A Negligência da Dívida Técnica e seus Impactos na Sustentabilidade de Sistemas de Informação

Gabriel Bastos[1] **, Nilson Lazarin**[1] **, Diego Castro**[1,2]

[1] Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (CEFET/RJ) Rio de Janeiro – RJ – Brasil ,
[2] Universidade Federal do Rio de Janeiro (UFRJ) Rio de Janeiro – RJ – Brasil

[gabriel.bastos.2@aluno, nilson.lazarin@, diego.castro@]cefet-rj.br

Abstract. *Information Systems (IS) are constantly advancing under pressure to deliver features ever faster, often prioritizing immediate results at the expense of internal product quality. This environment promotes the accumulation of technical debt, especially in invisible forms that do not immediately affect system functionality but gradually compromise architectural integrity, data consistency, security, and maintainability. Over time, this degradation amplifies complexity, increases maintenance costs, decreases efficiency, and limits the potential for sustainable system development. To address this situation, governance must prioritize refactoring and the implementation of preventive code fixes. Furthermore, the adoption of good development practices is essential. This article examines how governance negligence, combined with deadline pressure, contributes to the accumulation of tech debt, emphasizing the need for metric creation and refactoring to maintain the structural health of the system.*

Resumo. *Os Sistemas de Informação (SI) avançam constantemente sob a pressão por entregas de funcionalidades cada vez mais rápidas, frequentemente priorizando resultados imediatos em detrimento da qualidade interna do produto. Esse ambiente promove o acúmulo de dívida técnica, especialmente em formas invisíveis que não afetam imediatamente a funcionalidade do sistema. Entretanto, gradualmente comprometendo a integridade arquitetural, a consistência dos dados, a segurança e a manutenibilidade. Com o tempo, essa degradação amplifica a complexidade, aumenta os custos de manutenção, diminui a eficiência e limita o potencial de desenvolvimento sustentável do sistema. Para lidar com essa situação, é necessário que a governança priorize refatoração e a implementação de correções preventivas no código. Além disso, a adoção de boas práticas de desenvolvimento é fundamental. Este artigo examina como a negligência da governança, aliada à pressão por prazos de entrega, contribui para o acúmulo de dívida técnica, enfatizando a necessidade de criação de métricas e refatoração para a manutenção da sáude estrutural do sistema.*

1. Introdução

A evolução dos Sistemas de Informação (SI) ocorre em um cenário caracterizado pela crescente demanda por novas funcionalidades e pela pressão por entregas rápidas [Melo and Ferreira 2010]. Nesse contexto, as organizações priorizam resultados no curto

prazo, em detrimento da qualidade interna dos sistemas [de Melo et al. 2021]. Como consequência, observa-se o acúmulo de dívida técnica, a qual nem sempre é visível, impactando diretamente a qualidade, complexidade, manutenibilidade e evolução dos sistemas [Franco et al. 2018, Oliveira et al. 2020].

Essas dívidas são comumente apresentadas como um custo invisível e de difícil identificação por meio de mecanismos tradicionais de análise, mas que se manifestam progressivamente durante o desenvolvimento do sistema. Suas principais consequências estão diretamente ligadas ao aumento de esforço de manutenção, redução da produtividade dos desenvolvedores e degradação da qualidade do software [Femenick 2005, Khan et al. 2022]. No contexto de SI, esses impactos ainda são mais críticos, não se limitando ao domínio técnico, mas também abrangendo a governança, gestão de informações e tomada de decisões críticas.

Embora amplamente discutida na Engenharia de Software (ES), essas dívidas devem ser analisadas também sob uma perspectiva de natureza organizacional, informacional e estratégica nos SI, reduzindo o tratamento apenas em perspectiva técnica. Essa limitação contribui para sua negligência de mecanismos eficazes de identificação, mensuração e governança, sendo frequentemente negligenciada nos processos de tomada de decisão que consideram adequadamente o valor de negócio e restrições de recursos [De Almeida et al. 2021].

Considerando esse panorama, este trabalho parte da seguinte provocação: a dívida técnica invisível não é apenas um problema técnico, mas resulta de falhas estruturais na priorização e na visibilidade dos custos associados a evolução do sistema [Lenarduzzi et al. 2021]. A literatura direciona que decisões de planejamento e gestão são os principais fatores de criação de dívida técnica, devido ao curto prazo disponível para o desenvolvimento e mudanças de regras de negócio. Partindo dessa perspectiva, defende-se que a ausência de mecanismos para tornarem explícitas essas falhas para os gestores, contribui para a negligência, visto que a falta de alinhamento entre prioridades técnicas e as visões de governança dificultam a gestão do sistema [De Almeida et al. 2021].

Como contribuição, este artigo propõe uma reflexão conceitual da dívida técnica invisível em SI, debatendo seus efeitos e implicações para a governança e sustentabilidade do software. Para sustentar essa ideia, foi utilizado um caso ilustrativo de um sistema real desenvolvido ao longo de múltiplos ciclos acadêmicos, no qual foram identificados indícios de dívidas técnicas invisíveis nas dimensões arquitetural, de dados e de segurança. Os efeitos observados foram limitações na coleta, integração e interpretação de dados sobre a dívida técnica [Avgeriou et al. 2025a]. Esse estudo de caso foi utilizado para refletir como essas dívidas surgem e causam impactos negativos de forma silenciosa na evolução do sistema, permitindo criar lições iniciais e levar questões relevantes para a área de SI.

2. Problematização e Provocações

A literatura de ES reconhece a dívida técnica como decisão de curto prazo, porém caso não resolvida, pode acarretar atraso no desenvolvimento quando identificada como limitação à evolução de funcionalidades ou como fonte de problemas. Sua postergação de refatoração contribui para uma degradação exponencial da qualidade interna do produto, tornando as mudanças mais trabalhosas e demoradas com o decorrer do tempo

[Couto et al. 2011, Fowler and Beck 2013]. No entanto, essa interpretação se mostra insuficiente para explicar os desafios observados em contextos reais de SI, nos quais os impactos da dívida vão além do domínio técnico, influenciando decisões organizacionais e de governança ao longo prazo.

Em ambientes orientados a entregas contínuas, é possível observar um padrão comportamental recorrente, onde as decisões técnicas são colocadas em posição secundária de forma sistemática às pressões de curto prazo, priorizando novas funcionalidades em detrimento da integridade estrutural e correções de problemas possivelmente gerados em outras entregas[De Almeida et al. 2021, Avgeriou et al. 2023]. O acúmulo invisível da dívida técnica no decorrer do ciclo de vida do sistema contribui para o agravamento desses problemas, bem como para a redução da visibilidade, uma vez que não é possivel que se manifeste em métricas tradicionais de desempenho. Tal fato revela uma lacuna crítica das organizações em tornar a dívida técnica um mecanismo formal de decisão de negócio. Essa invisibilidade compromete decisões estratégicas, criando um ciclo de degradação estrutural contínuo por conta da postergação de uma reestruturação e de melhoria [Avgeriou et al. 2025a].

As métricas tradicionais são avaliadas com base em indicadores financeiros e de negócio. Dentre os principais, destacam-se critérios de avaliação como volume de vendas, receita, lucro, custos operacionais e retorno sobre investimento [Marcos Henrique de Moraes et al. 2024, Digkas et al. 2021]. Embora sejam primordiais para avaliar resultados econômicos e orientar decisões estratégicas, não identificam aspectos relacionados à qualidade interna do software como acúmulo de dívida técnica, complexidade estrutural e manutenibilidade, o que contribui para sua baixa visibilidade nos processos de decisão organizacional sobre o tema.

Diante desse cenário, é possível evidenciar um desalinhamento estrutural entre decisões técnicas e gerenciais, de tal modo que equipes de desenvolvimento apresentam planos e reconhecem como demandas necessárias a refatoração e manutenção estrutural preventiva, entretanto tais atividades comumente não são priorizadas em nível organizacional por não apresentarem retorno financeiro imediato [Júnior and Oliveira 2012]. Desse modo, essa dívida não é tratada como risco estratégico e passa a ser percebida como responsabilidade operacional, deslocada para o nível dos desenvolvedores. Com isso, muitos desenvolvedores ficam desmotivados em atuarem em projetos nos quais não são priorizadas as correções preventivas ou pontuais, exigindo esforço adicional para entregar o mesmo resultado e tentar resolver erros gerados [Avgeriou et al. 2023].

Nesse contexto, este trabalho propõe três provocações centrais. Primeiro, a dívida técnica invisível não deve ser compreendida exclusivamente técnica, mas como fenômeno sociotécnico de decisões organizacionais e limitações nos mecanismos de governança [Galster et al. 2024]. Segundo, estratégias reativas baseadas em refatoração pontual são insuficientes para lidar com a situação, uma vez que não atacam as causas estruturais de sua geração. Terceiro, a ausência de mecanismos que tornem explícitos os impactos da dívida técnica compromete a sustentabilidade dos SI, ao dificultar sua incorporação em decisões orientadas a valor de negócio [Bischoff et al. 2022].

Sob essa ótica, a refatoração não deve ser considerada apenas um recurso técnico, mas como decisão estratégica que compete por alocação de recursos organizacionais, pla-

nejamento de projeto e produtividade [Gupta 2025]. No entanto, sua adoção enfrenta diversas barreiras significativas, principalmente a dificuldade de mostrar em números sua necessidade para a governança, dificultando a visualização dos seus benefícios no curto prazo [Avgeriou et al. 2025b, Avgeriou et al. 2023]. Como consequência, as organizações tendem a postergar tal atividade até que seu acumulo de complexidade estrutural, levado por essa decisão gerencial aumenta os custos de manutenção e eleva riscos operacionais, podendo gerar vulnerabilidades de segurança e não conformidade regulatória [Buschmann 2011, Alves and Spínola 2017, Fowler and Beck 2013].

Esses efeitos se tornam críticos na visão regulatória e de gestão de dados, nos quais falhas estruturais comprometem a integridade e a segurança das informações. Assim, representa um risco altamente relevante por conta da Lei Geral de Proteção de Dados (LGPD), podendo gerar multas por descumprimento da legislação. Cabe ainda ressaltar que, um software que tenha dados comprometidos, pode acarretar na perda gradual desses usuários e dos demais que não se sentirem seguros em utilizar [Santos et al. 2025]. Ainda assim, tais riscos permanecem frequentemente subestimados, justamente por sua natureza pouco definida e de manifestação tardia.

Estudos indicam que parcela relevante dos esforços de desenvolvimento em grandes organizações são voltados para a dívida técnica [Paudel et al. 2024]. Esse percentual, evidencia impacto significativo na produtividade dos programadores, gerando um descontentamento e um desânimo durante a criação de novas funcionalidades. Estudos mostram que apesar de cerca de 117 métricas propostas para mensurar a dívida técnica, nenhuma delas é capaz de identificar de forma confiável a necessidade de uma intervenção, somente com o olhar de profissionais técnicos foi capaz de sinalizar quais ações sao prioritárias [Avgeriou et al. 2023]. Fica evidente a necessidade de alocar recursos estratégicos como profissionais qualificados, capazes de relatar problemas estruturais críticos para a governança, sendo uma abordagem sustentavel e evolutiva do SI.

2.1. Refatoração: Custo Técnico ou Investimento Estratégico?

Apesar de amplamente defendida pela literatura, a refatoração permanece como prioridade secundária em contextos organizacionais, evidenciando uma lacuna entre conhecimento técnico e prática gerencial [Sandberg et al. 2015]. Sob a perspectiva de negócio, a ausência de manutenção preventiva contribui para uma degradação estrutural dos sistemas, resultando em demora na implantação de novas funcionalidades, além de perder a credibilidade e a confiança dos usuários, especialmente em cenários envolvendo falhas de segurança. Como consequência desses fatos, o impacto na geração de valor é negativo acarretando perdas de receita no médio e longo prazo [Fowler and Beck 2013].

Entretanto, deve-se analisar que a refatoração constitui um meio para alcançar o objetivo esperado e que não se reflete de forma imediata nos indicadores financeiros tradicionais. No curto prazo, o lucro fica estável, podendo declinar por não apresentar atualizações constantes para seus usuários. Esse cenário contribui para que gestores posterguem essas iniciativas, na medida em que os gestores enfrentam limitações na mensuração dos benefícios associados a mitigação da degradação de curto prazo [Avgeriou et al. 2025b]. Como resultado, evidencia-se uma limitação nos mecanismos de avaliação, que tendem a privilegiar ganhos imediatos em detrimento da sustentabilidade estrutural de SI [Avgeriou et al. 2025b, Sandberg et al. 2015]. Por isso, vale

ressaltar que esse processo deve ser feito de forma estratégica e gradual para que não comprometa a saúde financeira da empresa e sua evolução contínua do sistema [Buschmann 2011, Alves and Spínola 2017].

A postergação da refatoração também implica negativamente nos indicadores da empresa, ainda que de forma pouco perceptível no curto prazo. Ao longo do tempo, o acúmulo de dívida técnica aumenta a complexidade do sistema, exigindo maior esforço para novas implementações e eleva os custos associados as equipes de desenvolvimento [Buschmann 2011]. Essa crescente complexidade, coloca a organização dependente de profissionais que já tenham domínio do ecossistema, dificultando a integração de novos programadores por precisar de um longo tempo de adaptação e passagem de conhecimento, uma vez que a modificação de uma parte do código pode por sua vez alterar outras, acarretando num novo erro ou vulnerabilidade [Buschmann 2011].

Dessa forma, as discussões sobre refatoração de sistemas no âmbito organizacional exige uma mudança de perspectiva: decisões de natureza técnica e estrutural e de longo prazo precisam ser incorporadas a agendas estratégicas da governança. A refatoração deve ser compreendida pela governança não só como custo operacional isolado mas como parte sustentável de sua organização, ainda que seus benefícios não sejam visíveis de forma imediata por métricas tradicionais de desempenho.

3. Caso Empírico: Dívida Técnica Invisível em um Sistema de Informação Real

O projeto Web Gerenciador para Instituicões Assistenciais (WeGIA) foi desenvolvido por alunos do curso técnico do CEFET-RJ - campus Nova Friburgo. Sua criação surgiu pela falta de softwares *Open Sources*, cujo seu objetivo central é melhorar a gestão, controle e transparência das entidades publicas [Lazarin et al. 2024]. Apesar da relevância do projeto, este foi elaborado sem a observância de boas práticas, além de ter sido desenvolvido por alunos frequentemente sem experiência anterior e aprendendo durante o processo. Sua evolução incremental e sua constante rotatividade de programadores criaram um ambiente propício para o acúmulo de decisões técnicas de curto prazo, o que ocasionou no acúmulo de dívidas ténicas [Codabux et al. 2024, Sprague and Institute of Electrical and Electronics Engineers 2010].

Com o passar do tempo, a aplicação que inicialmente foi criado para ampliar a eficiência e a transparência organizacional passou a representar um obstáculo técnico à sua evolução, dificultando a manutenção pelos desenvolvedores. Cada funcionalidade incrementada se tornava evidente a necessidade de reestruturação do código associado, porém, essa demanda era postergada em favor das entregas visíveis [Lenarduzzi et al. 2021]. Temas como arquitetura, qualidade de código, modelagem de dados e segurança deixaram de ocupar posição estratégica, sendo frequentemente atribuídas aos desenvolvedores como resposabilidade operacional.

A ausência da centralização de uma governança arquitetural explícita e a alta rotatividade de desenvolvedores, aliada à falta de documentação técnica do produto contribuíram para a criação e acúmulo de dívidas técnicas [Ahmad et al. 2025]. Novas funcionalidades eram implementadas de forma local, sem se preocupar com a estrutura do sistema, o que gradativamente ocasionava dados redundantes, tabelas desnecessárias e complexidade estrutural do código

[Sprague and Institute of Electrical and Electronics Engineers 2010]. Suas consequências não eram percebidas pelos usuários, mas começaram a ficar evidentes para os desenvolvedores. Relatos indicavam dificuldade em prosseguir com novas funcionalidades e realizar manutenção nas demandas, o que acarretava atraso nas entregas [Bogner et al. 2021, Silveira Neto et al. 2025].

A ausência de testes automatizados reforçava um ambiente de insegurança evolutiva. À medida que o software evoluía, pequenas alterações passaram a demandar trabalho manual extensivo para que fossem validadas todas as funcionalidades. O modelo de dados, por sua vez, apresentava inconsistências estruturais advindas da ausência de padronização e normalização, reflexo da falta de priorização estratégica. De forma paralela, mecanismos de controle de acesso e validação de dados eram implementados sem uniformidade, evidenciando lacunas na abordagem de segurança.

O resultado foi a configuração de um ambiente funcional, porém estruturalmente fragilizado, gerando um exemplo concreto de como a dívida técnica se consolida de maneira silenciosa. A sustentabilidade do software passou a ser compreendida pela crescente dificuldade em modificá-lo com segurança, previsibilidade e estabilidade [Silveira Neto et al. 2025]. Nesse contexto, foi conduzida uma refatoração estrutural abrangente do sistema. A intervenção contemplou a migração para uma arquitetura baseada em módulos, com o uso de *frameworks* de mercado. Paralelamente, problemas estruturais foram mitigados por meio da normalização dos dados, priorizando a segurança da aplicação [Codabux et al. 2024]. As três dimensões das dívidas técnicas identificadas foram: arquitetural, de dados e de segurança.

3.1. Dívida Arquitetural

A ausência de decisões arquiteturais explicitamente definidas tende a produzir projetos frágeis, com desorganização e perda de coerência estrutural. No caso do sistema Wegia [Lazarin et al. 2024], a evolução incremental do sistema ocorreu sem definições claras de fronteiras modulares, responsabilidades e contratos entre os componentes, resultando em crescimento acoplado e não planejado.

Embora a solução fosse apresentada ao cliente composta por distintos módulos, sua organização interna não refletia essa separação. A dependência entre arquivos e componentes impossibilitava a disponibilização isolada de somente alguns módulos, exigindo a implantação integral da aplicação a cada funcionalidade. Esse cenário gerava consumo desnecessário de recursos computacionais e aumentava a complexidade de manutenção e evolução [Shahin et al. 2017].

Essa estrutura evidencia uma característica recorrente da dívida arquitetural invisível, o sistema permanece funcional na perspectiva do usuário final, mas estruturalmente se torna gradativamente mais rígido para evoluções [Bogner et al. 2021, Verdecchia et al. 2020]. Nenhuma decisão isolada era suficientemente grave para a governança justificar uma intervenção. No entanto, a degradação se consolidava de forma distribuída e silenciosa ao longo dos ciclos de desenvolvimento [Verdecchia et al. 2020].

A intervenção realizada, após uma análise crítica de um profissional qualificado da área, consistiu na reformulação arquitetural em módulos, delimitando de forma clara as responsabilidades e separando fisicamente os componentes. Essa segregação foi orientadapelo domínio do negócio da aplicação, dividindo as responsabilidades em regra de

negócio, interface, persistência de dados e mecanismos de segurança. Essa reestruturação reduziu o acoplamento e estabeleceu contratos entre os componentes [Shahin et al. 2017].

A melhoria estrutural vai além da melhoria na arquitetura, ela evidencia a sustentabilidade de sistemas que possuem alto grau de complexidade. Quando tais decisões são postergadas ou negligenciadas, o software tende a acumular complexidade estrutural silenciosa, comprometendo sua longevidade [Verdecchia et al. 2020]. A arquitetura não é um artefato inicial no desenvolvimento do projeto, ela é uma decisão contínua da governança.

3.2. Dívida de Dados

Sistemas cujo modelo de dados é concebido sem rigor estrutural tendem a tornar-se progressivamente insustentáveis a médio e longo prazo. No caso analisado, a evolução contínua produziu uma base de dados volumosa, porém estruturalmente frágil. A negligência da normalização adequada, juntamente com a redundância de dados, comprometia a coerência do modelo relacional, aumentava a complexidade das consultas e o risco de inconsistência informacional [Albarak et al. 2017].

Apesar desses impactos, o sistema continuava a operar com êxito do ponto de vista funcional. Suas fragilidades não são perceptíveis de forma direta para os usuários e gestores. Essa invisibilidade constitui o principal desafio dessa categoria de dívida técnica, sua operação continua aparentemente estável, mas sua evolução torna-se um desafio complexo. A ausência de normalização impossibilita a reutilização de dados, tornando as tabelas cada vez maiores e concentrando as responsabilidades em estruturas únicas e monolíticas [Albarak et al. 2017].

Essa situação expõe uma característica típica associada a dívidas técnicas de dados invisíveis, com o decorrer do tempo fica evidente a dificuldade na introdução de novas funcionalidades e complexidade crescente nas consultas e sintomas de rigidez estrutural que se manifestam antes de qualquer falha perceptível para o cliente. Em estágios mais avançados, tal configuração pode acarretar em problemas visíveis, como lentidão nas consultas e duplicação de informações [Bogner et al. 2021].

A estratégia adotada concentrou-se na modelagem de um novo banco de dados a partir da base existente, conduzida de forma modular para preservar a coerência entre domínios e funcionalidades. Foram aplicadas boas práticas de normalização, redefinição de responsabilidades e eliminação de redundâncias [Albarak et al. 2017]. Além disso, foram revistos os tipos de dados e o modelo de armazenamento de arquivos, substituindo o armazenamento direto em formato binário por referências estruturadas externas, reduzindo acoplamento e melhorando a escalabilidade do sistema.

Os benefícios destas modificações demonstram que a modelagem de dados não é uma etapa inicial do projeto, mas um elemento de governança contínuo. Ao priorizar velocidade nas entregas em detrimento da qualidade da arquitetura, compromete-se aspectos estruturais do sistema, correndo o risco de se construir novas funcionalidades com estruturas vulneráveis, cujo custo pela reestruturação cresce de forma não linear [Bogner et al. 2021, Khan et al. 2022].

3.3. Dívida de Segurança

A ausência de padrões de segurança compromete sistemas de informação não apenas de forma técnica, mas também organizacional e jurídica. Ao analisar o Wegia [Lazarin et al. 2024], a introdução acelerada de novas funcionalidades, juntamente com prazos curtos e priorização de entregas visíveis gera fragilidades no controle de acesso e na proteção dos dados [Khan et al. 2022]. Embora o sistema continue funcional, as vulnerabilidades permaneciam invisíveis para os usuários operacionais, expondo informações sensíveis e permitindo acessos incoerentes com os perfis definidos [Codabux et al. 2024], originando mais de 160 vulnerabilidades no sistema [LabRedes CEFET/RJ 2026].

Esse cenário evidencia a característica central da dívida de segurança invisível, onde sua manifestação só se torna perceptível por meio de exploração técnica [Khan et al. 2022]. As vulnerabilidades estruturais permanecem latentes e se manifestam apenas em auditorias especializadas, testes de penetração ou incidentes concretos. Como resultado de sua identificação tardia, o custo de correção torna-se superior ao de prevenção.

A negligência de planos de validação, incluindo testes de vulnerabilidade, análise de dependências e revisão de políticas de acesso reforça sua fragilidade. Tal ausência não se configura como técnica, mas organizacional, refletindo a baixa prioridade institucional atribuída à segurança em favor de resultados rápidos e operacionalmente visíveis. [Khan et al. 2022]

Além dos benefícios técnicos advindos das modificações, essa reestruturação evidencia que segurança não deve ser tratada como requisito complementar ou tarefa de segunda ordem. Quando postergada, a dívida de segurança se converte em risco jurídico, reputacional e operacional, cujo custo de mitigação se torna superior ao ed prevenção que teria sido necessário desde o início [Lenarduzzi et al. 2021].

A experiência analisada demonstra que a comunidade de SI frequentemente associa produtividade e sucesso às entregas funcionais, negligenciando métricas relacionadas à proteção estrutural. Tal postura reforça a necessidade de incorporar, à governança, uma equipe especializado e dedicado para encontrar e combater essas dívidas, sendo possível a geração de indicadores de maturidade e qualidade estrutural às boas práticas de governança em projetos de software [Codabux et al. 2024].

4. Conclusão

A pressão por entregas rápidas e contínuas tende a gerar prejuízos estruturais nos SI, levando equipes de desenvolvimento a construírem novas funcionalidades sem endereçar as dívidas técnicas acumuladas ao longo do projeto, produzindo custos invisíveis para as empresas. A postergação sistemática da refatoração acarreta degradação estrutural progressiva do software, cujos efeitos só se tornam perceptíveis quando o custo de manutenção já é maior do que o de prevenção.

O principal aprendizado ao longo do artigo é que a sustentabilidade de SI depende da capacidade organizacional de tornar visíveis os custos estruturais. Enquanto a governança não implementar mecanismos para incorporar a dívida técnica às decisões estratégicas da empresa, alocando recursos para mitigação de riscos, o ciclo de degradação tende a se intensificar, tornando inviável a continuidade de projetos, independentemente

das competências técnicas das equipes envolvidas. Esse debate é relevante para a área de SI, pois coloca a qualidade interna como elemento central do valor organizacional e não apenas no domínio da ES.

Por fim, este estudo evidencia a necessidade de aprofundamento das discussões sobre a lacuna informacional das dívidas técnicas invisíveis. Estudos futuros devem avançar em três direções complementares para fomentar debates e soluções para o problema: o desenvolvimento de métricas e modelos capazes de quantificar os impactos das dívidas técnicas sob uma abordagem de risco organizacional, a investigação de abordagens empíricas que tornam clara a relação entre qualidade estrutural e indicadores de sustentabilidade em SI, e a proposição de mecanismos institucionais capazes de integrar decisões de refatoração aos processos formais de planejamento estratégico, evitando assim os padrões atuais de postergação sistemática.

Referências

Ahmad, N., Su, R., Esposito, M., Janes, A., Lenarduzzi, V., and Taibi, D. (2025). Architectural Degradation: Definition, Motivations, Measurement and Remediation Approaches. DOI: 10.48550/ARXIV.2507.14547.

Albarak, M., Alrazgan, M., and Bahsoon, R. (2017). Identifying and Managing Technical Debt in Database Normalization Using Machine Learning and Trade-off Analysis. DOI 10.48550/ARXIV.1711.06109.

Alves, N. S. R. and Spínola, R. O. (2017). Organização do Corpo de Conhecimento sobre Dívida Técnica: Tipos, Indicadores, Estratégias de Gerenciamento e Causas. In *Anais do XVI Simpósio Brasileiro de Qualidade de Software (SBQS 2017)*, pages 340–354, Brasil. Sociedade Brasileira de Computação - SBC. DOI: 10.5753/sbqs.2017.15116.

Avgeriou, P., Ozkaya, I., Chatzigeorgiou, A., Ciolkowski, M., Ernst, N. A., Koontz, R. J., Poort, E., and Shull, F. (2023). Technical Debt Management: The Road Ahead for Successful Software Delivery. In *2023 IEEE/ACM International Conference on Software Engineering: Future of Software Engineering (ICSE-FoSE)*, pages 15–30, Melbourne, Australia. IEEE. DOI: 10.1109/ICSE-FoSE59343.2023.00007.

Avgeriou, P., Ozkaya, I., Koziolek, H., Codabux, Z., and Ernst, N. (2025a). Manifesto from Dagstuhl Perspectives Workshop 24452 – Reframing Technical Debt. DOI: 10.48550/ARXIV.2505.13009.

Avgeriou, P., Ozkaya, I., Koziolek, H., Codabux, Z., and Ernst, N. (2025b). Reframing Technical Debt (Dagstuhl Perspectives Workshop 24452). Technical report, Schloss Dagstuhl – Leibniz-Zentrum für Informatik. DOI: 10.4230/DAGREP.14.11.16.

Bischoff, Y., Van Der Wiel, R., Van Den Hooff, B., and Lago, P. (2022). A Taxonomy About Information Systems Complexity and Sustainability. In *Advances and New Trends in Environmental Informatics*, pages 17–33. Springer, Cham. DOI: 10.1007/978-3-030-88063-7_2.

Bogner, J., Fritzsch, J., Wagner, S., and Zimmermann, A. (2021). Industry practices and challenges for the evolvability assurance of microservices: An interview study and systematic grey literature review. *Empirical Software Engineering*, 26(5):104. DOI: 10.1007/s10664-021-09999-9.

Buschmann, F. (2011). To Pay or Not to Pay Technical Debt. *IEEE Software*, 28(6):29–31. DOI: 10.1109/MS.2011.150.

Codabux, Z., Zakia Sultana, K., and Chowdhury, M. N. (2024). A catalog of metrics at source code level for vulnerability prediction: A systematic mapping study. *Journal of Software: Evolution and Process*, 36(7):e2639. DOI: 10.1002/smr.2639.

Couto, C., Valente, M. T., and Bigonha, R. D. S. (2011). Avaliação de Causalidade entre Métricas de Qualidade Interna e Defeitos. In *Anais do X Simpósio Brasileiro de Qualidade de Software (SBQS 2011)*, pages 279–293, Brasil. Sociedade Brasileira de Computação - SBC. DOI: 10.5753/sbqs.2011.15401.

De Almeida, R. R., Do Nascimento Ribeiro, R., Treude, C., and Kulesza, U. (2021). Business-Driven Technical Debt Prioritization: An Industrial Case Study. In *2021 IEEE/ACM International Conference on Technical Debt (TechDebt)*, pages 74–83, Madrid, Spain. IEEE. DOI: 10.1109/TechDebt52882.2021.00017.

de Melo, A. C. C., Fagundes, R. A. d. A., Lima, J. V. V., Alencar, F., and Santos, W. B. (2021). Identificação e mensuração da dívida técnica de requisitos: um survey na indústria de software. In *WER2021-24th Workshop on Requirements Engineering, Brasília, Brazil*. DOI: 10.29327/1298728.24-10.

Digkas, G., Ampatzoglou, A., Chatzigeorgiou, A., Avgeriou, P., Matei, O., and Heb, R. (2021). The Risk of Generating Technical Debt Interest: A Case Study. *SN Computer Science*, 2(1):12. DOI:10.1007/s42979-020-00406-6.

Femenick, T. R. (2005). A PROBLEMÁTICA E A SOLUÇÃO PARA OS "CUSTOS INVISÍVEIS" E "CUSTOS OCULTOS". *Revista UNI-RN*, 4(1/2):49–49. `https://revistas.unirn.edu.br/index.php/revistaunirn/article/view/106`.

Fowler, M. and Beck, K. (2013). *Refactoring: improving the design of existing code*. The Addison-Wesley object technology series. Addison-Wesley, Boston, 28. printing edition.

Franco, E. F., Hirama, K., and Rossi, R. (2018). Uma análise qualitativa-sistemática da relação entre o acúmulo de dívida técnica e a satisfação de usuários ao longo da operação de pacotes de software empresarial. *Revista Gestão da Produção Operações e Sistemas*, 13(4):263–263. DOI: 10.15675/gepros.v13i4.2033.

Galster, M., Institute of Electrical and Electronics Engineers, and Association for Computing Machinery, editors (2024). *2024 ACM/IEEE International Conference on Technical Debt: TechDebt 2024: 14-15 April 2024, Lisbon, Portugal: proceedings*. IEEE, Piscataway, NJ.

Gupta, K. (2025). Measuring the Impact of Technical Debt on Development Effort in Software Projects. DOI:10.48550/ARXIV.2502.16277.

Júnior, C. and Oliveira, A. L. d. (2012). Métricas como Ferramenta de Auxílio para o Gerenciamento de Dívida Técnica em Produtos de Software. `https://repositorio.ufpe.br/handle/123456789/11379`.

Khan, R. A., Khan, S. U., Khan, H. U., and Ilyas, M. (2022). Systematic Literature Review on Security Risks and its Practices in Secure Software Development. *IEEE Access*, 10:5456–5481. DOI: 10.1109/ACCESS.2022.3140181.

LabRedes CEFET/RJ (2026). Wegia: Security. `https://github.com/LabRedesCefetRJ/WeGIA/security`.

Lazarin, N., Escalfoni, R. E., and Ferreira, V. (2024). WeGIA: Web Gerenciador para Instituições Assistenciais. In *Anais do XXI Congresso Latino-Americano de Software Livre e Tecnologias Abertas*, pages 322–330, Porto Alegre, RS, Brasil. SBC. DOI: 10.5753/latinoware.2024.245668.

Lenarduzzi, V., Besker, T., Taibi, D., Martini, A., and Arcelli Fontana, F. (2021). A systematic literature review on Technical Debt prioritization: Strategies, processes, factors, and tools. *Journal of Systems and Software*, 171:110827. DOI:10.1016/j.jss.2020.110827.

Marcos Henrique de Moraes, João Manoel da Silva, Joaquim M. F. Antunes Neto, and Wladimir José Camilo Menegassi (2024). DATA WAREHOUSE E USABILIDADE DO MODELO DIMENSIONAL ESTRELA: IMPLICAÇÕES ORGANIZACIONAIS E O PAPEL DA GESTÃO DA TECNOLOGIA DA INFORMAÇÃO. DOI: 10.5281/ZENODO.12726974.

Melo, C. d. O. and Ferreira, G. R. (2010). Adoção de métodos ágeis em uma instituição pública de grande porte-um estudo de caso. In *WORKSHOP BRASILEIRO DE MÉTODOS ÁGEIS (WBMA)*, pages 112–125.

Oliveira, I., Marques-Neto, H. T., and Xavier, L. (2020). Analisando estratégias para identificação de dívidas técnicas. In *Workshop de Visualização, Evolução e Manutenção de Software (VEM)*, pages 9–16. SBC. DOI: 10.5753/vem.2020.14523.

Paudel, B., Gonzalez-Huerta, J., Zabardast, E., and Klotins, E. (2024). Towards Measuring the Impact of Technical Debt on Lead Time: An Industrial Case Study. DOI:10.48550/ARXIV.2406.01578.

Sandberg, A., Staron, M., and Antinyan, V. (2015). Towards Proactive Management of Technical Debt by Software Metrics. In *CEUR Workshop Proceedings*, pages 1–15. `https://ceur-ws.org/Vol-1525/paper-01.pdf`.

Santos, G. R., Gama, G. Q., Machado, P. R., and Teixeira, J. P. M. (2025). Lei geral de proteÇÃo de dados pessoais (lgpd) e a proteÇÃo de dados sensÍveis no poder judiciÁrio: Um estudo sobre o sigilo dos dados de servidores da justiÇa. *Revista ft*. DOI: 10.69849/revistaft/ar10202511161843.

Shahin, M., Ali Babar, M., and Zhu, L. (2017). Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices. *IEEE Access*, 5:3909–3943. DOI:10.1109/ACCESS.2017.2685629.

Silveira Neto, M. V. D., Malucelli, A., and Reinehr, S. (2025). Agile Software Architecture: Perceptions on Quality and Architectural Technical Debt Management. In *Anais do XIX Simpósio Brasileiro de Componentes, Arquiteturas e Reutilização de Software (SBCARS 2025)*, pages 1–12, Brasil. Sociedade Brasileira de Computação. DOI:10.5753/sbcars.2025.11890.

Sprague, R. H. and Institute of Electrical and Electronics Engineers, editors (2010). *Software entropy in agile product evolution*. IEEE, Piscataway, NJ.

Verdecchia, R., Kruchten, P., and Lago, P. (2020). Architectural Technical Debt: A Grounded Theory. In Jansen, A., Malavolta, I., Muccini, H., Ozkaya, I., and Zimmermann, O., editors, *Software Architecture*, volume 12292, pages 202–219. Springer International Publishing, Cham. DOI:10.1007/978-3-030-58923-3_14.

Minibiografias

Gabriel Bastos

Estudante do curso de Sistemas de Informação do Centro Federal de Educação Tecnológica do Rio de Janeiro (CEFET/RJ). Possui interesse em múltiplas áreas relacionadas ao desenvolvimento de software, com ênfase na arquitetura de sistemas, boas práticas, e nos princípios de refatoração. Ademais, já exerce atividades profissionais nessa área desde 2022, evidenciando sua experiência prática. Ao longo de sua trajetória acadêmica e profissional, participou em múltiplos projetos com a finalidade de reestruturar sistemas legados, assim como implementar padrões de projeto que visam auxiliar na manutenção e na escalabilidade dos sistemas.

Nilson Lazarin

Professor do Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (Cefet/RJ). Graduado em Tecnologia da Computação (2006) e Pós-graduado em Computação em Rede (2009) pela Universidade Paranaense (UNIPAR); Mestre em Sistemas e Computação (2012) pelo Instituto Militar de Engenharia (IME). Doutorando em Ciência da Computação pela Universidade Federal Fluminense (UFF). Possui experiência em Ciência da Computação, atuando nos seguintes temas: Segurança de Sistemas de Informação, Sistemas Embarcados e Sistemas Multiagentes.

Diego Castro

Professor no CEFET/RJ em regime de dedicação exclusiva. Doutor (2025) e Mestre (2020) em Engenharia de Software pela COPPE/UFRJ. Graduado (2017) em Ciência da Computação pela UERJ. Possui ampla formação complementar, incluindo especializações em Projetos de Cloud (2021), Análise de Dados (2021) e MBA em Segurança da Informação (2024), com destaque para a pós-graduação em Segurança e Defesa Cibernética pela Escola Superior de Guerra (ESG). Desde 2016, atua em projetos de desenvolvimento de software na indústria, com experiência no planejamento, gestão e construção de aplicações web (cloud e on-premise), mobile e desktop.