Do *Code Smell* ao Código Limpo: Uma Experiência Prática no Ensino de Refatoração para Manutenção de Software

João Teixeira do Nascimento, Caio Rian R. de Sousa, Lara Gabrielly Lima, Edivar C. Carvalho Filho, Guilherme P. Borges, Jacilane H. Rabelo, Carla Ilane Moreira Bezerra

[1]Universidade Federal do Ceará (UFC)

{joao.nascimento, caio.rianbr, edivarcruz, guilhermepereira}@alu.ufc.br

laragabrielly@alu.ufc.br, jacilane.rabelo@ufc.br, carlailane@ufc.br

Abstract. *Teaching future Computer Science professionals to write high-quality code is essential. This work presents an experience report on teaching code smells and code refactoring in a software maintenance class with 42 students. Four aspects were analyzed: (i) students' perception of refactoring quality, (ii) challenges faced in identifying and correcting code smells, (iii) skills developed and benefits gained through practice, and (iv) the most commonly used techniques in code smells refactoring. The results highlight the benefits of teaching code smells refactoring and propose a practical methodology for this instruction.*

Resumo. *Ensinar futuros profissionais de Computação a escrever código de alta qualidade é essencial. Este trabalho apresenta um relato de experiência sobre o ensino de code smells e refatoração de código em uma turma de manutenção de software composta por 42 estudantes. Foram analisados quatro aspectos: (i) a percepção dos alunos sobre a qualidade das refatorações, (ii) os desafios enfrentados na identificação e correção de code smells, (iii) as habilidades desenvolvidas e os benefícios adquiridos com a prática, e (iv) as técnicas mais utilizadas na refatoração de code smells. Os resultados destacam os benefícios de ensinar refatoração de code smells e propõem uma metodologia prática para esse ensino.*

1. Introdução

Code smells são sintomas de código de baixa qualidade que indicam possíveis problemas na arquitetura de um software [Fowler 2018]. O termo foi popularizado por [Fowler 2018], que propuseram 22 *code smells*, posteriormente ampliados por outros autores como [Mäntylä e Lassenius 2006] e [Wake 2004], cujas taxonomias fundamentam ferramentas atuais de detecção [Kaur 2020]. Embora *code smells* não afetem diretamente o funcionamento do sistema, dificultam futuras manutenções.

[Fowler 2018] e [Wake 2004] também propuseram soluções de refatoração para eliminar esses problemas, educando desenvolvedores a adotar melhores práticas. Tendo em vista que um mesmo *code smell* pode ser solucionado de diferentes formas, portanto entender as técnicas de refatoração mais adequadas é essencial [Fowler 2018].

Produzir um código de qualidade é importante para fases posteriores ao desenvolvimento do software, como a etapa de manutenção que muitas vezes despende o maior

valor agregado do orçamento do sistema, chegando a 80% do valor total do sistema. Tendo em vista isso, a importância de se produzir códigos de qualidade que reduzam os gastos de manutenção do software, tornam-se habilidades prioritárias para os profissionais que queiram se destacar na indústria de desenvolvimento de sistemas [Al Dallal 2013].

O ensino de refatoração de *code smells* mostrou diversos benefícios pedagógicos ao desenvolver habilidades de melhoria de código-fonte, pensamento crítico para lidar com indicações de más práticas de programação no código e treinar a revisão de código existente [AlOmar et al. 2023, Bezerra et al. 2024]. Cada vez mais, a indústria de desenvolvimento exige habilidades bem desenvolvidas de produção de código de qualidade [Chren et al. 2022].

Este trabalho apresenta um relato de experiência sobre o ensino de *code smells* e refatoração de código em projetos Java conduzido em uma turma de manutenção de *software* com 42 estudantes. Buscamos entender e documentar as dificuldades associadas a condução da atividade, qualificamos as refatorações praticadas pelos estudantes do estudo, e identificamos as percepções e habilidades adquiridas pelos estudantes com a prática de refatoração dos *code smells*.

2. Trabalhos Relacionados

[Tan e Poskitt 2024] apresentam uma abordagem diferente para o ensino da refatoração. Inicialmente, os estudantes realizam um exercício de programação no qual, sem perceber, produzem um código funcional, mas com *code smells*. Depois, aprendem a identificá-los e refatorá-los. O estudo, conduzido com 35 universitários iniciantes, comparou essa abordagem com um método tradicional, no qual refatoravam um código desconhecido. Os resultados mostram que a familiarização levou a maior taxa de identificação de *code smells* e sucesso na refatoração, indicando uma aplicação mais eficaz dos conceitos.

[Bezerra et al. 2024] apresentam as percepções e os obstáculos enfrentados por estudantes de graduação em relação ao ensino da qualidade do código por meio da refatoração de *code smells*. Os resultados mostram algumas vantagens, como a melhoria na resolução de problemas e nas habilidades interpessoais, assim como diversas dificuldades, entre elas o fato de que a refatoração de código pode resultar em novos *code smells*, que exigem refatorações adicionais. Ademais, também perceberam que a maior dificuldade dos estudantes foi a compreensão do código-fonte a ser refatorado.

[Rabelo et al. 2018] destacam a importância da refatoração na prática para garantir um código bem estruturado e de qualidade. Isso pode ser feito por meio de ferramentas ou da técnica de revisão de código. Nesse contexto, [Chaaban et al. 2023] apresentam um relato de experiência sobre um curso prático de refatoração de código oferecido a universitários do interior do Ceará. O curso foi ministrado para 20 alunos iniciantes em Computação. Os resultados indicam que os estudantes gostaram do curso com atividade prática e que mais de 80% compreenderam os conceitos de refatoração e se sentiram aptos a aplicá-los na prática.

Similar aos estudos apontados, essa pesquisa apresenta um relato de experiência do ensino de refatoração de *code smells*. No entanto, utiizamos uma maior quantidade de estudantes na prática de refatoração de 6 tipos de *code smells* em 23 projetos Java. Outro diferencial deste trabalho é analisar a percepção dos estudantes e habilidades adquiridas.

3. Metodologia

3.1. Objetivo e Questões de Pesquisa

Este artigo apresenta um relato de experiência sobre o ensino da refatoração de *code smells*, realizado com alunos da disciplina de Manutenção de Software, ofertada como optativa no 7º semestre dos cursos de Bacharelado em Engenharia de Software e Ciência da Computação. Nós investigamos a qualidade das refatoração, as principais habilidades adquiridas pelos estudantes durante a refatoração dos *code smells*, bem como a dificuldade de refatoração dos *code smells* pelos alunos. As questões de pesquisa (QP_s) são apresentadas a seguir.

- **QP_1: Qual a qualidade das refatorações percebida pelos estudantes?**
- **QP_2: Quais as principais habilidades adquiridas pelos estudantes durante a refatoração dos *code smells*?**
- **QP_3: Quais as percepções dos estudantes em relação aos desafios enfrentados na prática de refatoração de *code smells*?**

3.2. Etapas do estudo

A Figura 1 exibe as etapas necessárias para a realização do estudo.

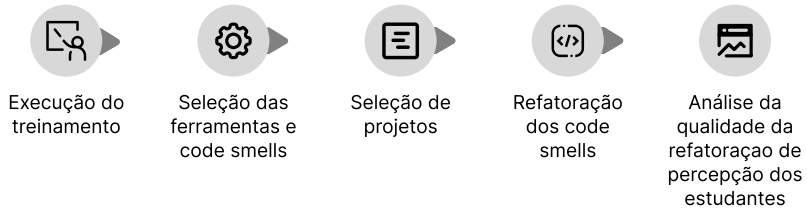

Figura 1. Etapas do estudo

Etapa 1: Execução do treinamento. A prática de refatoração dos *code smells* foi conduzida com uma turma da disciplina de Manutenção de Software da Universidade Federal do Ceará – Campus Quixadá, composta por 42 alunos. A Tabela 1 apresenta o perfil da turma e sua experiência com o tema. O conteúdo do treinamento foi planejado conforme esse perfil.

Na primeira aula, com duração de duas horas, apresentamos os principais *code smells* e técnicas de refatoração, com exemplos práticos em Java. Uma semana depois, realizamos uma segunda aula, de uma hora e dez minutos, voltada para instruções de download, instalação e uso das ferramentas PMD[1] e Designite[2].

Para a prática, os alunos receberam um sistema com diversas ocorrências de *code smells*. Também elaboramos um material de apoio explicativo, abordando os conceitos e tipos mais relevantes de *code smells*, além das técnicas mais utilizadas para refatoração. Complementarmente, preparamos documentos de suporte, como instruções ilustradas de instalação e uso das ferramentas, descrição geral do trabalho e um vídeo demonstrando o processo de entrega das refatorações via GitHub, para facilitar a análise manual posterior.

[1]https://pmd.github.io/
[2]https://www.designite-tools.com/

Etapa 2: Seleção das ferramentas e *code smells*. Inicialmente, selecionamos as ferramentas de apoio à execução das atividades exercidas no estudo. Existem várias ferramentas de deteção de *code smells* [Pereira dos Reis et al. 2022]. Para seleção utilizamos os critérios: (i) facilidade de instalação e uso; (ii) utilização ampla na comunidade acadêmica e indústria; (iii) detecção de uma maior quantidade de *code smells*. Após aplicado esses critérios selecionamos a ferramenta PMD e Designite. Os *code smells* selecionados a serem refatorados pelos estudantes foram escolhidos de acordo com a detecção pelas ferramentas selecionadas e também ser um dos *code smells* classificados por [Fowler 2018]. Os *code smells* selecionados foram: *god class*, *data class*, *complex conditional*, *long method*, *long parameter list* e *magic number*.

Tabela 1. Perfil dos estudantes

ID	Desenvolvimento	Java	*Code Smells*	Refatoração
P1	2 anos	Nenhum	Intermediário	Básico
P3	3 anos	Avançado	Nenhum	Intermediário
P4	2 anos	Básico	Nenhum	Intermediário
P5	1 ano	Mínimo	Intermediário	Básico
P6	3 anos	Avançado	Avançado	Especialista
P7	2 anos	Intermediário	Básico	Mínimo
P8	$>$ 1 ano	Mínimo	Nenhum	Mínimo
P9	3 anos	Básico	Nenhum	Mínimo
P10	2 anos	Intermediário	Básico	Básico
P11	2 anos	Básico	Nenhum	Mínimo
P12	$>$ 1 ano	Mínimo	Intermediário	Básico
P13	4 anos	Mínimo	Básico	Básico
P14	3 anos	Básico	Básico	Básico
P15	2 anos	Nenhum	Básico	Básico
P16	3 anos	Avançado	Mínimo	Intermediário
P17	1 ano	Mínimo	Nenhum	Mínimo
P18	4 anos	Intermediário	Intermediário	Intermediário
P19	1 ano	Mínimo	Básico	Intermediário
P20	3 anos	Básico	Nenhum	Básico
P21	$>$ 1 ano	Nenhum	Básico	Intermediário
P22	1 ano	Nenhum	Básico	Avançado
P24	1 ano	Básico	Básico	Básico
P25	$>$ 1 ano	Mínimo	Básico	Básico
P26	2 anos	Intermediário	Intermediário	Avançado
P27	2 anos	Intermediário	Especialista	Avançado
P29	3 anos	Avançado	Avançado	Avançado
P30	2 anos	Mínimo	Básico	Intermediário
P31	2 anos	Nenhum	Mínimo	Mínimo
P32	2 anos	Mínimo	Nenhum	Básico
P33	2 anos	Nenhum	Nenhum	Mínimo
P34	2 anos	Nenhum	Básico	Intermediário
P35	2 anos	Básico	Nenhum	Mínimo
P36	3 anos	Intermediário	Básico	Básico
P37	1 ano	Nenhum	Mínimo	Mínimo
P38	3 anos	Mínimo	Intermediário	Básico
P39	4 anos	Intermediário	Mínimo	Básico
P40	5 anos	Básico	Mínimo	Intermediário
P41	2 anos	Mínimo	Básico	Básico
P42	$>$ 1 ano	Básico	Básico	Mínimo

Etapa 3: Seleção dos projetos. Selecionamos 89 projetos Java de fácil entendimento para os estudantes. Filtramos 36 projetos que tiveram o mínimo de 4 *code smells* diferentes e que são *open-source*. Ao final, as duplas de estudantes selecionaram 23 projetos conforme a Tabela 2.

Etapa 4: Refatoração dos *code smells*. Na prática, os estudantes deveriam refatorar manualmente 20 ocorrências de 6 tipos de *code smells*, identificados nos proje-

Tabela 2. Principais características dos sistemas

Sistema	estudantes	Linhas de código
S1	P1, P2	214685
S2	P3, P4	6765
S3	P5, P6	2595
S4	P7, P8	11587
S5	P9	4782
S6	P10, P11	4698
S7	P12, P13	2175
S8	P14, P15	12309
S9	P16, P17	11290
S10	P18, P19	2464
S11	P20	4220
S12	P21, P22	8430
S13	P23, P24	3258
S14	P25, P26	1763
S15	P27, P28	7269
S16	P29	8250
S17	P30, P31	6801
S18	P32, P33	2900
S19	P34, P35	5128
S20	P36, P37	6126
S21	P38, P39	2069
S22	P40	4446
S23	P41, P42	6759

tos selecionados. A atividade poderia ser realizada individualmente ou em duplas, sem repetição de sistemas entre os grupos. A execução ocorreu ao longo de duas semanas, com entrega dos resultados via repositórios no *Github*. As refatorações deveriam seguir a orientação de criar uma nova *branch* para cada ocorrência, realizando um *commit* após a sua remoção. Para apoio, foi disponibilizado um canal de dúvidas no Discord[3]. Recomendamos o uso das ferramentas PMD e Designite em dois momentos: (i) para selecionar os *code smells* a serem refatorados; e (ii) ao final de cada refatoração, para verificar se a ocorrência foi de fato eliminada. Os estudantes foram orientados a aplicar a técnica de refatoração mais adequada a cada situação, com base nas recomendações mais comuns para o tipo de *code smell* identificado. Como material de consulta, foram indicados os sites[4][5], que apresentam diversos exemplos práticos de técnicas de refatoração reconhecidas na literatura. Ao final da atividade, os estudantes deveriam entregar uma apresentação em vídeo e um relatório contendo o *link* para o repositório com as *branchs* criadas. No relatório, precisavam descrever as ocorrências detectadas no início e ao término da refatoração de cada tipo de *code smell* escolhido (por exemplo, após a refatoração completa de todas as instâncias de *data class*).

Análise da qualidade das refatorações e percepção dos estudantes. Nesse passo, foram analisadas as modificações feitas na refatoração dos *code smells*, com base no histórico de *commits* das *branchs* criadas pelos estudantes. Embora eles tenham descrito nos próprios *commits* as técnicas aplicadas, realizamos uma verificação manual do código para validar a adequação das refatorações. A análise considerou aspectos como: (i) se a refatoração eliminou o *code smell*; (ii) se foi apropriada ao contexto; e (iii) se seguiu as recomendações da documentação.

[3]https://discord.com/

[4]`https://refactoring.guru/`

[5]`https://luzkan.github.io/smells/`

Além disso, os estudantes responderam a um questionário sobre a prática, abordando tipos de *code smells* mais difíceis, técnicas mais utilizadas, dificuldades encontradas e habilidades desenvolvidas. As respostas abertas foram analisadas por codificação aberta e axial [Bryant e Charmaz 2010], destacando os principais desafios e benefícios percebidos pelos estudantes.

Todos os dados da prática estão disponíveis no Zenodo[6].

4. Resultados

4.1. Qualidade das refatorações percebida pelos estudantes (QP_1)

Respondemos à QP_1 analisando a qualidade das refatorações com base em três critérios: (i) o *code smell* deixou de ser detectado pela ferramenta após a refatoração; (ii) a refatoração seguiu uma lógica de tratamento aceitável para o problema identificado; e (iii) o tratamento aplicado estava entre os recomendados nos materiais de apoio. Esses aspectos estão representados na Tabela 3 pelas colunas *Removido*, *Satisfatório* e *Tratamento Conhecido*, respectivamente.

Cada critério possui sua especificidade: uma refatoração pode ter seguido a documentação sem remover o *code smell*, ou pode ter removido o *code smell* sem utilizar uma abordagem adequada. Por exemplo, a simples exclusão de um método pode fazer com que o problema deixe de ser detectado pela ferramenta, mas isso não configura, necessariamente, uma refatoração satisfatória ou adequada.

Tabela 3. Qualidade das refatorações dos *code smells* de acordo com a percepção dos estudantes

Code smells	Removido	Satisfatório	Tratamento Conhecido	Refatorações Totais
Magic number	82	86	80	111
Data class	8	13	15	50
Complex conditional	25	22	18	34
Long parameter list	17	17	16	32
Long method	14	16	18	26
God class	5	5	6	12
Total	**151**	**159**	**153**	**265**

Analisando a Tabela 3, observa-se uma proximidade entre as quantidades de refatorações consideradas corretas segundo os três critérios. Isso sugere que refatorações bem avaliadas em um dos critérios tendem também a serem bem avaliadas nos demais, o que reforça sua qualidade geral. A principal exceção é o *code smell Data class*, que apresentou menor consistência entre os critérios.

Outro ponto importante é que, para a maioria dos tipos de *code smells*, ao menos metade das refatorações foram avaliadas como satisfatórias em algum dos critérios, o que evidencia a relevância das modificações realizadas no tratamento dos problemas detectados.

A Tabela 4 apresenta a mesma avaliação anterior, mas agora distribuída por sistemas desenvolvidos. As colunas seguem os mesmos critérios anteriores: remoção do *code smell*, qualidade satisfatória da refatoração, uso de tratamento conhecido e total de refatorações realizadas.

[6] https://zenodo.org/records/15078706

Tabela 4. Qualidade das refatorações na percepção dos estudantes por sistema

Sistema	Removido	Satisfatório	Tratamento Conhecido	Refatorações totais
S1	4	0	0	20
S3	12	13	17	20
S4	13	14	12	14
S5	0	0	0	15
S7	7	12	11	15
S8	7	5	5	20
S9	1	4	3	9
S10	18	18	17	19
S11	3	8	6	25
S12	14	9	15	16
S13	15	16	15	16
S18	18	22	22	23
S19	7	7	7	7
S20	9	9	5	10
S21	7	2	1	14
S23	16	20	17	22

A Tabela 4 permite observar a qualidade das refatorações por sistema, evidenciando aqueles que mais se destacaram em termos de boas práticas aplicadas. Os sistemas **S3, S4, S10, S12, S13, S18, S20 e S23** apresentaram as maiores quantidades de refatorações bem executadas, enquanto os sistemas **S1 e S5** tiveram desempenho inferior, com baixos índices de qualidade nas refatorações realizadas.

4.2. Habilidades adquiridas e benefícios da prática de refatoração de *code smells* percebidas pelos estudantes ($\mathbf{QP_2}$)

Na QP_2 identificamos as habilidades adquiridas indicadas pelos estudantes após concluírem a prática de refatoração de *code smells*. As Habilidades foram classificadas de acordo com [Matturro et al. 2019], que identificaram habilidades exigidas para um engenheiro de software. A Tabela 5 apresenta as habilidades adquiridas pelos estudantes durante a prática, permitimos que os estudantes selecionassem várias opções caso fosse necessário. As habilidades mais mencionadas pelos estudantes foram: capacidade de análise e resolução de problemas. Além de também trabalho em equipe, pensamento crítico e gerência de mudanças. Essas habilidades são inerentes da prática de programação. Acreditamos que a prática auxilie a lidar com códigos que não foram escritos pelos próprios estudantes e de saber lidar com a resolução desses problemas.

Também identificamos com os estudantes, possíveis benefícios da prática de refatoração de *code smells*. A Tabela 6 mostra os benefícios que os estudantes sugeriram terem adquirido após a prática e filtramos as respostas que eram similares. O benefício mais citado foi legibilidade do código, de fato melhorando a manutenção do software. Outros benefícios citados também indicam a melhoria do código para futuras manutenções.

4.3. Percepções dos estudantes em relação aos desafios enfrentados na prática de refatoração de *code smells* ($\mathbf{QP_3}$)

Para responder a QP_3, utilizamos um questionário de entendimento das percepções dos estudantes, no qual perguntamos sobre os elementos do estudo que tiveram mais destaques e/ou precisaram de mais atenção em relação aos tópicos de *code smells*, ferramentas, técnicas de refatoração e uma visão geral da prática.

Cobrimos as percepções relacionadas aos *code smells* estudados inicialmente perguntando sobre os *code smells* mais difíceis de refatorar. Na Tabela 7, podemos destacar

Tabela 5. Habilidades adquiridas pelos estudantes após a prática de refatoração dos *code smells*

Habilidade Adquirida	Participantes	Total
Capacidade de análise	P1, P2, P3, P4, P5, P6, P7, P8, P9, P10, P11, P12, P13, P14, P15, P18, P19, P20, P22, P24, P26, P27, P28, P29, P31, P32, P33, P34, P35, P36, P38, P39, P40, P41, P42	35
Resolução de problemas	P1, P2, P3, P4, P6, P7, P9, P11, P12, P13, P15 P18, P19, P22, P24, P27, P29, P32, P33, P34, P35, P36, P38, P39, P40, P41, P42	27
Trabalho em equipe	P1, P2, P4, P5, P10, P12, P13, P15, P17, P18 P19, P21, P24, P26, P32, P34, P35, P36, P39, P42	20
Pensamento crítico	P2, P10, P11, P12, P15, P22, P24, P26, P27, P28, P29, P34, P35, P36, P38, P42	16
Gerência de mudanças	P3, P5, P9, P13, P15, P16, P18, P20, P28 P32, P34, P35, P38, P39	14
Capacidade metódica	P9, P14, P15, P17, P19, P20, P26, P28, P29 P30, P34, P36	12
Criatividade	P1, P6, P11, P12, P14, P15, P17, P24, P34, P35, P36	11
Gerência de decisão	P1, P3, P9, P15, P24, P26, P34, P41	8
Orientação a resultados	P3, P6, P14, P15, P18, P19, P32, P34	8

Tabela 6. Benefícios de fazer parte da prática segundo os estudantes

Benefício	Participantes	Total
Legibilidade	P2, P3, P8, P9, P12, P22, P26, P28, P29, P31, P35, P36, P38, P40	14
Melhorar o código	P2, P14, P18, P21, P27	5
Manter código organizado	P5, P10, P17, P19, P29	5
Código limpo	P4, P19, P32, P36	4
Melhor entendimento do código	P9, P17, P33, P39	4
Melhorar a habilidade de refatoração	P1, P11, P20, P24	4
Facilidade de manutenção	P10, P12, P19, P21	4
Simplificação do código	P15, P18, P26	3
Capacidade de Análise	P6, P11	2
Maior compreensão	P8, P10	2
Reuso de código	P12, P38	2
Facilidade de encontrar problemas	P9, P13	2
Outros Benefícios	P1, P7, P9, P16, P21, P24, P28, P29, P30, P34, P40, P41, P42	13

que *long method* foi o *code smell* mais difícil de refatorar (19 citações). Outros *code smells* tiveram uma quantidade de citações próximas, com excessão do *magic number* que apesar de ser o *code smell* mais refatorado na prática, de acordo com a Tabela 7, ele foi indicado por somente um estudante. Vale ressaltar que os estudantes poderiam marcar mais de um *code smell* do estudo como sendo difícil de refatorar.

Lidamos com as percepções relacionadas as técnicas de refatoração dos *code smells* da prática requisitando a visão dos estudantes sobre qual foi a técnica de refatoração mais difícil de implementar durante as refatorações. A Tabela 8 indica que a técnica mais difícil de empregar nas refatorações foi o *extract method*, sendo mencionado por 11 estudantes, seguida por *extract class* e *introduce parameter object* que têm 9 e 5 menções. Permitimos que os estudantes registrassem qualquer nome de técnica em um campo de resposta aberta.

Abrangemos as percepções dos estudantes relacionadas às ferramentas de detecção de *code smells* indagando sobre a possibilidade de ter ocorrido alguma dificuldade durante o uso das ferramentas. Ao todo 11 estudantes confirmaram que enfrentaram algum tipo de problema nas ferramentas ao usarem as ferramentas de detecção de *code*

Tabela 7. *Code smells* mais difíceis de refatorar segundo os estudantes

Code Smell	Participantes	Total
Long method	P7, P8, P9, P11, P13, P16, P18, P19, P21, P24, P26, P29, P30, P34, P35, P36, P38, P41, P42	19
God class	P4, P7, P8, P9, P10, P12, P13, P14, P27, P28, P29, P31, P32, P36, P38, P42	16
Long parameter list	P6, P14, P15, P16, P17, P18, P19, P32, P33, P35, P40	11
Data class	P1, P2, P3, P11, P17, P20, P22, P28, P32, P39	10
Complex conditional	P1, P5, P22, P27, P30, P31, P42	7
Magic number	P15	1

Tabela 8. Técnica de refatoração mais difícil de aplicar segundo os estudantes

Técnica de Refatoração	Participantes	Total
Extract method	P3, P7, P8, P11, P19, P20, P22, P26, P39, P41, P42	11
Extract class	P1, P9, P10, P11, P12, P17, P24, P35, P36	9
Introduce parameter object	P13, P27, P32, P33, P34	5
Move method	P16, P21, P36	3
Replace magic number	P18, P28, P35	3
Decompose conditional	P30, P31, P40	3
Encapsulate field	P2	1
Self encapsulate field	P4	1
Preserve whole object	P6	1
Replace method object	P14	1
Replace conditional polymorphism	P38	1

smells. Enquanto, que 28 estudantes afirmaram não terem enfrentado nenhuma adversidade no uso de qualquer uma das ferramentas de detecção de *code smells*. Por fim, tratamos as percepções relacionadas a prática de refatoração de *code smells*. Para isso, perguntamos aos estudantes quais dificuldades eles enfrentaram durante a prática, conforme a Tabela 9.

A Tabela 9 revela que **entender do código** é a dificuldade mais mencionada pelos estudantes durante a execução da prática (10 citações). Solicitamos que os estudantes detalhassem suas dificuldades em uma questão aberta, onde separamos os assuntos descritos em tópicos e aglutinamos as respostas que fazem parte do mesmo assunto para formularmos uma análise quantitativa dessas respostas. Excluímos da análise as dificuldades que foram mencionadas por somente um participante, esses assuntos foram indicados em **problemas pouco mencionados**.

Tabela 9. Dificuldades enfrentadas pelos estudantes durante a prática

Dificuldade	Participantes	Total
Entender o código	P7, P8, P9, P10, P11, P26, P27, P28, P41, P42	10
Encontrar *code smells*	P1, P17, P30, P31, P32, P42	6
Entender como refatorar	P3, P13, P15, P16, P18, P33	6
Usar Java	P4, P12, P14, P19, P22, P34	6
Selecionar refatoração ideal	P1, P6, P7, P8, P21, P24	6
Usar ferramentas de detecção	P16, P20, P36, P38	4
Manter funcionalidades	P5, P29, P39	3
Problemas pouco mencionados	P2*, P31, P35*, P36, P40, P42	6

5. Lições Aprendidas

A aplicação da prática nos permitiu identificar pontos de melhoria importantes para futuras edições. Percebemos a necessidade de uma ambientação prévia nos sistemas utilizados, já que muitos estudantes relataram dificuldade em compreender o código, o que impactou a execução das refatorações. Também observamos que a variação na complexidade dos sistemas pode ter influenciado a qualidade das entregas, reforçando a importância de um balanceamento mais criterioso entre os projetos.

A estrutura de entrega via *branchs* e *commits* individuais facilitou a análise manual das refatorações, mas exigiu organização que nem todos os estudantes tinham. Em futuras turmas, consideramos reforçar esse ponto nas instruções. De modo geral, a prática foi bem recebida, gerando engajamento e contribuindo para o desenvolvimento de habilidades relevantes na formação dos estudantes.

6. Limitações do estudo

Como limitações do estudo, tem-se: (i) pouca experiência dos estudantes com a refatoração de *code smells* que buscamos diminuir com o treinamento de refatoração de código, (ii) baixa variedade de *code smells* no estudo que buscamos aumentar com o uso de duas ferramentas de detecção, (iii) a linguagem de programação se mostrou uma barreira para os estudantes que utilizam soluções mais atuais de programação, (iv) sistemas que os estudantes não tinham familiaridade e (v) a variação na complexidade dos sistemas pode ter afetado os resultados.

7. Conclusão

Neste trabalho, apresentamos um relato de experiência sobre o ensino de refatoração de *code smells*. Para isso, propusemos um treinamento nos conteúdos de refatoração de *code smells* em uma turma de manutenção de software e uma prática de refatoração em sistemas Java. Os resultados obtidos mostram que os estudantes conseguiram executar boa parte das refatorações (150 de 265), e receberam boas avaliações manuais da qualidade do código refatorado. Na percepção dos estudantes os *code smells* mais difíceis de refatorar foram o *long method* e *god class*, e as técnicas de refatoração mais difíceis de aplicar foram (*extract method* e *extract class*). Segundo os estudantes, a prática proporcionou habilidades e benefícios importantes para o engenheiro de software como, a capacidade analítica e resolução de problemas, seguido da melhoria na legibilidade do código.

Como trabalhos futuros, pretende-se: (i) utilizar outras ferramentas de detecção de *code smells*, (ii) analisar coocorrências de *code smells*, (iii) analisar outros sistemas em outras linguagens de programação, e (iv) avaliar o impacto das refatorações na previsão de débito técnico dos sistemas.

Agradecimentos

Agradecemos à bolsa PIBIC (Programa Institucional de Bolsas de Iniciação Científica) e PIBIT (Programa Institucional de Bolsas de Iniciação em Desenvolvimento Tecnológico e Inovação) da UFC Quixadá pelo apoio financeiro e incentivo durante o desenvolvimento deste trabalho, possibilitando a dedicação e o aprimoramento das atividades de pesquisa.

Referências

Al Dallal, J. (2013). Object-oriented class maintainability prediction using internal quality attributes. *Information and Software Technology*, 55(11):2028–2048.

AlOmar, E. A., AlOmar, S. A., and Mkaouer, M. W. (2023). On the use of static analysis to engage students with software quality improvement: An experience with pmd. In *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, pages 179–191. IEEE.

Bezerra, C., Alves, V. A., Lobo, A. H., Queiroz, J. P., Lima, L., and Meirelles, P. (2024). Contributing to open-source projects in refactoring code smells: A practical experience in teaching software maintenance. In *Simpósio Brasileiro de Engenharia de Software (SBES)*, pages 399–409. SBC.

Bryant, A. and Charmaz, K. (2010). *The SAGE handbook of grounded theory*. Sage publications.

Chaaban, P. C., Medeiros, R., Sousa, J. L., Rocha, M., Maia, G., Lima, I., Batista, R. D., and Rabelo, J. d. H. (2023). Codesmells? aqui não! limpando e refatorando códigos na prática: Um relato de experiência da execução do curso codesmells na prática. In *Workshop sobre Educação em Computação (WEI)*, pages 122–132. SBC.

Chren, S., Macák, M., Rossi, B., and Buhnova, B. (2022). Evaluating code improvements in software quality course projects. In *Proceedings of the 26th International Conference on Evaluation and Assessment in Software Engineering*, pages 160–169.

Fowler, M. (2018). *Refactoring*. Addison-Wesley Professional.

Kaur, A. (2020). A systematic literature review on empirical analysis of the relationship between code smells and software quality attributes. *Archives of Computational Methods in Engineering*, 27(4):1267–1296.

Mäntylä, M. V. and Lassenius, C. (2006). Subjective evaluation of software evolvability using code smells: An empirical study. *Empirical Software Engineering*, 11:395–431.

Matturro, G., Raschetti, F., and Fontán, C. (2019). A systematic mapping study on soft skills in software engineering. *J. Univers. Comput. Sci.*, 25(1):16–41.

Pereira dos Reis, J., Brito e Abreu, F., de Figueiredo Carneiro, G., and Anslow, C. (2022). Code smells detection and visualization: a systematic literature review. *Archives of Computational Methods in Engineering*, 29(1):47–94.

Rabelo, A., Maia, L. C. G., and Parreiras, F. S. (2018). Performance analysis of computer science students in programming learning. In *Workshop sobre Educação em Computação (WEI)*. SBC.

Tan, I. and Poskitt, C. M. (2024). Fixing your own smells: Adding a mistake-based familiarisation step when teaching code refactoring. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*, pages 1307–1313.

Wake, W. C. (2004). *Refactoring workbook*. Addison-Wesley Professional.