REVISTA TÓPICOS
ISSN 2965-6672 | Qualis A2

REFATORAÇÃO, CODE SMELLS E EFICIÊNCIA ENERGÉTICA DE SOFTWARE: UMA REVISÃO NARRATIVA DA LITERATURA

REFACTORING, CODE SMELLS AND SOFTWARE ENERGY EFFICIENCY: A NARRATIVE LITERATURE REVIEW

Ciências Exatas e da Terra, Engenharias • 24/11/2026

REGISTRO DOI: 10.70773/revistatopicos/795466372

Carlos Eduardo de Oliveira Curti[1]

Lucas de Souza Cardia[2]

Altemar Sales de Oliveira[3]

RESUMO

Este estudo analisa os efeitos da refatoração e da remoção de code smells, indícios de problemas na organização do código, sobre o consumo de energia. A revisão narrativa reúne 18 estudos de intervenção com avaliação energética e três estudos contextuais, selecionados por busca dirigida e examinados nas fontes acessíveis. A síntese compara transformações, tarefas e componentes medidos. Os resultados incluem reduções, aumentos e diferenças não detectadas estatisticamente. Trocas de coleções dependem das operações e dos dados; intervenções móveis dependem dos recursos ativados; combinações de mudanças nem sempre preservam os ganhos individuais. A heterogeneidade experimental impede estimar uma economia média geral. Conclui-se que benefícios estruturais não garantem economia energética, que exige avaliação para um serviço equivalente. A seleção não exaustiva e a concentração em Java e Android limitam a generalização.

Palavras-chave: refatoração; code smells; eficiência energética; qualidade de software; revisão de literatura.

ABSTRACT

This study examines how refactoring and code smell removal affect software energy consumption. Code smells are indicators of potential problems in code organisation. The narrative review includes 18 intervention studies assessing energy and three contextual studies, selected through targeted searches and examined using accessible sources. The synthesis compares transformations, workloads, and measured components. Results include reductions, increases, and differences not detected statistically. Collection substitutions depend on operations and data; mobile interventions depend on activated resources; combinations of changes do not always preserve individual gains. Experimental

heterogeneity prevents estimating a general average saving. Structural benefits therefore do not guarantee energy savings, which require assessment for an equivalent service. Non-exhaustive selection and the concentration on Java and Android limit generalisation.
Keywords: refactoring; code smells; energy efficiency; software quality; literature review.

1. INTRODUÇÃO

Refatorar é reorganizar o código para facilitar sua compreensão e manutenção, preservando o comportamento que o programa oferece a seus usuários e a outros sistemas (Fowler, 2004, 2018). Ao modificar classes, métodos ou estruturas de dados, a refatoração também pode alterar as operações executadas pelo computador. Por isso, sua relação com o consumo de energia precisa ser investigada, e não presumida a partir da qualidade aparente do código.

Os resultados disponíveis não apontam uma direção única. Verdecchia *et al.* (2018) encontraram diferença global significativa entre tratamentos em uma das três aplicações avaliadas, enquanto Ournani *et al.* (2021) não identificaram efeito substancial atribuível às refatorações nas funcionalidades examinadas. Em outro desenho, Wang, Li e Tan (2026) mostraram que alterar o volume de entrada pode mudar a classificação energética de uma mesma transformação. Esses trabalhos não constituem repetições do mesmo experimento: diferem nas alterações, nas tarefas e nos componentes medidos. Compará-los exige considerar essas diferenças.

Revisões anteriores ajudam a situar o problema. Connolly Bree e Ó Cinnéide (2025) discutem a relação entre projeto de software e energia, incluindo a qualidade dos experimentos. Poy *et al.* (2025) tratam de padrões de projeto, code smells e refatoração; Fleischle *et al.* (2026) abrangem práticas de programação. Souza, Souza e Nunes (2025) organizam técnicas e ferramentas associadas à eficiência energética do código. O presente trabalho complementa esses recortes com uma síntese centrada nas condições que sustentam cada resultado, sem reivindicar cobertura exaustiva ou superioridade sobre outras revisões.

O objetivo é analisar criticamente as evidências sobre os efeitos da refatoração e da remoção de code smells no consumo energético de software, considerando seus contextos de aplicação e limitações. A pergunta orientadora é: em quais condições as intervenções examinadas alteram o consumo de energia, e o que limita o uso desses resultados em outros sistemas? Para respondê-la, a revisão relaciona três aspectos: a transformação aplicada, a tarefa executada e o objeto da medição. Essa organização permite discutir oportunidades de economia sem confundir qualidade estrutural, desempenho e energia.

2. FUNDAMENTAÇÃO CONCEITUAL

2.1. Refatoração e Qualidade Estrutural

A definição de refatoração estabelece dois compromissos: melhorar a estrutura interna e preservar o comportamento observável (Fowler, 2004, 2018). Não é suficiente que a nova versão compile; ela deve continuar atendendo aos requisitos relevantes. Uma substituição de estrutura de dados, por exemplo, pode exigir a manutenção da

ordem dos elementos e do comportamento quando várias tarefas acessam os mesmos dados. Passar nos testes disponíveis oferece evidência limitada aos casos exercitados, como reconhecem os estudos de substituição de coleções de Manotas, Pollock e Clause (2014) e Oliveira *et al.* (2021).

Code smells são indícios que motivam a investigação da organização ou implementação do código. Não demonstram, por si sós, um defeito funcional ou desperdício de energia. Exemplos são métodos excessivamente longos e classes que concentram responsabilidades demais. Os catálogos também variam: Verdecchia *et al.* (2018) estudam problemas estruturais clássicos em Java, enquanto Palomba *et al.* (2019) incluem práticas específicas de aplicativos Android. Portanto, contar alertas removidos não equivale a medir economia energética.

Clean Code, ou código limpo, reúne recomendações de clareza e manutenção (Martin, 2008). Nesta revisão, o termo fornece contexto, mas não representa um tratamento experimental único. A análise considera alterações identificáveis, como extrair um método, trocar uma coleção ou liberar um recurso. Coleções são estruturas que armazenam e organizam grupos de elementos; diferentes implementações podem executar inserções, buscas e remoções de maneiras distintas, com custos energéticos dependentes do uso (Pereira *et al.*, 2016; Hasan *et al.*, 2016).

Também é necessário distinguir refatoração de otimização que altera o serviço. Reduzir a frequência de localização, por exemplo, pode economizar energia e diminuir a atualização oferecida ao usuário. Banerjee e Roychoudhury (2016) incluem escolhas dessa natureza. Seus resultados são pertinentes ao tema, mas essa parcela

das mudanças não deve ser apresentada como refatoração estritamente preservadora do comportamento. Da mesma forma, manutenibilidade, entendida como facilidade de modificar o software, precisa de avaliação própria; um indicador calculado a partir do código não mede diretamente o esforço humano de manutenção (Cruz *et al.*, 2019).

2.2. Energia, Potência e Desempenho

Energia e potência representam grandezas diferentes. A potência indica a taxa de transferência de energia e é expressa em watts (W). A energia acumulada durante uma execução depende da potência ao longo do tempo e pode ser expressa em joules (J). Para o intervalo entre os instantes inicial e final, essa relação é:

$$E = \int_{t_0}^{t_1} P(t)\, dt \quad (1)$$

Na Equação 1, E é a energia, P(t) é a potência no instante t, e t_0 e t_1 delimitam a observação. Com potência em watts e tempo em segundos, obtém-se energia em joules. Também são usuais watt-hora (Wh) e quilowatt-hora (kWh); 1 Wh corresponde a 3.600 J. Uma execução mais curta não necessariamente consome menos energia, pois pode exigir potência maior. Essa separação entre duração, potência e consumo é necessária na avaliação de software (Hindle, 2016; Guldner *et al.*, 2024).

A carga elétrica é outra grandeza, expressa, por exemplo, em ampère-hora (Ah) ou miliampère-hora (mAh). Sua conversão em energia exige conhecer a tensão pertinente ao intervalo. Por isso, os resultados em mAh de Li e Halfond (2014) são tratados separadamente dos resultados em joules. Analogamente, a potência

em watts apresentada por Sehgal *et al.* (2022) não é tomada como energia por tarefa. A Tabela 1 reúne as distinções usadas na leitura dos estudos.

Tabela 1. Grandezas consideradas na interpretação dos estudos

Grandeza	**Unidade usual**	**O que informa**
Energia	Joule (J), watt-hora (Wh) ou quilowatt-hora (kWh)	Consumo acumulado no período e nos componentes observados.
Potência	Watt (W)	Taxa instantânea ou média de transferência de energia.
Carga elétrica	Ampère-hora (Ah), miliampère-hora (mAh) ou microampère-hora (µAh)	Quantidade de carga; converter em energia exige informação sobre a tensão.
Desempenho	Milissegundo (ms), segundo (s) ou operações por segundo	Tempo de resposta, duração ou quantidade de trabalho concluído.
Utilização	Ocupação do processador em %; memória em bytes	Uso de recursos; não informa energia isoladamente.

Fonte: elaboração própria, com base nas grandezas e no planejamento de medição discutidos por Guldner *et al.* (2024).

A avaliação de eficiência também precisa definir o trabalho útil: quantas requisições foram concluídas, quais operações foram executadas e quais requisitos foram atendidos. Consumir menos porque se entregou menos serviço não demonstra maior eficiência para a mesma tarefa. Guldner *et al.* (2024) propõem organizar a medição a partir do cenário, do ambiente e dos recursos observados. Seguindo essa preocupação, esta revisão evita usar tempo de

execução ou ocupação do processador como substitutos de uma avaliação de energia.

2.3. Condições para Comparar os Estudos

Esta revisão utiliza três condições de comparação, aqui denominadas correspondências da intervenção, da tarefa e da medição. Elas são uma forma de organizar a leitura das evidências; não constituem uma escala validada de qualidade nem uma fórmula para prever ganhos.

Intervenção: o que foi modificado? É preciso identificar as operações alteradas, as mudanças realizadas em conjunto e o comportamento preservado. Uma troca de coleção não é equivalente à extração de um método apenas porque ambas reorganizam o código. Os resultados de Hasan *et al.* (2016) e Noureddine e Rajan (2015) mostram a importância de descrever a mudança concreta.

Tarefa: o que o programa executou? A carga de trabalho compreende entradas, operações e frequência de uso. Um benchmark é uma tarefa preparada para avaliar um sistema; um microbenchmark concentra essa avaliação em operações pequenas. Ele pode ajudar a estudar um mecanismo, mas seu resultado não representa automaticamente o aplicativo completo. O contraste entre Li e Halfond (2014) e Sahin, Pollock e Clause (2016) será discutido sob essa perspectiva.

Medição: quais componentes e qual intervalo foram observados? A fronteira de medição delimita se o resultado se refere ao processador, a um conjunto de componentes, ao aparelho completo ou a uma parcela estimada por modelo. A janela de observação

delimita o período incluído, que pode abranger inicialização, espera e encerramento. Resultados na mesma unidade ainda podem cobrir objetos distintos (Hindle, 2016; Raffin; Trystram, 2025).

Por fim, diferença estatística e importância prática não são sinônimos. O valor-p indica a compatibilidade dos dados com um modelo estatístico e suas hipóteses; não informa a probabilidade de a refatoração economizar energia. Seu resultado deve ser lido junto à magnitude da diferença, à dispersão e ao desenho experimental (Wasserstein; Lazar, 2016). Nas tabelas, α indica o limiar usado pelos autores para a decisão estatística. A expressão "não foi detectada diferença" não afirma equivalência energética entre as versões. Essa distinção orienta a leitura de Verdecchia *et al.* (2018), Ournani *et al.* (2021) e dos demais experimentos.

3. PROCEDIMENTOS DA REVISÃO

3.1. Tipo de Revisão e Questões

O trabalho é uma revisão narrativa crítica, com busca dirigida e seleção intencional de fontes pertinentes. O objetivo é explicar resultados e limitações de um conjunto de estudos, sem estimar a frequência de benefícios em toda a literatura. A análise foi orientada por três questões:

- **Q1.** Quais alterações de código foram avaliadas quanto ao efeito energético e em quais plataformas?

- **Q2.** Como a energia foi medida ou estimada, e quais componentes foram incluídos?

- **Q3.** Quais conclusões são sustentadas pelas tarefas e condições efetivamente avaliadas?

3.2. Busca e Seleção das Fontes

A busca dirigida e a conferência documental foram concluídas em 14 de novembro de 2026. As consultas utilizaram tema, título, autoria e identificador digital de objeto (DOI, do inglês Digital Object Identifier). Foram utilizados termos em português e inglês relacionados a refatoração, code smells, consumo de energia e coleções Java, sem fixar um ano inicial de publicação. Exemplos de consultas registradas são "refactoring code smells energy consumption experiment", "Android energy refactoring performance" e "Java collections energy consumption".

As buscas localizaram páginas de editoras, registros bibliográficos, repositórios institucionais, manuscritos disponibilizados por autores e depósitos de pré-publicações, como o arXiv. Também foram seguidas referências de trabalhos pertinentes, incluindo as relações entre Cruz e Abreu (2017), Li e Halfond (2014) e Sahin, Pollock e Clause (2016). Os registros da busca, os arquivos consultados e os localizadores de resultados foram reunidos em fichas de apoio.

Não se aplicou uma estratégia única a todas as fontes, nem se realizou busca exaustiva em bases de indexação. A seleção procurou reunir intervenções e contextos distintos, incluindo reduções, aumentos e resultados sem diferença detectada. Essa escolha permite discutir contrastes, mas não torna a amostra representativa de toda a produção científica. Houve apoio de inteligência artificial (IA) à busca, à organização das fichas, à conferência documental e à

redação. Esse apoio não constitui seleção ou avaliação independente por dois pesquisadores.

3.3. Critérios de Inclusão e Exclusão

O núcleo de análise reúne estudos em português ou inglês que descrevem uma alteração identificável no código, comparam condições e avaliam energia por medição ou estimativa declarada. Foi exigido acesso ao relato completo do experimento, admitindo-se versões dos autores e relatos completos dos estudos em tese identificada. A descrição de instrumento, tarefa, repetições e validação funcional foi examinada criticamente; informações ausentes foram registradas como limitações, sem serem preenchidas por suposição.

Mudanças exclusivamente de infraestrutura, materiais promocionais e duplicatas não integram o núcleo. Resultados apenas de tempo ou utilização de recursos não foram convertidos em energia. Revisões e mapeamentos fornecem contexto, sem recontar seus estudos como novas observações. O artigo de Oliveira *et al.* (2021) representa a extensão da mesma linha de trabalho publicada em 2019; as duas versões não foram contadas como replicações independentes.

Três estudos foram mantidos como contextuais: Li e Halfond (2014), por apresentar carga elétrica; Sehgal *et al.* (2022), por enfatizar potência estimada; e Rua e Saraiva (2024), por observar a evolução de aplicativos com mudanças de diferentes tipos entre versões. Eles ajudam a discutir os limites de interpretação, mas não são equiparados aos experimentos de intervenção energética do núcleo.

3.4. Estudos Analisados e Função das Demais Referências

O conjunto comparativo contém 21 artigos: 18 estudos centrais, identificados nas Tabelas 2 e 3, e os três estudos contextuais descritos na seção anterior. O Apêndice A relaciona cada artigo à sua contribuição e às seções em que é utilizado. As referências conceituais, metodológicas e auxiliares são discriminadas segundo sua função na revisão.

Os estudos de Sahin, Pollock e Clause (2014, 2016) foram examinados nos relatos completos apresentados nas seções 4.4 e 4.6 da tese de Sahin (2017). Os respectivos arquivos editoriais não foram obtidos; portanto, não se afirma ter conferido sua redação integral. Anwar, Pfahl e Srirama (2019), Misu *et al.* (2025) e Raffin e Trystram (2025) foram consultados em versões depositadas pelos autores. Wang, Li e Tan (2026) é uma pré-publicação, ou preprint, e recebe essa identificação na interpretação dos resultados.

Imran *et al.* (2024) foi acessado por resumo e metadados. É utilizado para indicar a presença do tema em sistemas de nuvem, fora dos 21 artigos comparativos e sem extração experimental detalhada. O acesso a Poy *et al.* (2025) e Fleischle *et al.* (2026) foi parcial, limitando seu uso à descrição de escopo. A consulta de Connolly Bree e Ó Cinnéide (2025) foi dirigida às tabelas de avaliação e à discussão disponíveis, sem alegação de leitura contínua de todas as páginas.

Fowler (2004, 2018) e Martin (2008) fundamentam conceitos. Hindle (2016), Guldner *et al.* (2024) e Raffin e Trystram (2025) apoiam a análise da medição; Wasserstein e Lazar (2016), a interpretação estatística. As revisões citadas e Souza, Souza e Nunes (2025) situam o tema. Cruz *et al.* (2019), Xu *et al.* (2022) e Teixeira (2025) são referências auxiliares para discutir manutenção, preservação do

serviço e desempenho; não aumentam a contagem do conjunto comparativo.

3.5. Extração e Análise Crítica

Para cada estudo, foram registrados amostra, linguagem, alteração de código, instrumento, ambiente, tarefa, repetições, análise estatística, componentes medidos e limitações. A leitura distinguiu projetos, versões e execuções, assim como a amostra geral e o subconjunto realmente submetido à intervenção. A Tabela 3 informa os principais localizadores usados na conferência dos procedimentos e resultados.

As evidências foram organizadas por mecanismos e condições de avaliação, comparando convergências e divergências entre os trabalhos. Foram examinadas as correspondências da intervenção, da tarefa e da medição, sem atribuir uma nota agregada de qualidade. A possível reutilização de aplicativos e tarefas entre publicações foi considerada ao discutir a diversidade do conjunto.

Valores publicados foram confrontados com tabelas, figuras ou seções das versões consultadas. Cálculos realizados nesta revisão a partir de valores publicados e inconsistências internas das fontes são identificados no texto. "Não informado" significa que a informação não foi localizada no relato consultado; acesso apenas ao resumo não permite concluir que ela inexista no artigo completo.

Não houve reprodução de experimentos, reanálise de dados brutos, seleção independente por dois revisores ou metanálise. Percentuais de tarefas, plataformas e componentes diferentes não foram combinados. As conclusões representam uma interpretação crítica

das evidências disponíveis, e não uma estimativa única do efeito de refatorar.

4. SÍNTESE DAS EVIDÊNCIAS

4.1. Organização dos Estudos Comparados

Os estudos comparados investigam desde a reorganização de classes e métodos até a substituição de estruturas de dados e a liberação de recursos do dispositivo. Em comum, procuram verificar se uma alteração no código modifica o consumo necessário à execução. Entretanto, avaliam tarefas e partes do equipamento diferentes, o que exige interpretar cada resultado em seu contexto.

O núcleo contém 18 estudos com intervenções e energia medida ou estimada. Três artigos complementam a comparação: Li e Halfond (2014), com resultados em carga elétrica; Sehgal *et al.* (2022), com potência estimada; e Rua e Saraiva (2024), com associações entre mudanças de versões e consumo. Esses 21 artigos não equivalem a 21 experimentos causais independentes. Imran *et al.* (2024), consultado parcialmente, permanece fora dessa contagem.

A Tabela 2 apresenta intervenções, resultados e limites; a Tabela 3 reúne as condições de execução e medição. Nas subseções seguintes, os estudos são aproximados pelos mecanismos examinados e pelas condições que explicam convergências ou restringem comparações. Os percentuais não são agregados em uma economia média.

Tabela 2. Contexto, intervenções, resultados e limites dos 18 estudos centrais.

Estudo	Amostra e intervenção	Resultado e limite
Manotas, Pollock e Clause (2014)	Sete sistemas Java; substituições individuais e combinadas de coleções pela implementação SEEDSapi do Software Engineer's Energy-optimization Decision Support framework (SEEDS), sistema de apoio à escolha de otimizações energéticas.	Seis sistemas: reduções significativas de 2%–17%. Em sete de doze variantes, a mudança individual supera a combinação. O mesmo conjunto de testes orienta busca e avaliação; gasto energético da busca não incluído nas reduções.
Sahin, Pollock e Clause (2014)	Nove sistemas Java; 197 versões modificadas por seis tipos de refatoração, além de nove originais, avaliadas em duas máquinas virtuais Java, ambientes que executam o código dessa linguagem.	Dos 394 contrastes versão/plataforma, 109 são significativos: 56 aumentos e 53 reduções. Não são projetos independentes. Seleção manual, cobertura parcial e um hardware limitam aplicação a outros sistemas; relato consultado na tese.
Noureddine e Rajan (2015)	Onze pequenos exemplos C++ com os padrões de projeto Decorator e Observer; redução da criação de objetos e de consultas repetidas.	Reduções médias do custo energético adicional: 12,23% e 6,95%, respectivamente. Energia do processador estimada, exemplos mínimos e ausência de análise inferencial. Não demonstra economia da instalação inteira nem vantagem geral de eliminar padrões.
Pereira *et al.* (2016)	Cinco soluções estudantis do mesmo sistema; substituições de coleções orientadas por operações.	Reduções descritivas de 4,37%–11,05%; projeto 3, troca da lista encadeada LinkedList pela lista ArrayList, apoiada em arranjo redimensionável: 25,024 → 22,259 J. Teste inferencial não

		informado. Operação, tamanho e requisitos de comportamento condicionam escolha; amostra não representa cinco domínios industriais.
Hasan *et al.* (2016)	Seis sistemas Java em Android; trocas de listas favoráveis ou deliberadamente desfavoráveis.	Gson aumenta 309%; simulador reduz 38%; outros efeitos são menores. Gson inclui mudanças nos testes. Conjuntos de testes parcialmente compatíveis e uma plataforma limitam aplicação a outros sistemas; energia maior não implica potência proporcionalmente maior.
Sahin, Pollock e Clause (2016)	Oito aplicativos Android, 17 cenários e 32 versões modificadas, além de oito versões de referência; quatro recomendações de desempenho.	Três de 136 contrastes significativos, todos no Galaxy S5; nenhum no Nexus 4. Maior autonomia projetada: aproximadamente 1%. Versão de referência preparada, cobertura parcial e multiplicidade sem ajuste declarado limitam inferência.
Banerjee e Roychoudhury (2016)	Dez aplicativos Android; liberação de sensores, sistema de posicionamento global (GPS, Global Positioning System) e wakelocks, bloqueios que mantêm recursos ativos.	Reduções relatadas de 3%–29% frente ao original, sem teste inferencial informado. Janela inclui espera e pós-encerramento. Algumas alterações envolvem qualidade de serviço, ultrapassando refatoração estritamente preservadora do comportamento observável.
Cruz e Abreu (2017)	Seis aplicativos Android; oito práticas indicadas pelo Lint, ferramenta de análise de código, aplicadas	WriteilyPro, reutilização de referências da interface (ViewHolder): −4,50%; corrigir o redesenho da mesma área (Overdraw) aumenta dois aplicativos em cerca de 2%;

	individualmente e em conjunto.	outras práticas sem diferença detectada. Processador, unidade gráfica e memória não representam aparelho inteiro. Preparação, redesenhos e tarefa condicionam o balanço.
Morales *et al.* (2018)	Energy-Aware Refactoring approach for MObile apps (EARMO): busca de refatorações que considera energia; oito antipadrões em vinte aplicativos Android, com validação energética posterior em treze.	Reduções significativas em três dos treze. Previsões e medições divergem: Tapsoffire, 3,30 versus 6,59 J. Os 29,71 minutos adicionais de words são autonomia extrapolada, não descarga observada.
Verdecchia *et al.* (2018)	Três aplicações Java com mapeamento objeto-relacional, que relaciona objetos a dados de bancos relacionais; cinco tipos de problemas tratados isoladamente e em combinações.	JTrac: diferença global significativa entre tratamentos; maior redução mediana de 49,9%. CashManager, dependência excessiva de dados de outra classe (Feature Envy): aumento descritivo de 4,2% da mediana, sem significância global. Combinação não preserva melhor redução isolada. Estudo piloto não demonstra regressão generalizável.
Anwar, Pfahl e Srirama (2019)	Três aplicativos Android; cinco smells clássicos, tratamentos individuais e permutados, com rastros.	Calculator, remoção de código duplicado (Duplicate Code): −10,8%; Openflood tem aumentos descritivos, sem significância global. Ausência de diferença temporal significativa não prova equivalência. Poucos aplicativos, um aparelho e cenários específicos restringem aplicação a outros sistemas.

Palomba *et al.* (2019)	Sessenta aplicativos Android; remoção de smells específicos em métodos.	Reduções significativas em quatro tipos: acesso interno por métodos de escrita (Internal Setter), tarefas não encerradas (Leaking Thread), métodos que ignoram o estado do objeto (Member Ignoring Method) e laços classificados como lentos (Slow Loop). Quatro tipos sem efeito observado. Medianas divergem no texto e na tabela da fonte. Estimativa por método não equivale a energia do aplicativo.
Couto, Saraiva e Fernandes (2020)	609 projetos examinados, 239 com padrões; 44 condições refatoráveis, incluindo padrões isolados e combinações, avaliadas com Chimera.	22 ligações positivas e dezesseis negativas entre intervenções individuais e combinações. Negativa significa benefício inferior ao individual, não necessariamente aumento contra original. Cobertura limitada; 416 resultados não são projetos independentes.
Oliveira *et al.* (2021)	CT+, ferramenta de recomendação de coleções Java: 17 sistemas, 64 versões; escolhas orientadas por perfil e dispositivo.	Joda-Time/Dell: −6,66%; Cassandra/Kafka sem diferenças significativas nos seis perfis. Graphchi depende do perfil, com discrepância textual no máximo. Família 2019/2021 conta uma vez. Testes não garantem preservação do comportamento geral.
Ournani *et al.* (2021)	Sete projetos Java; histórico de 16.046 registros de alterações (commits), filtrado por construção, cobertura e mudanças.	Validações enfraquecem atribuição à refatoração. Possível efeito remanescente de 2% convive com desvio-padrão de 5%. Histórico não equivale a pares medidos; 25 regras recebem filtragem adicional.

		Conclusão restrita às funcionalidades avaliadas.
Şanlıalp, Öztürk e Yiğit (2022)	Dois programas 2048, C# e Java/Android; cinco transformações e combinações, versões de referência adaptadas.	Android, combinação selecionada: 184,71→168,46 J; redução calculada de 8,80%, divergente dos 9,33% publicados. Computador, outra combinação: −4,83%. Sequência da tarefa insuficientemente descrita e inconsistências limitam inferência; não sustenta ranking entre linguagens.
Misu *et al.* (2025)	13.103 testes de doze projetos Java; intervenção em 66 e 79 testes do Commons Lang.	Reduções calculadas dos valores de consumo reportados: 9,7% e 9,3%. Distinguir esses contrastes do rateio por linhas de código dentro de testes com smells. Um projeto e fronteira do processador limitam aplicação a outros sistemas.
Wang, Li e Tan (2026)	Preprint: 64 itens/61 tipos e 481 commits de 430 projetos Java.	29/64 itens mudam de classificação; 10/64 invertem sinal significativo. Apenas 36/481 commits têm diferença significativa. Proporções dos casos elegíveis; múltiplos contrastes, um ambiente e mudanças combinadas limitam aplicação a outros sistemas e atribuição.

Fonte: elaboração própria com base nos estudos citados.

Tabela 3. Instrumentação, tarefa, repetição e análise dos 18 estudos centrais.

Estudo	**Instrumento e componentes**	**Tarefa e repetições**	**Análise e localização na**

			fonte
Manotas, Pollock e Clause (2014)	Plataforma Low Power Energy Aware Processing (LEAP), com sensores; Intel Atom/XUbuntu, equipamento USB-6215 a 10 mil coletas por segundo. Integra potência de componentes em energia; coleta em outra máquina.	Conjuntos de testes dos sistemas; dez execuções finais por versão. A mesma tarefa orienta a busca e a avaliação das variantes escolhidas.	Kruskal–Wallis, α=0,05; diferenças entre médias. Intervalo de confiança não informado. §§3.2–4.3, Tabelas 2–4, PDF p.7–10.
Sahin, Pollock e Clause (2014)	LEAP físico em plataforma x86; energia dos componentes instrumentados. Máquina virtual Java 6 e máquina virtual Java 7, mesmo hardware.	Conjuntos de testes dos nove sistemas; 25 execuções por versão/plataforma. Transformações selecionadas manualmente; cobertura limita caminhos avaliados.	Mann–Whitney–Wilcoxon, α=0,05. Relato apresentado na tese de Sahin (2017), §4.4, p.26–45; Tabela 4.4, p.38. Arquivo editorial não acessado.
Noureddine e Rajan (2015)	JOLINAR 2, ferramenta baseada em modelos: energia do processador estimada. ThinkPad X220/i5-2540M, Clang 3.5/OpenJDK 1.7.0_65. Não mede toda a máquina.	Sondagem: 100 mil iterações; transformação: um milhão; dez repetições. Saídas comparadas textualmente, sem prova formal de equivalência.	Sem teste inferencial, desvio-padrão ou intervalo de confiança informados. §§II–III; Figuras 3–4; páginas impressas 624–626, PDF p.3–5.

Pereira *et al.* (2016)	jRAPL, interface Java para Running Average Power Limit (RAPL), que fornece contadores de energia de componentes. Energia do processador e tempo; i3-3240, Linux, Java 1.8.0_66. Não mede toda a instalação.	Perfis por operação/tamanho; sete cenários nas aplicações. Aquecimento prévio e dez execuções, retirando os 20% maiores e 20% menores valores.	Média restante; teste de hipótese, desvio-padrão e intervalo de confiança das aplicações não informados. §§2.2 e 3.1–3.4; Tabelas 2–3, p.16–19.
Hasan *et al.* (2016)	GreenMiner/INA219, energia do aparelho Android; aplicativo de teste JUnit com tela vazia. Modelo e versão Android não informados.	Perfis: treze tamanhos, vinte repetições e repetições internas. Validação com conjuntos de testes parcialmente compatíveis.	Intervalo de confiança de 95% nos perfis; intervalo de confiança, dispersão e p-valores por aplicação não informados. §3 e §6; Tabela 2, PDF p.2–3 e 9–10.
Sahin, Pollock e Clause (2016)	KORAD/Arduino/INA219, aproximadamente mil coletas por segundo; integra corrente×tensão. Nexus 4/Dalvik e Galaxy S5/Android Runtime (ART), ambiente de execução Android; energia do aparelho.	Dezessete cenários reproduzidos pela ferramenta RERAN, 19–273 s; trinta execuções por condição. Reproduções de interações distintas entre aparelhos; comparação interna a cada plataforma.	Mann–Whitney–Wilcoxon e A12; α=0,05; ajuste múltiplo não declarado. Tese, §4.6, p.74–89; Figura 4.15/Tabela 4.15.

Banerjee e Roychoudhury (2016)	Monsoon alimenta Samsung S4/Android 4.4.2 a 4,2 volts; energia do telefone, incluindo recursos ativos durante esperas.	MonkeyRunner; 15 s entre eventos e 45 s após abrir/fechar. Janela inclui pós-encerramento; repetições não informadas.	Desvio-padrão, intervalo de confiança e teste de hipótese não informados na avaliação. §§4.1–4.2 e Tabela 2, PDF p.7–8; qualidade de serviço discutida em §3.1.
Cruz e Abreu (2017)	Sensores ODROID-XU/Android 4.2.2; integração trapezoidal da potência dos processadores principal e secundário, da unidade gráfica e da memória. Exclui tela, GPS e rede.	Roteiros de interface, tarefas repetidas 10–200 vezes; trinta execuções intercaladas por versão. Após descarte, n=28–30.	Shapiro–Wilk, Welch bilateral, Cohen d; α=0,05; exclusão além de média ± dois desvios-padrão. Intervalo de confiança/ajuste múltiplo não informados. Tabelas III–V, p.52–54.
Morales *et al.* (2018)	TiePie HS5/uCurrent, 125 mil coletas por segundo; Nexus 4/Android 5.1.1. Instrumentação física do telefone; procedimento descrito associa amostras a métodos e filtra os externos ao código da aplicação.	Cenários roteirizados; trinta execuções. Treze aplicativos na validação posterior, subconjunto dos vinte avaliados pela abordagem.	Mann–Whitney e Cliff's delta. §3.2, p.7: atribuição e filtragem por métodos; §5.4, p.19–22: validação e Tabela 16. Autonomia adicional projetada por repetição de cenários.

Verdecchia *et al.* (2018)	Wattsup PRO, servidor HP DL360 G7; ambientes virtualizados Windows Server/vSphere. Energia do servidor físico em Wh, sem isolar o trecho modificado.	LoadComplete reproduz cenários de navegador; 215 amostras totais distribuídas em 70/70/75. Comparação por aplicação e tratamento.	Análise de variância e Cliff's delta; α=0,05. §6.1 e Tabela 7, p.375 impressa/p.12 do PDF. Teste global entre tratamentos; testes posteriores dos pares não apresentados.
Anwar, Pfahl e Srirama (2019)	Monsoon a 5 mil coletas por segundo; LG Spirit Y70/Android 5.0.1. Energia do dispositivo com subtração do consumo ocioso.	Cenários Espresso e inspeção de rastros para confirmar execução do código alterado; dez execuções por condição.	Análise de variância/Cliff's delta; resultados temporais também testados. §III, p.3–4; Tabelas V–VI, p.5–6 do depósito arXiv. Ausência de significância temporal não demonstra equivalência.
Palomba *et al.* (2019)	Power Estimation Tool for Android (PETrA): estima energia por período de execução/método usando registros e perfis. Nexus 4/Android 5.1.1; não mede fisicamente cada método.	Monkey, dez execuções de aproximadamente cinco minutos; mesmos cenários após intervenção. Cobertura não garante todos os caminhos alterados.	Mann–Whitney/Cliff's delta. Métodos em §2; resultados em §3.3, p.51 editorial. Narrativa e Tabela 3 divergem nas descritivas; medianas não usadas como magnitudes inequívocas.

Couto, Saraiva e Fernandes (2020)	Qualcomm Trepn em três Nexus 5/Android 6.0.1; energia reportada no dispositivo, sem instrumentação externa de referência.	25 cenários de 25 eventos MonkeyRunner; sequências idênticas antes/depois, no mesmo aparelho. Aquecimento prévio e resfriamento de cinco segundos.	Mann–Whitney e Cohen d, comparando intervenções/ combinações. §§VI.A–C, PDF p.7–10. Cenários distintos não são repetições idênticas; cobertura aleatória é limitação declarada.
Oliveira *et al.* (2021)	Computadores: jRAPL/RAPL, domínios somados não explicitados; Dell, servidor Xeon e Asus. Dispositivos móveis: modelo Android Power Profile, fronteira diferente.	Conjuntos de testes e programas que acionam os sistemas; trinta execuções por versão. Na construção dos perfis de operações, aquecimento de até 10% da carga; experimento adicional varia esses perfis.	Shapiro–Wilk, Wilcoxon–Mann–Whitney, Cliff's delta; α=0,05, sem retirar valores atípicos. Tabelas 4/6, p.19/23; §6.2/Figura 2, p.30–35; intervalo de confiança não informado.
Ournani *et al.* (2021)	RAPL, energia global de processador e memória dinâmica; i7-6600U/Ubuntu 18.04, OpenJDK 8 ou 11. Atividade subjacente integra a observação.	Pequenos testes com Java Microbenchmark Harness (JMH) e cobertura por Java Code Coverage (JaCoCo) de funcionalidades selecionadas; 100–1.000 medidas, reamostragem estatística com reposição (bootstrap).	Wilcoxon na validação dos candidatos; inspeção/ microbenchmarks investigam atribuição. §3.2, Tabelas 2–3, p.40–43. Limiar inicial de 5% não é tamanho mínimo universal de efeito.

		Commits filtrados por construção, cobertura e mudanças coexistentes.	
Şanlıalp, Öztürk e Yiğit (2022)	Intel Power Gadget/Windows e Trepn/Android; energia calculada por potência média×tempo. Canais/domínios agregados insuficientement e explicitados.	Dez execuções de 300 s por condição; versões de referência adaptadas. Sequência de jogadas e igualdade do trabalho útil não demonstradas.	Wilcoxon pareado, α=0,05; médias e desvio-padrão. Justificativa do pareamento, intervalo de confiança e correção múltipla não informados. §§3.2–3.4, Tabelas 5/8, p.7–11.
Misu *et al.* (2025)	Intel PowerLog 3.7.0, cinco MacBook Air; energia do pacote do processador. Rateio posterior por linhas de código não mede trechos.	13.103 testes, cinco execuções/teste; mediana e resfriamento de 30 s. Intervenções repetem o protocolo no Commons Lang.	Welch/Cohen d; Kendall para correlações. Bonferroni α=0,002 restrito à segunda questão do estudo. §§3.3–3.5 e 4.1.1–4.1.3, p.5–11 do preprint.

Wang, Li e Tan (2026)	JoularJX/powermetrics em Mac mini M4 Pro/macOS/kit de desenvolvimento Java 25; não RAPL. Domínios físicos agregados não discriminados. Registro de execução separado da coleta energética.	Cinco cargas mais agregado; conjuntos de testes nos commits. Trinta medições pareadas, ordem aleatória, aquecimento e resfriamento; trabalho fixado dentro do par.	Wilcoxon pareado/Cliff's delta; p<0,05. Benjamini-Hochberg em associações, não genericamente nas classificações energéticas. §§3.3/4.1/5.1, p.14–17 e 22–27.

Fonte: elaboração própria com base nos estudos citados. Os números de seções, tabelas, figuras e páginas nesta coluna remetem aos estudos originais. Cohen d, delta de Cliff e A12 são medidas de tamanho de efeito; não são valores-p.

4.2. Operações, Coleções e Padrões de Projeto

Uma primeira distinção separa melhorar a organização do código de reduzir o trabalho executado. Ao avaliar seis refatorações em nove sistemas Java, Sahin, Pollock e Clause (2014) encontraram 109 diferenças significativas entre 394 comparações de versão e ambiente: 56 aumentos e 53 reduções. Verdecchia *et al.* (2018), por sua vez, observaram resultados diferentes entre três aplicações Java em servidor. A convergência está na ausência de vantagem energética garantida pelo nome da refatoração; cada ocorrência modifica um programa e caminhos de execução específicos.

O segundo estudo também esclarece por que economias individuais não podem ser somadas. No JTrac, os tratamentos separados de Feature Envy, uso excessivo de dados de outra classe, e

Long Method, método considerado excessivamente longo, reduziram a mediana de 11,68 para 5,85 Wh: 49,9% em cada caso. A combinação de cinco tipos reduziu apenas 10,7%. No CashManager, o tratamento de Feature Envy apresentou aumento descritivo de 4,2%, mas a análise global não detectou diferença entre tratamentos. No JTrac, a significância foi global entre tratamentos; não foi informado teste posterior que confirme cada contraste individual. A comparação sustenta dependência do sistema, mas não demonstra que uma transformação idêntica produziu efeitos contrários comprovados (Verdecchia *et al.*, 2018).

As substituições de coleções oferecem alterações mais diretamente relacionadas às operações executadas. Manotas, Pollock e Clause (2014) buscaram implementações alternativas com o SEEDS e obtiveram reduções significativas de 2% a 17% em seis de sete sistemas. Pereira *et al.* (2016) aplicaram recomendações baseadas nas operações de coleções a cinco soluções estudantis do mesmo problema; em uma delas, trocar LinkedList por ArrayList, duas implementações de listas, alterou a média de 25,024 para 22,259 J. Os resultados mostram oportunidades concretas, mas dependem das operações e dos tamanhos avaliados. Em Pereira *et al.* não foi informado teste de significância para os resultados das aplicações.

Nesse grupo também aparecem combinações inferiores às melhores escolhas isoladas. Isso ocorreu em sete das doze variantes bem-sucedidas do SEEDS, aproximando o resultado de Manotas, Pollock e Clause (2014) daquele de Verdecchia *et al.* (2018). Há ainda uma restrição de avaliação: os testes usados pelo SEEDS orientam a busca e avaliam a versão escolhida. Passar nesses testes não demonstra economia em outra tarefa nem preservação de todos os

requisitos, como ordem dos elementos e acesso simultâneo aos dados.

Trocas de listas também podem aumentar o consumo. Hasan *et al.* (2016) executaram testes Java em Android e observaram aumento de 309% no Gson após uma troca deliberadamente desfavorável, enquanto a troca inversa reduziu um simulador de bolsa em 38%. No Gson, parte das alterações atingiu os próprios testes; portanto, o resultado não se restringe ao código de produção. Potência semelhante e execução mais longa explicam a diferença entre aumento de energia e aumento de potência. Em contraste com esses extremos, Oliveira *et al.* (2021), ao recomendar coleções em 17 sistemas com o CT+, não detectaram diferenças significativas em Cassandra e Kafka nos seis perfis examinados. O conjunto não estabelece uma lista universalmente mais econômica.

Diminuir operações redundantes é outro mecanismo possível. Noureddine e Rajan (2015) reduziram construções de objetos no padrão Decorator e consultas repetidas ao estado no padrão Observer. Em onze pequenos exemplos C++, as reduções médias do custo energético adicional associado aos padrões foram 12,23% e 6,95%, respectivamente. São estimativas de energia do processador, não economias de toda a máquina. O estudo identifica operações que podem ser evitadas, sem demonstrar vantagem geral de eliminar padrões de projeto, isto é, soluções recorrentes de organização do software.

A escala dos exemplos e a definição da tarefa também limitam os resultados favoráveis de Şanlıalp, Öztürk e Yiğit (2022), em dois jogos 2048, um em C# e outro em Java. Uma combinação Android passou de 184,71 para 168,46 J; a redução calculada é 8,80%, divergindo dos

9,33% publicados. As medições duraram 300 segundos, mas a sequência de jogadas é insuficientemente descrita. Assim como nos exemplos de padrões, o resultado precisa permanecer ligado ao programa avaliado; não permite comparar a eficiência geral das linguagens.

Esses experimentos contrastam com o exame da evolução de projetos por Ournani *et al.* (2021). Após filtrar um histórico de 16.046 alterações registradas em sete projetos Java, os autores não identificaram efeito substancial atribuível à refatoração nos casos exercitados. Um possível efeito de aproximadamente 2% coexistia com desvio-padrão de 5%. O histórico inteiro não corresponde ao número de pares medidos. A divergência em relação aos ganhos anteriores não prova incompatibilidade: alterações selecionadas do desenvolvimento corrente e transformações escolhidas para testes controlados respondem a comparações diferentes.

4.3. Tarefas e Recursos Físicos em Android

Em dispositivos móveis, reduzir o custo de uma operação não garante redução equivalente durante o uso do aplicativo. Li e Halfond (2014) encontraram ganhos em pequenos programas que repetem operações selecionadas, chamados microbenchmarks. Armazenar previamente o comprimento de um vetor reduziu em 10% o valor observado. Esse resultado é expresso em mAh, unidade de carga elétrica, e não pode ser convertido diretamente em joules sem informação de tensão. Já Sahin, Pollock e Clause (2016), ao aplicar quatro recomendações de desempenho a oito aplicativos e tarefas interativas, detectaram diferenças significativas em apenas três de 136 comparações. O contraste sugere que a participação da

operação alterada na tarefa importa, embora os estudos não sejam replicações controladas entre si.

O nível da medição também muda a interpretação. Palomba *et al.* (2019) estimaram energia de métodos em 60 aplicativos e relataram reduções após remover quatro dos oito tipos de problemas examinados. Entre eles estavam acessos internos por métodos de escrita de atributos e tarefas de execução que permaneciam ativas indevidamente. A ferramenta PETrA atribui consumo a partir de registros da execução e perfis; não mede fisicamente cada método. Isso impede projetar automaticamente os ganhos locais sobre o aplicativo inteiro. Além disso, divergências entre o texto e uma tabela do estudo restringem o uso de algumas medianas como valores inequívocos.

Morales *et al.* (2018) acrescentam a comparação entre previsão e medição. Sua abordagem EARMO busca refatorações considerando energia e qualidade estrutural; dos vinte aplicativos estudados, treze receberam validação energética posterior e três apresentaram reduções significativas. No Tapsoffire, a previsão de 3,30 J contrasta com 6,59 J medidos. A coleta usa instrumentação física, mas o procedimento descrito associa as amostras a chamadas de métodos e filtra os externos ao código da aplicação. A validação posterior remete a esse procedimento; seus valores não devem ser tratados automaticamente como consumo total do aparelho durante toda a tarefa. O resultado complementa Palomba *et al.* (2019): estimar consumo pode orientar a escolha, mas não substitui a confirmação na escala de interesse. Os 29,71 minutos adicionais projetados para words também não representam uma descarga completa observada. Sahin, Pollock e Clause (2016) projetaram ganho máximo

de autonomia de aproximadamente 1%, igualmente condicionado à repetição do cenário.

As diferenças entre tarefas reaparecem nas correções de práticas Android. Anwar, Pfahl e Srirama (2019) observaram reduções medianas de aproximadamente 10,8% e 10,5% no Calculator ao tratar código duplicado e verificações de tipo. No Openflood, entretanto, os aumentos descritivos coexistiram com resultado global não significativo. Cruz e Abreu (2017) também encontraram efeitos distintos: ViewHolder, que reutiliza referências a elementos da interface, reduziu o consumo do WriteilyPro de 119,7 para 114,3 J; corrigir Overdraw, desenhos repetidos sobre uma mesma área, aumentou em cerca de 2% o consumo de dois aplicativos. Seus dados abrangem processador, unidade gráfica e memória, sem tela e rádio. Os resultados reforçam a necessidade de testar a prática na tarefa e nos componentes efetivamente observados.

Essa cautela alcança as combinações. Couto, Saraiva e Fernandes (2020) examinaram padrões em 239 de 609 projetos e avaliaram 44 condições refatoráveis, incluindo padrões isolados e combinações. Das 38 relações significativas entre alterações individuais e combinadas, dezesseis indicaram benefício combinado inferior ao individual. Isso não significa necessariamente aumento frente à versão original. A interpretação converge com SEEDS e JTrac: a versão final precisa ser medida, porque reunir escolhas promissoras não preserva obrigatoriamente a melhor economia isolada (Manotas; Pollock; Clause, 2014; Verdecchia *et al.*, 2018).

Um mecanismo diferente envolve o tempo em que componentes físicos permanecem ativos. Banerjee e Roychoudhury (2016) trataram a ativação e liberação de localização por GPS, sensores e

wakelocks, bloqueios que impedem o dispositivo de entrar em repouso. Relataram reduções de 3% a 29% em dez aplicativos, sem informar repetições, dispersão ou teste de significância nessa avaliação. A janela inclui espera e período posterior ao encerramento, quando recursos não liberados continuam consumindo. Parte das mudanças também altera precisão ou frequência de atualização; nesse caso, muda a qualidade do serviço, não apenas a organização interna do código. Os ganhos não são diretamente comparáveis aos de tarefas que encerram a medição ao terminar a computação.

4.4. Sistemas em Nuvem e Acesso à Evidência

Os resultados anteriores não permitem concluir como as mesmas alterações se comportariam em serviços distribuídos, nos quais comunicação e infraestrutura participam da execução. Imran *et al.* (2024) abordam refatoração orientada a energia em sistemas de nuvem, mas somente resumo e metadados estiveram disponíveis nesta consulta. Sua contribuição é indicar esse contexto de investigação. Sem o texto experimental, não foi possível verificar instrumentos, tarefas, repetições ou percentuais; esses detalhes não sustentam a comparação. A restrição pertence ao acesso obtido, não demonstra deficiência da publicação nem ausência de pesquisas sobre nuvem.

4.5. Código de Testes e Atribuição de Energia

O código usado para avaliar um programa também consome energia e pode ser objeto de manutenção. Isso aproxima dois resultados diferentes: Hasan *et al.* (2016) mostram que alterar coleções nos testes interfere na interpretação do consumo atribuído

à aplicação; Misu *et al.* (2025) investigam diretamente a refatoração dos próprios testes. Neste segundo caso, a economia durante a verificação do software é o desfecho de interesse, e não deve ser apresentada como redução durante seu uso normal.

Misu *et al.* (2025) analisaram 13.103 testes de doze projetos e concentraram as intervenções no Commons Lang. Em subconjuntos de 66 e 79 testes, os valores de consumo reportados passaram, respectivamente, de 204,922 para 184,974 J e de 204,048 para 185,017 J. As reduções calculadas desses valores são 9,7% e 9,3%, com diferenças significativas nas análises relatadas. O procedimento usa medianas de cinco repetições por teste, mas não identifica expressamente esses quatro agregados como médias. A concentração em um projeto limita a transferência para os demais.

É necessário separar essa comparação executada do rateio feito pelos mesmos autores: parte do consumo foi distribuída entre linhas com e sem problemas em proporção ao número de linhas. Os valores resultantes não são medições isoladas desses trechos nem comparação entre duas populações independentes de testes. Como a ferramenta observa o conjunto de componentes do processador, a atribuição ao código depende desse procedimento adicional. Essa distinção preserva o resultado da intervenção sem conferir ao rateio uma precisão que ele não possui.

4.6. Carga de Trabalho e Evolução de Versões

Dois desenhos ajudam a entender por que a quantidade e o tipo de trabalho alteram as conclusões. Oliveira *et al.* (2021) variam os perfis, conjuntos de operações usados para orientar a escolha de coleções. Wang, Li e Tan (2026) variam o tamanho da entrada do mesmo item

de refatoração. No CT+, Graphchi apresentou reduções diferentes conforme o perfil; para o perfil de carga dobrada, identificado como N2 no estudo, o texto informa 12,73%, enquanto a figura apresenta 16,34%. A divergência restringe o uso do máximo, mas não elimina a dependência do perfil observada no experimento.

No estudo de Wang, Li e Tan (2026), ainda em pré-publicação, 29 de 64 itens mudaram de classificação energética entre tamanhos de entrada; dez apresentaram aumento significativo em uma condição e redução significativa em outra. Na parte baseada em alterações reais, foram detectadas diferenças em 36 de 481 comparações, abrangendo quinze aumentos e 21 reduções. As medições usam JoularJX com powermetrics em macOS, e não RAPL. As diferenças entre as duas partes não isolam o efeito da carga, pois projetos, mudanças e tarefas também diferem. Tampouco uma comparação não significativa demonstra equivalência energética.

Quantidade de trabalho não deve ser confundida com duração. Sehgal *et al.* (2022), mantidos como evidência contextual, relataram menor potência estimada nos dezoito programas após refatoração, em execuções de quinze minutos. O valor-p de 0,077562 atende ao limiar de 10% adotado pelos autores, mas não ao de 5%. Além disso, igual duração não comprova igual serviço concluído. Os resultados em watts não devem ser tratados como joules por tarefa. Essa limitação se aproxima da descrição insuficiente da sequência de jogadas nos experimentos de Şanlıalp, Öztürk e Yiğit (2022).

Por fim, observar muitas versões amplia os contextos, mas não separa necessariamente as causas. Rua e Saraiva (2024) estudaram 215 aplicativos e 1.322 versões. Nas 44 transições selecionadas para análise, refatorações coocorreram com treze de 24 aumentos e dez

de vinte quedas de energia. São grupos definidos pelo resultado, não probabilidades de piorar ou melhorar ao refatorar. Mudanças de funcionalidades, interface e dependências coexistem. A contribuição contextual consiste em mostrar essa sobreposição, complementando os estudos de intervenções selecionadas sem substituí-los por uma estimativa causal de maior escala.

4.7. Síntese Pelas Três Correspondências

As comparações sustentam três perguntas articuladas. Primeiro, o que mudou na execução? Trocar uma coleção, reduzir consultas, liberar um sensor e reorganizar métodos não são intervenções equivalentes. Segundo, qual trabalho foi realizado? Repetir uma operação isolada, usar uma interface e executar testes distribuem os custos de maneiras diferentes. Terceiro, o que foi medido? Energia do processador, do aparelho ou atribuída a métodos tem alcances distintos; potência e carga elétrica representam outras grandezas.

A Tabela 4 organiza essas correspondências e os limites que acompanham cada interpretação.

Tabela 4. Condições para comparar e aplicar os resultados

Correspon-dência	**Condição de leitura**	**Exemplos e limites**
O que mudou? Comportamento preservado	Identificar operações alteradas e serviço preservado; nome de smell ou interface comum não garante mudança operacional equivalente.	Coleções em Manotas, Pollock e Clause (2014), Pereira *et al.* (2016) e Oliveira *et al.* (2021) exigem requisitos de ordenação/concorrência. Banerjee e Roychoudhury (2016) incluem alterações de qualidade de serviço.

O que mudou? Combinações e causas	Separar efeito individual, combinação e mudanças coexistentes; não somar ganhos nem tratar associação como intervenção isolada.	SEEDS apresenta escolhas individuais superiores às combinações. Couto, Saraiva e Fernandes (2020) comparam combinação ao individual. Rua e Saraiva (2024) observam refatorações junto a outras mudanças, sem identificar efeito causal próprio.
Qual tarefa? Operações e frequência	Comparar entradas, operações e frequência; benefício local pode ser diluído por outros custos da tarefa.	Li e Halfond (2014) concentram microprogramas na operação. Sahin, Pollock e Clause (2016) usam tarefas interativas. Wang, Li e Tan (2026) variam entradas; Oliveira *et al.* (2021), perfis usados para recomendar coleções.
Qual tarefa? Período e trabalho concluído	Duração igual não assegura serviço equivalente; explicitar inicialização, espera, pós-encerramento e cobertura.	Sehgal *et al.* (2022) e Şanlıalp, Öztürk e Yiğit (2022) usam janelas fixas com trabalho pouco especificado. Banerjee e Roychoudhury (2016) incluem espera e pós-encerramento para observar recursos não liberados.
O que foi medido? Grandeza e componentes	Distinguir energia, potência e carga; joules em fronteiras diferentes também não são diretamente intercambiáveis.	Li e Halfond (2014): mAh; Sehgal *et al.* (2022): W. Cruz e Abreu (2017): processador, unidade gráfica e memória. Hasan *et al.* (2016): aparelho. Noureddine e Rajan (2015): energia do processador estimada e custo energético adicional dos padrões.
O que foi medido? Estimativa e incerteza	Declarar modelo, comparador, repetições e família de testes; ausência de significância não prova equivalência.	Palomba *et al.* (2019) estimam por método; Misu *et al.* (2025) rateiam por linhas de código. Morales *et al.* (2018) distinguem previsão e medição. CashManager em Verdecchia *et*

		al. (2018) tem aumento descritivo, não regressão significativa.

Fonte: elaboração própria com base nos estudos citados.

A convergência entre os estudos está na necessidade de descrever operações, preservar o serviço comparado e avaliar a versão resultante. As divergências indicam condições a investigar, sem autorizar escolher apenas resultados favoráveis ou tratar toda variação como efeito demonstrado. Há economias específicas, aumentos e diferenças não detectadas. Nenhuma dessas classes, isoladamente, estabelece uma regra geral sobre refatorar. Para orientar manutenção, o resultado precisa permanecer ligado à transformação, à tarefa e à medição que o produziram.

5. DISCUSSÃO

5.1. O Que Explica a Diversidade dos Resultados

O conjunto examinado sugere que o efeito energético se relaciona às operações efetivamente alteradas e executadas, e não apenas à classificação da mudança como refatoração. Estudos de coleções mostram que substituir uma implementação modifica custos de acesso, inserção e outras operações, cujo peso depende do uso da estrutura. Os resultados são condicionados à combinação entre implementação, operações e volume de dados (Manotas; Pollock; Clause, 2014; Pereira *et al.*, 2016; Hasan *et al.*, 2016; Oliveira *et al.*, 2021). A interpretação desta revisão é que um nome comum de transformação não constitui evidência suficiente para recomendar o mesmo resultado em outro sistema.

A tarefa ajuda a compreender por que ganhos locais podem perder relevância na aplicação. A operação modificada pode representar uma parcela pequena da execução, coexistindo com espera, acesso a recursos e outras atividades. O contraste entre microprogramas e tarefas interativas é compatível com essa explicação, mas não a isola, pois os estudos também diferem em aparelho e procedimento de avaliação (Li; Halfond, 2014; Sahin; Pollock; Clause, 2016). A variação controlada das entradas em Wang, Li e Tan (2026) oferece evidência mais direta de dependência da carga, restrita aos itens avaliados.

Outra convergência é a ausência de garantia de aditividade: combinar alterações favoráveis não preserva necessariamente o melhor resultado individual. Uma combinação pode economizar menos que uma mudança isolada e ainda consumir menos que a versão original. Por isso, a interpretação exige explicitar os dois comparadores e avaliar a composição final (Manotas; Pollock; Clause, 2014; Couto; Saraiva; Fernandes, 2020).

5.2. O Que a Medição Permite Concluir

Uma diferença energética só pode ser interpretada em relação ao que foi observado. Energia de componentes, energia do aparelho e estimativa atribuída a um método cobrem objetos distintos. A mensuração do processador, da unidade gráfica e da memória, por exemplo, não inclui necessariamente tela e comunicação; uma estimativa por método depende do procedimento que distribui o consumo entre trechos (Cruz; Abreu, 2017; Palomba *et al.*, 2019). O planejamento da medição deve explicitar componentes, período de observação e tarefa, como propõem Hindle (2016) e Guldner *et al.* (2024).

Essa exigência impede tratar carga elétrica em mAh, potência em watts e energia em joules como resultados intercambiáveis. Também impede interpretar o rateio por linhas de código de Misu *et al.* (2025) como medição independente de cada trecho. Esses resultados podem contribuir para a discussão, desde que sua natureza permaneça explícita. O nome da ferramenta não basta para identificar a fronteira: os componentes cobertos dependem do equipamento e da configuração empregados (Raffin; Trystram, 2025).

A força da conclusão depende ainda da análise das diferenças. Um aumento descritivo sem significância, como no CashManager, não sustenta a mesma afirmação que uma diferença estatisticamente detectada; a ausência de significância, por sua vez, não demonstra consumo idêntico (Verdecchia *et al.*, 2018). Tamanho da diferença, variabilidade e procedimento estatístico precisam ser examinados conjuntamente (Wasserstein; Lazar, 2016). Divergências entre valores publicados, identificadas em Palomba *et al.* (2019) e Şanlıalp, Öztürk e Yiğit (2022), também restringem a precisão das magnitudes reproduzidas, mesmo quando a direção do resultado é relatada pelos autores.

5.3. Como Distinguir o Efeito da Refatoração

O contraste entre versões não identifica automaticamente a causa de uma diferença. Na evolução de aplicativos, refatorações podem acompanhar novas funcionalidades, alterações de interface e atualizações de dependências. A coexistência dessas mudanças amplia a descrição do desenvolvimento real, mas impede atribuir o resultado exclusivamente à refatoração (Rua; Saraiva, 2024). A mesma restrição aparece no estudo auxiliar de Teixeira (2025):

otimizações e reorganizações são realizadas em etapas, porém reunidas na avaliação final de desempenho, sem mensuração energética. A ordem das mudanças não substitui uma comparação que separe seus efeitos.

Também é necessário verificar o serviço entregue. Reduzir precisão, frequência de atualização ou funcionalidade pode diminuir o consumo sem tornar mais eficiente a execução da mesma tarefa. Banerjee e Roychoudhury (2016) admitem escolhas de qualidade de serviço em parte das intervenções; Xu *et al.* (2022), em pesquisa auxiliar sobre parâmetros, descartam oportunidades aparentes após verificar comportamento e funcionamento. Nesta revisão, esses casos delimitam a definição de refatoração e mostram que uma redução energética precisa ser interpretada junto aos requisitos preservados.

5.4. Contribuição em Relação às Revisões Existentes

A discussão das condições experimentais não é exclusiva deste trabalho. Connolly Bree e Ó Cinnéide (2025) examinam a qualidade dos estudos e as inconsistências dos resultados; Souza, Souza e Nunes (2025) organizam técnicas e ferramentas associadas à eficiência energética. As sínteses de Poy *et al.* (2025) e Fleischle *et al.* (2026) situam recortes mais amplos, embora o acesso parcial obtido nesta consulta limite uma comparação detalhada de seus procedimentos.

A contribuição aqui proposta é organizar os contrastes pelas correspondências entre intervenção, tarefa e medição. Esses critérios ajudam a distinguir resultados aparentemente contraditórios e a explicitar quais condições precisariam ser verificadas antes de

aplicar uma conclusão a outro sistema. Trata-se de uma organização interpretativa da evidência selecionada, não de um modelo validado para prever economia nem de uma classificação geral da qualidade dos estudos.

5.5. Implicações para Manutenção e Pesquisa

Para a manutenção, uma expectativa de economia deve ser tratada como hipótese a verificar. A equipe precisa descrever a transformação, selecionar uma tarefa representativa, verificar o comportamento preservado e comparar a energia necessária para concluir o mesmo trabalho. Em sequências de mudanças, deve avaliar também a versão final. Essas recomendações decorrem da dependência da tarefa e das interações observadas nos estudos, e não de uma garantia de benefício para qualquer refatoração (Manotas; Pollock; Clause, 2014; Couto; Saraiva; Fernandes, 2020; Oliveira *et al.*, 2021).

Qualidade e energia exigem avaliações próprias. Cruz *et al.* (2019) analisam métricas estáticas de manutenibilidade em alterações orientadas a energia, sem confirmar experimentalmente a economia dessas alterações nem medir diretamente o esforço humano de manutenção. Seu uso como estudo auxiliar não autoriza estabelecer uma troca quantitativa entre facilidade de manutenção e joules. Uma mudança motivada por clareza pode ser justificada por esse objetivo, enquanto seus efeitos energéticos permanecem sujeitos a avaliação.

Para pesquisas futuras, são pertinentes comparações que mantenham a transformação e variem controladamente entradas, equipamento ou ambiente de execução. Avaliar tarefas diferentes

das utilizadas para escolher a versão otimizada também ajudaria a examinar a generalização. Relatos conjuntos de energia, comportamento, desempenho e manutenção permitiriam distinguir benefícios e custos sem presumir que esses atributos variem sempre na mesma direção.

6. LIMITAÇÕES DA REVISÃO

Esta revisão narrativa utiliza busca dirigida e seleção intencional; não pretende recuperar exaustivamente a literatura nem estimar a frequência de resultados favoráveis. O conjunto comparativo contém 18 estudos centrais com intervenções e avaliação energética e três contextuais: Li e Halfond (2014), com carga elétrica; Sehgal *et al.* (2022), com potência estimada; e Rua e Saraiva (2024), com observações da evolução de versões. Não houve seleção independente por dois revisores. Idiomas, disponibilidade dos textos e encadeamentos de referências condicionam a cobertura.

Os estudos concentram-se em Java e Android, poucas plataformas e tarefas delimitadas. Os exemplos em C++ e C# ampliam o recorte, mas não eliminam essas restrições. Algumas publicações reutilizam aplicações e cenários, e extensões de um trabalho não constituem replicações independentes. A diversidade de unidades, componentes observados e versões de referência impede agregar os percentuais como uma economia média da refatoração.

Os relatos de Sahin, Pollock e Clause (2014, 2016) foram consultados na tese do autor, sem cotejo integral com os artigos editoriais. Há também manuscritos de autores e pré-publicações; Wang, Li e Tan (2026) permanece nessa condição. Imran *et al.* (2024) foi consultado apenas em resumo e metadados, fora dos 21 estudos comparativos.

O acesso a algumas revisões foi parcial ou dirigido, conforme registrado na metodologia.

Não foram reproduzidos experimentos nem reanalisados dados brutos. A conferência documental não confirma a coleta original e não resolve todas as inconsistências de relato. O viés de publicação não foi quantificado. Assim, as conclusões não determinam a probabilidade de economia em um sistema escolhido ao acaso nem o benefício ambiental de todo o ciclo de desenvolvimento.

7. CONCLUSÃO

As evidências selecionadas indicam que refatorações e remoções de code smells podem alterar o consumo de energia, com reduções, aumentos e diferenças não detectadas nas condições examinadas. O conjunto não sustenta previsão de economia apenas pelo tipo ou pela quantidade de problemas estruturais removidos. A interpretação depende das operações modificadas, da tarefa que as executa e dos componentes incluídos na avaliação.

A revisão contribui ao tornar essas condições explícitas por meio das correspondências entre intervenção, tarefa e medição. Para decisões de manutenção, isso significa verificar se a versão alterada entrega o mesmo serviço e avaliar sua energia no contexto relevante. Benefícios de compreensão, manutenção e desempenho permanecem objetivos próprios; sua demonstração não substitui a avaliação energética. Uma recomendação defensável deve indicar o resultado observado e os limites dentro dos quais ele pode orientar outras intervenções.

DECLARAÇÕES

Conflito de interesses. Os autores declaram não haver conflito de interesses.

Contribuição dos autores. Curti e Cardia realizaram a busca bibliográfica, a extração de dados e a redação do manuscrito. Oliveira orientou o desenho da revisão e revisou criticamente o texto.

Financiamento. A pesquisa não recebeu financiamento específico.

Disponibilidade de dados. Os dados analisados provêm integralmente dos estudos publicados relacionados nas referências. As Tabelas 2 e 3 e o Apêndice A apresentam a base documental da síntese. Fichas de extração e registros de busca foram mantidos como material de trabalho. Não foram produzidos novos dados experimentais.

REFERÊNCIAS BIBLIOGRÁFICAS

ANWAR, H.; PFAHL, D.; SRIRAMA, S. N. Evaluating the Impact of Code Smell Refactoring on the Energy Consumption of Android Applications. In: **Proceedings of the 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA).** Kallithea-Chalkidiki: IEEE, 2019. p. 82-86. DOI: 10.1109/SEAA.2019.00021.

BANERJEE, A.; ROYCHOUDHURY, A. Automated re-factoring of Android apps to enhance energy-efficiency. In: **Proceedings of the International Conference on Mobile Software Engineering and Systems (MOBILESoft).** New York: ACM, 2016. p. 139-150. DOI: 10.1145/2897073.2897086.

CONNOLLY BREE, D.; Ó CINNÉIDE, M. How Software Design Affects Energy Performance: A Systematic Literature Review. **Journal of Software: Evolution and Process,** v. 37, n. 4, e70014, 2025. DOI: 10.1002/smr.70014.

COUTO, M.; SARAIVA, J.; FERNANDES, J. P. Energy Refactorings for Android in the Large and in the Wild. In: **Proceedings of the 2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER).** London, ON: IEEE, 2020. p. 217-228. DOI: 10.1109/SANER48275.2020.9054858.

CRUZ, L.; ABREU, R. Performance-based Guidelines for Energy Efficient Mobile Applications. In: **Proceedings of the 2017 IEEE/ACM International Conference on Mobile Software Engineering and Systems (MOBILESoft).** IEEE, 2017. p. 46-57. DOI: 10.1109/MOBILESoft.2017.19.

CRUZ, L.; ABREU, R.; GRUNDY, J.; LI, L.; XIA, X. Do Energy-oriented Changes Hinder Maintainability? In: **Proceedings of the 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME).** IEEE, 2019. p. 29-40. DOI: 10.1109/ICSME.2019.00013.

FLEISCHLE, A.-C.; MARKOVA, Z.; WITTGES, H.; KRCMAR, H. Programming Practices Influencing Software Energy Efficiency: A Systematic Literature Review. In: **Advances and New Trends in Environmental Informatics.** Cham: Springer, 2026. p. 187-207. DOI: 10.1007/978-3-032-22726-3_11.

FOWLER, M. **Definition of Refactoring.** 2004. Disponível em: https://martinfowler.com/bliki/DefinitionOfRefactoring.html. Acesso em: 14 nov. 2026.

FOWLER, M. **Refactoring: Improving the Design of Existing Code.** 2. ed. Boston: Addison-Wesley, 2018. Disponível em: https://www.martinfowler.com/books/refactoring.html. Acesso em: 14 nov. 2026.

GULDNER, A. *et al.* Development and evaluation of a reference measurement model for assessing the resource and energy efficiency of software products and components - Green Software Measurement Model (GSMM). **Future Generation Computer Systems,** v. 155, p. 402-418, 2024. DOI: 10.1016/j.future.2024.01.033.

HASAN, S.; KING, Z.; HAFIZ, M.; SAYAGH, M.; ADAMS, B.; HINDLE, A. Energy profiles of Java collections classes. In: **Proceedings of the 38th International Conference on Software Engineering (ICSE).** New York: ACM, 2016. p. 225-236. DOI: 10.1145/2884781.2884869.

HINDLE, A. Green Software Engineering: The Curse of Methodology. In: **Proceedings of the 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER).** Osaka: IEEE, 2016. p. 46-55. DOI: 10.1109/SANER.2016.60.

IMRAN, A.; KOSAR, T.; ZOLA, J.; BULUT, M. F. Towards Sustainable Cloud Software Systems through Energy-Aware Code Smell Refactoring. In: **Proceedings of the 2024 IEEE 17th International Conference on Cloud Computing (CLOUD).** Shenzhen: IEEE, 2024. p. 223-234. DOI: 10.1109/CLOUD62652.2024.00034.

LI, D.; HALFOND, W. G. J. An investigation into energy-saving programming practices for Android smartphone app development. In: **Proceedings of the 3rd International Workshop on Green and Sustainable Software (GREENS).** New York: ACM, 2014. p. 46-53. DOI: 10.1145/2593743.2593750.

MANOTAS, I.; POLLOCK, L.; CLAUSE, J. SEEDS: a software engineer's energy-optimization decision support framework. In: **Proceedings of the 36th International Conference on Software Engineering (ICSE).** New York: ACM, 2014. p. 503-514. DOI: 10.1145/2568225.2568297.

MARTIN, R. C. **Clean Code: A Handbook of Agile Software Craftsmanship.** Upper Saddle River: Prentice Hall, 2008. Disponível em: https://www.pearson.com/en-us/subject-catalog/p/redirected-product/P200000009044/9780132350884. Acesso em: 14 nov. 2026.

MISU, M. R. H. *et al.* Test smell: A parasitic energy consumer in software testing. **Information and Software Technology,** v. 181, 107671, 2025. DOI: 10.1016/j.infsof.2025.107671.

MORALES, R.; SABORIDO, R.; KHOMH, F.; CHICANO, F.; ANTONIOL, G. EARMO: An Energy-Aware Refactoring Approach for Mobile Apps. **IEEE Transactions on Software Engineering,** v. 44, n. 12, p. 1176-1206, 2018. DOI: 10.1109/TSE.2017.2757486.

NOUREDDINE, A.; RAJAN, A. Optimising energy consumption of design patterns. In: **Proceedings of the 2015 IEEE/ACM 37th International Conference on Software Engineering (ICSE).** IEEE, 2015. v. 2, p. 623-626. DOI: 10.1109/ICSE.2015.208.

OLIVEIRA, W.; OLIVEIRA, R.; CASTOR, F.; PINTO, G.; FERNANDES, J. P. Improving energy-efficiency by recommending Java collections. **Empirical Software Engineering,** v. 26, n. 3, art. 55, 2021. DOI: 10.1007/s10664-021-09950-y.

OURNANI, Z.; ROUVOY, R.; RUST, P.; PENHOAT, J. Tales from the Code #1: The Effective Impact of Code Refactorings on Software Energy Consumption. In: **Proceedings of the 16th International**

Conference on Software Technologies (ICSOFT). Lisboa: SCITEPRESS, 2021. p. 34-46. DOI: 10.5220/0010517900340046.

PALOMBA, F.; DI NUCCI, D.; PANICHELLA, A.; ZAIDMAN, A.; DE LUCIA, A. On the impact of code smells on the energy consumption of mobile applications. **Information and Software Technology,** v. 105, p. 43-55, 2019. DOI: 10.1016/j.infsof.2018.08.004.

PEREIRA, R.; COUTO, M.; SARAIVA, J.; CUNHA, J.; FERNANDES, J. P. The influence of the Java collection framework on overall energy consumption. In: **Proceedings of the 5th International Workshop on Green and Sustainable Software (GREENS).** New York: ACM, 2016. p. 15-21. DOI: 10.1145/2896967.2896968.

POY, O.; MORAGA, M. Á.; GARCÍA, F.; CALERO, C. Impact on energy consumption of design patterns, code smells and refactoring techniques: A systematic mapping study. **Journal of Systems and Software,** v. 222, 112303, 2025. DOI: 10.1016/j.jss.2024.112303.

RAFFIN, G.; TRYSTRAM, D. Dissecting the Software-Based Measurement of CPU Energy Consumption: A Comparative Analysis. **IEEE Transactions on Parallel and Distributed Systems,** v. 36, n. 1, p. 96-107, 2025. DOI: 10.1109/TPDS.2024.3492336.

RUA, R.; SARAIVA, J. A large-scale empirical study on mobile performance: energy, run-time and memory. **Empirical Software Engineering,** v. 29, n. 1, art. 31, 2024. DOI: 10.1007/s10664-023-10391-y.

SAHIN, C. **Empirically Investigating Energy Impacts of Software Engineering Decisions.** 2017. Tese (Doutorado em Ciência da Computação) - University of Delaware, 2017. Disponível em:

https://udspace.udel.edu/server/api/core/bitstreams/39cb4335-178c-4e57-ad3d-e08173f35c0b/content. Acesso em: 14 nov. 2026.

SAHIN, C.; POLLOCK, L.; CLAUSE, J. From benchmarks to real apps: Exploring the energy impacts of performance-directed changes. **Journal of Systems and Software,** v. 117, p. 307-316, 2016. DOI: 10.1016/j.jss.2016.03.031.

SAHIN, C.; POLLOCK, L.; CLAUSE, J. How do code refactorings affect energy usage? In: **Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '14).** Torino: ACM, 2014. Art. 36, p. 1-10. DOI: 10.1145/2652524.2652538.

ŞANLIALP, İ.; ÖZTÜRK, M. M.; YİĞİT, T. Energy Efficiency Analysis of Code Refactoring Techniques for Green and Sustainable Software in Portable Devices. **Electronics,** v. 11, n. 3, art. 442, 2022. DOI: 10.3390/electronics11030442.

SEHGAL, R.; MEHROTRA, D.; NAGPAL, R.; SHARMA, R. Green software: Refactoring approach. **Journal of King Saud University - Computer and Information Sciences,** v. 34, n. 7, p. 4635-4643, 2022. DOI: 10.1016/j.jksuci.2020.10.022.

SOUZA, A. de M. e; SOUZA, P. C. de; NUNES, E. P. dos S. Desenvolvimento Sustentável de Software e Eficiência Energética do Código: Um Mapeamento Sistemático. In: **Anais da I Escola Regional de Sistemas de Informação de Mato Grosso.** Cuiabá: Sociedade Brasileira de Computação, 2025. p. 102-110. DOI: 10.5753/ersimt.2025.8884.

TEIXEIRA, H. P. **Análise de desempenho, otimização e refatoração do servidor Radar Livre.** 2025. Trabalho de Conclusão de Curso (Graduação em Engenharia de Software) - Universidade Federal do Ceará, Quixadá, 2025. Disponível em: https://repositorio.ufc.br/handle/riufc/80581. Acesso em: 14 nov. 2026.

VERDECCHIA, R.; SAEZ, R. A.; PROCACCIANTI, G.; LAGO, P. Empirical Evaluation of the Energy Impact of Refactoring Code Smells. In: **ICT4S2018: Proceedings of the 5th International Conference on Information and Communication Technology for Sustainability.** Toronto: EasyChair, 2018. p. 365-383. (EPiC Series in Computing, v. 52). DOI: 10.29007/dz83.

WANG, H.; LI, H.; TAN, S. H. **Understanding the Energy Impact of Software Refactoring: A Workload-Aware Study of Controlled Examples and Real-World Commits.** 2026. Pré-publicação. arXiv:2608.06620v1. Disponível em: https://arxiv.org/abs/2608.06620v1. Acesso em: 14 nov. 2026.

WASSERSTEIN, R. L.; LAZAR, N. A. The ASA's Statement on p-Values: Context, Process, and Purpose. **The American Statistician,** v. 70, n. 2, p. 129-133, 2016. DOI: 10.1080/00031305.2016.1154108.

XU, Q.; DAVIS, J. C.; HU, Y. C.; JINDAL, A. An Empirical Study on the Impact of Deep Parameters on Mobile App Energy Usage. In: **Proceedings of the 2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER).** IEEE, 2022. p. 844-855. DOI: 10.1109/SANER53432.2022.00103.

APÊNDICE A — Estudos e contribuições

O Quadro A1 relaciona os 21 artigos do conjunto comparativo: 18 estudos centrais de intervenção com avaliação energética e três contextuais. As seções indicam os principais trechos de análise.

Quadro A1 - Contribuição e uso dos 21 artigos do conjunto comparativo

Estudo e função	**Contribuição efetivamente utilizada**	**Seções de uso**
Manotas, Pollock e Clause (2014) Núcleo	Trocas de coleções orientadas por busca mostram que implementação e combinações precisam ser avaliadas na tarefa executada.	4.2; 4.3; 5.1; 5.5
Sahin, Pollock e Clause (2014) Núcleo	Distingue o efeito de instâncias concretas de refatoração da expectativa baseada apenas no nome da transformação. Relato completo consultado na tese de Sahin (2017).	4.2
Noureddine e Rajan (2015) Núcleo	Transformações de Decorator e Observer permitem discutir custos de criação de objetos e chamadas, preservando a distinção entre padrão e sua implementação.	4.2
Pereira *et al.* (2016) Núcleo	Relaciona perfis de operações de coleções à escolha de implementações em programas completos.	4.2; 5.1
Hasan *et al.* (2016) Núcleo	Mostra que operação, posição e tamanho das coleções alteram a direção do resultado; inclui variantes deliberadamente desfavoráveis.	4.2; 4.5; 5.1
Sahin, Pollock e Clause (2016) Núcleo	Contrasta recomendações de desempenho com tarefas interativas e evidencia a dificuldade de transportar ganhos locais para o uso de aplicativos.	4.3; 5.1

	Relato completo consultado na tese de Sahin (2017).	
Banerjee e Roychoudhury (2016) Núcleo	Relaciona energia à duração de ativação e à liberação de recursos físicos, inclusive após o encerramento da aplicação.	4.3; 5.3
Cruz e Abreu (2017) Núcleo	Confronta práticas de desempenho com resultados favoráveis, adversos e sem diferença detectada segundo aplicação e cenário.	4.3; 5.2
Morales *et al.* (2018) Núcleo	A comparação entre previsão e validação física permite separar a seleção de refatorações do benefício efetivamente observado.	4.3
Verdecchia *et al.* (2018) Núcleo	Permite discutir dependência do sistema e efeitos de combinações, distinguindo diferenças estatísticas de variações descritivas.	4.2; 4.3; 5.2
Anwar, Pfahl e Srirama (2019) Núcleo	Confronta cinco tipos de smells em aplicações Android e mostra que tipos e aplicações não compartilham necessariamente o mesmo resultado.	4.3
Palomba *et al.* (2019) Núcleo	Permite analisar refatorações na granularidade de métodos e explicitar o alcance de uma estimativa atribuída a trechos.	4.3; 5.2
Couto, Saraiva e Fernandes (2020) Núcleo	Mostra que combinar mudanças não preserva necessariamente a melhor redução individual e exige declarar o comparador.	4.3; 5.1; 5.5
Oliveira *et al.* (2021) Núcleo	Recomendações de coleções relacionam resultados a perfil, dispositivo, tipo de coleção e condições de concorrência.	4.2; 4.6; 5.1; 5.5
Ournani *et al.* (2021)	A filtragem de commits e a inspeção de resultados delimitam a atribuição causal	4.2

Núcleo	de pequenas diferenças a refatorações.	
Şanlıalp, Öztürk e Yiğit (2022) Núcleo	Avalia alterações isoladas e combinadas em C# e Java; permite discutir a definição da tarefa, a duração fixa e a inconsistência entre médias e percentuais publicados.	4.2; 4.6; 5.2
Misu *et al.* (2025) Núcleo	Distingue associações sobre smells de teste de intervenções antes/depois e da atribuição de consumo por linhas.	4.5; 5.2
Wang, Li e Tan (2026) Núcleo	A manipulação da entrada permite discutir sensibilidade à carga e contraste entre exemplos controlados e commits reais. Pré-publicação.	4.6; 5.1
Li e Halfond (2014) Contextual	Práticas locais em microbenchmarks ajudam a contrastar exposição intensa à operação com tarefas interativas.	4.3; 5.1; 5.2
Sehgal *et al.* (2022) Contextual	Oferece contraponto de remoção de smells com potência estimada, útil para separar watts de energia por tarefa.	4.6
Rua e Saraiva (2024) Contextual	Amplia a observação da evolução de aplicativos e evidencia que coocorrência de refatoração e mudança energética não isola causa.	4.6; 5.3

Fonte: elaboração própria. As seções indicam os trechos principais de análise, além das citações de identificação na metodologia.

O Quadro A2 apresenta quatro estudos auxiliares, que não integram o conjunto comparativo. Os trabalhos conceituais, as revisões e as referências metodológicas têm suas funções descritas na Seção 3.4.

Quadro A2 - Função dos estudos auxiliares

Estudo e função	**Contribuição efetivamente utilizada**	**Seções de uso**

Imran *et al.* (2024) Auxiliar	Situa a existência de pesquisa sobre refatoração energética em sistemas de nuvem, sem sustentar a extração de resultados experimentais detalhados. Acesso limitado a resumo e metadados.	4.4
Cruz *et al.* (2019) Auxiliar	Exige avaliar manutenibilidade como desfecho próprio, separando métrica estática de economia energética confirmada.	2.1; 5.5
Xu *et al.* (2022) Auxiliar	A validação de funcionalidade elimina aparentes ganhos em ajustes de parâmetros e explicita a preservação do serviço.	5.3
Teixeira (2025) Auxiliar	Exemplifica a dificuldade de atribuir ganhos quando intervenções em etapas são reunidas no mesmo contraste antes/depois.	5.3

Fonte: elaboração própria. As seções indicam os trechos principais de análise, além das citações de identificação na metodologia.

A tese de Sahin (2017) é a fonte dos relatos de dois artigos e não é contada como um terceiro experimento. A extensão de Oliveira *et al.* (2021) representa a mesma família de pesquisa de 2019. Documentos de apoio e versões de um mesmo trabalho não são contados como estudos independentes.

[1] Curso de Engenharia de Software, Universidade de Vassouras, Vassouras, RJ, Brasil.

[2] Curso de Engenharia de Software, Universidade de Vassouras, Vassouras, RJ, Brasil.

[3] Curso de Engenharia de Software, Universidade de Vassouras, Vassouras, RJ, Brasil.