Análise da Influência da Aprendizagem Baseada em Projeto na Qualidade de Código de Jogos Digitais

Pedro H. Belo[1], **Laerte Xavier**[1]

[1]Pontifícia Universidade Católica de Minas Gerais (PUC MINAS)
Belo Horizonte – MG – Brazil

pedrohenrique1550@hotmail.com, laertexavier@pucminas.br

Abstract. *Video game development requires skills such as physical simulation and object animation. Teaching these skills requires methodological approaches that integrate theory and practice, where Project-Based Learning (PBL) stands out. However, there is a gap regarding the impact of PBL on the quality of digital game code, particularly in relation to "game smells", which indicate issues that affect maintainability and the gaming experience. This study evaluates 201 student-developed games over six years, finding an increase in "game smells" as the courses progress, a high incidence of less critical "game smells," and a significant trade-off between game size and incidence of "game smells".*

Resumo. *O desenvolvimento de videogames requer habilidades como, simulação de física e animação de objetos. O ensino dessas habilidades requer abordagens metodológicas que integrem teoria e prática, onde, a Aprendizagem Baseada em Projetos (PBL), se destaca. No entanto, há uma lacuna sobre o impacto do PBL na qualidade do código de jogos digitais, particularmente em relação aos "game smells", que indicam problemas que afetam a manutenibilidade e a experiência de jogo. Este estudo avalia 201 jogos desenvolvidos por estudantes ao longo de seis anos, encontrando um aumento nos "game smells" à medida que os cursos progridem, uma alta incidência de "game smells" menos críticos e uma correlação significativa entre o tamanho dos jogos e a incidência de "game smells".*

1. Introdução

Jogos digitais movimentaram globalmente, em 2022, o montante de US$ 197 bilhões, com uma expectativa de crescimento médio de 12,1% ao ano, até 2027 [SEBRAE 2024]. Ainda que o mercado esteja em crescimento, as competências para desenvolvimento de jogos digitais ainda representam um nicho de mercado [Borrelli et al. 2020]. Isso porque, ao desenvolver um jogo, os programadores lidam com diferentes competências, como simulação de física, animação de objetos e sincronização de jogos online [Bosco et al. 2023, Chueca et al. 2024]. Com o aumento da demanda, e uma indústria mais exigente, se faz necessário a busca por métodos de ensino que aproximem ao máximo o conteúdo teórico com a realidade do mercado [Tonhão et al. 2021]. Dentre os métodos de ensino para enfatizar práticas reais, pode-se destacar a Aprendizagem Baseada em Projeto (PBL, do inglês *Project-Based Learning*), que consiste em uma abordagem que envolve os alunos na construção ativa do conhecimento, por meio de resolução de problemas, os incentivando a desenvolverem projetos significativos e do mundo real, aproximando assim, a teoria e a prática. [Pérez and Rubio 2020, Guo et al. 2020].

Estudos anteriores abordam o uso do PBL no contexto universitário. Alguns tratam do engajamento dos alunos [Morais et al. 2021], outros abordam a aplicação prática em sala de aula [Pérez and Rubio 2020]. Há, ainda, estudos que demonstram as experiências acerca da utilização de ferramentas que auxiliam a aplicação do PBL [Gupta 2022]. Outros estudos demonstram maneiras de garantir o aprendizado considerando qualidade de código [Kirk et al. 2020]. Contudo, tais estudos consideram um curto período de atuação dos alunos, além de considerar um contexto geral de desenvolvimento na Engenharia de Software (SE, do inglês *Software Engineering*). Com a multidisciplinaridade necessária para o desenvolvimento de jogos [Nardone et al. 2023], as diferenças entre SE e a Engenharia de Software de Jogos (GSE, do inglês *Game Software Engineering*) ficam em evidência, incluindo equipes altamente iterativas, gerenciamento de muitos ativos, testes focados na experiência do jogador, requisitos subjetivos e uso de motores de jogos complexos [Chueca et al. 2024]. Dessa forma, o problema abordado neste trabalho refere-se à **carência de estudos que investiguem a evolução da qualidade de código no contexto de jogos digitais, desenvolvidos em disciplinas que adotam, continuadamente, a abordagem PBL.**

Particularmente, investiga-se neste trabalho a ocorrência de *game smells* em trabalhos desenvolvidos em disciplinas de um curso superior de Jogos Digitais. Os *game smells* representam uma sub-categoria de *code smells*. Estes, por sua vez, representam uma parcela significativa, dos custos associados à manutenção de software [Sobrinho et al. 2021, Almogahed et al. 2023]. Além disso, jogos digitais tendem a ser produtos com um ciclo de vida muito mais limitado do que outros produtos de software [Chueca et al. 2024]. Isso se dá em função de aspectos como prazos fixos, gerenciamento de grandes conteúdos e processo de desenvolvimento demasiadamente interativo, com o design sendo feito junto a implementação do próprio software [Chueca et al. 2024]. Todos esses elementos levam os desenvolvedores a tomarem decisões complexas de design e implementação [Nardone et al. 2023], levando-os a escrever código propenso à introdução de *code smells* [Sehgal et al. 2022, Borrelli et al. 2020].

Assim, o presente estudo objetiva **avaliar a evolução da qualidade de código e ocorrência de *game smells* em projetos desenvolvidos por alunos em disciplinas que adotam o PBL continuamente.** Para tanto, busca-se responder as seguintes RQs (do inglês, *Research Questions*):

RQ1. Qual é a relação entre a ocorrência de *game smells* e o nível de maturidade dos alunos em disciplinas que adotam o PBL?

RQ2. Quais os principais tipos de *game smells* identificados na evolução das disciplinas que adotam PBL?

RQ3. Qual é a relação entre o tamanho dos projetos desenvolvidos em disciplinas que adotam PBL e os *game smells* identificados?

Como resultado, observou-se que existe uma relação entre a incidência de *game smells* e o avanço da maturidade dos alunos em disciplinas que adotam continuamente a abordagem PBL. A análise indica que a incidência de *game smells* aumenta à medida que a maturidade dos alunos cresce, possivelmente motivada pelo aumento do escopo dos trabalhos. Isso é corroborado pela correlação mais forte observada entre o tamanho dos jogos e a incidência de *game smells*. Por fim, os resultados deste trabalho podem ser utilizados como guia para cursos de graduação em Jogos Digitais, uma vez que aponta a

importância do aumento da complexidade dos problemas a serem resolvidos pelos alunos, em consonância com a demanda crescente da qualidade dos projetos desenvolvidos.

O restante do trabalho está organizado em oito seções. A Seção 2 aborda o referencial teórico com o detalhamento de alguns conceitos relevantes na área do estudo. A Seção 3 conta com os trabalhos relacionados da área, enquanto a Seção 4 descreve os materiais e métodos utilizados no estudo. As seções 5 e 6 apresentam e discutem os resultados obtidos, respectivamente. A Seção 7 discute as ameaças a validade e, por fim, a seção 8 apresenta as conclusões do estudo.

2. Fundamentação Teórica

Code smells referem-se a características ou padrões de código-fonte que indicam a possível presença de problemas no design ou na implementação de um software [Fowler 2018]. Eles proveem indícios de que partes do código podem não estar otimizadas, eficientes, ou podem ser difíceis de entender e manter [Palomba et al. 2014]. Como consequência, esses *code smells* podem aumentar a complexidade, causar problemas de escalabilidade, desempenho e o aumento da dívida técnica [Sobrinho et al. 2021].

A maioria dos catálogos, como o de Fowler (2018), focam em *code smells* num contexto geral, e não conseguem descrever todos os problemas de design de código de videogames [Agrahari and Chimalakonda 2022]. Por isso, Nardone et al. (2023) desenvolveram um catálogo específico de *code smells* focados no contexto de videogames, chamados de *game smells*. Tal catálogo dividiu os *game smells* em cinco categorias, além de coletarem opiniões de desenvolvedores de jogos sobre a criticidade dos mesmos.

A primeira categoria, ***design and game logic***, inclui *smells* relacionados às decisões de arquitetura geral do jogo, design de baixo nível, a organização de objetos do jogo e escolhas de implementação. Nela há um total de 9 *game smells*, onde todos, exceto três, são percebidos como críticos e altamente críticos. Um exemplo de *smell* desta categoria é *Creating components/objects at run-time*, que no contexto de jogos digitais, a criação rápida de objetos é crucial, logo criar e destruir objetos individualmente pode comprometer significativamente o desempenho, portanto, recomenda-se o uso de *pools* de objetos para mitigar problemas de desempenho [Nardone et al. 2023].

A segunda categoria, de **multiplayer**, trata de decisão ruins relacionadas ao design e implementação do componente multiplayer de um jogo. Nela há um total de 4 *game smells*, onde todos eles foram percebidos como críticos e altamente críticos. Um exemplo é o *Storage of game status on clients*, que pode causar problemas de sincronização e vulnerabilidades de segurança, como ataques de trapaça. Para este, recomenda-se armazenar o status no servidor e implementar medidas de segurança, como a serialização de dados e controle de autoridade sobre modificações no estado do jogo [Nardone et al. 2023].

Na terceira categoria, **animação**, o foco está em como os objetos do jogo são animados, com 6 *smells* percebidos de forma neutra. Um exemplo é *continuously checking if position/rotation is within the boundary*, que trata do movimento de objetos em regiões confinadas. Verificações contínuas podem causar atraso ou comportamento inadequado. A solução sugerida é o *hash espacial*, que melhora o desempenho ao dividir a cena em seções e limitar as verificações de colisão a objetos dentro dessas seções [Nardone et al. 2023].

Em se tratando da **física** do jogo, esta categoria irá tratar da maneira como a física do jogo está sendo computada, por meio de scripts ou aproveitando componentes existentes e configurando colisores. Nela há 3 *smells* com a maioria considerado crítico e altamente crítico. Um exemplo é *Heavy weight physics computation* que trata de cálculos de física no *loop* principal, resultando em possíveis gargalos de desempenho. Uma das estratégias são atualizações baseadas em eventos externos ou intervalos fixos e técnicas como tabelas de consulta para acelerar o cálculo da física [Nardone et al. 2023].

Por fim, na categoria **renderização**, a preocupação é com questões relacionadas à maneira como os objetos são desenhados/renderizados, nela há um total de 6 *smells*, com a maioria percebidos como críticos e altamente críticos. Um exemplo é o *Lack of optimization when rendering objects*, que detecta desenho/renderização de objetos não otimizada corretamente, que pode causar quedas de desempenho. Estratégias como a aplicação de materiais transparentes e o uso de níveis de detalhe podem melhorar a eficiência. Recomenda-se também o uso de seleção de oclusão para limitar a renderização apenas aos objetos visíveis pela câmera [Nardone et al. 2023].

3. Trabalhos Relacionados

Estudos anteriores exploraram a qualidade de código de projetos Unity, por exemplo, Nardone et al. (2023) usam uma abordagem empírica para identificar e catalogar *game smells* no desenvolvimento de jogos digitais, envolvendo mineração de discussões em fóruns de desenvolvimento de jogos, seguida por uma análise manual das postagens para identificar, categorizar e avaliar a relevância dos *game smells*. Já Bosco et al. (2023) apresentam uma ferramenta de detecção de *game smells* em projetos de desenvolvimento de jogos na plataforma Unity, o UnityLint. O método empregado envolve o uso de uma catálogo de *game smells* específicos de jogos na Unity, seguido pela implementação do UnityLint para detectar 18 desses *game smells* por meio de análise estática do código-fonte e metadados dos projetos, com uma microprecisão de 78%, uma macroprecisão de 92%, uma microrecall de 94% e uma macrorecall de 85%. A escolha do UnityLint se justifica pelo uso do Unity como plataforma no curso de Jogos Digitais, além de seu uso na avaliação dos jogos analisados. Além disso, o catálogo é utilizado para auxiliar na avaliação dos *game smells* nos jogos avaliados.

No contexto de trabalhos sobre PBL, Souza et al. (2019) investigam as percepções dos estudantes na educação em SE, destacando a melhoria em requisitos de software, design e métodos ágeis. Já Gupta (2022) se concentra em avaliar o impacto da aprendizagem baseada em design (DBL, do inglês, *Desing Based Learnig*) e PBL no ensino de SE, mostrando que aumentam o envolvimento e o desempenho dos estudantes. Por fim, Lu et al. (2019) introduz o paradigma de inspeção contínua no ambiente de sala de aula, conduzindo um experimento controlado. Os resultados mostraram que a inspeção contínua auxiliou os alunos a identificar maus hábitos de codificação, e reduzir problemas de qualidade de código. Este trabalho complementa os estudos relacionados, ao mostrar dados quantitativos da evolução da qualidade de código, contribuindo com a escassez de estudos envolvendo o PBL no contexto de GSE.

4. Materiais e Métodos

Nesta seção, são apresentados os métodos e ferramentas utilizadas para realização deste estudo empírico de cunho exploratório.

4.1. Obtenção dos Dados

Para responder as RQs propostas, analisou-se o código-fonte resultante do desenvolvimento de software em projetos Unity, do curso de Jogos Digitais da Pontifícia Universidade Católica de Minas Gerais (PUC MINAS). Até 2019, os jogos desenvolvidos no curso eram gravados em DVDs e armazenados na biblioteca da PUC MINAS. Assim, foram lidos 366 discos de DVDs que correspondem aos jogos desenvolvidos nos anos de 2014 até 2019, provenientes das disciplinas de Trabalho Interdisciplinar de Software (TIS), ministradas semestralmente do primeiro ao quarto período, e de Trabalho de Conclusão de Curso (TCC) I e II, do quinto (TCC I) e sexto período (TCC II). Essas disciplinas adotaram a metodologia PBL ao longo de todos os anos analisados, ministradas por diversos professores.

A cada semestre, as disciplinas abordam desafios diferentes, desenvolvidos utilizado metodologias específicas, de maneira progressiva ao longo dos semestres [Nelson et al. 2017]. No TIS I, os alunos seguem um método ágil de desenvolvimento para jogos do gênero *Shoot'em up*; já no TIS II, concentram-se em jogos *mobile*; no TIS III, exploram-se os *Serious Games*, realidade virtual e integração com a sociedade; no TIS IV, o foco recai sobre a criação de curvas de dificuldade, jogos 3D e aprimoramento de controles e mecânicas de movimento; o já os trabalhos dos períodos V e VI são dedicados ao TCC. É importante destacar que, naquela época, o curso continha 6 períodos.

Devido à pandemia de 2020, os jogos produzidos no curso deixaram de ser gravados fisicamente e passaram a ser entregues digitalmente, somente com o executável do jogo. Isto é, os trabalhos passaram a ser entregues sem o código-fonte, motivo pelo qual não foi possível fazer análise de anos posteriores a 2019.

4.2. Coleta de Dados

A coleta e análise de dados, se deu com as seguintes etapas: (i) leitura dos DVDs; (ii) inspeção manual em busca do código-fonte; (iii) transferência dos dados para o BlobStorage do Azure; (iv) análise estática usando o UnityLint, para coleta dos *game smells*; (v) análise estática do Qodana para coleta de métricas de tamanho de projeto; (vi) tratamento dos dados coletados, onde os dados extraídos previamente na etapa de coleta de metadados e análise estática de código são higienizados e tratados; (vii) análise dos dados coletados, na qual os dados coletados são analisados manualmente; (viii) por fim, são elaborados gráficos para ilustrar as análises realizadas.

Do total de 366 DVDs inicialmente examinados, 165 foram excluídos devido a dois principais motivos: a corrupção do disco (16 unidades), falta de identificação do período correspondente (20 unidades) e a ausência de código-fonte nos DVDs (129 discos). Nesse contexto, ressalta-se que a razão pela qual 129 DVDs não continham código-fonte está relacionada ao projeto pedagógico do curso de Jogos Digitais da PUC MINAS na época da execução dos trabalhos. Até 2019, o desenvolvimento dos projetos ocorria ao longo de um ano, não semestralmente. Em outras palavras, os projetos eram concebidos durante um período e os DVDs eram entregues somente com o projeto; posteriormente, no semestre seguinte, o código era desenvolvido e o jogo finalizado, resultando na entrega de um novo DVD, por isso, trabalhos do primeiro período, por exemplo, foram totalmente desconsiderados. Outro aspecto notado está relacionado a um possível erro na entrega do disco, com alguns DVD com a pasta do código-fonte vazia.

4.3. Análise de dados

Os jogos foram analisados usando duas abordagens. A primeira dedica-se em avaliar o tamanho do código, utilizando a ferramenta de análise estáticas comuns, como o Qodana, desenvolvido pela Jetbrains. Em seguida, tais jogos foram analisados pelo UnityLint, outra ferramenta de análise estática, com o foco em projetos Unity, mesma plataforma utilizada no curso, e especializada em buscar *game smells* [Bosco et al. 2023]. As métricas utilizadas foram: (i) Linhas de código não comentadas, **NCLOC**, (do inglês, Non-Comment Lines of Code) e (ii) ***Game smells***, a qual é o número total de más práticas específicas de jogos digitais encontradas pelo UnityLint.

Para avaliar as métricas obtidas, foi utilizado Python com a biblioteca Pandas para criar dois tipos de *DataFrames*: o primeiro contendo o ID do jogo, o número de linhas de código (NLOC) e a quantidade de *game smells*; o segundo com os IDs dos jogos e os *smells* categorizados por tipo e quantidade. A partir dos dados consolidados, a biblioteca SciPy foi utilizada para calcular a correlação de Spearman e realizar o teste de Kruskal entre as variáveis. Particularmente, utilizou-se Spearman considerando que os dados não estão normalizados, com o *p-value* do teste Shapiro-Wilk apresentando um valor de 1.8365×10^{-12}.

5. Resultados

Os resultados apresentados nesta seção incluem a caracterização dos dados coletados e os resultados das métricas calculadas para as RQs definidas.

5.1. Caracterização do Conjunto de Dados

A Tabela 1 mostra a distribuição anual e por período dos jogos coletados. Em média, são entregues 35 jogos por ano, com destaque para 2018, que teve 58 jogos. A coleta de dados diminui nos períodos mais avançados, possivelmente devido a falhas na coleta ou desistências. No segundo período, foram entregues 69 jogos, enquanto no sexto período, correspondente ao TCC, foram entregues 22 jogos.

Tabela 1. Contagem de dados obtidos por ano e período

Anos	Períodos					Total Geral
	2	3	4	5	6	
2014	5	3	4	–	–	12
2015	11	8	6	9	3	37
2016	16	6	4	3	5	34
2017	8	8	3	6	–	25
2018	20	11	10	9	8	58
2019	9	8	4	8	6	35
Total Geral	69	44	31	35	22	201

Considerando as métricas de número de linhas de código, ncloc, a Figuta 1 mostra uma progressão no tamanho dos jogos à medida que as disciplinas avançam. Inicialmente, a mediana é de 2.790 linhas de código no segundo período, 5.520 para o terceiro, 11.820

para o quarto, 12.690 no quinto, e por fim, 20.000 linhas no sexto período. Usando o teste Kruskal-Wallis, o *p-value* foi de 1.66×10^{-10}, indicando que a diferença nas medianas são estatisticamente significativas.

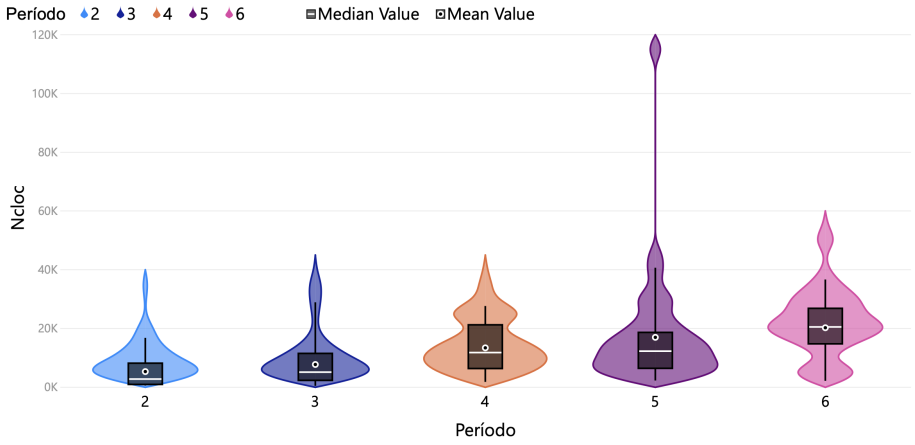

Figura 1. Ncloc por período

5.2. Qual é a relação entre a ocorrência de *game smells* e o nível de maturidade dos alunos em disciplinas que adotam o PBL

Para investigar a RQ1, foi analisada a distribuição dos *game smells* identificados pelo UnityLint ao longo das disciplinas, abrangendo do segundo ao sexto período, conforme ilustrado na Figura 2.

A análise dos quartis indica que, no segundo período, o primeiro quartil mostra até 50 *game smells*, a mediana revela até 130, e o terceiro quartil indica até 190, com o máximo chegando a 1.250. No terceiro período, os valores aumentam ligeiramente: o primeiro quartil tem 120 *game smells*, a mediana chega a 200, e o terceiro quartil alcança 270, com o máximo em 780. O quarto período continua essa tendência, com o primeiro quartil apresentando até 210 *game smells*, a mediana atingindo 360, e o terceiro quartil 630, enquanto o valor máximo é de 1.140. No quinto período, há um aumento significativo: o primeiro quartil tem 180 *game smells*, a mediana chega a 320, e o terceiro quartil atinge 540, com o máximo alcançando 2.530. Finalmente, no sexto período, correspondente à entrega do TCC, o primeiro quartil revela até 280 *game smells*, a mediana apresenta 580, e o terceiro quartil possui até 2.090 ocorrências.

O teste de Kruskal-Wallis, resultou em um *p-value* de 4.63×10^{-11}, indicando que a hipótese nula pode ser rejeitada, evidenciando diferenças significativas entre as medianas dos períodos. Já o teste de correlação de Spearman teve como resultado um *p-value* igual a 1.0623×10^{-14}, que indica que, a hipótese nula, foi rejeitada. Isto é, existem evidências que há uma correlação significativa entre as variáveis. Já o coeficiente de correlação obtido, *r-value*, foi de 0,5212, indicando uma correlação positiva moderada. Isso denota que, a medida que a disciplina progride, a incidência de *game smells* aumenta.

5.3. Quais os principais tipos de *game smells* identificados na evolução das disciplinas que adotam PBL

Do total de 18 tipos de *smells* detectáveis pelo UnityLint, foram encontrados na base de dados analisada 11 tipos diferentes, com um total de 63.148 ocorrências. Desse total, *Poor*

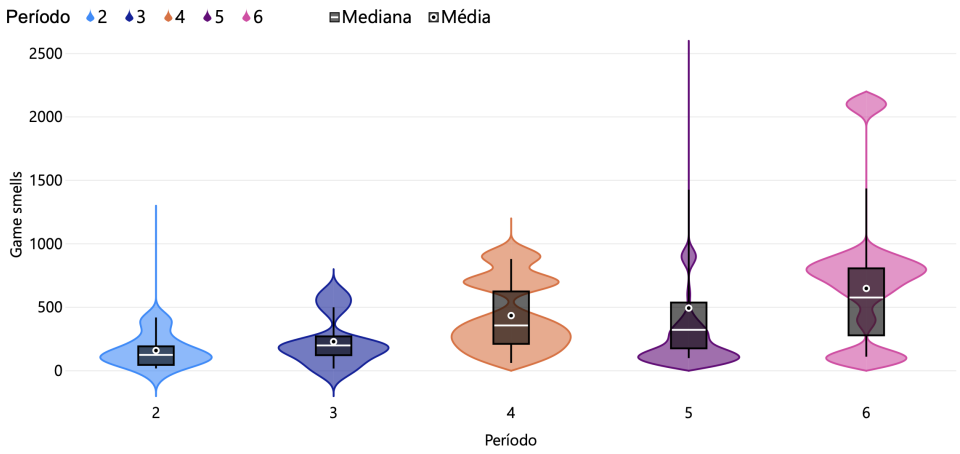

Figura 2. Distribuição de *game smells* por período

State Design e *Dependency Between Objects* correspondem a 91,3% das ocorrências.

Quanto à criticidade dos *game smells*, observa-se que 55,76% são classificados como *critical*, sendo os principais tipos *Poor State Design* e *Instantiate Destroy*. Outros 44,24% classificados como *neutral*, com exemplos como *Dependency Between Object*. Por fim, apenas 0,0048% são classificados como *highly*, sendo *Heavy Physics Computation* o único com esta criticidade.

A Figura 3 apresenta a distribuição dos *game smells* ao longo dos períodos. Nota-se que os *game smells* mais frequentes, *Poor State Design* e *Dependency Between Objects*, ocorrem em todos os períodos, com o primeiro distribuído de maneira quase uniforme entre os períodos, e o segundo, mais concentrado em períodos mais avançados. Para esse *smells*, destaca-se a sua ocorrência no quinto período, com 34,27% *smells*, seguido do sexto período, com 24,46%. O quinto período destaca-se dentre os outros períodos, visto que, foi identificado *smells* que só ocorrem neste período, além de, geralmente, a maior incidência de *smells* recair sobre ele. Destaca-se ainda que, a medida que a disciplina avança, alguns *smells* perdem a representatividade, como o *Find Methods*, com 56,38%, no segundo período, contra 4,86% no sexto período.

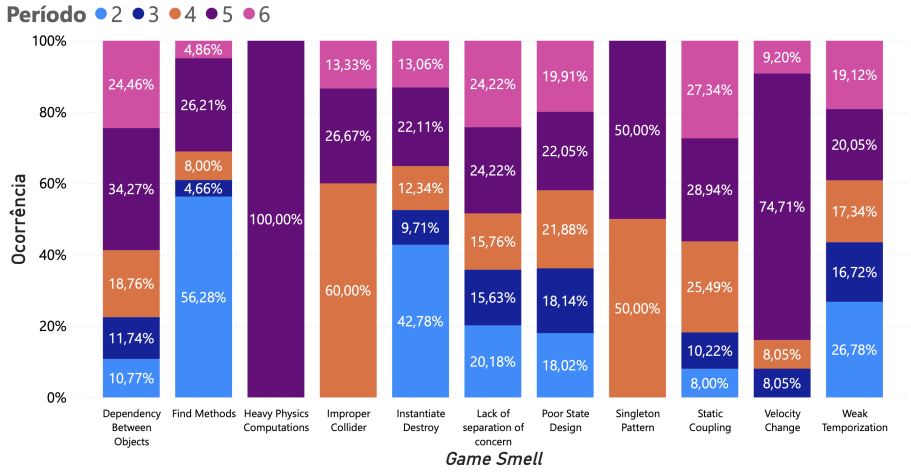

Figura 3. Distribuição dos *game smells* por período

5.4. Qual é a relação entre o tamanho dos projetos desenvolvidos em disciplinas que adotam PBL e os *game smells* identificados?

Para investigar a RQ3, foi analisada a dispersão dos *game smells* em relação ao tamanho dos projetos (ncloc), conforme apresentado na Figura 4. Na figura, observa-se que projetos maiores tendem a apresentar uma maior incidência de *game smells*.

A análise dos quartis revela aspectos importantes sobre a distribuição do tamanho dos projetos. Cerca de 25% dos dados possuem 2.372 linhas de código, 50% possuem 7.168 linhas, 75% têm até 15.280, e o quarto quartil chega a até 115.703 linhas. A disparidade entre o terceiro e o quarto quartil se deve à presença de *outliers*, como os jogos "Elemental Ranger" com 115.703 linhas e "Wells", com 51.383.

O *p-value* resultante foi significativamente baixo, com uma magnitude de 2.3752×10^{-54}, indicando rejeição da hipótese nula. O coeficiente de correlação *r-value* foi calculado em 0,8493, revelando uma correlação forte entre as variáveis. Isso sugere que, à medida que o tamanho dos jogos aumenta, a incidência de *game smells* também cresce.

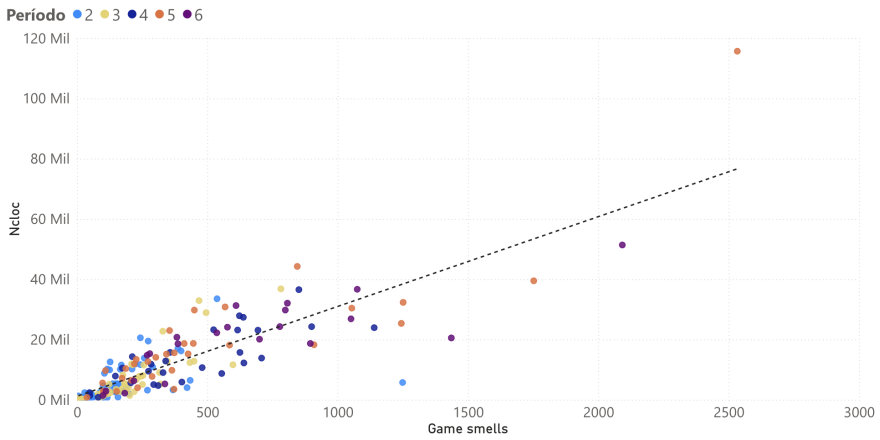

Figura 4. Dispersão entre Game smells e ncloc

6. Discussão dos resultados

Esta seção apresenta as implicações geradas pelas RQs respondidas neste estudo.

1. A incidência de *Game smells* aumenta a medida que a maturidade dos alunos progridem. Entretanto, tal aumento não pode ser explicado somente pelo progresso da maturidade. Isso porque, a relação estatística analisada na RQ1 possui um *r-value* de 0,5212, considerado uma relação moderada. Contudo, o *r-value* entre os *smells* e o ncloc, foi significativamente maior, 0,8493. Isso sugere que não é apenas a evolução na maturidade dos alunos que leva a um aumento na incidência de *game smells*, mas sim que, períodos maiores exigem projetos maiores e mais complexos, o que consequentemente eleva a incidência desses *smells*.

2. Foram encontrados 11 tipos diferentes de *game smells*, destacando-se três principais: *Poor State Design*, *Dependence Between Objects* e *Heavy Physics Computation*. Nardone et al. (2023) [Nardone et al. 2023] avaliam *Poor State Design* como crítico devido à dificuldade de manutenção, sugerindo o uso de *design patterns*, embora alertem

que esses padrões ”dificilmente facilitam as coisas”e que o foco deve ser em manter a aplicação simples. O segundo *game smell* mais frequente, *Dependency Between Objects*, está relacionado à alta dependência entre classes, comprometendo a manutenção e elevando o risco de falhas [Beck 2013]. Contudo, os desenvolvedores preferem priorizar desempenho ao desacoplamento [Nardone et al. 2023]. Por fim, o *Heavy Physics Computation*, com três ocorrências, foi considerado o mais crítico, pois pode causar queda de desempenho devido ao processamento excessivo, ocorrendo apenas no quinto período (TCC I), em um único jogo, chamado Relicário.

Nota-se também que a progressão dos trabalhos reduziu a presença de alguns *game smells*, como o *Instantiate Destroy* e *Find Methods*. Onde este último tinha uma presença de 56,28% no segundo período, seguido de uma queda significativa, para 4,66% no terceiro período e com 4,86% no sexto. Tal queda pode ser explicada pela introdução a padrões de projetos na ementa da disciplina do terceiro período [PUC MINAS 2024].

3. Projetos maiores refletem uma maior incidência de *game smells*. A análise estatística dos dados obtidos na RQ3 revela uma correlação notável entre as variáveis (ncloc) e *game smells*, evidenciada por um *r-value* de 0.8493, indicando uma relação estatisticamente robusta. Assim, é possível inferir que quanto maior o projeto, maior a probabilidade de ocorrência de *game smells*. Esse padrão é compreensível, uma vez que projetos maiores tendem a envolver escopos mais amplos e complexos.

7. Ameaças à Validade

A validade de construção é limitada considerando a baixa representatividade estatística da amostra, problemas na leitura dos DVDs e à impossibilidade de obter dados recentes. Todos os jogos disponíveis foram analisados para mitigar essas ameaças. Quanto à validade interna, identificou-se a possibilidade de erros na transferência de arquivos, aceitando-se apenas jogos copiados sem erros. Quanto a validade externa, não é possível generalizar os resultados, considerando que, eles refletem um contexto educacional específico. Para reduzir essa fragilidade, foram considerados 201 jogos de 2014 a 2019, totalizando 2 milhões de linhas de código, produzidos por diversos alunos e orientados por diversos professores.

8. Conclusão

Este estudo explorou o impacto do PBL na qualidade do código de jogos digitais desenvolvidos ao longo de seis anos. A amostra abrange períodos de 2014 a 2019, com uma amostra de 201 jogos digitais, do segundo ao sexto período. Para isso, utilizou-se métricas de tamanho de código e *game smells*, analisando seu comportamento e correlacionando com os períodos acadêmicos. Os resultados indicam uma correlação significativa entre as métricas analisadas e a progressão dos alunos. Embora a amostra tenha limitações devido à ausência de código-fonte em períodos mais recentes e problemas na leitura dos discos, os dados sugerem um aumento na quantidade de *game smells* à medida que as disciplinas avançam. No entanto, não é possível afirmar que a abordagem PBL foi a causa desse aumento na incidência de *game smells* ou que a qualidade do código piorou. Uma das hipóteses para tal resultado é que, em cada período, o escopo do trabalho varia tanto em quantidade quanto em tipo de abordagem, expondo os alunos a diferentes tecnologias e métodos de implementação. Essa variação pode resultar em uma maior incidência de alguns *game smells* em certos períodos e menor em outros. Por outro lado, a progressão

das disciplinas mostrou-se benéfica na redução de alguns *smells*, como *Find Methods* e *Instantiate Destroy*. Como trabalhos futuros, sugere-se a aplicação desta metodologia em outras instituições que adotam PBL, de modo a comparar resultados.

Pacote de Replicação

O pacote de replicação deste trabalho encontra-se disponível em: `https://zenodo.org/records/13345975`

Referências

Agrahari, V. and Chimalakonda, S. (2022). A catalogue of game-specific anti-patterns. In *Proceedings of the 15th Innovations in Software Engineering Conference*, ISEC '22, New York, NY, USA. Association for Computing Machinery.

Almogahed, A., Omar, M., Zakaria, N. H., Muhammad, G., and AlQahtani, S. A. (2023). Revisiting scenarios of using refactoring techniques to improve software systems quality. *IEEE Access*, 11:28800–28819.

Beck, F. (2013). Analysis of multi-dimensional code couplings. In *2013 IEEE International Conference on Software Maintenance*, pages 560–565. IEEE.

Borrelli, A., Nardone, V., Di Lucca, G. A., Canfora, G., and Di Penta, M. (2020). Detecting video game-specific bad smells in unity projects. In *2020 IEEE/ACM 17th International Conference on Mining Software Repositories (MSR)*, pages 198–208.

Bosco, M., Cavoto, P., Ungolo, A., Muse, B. A., Khomh, F., Nardone, V., and Di Penta, M. (2023). Unitylint: A bad smell detector for unity. In *2023 IEEE/ACM 31st International Conference on Program Comprehension (ICPC)*, pages 186–190.

Chueca, J., Verón, J., Font, J., Pérez, F., and Cetina, C. (2024). The consolidation of game software engineering: A systematic literature review of software engineering for industry-scale computer games. *Information and Software Technology*, 165:107330.

Fowler, M. (2018). *Refactoring: Improving the Design of Existing Code*. Addison-Wesley, 2nd edition.

Guo, P., Saab, N., Post, L. S., and Admiraal, W. (2020). A review of project-based learning in higher education: Student outcomes and measures. *International Journal of Educational Research*, 102:101586.

Gupta, C. (2022). The impact and measurement of today's learning technologies in teaching software engineering course using design-based learning and project-based learning. *IEEE Transactions on Education*, 65(4):703–712.

Kirk, D., Crow, T., Luxton-Reilly, A., and Tempero, E. (2020). On assuring learning about code quality. In *Proceedings of the Twenty-Second Australasian Computing Education Conference*, ACE'20, page 86–94, New York, NY, USA. Association for Computing Machinery.

Lu, Y., Mao, X., Wang, T., Yin, G., and Li, Z. (2019). Improving students' programming quality with the continuous inspection process: a social coding perspective. *Frontiers of Computer Science*, 14.

Morais, P., Ferreira, M. J., and Veloso, B. (2021). Improving student engagement with project-based learning: A case study in software engineering. *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*, 16(1):21–28.

Nardone, V., Muse, B., Abidi, M., Khomh, F., and Di Penta, M. (2023). Video game bad smells: What they are and how developers perceive them. *ACM Trans. Softw. Eng. Methodol.*, 32(4).

Nelson, M. A. V., Carneiro, R. V., and Costa, M. R. (2017). Interdisciplinary software projects as an active methodology to practice for the profession. In *2017 IEEE/ACM 1st International Workshop on Software Engineering Curricula for Millennials (SECM)*, pages 28–32.

Palomba, F., Bavota, G., Di Penta, M., Oliveto, R., and De Lucia, A. (2014). Do they really smell bad? a study on developers' perception of bad code smells. In *2014 IEEE International Conference on Software Maintenance and Evolution*, pages 101–110.

Pérez, B. and Rubio, A. L. (2020). A project-based learning approach for enhancing learning skills and motivation in software engineering. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, SIGCSE '20, page 309–315, New York, NY, USA. Association for Computing Machinery.

PUC MINAS (2024). Grade curso jogos digitais. `https://portal.pucminas.br/ensino/grade/programa.php?curso=375`. Acesso em 20 de maio de 2024.

SEBRAE (2024). Tendências para a indústria de games em 2024. `https://digital.sebraers.com.br/blog/mercado/tendencias-para-a-industria-de-games-em-2024/`. Acesso em 20 de março de 2024.

Sehgal, R., Mehrotra, D., Nagpal, R., and Sharma, R. (2022). Green software: Refactoring approach. *Journal of King Saud University - Computer and Information Sciences*, 34(7):4635–4643.

Sobrinho, E. V. d. P., De Lucia, A., and Maia, M. d. A. (2021). A systematic literature review on bad smells–5 w's: Which, when, what, who, where. *IEEE Transactions on Software Engineering*, 47(1):17–66.

Souza, M., Moreira, R., and Figueiredo, E. (2019). Students perception on the use of project-based learning in software engineering education. In *Proceedings of the XXXIII Brazilian Symposium on Software Engineering*, SBES '19, page 537–546, New York, NY, USA. Association for Computing Machinery.

Tonhão, S., Medeiros, A., and Prates, J. (2021). Uma abordagem prática apoiada pela aprendizagem baseada em projetos e gamificação para o ensino de engenharia de software. In *Anais do Simpósio Brasileiro de Educação em Computação*, pages 143–151, Porto Alegre, RS, Brasil. SBC.