Explorando Decisões Intertemporais para Lidar com a Degradação de Arquitetura de Software

José Augusto Lajo Vieira Vital, Awdren de Lima Fontão

Faculdade de Computação, Universidade Federal do Mato Grosso do Sul (UFMS)
Campo Grande, Mato Grosso do Sul, Brazil
{jose.vital, awdren.fontao}@ufms.br

Abstract

Este trabalho explora a degradação da arquitetura de software, um problema recorrente e desafiador no campo do desenvolvimento de sistemas complexos. À medida que os sistemas evoluem, a arquitetura subjacente frequentemente se deteriora devido a modificações incrementais, falta de documentação e comunicação ineficaz entre as equipes de desenvolvimento. Este estudo foca em como decisões intertemporais podem ser aplicadas para mitigar a vaporização do conhecimento arquitetural, com base em um estudo de caso do aplicativo OBCOOP, e visa propor soluções para preservar e recuperar esse conhecimento, especialmente em ambientes ágeis e distribuídos.

1 Introdução

1.1 Contexto

A arquitetura de software é o alicerce dos sistemas complexos, fornecendo uma estrutura organizada para os componentes e suas interações, de modo a atender aos requisitos funcionais e não funcionais [Bass et al. 2003, Li et al. 2022, Kruchten 2004, Taylor et al. 2014, Avgeriou et al. 2008].

A degradação arquitetural ocorre devido a fatores como modificações incrementais realizadas sob pressão de prazos curtos, frequentemente sem planejamento adequado. Essas mudanças podem desviar a arquitetura de seus objetivos originais, acumulando uma "dívida arquitetural" que aumenta a complexidade e dificulta o gerenciamento do sistema [Bosch 2000, Li et al. 2022].

Outro fator crítico é a falta de documentação adequada, especialmente em equipes que seguem metodologias ágeis, onde a prioridade é dada à flexibilidade e à entrega rápida. Essa prática negligencia o registro do conhecimento arquitetural (CA, do inglês "Architectural Knowledge" - AK), como o racional por trás de decisões, trade-offs e a visão geral da arquitetura. Isso resulta no fenômeno de "vaporização do conhecimento arquitetural"

(VCA), onde informações essenciais se perdem ou se tornam inacessíveis ao longo do tempo [Borrego et al. 2019b].

Em ambientes de desenvolvimento global e distribuído, a comunicação assíncrona e ferramentas instantâneas, como Slack e e-mails, embora facilitem a colaboração, não garantem a preservação estruturada do conhecimento arquitetural. Isso pode levar a inconsistências na implementação e dificuldades futuras de manutenção e integração do sistema [Farenhorst and de Boer 2010, Serban and Visser 2022].

Além disso, a degradação arquitetural se manifesta tecnicamente, com aumento do acoplamento, diminuição da coesão e dependências desnecessárias entre módulos, impactando diretamente a escalabilidade e a capacidade de realizar manutenções e melhorias [Bosch 2000].

Para mitigar esses desafios, é fundamental adotar práticas eficientes de documentação, governança arquitetural contínua e decisões intertemporais — que equilibram necessidades de curto prazo com implicações de longo prazo para a arquitetura do sistema. Além disso, o versionamento e a co-arquitetura entre componentes podem oferecer soluções sustentáveis para a manutenção de sistemas complexos [Dalkir 2011, Lewis et al. 2021].

Este estudo aborda esses problemas propondo soluções que preservem o conhecimento arquitetural e garantam a sustentabilidade na evolução dos sistemas, mitigando os efeitos ilustrados na Figura 1.

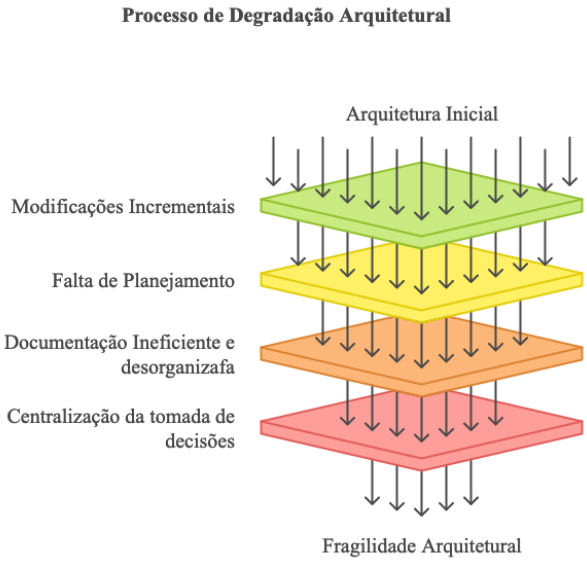

Figure 1: Representação da perda de informações por conta da vaporização.

1.2 Motivação

A degradação da arquitetura de software compromete a qualidade, manutenção e evolução de sistemas complexos ao longo do tempo. Esse processo decorre de adaptações desordenadas às mudanças de requisitos e tecnologia, agravadas em ambientes ágeis onde o foco em entregas rápidas frequentemente desconsidera o planejamento arquitetural de longo prazo. Como

resultado, acumulam-se dívidas técnicas e arquiteturais, elevando os custos de manutenção e dificultando melhorias futuras [Bass et al. 2003, Li et al. 2022].

Além disso, equipes distribuídas enfrentam desafios adicionais devido à comunicação informal e dispersa, contribuindo para a vaporização do conhecimento arquitetural. Informações críticas se perdem, gerando inconsistências no design e dificultando alterações eficientes [Sohan et al. 2016, Borrego et al. 2019b].

Este estudo busca explorar como decisões intertemporais — avaliando benefícios imediatos versus impactos a longo prazo — podem mitigar a degradação arquitetural e preservar o conhecimento. Práticas como documentação eficaz, comunicação estruturada e governança contínua são propostas para promover a sustentabilidade e qualidade do software, mesmo em cenários ágeis e distribuídos [Dalkir 2011, Lewis et al. 2021].

1.3 Problema

A degradação da arquitetura de software é um dos desafios mais significativos enfrentados por equipes que trabalham com sistemas complexos e de longa duração. À medida que esses sistemas evoluem e são constantemente modificados para atender a novos requisitos, a arquitetura inicial, cuidadosamente projetada para cumprir objetivos específicos, tende a se desalinhar [Li et al. 2022]. Com o tempo, essa arquitetura pode se tornar mais frágil, difícil de entender e propensa a falhas. A principal causa disso reside na combinação de modificações incrementais, frequentemente realizadas sem um planejamento adequado, e na falta de documentação e comunicação eficazes entre os membros da equipe [Borrego et al. 2019b]. A Figura 2 ilustra algumas das consequências da degradação arquitetônica.

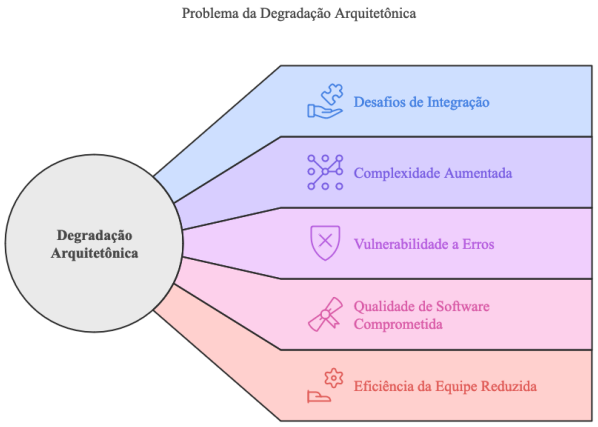

Figure 2: Representação de algumas das consequências da degradação arquitetônica.

Esse fenômeno é exacerbado pela vaporização do conhecimento, que ocorre quando o conhecimento sobre as decisões arquiteturais — incluindo o racional, os trade-offs e as justificativas que guiaram o design inicial — não é devidamente registrado ou compartilhado [Borrego et al. 2019b]. Em muitos casos, essas informações permanecem na memória dos desenvolvedores, sendo transmitidas por meio de discussões informais ou não documentadas de maneira formal. À medida que o tempo passa e novos membros entram nas equipes, esse conhecimento se perde, criando uma desconexão entre o design original e o sistema em

constante evolução. Isso resulta em decisões mal informadas, que podem comprometer ainda mais a integridade da arquitetura.

O problema é ainda mais agudo em ambientes de desenvolvimento ágil e distribuído. Metodologias ágeis, como Scrum e Kanban, privilegiam a entrega rápida de funcionalidades e a adaptação constante às mudanças, muitas vezes em detrimento de um planejamento arquitetural mais profundo [VersionOne 2017]. Embora essas metodologias tragam agilidade e flexibilidade, elas também podem resultar em decisões arquiteturais tomadas rapidamente, sem considerar adequadamente seus impactos a longo prazo [Li et al. 2022]. A falta de documentação formal em ambientes ágeis — muitas vezes considerada como uma barreira para a velocidade de desenvolvimento — contribui ainda mais para a degradação da arquitetura [Sohan et al. 2016].

Outro desafio significativo nesse contexto é o desenvolvimento distribuído, onde equipes localizadas em diferentes regiões geográficas precisam colaborar por meio de ferramentas digitais, como e-mails, mensagens instantâneas e reuniões virtuais. Embora essas ferramentas facilitem a comunicação, elas frequentemente falham em capturar o contexto completo das decisões arquiteturais e, mais importante, em preservar esse conhecimento de maneira organizada e acessível para toda a equipe [Farenhorst and de Boer 2010]. Decisões críticas que deveriam ser documentadas formalmente acabam sendo feitas em conversas informais, resultando em uma perda de conhecimento fundamental para a continuidade do projeto [Lewis et al. 2021].

1.3.1 Decisões Intertemporais

Para garantir que decisões intertemporais sejam devidamente analisadas e rastreadas ao longo do tempo, ferramentas como os Registros de Decisões Arquiteturais (RDAs) tornam-se indispensáveis. Decisões intertemporais são aquelas que consideram tanto os benefícios imediatos quanto os impactos a longo prazo para a arquitetura do sistema. Em contextos de desenvolvimento ágil, essas decisões ajudam a equilibrar a necessidade de entregas rápidas com a sustentabilidade da arquitetura [Martínez-Fernández et al. 2020]. Por exemplo, optar por uma solução temporária pode atender a um prazo de curto prazo, mas resultar em uma dívida arquitetural significativa no futuro [Bosch 2000]. Assim, a análise intertemporal promove escolhas mais informadas, baseadas em uma avaliação cuidadosa de trade-offs.

Ao considerar decisões intertemporais, é fundamental incorporar práticas que incentivem a documentação detalhada do raciocínio por trás de cada escolha. Isso não apenas auxilia na preservação do conhecimento arquitetural, mas também facilita revisões futuras e adaptações conforme o sistema evolui [Lewis et al. 2021]. Em ambientes distribuídos, essa abordagem se torna ainda mais relevante, pois a dispersão geográfica das equipes pode dificultar a comunicação direta e levar a interpretações divergentes das escolhas arquiteturais [Serban and Visser 2022].

1.3.2 Registros de Decisões Arquiteturais (RDAs)

Os Registros de Decisões Arquiteturais (RDAs, do inglês *Architecture Decision Records* - ADR) são documentos formais que registram cada decisão arquitetural importante tomada ao longo do desenvolvimento. Cada RDA descreve a decisão, suas justificativas, os impactos

esperados e as alternativas consideradas. O uso de RDAs é uma prática recomendada para mitigar a vaporização do conhecimento arquitetural e preservar o histórico das decisões de design do sistema [Farenhorst and de Boer 2010].

Além de melhorar a rastreabilidade, os RDAs permitem uma análise retrospectiva mais eficiente, identificando padrões de decisão que podem ser aplicados em projetos futuros. Por exemplo, ao documentar trade-offs e lições aprendidas, equipes conseguem otimizar processos e evitar erros recorrentes [Kruchten 2004]. Em ambientes ágeis e distribuídos, os RDAs desempenham um papel ainda mais crítico, pois oferecem uma base comum de referência para todos os membros da equipe, independentemente de sua localização ou momento de ingresso no projeto [Sohan et al. 2016].

1.3.3 Condensação do Conhecimento Arquitetural

A condensação do conhecimento arquitetural é uma prática de captura e organização do CA de forma eficiente e acessível, geralmente por meio de ferramentas como o ArchiKCo e técnicas de marcação social. Esta prática visa mitigar a vaporização do CA em ambientes ágeis e distribuídos, proporcionando um método leve e contínuo para registrar e recuperar o conhecimento arquitetural sem sobrecarregar a equipe com documentação formal [Lewis et al. 2021].

A condensação do conhecimento é particularmente útil em projetos de longa duração, onde a rotatividade de membros da equipe e a evolução contínua do sistema tornam o acesso rápido e organizado ao CA uma necessidade crítica. Ferramentas como o ArchiKCo facilitam a navegação por repositórios de conhecimento, enquanto práticas de documentação incremental garantem que o CA seja atualizado regularmente [Lewis et al. 2021]. Essa abordagem, aliada a um esforço colaborativo de documentação, promove a sustentabilidade do sistema ao longo de todo o seu ciclo de vida.

1.4 Conexão entre os Conceitos

Os conceitos apresentados estão interligados, formando uma base para compreender a degradação arquitetural e as estratégias de mitigação. A degradação arquitetural é agravada pela vaporização do conhecimento arquitetural, que dificulta a rastreabilidade de decisões e aumenta a dívida arquitetural. Ferramentas como RDAs e práticas de condensação do conhecimento ajudam a preservar e organizar o conhecimento arquitetural. Em ambientes ágeis e distribuídos, decisões intertemporais e documentação robusta são essenciais para garantir a sustentabilidade e a qualidade do sistema.

A dívida arquitetural, resultante de decisões rápidas e temporárias para atender a prazos curtos, acumula custos futuros, tornando o sistema mais difícil de manter e expandir, comprometendo sua flexibilidade e produtividade das equipes [Bosch 2000]. Este estudo busca responder: **como mitigar a degradação da arquitetura de software em ambientes ágeis e distribuídos, reduzindo a vaporização do conhecimento arquitetural e o acúmulo de dívida arquitetural?**

A degradação arquitetural impacta a integração de novos componentes, aumenta a complexidade e a vulnerabilidade do sistema, comprometendo sua qualidade e a

eficiência das equipes. Este estudo investiga como decisões intertemporais podem equilibrar entregas rápidas com a manutenção de uma arquitetura robusta e sustentável [Martínez-Fernández et al. 2020].

1.5 Objetivo do Estudo

O objetivo principal deste estudo é investigar como decisões intertemporais podem ser aplicadas para mitigar a degradação da arquitetura de software, especialmente em ambientes ágeis e distribuídos. Ao longo do ciclo de vida de um sistema de software, a arquitetura frequentemente passa por um processo de degradação, que pode resultar na perda de conhecimento crítico e na redução da qualidade do sistema. Essa degradação é exacerbada pela vaporização do conhecimento arquitetural, onde decisões importantes sobre o design da arquitetura não são formalmente documentadas ou compartilhadas adequadamente, levando a uma desconexão entre a intenção original da arquitetura e o estado atual do sistema [Borrego et al. 2019b].

Para contextualizar e validar os objetivos deste estudo, serão utilizados dois estudos de caso: o projeto **OBCOOP** e as **RDA**, estas últimas sendo objeto de estudo no trabalho de conclusão de curso do aluno Enzo Paiva. As RDAs são uma prática recomendada para registrar formalmente as decisões arquiteturais ao longo do desenvolvimento de um sistema de software, promovendo a preservação do conhecimento arquitetural e a rastreabilidade das decisões [Farenhorst and de Boer 2010].

O **OBCOOP** é um aplicativo móvel desenvolvido pela Universidade Federal do Mato Grosso do Sul, com o objetivo de conectar fornecedores e consumidores, facilitando transações comerciais e a negociação de produtos diretamente entre as partes. Como o OBCOOP evoluiu ao longo do tempo, enfrentou desafios significativos relacionados à degradação de sua arquitetura, principalmente devido à pressão por entregas rápidas e à natureza ágil e distribuída da equipe de desenvolvimento. A falta de documentação formal e a comunicação informal das decisões arquiteturais resultaram em dificuldades na manutenção e evolução do sistema, gerando aumento da complexidade e perda de clareza no design original.

Já o estudo das **RDAs** se concentra na aplicação de registros formais de decisões arquiteturais, utilizados como uma prática recomendada para mitigar a vaporização do conhecimento arquitetural. No trabalho de Enzo Paiva, as RDAs foram empregadas para capturar as decisões arquiteturais ao longo do tempo, reduzindo a perda de conhecimento e promovendo a rastreabilidade e clareza das decisões em sistemas complexos e em constante evolução [Farenhorst and de Boer 2010].

Assim, este estudo tem como objetivo investigar como as decisões intertemporais podem ser aplicadas para mitigar a degradação arquitetural no contexto de projetos como o OBCOOP e as RDAs. As decisões intertemporais são aquelas que equilibram benefícios de curto prazo com impactos de longo prazo, considerando os trade-offs envolvidos em decisões rápidas que podem comprometer a sustentabilidade do sistema no futuro. A partir da análise desses dois estudos de caso, serão propostas estratégias para preservar o conhecimento arquitetural e evitar o acúmulo de "dívida arquitetural" — o custo futuro associado a decisões arquiteturais mal fundamentadas ou incompletas [Bosch 2000].

Além de identificar os fatores que contribuem para a degradação da arquitetura nos dois casos, este estudo também busca propor soluções que possam ser generalizadas para outros

projetos em ambientes ágeis e distribuídos. Entre essas soluções, estão:

- **Documentação eficaz e contínua**: Ferramentas e métodos que permitam capturar e preservar decisões arquiteturais de forma estruturada e acessível, evitando a perda de conhecimento ao longo do tempo [Dalkir 2011, Lewis et al. 2021].
- **Comunicação estruturada**: Adoção de práticas que garantam que a comunicação entre as equipes, especialmente em ambientes distribuídos, seja formalizada e documentada adequadamente, minimizando o impacto da comunicação informal e assíncrona [Serban and Visser 2022].
- **Decisões intertemporais**: Desenvolvimento de frameworks e práticas que ajudem as equipes a equilibrar a necessidade de decisões rápidas com a necessidade de preservar a integridade arquitetural a longo prazo [Martínez-Fernández et al. 2020].

Ao validar essas propostas no contexto do OBCOOP e das RDAs, espera-se que este estudo forneça insights valiosos sobre como equipes de desenvolvimento podem mitigar a degradação arquitetural em seus projetos, garantindo a sustentabilidade e a qualidade do software ao longo do tempo.

1.6 Metodologia de Pesquisa

Este estudo adota uma abordagem exploratória, combinando uma revisão bibliográfica não sistemática com a análise de dois estudos de caso práticos: o projeto **OBCOOP** e o uso de **RDA** (Registros de Decisões Arquiteturais, do inglês "Architecture Decision Records" - ADR), utilizados no trabalho de conclusão de curso de Enzo Paiva.

O projeto **OBCOOP** será analisado em profundidade para identificar como a falta de documentação formal e a comunicação informal nas equipes de desenvolvimento ágeis e distribuídas contribuiu para a degradação da arquitetura do sistema. Serão realizadas entrevistas com os desenvolvedores, análise documental e observação direta do projeto para coletar informações sobre as decisões arquiteturais e o impacto dessas decisões na evolução do sistema. O objetivo é entender os desafios enfrentados pelo OBCOOP e propor soluções baseadas em decisões intertemporais que possam ser aplicadas para mitigar a degradação arquitetural [Borrego et al. 2019b]. A Figura 3 apresenta uma representação teórica da metodologia adotada para manter a integridade do conhecimento arquitetural.

Figure 3: Metodologia teórica para manter a integridade do conhecimento arquitetural.

Além disso, será utilizado o estudo de caso dos **RDAs**, que foram analisados no trabalho de conclusão de curso de Enzo Paiva, para investigar como o uso de registros formais de decisões arquiteturais pode ajudar a preservar o conhecimento arquitetural ao longo do tempo. A prática dos RDAs será analisada como uma solução complementar que facilita a rastreabilidade e a clareza das decisões arquiteturais, reduzindo a perda de conhecimento e promovendo uma arquitetura mais sustentável [Farenhorst and de Boer 2010].

Esses dois estudos de caso fornecerão uma base sólida para validar as estratégias propostas no estudo, que incluem:

- **Revisão bibliográfica não sistemática.**: A literatura sobre degradação arquitetural, vaporização do conhecimento arquitetural e decisões intertemporais será revisada para fornecer uma base teórica para o estudo [Li et al. 2022, Lewis et al. 2021].
- **Entrevistas semiestruturadas**: Desenvolvedores envolvidos nos projetos do OBCOOP e nas RDAs serão entrevistados para compreender suas experiências e os desafios enfrentados na preservação do conhecimento arquitetural.
- **Análise documental**: Documentos e registros do OBCOOP e dos RDAs analisados para identificar lacunas de documentação e práticas que possam ser aprimoradas visando maior organização e qualidade no projeto.
- **Estudo participativo**: A partir dos dados coletados, será feita uma análise detalhada das práticas adotadas nos dois projetos e das possíveis soluções para melhorar a governança arquitetural e evitar a degradação [Bosch 2000].

Além disso, a revisão bibliográfica não sistemática ajudará a identificar as melhores práticas e estratégias que têm sido adotadas por outros projetos para lidar com a

degradação arquitetural e a vaporização do conhecimento arquitetural. Essas práticas incluem a adoção de documentação eficaz e contínua, comunicação estruturada, decisões intertemporais, e uma governança arquitetural ativa que visa equilibrar a flexibilidade das metodologias ágeis com a necessidade de preservação arquitetural de longo prazo [Dalkir 2011, Serban and Visser 2022].

Por fim, as propostas derivadas dos estudos de caso e da revisão bibliográfica não sistemática serão validadas em um contexto prático, com recomendações específicas sobre como aplicar as estratégias discutidas para mitigar a degradação arquitetural e preservar o conhecimento arquitetural, especialmente em ambientes ágeis e distribuídos [Farenhorst and de Boer 2010, Borrego et al. 2019b].

2 Referencial Teórico

2.1 Introdução

Este estudo investiga a degradação arquitetural e a vaporização do conhecimento arquitetural em ambientes ágeis e distribuídos. A degradação arquitetural é um problema recorrente, resultante de modificações incrementais que comprometem a arquitetura original de um sistema, acumulando o que é conhecido como "dívida arquitetural" [Borrego et al. 2019a]. Além disso, a falta de uma documentação formal de CA contribui para a vaporização do conhecimento arquitetural, na qual informações críticas sobre o design e as decisões arquiteturais são perdidas ou esquecidas ao longo do tempo, especialmente em equipes distribuídas [Farenhorst and de Boer 2010]. Esses dois problemas estão profundamente conectados, pois a perda de conhecimento arquitetural frequentemente acelera a degradação do sistema, criando um ciclo difícil de interromper.

2.2 Degradação Arquitetural

A degradação arquitetural refere-se ao processo de deterioração da estrutura do sistema ao longo do tempo, causado principalmente por modificações incrementais que visam resolver problemas imediatos, mas que, ao mesmo tempo, comprometem a coesão e a manutenibilidade da arquitetura [Li et al. 2022, Bosch 2000, Martínez-Fernández et al. 2020, Jansen and Bosch 2020]. Além disso, decisões de curto prazo feitas sob pressão para atender a prazos ou requisitos emergentes muitas vezes acumulam o que é conhecido como "dívida arquitetural", aumentando os custos de manutenção e dificultando a evolução do sistema [Bosch 2000]. Essa situação se agrava pela falta de práticas formais de documentação, conectando diretamente a degradação à vaporização do conhecimento arquitetural.

2.3 Conhecimento Arquitetural (CA)

O conhecimento arquitetural é o conjunto de decisões, justificativas e trade-offs que moldam o design e a estrutura de um sistema de software. Ele inclui as escolhas de design, restrições e objetivos de qualidade que influenciam a arquitetura. Além de ser fundamental para o

entendimento e a evolução de sistemas complexos, o CA desempenha um papel crucial na mitigação da degradação arquitetural, pois decisões informadas e bem documentadas podem reduzir o acúmulo de dívida arquitetural [Farenhorst and de Boer 2010]. Assim, preservar o CA é essencial, especialmente em projetos de longa duração ou em ambientes ágeis e distribuídos, onde o conhecimento sobre o sistema pode se dispersar rapidamente.

2.4 Conceitos Básicos

Esta seção apresenta as principais entidades e objetos mencionados neste estudo, os quais são fundamentais para compreender a degradação arquitetural e as estratégias para mitigar a vaporização do conhecimento arquitetural.

2.4.1 Vaporização do Conhecimento Arquitetural (VCA)

A vaporização do conhecimento arquitetural ocorre quando o conhecimento sobre decisões e a estrutura do sistema se perdem devido à falta de documentação ou à comunicação informal [Borrego et al. 2019b, Farenhorst and de Boer 2010, Avgeriou et al. 2008, Sohan et al. 2016]. Esse fenômeno é particularmente comum em metodologias ágeis, onde a ênfase na flexibilidade e na entrega rápida frequentemente reduz a prioridade dada à documentação formal [Borrego et al. 2019a, Lewis et al. 2021]. A VCA agrava a degradação arquitetural, dificultando a rastreabilidade de decisões passadas e aumentando a complexidade de futuras modificações no sistema.

O gráfico apresentado na Figura 4 ilustra a redução da retenção de conhecimento em função do tempo, destacando a natureza exponencial da vaporização quando práticas inadequadas de documentação e comunicação persistem. Conforme apontado por Lima e Gosling (2005) em seu estudo sobre a espiral do conhecimento, a retenção de informações depende de processos contínuos de captura, organização e compartilhamento do conhecimento. A ausência desses processos acelera a perda de informações críticas, especialmente em equipes distribuídas ou em ambientes de alta rotatividade, como frequentemente encontrado em projetos de software.

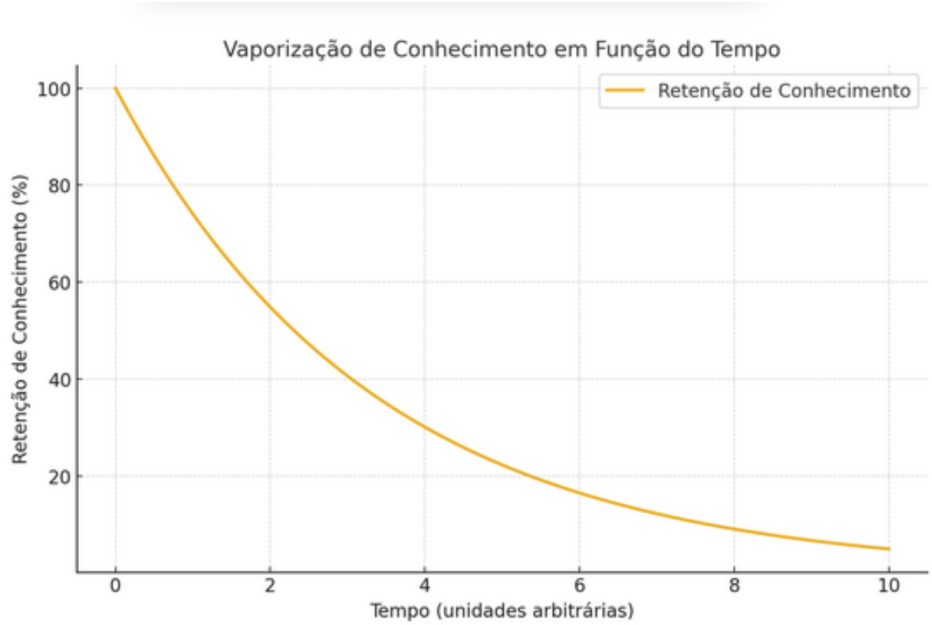

Figure 4: Redução da retenção de conhecimento em função do tempo

2.4.2 Decisões Arquiteturais

Uma das consequências mais críticas da vaporização do conhecimento arquitetural é a dificuldade em rastrear e compreender as decisões arquiteturais tomadas, essenciais para a evolução do sistema. As decisões arquiteturais são escolhas críticas realizadas durante o design do sistema que afetam sua estrutura e comportamento. Cada decisão inclui justificativas, impactos e possíveis alternativas. Em muitos casos, essas decisões são documentadas em registros conhecidos como RDAs (Registros de Decisões Arquiteturais, do inglês *Architecture Decision Records* - ADR), que ajudam a preservar e rastrear o conhecimento sobre o histórico de design do sistema [Farenhorst and de Boer 2010].

2.4.3 Documentação em Ambientes Ágeis

Embora a documentação formal seja frequentemente vista como uma barreira à agilidade, ela é fundamental para mitigar a degradação e a vaporização do conhecimento arquitetural. Práticas de documentação eficazes são críticas em ambientes ágeis, onde a flexibilidade e a comunicação contínua são priorizadas [VersionOne 2017]. Isso pode levar à perda de conhecimento crítico, dificultando a rastreabilidade de decisões arquiteturais e o entendimento da evolução do sistema [Dalkir 2011]. Além disso, técnicas como a marcação social (*social tagging*) e o uso de RDAs podem facilitar a recuperação e organização do conhecimento em equipes distribuídas [Lewis et al. 2021].

2.4.4 Documentação em Ambientes Distribuídos

Embora os desafios da documentação sejam significativos em ambientes ágeis, eles são ampliados em equipes distribuídas, onde barreiras geográficas e culturais adicionam novas ca-

madas de complexidade. Ambientes distribuídos, onde equipes estão localizadas geograficamente em diferentes regiões, representam desafios únicos para a documentação e preservação do conhecimento arquitetural. Estudos mostram que a centralização da documentação em repositórios acessíveis, como Notion ou Confluence, combinada com o uso de RDAs, pode mitigar significativamente essas perdas [Kruchten 2004, Farenhorst and de Boer 2010]. Em suma, práticas de documentação contínua e colaborativa são essenciais para preservar o CA em ambientes distribuídos [Lewis et al. 2021].

2.4.5 Dívida Arquitetural

A dívida arquitetural é um conceito relacionado ao custo futuro associado a decisões arquiteturais rápidas ou inadequadas, muitas vezes tomadas sob pressão de prazos curtos. Essas decisões, quando não acompanhadas de práticas adequadas de documentação e manutenção, podem comprometer a capacidade de adaptação e expansão do sistema ao longo do tempo [Bosch 2000].

Assim como a vaporização do conhecimento arquitetural (VCA) dificulta a rastreabilidade, a dívida arquitetural agrava os desafios de manutenção ao introduzir complexidades desnecessárias na arquitetura do sistema. Essas complexidades podem incluir acoplamento excessivo entre módulos, dependências não planejadas e violações de princípios arquiteturais, dificultando a evolução do sistema e aumentando significativamente os custos de manutenção.

Se não gerida adequadamente, a dívida arquitetural tende a crescer de forma acumulativa, resultando em uma redução exponencial na eficiência do desenvolvimento e na escalabilidade do sistema. Portanto, a implementação de práticas contínuas de refatoração, monitoramento técnico e documentação estruturada é essencial para mitigar seus efeitos, garantindo a sustentabilidade arquitetural a longo prazo.

2.4.6 Ambientes Ágeis e Distribuídos (AAD)

A preservação do conhecimento arquitetural torna-se ainda mais crítica em ambientes ágeis e distribuídos, onde as equipes precisam lidar com desafios adicionais de colaboração e comunicação em larga escala. Esses ambientes privilegiam a comunicação assíncrona e não estruturada, o que contribui para a fragmentação do conhecimento arquitetural (CA) e, muitas vezes, dificulta a documentação formal [Farenhorst and de Boer 2010, Serban and Visser 2022].

2.4.7 Decisões Intertemporais

Decisões intertemporais referem-se a escolhas arquiteturais que equilibram benefícios imediatos e impactos de longo prazo no sistema. Em metodologias ágeis, essas decisões são cruciais para atender às demandas de entregas rápidas sem comprometer a sustentabilidade da arquitetura [Martínez-Fernández et al. 2020]. Por exemplo, uma solução temporária pode viabilizar um prazo de curto prazo, mas acarretar uma dívida arquitetural significativa no futuro [Bosch 2000]. A análise intertemporal, portanto, fomenta escolhas mais informadas, baseadas em uma avaliação criteriosa de trade-offs entre desempenho, custo e manutenção.

Para garantir a eficácia das decisões intertemporais, ferramentas como os Registros de Decisões Arquiteturais (RDAs) desempenham um papel essencial. Elas não apenas documentam o raciocínio por trás de cada escolha, mas também facilitam revisões futuras e adaptações

conforme o sistema evolui [Lewis et al. 2021]. Em equipes distribuídas, essa prática é ainda mais relevante, uma vez que a dispersão geográfica pode dificultar a comunicação direta, aumentando o risco de interpretações divergentes [Serban and Visser 2022].

2.4.8 Registros de Decisões Arquiteturais (RDAs)

Os Registros de Decisões Arquiteturais (RDAs, do inglês *Architecture Decision Records* - ADR) são documentos estruturados que registram decisões arquiteturais significativas tomadas ao longo do desenvolvimento. Cada RDA inclui a descrição da decisão, justificativas, impactos esperados e alternativas consideradas. Essa prática é essencial para mitigar a vaporização do conhecimento arquitetural e manter o histórico das decisões de design [Farenhorst and de Boer 2010].

Além de melhorar a rastreabilidade, os RDAs permitem uma análise retrospectiva eficiente, identificando padrões e lições aprendidas que podem ser aplicadas em projetos futuros. Em ambientes ágeis e distribuídos, eles oferecem uma base de referência comum para todos os membros da equipe, independentemente de sua localização ou tempo de ingresso no projeto [Sohan et al. 2016]. Isso os torna especialmente úteis em equipes com alta rotatividade, pois ajudam a preservar o conhecimento arquitetural e evitar dependências excessivas de indivíduos específicos.

Por fim, ao registrar trade-offs e decisões intertemporais de forma clara e consistente, os RDAs promovem maior transparência, reduzem erros recorrentes e contribuem para a resiliência e evolução contínua do sistema [Kruchten 2004].

2.4.9 Condensação do Conhecimento Arquitetural

A condensação do conhecimento arquitetural (CA, do inglês *Architectural Knowledge* - AK) é uma prática de captura e organização do CA de forma eficiente e acessível, geralmente por meio de ferramentas como o ArchiKCo e técnicas de marcação social. Esta prática visa mitigar a vaporização do CA em ambientes ágeis e distribuídos, proporcionando um método leve e contínuo para registrar e recuperar o conhecimento arquitetural sem sobrecarregar a equipe com documentação formal [Lewis et al. 2021].

A condensação do conhecimento é particularmente útil em projetos de longa duração, onde a rotatividade de membros da equipe e a evolução contínua do sistema tornam o acesso rápido e organizado ao CA uma necessidade crítica. Ferramentas como o ArchiKCo facilitam a navegação por repositórios de conhecimento, enquanto práticas de documentação incremental garantem que o CA seja atualizado regularmente [Lewis et al. 2021]. Essa abordagem, aliada a um esforço colaborativo de documentação, promove a sustentabilidade do sistema ao longo de todo o seu ciclo de vida.

3 Algoritmo ReduceKnowledgeVaporization

Esta seção descreve o algoritmo *ReduceKnowledgeVaporization*, projetado para reduzir a vaporização do conhecimento arquitetural em equipes ágeis e distribuídas. O algoritmo tem como objetivo capturar, organizar e preservar informações arquiteturais de forma eficiente,

promovendo uma melhor gestão do conhecimento e comunicação entre as equipes de desenvolvimento [Lewis et al. 2021]. É importante ressaltar que a aplicação desse algoritmo comportamental não exige a obrigatoriedade de seguir todos os passos rigidamente, podendo ser adaptado e flexibilizado conforme a cultura e o modelo organizacional da equipe. A lógica do algoritmo é representada na Figura 5 por uma espiral, simbolizando sua natureza cíclica e contínua: a cada nova fase de maturidade do projeto, as etapas são executadas novamente para garantir que o conhecimento arquitetural seja mantido e o projeto permaneça organizado ao longo do tempo.

Figure 5: Modelo comportamental para lidar com a degradação da arquitetura de software.

Algorithm 1 *ReduceKnowledgeVaporization* (Parte 1)

Data: Projeto de desenvolvimento de software em ambiente ágil e distribuído
Result: Redução da vaporização do conhecimento arquitetural e aumento da qualidade do produto final

Etapa 1: Inicialização
1. Configurar ferramentas de comunicação e colaboração (ex: Slack, Microsoft Teams, Trello, Jira, Confluence)
2. Estabelecer um repositório central de documentação (ex: Notion)
3. Definir templates e padrões de comunicação e documentação
4. Implementar treinamentos sobre boas práticas de documentação e comunicação

Etapa 2: Início do Projeto
1. Criar canais específicos nas ferramentas de comunicação para tópicos do projeto (ex: arquitetura, design, suporte)
2. Configurar espaços de projeto em ferramentas de gerenciamento de tarefas e documentação (ex: Trello, Jira)
3. Designar responsáveis pela documentação e rotacioná-los para evitar a centralização do conhecimento

Etapa 3: Documentação de Requisitos
1. Registrar atas de reuniões de levantamento de requisitos no repositório central
2. Associar requisitos às decisões arquiteturais e refletir no backlog
3. Manter o sistema de gerenciamento de tarefas atualizado com mudanças nos requisitos

Etapa 4: Comunicação Eficaz
1. Registrar discussões decisivas nas ferramentas de comunicação e transcrevê-las para o repositório de conhecimento
2. Atualizar o sistema de gerenciamento de tarefas com mudanças e decisões arquiteturais importantes
3. Documentar decisões usando documentos curtos e diretos

Etapa 5: Prática de Documentação Incremental
1. Estabelecer checkpoints regulares para revisar e atualizar a documentação ao longo do projeto
2. Associar documentos a *pull requests* ou *issues* para facilitar a rastreabilidade

Etapa 6: Produção de Artefatos
1. Criar documentos de design técnico no repositório de conhecimento, acessíveis a toda a equipe
2. Associar commits de código a tarefas e documentos no sistema de gerenciamento de tarefas para rastreabilidade
3. Incluir referências aos documentos de design nos comentários do código

Etapa 7: Revisão e Refinamento
1. Realizar revisões colaborativas de código e design, garantindo que as mudanças estejam na documentação
2. Manter a documentação sempre atualizada com o progresso do projeto
3. Adicionar melhorias e insights aos documentos após cada revisão

Algorithm 2 *ReduceKnowledgeVaporization* (Parte 2)

Data: Projeto de desenvolvimento de software em ambiente ágil e distribuído
Result: Redução da vaporização do conhecimento arquitetural e aumento da qualidade do produto final

Etapa 9: Armazenamento e Acesso
1. Armazenar toda a documentação em um repositório central de fácil acesso
2. Definir permissões adequadas para facilitar o acesso sem comprometer a segurança
3. Estruturar as informações com boas práticas de nomenclatura e tags

Etapa 11: Treinamento Contínuo
1. Realizar sessões de treinamento regulares sobre práticas de documentação e comunicação
2. Designar mentores para auxiliar novos membros na adaptação

Etapa 12: Validação Contínua
1. Realizar auditorias periódicas para verificar a conformidade com os padrões estabelecidos
2. Coletar feedback contínuo da equipe sobre as práticas de documentação
3. Estabelecer métricas para monitorar a qualidade da documentação

Etapa 13: Encerramento do Projeto
1. Realizar uma sessão de encerramento documentando todas as decisões arquiteturais e lições aprendidas
2. Compilar e arquivar um relatório final no repositório central para consultas futuras
3. Disponibilizar toda a documentação para uso em projetos futuros

4 Estudo de Caso Participativo

4.1 Introdução

Esta seção explora o estudo de caso participativo do projeto OBCOOP, onde o algoritmo *ReduceKnowledgeVaporization* foi aplicado para mitigar a degradação arquitetural e aumentar a retenção de conhecimento dentro da equipe. Com a implementação de práticas específicas voltadas para melhorar a comunicação, descentralizar o conhecimento e promover uma documentação mais robusta, os membros da equipe relataram mudanças significativas nas dinâmicas de trabalho. As percepções individuais dos participantes foram analisadas para identificar tanto os pontos fortes quanto as áreas que ainda requerem melhorias.

4.2 Questão de Pesquisa

A questão de pesquisa central deste estudo de caso é: *Como a aplicação do algoritmo ReduceKnowledgeVaporization impacta a retenção de conhecimento arquitetural e a eficiência das práticas de desenvolvimento em ambientes ágeis e distribuídos?* Em particular, busca-se entender como a centralização da documentação, a descentralização de responsabilidades e a introdução de práticas de comunicação estruturadas afetam o alinhamento e a sustentabilidade do conhecimento arquitetural dentro da equipe. Esta questão reflete a necessidade de estratégias que reduzam a vaporização do conhecimento em projetos distribuídos, como sugerido por Borrego et al. (2019) [Borrego et al. 2019b].

4.3 Execução e Coleta de Dados

A execução deste estudo de caso envolveu a implementação de diversas práticas do algoritmo *ReduceKnowledgeVaporization* no projeto OBCOOP. Foram realizadas entrevistas com par-

ticipantes identificados como P1, P2 e P3, que compartilharam suas percepções sobre as mudanças e os resultados observados.

Abaixo, é possível visualizar em uma tabela o perfil dos participantes entrevistados, incluindo suas funções no projeto, responsabilidades principais, tempo de experiência na área e tempo no projeto.

Table 1: Perfil dos participantes entrevistados.

Participante	Função no Projeto	Responsabilidade Principal	Tempo de Experiência na Área (anos)	Tempo no Projeto (meses)
P1	- Gerente de Projeto - Scrum Master - Desenvolvedor - Analista de Requisitos	- Gerenciar equipe de desenvolvimento - Administrar cerimônias Scrum - Manter comunicação com Stakeholders - Desenvolver telas e integrações - Criar e refinar requisitos	3	18
P2	- Arquiteto-líder	- Gerenciar o time de desenvolvimento de ADRs	6	6
P3	- Desenvolvedor backend - Arquiteto de software - Product Owner	- Desenvolver códigos no backend do projeto - Tomar decisões arquiteturais - Coletar requisitos	2	11

A análise do perfil dos participantes apresentado na Tabela 1 revela uma concentração significativa de funções no participante P1, que acumula múltiplas responsabilidades no projeto, como gerenciamento, desenvolvimento e análise de requisitos. Essa centralização de funções pode gerar desafios para a equipe, como a dificuldade em manter uma organização eficiente e a dependência excessiva de um único membro para várias atividades críticas. Além disso, essa sobrecarga de responsabilidades pode impactar negativamente o fluxo de trabalho, aumentando os riscos de gargalos e dificultando a distribuição de informações entre os integrantes do projeto. Portanto, fica evidente a necessidade de adotar estratégias para descentralizar responsabilidades e promover uma distribuição mais equilibrada das tarefas, garantindo maior colaboração e resiliência organizacional.

Para lidar com os desafios identificados, foi aplicada a lógica do algoritmo comportamental desenvolvido neste estudo, o *ReduceKnowledgeVaporization*. Esse algoritmo foi projetado para mitigar a centralização de informações e promover uma gestão mais equilibrada do conhecimento arquitetural. Sua aplicação no contexto do projeto permitiu a análise detalhada do comportamento dos integrantes e a introdução de práticas que visam descentralizar responsabilidades, melhorar a comunicação e facilitar o compartilhamento de informações entre os membros da equipe. Essa abordagem buscou garantir que o conhecimento crítico do projeto não ficasse restrito a um único participante, promovendo maior colaboração e resiliência no desenvolvimento, ao mesmo tempo em que evitava a perda de informações devido à sobrecarga de atribuições concentradas em apenas um indivíduo.

4.4 Problemas Relatados pelos Participantes

Durante a análise dos problemas levantados por P1, P2 e P3, foi possível identificar diversas dificuldades enfrentadas pela equipe no contexto do projeto. Essas dificuldades estão diretamente relacionadas aos desafios de comunicação, documentação e gestão de conhecimento arquitetural, agravados pela falta de práticas estruturadas antes da aplicação do algoritmo *ReduceKnowledgeVaporization*. A seguir, destacamos os principais problemas identificados:

- **Comunicação Ineficiente:**

  - Conforme relatado por P1 e P3, a comunicação interna da equipe era prejudicada pela ausência de reuniões síncronas e pela falta de canais apropriados para a troca de informações. Essa situação dificultava a resolução de impedimentos e o alinhamento entre os membros.
  - P2 destacou que a comunicação com agentes externos também era limitada, em parte devido à vaporização do conhecimento arquitetural, o que comprometia o entendimento claro do domínio do projeto.

- **Centralização de Conhecimento:**
  - P1 apontou a concentração de responsabilidades em poucos membros, sem a designação formal de revisores para as tarefas, o que dificultava a disseminação do conhecimento e aumentava os riscos de dependência de indivíduos específicos.
  - A falta de práticas de documentação colaborativa, como mencionado por P3, dificultava a criação de um ambiente de cooperação e transparência.

- **Documentação Inadequada:**
  - A equipe enfrentava problemas na organização de backlog, releases e requisitos, conforme relatado por P1, além de dificuldade em rastrear decisões e argumentos fornecidos nos debates, como destacado por P3.
  - A ausência de padrões claros de nomenclatura e tags tornava a busca por informações ineficiente, agravando a perda de decisões críticas para o projeto.

- **Perda de Decisões Arquiteturais:**
  - P3 destacou que as ADRs (Arquitetural Decision Records) não eram gerenciadas de forma eficaz, dificultando a revisão e rastreabilidade de decisões arquiteturais.
  - P2 mencionou que a alta vaporização do conhecimento arquitetural resultava na perda de decisões importantes e dificultava o alinhamento da equipe com o histórico do projeto.

- **Falta de Feedback Contínuo:**
  - A interrupção das Sprints Retro, conforme apontado por P1, limitava a oportunidade de revisão e melhorias no processo de trabalho, reduzindo o alinhamento da equipe e dificultando a identificação de problemas recorrentes.

- **Integração de Novos Membros:**
  - O onboarding insuficiente, citado por P1, gerava uma curva de aprendizado longa e baixa produtividade inicial para novos integrantes da equipe, impactando o ritmo de desenvolvimento.

Esses problemas refletem a necessidade de práticas estruturadas para descentralizar o conhecimento, melhorar a comunicação e criar um ambiente de colaboração mais eficiente. A

Tabela 6 apresenta os principais problemas levantados pelos participantes durante o estudo, categorizados de acordo com sua natureza e impacto no desenvolvimento do projeto.

A aplicação do algoritmo *ReduceKnowledgeVaporization* busca mitigar essas dificuldades ao promover a documentação contínua, a rastreabilidade das decisões e a organização da equipe de forma sustentável. Com base nos problemas identificados, o algoritmo foi adaptado para abordar as causas mais recorrentes, como a falta de centralização de informações, a dificuldade de rastrear decisões e os desafios de comunicação em equipes ágeis e distribuídas.

Problemas	P1	P2	P3
Comunicação Ineficiente	✔		✔
Centralização do Conhecimento	✔	✔	
Documentação Inadequada	✔	✔	
Perda de Decisões Arquiteturais		✔	✔
Falta de Feedbacks 1:1	✔		
Dificuldade de Integração de Novos Membros	✔		

Figure 6: Principais problemas identificados pelos participantes e seus impactos no processo de desenvolvimento.

O gráfico apresentado na Figura 7 destaca os principais problemas relatados pela equipe durante o desenvolvimento do projeto, ilustrando a quantidade de reclamações relacionadas a cada questão identificada. Dentre os problemas mais recorrentes estão a comunicação ineficiente, a centralização do conhecimento e a documentação inadequada, todos diretamente ligados à falta de práticas estruturadas para gerenciar o conhecimento arquitetural e promover a colaboração. Além disso, questões como a perda de decisões arquiteturais, a falta de feedbacks 1:1 e a dificuldade de integração de novos membros também evidenciam a necessidade de melhorias na gestão de processos e na estruturação das interações entre a equipe. O gráfico serve como base para a aplicação do algoritmo *ReduceKnowledgeVaporization*, visando mitigar essas dificuldades e promover um ambiente mais colaborativo e eficiente.

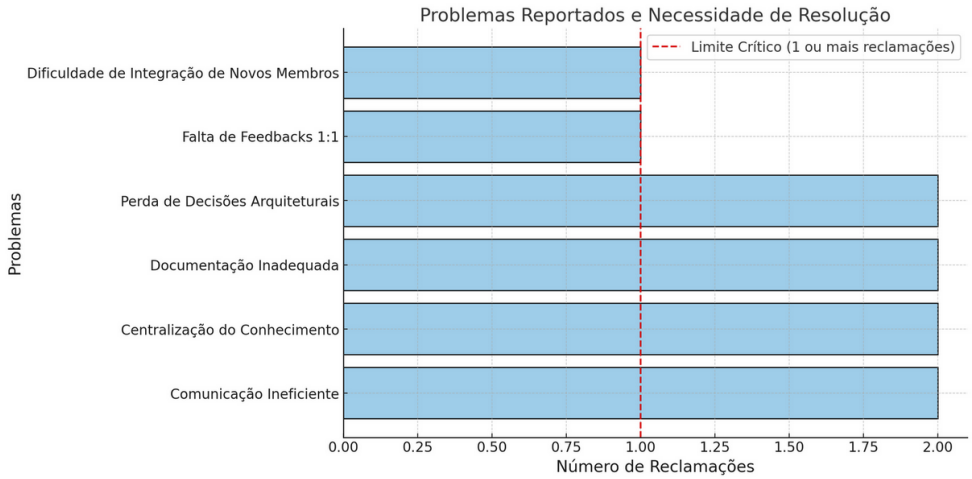

Figure 7: Enter Caption

5 Aplicação do Algoritmo

A aplicação do algoritmo *ReduceKnowledgeVaporization* foi conduzida com o objetivo de resolver os problemas identificados na comunicação, documentação e gestão do conhecimento arquitetural. A implementação foi realizada de forma parcial e iterativa, ou seja, as etapas do algoritmo foram sendo adaptadas e formalizadas conforme os feedbacks dos integrantes da equipe retornavam respostas positivas e evidências de melhorias no processo. Essa abordagem incremental permitiu ajustar o algoritmo às necessidades específicas da equipe e ao contexto organizacional, garantindo maior aderência e eficácia.

O foco principal da aplicação foi na análise comportamental e qualitativa do projeto em um contexto geral. Além de resolver problemas técnicos, buscou-se entender como o comportamento dos integrantes da equipe impactava o engajamento, a colaboração e a preservação do conhecimento arquitetural. O objetivo era não apenas organizar informações, mas também fomentar uma cultura de comunicação clara e rastreável.

Para a equipe do projeto **OBCOOP**, o algoritmo foi aplicado ao longo de 2 meses. Nesse período, o grupo trabalhou em sprints de duas semanas, totalizando 4 ciclos completos de desenvolvimento nos quais as mudanças comportamentais propostas pelo algoritmo foram testadas e refinadas. Durante cada sprint, as decisões arquiteturais foram registradas e revisadas, e as ferramentas utilizadas para documentação e comunicação foram ajustadas com base nas necessidades identificadas. Esse processo cíclico possibilitou a identificação de gargalos, a implementação de melhorias e a validação contínua da eficácia do algoritmo no ambiente ágil e distribuído.

5.1 Expectativas com a Aplicação do Algoritmo

As principais expectativas com a aplicação do algoritmo incluíam:

- **Melhoria na Comunicação:** Criar canais mais eficientes para troca de informações, tanto internos quanto com agentes externos, promovendo a sincronia da equipe e reduzindo a ocorrência de impedimentos não resolvidos.
- **Descentralização do Conhecimento:** Implementar práticas que evitassem a concentração de responsabilidades em poucos integrantes, promovendo uma distribuição mais equilibrada das tarefas e incentivando a colaboração.
- **Documentação Eficiente:** Estabelecer padrões claros de nomenclatura, tags e templates para facilitar a busca, o registro e o acesso às informações importantes, garantindo a rastreabilidade das decisões.
- **Gestão de Decisões Arquiteturais:** Melhorar a criação, organização e revisão de ADRs (*Architectural Decision Records*), garantindo o registro formal das decisões arquiteturais e facilitando a compreensão do histórico.
- **Feedback Contínuo:** Reintroduzir cerimônias como as Sprints Retro para criar um espaço estruturado para revisão, feedback e melhoria contínua do processo de trabalho.

- **Integração de Novos Membros:** Tornar o processo de onboarding mais eficiente, reduzindo a curva de aprendizado e aumentando a produtividade inicial de novos integrantes da equipe.

5.2 Resultados Após a Aplicação do Algoritmo

Após a aplicação do algoritmo, os resultados atenderam grande parte das expectativas e refletiram nas percepções dos participantes:

- **Comunicação:** A implementação de dailys síncronas no Meet, como destacado por P1, trouxe maior comprometimento da equipe e agilidade na resolução de impedimentos. P2 corroborou que a comunicação com agentes externos melhorou devido a um entendimento mais claro do domínio do projeto, possibilitado pela redução da vaporização do conhecimento.

- **Descentralização do Conhecimento:** A designação obrigatória de revisores para as tarefas, mencionada por P1, resultou em uma melhor distribuição de responsabilidades e na diminuição da centralização do conhecimento. P3 apontou que isso criou um ambiente mais colaborativo e transparente, facilitando o compartilhamento de informações.

- **Documentação:** A adoção de templates padronizados e boas práticas de nomenclatura e tags no Notion, como relatado por P1, organizou o repositório de documentos, tornando a busca por informações mais eficiente. P3 destacou que a criação de documentos mais acessíveis ajudou toda a equipe a compreender melhor o projeto.

- **Gestão de Decisões Arquiteturais:** A introdução de ADRs possibilitou o registro formal e estruturado das decisões arquiteturais, conforme relatado por P3. Esse processo facilitou a rastreabilidade das escolhas feitas e melhorou a agilidade da equipe na revisão e no debate sobre novas decisões.

- **Feedback Contínuo:** A reintrodução das Sprints Retro, mencionada por P1, fortaleceu o alinhamento da equipe e proporcionou um espaço para coletar feedback e implementar melhorias contínuas.

- **Integração de Novos Membros:** O onboarding aprimorado encurtou a curva de aprendizado, aumentando a produtividade inicial dos novos integrantes, como observado por P1. Isso garantiu que novos membros fossem integrados mais rapidamente à equipe e ao fluxo de trabalho.

Esses resultados demonstram como a aplicação do algoritmo *ReduceKnowledgeVaporization* atendeu às expectativas iniciais, promovendo um ambiente de trabalho mais eficiente, colaborativo e organizado. A Tabela 8 apresenta as menções feitas pelos participantes (P1, P2 e P3) sobre a efetividade das soluções implementadas após a aplicação do algoritmo. As principais dificuldades iniciais, como comunicação ineficiente, centralização do conhecimento e documentação inadequada, foram consistentemente apontadas como mitigadas por todos os participantes, indicando que as estratégias aplicadas impactaram positivamente a

dinâmica da equipe e o fluxo de trabalho. Além disso, outras questões, como a perda de decisões arquiteturais e a integração de novos membros, também foram mencionadas como áreas com melhorias significativas.

Mensão de Solução	P1	P2	P3
Comunicação Ineficiente	✔	✔	✔
Centralização do Conhecimento	✔	✔	✔
Documentação Inadequada	✔	✔	✔
Perda de Decisões Arquiteturais	✔	✔	✔
Falta de Feedbacks e 1:1	✔		✔
Dificuldade de Integração de Novos Membros	✔		

Figure 8: Menção sobre solução pós-aplicação do algoritmo por participantes P1, P2 e P3.

O gráfico apresentado na Figura 9 destaca os problemas identificados inicialmente no projeto e o número de participantes que confirmaram sua resolução após a aplicação do algoritmo *ReduceKnowledgeVaporization*. Cada problema foi avaliado pelos integrantes da equipe, e a meta de resolução foi considerada atingida para problemas que receberam ao menos duas confirmações de melhoria. Entre os problemas mais críticos que foram amplamente solucionados estão a comunicação ineficiente, a centralização do conhecimento e a documentação inadequada, evidenciando a eficácia do algoritmo na criação de um ambiente mais colaborativo e eficiente. Por outro lado, problemas como a dificuldade de integração de novos membros ainda demandam atenção, embora tenham apresentado progresso significativo. Esse gráfico reforça a importância de ajustes contínuos no processo para abordar completamente todos os desafios enfrentados pela equipe.

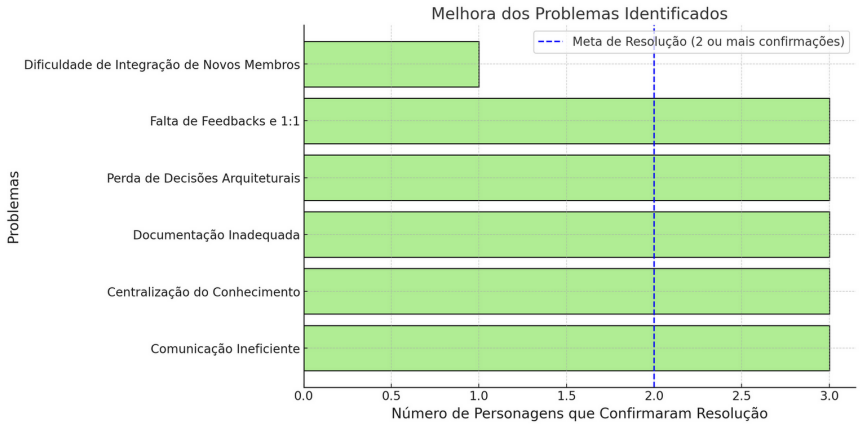

Figure 9: Confirmação da resolução dos problemas identificados após a aplicação do algoritmo *ReduceKnowledgeVaporization*.

5.3 Métricas de Engajamento no Processo de RDA

Com base nos dados fornecidos sobre o engajamento da equipe em torno do processo de produção de RDAs (Registros de Decisões Arquiteturais, do inglês *Architecture Decision Records* - ADR), é possível observar uma correlação direta entre a forma como os RDAs foram documentados e discutidos e o nível de engajamento dos integrantes do projeto. A análise detalhada dos diferentes períodos de trabalho permite identificar que ajustes no fluxo de trabalho, na organização das reuniões e na forma como a comunicação foi estruturada impactaram significativamente a participação dos membros da equipe.

O algoritmo "ReduceKnowledgeVaporization" pode ser diretamente aplicado a esse contexto, proporcionando uma estrutura mais robusta e centralizada para a criação, acompanhamento e revisão de RDAs. A implementação de ferramentas adequadas para a documentação, comunicação e rastreabilidade das decisões facilita o engajamento da equipe, além de promover um ambiente colaborativo onde a troca de ideias é constante [Borrego et al. 2019b]. A Figura 10 ilustra a variação do número de comentários ao longo do tempo, demonstrando os períodos de maior interação da equipe.

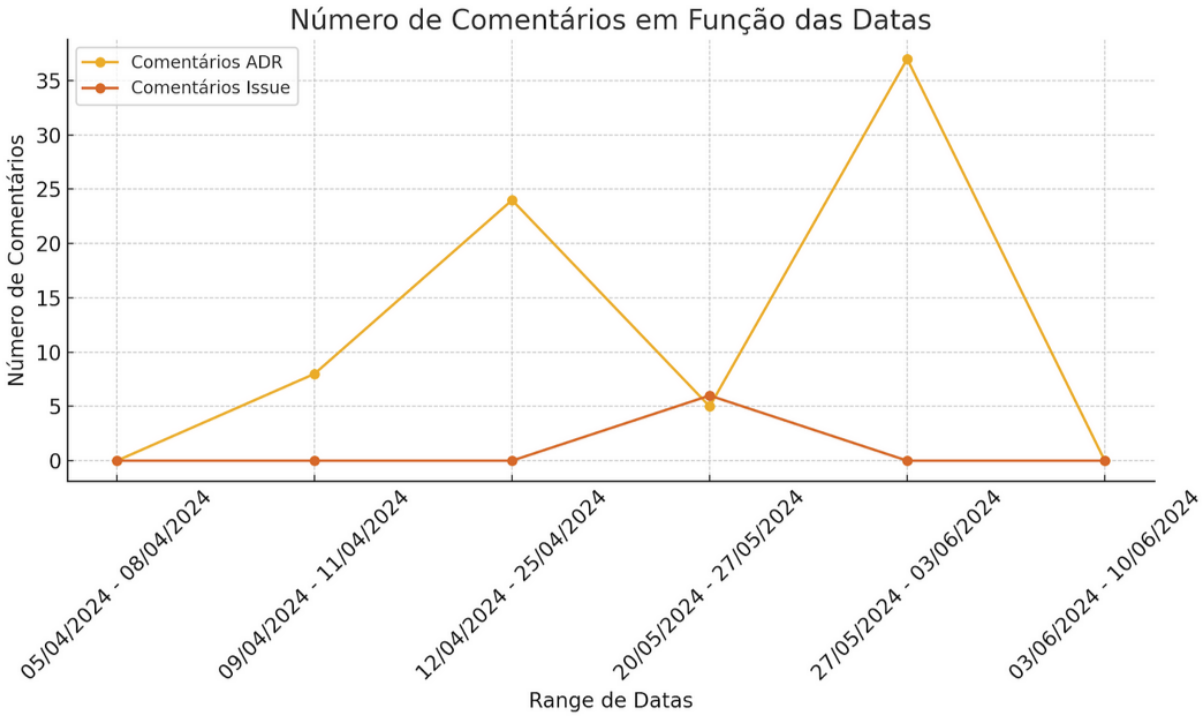

Figure 10: Número de Comentários em Função das Datas.

5.4 Tendência de Alta e Impacto na Qualidade do Desenvolvimento

A análise do gráfico na Figura 5 revela uma tendência de alta no número de comentários ao longo do tempo, o que sugere um aumento no engajamento da equipe em relação às discussões e revisões das decisões arquiteturais. Essa tendência reflete uma melhora significativa no processo de desenvolvimento de software, uma vez que um maior número de comentários está associado a uma maior participação e envolvimento dos membros nas discussões críticas sobre a arquitetura do sistema.

Esse crescimento no engajamento pode ser interpretado como um indicativo de que o algoritmo "ReduceKnowledgeVaporization" está contribuindo para uma melhor organização e documentação do conhecimento arquitetural. Com o uso desse algoritmo, a equipe consegue manter registros claros e acessíveis das decisões, o que facilita a comunicação, a colaboração e a rastreabilidade [Lewis et al. 2021]. Como resultado, o processo de desenvolvimento torna-se mais estruturado e eficiente, minimizando o risco de decisões arquiteturais fragmentadas e dispersas.

A tendência de alta no engajamento reflete diretamente no produto final, pois uma equipe mais engajada e informada é capaz de tomar decisões mais fundamentadas, considerando o histórico e os impactos de cada escolha arquitetural. Além disso, a documentação constante e colaborativa reduz a possibilidade de retrabalho e erros, o que melhora a qualidade do software entregue e diminui os custos associados à manutenção e evolução do sistema. Assim, observa-se que o aumento no engajamento, impulsionado pela aplicação do algoritmo "ReduceKnowledgeVaporization", contribui não apenas para a preservação do conhecimento arquitetural, mas também para a construção de um produto final mais robusto e sustentável [Borrego et al. 2019b].

Table 2: Métricas de Engajamento com Base nas Versões de Fluxo de RDA

Versão	Período	RDAs Criadas	Comentários RDA	Ideias Propostas	Comentários Issue	Nº de Integrantes	Nº Engajados
V1	05/04/2024 - 08/04/2024	1 (RDA-001)	0	N/A	N/A	1	0
V2	09/04/2024 - 11/04/2024	4 (RDA-002 - RDA-005)	8	N/A	N/A	2	2
V3	12/04/2024 - 25/04/2024	7 (RDA-006 - RDA-012)	24	N/A	N/A	5	2
V4	20/05/2024 - 27/05/2024	1 (RDA-013)	5	1	6	6	4
V5	27/05/2024 - 03/06/2024	4 (RDA-014 - RDA-017)	37	0	0	6	6
V6	03/06/2024 - 10/06/2024	5 (RDA-018 - RDA-022)	0	0	0	6	0

5.4.1 Observações e Análise de Comportamento

No início do projeto, com apenas um integrante, a equipe utilizava Google Docs e reuniões presenciais para documentar decisões básicas. Esse modelo inicial foi suficiente, mas limitou o engajamento e a colaboração, refletindo um baixo nível de atividade, com apenas um RDA registrado (Tabela 2).

Com a adição de novos membros, o número de RDAs aumentou para quatro, acompanhado por um crescimento significativo nos comentários e discussões, impulsionado pelas reuniões semanais e documentação no Google Docs. Apesar disso, a equipe sentiu a necessidade de uma estrutura mais robusta para gerenciar o conhecimento [Borrego et al. 2019a].

A implementação do GitHub para registro de RDAs e reuniões assíncronas, alinhada ao algoritmo *ReduceKnowledgeVaporization*, promoveu maior colaboração e organização centralizada do conhecimento. Esse novo modelo resultou em um aumento expressivo

nos comentários (Tabela 2), embora nem todos os integrantes tivessem engajamento igual [Farenhorst and de Boer 2010].

Com a equipe consolidada e composta por seis integrantes, o uso de issues e a documentação de comentários nos RDAs fortaleceram a comunicação e a rastreabilidade das decisões. Esse modelo elevou o engajamento ao seu ápice, com 37 comentários registrados, promovendo uma cultura de colaboração contínua [Dalkir 2011].

Entretanto, uma queda no engajamento foi observada posteriormente, atribuída à formalização excessiva da comunicação e sobrecarga de avisos, indicando a necessidade de ajustes no processo para manter a motivação da equipe (Tabela 2) [Lewis et al. 2021].

5.5 Conclusão do Estudo de Caso

A aplicação do algoritmo *ReduceKnowledgeVaporization* no projeto OBCOOP trouxe melhorias claras em comunicação, descentralização do conhecimento e documentação. A reestruturação das *dailys* e o uso do Notion como repositório central aumentaram o alinhamento e a produtividade da equipe. A introdução de RDAs (*Architecture Decision Records*) e a designação de revisores promoveram maior colaboração e transparência, reduzindo a vaporização do conhecimento arquitetural e fortalecendo a confiabilidade das decisões [Farenhorst and de Boer 2010].

Apesar dessas melhorias, desafios como a documentação de requisitos destacam a necessidade de aprimorar práticas para preservar o conhecimento crítico de negócios. As soluções implementadas, como centralização da documentação e clareza nas responsabilidades, demonstram o alinhamento do algoritmo com os problemas enfrentados, promovendo um fluxo de trabalho mais ágil e colaborativo [Borrego et al. 2019b]. Assim, o algoritmo contribuiu para uma arquitetura de software mais sólida e sustentável, atendendo às demandas do projeto OBCOOP [Lewis et al. 2021].

6 Conclusão

Este estudo demonstrou que o algoritmo *ReduceKnowledgeVaporization* teve impactos positivos no desenvolvimento de software, especialmente em ambientes ágeis e distribuídos. A análise das métricas de engajamento (Tabela 2) mostrou uma correlação entre sua aplicação e o aumento do engajamento da equipe na documentação e discussão de decisões arquiteturais. Essa abordagem facilitou a preservação do conhecimento arquitetural e promoveu um ambiente colaborativo, onde decisões críticas foram documentadas e revisadas continuamente [Lewis et al. 2021]. A Figura 11 apresenta os resultados obtidos com a aplicação das boas práticas propostas, destacando os benefícios de médio e longo prazo no desenvolvimento de software.

Os comentários dos participantes reforçam os resultados obtidos. Conforme destacado por P1, a centralização da documentação e a designação de revisores para as tarefas reduziram a perda de conhecimento e melhoraram a confiabilidade das entregas. P2 afirmou que a aplicação parcial do algoritmo foi suficiente para observar avanços significativos na retenção de informações e na comunicação da equipe, tanto internamente quanto com agentes externos. P3 complementou, relatando que a utilização de RDAs não só aumentou a agilidade da

equipe, mas também melhorou a rastreabilidade das decisões arquiteturais, permitindo maior colaboração e transparência.

Uma das principais vantagens do algoritmo é a facilidade de manutenção futura. Com a documentação e decisões arquiteturais registradas e organizadas, equipes futuras poderão acessar rapidamente o histórico, compreendendo o racional por trás de cada escolha e os trade-offs envolvidos. Essa rastreabilidade reduz a dependência de conhecimento tácito, frequentemente perdido com a rotatividade da equipe, e contribui para uma base sólida de conhecimento arquitetural [Borrego et al. 2019a].

Além disso, a centralização da documentação garante que novas integrações ou modificações possam ser realizadas com segurança e rapidez. O uso de RDAs (*Architecture Decision Records*) e práticas de comunicação documentada facilita a compreensão das interdependências do sistema, reduzindo riscos de inconsistências e acelerando as fases de planejamento e manutenção [Dalkir 2011, Farenhorst and de Boer 2010].

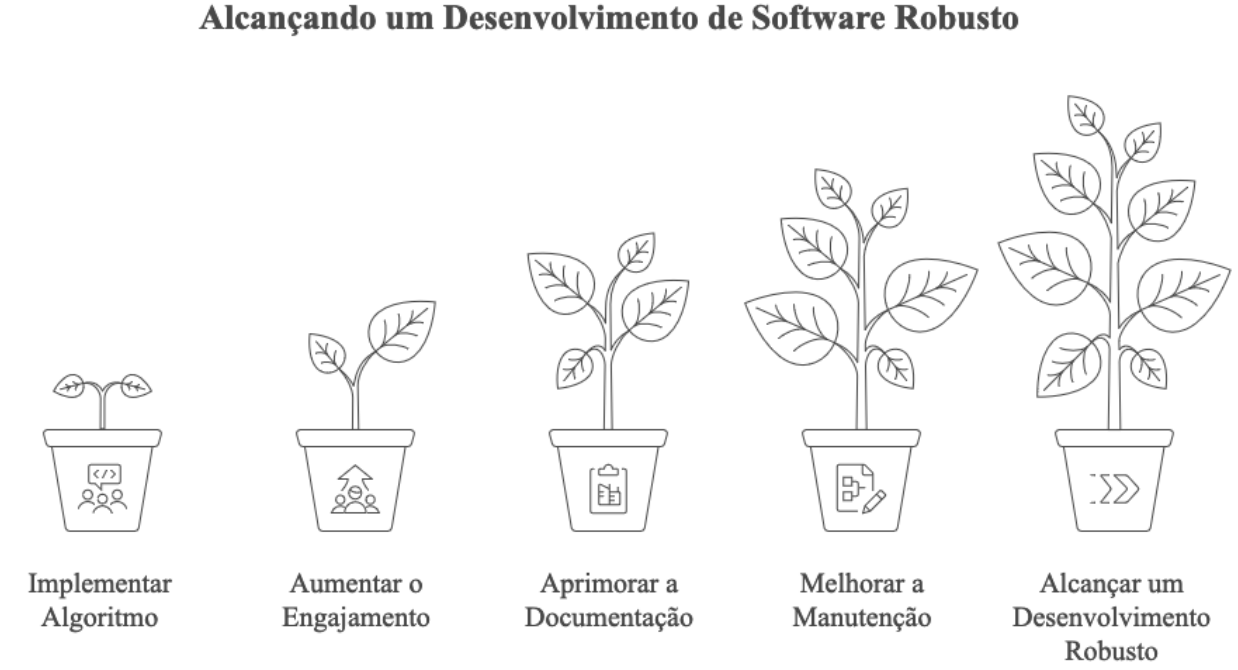

Figure 11: Resultados ao aplicar boas práticas no desenvolvimento de software no médio e longo prazo.

Ao longo do tempo, a adoção do algoritmo demonstrou benefícios significativos para a equipe, melhorando a rastreabilidade das decisões e aumentando a participação ativa dos integrantes, como evidenciado pelo crescimento nos comentários e discussões sobre ADRs (Figura 6). O engajamento crescente ressalta a importância de uma documentação clara e acessível, que incentiva contribuições ativas nas decisões de design. Além disso, práticas assíncronas e colaborativas, como reuniões semanais e avisos periódicos, promoveram uma comunicação mais eficiente e alinhada entre os membros do projeto.

Os resultados dessas boas práticas e da aplicação do algoritmo tornam-se evidentes no médio e longo prazo, refletindo-se na maturidade da equipe e na qualidade das entregas.

Essa evolução contribui para um desenvolvimento de software mais robusto e sustentável [Lewis et al. 2021].

6.1 Melhorias e Complementos ao Algoritmo ReduceKnowledgeVaporization

Apesar dos resultados positivos, o algoritmo *ReduceKnowledgeVaporization* pode ser aprimorado para aumentar o engajamento da equipe e a eficácia da documentação. Uma abordagem mais personalizada para a frequência de notificações e avisos, ajustável conforme as necessidades da equipe, ajudaria a evitar sobrecarga de informações que, em estágios avançados, impactaram negativamente a motivação da equipe.

A integração de mecanismos automatizados de captura e armazenamento de conhecimento arquitetural é outra melhoria relevante. Ferramentas que sincronizam documentação com o código-fonte e plataformas de comunicação, como o GitHub, podem reduzir a dependência de atualizações manuais e garantir informações sempre alinhadas ao progresso do projeto [Sohan et al. 2016].

Além disso, o uso de inteligência artificial (IA) para transcrever reuniões técnicas, capturar decisões e sugerir soluções pode otimizar o fluxo de trabalho, permitindo que a equipe foque em decisões estratégicas enquanto tarefas repetitivas são automatizadas. Isso também minimiza o risco de perda de informações importantes.

Por fim, um processo de revisão contínua, com sessões regulares de feedback sobre a eficácia da documentação, asseguraria que o algoritmo se adaptasse às mudanças e permanecesse alinhado às necessidades da equipe ao longo do projeto [Borrego et al. 2019a].

Além dessas sugestões, complementos específicos podem fortalecer ainda mais a preservação do conhecimento arquitetural:

- **Mapeamento de Conhecimento Crítico:** Identificar e priorizar informações essenciais, como dependências entre sistemas e componentes críticos, para reduzir a chance de perda de dados vitais [Borrego et al. 2019a].
- **Automação da Documentação:** Capturar metadados diretamente de commits e *pull requests*, integrando ferramentas de versionamento ao repositório de conhecimento para gerar documentação automaticamente [Sohan et al. 2016].
- **Documentação Incremental:** Revisar e atualizar a documentação em ciclos regulares de desenvolvimento, garantindo que ela permaneça em sincronia com o progresso do projeto e prevenindo a acumulação de dívida documental.
- **Base de Conhecimento Centralizada:** Consolidar documentações de arquitetura, decisões e práticas recomendadas em um único repositório para facilitar o acesso e acelerar o onboarding de novos membros.
- **Repositório de Lições Aprendidas:** Registrar obstáculos e soluções ao longo do projeto em um repositório dedicado, servindo como referência para projetos futuros e capacitação de novos integrantes [Dalkir 2011].

Essas melhorias reforçam os impactos positivos do algoritmo *ReduceKnowledgeVaporization*, garantindo que ele permaneça uma ferramenta eficaz para preservar o conhecimento arquitetural e apoiar o desenvolvimento sustentável de software em ambientes ágeis e distribuídos.

6.2 Extensão do Algoritmo *ReduceKnowledgeVaporization*

O algoritmo *ReduceKnowledgeVaporization* foi revisado para incluir melhorias específicas que visam aprimorar a preservação e organização do conhecimento arquitetural, especialmente em ambientes ágeis e distribuídos. As novas linhas incorporadas estão claramente identificadas e destacadas em verde para facilitar a leitura.

Algorithm 3 *ReduceKnowledgeVaporization* (Parte 1)

Data: Projeto de desenvolvimento de software em ambiente ágil e distribuído
Result: Redução da vaporização do conhecimento arquitetural e aumento da qualidade do produto final

Etapa 1: Inicialização

1. Configurar ferramentas de comunicação e colaboração (ex: Slack, Microsoft Teams, Trello, Jira, Confluence)
2. Estabelecer um repositório central de documentação (ex: Notion)
3. Definir templates e padrões de comunicação e documentação
4. Implementar treinamentos sobre boas práticas de documentação e comunicação
5. Identificar e mapear áreas críticas de conhecimento, priorizando dependências e componentes essenciais.

Etapa 2: Início do Projeto

1. Criar canais específicos nas ferramentas de comunicação para tópicos do projeto (ex: arquitetura, design, suporte)
2. Configurar espaços de projeto em ferramentas de gerenciamento de tarefas e documentação (ex: Trello, Jira)
3. Designar responsáveis pela documentação e rotacioná-los para evitar a centralização do conhecimento
4. Estabelecer uma base inicial de conhecimento com informações essenciais e lições aprendidas de projetos similares.

Etapa 3: Documentação de Requisitos

1. Registrar atas de reuniões de levantamento de requisitos no repositório central
2. Associar requisitos às decisões arquiteturais e refletir no backlog
3. Manter o sistema de gerenciamento de tarefas atualizado com mudanças nos requisitos
4. Vincular requisitos a *pull requests* e commits relevantes para rastreabilidade contínua.

Etapa 4: Comunicação Eficaz

1. Registrar discussões decisivas nas ferramentas de comunicação e transcrevê-las para o repositório de conhecimento
2. Atualizar o sistema de gerenciamento de tarefas com mudanças e decisões arquiteturais importantes
3. Documentar decisões usando documentos curtos e diretos
4. Adotar práticas assíncronas para minimizar gargalos em decisões críticas.

Etapa 5: Prática de Documentação Incremental

1. Estabelecer checkpoints regulares para revisar e atualizar a documentação ao longo do projeto
2. Associar documentos a *pull requests* ou *issues* para facilitar a rastreabilidade
3. Automatizar notificações para revisar documentos desatualizados ou inconsistentes.

Etapa 6: Produção de Artefatos

1. Criar documentos de design técnico no repositório de conhecimento, acessíveis a toda a equipe
2. Associar commits de código a tarefas e documentos no sistema de gerenciamento de tarefas para rastreabilidade
3. Incluir referências aos documentos de design nos comentários do código
4. Adotar ferramentas visuais para representar dependências arquiteturais e fluxos de trabalho.

Etapa 7: Revisão e Refinamento

1. Realizar revisões colaborativas de código e design, garantindo que as mudanças estejam na documentação
2. Manter a documentação sempre atualizada com o progresso do projeto
3. Adicionar melhorias e insights aos documentos após cada revisão
4. Implementar métricas para avaliar a qualidade e o impacto da documentação nas entregas.

Algorithm 4 *ReduceKnowledgeVaporization* (Parte 2)

Data: Projeto de desenvolvimento de software em ambiente ágil e distribuído
Result: Redução da vaporização do conhecimento arquitetural e aumento da qualidade do produto final

Etapa 8: Automatização da Documentação
1. Utilizar ferramentas de documentação automatizada integradas ao código-fonte para capturar decisões automaticamente
2. Implementar scripts que associem metadados a cada commit, vinculando-os aos RDAs
3. Adotar integração contínua para atualizar automaticamente documentos relacionados às mudanças.

Etapa 9: Armazenamento e Acesso
1. Armazenar toda a documentação em um repositório central de fácil acesso
2. Definir permissões adequadas para facilitar o acesso sem comprometer a segurança
3. Estruturar as informações com boas práticas de nomenclatura e tags
4. Configurar notificações automáticas para atualizações ou inconsistências de acesso.

Etapa 10: Feedback e Iteração Contínua
1. Realizar sessões regulares de feedback para identificar lacunas na documentação
2. Incorporar retrospectivas específicas para avaliar a eficácia das práticas de documentação
3. Adotar métricas de engajamento para monitorar a participação da equipe nas práticas documentais.

Etapa 11: Treinamento Contínuo
1. Realizar sessões de treinamento regulares sobre práticas de documentação e comunicação
2. Designar mentores para auxiliar novos membros na adaptação
3. Adicionar simulações de cenários para reforçar a aplicação prática das práticas documentais.

Etapa 12: Validação Contínua
1. Realizar auditorias periódicas para verificar a conformidade com os padrões estabelecidos
2. Coletar feedback contínuo da equipe sobre as práticas de documentação
3. Estabelecer métricas para monitorar a qualidade da documentação
4. Criar relatórios trimestrais com indicadores de impacto das práticas implementadas.

Etapa 13: Encerramento do Projeto
1. Realizar uma sessão de encerramento documentando todas as decisões arquiteturais e lições aprendidas
2. Compilar e arquivar um relatório final no repositório central para consultas futuras
3. Disponibilizar toda a documentação para uso em projetos futuros
4. Conduzir uma análise de retrospectiva final para avaliar a eficácia do algoritmo durante o ciclo do projeto
5. Identificar e registrar práticas bem-sucedidas que possam ser replicadas em novos projetos.

As melhorias implementadas no algoritmo foram percebidas de forma incremental durante sua aplicação, alinhando-se aos desafios de erosão arquitetural em ambientes ágeis e distribuídos [1]. Práticas como documentação incremental e automação de registros foram cruciais para mitigar esse problema, proporcionando maior rastreabilidade e clareza nas decisões [2]. A integração de ferramentas automatizadas e treinamentos contínuos reforçou os princípios de gerenciamento do conhecimento arquitetural [3], enquanto revisões colaborativas e checkpoints regulares melhoraram a qualidade do produto final, alinhando-se às boas práticas consolidadas na literatura [4].

Essas estratégias demonstram que o uso iterativo do algoritmo não só reduz a vaporização do conhecimento, mas também eleva a maturidade e a eficiência dos processos arquiteturais em projetos de software complexos, corroborando a necessidade de práticas robustas para preservar o conhecimento arquitetural [5, 6].

Dedicatória

"Ao meu pai, que me revelou a ciência como uma expressão de amor, e com ela, forjou em mim o espírito de um cientista."

Agradecimentos

Agradeço ao meu pai, Manoel Vieira Filho, por seu apoio e incentivo ao longo da minha trajetória acadêmica. À minha mãe, Maria Macarena Moreira Lajo Makaron, e ao meu padrasto, Elias Makaron Neto, pela dedicação e suporte em todas as etapas da minha formação.

Meu reconhecimento ao Professor Awdren de Lima Fontão pela orientação e contribuição fundamental para a realização deste estudo.

Agradeço também a todos que, de forma direta ou indireta, contribuíram para a construção deste trabalho, seja por meio de ideias, discussões ou pelo impacto que tiveram na formação de quem sou hoje. Obrigado a todos.

References

[Avgeriou et al. 2008] Avgeriou, P., Lago, P., and Kruchten, P. (2008). Architectural knowledge and rationale: Issues, challenges, and needs. *Journal of Systems and Software*, 81(9):1430–1442.

[Bass et al. 2003] Bass, L., Clements, P., and Kazman, R. (2003). *Software Architecture in Practice*. Addison-Wesley Professional.

[Borrego et al. 2019a] Borrego, G., Morán, A. L., Palacio, R. R., Vizcaíno, A., and García, F. O. (2019a). Towards a reduction in architectural knowledge vaporization during agile global software development. *Information and Software Technology*, 112:68–82.

[Borrego et al. 2019b] Borrego, G., Morán, A. L., Palacio, R. R., Vizcaíno, A., and García, F. O. (2019b). Towards a reduction in architectural knowledge vaporization during agile global software development. *Information and Software Technology*, 112:68–82.

[Bosch 2000] Bosch, J. (2000). *Design and Use of Software Architectures: Adopting and Evolving a Product-Line Approach*. Addison-Wesley.

[Dalkir 2011] Dalkir, K. (2011). *Knowledge Management in Theory and Practice*. MIT Press.

[Farenhorst and de Boer 2010] Farenhorst, R. and de Boer, R. (2010). Architectural knowledge management: Supporting architects and auditors. *IEEE Software*, 27(6):26–31.

[Jansen and Bosch 2020] Jansen, A. and Bosch, J. (2020). Managing architectural knowledge in agile projects: Current practices and future directions. *Journal of Systems and Software*, 164:110578.

[Kruchten 2004] Kruchten, P. (2004). *The Rational Unified Process: An Introduction*. Addison-Wesley.

[Lewis et al. 2021] Lewis, G. A., Ozkaya, I., and Xu, X. (2021). Software architecture challenges for ml systems. *IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 634–638.

[Li et al. 2022] Li, R., Liang, P., Soliman, M., and Avgeriou, P. (2022). Understanding software architecture erosion: A systematic mapping study. *Journal of Software: Evolution and Process*, 34(3):e2423.

[Martínez-Fernández et al. 2020] Martínez-Fernández, S. et al. (2020). Evolvability of machine learning-based systems: An architectural design decision framework. *Information and Software Technology*, 127:106363.

[Serban and Visser 2022] Serban, A. and Visser, J. (2022). Adapting software architectures to machine learning challenges. *Empirical Software Engineering*.

[Sohan et al. 2016] Sohan, S. K. et al. (2016). Auto-tagging emails in agile projects to link with user stories. *IEEE Transactions on Software Engineering*, 42(3):213–226.

[Taylor et al. 2014] Taylor, R. N., Medvidovic, N., and Dashofy, E. M. (2014). Software architecture: Foundations, theory, and practice. *Addison-Wesley*.

[VersionOne 2017] VersionOne (2017). 11th annual state of agile report. `https://explore.versionone.com/state-of-agile/versionone-11th-annual-state-of-agile-report-2`. Acessado em Outubro de 2024.