Uma Revisão Sistemática sobre Importância da Priorização de Solicitações Conforme Seus Impactos na Gestão de Manutenção de *Software* no Contexto Brasileiro

Jessica M. Giroldo, André C. da Silva

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas
Instituto Federal de Educação, Ciência e Tecnologia de São Paulo (IFSP)
Campus Hortolândia – SP – Brasil

jessica.moretti@aluno.ifsp.edu.br, andre.constantino@ifsp.edu.br

Abstract. *In an increasingly digital world, software maintenance plays a crucial role in maintaining, updating and improving systems. Despite its relevance, software maintenance is often underestimated, with little attention given to specific approaches to this phase. In this study, we explore software maintenance management practices, focusing on analyzing the impact of changes and their potential to increase efficiency. Doing a systematic review, we conducted an analysis of seven relevant articles in the area, synthesizing the results found. The insights obtained in this study highlight the importance of analyzing, planning and implementing specific strategies for software maintenance, aiming to optimize the long-term success of the product.*

Resumo. *Em um mundo cada vez mais digital, a manutenção de software desempenha um papel crucial na conservação, atualização e aprimoramento dos sistemas. Apesar de sua relevância, a manutenção de software muitas vezes é subestimada, com pouca atenção dada a abordagens específicas para essa fase. Neste estudo, exploramos práticas de gestão de manutenção de software, focando na análise de impacto das mudanças e seu potencial para aumentar a eficiência. Empregando a revisão sistemática, conduzimos uma análise de sete artigos relevantes na área, sintetizando os resultados encontrados. Os insights obtidos destacam a importância da análise, planejamento e implementação de estratégias específicas para a manutenção de software, visando otimizar o sucesso do produto a longo prazo.*

1. Introdução

O processo de gestão de manutenção de *software* influencia diretamente sua funcionalidade, adaptabilidade e alinhamento às demandas de negócios após a transição para a fase de manutenção. Práticas eficazes são vitais para a continuidade e valor do *software*, conforme afirma Torres (2015).

Segundo Vale (2019), o Departamento de Estatísticas do Trabalho dos Estados Unidos (*Bureau of Labor Statistics*) estimou em 2010 que cerca de 68% dos desenvolvedores de *software* no país estão envolvidos em atividades de manutenção de *software*. Essa realidade não deve ser muito diferente no Brasil, uma vez que, após a construção de produtos, surge a necessidade de mantê-los. Todos os dias, em todo o mundo, novos produtos alcançam a fase de manutenção.

Normalmente a fase de manutenção e evolução de *software* é a fase mais longa do seu ciclo de vida. Segundo Vale (2019), o momento que define a transição do desenvolvimento para a manutenção é o ponto onde o negócio do seu cliente passa a depender do seu produto. O *software* é lançado e colocado em uso, então, a manutenção envolve a correção de erros que não foram descobertos em estágios iniciais do ciclo de vida, com melhora da implementação das unidades do sistema e ampliação de seus serviços em resposta às descobertas de novos requisitos, conforme descrito por Sommerville (2011).

Dada a importância que esta fase possui no ciclo de vida de um *software*, torna-se evidente o emprego de abordagens específicas para ela, a qual frequentemente é subestimada. Para se compreender o espaço de trabalho, ao realizar uma busca simples usando a *string* de busca "gestão AND manutenção AND *software*" no *Google Scholar* são retornados aproximadamente 97.800 (busca realizada em 27 de outubro de 2023). Desses resultados, 7.290 foram publicados em 2023, e 16.700 foram publicados em 2022. No entanto, a maioria dos artigos concentra-se em novos projetos e oferece poucas perspectivas sobre a manutenção do *software* após ele ser colocado em uso.

O objetivo deste estudo é analisar práticas de gestão que são focadas na análise de impactos das modificações que o *software* vai sofrer durante seu uso e os efeitos na qualidade das entregas de novas versões, atualizações e correções, com base em publicações em português da comunidade científica brasileira.

Neste trabalho iremos abordar na Seção 2 os conceitos de qualidade de *software*, manutenção e evolução de *software* e análise de impacto. Usando o método de revisão sistemática, exposto na Seção 3, formulamos uma pergunta-chave que direcionou a seleção das palavras-chave para a busca de artigos relacionados ao tema. A partir dessa pesquisa, descrita na Seção 4, identificamos e analisamos 7 artigos que compõem a síntese deste estudo, apresentada na Seção 4.6. Finalizamos o texto com as conclusões que são apresentadas na Seção 5.

2. Referencial Teórico

Nesta seção abordaremos os conceitos básicos ao qual este trabalho articula, que são a qualidade de *software*, a manutenção e tipos de manutenção de *software*, e a análise de impacto.

2.1 Qualidade de *software*

Qualidade é uma medida da adequação de um produto ou serviço às necessidades do cliente, abrangendo todas as características que contribuem para sua capacidade de satisfazer as demandas e expectativas, sendo o principal fator que os consumidores buscam ao decidir pela aquisição de um produto ou serviço, como citado por Salgueiro et al. (2005).

Ainda segundo este artigo, a qualidade de *software* pode ser entendida como o atendimento das necessidades ou propósitos dos usuários. Ela abrange diversos atributos, como funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade, portabilidade e segurança. A qualidade de *software* é crucial para garantir que o produto atenda às necessidades do usuário, seja confiável e seguro e possa ser mantido e evoluído ao longo do tempo. Para garantir a qualidade do *software*, são aplicadas várias práticas e técnicas, incluindo testes de *software*, revisões de código, padrões de codificação, boas práticas de engenharia de *software* e processos de garantia de qualidade.

Manutenibilidade é um dos fatores da qualidade de *software* como citado por Sommerville (2011), pois se refere à facilidade com que um *software* pode ser adaptado, aprimorado e/ou corrigido para satisfazer os requisitos especificados, conforme a vontade do cliente.

A ISO (*International Organization for Standardization*) é uma organização internacional que promove a padronização de produtos e serviços por meio de normas internacionais. O IEC (*International Electrotechnical Commission*) é a Comissão Eletrotécnica Internacional, responsável pela normatização de tecnologias elétricas, eletrônicas e correlatas. Ambas as organizações colaboram para desenvolver normas conjuntas, conhecidas como ISO/IEC, conforme Lucas (2016). Neste contexto, a norma ISO/IEC 14764 oferece um *framework* que visa facilitar o planejamento, gerenciamento e execução das atividades de manutenção, abrangendo variáveis como o tamanho e complexidade do *software*, a criticidade das modificações e o domínio do negócio.

Segundo Sommerville (2011), mudanças de processos com o objetivo de trazer melhorias devem ser realizadas para trazer qualidade, Lucas (2016) menciona que dentre as melhorias nos processos de *software*, temos os modelos de maturidade de *software*, que visam desenvolver e avaliar a maturidade das empresas em relação aos processos ao longo do ciclo de vida do *software*. Modelos como o CMMI (*Capability Maturity Model Integration*) e o MR MPS (Modelo de Referência para Melhoria de Processos do *Software*) estabelecem padrões compostos por práticas globais para avaliar e aprimorar a eficiência das organizações em diferentes estágios do desenvolvimento de *software*.

2.2 Manutenção de *Software*

A manutenção de *software* é uma parte do ciclo de vida de desenvolvimento do *software*. É o processo geral de modificação de um sistema depois que ele foi colocado em uso, definido por Sommerville (2005).

Segundo a norma ISO/IEC 14764 (2006), a manutenção de *software* é o processo de modificar um produto de *software* após sua entrega ao cliente. Essa modificação ocorre devido à identificação de problemas ou necessidades após o *software* estar em uso. A manutenção deve englobar todas as atividades e tarefas necessárias para alterar um produto de *software* existente, garantindo a preservação de sua integridade.

Certamente, o *software* passará por modificações após sua entrega ao usuário final , conforme cita Paduelli (2007). Essas alterações podem ocorrer devido à descoberta de erros, necessidade de adaptação a mudanças estruturais feitas pelo cliente ou solicitação de funcionalidades adicionais ou melhorias de desempenho.

Em uma versão mais atualizada do livro de Sommerville (2011), ele afirma que determinar o momento em que um *software* está 'pronto para uso' ou qual o seu estágio inicial de desenvolvimento está se tornando cada vez mais irrelevante, uma vez que, ao longo dos anos, poucas aplicações são desenvolvidas completamente do zero. Nesse contexto, considerar a engenharia de *software* como um processo evolutivo torna-se mais coerente. O *software* é continuamente modificado ao longo de seu ciclo de vida para se adaptar aos requisitos e às necessidades em constante crescimento dos usuários.

2.2.1 Tipos de manutenção de *software*

Existem alguns tipos de manutenção de *software*, cada uma com seu objetivo específico. Vargas e Pereira (2019) listaram os principais e mais conhecidos tipos de manutenção de *software*: manutenção adaptativa, manutenção corretiva, manutenção evolutiva e manutenção preventiva, conforme Figura 1.

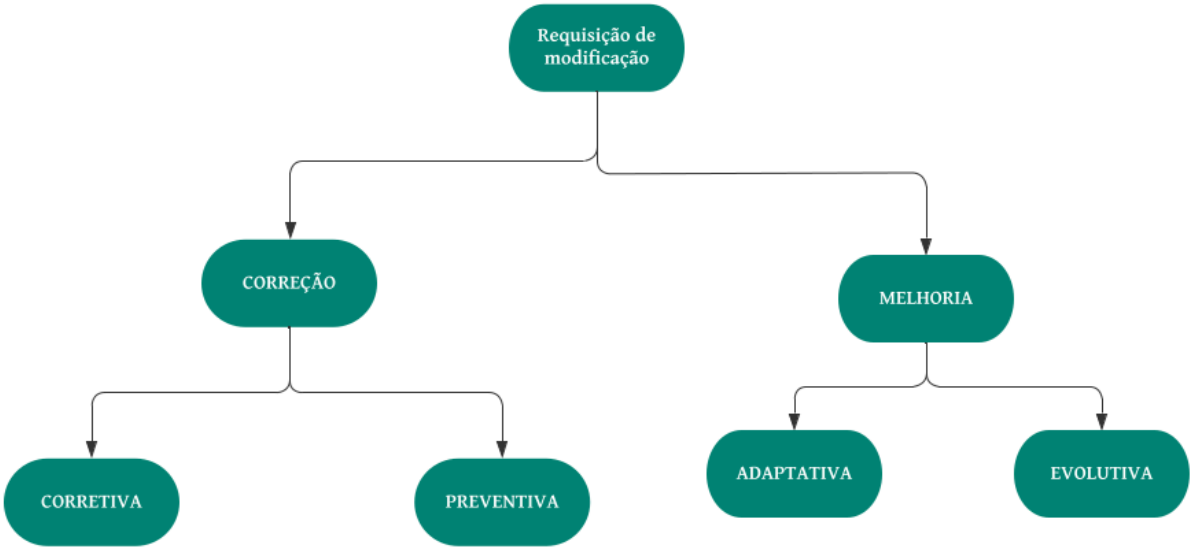

Figura 1 - Tipos de manutenção de *software*

A manutenção adaptativa consiste em ajustes no *software* para se adequar a uma nova realidade ou ambiente externo, como mudanças de plataforma, sistema operacional, ambiente operacional, regras de negócio ou legislação governamental.

A manutenção corretiva busca corrigir *bugs*, falhas e defeitos no *software*, em muitos casos identificados pelos usuários durante o uso. Os desenvolvedores são acionados para resolver esses problemas rapidamente, evitando prejuízos à empresa. Às vezes, a solução pode ser temporária até que uma correção definitiva seja implementada posteriormente.

A manutenção evolutiva se trata de modificações que introduzem novas funcionalidades e aprimoramentos solicitados pelos usuários ou novidades no mercado. Essas solicitações resultam em melhorias no sistema, que visam acompanhar as demandas do mercado e dos usuários, mantendo o *software* atualizado e competitivo.

Por fim, a manutenção preventiva envolve monitorar continuamente o *software* já em produção para identificar e corrigir falhas antes que elas se tornem problemas operacionais. Esse tratamento visa aprimorar a confiabilidade do sistema e sua manutenibilidade.

A norma ISO/IEC 14764 (2006) define manutenibilidade como a capacidade do produto de *software* ser modificado. Essas modificações podem englobar correções, melhorias ou adaptações do *software* para lidar com alterações no ambiente, requisitos e especificações funcionais.

Sommerville (2011) menciona que a distribuição dos custos de manutenção pode variar entre empresas. Globalmente, a manutenção corretiva não é a mais custosa (consumindo 17% dos esforços), sendo que a manutenção de aperfeiçoamento consome mais recursos destinados à manutenção (consumindo 65% dos esforços). A Figura 2 ilustra essa distribuição aproximada de custos.

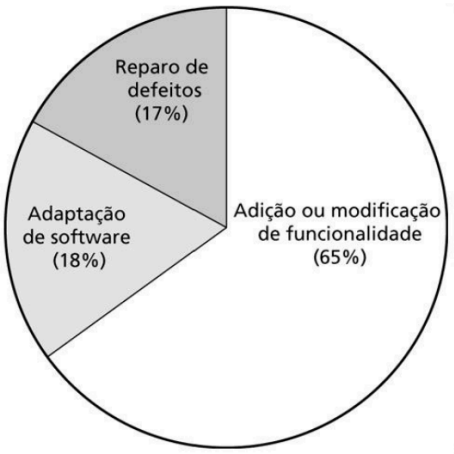

Figura 2. Distribuição dos esforços de manutenção. Fonte: Sommerville (2011, pág. 171)

Na fase de manutenção de um *software*, os responsáveis podem enfrentar desafios para compreender as funcionalidades existentes, resultando em atrasos nas modificações e transtornos para os envolvidos. A manutenibilidade do sistema é crucial não apenas para o desempenho do desenvolvedor atual, mas também para facilitar futuros trabalhos na continuidade do projeto.

Com base em estudos referenciados por Thomazinho (2018), diversos desafios surgem na manutenção e evolução de *software*. Entre eles, destacam-se os elevados custos associados a esta fase, a necessidade de estimar o esforço de manutenção, o qual geralmente consome a maior parte do trabalho dos desenvolvedores, atender a pedidos de usuários e a falta de conhecimento e processos nas atividades de manutenção pela equipe responsável. Nesse contexto, sustenta-se a importância da implementação de processos adequados relacionados às tarefas de manutenção.

O mesmo artigo de Thomazinho (2018) sugere algumas melhores práticas para amenizar esses problemas. Estas incluem a avaliação, seleção e implementação de processos que facilitem o planejamento das atividades de manutenção, a adoção de metodologias ágeis e uma análise cuidadosa do tempo necessário para implementar melhorias na manutenção, determinando o momento ideal para sua execução. Além disso, algumas metodologias, como por exemplo o Kanban, são úteis para apoiar a manutenção de *software*, fornecendo visibilidade das tarefas, auxiliando na priorização e facilitando a sincronização com outras equipes.

2.3 Análise de impacto

A avaliação de impacto na manutenção de *software* é um processo que visa determinar como uma modificação ou atualização em um *software* existente afetará o sistema como um todo. Isso envolve analisar como a alteração planejada pode influenciar outras partes do *software*, identificar possíveis problemas ou efeitos colaterais e avaliar os riscos associados à implementação da mudança.

De acordo com Hattori (2008), as duas áreas fundamentais na análise de impacto são a análise de dependência e a análise de rastreabilidade. Essas áreas, complementares entre si, tratam a análise de impacto de perspectivas diferentes e oferecem vantagens específicas para aumentar o potencial de identificação dos impactos no *software*.

A análise de dependência examina as relações entre os componentes do sistema de *software*, explorando como uma alteração em um componente pode afetar outros componentes que dependem dele. Por exemplo, ao modificar um módulo do sistema, a análise de dependência auxilia na identificação dos módulos que podem ser impactados por essa mudança.

Já a análise de rastreabilidade estuda a capacidade de rastrear e compreender a origem e o destino das mudanças em um sistema de *software*. Isso envolve a documentação e o gerenciamento das relações entre requisitos, código-fonte, testes e outras partes do sistema. A análise de rastreabilidade é fundamental para garantir que todas as mudanças sejam adequadamente documentadas e compreendidas, facilitando assim a manutenção e a evolução contínua do *software*.

3. Metodologia

Para alcançar o objetivo proposto, analisar práticas de gestão que são focadas na análise de impactos das modificações que o *software* vai sofrer durante o seu uso bem como os efeitos na qualidade, este trabalho utiliza uma abordagem de revisão sistemática. Segundo Kitchenahm (2007), a revisão sistemática é um meio de identificar, avaliar e interpretar todas as pesquisas disponíveis relevantes para uma determinada pergunta de pesquisa ou um fenômeno de interesse.

Segundo este artigo os passos que se enquadram neste método são:

1. Formulação da pergunta da pesquisa: Nesta etapa, é questionado sobre um tema de interesse quando existe a necessidade de conhecer melhor sobre um assunto e são definidos os conteúdos que são a base para serem incluídos na pesquisa;
2. Seleção de fontes de dados: Buscar fontes de dados confiáveis, tais como periódicos científicos e diretórios acadêmicos, utilizando ferramentas de busca apropriadas para localizar e identificar estudos pertinentes;
3. Seleção das palavras-chaves e critérios de restrições: Identificar as palavras-chave relacionadas ao tema da pesquisa e definir os critérios de restrição, que são parâmetros estabelecidos para filtrar os estudos encontrados durante a busca, garantindo que apenas aqueles que atendam a determinados critérios sejam incluídos na revisão sistemática;
4. Realização das buscas nas fontes de dados selecionadas: Com as palavras chaves e critérios definidos, fazer a busca na fonte de dados selecionada para obter os resultados;
5. Avaliação dos títulos e resumos dos estudos encontrados durante a busca para determinar quais atendem aos critérios de inclusão: Fazer a leitura dos resumos e verificar se atende aos critérios definidos;
6. Análise dos estudos selecionados: Análise detalhada dos artigos encontrados para identificar padrões, tendências ou variações na literatura;
7. Sintetização dos resultados e conclusões dos estudos selecionados: Esta fase envolve resumir e integrar as principais descobertas, destacando padrões, tendências ou variações na literatura;
8. Apresentação dos resultados e conclusões: Discorrer sobre as descobertas e *insights* obtidos durante a revisão sistemática.

4. Desenvolvimento

Esta Seção tem como objetivo descrever as etapas do desenvolvimento deste trabalho, ou seja, a execução dos passos para a realização de uma revisão sistemática.

4.1 Formulação da pergunta de pesquisa

Em ambientes de trabalho, muitas vezes vemos o time de produtos de *software* ser preparado e qualificado para atuar em novos projetos de *software*, porém com pouco conhecimento em como atuar na manutenção e evolução de *software*. Entender a importância desta fase é crucial para manter o *software* com qualidade. Além disso, a falta de conhecimento em como analisar e lidar com os impactos das alterações, pode causar insatisfação no usuário final. A partir dessa premissa, definiu-se a seguinte pergunta de pesquisa:

Como a gestão da manutenção de *software*, com foco na priorização de demandas baseada no impacto, influencia a qualidade das entregas após a transição para a fase de manutenção, com ênfase em publicações em português da comunidade científica brasileira?

4.2 Seleção de fonte de dados

Devido a limitações de conhecimento de idioma e do escopo de se compreender o cenário nacional, optou-se por analisar somente artigos na língua portuguesa.

Ao explorar fontes de dados como: Portal de Periódicos CAPES, BDTD (Biblioteca Digital Brasileira de Teses e Dissertações), Lume (Repositório Digital da UFRGS) e SciELO Brasil (*Scientific Electronic Library Online*), por exemplo, não encontramos resultados correspondentes aos termos relacionados aos conceitos apresentados na fundamentação teórica.

Para a pesquisa, foi utilizado o Google Acadêmico como plataforma principal, utilizando apenas palavras em português. Ele é uma ferramenta de busca para trabalhos acadêmicos, como artigos de revistas científicas, teses, dissertações, resumos de conferências e livros com uma ampla e relevante variedade da literatura acadêmica, o que nos motivou a adotar essa fonte de dados nesta pesquisa.

4.3 Seleção das palavras-chaves e critérios de restrições

Inicialmente, a *string* definida foi: (("manutenção de *software*") AND (priorização) AND (risco OR riscos) AND (qualidade)). A ideia era utilizar o termo 'risco' em vez de 'impacto' na busca desta pesquisa. Muitos dos artigos encontrados neste momento se concentravam principalmente no desenvolvimento de um novo sistema que ainda não entrou em produção sem abordar a manutenção. Com os estudos dos artigos, foi notado que ao longo dos anos o termo 'análise de impacto' foi mais frequentemente utilizado para avaliar o custo e o risco associados a determinadas tarefas. Após essa modificação na *string* de busca, foram encontrados mais artigos relacionados à manutenção de *software* do que anteriormente, além de serem artigos mais atuais.

Além disso, na *string* inicial não incluímos a palavra 'evolução'. Após analisar os estudos, percebemos a necessidade de acrescentá-la na *string*, para trazer mais artigos relacionados ao momento em que o *software* passa a ser utilizado e são necessárias correções, adaptações e evoluções. A evolução de *software* visa expandir e melhorar suas funcionalidades para atender às demandas em constante mudança dos usuários e do mercado, o que também é relevante para este estudo. Com essa modificação, encontramos mais artigos

relacionados a entender como a análise de impactos pode influenciar a qualidade do *software* em uso, o que também ressalta a importância do conhecimento do indivíduo para a elaboração de uma *string* de busca mais adequada que consiga responder a questão de pesquisa.

Finalizadas as modificações, a *string* com as palavras chaves definidas nesta pesquisa foi: ("gestão" AND ("manutenção de *software*" OR "evolução de *software*") AND "priorização" AND ("impacto" OR "impactos") AND "qualidade").

O operador booleano 'AND', que significa 'E' em inglês, permite à base de dados buscar artigos científicos que contenham ambos os termos A e B em seus campos de busca, independentemente da ordem ou da distância entre os termos. Os resultados excluem artigos que contenham apenas A ou apenas B.

O operador booleano 'OR', que significa 'OU' em inglês, permite à base de dados buscar artigos científicos que contenham o termo ‘A’ ou o termo ‘B’ em seus campos de busca. Isso inclui artigos onde os termos aparecem separadamente ou juntos no texto. Portanto, na lista de resultados, haverá artigos que contenham somente ‘A’, somente ‘B’ e ambos os termos, independentemente da ordem ou distância entre eles no texto.

Neste contexto, a *string* foi criada esperando obter artigos relacionados a manutenção de *software* onde fossem citados os itens priorização, impacto e qualidade, onde em alguns artigos podem ser empregados a palavra impacto ou impactos.

Os critérios de restrições para a pesquisa foram: ordem de relevância, qualquer ano de publicação e qualquer idioma (pois foram utilizadas apenas palavras em português para a pesquisa).

4.4 Realização das buscas nas fontes de dados selecionadas

Com base na *string* de busca definida, foram encontrados um total de 621 artigos, os resumos dos 50 primeiros artigos foram lidos, sendo considerados suficientes para abordar a pergunta de pesquisa e propósito deste estudo. Após a leitura destes resumos, foram selecionados 15 artigos para leitura completa, ao final, 7 foram considerados adequados para o estudo, conforme será descrito nas subseções seguintes. O exposto, mostra apenas o resultado final de um processo de redefinição dos termos e realização da busca para aferir os retornos do repositório.

4.5 Avaliação dos títulos e resumos dos estudos encontrados durante a busca para determinar quais atendem aos critérios de inclusão

Após as modificações, foi realizada uma avaliação dos primeiros 50 artigos identificados usando a *string* de pesquisa, organizados por relevância, de acordo com os critérios do Google Acadêmico. Após leitura dos resumos destas 50 publicações, foram identificadas 15 artigos que continham informações relevantes para este estudo.

Alguns dos artigos lidos foram descartados devido à falta de menção ou à abordagem superficial do impacto das tarefas de manutenção e evolução de *software*. Embora contenham dados relevantes, como por exemplo o uso de *Kanban* para facilitar a manutenção ou estratégias de manutenção de *software* em *startups* em estágios iniciais (*early stage*), esses artigos foram excluídos por não estarem alinhados com o objetivo principal desta revisão,

portanto, ao final, resultaram 7 artigos que compõem a síntese apresentada na subseção a seguir.

4.6 Análise dos estudos selecionados

A Tabela 1 apresenta a listagem dos 7 artigos que foram selecionados para este estudo, apresentando o título do artigo, os autores, ano de publicação e um breve resumo. A primeira coluna enumera os artigos para facilitar a citação na seção 4.7 e os artigos estão organizados pela ordem de retorno da busca no *Google Scholar*.

A coluna 'Tipo de manutenção' refere-se aos tipos de manutenção abordados nos artigos encontrados, enquanto a coluna 'Resumo' contém um breve resumo do conteúdo de cada artigo.

Tabela 1. Listagem dos artigos selecionados na Revisão Sistemática

Número	Título do artigo	Autor	Ano publicação	Tipo de manutenção	Resumo
1	Proposta e Avaliação de um Processo para Agrupamento de Solicitações de Manutenções	APARECIDO, Gladston J; et al.	2010	Corretivas, adaptativas e evolutivas	O estudo apresenta o Processo para Agrupamento de Solicitações de Manutenção (PASM) como uma abordagem para guiar a priorização, agrupamento e execução de tarefas de manutenção de *software*.
2	Gestão da Manutenção de *Software*: Um Estudo de Caso	LUCAS, Ana Paula Lima.	2016	Corretivas, adaptativas e evolutivas	O estudo relata sobre uma organização que enfrentava problemas frequentes, como alta ocorrência de defeitos e retrabalhos, em seu processo de manutenção de *software*. Após uma análise preliminar do sistema, ficou claro que mudanças eram necessárias. Para abordar esses problemas, a organização decidiu adotar práticas do modelo de maturidade de manutenção de *software* (SMmm) e integrá-las a um processo adaptado às suas necessidades. Utilizando a plataforma *Team Foundation Service* (TFS), as práticas selecionadas foram implementadas, resultando em um processo definido que parcialmente implementa o modelo SMmm. Um estudo de caso foi conduzido para avaliar os benefícios dessa abordagem, revelando uma redução significativa na quantidade de defeitos após a adoção das práticas propostas.
3	Uma Abordagem de Priorização para Apoiar o Pagamento de Dívida Técnica Auto-admitida em Código-fonte	LIMA, Bruno Santos de	2021	Corretivas e evolutivas	O artigo propõe uma abordagem para lidar com a dívida técnica no código-fonte, destacando a importância de priorizar os itens de dívida técnica com base em critérios específicos, como impacto, complexidade e importância para os *stakeholders*.

Tabela 1. Listagem dos artigos selecionados na Revisão Sistemática (continuação)

Número	Título do artigo	Autor	Ano publicação	Tipo de manutenção	Resumo
4	Taxonomia de Riscos para Manutenção de *Software*	WEBSTER, Kênia Pereira Batista; et al.	2005	Corretivas, adaptativas e evolutivas	Os autores afirmam que, embora o gerenciamento de riscos seja uma prática fundamental na engenharia de *software*, sua aplicação à manutenção de *software* é pouco explorada. Este estudo busca abordar essa lacuna ao propor uma taxonomia de riscos específica para a manutenção de *software*, facilitando a identificação e gerenciamento de riscos.
5	Proposta de uma Disciplina sobre Manutenção de *Software* nas Graduações em Computação	ANQUETIL, Nicolas; et al.	2005	Corretivas, adaptativas, preventivas e evolutivas	Este artigo propõe a inclusão de uma disciplina de manutenção de *software* nos cursos de graduação em computação. A proposta é baseada na importância reconhecida da manutenção de *software* na prática, resultados positivos em cursos de pós-graduação e iniciativas internacionais.
6	Práticas Ágeis Aplicadas a um Processo de Manutenção de *Software*: Um Relato de Experiência	RAMOS, André L. B. M; et al.	2013	Corretivas, adaptativas e evolutivas	O objetivo deste trabalho é apresentar o processo de *software*, baseado no modelo ágil SCRUM, adotado na Divisão de Manutenção da Dataprev/PB. Ao final, é apresentada a sondagem de satisfação subjetiva da equipe de manutenção em relação à adoção do referido processo.
7	Estudo sobre o Gerenciamento de Riscos Técnicos em Manutenções Corretivas	SILVA J. G.T.; et al.	2013	Corretivas, adaptativas, preventivas e evolutivas	Este artigo discute como o gerenciamento de riscos técnicos pode aumentar a taxa de sucesso na entrega de projetos de manutenção de *software*. Ele apresenta os resultados de técnicas empregadas por uma empresa brasileira de desenvolvimento de *software*, que mostram uma redução na porcentagem de projetos entregues fora do prazo planejado, de 33,4% em 2008 para 25% em 2012.

4.7 Sintetização dos resultados e conclusões dos estudos selecionados

Com base nos artigos estudados que são focados em manutenção/evolução de *software* e análise do impacto das alterações necessárias nesta fase, destaca-se a importância fundamental de compreender os requisitos para manter a confiabilidade do *software*, onde a gestão eficaz de mudanças em ambientes de desenvolvimento e manutenção vai além da simples implementação de alterações até uma análise aprofundada dos potenciais riscos envolvidos. Existem estudos que mostram que até 90% do total de recursos consumidos ao longo da vida útil de um sistema são destinados à fase de manutenção (Serafim et al., 2022).

O artigo 1, de título "Proposta e Avaliação de um Processo para Agrupamento de Solicitações de Manutenções", de Aparecido et al. (2010), introduz a proposta do PASM (Processo de Agrupamento de Solicitações de Manutenção), descrevendo sua implementação na Divisão de Tecnologia da PUC Minas (DATAPUC) e os resultados positivos alcançados após essa implementação. Os princípios fundamentais da proposta envolvem a organização das tarefas para otimizar a eficiência das atividades e o registro adequado das solicitações para facilitar a organização das tarefas. O processo estabelece um período de um mês, durante

o qual várias etapas são realizadas semanalmente para atender às solicitações. Essas etapas incluem reuniões de planejamento, registro de solicitações, elaboração de propostas de projetos, avaliação e priorização de projetos, especificação de requisitos, validação das especificações de requisitos, implementação, validação da implementação, testes e implantação. A implementação do PASM demonstrou impactos positivos na eficiência e qualidade do atendimento das solicitações de manutenção na DATAPUC. Consequentemente, houve uma redução no tempo médio de atendimento, aumento da produtividade dos programadores e uma abordagem mais eficaz para a gestão de projetos de manutenção.

O artigo 6, "Práticas Ágeis Aplicadas a um Processo de Manutenção de *Software*: Um Relato de Experiência", por Ramos et al. (2013), embora não adote um modelo específico de priorização de solicitações, utiliza uma abordagem semelhante ao modelo mencionado anteriormente no artigo 1, sugerindo uma gestão ágil nas tarefas de manutenção. Inicialmente, é elaborada uma lista de demandas vindas da área de produtos, após uma interlocução com o cliente. Posteriormente, em uma reunião de planejamento envolvendo a equipe de produtos e o líder, são identificados e priorizados os casos a serem abordados nesta fase. Em princípio, mantém-se o planejamento estabelecido; contudo, caso surja uma demanda urgente, essa é priorizada, corrigida e implementada imediatamente no *software* em produção. Assim como no artigo anteriormente mencionado, destaca-se a melhoria no atendimento às solicitações resultantes do processo implementado para o tratamento da manutenção.

Dentre os vários tipos de manutenção de *software*, neste contexto, temos a dívida técnica que refere-se a compromissos assumidos durante o processo de desenvolvimento que podem resultar em consequências negativas no futuro. Esses compromissos geralmente envolvem escolhas que permitem a entrega mais rápida de uma funcionalidade ou solução no curto prazo, mas que resultam em custos adicionais a longo prazo devido à necessidade de retrabalho, manutenção mais difícil, maior propensão a *bugs* ou degradação do desempenho do sistema. A artigo 3, “Uma abordagem de priorização para apoiar o pagamento de dívida técnica auto-admitida em código-fonte”, de Lima (2021), traz uma forma de priorização destes item na manutenção do *software*, que indica detectar, organizar, identificar sua importância e complexidade, analisá-las para assim priorizá-las de forma adequada.

Em 2005 foi elaborado um estudo exposto no artigo 4, “Taxonomia de Riscos para Manutenção de *Software*”, de Webster et al. (2005), criando uma taxonomia de riscos para manutenção de *software*, o que não existia na literatura e também não encontrado até a presente data uma nova proposta do tipo. Como o nome sugere, a taxonomia é utilizada para identificar e classificar algo. Para a criação da taxonomia, o estudo examinou a literatura sobre manutenção de *software* e encontrou mais de 400 tipos de riscos. Para simplificar, foram aplicados alguns critérios para reduzir essa lista, como considerar apenas os riscos mencionados por dois ou mais autores e eliminar aqueles com significados semelhantes. No final, foram identificados 91 riscos relevantes. Considerados pelos autores como muita informação, esses riscos foram divididos em diferentes categorias para facilitar o seu gerenciamento. Essa abordagem permite uma gestão mais eficaz dos riscos ao longo do ciclo de vida do *software*, contribuindo com a análise de impacto, consequentemente melhorando a qualidade dos sistemas mantidos.

Na literatura brasileira encontramos alguns (poucos) artigos que analisam a eficiência da manutenção de *software*. Dentre eles, cita-se o artigo 7, “Estudo sobre o gerenciamento de riscos técnicos em manutenções corretivas”, de Silva *et al.* (2013), mencionando que os riscos relacionados ao processo de desenvolvimento de *software* podem ser: riscos técnicos,

riscos de cronograma, riscos de custos e riscos de financiamentos, que é alguns dos riscos citados no artigo 4. Com o foco no gerenciamento de riscos técnicos, ele é um estudo de caso de uma empresa de desenvolvimento de *software* focada em *Contact Center*, analisando o índice de sucesso de projetos de manutenção de *software*, antes e após adotarem novas técnicas de gerenciamento de riscos técnicos. Segundo os autores, a empresa crescia, porém tinha carência em processos estruturados, estimativas, entre outros, além de não utilizarem nenhuma metodologia de apoio na manutenção de *software*. Algumas áreas específicas ganharam novos gestores, incluindo claro, a área de manutenção de *software*. O processo de análise foi de 2008 até 2012, e com as novas técnicas abordadas, que foram um sucesso, reduziram a porcentagem de projetos de manutenção entregues fora do prazo de 33,4% em 2008, para 25,0% em 2009.

Dado as introduções da maioria dos artigos encontrados nesta pesquisa, a manutenção de *software* é tão pouco abordada, que nesta pesquisa encontramos o artigo 5, 'Proposta de uma Disciplina sobre Manutenção de *Software* nas Graduações em Computação', por Anquetil et al. (2005), elaborada em 2005 e aplicada neste mesmo ano no Centro Universitário de Goiás (UNIGOIÁS Anhanguera). A proposta é baseada na importância reconhecida da manutenção de *software* na prática, resultados positivos em cursos de pós-graduação e em algumas iniciativas internacionais. Os autores apresentam uma proposta de estrutura curricular para essa disciplina, destacando os principais tópicos a serem abordados, como por exemplo gestão da manutenção de *software*, qualidade na manutenção de *software*, documentação de *software*, ferramentas de apoio à manutenção e refatoração, itens que são citados nos outros artigos citados neste trabalho.

Dentre algumas iniciativas de melhoria de processos de *software*, destaca-se o *Capability Maturity Model Integration* (CMMI) e a Melhoria de Processo do *Software* Brasileiro (MR MPS). No entanto, essas abordagens são mais voltadas para o desenvolvimento e evolução de processos de *software*. Considerando a necessidade de abordar mais os *bugs* e a difilcudade de manutenibilidade que um *software* pode apresentar, o artigo 2, "Gestão da manutenção de *software*: Um estudo de caso", de Lucas (2016), se embasa nesse contexto para propor um processo específico voltado para a manutenção de *software*. Este processo adapta e implementa algumas práticas do *Maturity Model for* Software *Maintenance* (SMmm). Em uma abordagem quantitativa, o artigo apresenta um estudo de caso que investiga os desafios enfrentados na manutenção de *software* de uma empresa específica. O estudo concentra-se na implementação de práticas derivadas do *Maturity Model for Software Maintenance* (SMmm) utilizando o *Team Foundation Service - TFS*. O objetivo é avaliar se essa implementação contribui para a redução de defeitos introduzidos durante o processo de manutenção de *software*. Neste estudo, uma das primeiras práticas citadas para analisar os problemas/tarefas da manutenção, é a análise de impacto. Após adoção das práticas, os resultados analisados apontaram uma redução significativa da quantidade de defeitos.

Além disso, o artigo ressalta que, uma vez que o *software* é colocado em uso, as tarefas de manutenção, como a correção de *bugs* e implementação de melhorias, devem passar por um processo de avaliação inicial antes de serem aceitas ou rejeitadas. As tarefas aceitas devem ser documentadas e atribuídas a uma priorização adequada. Neste momento também é realizada uma análise de impacto nas tarefas para definir melhor como serão priorizadas.

Apesar de tratarem de aspectos diversos da manutenção de *software*, os estudos visam aprimorar os processos envolvidos nessa atividade, compartilhando o objetivo comum de melhorar a eficiência e a qualidade da manutenção de *software*.

O último passo da revisão sistemática é a apresentação dos resultados e conclusões que é apresentada a seguir na seção Conclusões.

5. Conclusões

Este estudo teve como objetivo analisar propostas ou relatos de práticas de gestão voltadas para a avaliação dos impactos das modificações que um *software* sofre durante seu uso, e como essa prática afeta a qualidade das entregas de novas versões, atualizações e correções. Para atingir esse objetivo, foi adotada a abordagem de revisão sistemática. Inicialmente, foi formulada uma pergunta de pesquisa, selecionada uma fonte de dados e definidas as palavras-chave e os critérios de restrição. Em seguida, foram realizadas buscas na fonte de dados selecionada, e os resumos dos 50 primeiros artigos encontrados foram avaliados. A partir dessa avaliação, foram selecionados 7 artigos para compor a síntese deste estudo.

Com base nos artigos, fica claro que manter requisitos claros e rastreabilidade das informações é fundamental para o gerenciamento eficiente, facilitando a manutenção. A análise de viabilidade e impacto nas mudanças, seguida pela compreensão do sistema e sua implementação, destaca a importância da documentação atualizada. Após a mudança, a atualização dos documentos é essencial para garantir que a documentação reflita precisamente o estado do sistema (Teixeira e Giglio, 2017). Os mesmos ainda afirmam que, a dificuldade atual da gerência de requisitos na manutenção encontra-se no fato que o foco das empresas é voltado principalmente para a etapa de desenvolvimento do *software* e mudanças são realizadas sem a devida análise de impacto e sem a atualização da documentação, gerando falhas, erros e reduzindo a qualidade do *software*. Com isso, identificamos que para atuar na manutenção de um *software* de forma adequada, uma das informações mais valiosas é ter o conhecimento máximo das informações, assim como conhecer detalhadamente a necessidade que vai ser corrigida ou implementada.

Foi reafirmada a importância de adotar métodos de medição da probabilidade e do impacto dos riscos, proporcionando uma base quantitativa para a priorização de tarefas de manutenção. Essa abordagem não apenas facilita a identificação de áreas críticas, mas também possibilita a alocação eficiente de recursos, assegurando que as demandas de manutenção sejam atendidas de maneira proporcional aos riscos associados.

Contudo, iniciativas de implementação de processos relacionados à manutenção e evolução de *software* podem enfrentar desafios. Conforme observado no artigo 2 por Lucas (2016), alguns membros da equipe demonstraram certa indisciplina, mas esse comportamento foi ajustado pelo coordenador da equipe. Além disso, o período de adaptação da equipe aos novos conceitos foi um tanto difícil, e palestras foram realizadas para compartilhar conhecimentos sobre as novas práticas. Em relação ao artigo 1 de Aparecido et al. (2010), embora não tenha havido uma redução significativa no número de solicitações de manutenção urgentes após a implementação do processo, algumas melhorias foram evidentes: o tempo de implementação das modificações foi reduzido.

Retomando a pergunta de pesquisa: “Como a gestão da manutenção de *software*, com foco na priorização de demandas baseada no impacto, influencia a qualidade das entregas após a transição para a fase de manutenção”, mesmo os artigos que não abordam diretamente

a análise de impacto (como, por exemplo, a proposta de uma disciplina sobre manutenção e a taxonomia de riscos), os trabalhos discutem a importância dos riscos associados à manutenção de *software*, uma área normalmente subestimada. Os estudos de caso de empresas destacam que a implementação de processos para gerenciar a fase de manutenção de *software*, que incluem a análise de riscos, resultou em melhorias significativas na entrega de tarefas, reduzindo a incidência de itens entregues fora do prazo e defeitos entregues.

A gestão eficaz da manutenção de *software* é fundamental para garantir a qualidade e o valor contínuo do produto. Esta revisão destacou a necessidade de abordagens que considerem os riscos e impactos das mudanças no *software*, visando melhorar a eficiência e a confiabilidade das operações de manutenção.

Como perspectivas para trabalhos futuros, seria útil investigar a viabilidade de conduzir avaliações de impacto para mudanças que envolvem hardware, como migrações de servidores, websites, *Data Centers*, entre outras. Além disso, considerando que muitas publicações de artigos brasileiros são disponibilizados em inglês, seria interessante realizar uma análise similar utilizando fontes de dados internacionais ou artigos publicados em inglês.

As disciplinas do curso de Análise e Desenvolvimento de Sistemas do IFSP aplicadas neste trabalho de conclusão de curso incluem: Engenharia de *Software*, Gestão de Projetos, Qualidade de *Software*, Metodologia de Pesquisa Científica e Tecnológica e Arquitetura de *Software*.

Referências

ANQUETIL, Nicolas; DIAS, Márcio Greyck Batista. **Proposta de uma disciplina sobre manutenção de *software* nas graduações em computação**, Anais do Congresso da Sociedade Brasileira de Computação de São Leopoldo, RS, 22 jul. 2005. Disponível em: https://www.researchgate.net/profile/Nicolas-Anquetil/publication/266532126_Proposta_de_uma_disciplina_sobre_manutencao_de_software_nas_graduacoes_em_computacao/links/55c1f29308ae4a2aa89326b4/Proposta-de-uma-disciplina-sobre-manutencao-de-software-nas-graduacoes-em-computacao.pdf . Acesso em: 6 set. 2023.

APARECIDO, Gladston J.; MALTA, Marcelo Nassau; ALMEIDA, Humberto Mossri de; MARQUES-NETO, Humberto T.; VALENTE, Marco Túlio. **Proposta e Avaliação de um Processo para Agrupamento de Solicitações de Manutenções**, IX Simpósio Brasileiro de Qualidade de *Software*, 7 jun. 2010. Disponível em: https://doi.org/10.5753/sbqs.2010.15429 . Acesso em: 5 fev. 2024.

HATTORI, Lile Palma. **Análise probabilística de impacto de mudanças baseada em históricos de mudanças do *software***, Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Campina Grande, 27 fev. 2008. Disponível em: http://dspace.sti.ufcg.edu.br:8080/jspui/handle/riufcg/12221 . Acesso em: 5 fev. 2024.

KITCHENHAM, B. Guidelines for performing Systematic Literature. **Reviews in *Software* Engineering**, [s. l.], ed. 2.3, 9 jul. 2007. Disponível em: https://legacyfileshare.elsevier.com/promis_misc/525444systematicreviewsguide.pdf . Acesso em: 23 ago. 2023.

LIMA, Bruno Santos de. **Uma abordagem de priorização para apoiar o pagamento de dívida técnica auto-admitida em código-fonte**, Dissertação (Bacharel em Ciência da

Computação) – Universidade Estadual Paulista (Unesp), 27 jan. 2021. Disponível em: https://repositorio.unesp.br/items/50678ddd-72fb-4dc1-80b1-3fa58b61abcc. Acesso em: 31 jan. 2024.

LUCAS, Ana Paula Lima. **Gestão da manutenção de *software*: Um estudo de caso**, Dissertação (Mestrado em Informática) – Pontifícia Universidade Católica do Rio de Janeiro, 27 set. 2016. Disponível em: https://www.maxwell.vrac.puc-rio.br/30262/30262.PDF . Acesso em: 5 fev. 2024.

NETO, João Rota. **O Manifesto Ágil**. Universidade Tecnológica Federal do Paraná, 2002. Disponível em:http://paginapessoal.utfpr.edu.br/frufrek/pos-web/p/arquivos/O_manifesto_agil.pdf. Acesso em: 02 nov. 2023.

PADUELLI, Mateus Maida. **Manutenção de *Software*: problemas típicos e diretrizes para uma disciplina específica**. Dissertação (Mestrado em Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, University of São Paulo, São Carlos, 2007. Disponível em: https://www.teses.usp.br/teses/disponiveis/55/55134/tde-21062007-154606/publico/DissertacaoMateusMaidaPaduelli_revisada.pdf. Acesso em: 03 nov. 2023.

RAMOS, André L. B. M.; LIMA, Thiago L. Luna de; CUNHA, Lívia Maria R. V.; TAVARES, Rafael L. A. **Práticas Ágeis Aplicadas a um Processo de Manutenção de *Software*: Um Relato de Experiência**, Anais do XII Simpósio Brasileiro de Qualidade de *Software*, 1 jun. 2013. Disponível em: https://doi.org/10.5753/sbqs.2013.15298 . Acesso em: 5 fev. 2024.

SALGUEIRO, Luiz Fernando; ARTE, Marcus Vinicius Do. **Modelos de Qualidade de *Software***, Projeto de Graduação (Bacharel em Sistemas de Informação) – Escola de Informática Aplicada da Universidade Federal do Estado do Rio de Janeiro (UNIRIO), 2005. Disponível em: https://bsi.uniriotec.br/wp-content/uploads/sites/31/2020/05/200508SalgueiroMarcus.pdf. Acesso em: 25 jan. 2024.

SERAFIM, Gabriela P; DUARTE, Ana M. Debiasi; DUARTE, Denio. **Análise Quantitativa da Aquisição de Dívida Técnica em Sistemas Legados**, Revista de Sistemas e Computação, Salvador, v. 12, 1 jan. 2022. Disponível em: https://revistas.unifacs.br/index.php/rsc/article/download/7569/4507 . Acesso em: 31 jan. 2024.

SILVA, João Gabriel Trajano da; VIEIRA, Robson; BERGAMASCHI, Marcelo Pereira; PENHA, Renato. **Estudo sobre o gerenciamento de riscos técnicos em manutenções corretivas**, Unisanta Science and Technology, v. 2, 6 dez. 2013. Disponível em: https://periodicos.unisanta.br/index.php/sat/article/download/191/177 . Acesso em: 5 fev. 2024.

SOMMERVILLE, Ian. **Engenharia de *Software***. 6. ed. São Paulo: Pearson Education do Brasil, 2003.

SOMMERVILLE, Ian. **Engenharia de *Software***. 9. ed. São Paulo: Pearson Education do Brasil, 2011.

TEIXEIRA, Rafaela Emília Zanetti; GIGLIO, Giuliano Prado de Morais, Estudo do desenvolvimento de uma ferramenta de Gerência de Requisitos para apoio aos processos de manutenção de *software*, **Caderno de Estudos em Sistemas de Informação**, v. 4, n. 2, 2017. Disponível em: http://seer.uniacademia.edu.br/index.php/cesi/article/view/1254 . Acesso em: 21 fev. 2024.

THOMAZINHO, Hellen Christine Seródio. **Uma análise do processo de manutenção de *software* em organizações de tecnologia da informação**, Dissertação (Mestrado em Informática) – Universidade Tecnológica Federal do Paraná, 19 fev. 2018. Disponível em: http://repositorio.utfpr.edu.br/jspui/handle/1/3275 . Acesso em: 5 fev. 2024.

TORRES, Joaquim. Gestão de produtos de *software*: **Como aumentar as chances de sucesso do seu *software***. 3. ed. [S. l.]: Casa do Código, 2020.

VALE, Alisson. Manutenção Ágil - Parte 1: **Dá pra ser Ágil na manutenção de produtos de *software*?**. Linkedin, 24 jan. 2019. Disponível em: https://www.linkedin.com/pulse/d%C3%A1-pra-ser-%C3%A1gil-na-manuten%C3%A7%C3%A3o-de-produtos-software-alisson-vale/ . Acesso em: 6 set. 2023.

VARGAS, Alessandra Alves Fonseca; PEREIRA, João Victor dos Santos. **Gerenciamento da Manutenção de *Software*, Pesquisa & Educação a Distância**, no. 16, São Paulo, 2019. Disponível em: http://www.revista.universo.edu.br/index.php?journal=2013EAD1&page=article&op=view&path%5B%5D=8294&path%5B%5D=3992 . Acesso em: 5 fev. 2024.

WEBSTER, Kênia Pereira Batista; OLIVEIRA, Káthia Marçal de; ANQUETIL, Nicolas. **Taxonomia de Riscos para Manutenção de *Software***, IV Simpósio Brasileiro de Qualidade de *Software*, 6 jun. 2005. Disponível em: https://doi.org/10.5753/sbqs.2005.16153. Acesso em: 6 set. 2023.

Documento Digitalizado Público

Anexo I - artigo - TCC

Assunto: Anexo I - artigo - TCC
Assinado por: Andre Constantino
Tipo do Documento: Relatório
Situação: Finalizado
Nível de Acesso: Público
Tipo do Conferência: Documento Digital

Documento assinado eletronicamente por:

- **Andre Constantino da Silva**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 18/03/2024 19:45:14.

Este documento foi armazenado no SUAP em 18/03/2024. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifsp.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 1615888
Código de Autenticação: 8065f651d1