rigorous and practical approach. PWS Publishing Co., 1998.

FENTON, N.; NEIL, M. **Software metrics:** roadmap. In Proceedings of the Conference on the Future of Software Engineering. ACM, pp. 357-370, 2000.

FERREIRA, K. A. et al. **Identifying thresholds for object-oriented software metrics.** Journal of Systems and Software, Vol. 85, No. 2, pp. 244-257, 2012.

FONTANA, F.; BRAIONE, P.; ZANONI, M. **Automatic detection of bad smells in code:** An experimental assessment. Journal of Object Technology, Vol. 11, No. 2, pp. 5: 1-38, 2012.

PUC-Rio - Certificação Digital Nº 1112618/CA

FONTANA, F.; FERME, V.; SPINELLI, S. **Investigating the impact of code smells debt on quality code evaluation**. Third International Workshop on Managing Technical Debt, IEEE, pp. 15-22, 2012.

FONTANA, F. et al. **An Experience Report on Using Code Smells Detection Tools**. Fourth International Conference on Software Testing, Verification and Validation Workshops (ICSTW), IEEE, pp. 450-457, 2011.

FOWLER, Martin. **Refactoring: Improving the Design of Existing Code**. New Jersey: Addison Wesley, 1999. 464 p.

GUO, Y. et al. **Domain-specific tailoring of code smells:** an empirical study. Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 2, 2010.

HARRISON, R.; COUNSELL, S.; NITHI, R. **Coupling metrics for object-oriented design**. Proceedings of the fifth International Software Metrics Symposium, pp. 150-157, 1998.

HENRY, S.; KAFURA, D. **Software structure metrics based on information flow**. IEEE Transactions on Software Engineering, n. 5, p. 510-518, 1981.

iPlasma: http://loose.upt.ro/reengineering/research/iplasma

InFusion: http://www.intooitus.com/inFusion.html

LANZA, M.; MARINESCU, R. **Object-Oriented Metrics in Practice**. Springer, 2006. 206 p.

LEHMAN, M. **Programs, life cycles, and laws of software evolution**. Proceedings of the IEEE, v. 68, n. 9, pp. 1060-1076, 1980.

LI, W.; SHATNAWI, R. **An empirical study of the bad smells and class error probability in the post-release object-oriented system evolution**. Journal of Systems and Software v.80, n 7, pp. 1120-1128, 2007.

LORENZ, M.; KIDD, J. **Object-oriented software metrics:** a practical guide. Prentice-Hall, Inc., 1994.

McCABE, T.J. **A Complexity Measure**. IEEE Transactions on Software Engineering, 2(4):308–320, December 1976.

MACÍA, I. et al. **On the Relevance of Code Anomalies for Identifying Architecture Degradation Symptoms**. Proceedings of the 16th European Conference on Software Maintenance and Reengineering (CSMR'12), Szeged, Hungary, March 2012.

MACÍA, I. et al. **Are Automatically-Detected Code Anomalies Relevant to Architectural Modularity? An Exploratory Analysis of Evolving Systems**. Proceedings of the 11th International Conference on Aspect-Oriented Software Development (AOSD'12), 2012.

MACÍA, I. ; GARCIA, A. ; STAA, A. V. **An Exploratory Study of Code Smells in Evolving Aspect-Oriented Systems**. Proceedings of the 10th International Conference on Aspect-Oriented Software Development (AOSD), 2011.

MOHA, N. et al. **From a domain analysis to the specification and detection of code and design smells**, Formal Aspects of Computing, v. 22, n 3-4, pp. 345-361, 2010.

PUC-Rio - Certificação Digital Nº 1112618/CA

MARINESCU, R. **Detecting design flaws via metrics in object-oriented systems**. 39th IEEE International Conference and Exhibition on Technology of Object-Oriented Languages and Systems, pp. 173-182, 2001.

MARINESCU, R. **Measurement and Quality in Object-Oriented Design**. Proceedings of the 21st IEEE International Conference on Software Maintenance, 2005.

MARINESCU, R. **Detection strategies:** Metrics-based rules for detecting design flaws. Proceedings of the 20th IEEE International Conference on Software Maintenance, pp. 350–359, 2004.

MARINESCU, C. et al. **iPlasma:** An integrated platform for quality assessment of object-oriented design. Proceedings of the 21st IEEE International Conference on Software Maintenance, 2005.

MURPHY-HILL, E; BLACK, A. **An interactive ambient visualization for code smells**. Proceedings of the 5th International Symposium on Software Visualization, pp. 5-14, 2010.

OLBRICH, S. et al. **The evolution and impact of code smells:** A case study of two open source systems. Proceedings of the 3rd International Symposium on Empirical Software Engineering and Measurement, pp. 390-400, 2009.

PARNAS, D. **On the criteria to be used in decomposing systems into modules**. Communications of the ACM, v. 15, n. 12, pp. 1053-1058, 1972.

PMD. http://pmd.sourceforge.net/

POSHYVANYK, D.; MARCUS, A. **The conceptual coupling metrics for object-oriented systems**. Proceedings of the 22nd IEEE International Conference on Software Maintenance, pp. 469-478, 2006.

RAPU, D. et al. **Using history information to improve design flaws detection**. Proceedings of the 8th European Conference on Software Maintenance and Reengineering, pp. 223-232, 2004.

RIEL, A. **Object-oriented design heuristics**. Addison-Wesley Longman Publishing Co., Inc., 1996.

ROBILLARD, M; MURPHY, G. **Representing Concerns in Source Code**. ACM Transactions on Software Engineering and Methodology, v. 16, n. 1, pp. 3, 2007.

ROBILLARD, M., WEIGAND-WARR, F. **ConcernMapper:** simple view-based separation of scattered concerns. Proceedings of the 2005 OOPSLA workshop on Eclipse technology eXchange, 2005.

SCHUMACHER, J. et al. **Building empirical support for automated code smell detection**. Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, pp. 8, 2010.

SUTTON JR, S. M; ROUVELLOU, I. **Concern modeling for aspect-oriented software development**. Aspect-Oriented Software Development, v. 1, pp. 479-505, 2004.

TARR, P. et al. **N degrees of separation: multi-dimensional separation of concerns**. Proceedings of the 21st ACM International Conference on Software Engineering, pp. 107-119, 1999.

PUC-Rio - Certificação Digital Nº 1112618/CA

TSANTALIS, N.; CHAIKALIS, T.; CHATZIGEORGIOU, A. **JDeodorant:** Identification and removal of type checking bad smells. Proceedings of the 12th IEEE Conference on Software Maintenance and Reengineering, pp 329–331, 2008.

WOHLIN, C.; RUNESON, P.; HÖST, M. **Experimentation in software engineering: an introduction**. Boston: Springer, 1999. 228 p.

ZHANG, M.; HALL, T.; BADDOO, N. **Code bad smells:** a review of current knowledge. Journal of Software Maintenance and Evolution: Research and Practice, v. 23, n. 3, pp. 179-202, 2011.

PUC-Rio - Certificação Digital Nº 1112618/CA