Ensino prático de manutenção de software: uma experiência com a dinâmica cliente-empresa

Miqueias Coelho[1], Lucas Reis[1], Bruno Silva[1], Rodrigo Nascimento[1], Davi Viana[1]

[1] Universidade Federal do Maranhão (UFMA)
São Luís – MA – Brasil

{miqueias.pereira, lucas.reis, bruno.carvalho1}@discente.ufma.br

rodrigo.nascimento@discente.ufma.br, davi.viana@ufma.br

Abstract. *Software maintenance activities are a significant part of the development process and are essential to ensuring proper software functioning. Therefore, it is necessary to adopt practical and flexible teaching methods for software engineering to teach future software engineers how to write and perform high-quality code maintenance. This paper presents an experience report on a practical approach to teaching software maintenance through client-company dynamics. Our sample involved 37 students from Federal University of Maranhão in Computer Engineering program, where we observed increased engagement in class participation. Knowledge was more deeply transmitted, as students organized themselves into groups and interacted with their peers' diverse experiences. Furthermore, we identify that students developed skills such as communication and code organization, in addition to increasing their knowledge and experience with software maintenance methods, tools like GitHub, and relational databases.*

Resumo. *A manutenção de software é uma parte significativa do processo de desenvolvimento, sendo essencial para garantir o funcionamento adequado do software. Portanto, é necessário adotar métodos de ensino práticos e flexíveis para Engenharia de Software a fim de ensinar os futuros profissionais a implementar manutenções de código com qualidade. Este artigo aborda um relato de experiência sobre uma abordagem prática de ensino para manutenção de software através da dinâmica cliente-empresa, visando compreender a relevância dessa abordagem para o aprendizado dos discentes. A amostra contou com 37 alunos da Universidade Federal do Maranhão do curso de Engenharia da Computação, resultando em aumento significativo no engajamento em sala de aula. O conhecimento foi transmitido de forma mais profunda à medida que os alunos se organizaram em grupos e interagiram com as diferentes experiências de seus colegas. Além disso, os discentes relataram desenvolvimento em habilidades como comunicação e organização de código, além do aumento de seu conhecimento e experiência sobre métodos de manutenção de software, ferramentas como GitHub e bancos de dados relacionais.*

1. INTRODUÇÃO

A manutenção de um software é uma atividade que ocorre ao final do ciclo de vida do software e pode gerar um impacto tanto em relação ao custo total do software, quanto ao

seu tempo de vida. Assim sendo, conhecimentos em manutenção de software revelam-se fundamentais para profissionais da computação. As atividades de manutenção de software são importantes para assegurar o correto funcionamento do sistema durante seu ciclo de vida, compondo cerca de 75% da força de trabalho no processo de desenvolvimento (MAGALHãES et al., 2020; LEVIN; YEHUDAI, 2019; SOMMERVILLE, 2015). Como exemplos, tem-se a adição de novas funcionalidades no software por meio de tarefas adaptativas, assim como tarefas evolutivas conforme as requisições de mudanças (do inglês, *change request*), ou por meio de tarefas corretivas a fim de corrigir defeitos expostos em relatórios de erros (LEVIN; YEHUDAI, 2019).

Deste modo, a adoção de métodos de ensino que simulem o ambiente do mercado de trabalho é essencial para a formação de profissionais mais conscientes e capacitados (CAVALCANTE et al., 2023). Destaca-se que na área de manutenção de software, o sucesso depende do desenvolvimento de habilidades que garantam adaptação a novas metodologias, ferramentas e tecnologias (OTEMAIER et al., 2024). Através da aplicação de estudos práticos, os alunos compreendem melhor os conceitos do tema abordado, sentem mais confiança em aplicá-los na prática, desenvolvem melhores práticas de codificação e desenvolvem habilidades em revisão de código com evolução do pensamento crítico (NASCIMENTO et al., 2025; ALOMAR; ALOMAR; MKAOUER, 2023). Assim sendo, o ensino de manutenção de software torna-se uma ferramenta indispensável para a formação de profissionais capazes não somente de desenvolver, mas também de manter o funcionamento dos softwares.

Neste contexto, este trabalho objetiva contribuir e apoiar o ensino de manutenção de software por meio de um relato de experiência. Aplicou-se uma abordagem prática de ensino simulando um ambiente de mercado de trabalho. Para tal, foi realizada uma dinâmica cliente-empresa na Universidade Federal do Maranhão, durante a disciplina de Engenharia de Software (ES) do curso de Engenharia da Computação. Esta abordagem visou preparar os estudantes frente a cenários de manutenção de software no mundo real. Efetivamente, 37 alunos participaram deste método de ensino, todos com conhecimentos básicos sobre o tema e a grande maioria possuía certa familiaridade sobre as ferramentas utilizadas. Mediante o *feedback* de 26 alunos da amostra, foi possível investigar como os alunos avaliaram a simulação de um ambiente real em comparação com aulas expositivas tradicionais. Observou-se uma receptividade expressiva sobre a abordagem utilizada, além de melhora significativa na compreensão e segurança dos alunos sobre o tema.

Este artigo organiza-se da seguinte forma: a Seção 2 apresenta os fundamentos teóricos; os trabalhos relacionados são abordados na Seção 3; a Seção 4 demonstra o método de pesquisa adotado; os resultados são apresentados na Seção 5; e, por fim, a Seção 6 apresenta as conclusões sobre este trabalho, assim como os trabalhos futuros.

2. FUNDAMENTAÇÃO TEÓRICA

À medida que a necessidade por software de qualidade cresce, a importância dos profissionais de Engenharia de Software aumenta na indústria (BARCELLOS et al., 2024). A ES enfatiza a necessidade de métodos, ferramentas e melhores práticas para garantir a qualidade do software, pois ele evoluirá ao longo do tempo, independentemente do domínio de sua aplicação, tamanho ou complexidade (PRESSMAN; MAXIM, 2016). Conforme Sneed e Brossler (2003), a manutenção de software é todo o trabalho realizado

em um produto de software após sua entrada em produção. Isso inclui correções, alterações, melhorias e otimizações, porém os autores ressaltam que deve haver um limite na taxa de aplicação de melhorias, a fim de distingui-las de um novo desenvolvimento que reutiliza partes do software atual.

Desta forma, o domínio da manutenção de software engloba modificações e aprimoramentos efetuados no software, extensões de suas funcionalidades e correção de defeitos (LIENTZ; SWANSON, 1980). Os autores classificam a manutenção de software em três tipos: Manutenção corretiva: com o objetivo de corrigir falhas (*bugs*), sejam elas emergenciais ou não; Manutenção adaptativa: refere-se à adequação do software ao ambiente. Atualizar serviços, fazer alterações conforme as exigências das empresas contratantes ou alterar processos de pagamento no software; e Manutenção perfectiva: diz respeito à adição de novos recursos ao software. É comum quando há necessidade de englobar novos requisitos ao software, devido a novos requisitos ou melhoria de desempenho.

Outro conceito presente nesta área é o da refatoração, sendo este, o processo de alterar um software de forma que não altere seu comportamento externo, porém melhore sua estruturação interna, ou seja, seu código (OPDYKE, 1992). O autor indica que refatorações não alteram o comportamento do sistema, isto é, mediante o mesmo conjunto de entradas, os valores de saída serão os mesmos antes da refatoração. A ideia é redistribuir classes, variáveis e métodos por meio de suas hierarquias para facilitar futuras adaptações e extensões (MENS; TOURWÉ, 2004). Embora esta prática não altere o comportamento do software, ela pode apoiar o projeto e a sua evolução do sistema, reestruturando um programa de forma a permitir que outras alterações sejam feitas com maior facilidade (OPDYKE, 1992).

Além destas definições, há também normativas sobre manutenção de software, por exemplo, a norma técnica ISO/IEC 12207. Esta é uma norma internacional que define um modelo de processos utilizados ao longo do ciclo de vida do software, englobando a manutenção de software como um de seus principais processos. Conforme a norma, o processo de manutenção é iniciado quando um software sofre modificações no código e na documentação associada devido a um erro, deficiência, problema, necessidade de melhoria, ou adaptação. O objetivo é modificar um software existente, preservando sua integridade de modo a concluir as modificações adequadamente (SINGH, 1996).

3. TRABALHOS RELACIONADOS

Em Lima e Rabelo (2023), utiliza-se a gamificação como ferramenta motivadora no ensino de manutenção de software, com foco em estudantes de ES e Ciência da Computação da Universidade Federal do Ceará (Campus Russas). O estudo detalha a implementação de um sistema gamificado, que inclui elementos como missões, pontos e *rankings*. A pesquisa analisou artefatos gerados pelos alunos e utilizou questionários como IMI e IMMS para avaliar a motivação. Os resultados preliminares indicam que 72% dos alunos demonstraram interesse nas atividades durante a gamificação e identificaram desafios comuns na elaboração de documentação de software.

O trabalho de Bezerra et al. (2024) apresenta uma prática para o ensino da refatoração de *code smells* em projetos de software de código aberto. A amostra englobou 29 estudantes de disciplinas de Qualidade e Manutenção de Software, analisando suas

percepções sobre as práticas de refatoração. Os autores identificaram as dificuldades enfrentadas, as técnicas de refatoração mais empregadas e o impacto dessas ações na qualidade interna do código. Além disso, o artigo destaca os benefícios pedagógicos e práticos da colaboração em projetos de código aberto como uma ferramenta eficaz para o aprendizado e a aplicação de conceitos de manutenção de software, preparando os alunos para os desafios da indústria.

Por sua vez, Gallagher, Fioravanti e Kozaitis (2019) descrevem uma abordagem para o ensino da manutenção de software a estudantes universitários em seu segundo ano (com três semestres de programação), transformando o projeto introdutório do curso em um exercício de evolução de software. Em vez de criar projetos do zero, os alunos trabalham com projetos de código aberto maduros e existentes, com histórico de revisões e *bugs*, simulando uma experiência realista. O curso utiliza uma metodologia de leitura e compreensão de programas complexos antes da escrita, enfatiza o trabalho em equipe e a avaliação é feita em grupo. Deste modo, a iniciativa individual e a comunicação são incentivadas. Os projetos envolvem a seleção e resolução de *issues* (*bugs* e melhorias) em bases de código de grande escala, com acompanhamento em laboratório e avaliações que focam tanto no desempenho técnico quanto na reflexão sobre os conceitos de ES.

Mediante os estudos apresentados, identifica-se que o ensino da manutenção de software é importante para o cenário tecnológico atual, porém carece de métodos de ensino com visões acadêmica e industrial. Assim, o diferencial da prática adotada é a mescla destas visões, focando em uma abordagem que fornece uma simulação do ambiente de mercado de trabalho, através da dinâmica cliente-empresa, a fim de contribuir para o ensino do tema.

4. MÉTODO DE PESQUISA

Esta seção apresenta o método de pesquisa (Figura 1) adotado para a aplicação de uma prática de ensino de manutenção de software mediante uma dinâmica simulada cliente-empresa.

Figura 1. Método de pesquisa aplicado ao longo da pesquisa.

O primeiro passo foi identificar e delimitar o tema, focando-se na aplicação de uma prática sobre manutenção de software. Os objetivos visaram à compreensão da área,

culminando em uma visão simulada de cenários do mercado de trabalho, destacando-se a proposição de soluções práticas para os problemas levantados pelo cliente. Ao final, os objetivos da pesquisa foram obtidos em texto livre.

A próxima etapa foi a execução de uma revisão da literatura que auxiliou na definição dos materiais a serem desenvolvidos. Optou-se pela aplicação de uma prática de ensino com aplicação de aulas teóricas e práticas, utilizando uma dinâmica simulada de cliente-empresa. Esta etapa caracteriza-se como uma revisão *ad-hoc*, uma vez que não foram executados processos sistemáticos.

Em seguida, produziram-se os materiais de apoio[1], sendo eles *slides* para a apresentação das aulas e do código base, assim como um documento *Word* a respeito do software para implementação dos tipos de manutenções abordadas neste estudo. Paralelamente, desenvolveu-se também um caderno virtual utilizando a plataforma *Notion*[2], contendo dicas e explicações sobre a parte teórica e a prática do estudo. O sistema "Pizza Mais" foi implementado para ser aplicado na dinâmica cliente-empresa, de modo a simular um software que necessita de manutenções. O sistema foi desenvolvido utilizando a linguagem de programação *Python*[3], devido à sua sintaxe de alto nível e utilização no curso de Engenharia da Computação. O software possuía arquitetura M.V.C (*model, view, controller*) e uma base de dados em *SQLite*[4] para gerenciamento de dados. Além disso, também foi estruturado um repositório[5] no *GitHub* com o código-base para realizar as manutenções necessárias.

De posse desses recursos, executaram-se as aulas teóricas sobre manutenção de software, a fim de nivelar e revisar o conteúdo previamente ministrado na disciplina. O objetivo foi incentivar ativamente a participação dos alunos, promovendo discussões e interações, visando proporcionar uma compreensão prática do tema. A aplicação da prática ocorreu ao longo de duas semanas, que abrangeram as aulas sobre o conteúdo, explicações sobre os desafios e horários dedicados a aulas tira-dúvidas. Todas as aulas ocorreram presencialmente e utilizaram-se de uma abordagem teórica e prática, bem como dos recursos de apoio mencionados anteriormente. Após a aplicação da parte teórica, os alunos precisaram analisar o código-base e compreender sobre a etapa de manutenção no desenvolvimento de software.

Para a execução da prática, os alunos poderiam compor grupos de até quatro pessoas, onde deveriam dividir as responsabilidades na própria equipe. Foi obrigatória a utilização do *GitHub*, de modo que as equipes realizassem clonagem do repositório do código-base. Além disso, era necessário que todos os membros realizassem *commits*, a fim de também serem avaliados individualmente.

Posteriormente, foi aplicado um *survey*[6] avaliativo com os alunos. Esse formulário foi elaborado por um dos autores desta pesquisa, para obtenção das percepções qualitativas e quantitativas dos discentes. Além disso, este material foi avaliado pelo docente responsável da disciplina e por dois doutorandos (todos autores deste artigo) que

[1]Acesso ao repositório - http://bit.ly/48cEDF3
[2]Plataforma de anotações - https://www.notion.com/
[3]Linguagem de programação utilizada - https://www.python.org/
[4]Banco de dados utilizado no software - https://www.sqlite.org/
[5]Acesso ao repositório no github - https://bit.ly/3MxOQTX
[6]Disponível no Zenodo - https://bit.ly/48cEDF3

participaram ativamente na implementação desta prática. Este formulário é um modo relevante para aquisição de dados a respeito da eficácia do método proposto, além de possibilitar quais pontos podem e devem ser melhorados.

5. RESULTADOS

Esta seção apresenta uma análise dos resultados obtidos com a aplicação da prática em manutenção de software, comparando-os com os objetivos estabelecidos.

5.1. Conhecimento prévio dos participantes sobre o tema

A amostra desta prática foi composta por discentes de uma turma de Engenharia de Software, na qual dos 40 alunos matriculados, 37 participaram ativamente. Destes, somente 21 discentes (cerca de 54% dos participantes) responderam ao formulário de caracterização enviado previamente. Mediante a Figura 2, identifica-se que a turma é majoritariamente iniciante no tema, possuindo somente conhecimento teórico ou pequena experiência prática. O restante dos participantes (23,8%) indicou possuir experiência moderada ou avançada, podendo, então, oferecer referências do mercado de trabalho aos demais participantes.

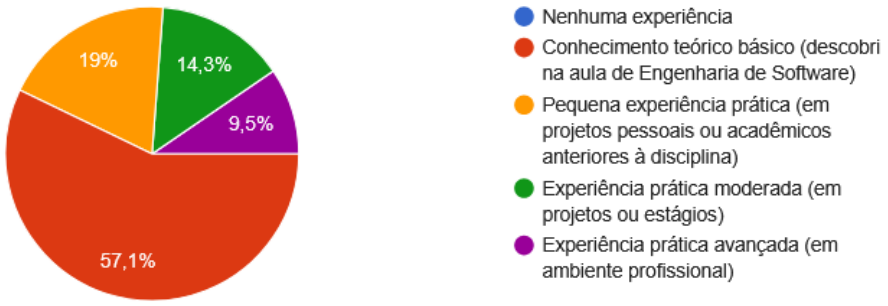

Figura 2. Experiência prévia dos participantes com manutenção de software.

5.2. Conhecimentos adquiridos através da abordagem prática

Através do formulário avaliativo[7], executado ao final do projeto, observa-se pela Figura 3, que 26 alunos (cerca de 70% da amostra) contribuíram com seus *feedbacks*. Os participantes foram questionados sobre o nível de conhecimento que eles adquiriram sobre manutenção de software. Os resultados foram categorizados em escala *likert* de quatro pontos, na qual 1 representa uma baixa aquisição de conhecimento e 4 representa uma aquisição muito alta.

Observa-se, via Figura 4, que não houve votos na escala "baixo", ou seja, todos os participantes afirmaram ter adquirido um nível significativo de conhecimento com a prática. Enquanto 3 alunos (cerca de 12%) indicaram uma aquisição moderada sobre o tema, 23 alunos (compondo cerca de 88% da amostra) indicaram a escala de aquisição de conhecimento como alta ou muito alta.

[7]Disponível no Zenodo - https://bit.ly/48cEDF3

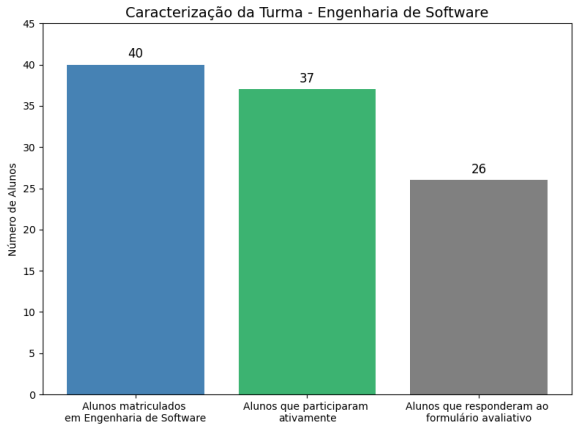

Figura 3. Distribuição dos respondentes ao formulário avaliativo.

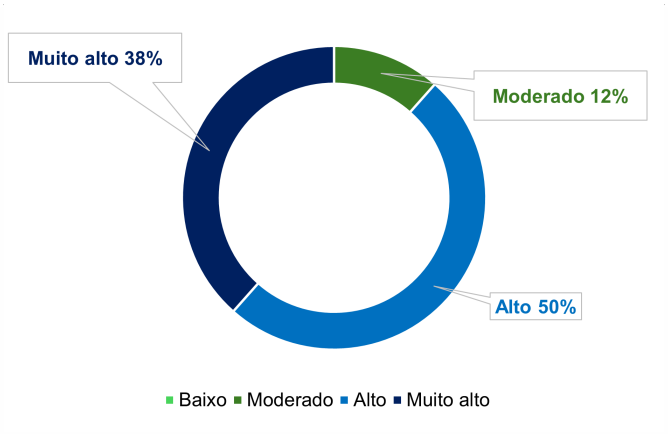

Figura 4. Nível de conhecimento adquirido pelos alunos sobre manutenção de software.

Comparando estes dados com o conhecimento prévio dos alunos (Figura 2), identifica-se um impacto positivo no aprendizado por meio da prática aplicada em sala de aula. Isto é reforçado pelos dados apresentados na Figura 5, na qual é possível identificar que 25 alunos (cerca de 96%) classificaram a prática utilizada como muito benéfica ao aprendizado da manutenção de software. Enquanto isso, somente 1 aluno classificou seu aprendizado como baixo.

Além disso, mediante a Figura 6, é possível observar que os discentes não apresentaram grandes dificuldades em identificar os tipos de manutenção propostos ao longo da prática. Cerca de 12 alunos (46,2%) afirmaram ter conseguido classificar as manutenções com segurança, enquanto 14 alunos (53,8%) tiveram algum grau de dificuldade. Além disso, não houve indicações para grandes graus de dificuldade ou de não conseguir classificar os tipos de manutenções solicitadas.

Esta percepção pode ser atribuída ao material de documentação do software, fornecido para execução da prática. Através da Figura 7, identifica-se que este documento auxiliou positivamente os alunos na compreensão do tema e na execução das atividades. Todos os participantes indicaram que o material foi de grande ajuda, com 21 alunos (81%) classificando a utilidade como muito alta e 5 alunos (19%) como alta.

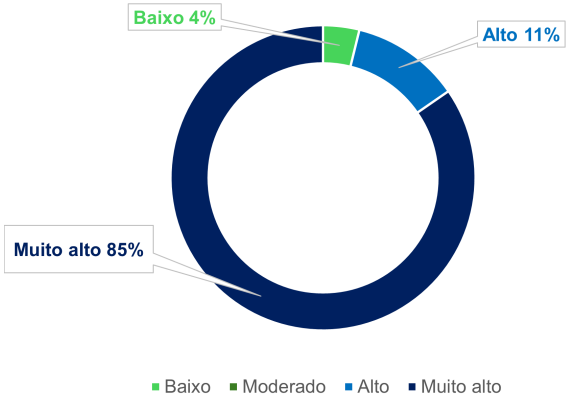

Figura 5. Percepção dos alunos sobre o auxílio da prática no ensino da manutenção de software.

Figura 6. Capacidade dos alunos em identificar os tipos de manutenção.

Através dos resultados, é possível observar que os alunos adquiriram e trocaram conhecimentos e experiências sobre a importância da manutenção de software; desenvolveram habilidades de versionamento de software enfrentando problemas no *GitHub*; além de aprenderam sobre a importância da comunicação e divisão das responsabilidades em uma equipe de desenvolvimento.

6. Conclusão

Este trabalho explorou e aplicou conceitos sobre manutenção de software no contexto acadêmico, com foco na turma de Engenharia de Software, na Universidade Federal do Maranhão do curso de Engenharia da Computação. Ao longo desta experiência, foram abordados aspectos teóricos do tema, os principais tipos de manutenção e impactos nas práticas de desenvolvimento de software. O método de pesquisa empregado envolveu a aplicação de uma dinâmica cliente-empresa, promovendo uma abordagem pedagógica prática a fim de exercitar aspectos técnicos necessários para o tema, assim como incentivar a colaboração entre os participantes.

Conclui-se que a proposta da abordagem prática proposta provou ser uma forma de ensino valiosa para a formação de engenheiros de software, preparando-os melhor frente a desafios reais do mercado de trabalho. A relação bem-sucedida dos conhecimentos teóricos e práticos da disciplina não somente melhora a eficiência no desenvolvimento de

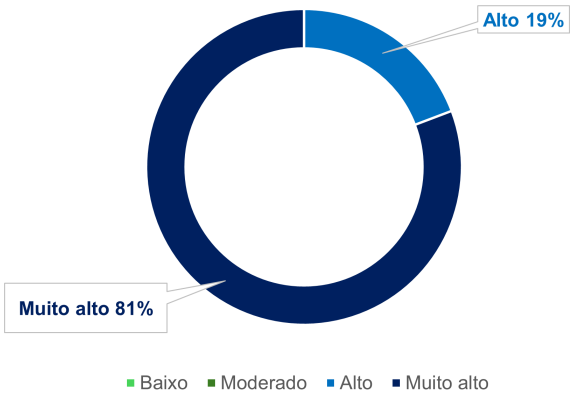

Figura 7. Percepção dos alunos sobre o documento de software.

software, mas também promove um senso de trabalho em equipe, essencial para enfrentar as demandas do mercado atual.

Como trabalhos futuros, alguns alunos deram a sugestão de explorar outras partes do desenvolvimento de software (autenticação) como *deploy*, conexão com API ou aplicação de microsserviços, mas claramente na grade da disciplina. Outra ideia é mudar a aplicação para uma API, desenvolvendo uma interface gráfica, a fim de que os alunos ainda teriam um ambiente mais simplificado, mas entenderiam sobre integração *back - front*.

Referências

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

BARCELLOS, M. et al. Using extension projects to improve software engineering education and software quality: The experience of the “ricardo de almeida falbo” software engineering practices laboratory. In: *Anais do XXIII Simpósio Brasileiro de Qualidade de Software*. Porto Alegre, RS, Brasil: SBC, 2024. p. 552–562. ISSN 0000-0000. Disponível em: ⟨https://sol.sbc.org.br/index.php/sbqs/article/view/32983⟩.

BEZERRA, C. et al. Contributing to open-source projects in refactoring code smells: A practical experience in teaching software maintenance. In: *Anais do XXXVIII Simpósio Brasileiro de Engenharia de Software*. Porto Alegre, RS, Brasil: SBC, 2024. p. 399–409. ISSN 2833-0633. Disponível em: ⟨https://sol.sbc.org.br/index.php/sbes/article/view/30380⟩.

CAVALCANTE, V. et al. Contributions of an extension course focused on good software engineering practices for students and it professionals. In: *Anais do XXII Simpósio Brasileiro de Qualidade de Software*. Porto Alegre, RS, Brasil: SBC, 2023. p. 301–310. ISSN 0000-0000. Disponível em: ⟨https://sol.sbc.org.br/index.php/sbqs/article/view/27117⟩.

GALLAGHER, K.; FIORAVANTI, M.; KOZAITIS, S. Teaching software maintenance. In: *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. [S.l.: s.n.], 2019. p. 353–362.

LEVIN, S.; YEHUDAI, A. *Towards Software Analytics: Modeling Maintenance Activities*. 2019. Disponível em: ⟨https://arxiv.org/abs/1903.04909⟩.

LIENTZ, B. P.; SWANSON, E. B. *Software maintenance management*. [S.l.]: Addison-Wesley Longman Publishing Co., Inc., 1980.

LIMA, I.; RABELO, J. A utilização da gamificação como ferramenta motivadora no ensino de manutenção de software. In: *Anais do XI Workshop de Visualização, Evolução e Manutenção de Software*. Porto Alegre, RS, Brasil: SBC, 2023. p. 11–15. ISSN 0000-0000. Disponível em: ⟨https://sol.sbc.org.br/index.php/vem/article/view/25842⟩.

MAGALHãES, N. et al. Suporte às atividades de manutenção de software em bases de dados abertas e distribuídas. In: *Anais do XXI Simpósio em Sistemas Computacionais de Alto Desempenho*. Porto Alegre, RS, Brasil: SBC, 2020. p. 227–238. ISSN 0000-0000. Disponível em: ⟨https://sol.sbc.org.br/index.php/sscad/article/view/14072⟩.

MENS, T.; TOURWÉ, T. A survey of software refactoring. *IEEE Transactions on software engineering*, IEEE, v. 30, n. 2, p. 126–139, 2004.

NASCIMENTO, J. et al. Do code smell ao código limpo: Uma experiência prática no ensino de refatoração para manutenção de software. In: *Anais do XXXIII Workshop sobre Educação em Computação*. Porto Alegre, RS, Brasil: SBC, 2025. p. 701–711. ISSN 2595-6175. Disponível em: ⟨https://sol.sbc.org.br/index.php/wei/article/view/36209⟩.

OPDYKE, W. F. *Refactoring object-oriented frameworks*. [S.l.]: University of Illinois at Urbana-Champaign, 1992.

OTEMAIER, K. et al. Immersive role-playing: An experience report on a promising approach to learning requirements elicitation. In: *Anais do XXIII Simpósio Brasileiro de Qualidade de Software*. Porto Alegre, RS, Brasil: SBC, 2024. p. 586–595. ISSN 0000-0000. Disponível em: ⟨https://sol.sbc.org.br/index.php/sbqs/article/view/32986⟩.

PRESSMAN, R.; MAXIM, B. *Engenharia De Software: UMA ABORDAGEM PROFISSIONAL*. MCGRAW HILL - ARTMED, 2016. ISBN 9788580555332. Disponível em: ⟨https://books.google.com.br/books?id=smNFvgAACAAJ⟩.

SINGH, R. International standard iso/iec 12207 software life cycle processes. *Software Process Improvement and Practice*, Citeseer, v. 2, n. 1, p. 35–50, 1996.

SNEED, H. M.; BROSSLER, P. Critical success factors in software maintenance: a case study. In: IEEE. *International Conference on Software Maintenance, 2003. ICSM 2003. Proceedings.* [S.l.], 2003. p. 190–198.

SOMMERVILLE, I. *Software Engineering*. 10th. ed. [S.l.]: Pearson, 2015. ISBN 0133943038.