UNIVERSIDADE FEDERAL DOS VALES DO JEQUITINHONHA E MUCURI

Bacharelado em Sistema de Informação

Matheus Alexandre Silva

MAPEAMENTO SISTEMÁTICO: SISTEMAS LEGADOS E SUA REENGENHARIA

Diamantina

2023

Matheus Alexandre Silva

MAPEAMENTO SISTEMÁTICO: SISTEMAS LEGADOS E SUA REENGENHARIA

Trabalho de Conclusão de Curso apresentado ao Departamento de Computação como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação pela Universidade Federal dos Vales do Jequitinhonha e Mucuri.

Orientador: Profa. Ma. Caroline Miranda Barroso

Diamantina

2023

ERRATA

SILVA, Matheus Alexandre. **Mapeamento Sistemático:** Sistemas Legados sua Reengenharia. 2023. 44 p. Trabalho de Conclusão de Curso (Graduação em Sistemas de Informação) – Faculdade de Ciências Exatas e Tecnológicas, Universidade Federal dos Vales do Jequitinhonha e Mucuri, Diamantina, 2023.

Página	Linha	Onde se lê:	Leia-se:
3	2	Sistemas Legados e sua Reengenharia	Mapeamento Sistemático: Sistemas Legados e sua Reengenharia

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DOS VALES DO JEQUITINHONHA E MUCURI

FOLHA DE APROVAÇÃO

Matheus Alexandre Silva

SISTEMAS LEGADOS E SUA REENGENHARIA

Trabalho de Conclusão de Curso apresentado ao Departamento de Computação como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação pela Universidade Federal dos Vales do Jequitinhonha e Mucuri.

Aprovada em 31/01/2023

BANCA EXAMINADORA

Prof (a) Caroline Miranda Barroso
Faculdade de Ciências Exatas - UFVJM

Caroline Queiroz Santos
Faculdade de Ciências Exatas - UFVJM

Cláudia Beatriz Berti
Faculdade de Ciências Exatas - UFVJM

Documento assinado eletronicamente por **Caroline Queiroz Santos**, **Servidor (a)**, em 31/01/2023, às 15:19, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Claudia Beatriz Berti**, **Servidor (a)**, em 31/01/2023, às 15:21, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Caroline Miranda Barroso**, **Servidor (a)**, em 31/01/2023, às 23:27, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://sei.ufvjm.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **0967888** e o código CRC **4429DE8D**.

Referência: Processo nº 23086.000729/2023-83 SEI nº 0967888

Dedico aos meus pais
Janilson de Souto Silva
Vânia Alcantara Alexandre
Pelo amor e carinho que me dão.
Pelos sacrifícios que passaram para eu conseguir essa vitória.

AGRADECIMENTOS

Deixo meu agradecimento primeiro à Deus por esta vivo e com saúde para realização dessa conquista que é a graduação.

Também agradecer à minha família, que me apoiou desde início desta trajetória, não me permitindo faltar nada, sempre encorajando e correndo atras como uma rede de apoio e proteção que me cerca para a cada dia eu realizar essa conquista.

Também agradeço à minha namorada, que iniciou o curso comigo e fizemos praticamente todas as aulas juntos, sendo minha parceira nos trabalhos das disciplinas e estudos nos momentos que antecedem as provas.

Aos professores do curso de Sistemas que me ensinaram uma profissão, me mostrou o “caminho das pedras” para me tornar um profissional.

A todos da Universidade, por proporcionar estrutura, ambiente e sempre de portas abertas para todos que procuram educação. Pois sem a universidade federal, o ensino superior não seria exclusivo para todos.

Em especial um agradecimento para minha orientadora Caroline Miranda Barroso, que aceitou esse desafio, que é esse Trabalho de Conclusão de Curso. Que antes era um monte de ideias, que com conversas e reuniões, conseguimos transformar nessa monografia. Além de me apresentar o Tiago, que me ajudou bastante na construção desse texto.

Obrigado à todos que participaram nesta etapa da minha vida... E talvez até breve.

“[...]As pedras não vão impedir
O destino que Deus reservou
Saber resistir é o segredo de um bom vencedor [...]”

Frederico Fagundes Fernandes Camacho / Jorge Leandro Pereira Da Silva / Diogo Mendonca Nogueira.

RESUMO

Na construção de um software, empresas investem nas melhores tecnologias, nas melhores metodologias existentes no mercado para aplicar e abstrair as regras de negócios. Ao longo do tempo, o software recebe constantes manutenções, tornando sua manutenção seja custosa. Embora seus dados sejam muito valiosos, o software acaba ficando ultrapassado, de modo a se tornar um Sistema Legado. Para solucionar esse problema, uma alternativa é a realização da Reengenharia. Sendo assim, este trabalho teve o propósito de realizar um levantamento acerca do estado da arte relacionado aos conceitos de Sistemas Legados e Reengenharia de Software a fim de identificar as melhores práticas da Reengenharia de Software para a atualização de um Sistema Legado. Para tal, foi realizada uma pesquisa qualitativa descritiva em dados secundários, com caráter exploratório por meio de mapeamento sistemático na literatura. O mapeamento foi realizado ao longo de trinta dias e, sobre os artigos encontrados foram aplicados dois filtros para o refinamento e seleção dos artigos mais relacionados à pesquisa. Os resultados evidenciam a evolução da definição de Sistemas Legados, a ampliação das subdivisões da Reengenharia de Software e apontam a técnica de Engenharia Reversa e a metodologia *Little Chicken* como as mais utilizadas até o momento.

Palavras chave: Sistema Legado. Reengenharia de Software. Mapeamento Sistemático. *Little Chicken*. Engenharia Reversa. Estado da Arte.

ABSTRACT

When building software, companies invest in the best technologies, in the best methodologies on the market to apply and abstract the business rules. Over time the software receives constant maintenance, making maintenance costly. Although your data is valuable, the software ends up becoming outdated, in order to become a Legacy System. To solve this problem, an alternative is to perform Reengineering. Therefore, this work had the purpose of carrying out a survey about the state of art related to the concepts of Legacy Systems and Software Reengineering, in order to identify the best practices of Software Reengineering for updating a Legacy System. To this end, a descriptive, qualitative research was carried out on secondary data, with an exploratory character through systematic mapping in the literature. The mapping was carried out over thirty days and, on the articles found, two filters were applied for the refinement and selection of the articles most related to the research. The results show the evolution of the definition of Legacy Systems, the expansion of Software Reengineering subdivisions and point out the Reverse Engineering technique and the Little Chicken methodology as the most used ones so far.

Keywords: Legacy System. Software Reengineering. Systematic Mapping. Little Chicken. Reverse Engineering. State of Art.

LISTA DE ILUSTRAÇÕES

Figura 1 – Representação de Reengenharia de Software.....19
Figura 2 – Representação da arquitetura de migração *Chicken Little*.....21
Figura 3 – Representação de Engenharia Reversa em ciclo de software.....22
Figura 4 – Quantidade de Artigos por Sub Questão (SQ).....30
Figura 5 – Artigos selecionados por ano de publicação.....31

LISTA DE QUADROS

Quadro 1 – *String* de busca.....24
Quadro 2 – Critérios para a Seleção dos Artigos.....25
Quadro 3 – Descrição de atividades para elaboração da pesquisa.....27
Quadro 4 – Resultado do Mapeamento Sistemático.....29
Quadro 5 – Vantagens e Desvantagens *Little Chicken*.....35
Quadro 6 – Definições por Tema.....36

LISTA DE ABREVIATURAS E SIGLAS

ACM	Association for Computing Machinery
FIFA	Federação Internacional de Futebol Associado
IEEE	Instituto de Engenheiros Eletricistas e Eletrônicos

SUMÁRIO

1 INTRODUÇÃO 14
1.1 Justificativa 15
1.2 Objetivo Geral 16
1.3 Objetivos Específicos 16

2 FUNDAMENTAÇÃO TEÓRICA 17
2.1 Referencial Teórico 17
2.1.1 Sistema Legado 17
2.1.2 Reengenharia 18
2.1.3 Reengenharia de Software 19
2.1.3.1 Chicken Little 20
2.1.3.2 Engenharia Reversa 21
2.2 Mapeamento Sistemático 22
2.2.1 Planejamento do Mapeamento Sistemático 23
2.2.2 Estratégia de Busca dos Estudos Primários 23
2.2.3 Critérios de Seleção de Artigos 24
2.2.4 Procedimentos para Seleção de Artigos 25
2.2.4.1 Filtro 1 25
2.2.4.2 Filtro 2 25
2.2.5 Execução do Mapeamento Sistemático 25
2.2.5.1 Filtro 1 25
2.2.5.2 Filtro 2 26

3 METODOLOGIA 27

4 ANÁLISE E DISCUSSÃO DOS RESULTADOS 29

5 CONSIDERAÇÕES FINAIS 39

REFERÊNCIAS 41

1 INTRODUÇÃO

O desenvolvimento de software possui vários processos até sua entrega. Muitos desses processos tendem a gerar riscos ao utilizar tecnologias antigas e metodologias inadequadas. Uma das partes mais importantes para a prevenção dos riscos é sua manutenção. A falta de manutenção e cuidado com a atualização arquitetural tende a tornar o sistema cada vez mais difícil e complexo (CANTO, 2021). Assim, denominamos de Softwares/Sistemas Legados, possuindo várias estratégias e análise para cada manter uma evolução adequada (CHERVENSKI, 2019).

A falta de atualização arquitetural pode ser chamada de Softwares Legados, possuindo várias estratégias e análise para cada tipo de manter uma evolução adequada do software (CHERVENSKI, 2019). De acordo com Akamine (2015) define “Sistemas Legados são considerados como sistemas sociotécnicos baseados em computadores que foram desenvolvidos no passado utilizando tecnologia mais antiga ou considerada obsoleta” (AKAMINE, 2015, p. 24). Borba (2017) complementa que “Sistema Legado vem como qualquer sistema da informação que resiste significativamente a qualquer modificação e evolução ao longo do tempo*”* (BORBA, 2017, p. 21).

A partir do momento que um Sistema Legado apresenta dificuldades em realizar manutenções, possui uma documentação incompleta ou antiquada e a tecnologia utilizada desde sua criação ser muito antiga, tal sistema pode indicar a necessidade da realização de um processo de reengenharia. Outro aspecto muito importante para a realização da reengenharia é o valor que o Sistema Legado tem para a empresa. Vale ressaltar, que nem todos os sistemas antigos, precisam se submeter a uma reengenharia. Um exemplo de sistema que possui grandes chances de não sofrer reengenharia, são sistemas bancários, pelo fato de possuir inúmeros dados de clientes e devido a aspectos ligados à segurança dos dados oferecidos pelo sistema. É importante observar que uma Reengenharia de Software mal sucedida, pode ocasionar grandes prejuízos para a empresa.

Uma forma de atualizar um Sistema Legado é realização da Reengenharia de Software. Ela oferece meios para atualizar os sistemas escritos de forma errônea, analisar a regra de negócio, alterar a arquitetura utilizada e suprir potenciais faltas de documentação (CANTO, 2021). Por meio da Reengenharia de Software é possível abstrair as funcionalidades principais do sistema e analisar o modelo construído (ALMEIDA, 2009). De acordo com Almeida (2009), a Reengenharia de Software consiste em reorganizar e modificar

um software, com o intuito de corrigir lógicas e códigos ruins, realizar mudanças e ajustes de tecnologia e potencial migração para novas plataformas.

Assim, a questão da presente pesquisa consiste em verificar qual estado da arte relacionado aos conceitos de Sistemas Legados e Reengenharia de Software a fim de identificar as melhores práticas da Reengenharia de Software para a atualização de um Sistema Legado.

1.1 Justificativa

Desde seu desenvolvimento, um software possui problemas ao se tornar obsoleto, com funções que não são úteis ao usuário ou até mesmo para outros sistemas que o integram (OLIVEIRA, 2003). Softwares obsoletos demandam a realização de constantes manutenções, tornando a manutenção e tempo de desenvolvimento mais custosas para a empresa.

Devido aos problemas que podem ocorrer em um Sistema Legado, a solução mais utilizada é a construção de um sistema do zero e não utilizar mais o Sistema Legado, no entanto, podem acontecer situações que não sejam viáveis utilizar essa metodologia, como: o longo tempo de desenvolvimento para criar um sistema do zero, a perca de regras de negócios contidas no Sistema Legado, dificuldades de realizar a transição de dados que o Sistema Legado possui e manter a segurança já existente no Sistema Legado que foi adquirida por meio de manutenção (CHERVENSKI, 2019).

Uma técnica utilizada para manter o Software Legado, mas não sofrer tanto com a manutenção é a Reengenharia de Software. De acordo com Oliveira (2003), a Reengenharia de Software é a parte responsável pela manutenção de sistemas que foram desenvolvidos de forma incompleta, tanto de documentação quanto de tecnologia. Por meio dela, é possível atualizar um Sistema Legado reduzindo a complexidade de sua manutenção e infraestrutura.

Com tudo, podemos salientar que a presente pesquisa se justifica pela importância de apresentar as melhores práticas da Reengenharia de Software para a atualização de um Sistema Legado, por meio do levantamento do estado da arte relacionado aos conceitos de Sistemas Legados e Reengenharia de Software.

1.2 Objetivo Geral

Realizar um levantamento acerca do estado da arte relacionado aos conceitos de Sistemas Legados e Reengenharia de Software a fim de identificar as melhores práticas da Reengenharia de Software para a atualização de um Sistema Legado.

1.3 Objetivos Específicos

a. Realizar um mapeamento sistemático sobre pesquisas que envolvam Sistemas Legados, Reengenharia de Software e suas principais metodologias.
b. Identificar processos, técnicas e métodos para atualizar um Sistema Legado utilizando a Reengenharia de Software.
c. Selecionar as melhores práticas da Reengenharia de Software para atualizar um Sistema Legado.

Esta pesquisa está divida em quatro capítulos além da introdução. O capítulo 2 de Fundamentação Teórica, que contém o referencial teórico (2.1) e o Mapeamento Sistemático (2.2), o capítulo 3 apresenta a Metodologia, o capítulo 4 trata sobre a Análise e Discussão dos Resultados e por fim o capítulo 5 com as Considederações Finais.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta a fundamentação teórica desta pesquisa contendo duas seções. A primeira seção, 2.1, trataremos sobre o Referencial Teórico, a segunda seção, 2.2, abordaremos sobre o Mapeamento Sistemático.

2.1 Referencial Teórico

Nesta seção será apresentado o Referencial Teórico desta pesquisa. Ele contém abordagens sobre Sistemas Legados, Reengenharia e Reengnharia de Software. Um destaque para o tema Reengenharia de Software, que possui subseção de *Chicken Little* e Engenharia Reversa.

2.1.1 Sistema Legado

Softwares são criados para suprir e automatizar as necessidades e objetivos de suas empresas. Entretanto, com o passar do tempo existe a necessidade de realizar manutenções e alterações nos requisitos do sistema impostos pelas necessidades dos usuários e das empresas que os utilizam.

As inúmeras mudanças realizadas neste sistema ao longo do tempo podem ocasionar em dificuldades, falta de evolução e renovação, tornando o sistema incapaz de realizar o máximo de tarefas necessárias. Assim, sistemas construídos há mais tempo e que sofrem muitas manutenções, são chamados de Sistemas Legados (CHERVENSKI, 2019).

Sistemas Legados não possuem uma única definição, tendo várias interpretações na comunidade e na literatura. Nilsson (2015) e Bisbal *et al*. (1999) (*apud* Chervenski, 2019), que definem Sistemas Legados em softwares construídos a muito tempo e utilizados a aproximadamente 10 anos ou mais, com tecnologias ultrapassadas, metodologias de desenvolvimento antigas e com documentação inadequadas.

Borba (2017) complementa que qualquer software resistente a manutenção e modificações por um período longo pode carregar prejuízos à empresa detentora. Alguns desses prejuízos são alto custo de manutenção, documentação incompleta, complexidade para integrar com outros softwares de terceiros, além de tecnologias ultrapassadas.

Já Akamine, (2015), apresenta algumas características do sistema que indicam que ele está se tornando legado, são eles:

a) Sua vida útil;
b) Seu tamanho de código fonte e funcionalidades;
c) Uma linguagem de programação antiga;
d) Quantidade de manutenção realizada no sistema;
e) Sua importância.

Assim, Sistemas Legados são sistemas que possuem anos de uso e desenvolvidos com tecnologia ultrapassada. Tal Sistema, não consegue acompanhar a evolução da tecnologia, se tornando desatualizado tecnologicamente (PINTO; BRAGA, 2004).

2.1.2 Reengenharia

O processo de Reengenharia tem como seu objetivo modernizar um software que é difícil de ser mantido, porém tem uma extrema importância para a organização que o utiliza. Como Almeida (2009) salienta, a Reengenharia não é só reescrita de código ou arquitetura, e sim, realizar redocumentação do sistema, organizar e reestruturar o sistema, remodelar o sistema para uma linguagem ou *framework* mais atual e atualizar a estrutura dos dados contidos. O mesmo, vai muito além de software, mas também pode ser aplicado em empresas, pessoas e processos, com um objetivo final de melhorias (OLIVEIRA, 2003).

Para Piekarski e Quináia, (2000) a Reengenharia tem uma definição de iniciar seu desenvolvimento baseado em um sistema já existente, ao contrário de iniciar seu desenvolvimento iniciado pela especificação escrita.

Para iniciar o estudo de Reengenharia de Software, deve-se entender sobre Reengenharia de Processos. A Reengenharia de Processos consiste em remodelar uma organização, já na Reengenharia de Software, o que faz é remodelar um sistema (OLIVEIRA, 2003). Almeida (2009) traduz que a Reengenharia de Processos, vai além de remodelar todas as tarefas realizadas, pois procura um resultado final inovador e satisfatório. Com isso, podemos concluir que, para uma empresa ter um bom resultado na sua Reengenharia de Software, é fundamental reprojetar seu processo de desenvolvimento, visando qualidade, custo e serviço.

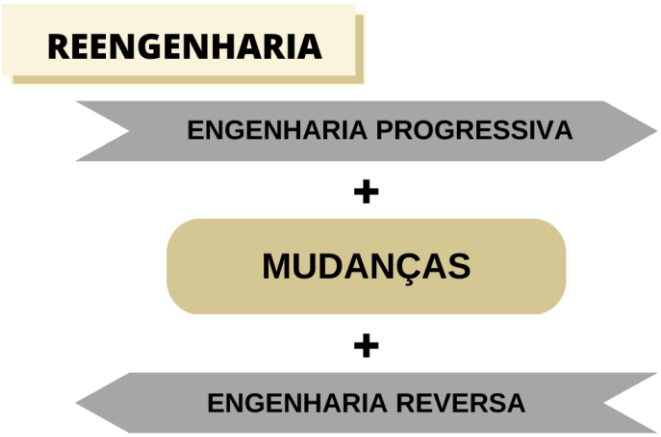
2.1.3 Reengenharia de Software

A Reengenharia de Software tem o objetivo de atualizar o Sistema Legado que é crítico para a empresa detentora e sua regra de negócio ou que os dados contidos nele são de extrema importância, e este sistema possui dificuldades para realizar manutenção e custos altos para a empresa (CANTO, 2021).

Assim, Canto (2021) demonstra que na literatura existem várias metodologias utilizadas na Reengenharia de Software. Das metodologias abordadas vale destacar o "*Cold Turkey*" e o "*Chicken Little*". A técnica "*Cold Turkey*" é uma estratégia de refazer o sistema inteiro do zero, ou seja, se utiliza a regra de negócio composta no Sistema Legado e realiza a Reengenharia de Software junto com a Reengenharia de processos. Já a técnica "*Chicken Little*" é uma estratégia de refazer o sistema aos poucos, por partes. Vale ressaltar que a técnica "*Cold Turkey*", é uma estratégia bem perigosa e muitas vezes não eficaz comparado com o "*Chicken Little*".

Juntos às técnicas apresentadas, uma subtécnica muito popular na literatura é a Engenharia Reversa, que consiste em utilizer uma abstração de nível baixo para o nível mais alto, com objetivo de analisar e criar uma documentação até chegar no desenvolvimento de software atualizado (OLIVEIRA, 2003). Esses níveis de abstração são análises da estrutura do software, que pode ser uma documentação, diagramas ou outra forma de representação para facilitar o processo de refatoração, possuindo um objetivo de utilizar como ponto de partida um código ou especificação do sistema criado (PIEKARSKI; QUINÁIA, 2000 e CHAVES, 2004). Assim a Reengenharia pode ser demonstrada na Figura 1 (CHAVES, 2004):

Figura 1- Representação de Reengenharia de Software



Fonte: Chaves, 2004. Adaptado

Na Figura 1, as Mudanças consistem em:

a) Mudança de funcionalidade: São mudanças devido a necessidade da regra de negócio.
b) Mudança de implementação: São mudanças no ambiente de criação, podendo ser linguagem, *framework* ou tecnologias.

A Engenharia Progressiva consta como o ciclo normal do seu desenvolvimento, envolve a coleta de requisitos; implementação dos requisitos; implantação/entrega do desenvolvimento para o cliente, seguindo a sequência estabelecida no início do projeto (PIEKARSKI; QUINÁIA, 2000). Um ponto muito importante sobre a Reengenharia de Software é que ela não é só realizada em Sistemas Legados.

Pode ocorrer casos do sistema utilizar a tecnologia atual, ou até ser recém criado, mas se possui sintomas de problemas e dificuldades de manutenção e for concluído que necessita de reconstrução, a Reengenharia pode ser aplicada (CHAVES, 2004).

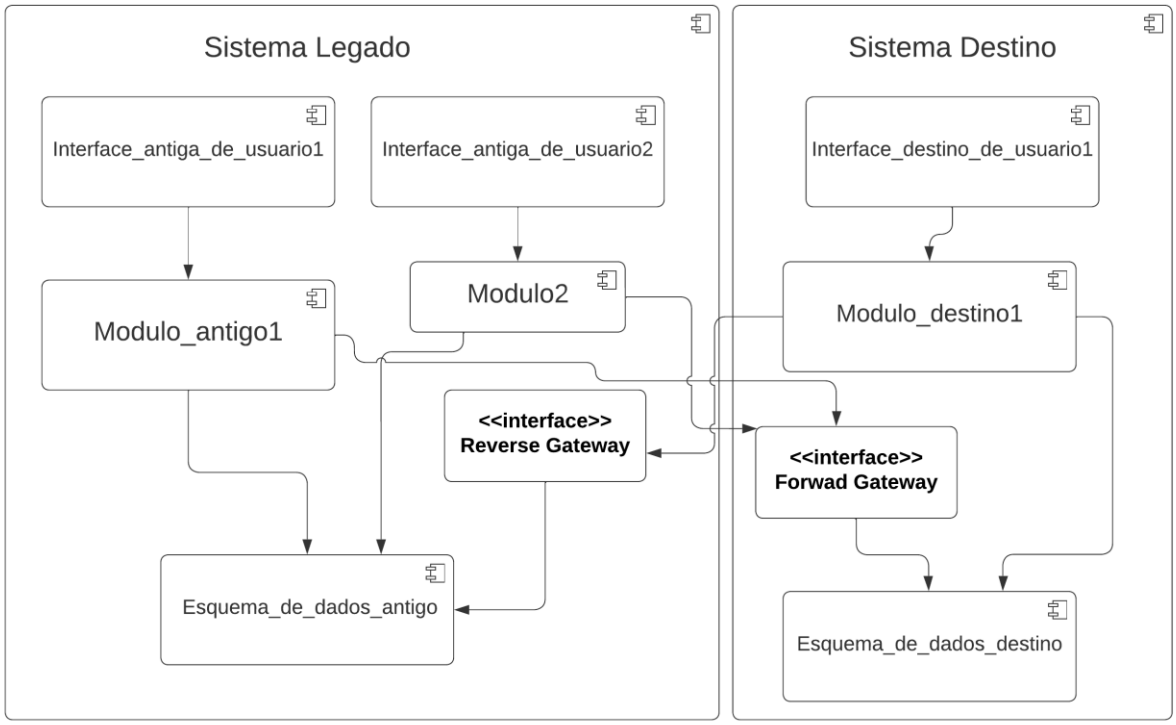
2.1.3.1 Chicken Little

A estratégia *Chicken Little* "tem como principal característica a migração incremental" (IGAWA, 2013, p.42). Ou seja, toda a parte que sofreu a Reengenharia opera em paralelo ao Sistema Legado, que também é denominado pela literatura de sistema destino.

Em seu início, o sistema destino é uma amostra do Sistema Legado, possuindo pequenos dados (ANDRADE, 2016). No decorrer do processo de Reengenharia, a migração de dados e funcionalidades do Sistema Legado para o sistema destino seja completa, o Sistema Legado fica inoperante (ANDRADE, 2016).

Para preparar o sistema destino de forma que ele fique apto para substituir o legado, os dois sistemas trabalham de forma cooperativa, essa cooperação é chamada de *gateways*. De acordo com Igawa (2013), os *gateways* consistem em "um módulo introduzido entre dois componentes para intermediar entre ambos*"* (IGAWA, 2013, p.42). A Figura 2 apresenta como é estruturada a representação da arquitetura de migração *Chiken Little*.

Figura 2 – Representação da arquitetura de migração *Chicken Little*



Fonte: BRODIE e Stonebraker, 1993. Adaptado

Igawa (2013) afirma que o sucesso no *Chicken Little*, o ponto inicial é realizar a migração na parte do sistema onde o módulo seja independente e confirmar que cada passo do processo tenha êxito, deixando mais seguro o processo de migração.

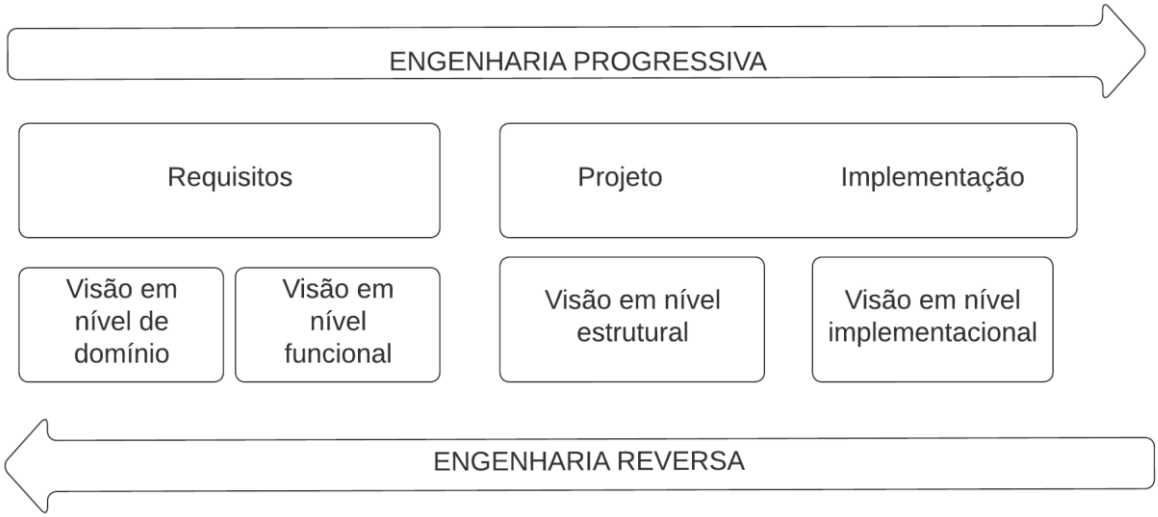
2.1.3.2 Engenharia Reversa

A Engenharia Reversa é parte do processo de software com objetivo de compreender melhor o sistema. De modo geral, pode ser aplicada na manutenção do software, realizar uma Reengenharia em Sistemas Legados ou analisar sistemas concorrentes (ALMEIDA, 2009).

Piekarski e Quináia (2000), conceitua a Engenharia Reversa como processo inverso à Engenharia Progressiva, caracterizado pelas atividades retroativas do ciclo de vida, que partem de um baixo nível de abstração para o mais alto nível. O ciclo de vida do software envolve a sua coleta de requisitos, desenvolvimento e entrega ao cliente. No entanto, na Engenharia Reversa, já temos o sistema pronto, como exemplo uma tela ou um botão. Essa parte do sistema pronto é chamada de nível mais baixo de abstração e o nível mais alto de

abstração é caracterizado pelo planejamento do funcionamento da tela, como exemplo planejar o funcionamento de uma classe ou método ou até mesmo como vai tratar as informações do banco de dados. A Figura 3 apresenta como é estruturada a representação da Engenharia Reversa em ciclo de software.

Figura 3 – Representação de Engenharia Reversa em ciclo de software



Fonte: COSTA, 1997. Adaptado

A expressão “engenharia reversa” tem origem no mundo do hardware, que obtinha o projeto a partir do produto final (PIEKARSKI; QUINÁIA, 2000), ou seja, estudava o produto da concorrência e analisava se colocaria no produto.

Já no ambiente de software, suas técnicas foram adaptadas, sendo uma delas, realizar uma análise básica do sistema e de sua estrutura, partindo do pressuposto que o analista seja outra pessoa que o projetista.

De acordo com Garcia (2005), em sistemas com alta complexidade, a Engenharia Reversa pode ser realizada via semi-automatizada utilizando ferramentas próprias, que analisa a semântica do código legado. Assim que obtiver o resultado esperado, realizar a refatoração, volta ao processo de engenharia progressiva.

2.2 Mapeamento Sistemático

Mapeamento sistemático, tem como objetivo obter uma linha temporal e criar uma base de dados para pesquisas futuras (DAMIAN, 2020). Assim, fica mais fácil identificar as

pesquisas e as lacunas existentes na academia. Por meio dele, é possível identificar o estado da arte de um determinado assunto.

Com intenção de verificar o estado da arte relacionado aos conceitos de Sistemas Legados e Reengenharia de Software a fim de identificar as melhores práticas da reengenharia para a atualização de um Sistema Legado, foi realizado um mapeamento sistemático com base nas diretrizes proposta por Kitchenham e Charters (2007 *apud* Damian, 2020).

2.2.1 Planejamento do Mapeamento Sistemático

Para a realização do mapeamento sistemático foi levantada a seguinte questão: **"Qual ou quais modelagem (modelagens) são aplicadas na Reengenharia de Software em Sistemas Legados?".**

Também foram definidas subquestões (SQ) para maiores resultados.

a) **SQ1: "Quais definições de Software Legado ou Sistema Legado?"** – Definida com o propósito de identificar conceitos e definições para Software Legado ou Sistema Legado.
b) **SQ2: "Quais definições de Reengenharia?" –** Definida com o propósito de identificar conceitos e definições para Reengenharia.
c) **SQ3: "Quais são as técnicas utilizadas para Reengenharia de Software?"** – Definida com o propósito de identificar as técnicas relatadas na literatura para Reengenharia de Software.
d) **SQ4: "Quais pesquisas realizadas de metodologia na Reengenharia de Software e Software/Sistema Legado que contribuíram para inovação?" –** Definida com o objetivo de compreender as pesquisas que podem contribuir a engenharia de software a usar metodologias de Reengenharia em software/Sistema Legado.

2.2.2 Estratégia de Busca dos Estudos Primários

Definição da *string* de busca: Foram escolhidos dois conceitos para representar a Reengenharia de Software, como mostra o Quadro 1:

Quadro 1 – *Strings* de busca

Definição	*String*
Software/Sistema Legado	*"Legacy software systems AND System Legacy"* *"Legacy software systems OR System Legacy"*
Reengenharia de Software	*"Legacy software systems AND Reengineering"* *"Legacy software systems OR Reengineering"*

Fonte: Damian (2020). Adaptado.

A primeira definição está relacionada ao contexto de software/Sistema Legado de forma ampla. A segunda definição é relacionada a Reengenharia de Software, focando nas diferentes utilizações, metodologias e definições da Reengenharia de Software existente.

Fonte de pesquisa: A biblioteca digital utilizada foram o ACM (*Association for Computing Machinery*) e IEEE *Xplore* (Instituto de Engenheiros Eletricistas e Eletrônicos). Os motivos da escolha foram a possibilidade de realizar filtros de pesquisa e adquirir as publicações gratuitamente.

Idiomas dos artigos: Os idiomas escolhidos foram o inglês e português. O idioma inglês, foi adotado por sua maioria em conferências e a dominância nos principais artigos da área da tecnologia. O idioma português foi adotado, por essa pesquisa ser em uma Universidade com idioma português.

2.2.3 Critérios de Seleção de Artigos

Baseado em Damian (2020) os critérios para selecionar os artigos são originados das strings de buscas. Com isso, "os critérios de seleção tem como objetivo identificar estudos que forneçam evidências para a questão de pesquisa" (DAMIAN, 2020, p. 48). Assim, foram criados critérios de inclusão e critérios de exclusão, para refinar os artigos do mapeamento, conforme indicado no Quadro 2:

Quadro 2 – Critérios para a Seleção dos Artigos

Critérios de Inclusão (CI)	
CI1	Podem ser selecionadas publicações que apresentam definições de Sistemas Legados ou softwares legados.
CI2	Podem ser selecionadas publicações que apresentam Reengenharia ou Reengenharia de Software.
CI3	Podem ser selecionadas publicações que apresentam técnicas ou modelagem de Reengenharia de Software.
Critérios de Exclusão (CE)	
CE1	Não serão selecionadas publicações que não satisfaçam a nenhum critério de inclusão.
CE2	Não serão selecionadas publicações em que o idioma seja diferente do exigido.
CE3	Não serão selecionadas publicações de artigos duplicados.

Fonte: Elaborado pelo autor.

2.2.4 Procedimentos para Seleção de Artigos

2.2.4.1 Filtro 1

A seleção dos artigos teve o seguinte formato:

a) Leitura dos resumos e/ou abstract dos artigos.
b) Remoção de artigos duplicados, conforme o critério de exclusão C3, nas *strings* pesquisadas e nas bibliotecas digitais procuradas.
c) Seleção dos artigos que atenderam aos critérios.

2.2.4.2 Filtro 2

A seleção dos artigos teve o seguinte formato:

a) Leitura completa dos artigos
b) Remoção dos artigos que não atenderam aos critérios

2.2.5 Execução do Mapeamento Sistemático

2.2.5.1 Filtro 1

O mapeamento sistemático foi realizado durante 10 dias úteis do mês de novembro de 2022. Foi reservado 1 dia para cada *String* e para cada biblioteca digital. Nesse período foram adquiridos 75 artigos. Dos 75 artigos encontrados, 39 foram selecionados da biblioteca ACM e 36 foram selecionados da biblioteca IEEE *Xplore*. Destes artigos encontrados, apenas 39 atenderam os critérios de inclusão, sendo 21 da ACM e 18 da IEEE *Xplore*.

2.2.5.2 Filtro 2

Na segunda filtragem dos artigos, foi realizado em um período de 10 dias do mês de dezembro. Foi reservado 1 semana para cada biblioteca selecionada. Nesse período 9 artigos da ACM foram selecionados por atenderem pelo menos 1 critério. Já os artigos do IEEE *Xplore* foram selecionados apenas 4 artigos pelo mesmo motivo.

3 METODOLOGIA

A presente pesquisa possui uma abordagem qualitativa e consiste em um estudo documental de caráter descritivo e exploratório (ARAGÃO, 2011; SCARPA; MARANDINO, 1999) envolvendo dados secundários, na qual buscamos realizar um levantamento acerca do estado da arte relacionado aos conceitos de Sistemas Legados e Reengenharia de Software a fim de identificar as melhores práticas da Reengenharia de Software para a atualização de um Sistema Legado.

Nesse intuito, realizamos um mapeamento sistemático em duas bibliotecas, ACM (*Association for Computing Machinery*) e IEEE *Xplorer* (Instituto de Engenheiros Eletricistas e Eletrônicos), e com o resultado obtido buscamos descobrir as principais técnicas e metodologias utilizadas pela academia ao lidar com softwares legados.

Resumidamente, as atividades realizadas para a elaboração da presente pesquisa podem ser distribuídas em 5 etapas, conforme o Quadro 3 apresenta:

Quadro 3 - Descrição de atividades para elaboração da pesquisa

Etapas	Descrição das Atividades
1°	Levantamentos iniciais sobre o tema para o referencial teórico
2°	Seleção dos critérios para a realização do mapeamento sistemático
3°	Aplicação do Mapeamento Sistemático
4°	Coleta dos resultados
5°	Análise e Discussão dos resultados

Fonte: Elaborado pelo autor.

Na etapa 1, que corresponde ao levantamento inicial sobre o tema, foi realizado pesquisas no *Google* Acadêmico (*Google Scholar*) na procura de artigos, porem foram encontrados poucos artigos sobre o tema e sua maioria foram encontrados trabalho de conclusão de curso de outras universidades. Com isso houve pesquisa na biblioteca do Departamento de Computação da Universidade Federal dos Vales do Jequitinhonha e Mucuri (UFVJM) para conter neste trabalho e também na realização de potencial trabalho futuro. No entanto, não foram encontrado nenhum trabalho no Departamento de Computação da UFVJM, motivando a realização de um mapeamento sistematico.

A seleção dos critérios (etapa 2) foram baseadas de acordo com tema da pesquisa, principalmente nas sub questões apresentadas, para melhor obtenção dos resultados. As etapas

3 e 4, referente à aplicação do mapeamento sistemático e à coleta dos artigos foram realizadas paralelamente. Consistiu no período de dez dias úteis com três horas de duração, na biblioteca da Universidade Federal dos Vales do Jequitinhonha e Mucuri. Como a coleta dos artigos foram realizados no período do evento Copa do Mundo da Fifa, nos dias dos jogos da seleção brasileira, não houve coleta de dados, devido a falta de condução para a biblioteca. Na etapa 5 que envolveu a análise e discussão dos resultados, foram examinados os documentos obtidos de forma a observar a questão da pesquisa e atender seus objetivos, geral e específicos.

4 ANÁLISE E DISCUSSÃO DOS RESULTADOS

O Quadro 4 apresenta os artigos selecionados após a aplicação dos filtros. Os resultados encontrados nesse mapeamento serão utilizados nos desenvolvimento dos trabalhos relacionados e nas estratégias de atualização adotadas.

Quadro 4 – Resultado do Mapeamento Sistemático

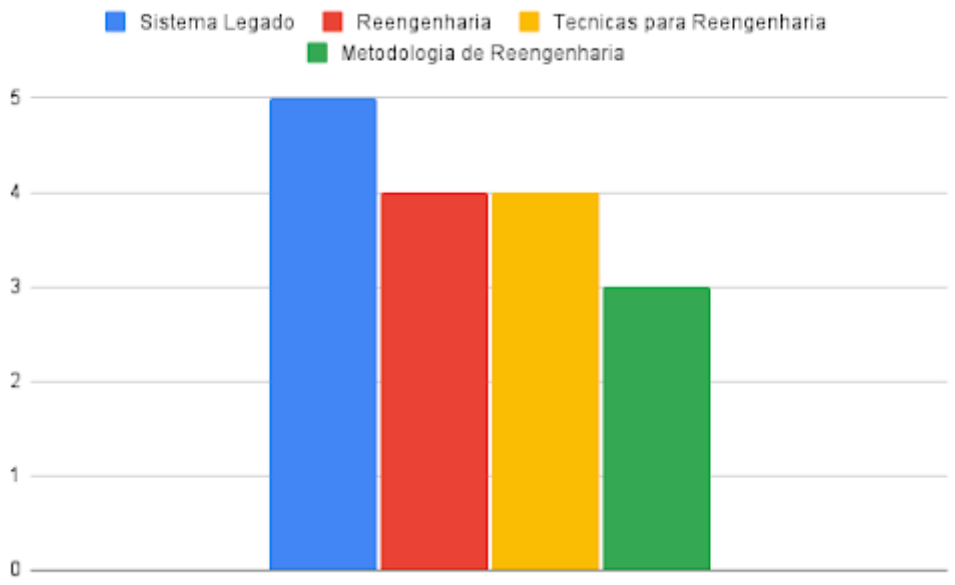
Biblioteca	Título do Artigo	Autoria	Ano de Publicação
ACM	*Experiences in Program Understanding*	Eric Buss e John Henshaw	1992
ACM	*Reverse Engineering Strategies for Software Migration*	Hausi A. Müller	1997
ACM	*Integrating Legacy Systems In Distributed Object Architecture*	Matjaz B. Juric, Ivan Rozman, Marjan Hericko, Tomaz Domajnko	2000
ACM	*Evolving Legacy Systems Features using Regression Test Case and Components*	Alok Mehta e George T. Heineman	2001
ACM	*Reengineering of database intensive application*	Rakesh Agarwal, Ajit Sarangi, Swati Das	2003
ACM	Processos de Planejamento da Reengenharia de Software apoiados por princípios de Usabilidade	Sérgio Luisir Díscola Junior e Júnia Coutinho Anacleto Silva	2003
ACM	*Reengineering Towards Components Using "Reconn-exion"*	Andrew Le Gear e Dr. Jim Buckley	2005
ACM	*A Survey on Survey of Migration of Legacy Systems*	A. Sivagnana Ganesan e T. Chithralekha	2016
ACM	*Understanding Legacy Systems in Light of Grounded Theory*	Alex Severo Chervenski e Andrea Sabedra Bordin	2020
IEEE	*Renaissance: A Method to Migrate from Legacy to Immortal Software Systems*	Marco Battaglia, Giancarlo Savoia e John Favaro	1998
IEEE	*Object-Oriented Reengineering: Patterns and Techniques*	Serge Demeyer e Stephane Ducasse, Oscar Nierstrasz	2005

IEEE	*Parallel Iterative Reengineering Model of Legacy Systems*	Xing Su, Xiaohu Yang, Juefeng Li e Di Wu	2009
IEEE	*Implementation Phases in Modernisation of Legacy Systems*	Humairath KM Abu Bakar, Rozilawati Razali e Dian Indrayani Jambari	2019

Fonte: Elaborado pelo autor.

Os resultados obtidos pelas filtragens do mapeamento sistemático foram criados em 4 divisões. Tais subdivisões envolveram: Sistema Legado; Reengenharia; Técnicas para Reengenharia; Metodologia. Na figura abaixo, tem os números por tema.

Figura 4 – Quantidade de Artigos por Sub Questão (SQ)

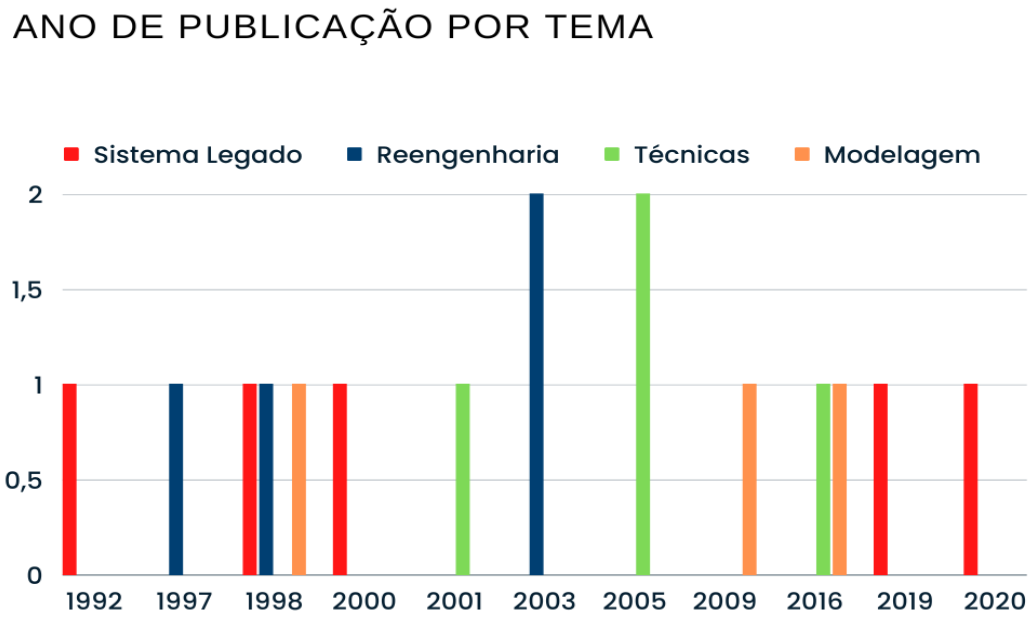


Fonte: Elaborado pelo autor.

Dessa forma, o tema que tivemos maior resultado foi sobre Sistema Legado com cinco artigos encontrados, logo em seguida temos quatro resultados com o tema Reengenharia e Técnicas e por fim com três resultados o tema Modelagem de Reengenharia. Vale ressaltar que existem artigos com mais de uma definição junta, então o total de artigos serão menores que o número de subdivisão.

Também foram analisados o número de artigos por ano, conforme é apresentado na Figura 5, a seguir:

Figura 5 – Artigos selecionados por ano de publicação



Fonte: Elaborado pelo autor.

Como apresentado na Figura 5, o tema Sistema Legado teve sua primeira publicação em 1992 e a última publicação em 2020, demonstrando ser o tema de pesquisa mais antigo e atual. Outra análise a se destacar, são duas pesquisas o artigo "Processos de Planejamento da Reengenharia de Software apoiados por princípios de Usabilidade" dos autores Junior e Silva (2003) e o artigo "*Reengineering of database intensive application*" dos autores Agarwal, Sarangi e Das (2003) publicadas no mesmo ano com tema Reengenharia, duas hipóteses para esse resultado seriam os resultados dos estudos sobre o "bug do milênio" e do crescimento da internet. Já sobre o tema Técnicas, também destaca as duas pesquisas a "*Reengineering Towards Components Using 'Reconnexion'* " de Gear e Buckley (2005) e o artigo "*Object-Oriented Reengineering: Patterns and Techniques"* dos autores Su *et al.(*2005) publicadas no mesmo ano. Em relação à Modelagem, obtivemos três pesquisas, uma a cada ano e tendo aproximadamente dez anos de intervalo de tempo.

Em relação à sub-questão **"Quais definições de Software Legado ou Sistema Legado?"**. Obtivemos resultados com artigos publicados em 1992 até 2020.

Sistemas Legados, possuem várias definições na literatura. Um dos conceitos é "grandes sistemas com os quais não sabemos lidar, mas que são vitais para nossa organização*"* (BATTAGLIA; SAVOIA; FAVARO, 1998, p.1). Anos depois, Juric *et al.* (2000) define Sistema Legado além de grandes sistemas tem que ser analisando a idade do

sistema desde sua construção. Outro ponto que o autor prioriza, são as características desse sistema, entre elas são:

a) Tem códigos existente;

b) É útil atualmente;

c) Está em uso atualmente;

d) A arquitetura não é distribuída e não Orientada a objetos.

Já em 1992, Buss e Henshaw, apresenta a característica de fragilidade do sistema, essa fragilidade tem como o exemplo da complexidade de manutenção e a necessidade de realizar manutenções imediatas. A explicação utilizada pelo autor para mostrar a fragilidade é a manutenção apelidada pelo autor como "cópia e mudança. Ao invés de entender e modificar uma função existente para implementar novos requisitos, uma cópia de uma similar existente é usada para implementar uma nova função modificada*"* (BUSS; HENSHAW, 1992, p. 25).

Em 2019 e 2020, os autores teorizam Sistemas Legados, pela vertente da arquitetura e tecnologia. Bakar, Razali e Jambari (2019) conceitua que "sistemas em execução que não estão em conformidade com padrões arquitetônicos emergentes, mas ainda tem a capacidade de atender algumas necessidades funcionais do negócio*"* (BAKAR; RAZALI; JAMBARI, 2019, p.2) Já Chervenski e Bordin (2020) analisa o Sistema Legado por meio da tecnologia utilizada, a análise consta pela tecnologia se tornar obsoleta, seu tamanho existente, seu tempo de desenvolvimento juntamente com sua operação e pela documentação presente no projeto.

No ano de 1992, Buss e Henshaw, inicia seu artigo com as seguintes informações: **cerca de 30% - 35% do custo total do ciclo de vida são consumidos na tentativa de entender o software que foi entregue para fazer mudanças.** Já 28 anos depois, os autores Chervenski e Bordin (2020) destacam a dificuldade dos desenvolvedores em realizarem manutenção no código legado pela falta de compreensão e habilidade de manutenção. Essa falta de compreensão traz consequências de aumento de custos da manutenção.

Perante à sub-questão **"Quais definições de Reengenharia?"**. Tivemos artigos selecionados nos períodos de 1997 até 2003. Em especial ao ano de 2003 que tivemos dois artigos publicados no mesmo ano.

Para definirmos a Reengenharia, Battaglia, Savoia e Favaro (1998, p.1), conceitua como "transformação sistemática de um sistema existente em uma nova forma para realizar melhorias de qualidade na operação, capacidade do sistema, funcionalidade e desempenho*"* (BATTAGLIA; SAVOIA; FAVARO, 1998, p.1).

Um ano antes, em 1997, Müller (1997) cita que a Reengenharia possui as atividades de re-documentar, reestruturação e transformação do código fonte, além de recuperar a abstração e reimplantação do código. Assim, seis anos depois, Junior e Silva (2003) complementa as atividades detalhando, que deve-se ter um planejamento prévio, esse planejamento deve atender às seguintes questões:

a) Justificar a necessidade de mudança de software;
b) Estudar a viabilidade da Reengenharia como forma de mudança;
c) Determinar suas atividades, custos e prazos.

*"*A Reengenharia busca otimizar o desempenho em relação a outras considerações em nível de subprocessos e tarefas*"* (AGARWAL; SARANGI; DAS, 2003, p. 2). O autor complementa que a Reengenharia vai além do software, existem outros tipos de Reengenharia em volta da Reengenharia de Software.

Para responder a sub-questão **"Quais são às técnicas utilizadas para Reengenharia de Software?"**. Obtemos quatro resultados nas datas de publicações de 2001 até 2016. Nas técnicas encontradas, em 2001, Mehta e Heineman (2001) apresentam uma técnica com base em testes de regressão; em 2005 obtivemos dois estudos publicados apresentando as técnicas *Software Reconnaissance* e *Software Reflexion Modelling* de Gear e Buckley (2005) e Engenharia Reversa por Demeyer, Ducasse e Nierstasz (2005).

A primeira técnica encontrada e conceituada, foi em 2001 por Mehta e Heinemant (2001), apresentando uma técnica com base em testes de regressão, utilizando três premissas. São essas:

a) Os casos de teste são selecionados considerando as características;
b) A implementação dos recursos são identificadas pelo exercício do sistema nos casos de teste selecionados;
c) O código identificado é refatorado em componentes e os resultados são medidos utilizando o funcionamento contínuo do sistema e custo de manutenção.

Além das premissas apresentadas, a utilização de linguagens modernas e análise da regra de negócio que será utilizada também são tratadas como parâmetro da Reengenharia (MEHTA; HEINEMAN, 2001, p.1).

Outras técnicas apresentadas anos mais tarde, são *Software Reconnaissance* e *Software Reflexion Modelling* (GEAR; BUCKLEY, 2005, p. 2). O *Software Reconnaissance* é uma técnica de localização de recursos através de testes direto no código, com foco na

procura de recursos (GEAR; BUCKLEY,2005, p.2). Já a técnica *Software Reflexion Modelling* define o modelo conceitual de alto nível do sistema. Esse modelo tem como base o mapeamento das partes e depois comparando com o código (GEAR; BUCKLEY, 2005, p. 2).

Por fim, temos a técnica mais encontrada em artigos. Essa técnica é a Engenharia Reversa. Demeyer, Ducasse e Nierstrasz (2005) conceituam engenharia reversa como o *"*processo de análise de um sistema a fim de expor sua estrutura e o projeto a um nível superior de abstração*"* (DEMEYER; DUCASSE; NIERSTRASZ, 2005, p.2). Onze anos mais tarde, em 2016, Ganesan e Chithralekha (2016) complementa o conceito de engenharia reversa, dizendo que é na engenharia reversa que há o entendimento, compreensão e recuperação da arquitetura do Sistema Legado.

Para responder a sub-questão **"Quais pesquisas realizadas de metodologias na Reengenharia de Software e software/Sistema Legado que contribuíram para inovação?".** Foram encontrados 3 artigos que ajudam nas pesquisas de metodologias, esses artigos variam dos anos 1998 até 2016.

A primeira pesquisa encontrada foi da metodologia *Renaissace*. É uma metodologia que se inicia na avaliação do software atual, sendo projetada para controlar custos. Assim, por meio dessa metodologia, o valor comercial e qualidade técnica definem quais componentes devem ser evoluídos (BATTAGLIA; SAVOIA; FAVARO, 1998, p.2). A metodologia *Renaissace*, tem seu foco na análise de custo/benefício e custo/risco, se tornando uma metodologia é bem parecida com "*Chicken Little*".

Em 2009, Su *et al.* (2009) propôs uma metodologia de Reengenharia iterativa paralelo. De acordo com os teóricos, a Reengenharia iterativa "paralelo permite que vários componentes e dados sejam operados em um ciclo durante o processo de Reengenharia*"* (SU *et al.*, 2009, p.2).

Ganesan e Chithralekha (2016) apresenta em seu trabalho vantagens e desvantagens das metodologias tradicionais. Em especial, no Quadro 5 será apresentado a vantagem e desvantagem "*Little Chicken*".

Quadro 5 – Vantagens e Desvantagens *Little Chicken*

Abordagem da Migração	Vantagens	Desvantagens
• Difere a funcionalidade e coloca *gateways* a partir de abordagem de banco de dados composto. • Utiliza *gateways* de avanço e retrocesso para proporcionar a interoperabilidade necessária. • Os Sistemas Legados e Alvo operam em paralelo durante todo o processo de migração (Reengenharia). • O sistema de informação operacional será um composto de sistema de informação destino e legado, utilizando gateways para fornecer a interoperabilidade necessária.	• O sistema de informação operacional será um composto de Sistema Legado e sistema destino • Adequado para migração de sistemas totalmente decomponíveis, semi descomponiceis e não decomponíveis	• Emprega portões complexos • A necessidade de interoperação dos Sistemas Legados e sistemas-alvo através de gateways acrescenta maior complexidade ao processo complexo já existente.

Fonte: Ganesan e Chithralekha, 2016. Adaptado

O Quadro 5 apresenta o método *Little Chicken*, mostrando como é a sua abordagem, demonstrado as suas vantagens e desvantagens. O destaque é a sua abordagem, que utiliza a criação de um novo modulo chamado de sistemas Alvo, recebendo os dados do Sistema Legados via "*gateways*", mostrando também sua desvantagem, que são dois sistemas funcionando paralelamente, aumentando sua complexidade.

Diante dos resultados encontrados, o Quadro 6 que sintetiza as principais definições encontradas por tema.

Quadro 6 – Principais Definições por Tema

Tema	Ano	Autor	Definição
Sistemas Legados	1992	Buss e Henshaw	“O sistema legado tem características de atender a manutenção e mudanças imediatas. Essas mudanças imediatas ajudam na integridade do sistema”
	1998	Battaglia, Savoia e Favaro	“grandes sistemas com os quais não sabemos lidar, mas que são vitais para nossa organização.”
	2000	Juric et al	Sistemas Legados possuem seguintes fundamentos: • Tem código existente; • É útil atualmente; • Está em uso atualmente; • A arquitetura não é distribuída e orientada a objetos
	2019	Bakar, Razali e Jambari	“sistemas em execução que não estão em conformidade com padrões arquitetônicos emergentes, mas ainda tem a capacidade de atender algumas necessidades funcionais do negócio”
	2020	Chervenski e Bordin	"Um sistema pode ser considerado legado através da tecnologia obsoleta utilizada para o seu desenvolvimento, pelo tamanho do sistema, pelo tempo em que foi desenvolvido e continua em operação, pelo status da documentação e pela importância que representa para a organização."
Reengenharia	1997	Muller	“O espectro das atividades de reengenharia inclui redocumentação, reestruturação do código fonte, transformação do código fonte, recuperação da abstração, e reimplementação.”
	1998	Battaglia, Savoia e Favaro	“transformação sistemática de um sistema existente em uma nova forma para realizar melhorias de qualidade na operação, capacidade do sistema, funcionalidade e desempenho”
	2003	Agarwal, Sarangi e Das	“A reengenharia busca otimizar o desempenho em relação a outras considerações em nível de subprocessos e tarefas”

Técnicas	2001	Mehta e Heineman	É apresentado a técnica de testes de regrassão que utilizam os seguntes premissas: • os casos de teste são selecionados considerando as características; • A implementação dos recursos (Funções) é identificada pelo exercício do sistema nos casos de teste selecionados; • O código identificado é refatorado em componentes e os resultados são medidos utilizando o funcionamento contínuo do sistema e o custo de manutenção”
	2005	Demeyer, Ducasse e Nierstrasz	Apresentada primeira definição de Engenharia Reversa: “processo de análise de um sistema a fim de expor sua estrutura e o projeto a um nível superior de abstração”
	2016	Ganesan e Chithralekha	"Engenharia reversa, compreensão do programa e recuperação da arquitetura"
Metodologias	1998	Battaglia, Savoia e Favaro	"Renaissance fornece um conjunto abrangente de técnicas para análise de custo/benefício dos componentes em avaliação, bem como a análise de custo/risco, onde os fatores de custo e risco de várias alternativas são examinado a fim de escolher a melhor combinação."
	2009	Su *et al.*	“Reengenharia interativa paralelo permite que vários componentes e dados sejam operados em um ciclo durante o processo de reengenharia”

Fonte: Elaborado pelo autor.

Um destaque do Quadro 6 é a evolução da definição do *“Sistemas Legados”*, que possui suas primeiras definições relacionados à idade, importância para a empresa e dificuldades de manutenção; e já nas últimas publicações os téoricos adicionaram a característica da arquitetura e sua tecnologia. Mais um destaque do quadro, são as definições de *“Reengenharia”* que fizeram analogia de Reengenharia com reconstrução, mas também frizando a importância da Reengenharia na documentação, nas regras de negócios e não priorizando somente o código-fonte. Outros pontos de destaque são as “*Técnicas*” e “*Metodologias*”, sendo o destaque das “Técnicas” a Engenharia Reversa que foi citada em

todos os artigos; enquanto nas “Metodologias” observa-se a priorização por metodologias que utilizam a estratégia incremental, como o caso da *Renaissance* e da Reengenharia interativa paralelo.

5 CONSIDERAÇÕES FINAIS

A partir da presente pesquisa foi possível evidenciar a importância da Reengenharia de Software em Sistemas Legados. Os Sistemas Legados possuem grande importância para as organizações, tanto comerciais quanto ao conjunto de dados obtidos. Mas sua manutenção é custosa e requer diversos cuidados. Uma potencial solução para reduzir os custos e melhorar a forma de manutenção dos Sistemas Legados, é a Reengenharia. A Reengenharia, não precisa necessariamente no sistema todo de uma vez, mas poderá ser realizada aos poucos. Sendo possível transformar o Sistema Legado em um "sistema atual", custando bem menos e com baixo risco de perda de regras de negócios e erros bem menores de execução.

Por meio do mapeamento sistemático foi possível realizar um levantamento acerca do estado da arte relacionado aos conceitos de Sistemas Legados e Reengenharia de Software a fim de identificar as melhores práticas da Reengenharia de Software para a atualização de um Sistema Legado.

Com isso, a definição que consideramos mais plausível para a Sistema Legados, é a definição de Battaglia, Savoia e Favaro (1998) que são "grandes sistemas com os quais não sabemos lidar, mas que são vitais para nossa organização*"* (BATTAGLIA; SAVOIA; FAVARO, 1998, p.1). Apresentando que em 1998, esses sistemas já possuiam uma forte importância e preocupação relacionado a manutenção.

Já em relação à Reengenharia, é importante salientar a apresentação de outros tipos de Reengenharia existentes atreladas à Reengenharia de Software, tais como: a Reengenharia de processos e Reengenharia de dados que pode ser feita antes, durante ou depois da Reengenharia de Software.

Assim, a definição que consideramos mais coerente, é adaptação do conceito de Agawal, Sarangi e Das (2003, p. 2) e Battaglia, Savoia e Favarol (1998, p.1). Com isso, temos o conceito de "Reengenharia como buscar otimizar o desempenho em relação a outras considerações em nível de subprocessos e tarefas" (AGAWAL; SARANGI; DAS, 2003, p.2). [...] "uma nova forma para realizar melhorias de qualidade na operação, capacidade do sistema, funcionalidade e desempenho*"* (BATTAGLIA; SAVOIA; FAVAROL, 1998, p.1).

Em relação às metodologia e técnicas, foram apresentadas a *Little Chicken*, Reengenharia Iterativa Paralelo e metodologia *Renaissace*. A metodologia e as técnicas são

utilizadas em conjunto, de forma que a técnica mais popular para adquirir melhor abstração é a Engenharia Reversa.

A metodologia mais tradicional é o *Little Chicken*, sendo a metodologia mais estudada pelo Ganessan e Chithralekha (2016). No entanto, a metodologia que teria maior destaque para organizações que trabalham com Sistemas Legados, seria o *Renaissace*, pelo fato de ter foco em custo/benefício e custo/risco, sendo alterado de acordo com o valor comercial. Um fato interessante da metodologia Renaissace é a semelhança com o *Little Chicken.*

Diante dos resultados obtidos foi possível observar a presença de pesquisas envolvendo Sistemas Legados na literatura a partir de 1992. Os estudos relacionados aos Sistemas Legados apresentaram diversas definições no decorrer do tempo. A princípio, o conceito de Sistemas Legados estava associado à idade do Sistema, agora alguns teóricos começam a analisar sua arquitetura e estrutura.

A mesma observação temos para Reengenharia, que se ramificou, existindo a Reengenharia de Processos, Reengenharia de Dados, Reengenharia de Software, entre outras. Tal fato evidencia que é possível realizar mais de uma Reengenharia em conjunto ou uma Reengenharia específica em processos de reestruturação do projeto.

Relacionado às melhores práticas, foi possível observar que as técnicas e as metodologias são efetuadas em conjunto. A única prática presente em todas as metodologias é a Engenharia Reversa, pois é considerada a melhor forma de obter as abstrações de nivel mais baixo para o nivel mais alto. Ela é utilizada em conjunto com todas as técnicas apresentadas. Vale salientar, que as técnicas e práticas podem se estender, podendo ser utilizadas separadamente ou em conjunto.

Quanto as metodologias, temos as tradicionais como *Little Chicken* que apresenta uma abordagem incremental e pode ser adaptada para a necessidade do projeto, um exemplo é o método Renaissace.

Para trabalhos futuros, é proposto a realização do mapeamento sistemático em outras bibliotecas e com mais tempo a fim de ampliar e aprofundar os resultados. Sugerimos também a realização de uma pesquisa prática empregando as técnicas e metodologias levantadas em um Sistema Legado específico.

REFERÊNCIAS

AGARWAL, R; SARANGI, A; DAS, S. Reengineering of database intensive application. **SIGSOFT Softw. Eng. Notes 28, 3 (May 2003)**. New York, NY, USA, v.28, n.3, p.1. 2003. Disponível em: https://doi.org/10.1145/773126.773136. Acesso em: 22 nov. 2022

AKAMINE, G. S. **Processo de avaliação de Sistemas Legados**. 2015. Trabalho de Conclusão de Curso (Bacharel em Engenharia de Software) – Universidade Federal do Pampa. Disponível em: https://repositorio.unipampa.edu.br/jspui/handle/riu/861. Acesso em: 16 jul. 2022

ALMEIDA, R. F. **Aplicação Prática de Técnica de Reengenharia de Software**. 2009. Trabalho de Conclusão de Curso (Bacharel em Ciência da Computação) – Departamento da Computação do Instituto de Ciências Exatas, Universidade Federal de Juiz de Fora.

ANDRADE, C. R. T. **Intergenicbd 2.0**. 2016. Trabalho de Conclusão de Curso (Bacharel em Sistemas de Informação) – Universidade de Caxias do Sul. Disponível em: https://repositorio.ucs.br/handle/11338/1553. Acesso em: 10 set. 2022

ARAGÃO, J. Introdução aos estudos quantitativos utilizados em pesquisas científicas. **Revista Práxos**, Volta Redonda, ano III, v. 3, n. 6, 2011. Disponível em: http://revistas.unifoa.edu.br/index.php/praxis/article/view/566. Acesso em: 10 jan. 2022

BAKAR, H.K.A; RAZALI, R; JAMBARI, D. I. Implementation Phases in Modernisation of Legacy Systems. **2019 6th International Conference on Research and Innovation in Information Systems (ICRIIS)**. Johor Bahru, Malaysia, p. 1-6. 2019. Disponível em: https://ieeexplore.ieee.org/document/9073628. Acesso em: 28 nov. 2022

BATTAGLIA, M; SAVOIA, G; FAVARO, J. Renaissance: A Method to Migrate from Legacy to Immortal Software Systems. **Proceedings of the Second Euromicro Conference on Software Maintenance and Reengineering**. Florence, Italy, p. 197-200. 1998. Disponível em: https://ieeexplore.ieee.org/document/665807. Acesso em: 28 nov. 2022

BISBAL, J. et al**. Legacy information systems: issues and directions**. IEEE Software, v. 16, 1999 *apud* CHERVENSKI, A. S. **Entendimento sobre Sistemas Legados à luz da Teoria Fundamentada em Dados**. 2019. Trabalho de Conclusão de Curso (Bacharelado em Engenharia de Software) - Universidade Federal do Pampa. Disponível em: https://repositorio.unipampa.edu.br/jspui/handle/riu/4835 . Acesso em: 15 jul. 2022

BORBA, P. E S. **Sistemas Legados:** Uma Proposta para Centralização de Autenticação. 2017. Trabalho de Conclusão de Curso (Graduação em Engenharia de Software) – Universidade de Brasília – UnB Faculdade UnB Gama - FGA. Disponível em: https://bdm.unb.br/handle/10483/19777 . Acesso em: 16 jul. 2022

BUSS, E.; HENSHAW, J**. Experiences in Program Understanding. In Proceedings of the 1992 conference of the Centre for Advanced Studies on Collaborative research - Volume 1 (CASCON '92).** IBM Press, p.157–189. 1992. Disponível em: https://dl.acm.org/doi/10.5555/962198.962211. Acesso em: 16 nov. 2022

CANTO, F. H. **Proposta de Método de Reengenharia de Sistemas Legados Desenvolvidos em PHP**. 2021. Trabalho de Conclusão de Curso (Especialisata em Engenhari de Software e Inovação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul. Disponível em: http://hdl.handle.net/10183/231566. Acesso em: 10 set. 2022

CHAVES, L. L. Sistemas Legados e a Aplicação de Processos de Reengenharia de Software. **1° International Conference on Information Systems and Technology Management – CONTECSI.** FEA/USP, São Paulo, Brasil. 2004

CHERVENSKI, A. S. **Entendimento sobre Sistemas Legados à luz da Teoria Fundamentada em Dados**. 2019. Trabalho de Conclusão de Curso (Bacharelado em Engenharia de Software) - Universidade Federal do Pampa. Disponível em: https://repositorio.unipampa.edu.br/jspui/handle/riu/4835 . Acesso em: 15 jul. 2022

CHERVENSKI, A. S; BORDIN, A. S. Understanding Legacy Systems in Light of Grounded Theory. **In Proceedings of the XXXIV Brazilian Symposium on Software Engineering (SBES '20).** Association for Computing Machinery, New York, NY, USA, p.344–353. 2020. Disponível em: https://doi.org/10.1145/3422392.3422400. Acesso em: 16 nov. 2022

DAMIAN, A. L. **Diretivas de Comunicabilidade para Artefatos de Software**. 2020. Dissertação – Instituto de Computação, Universidade Federal do Amazonas. Disponível em: https://tede.ufam.edu.br/handle/tede/7974. Acesso em: 31 out. 2022

DEMEYER, S; DUCASSE, S; NIERSTRASZ, O. Object-Oriented Reengineering: Patterns and Techniques. **21st IEEE International Conference on Software Maintenance (ICSM'05)**. Budapest, Hungary, p. 723-724. 2005. Disponível em: https://ieeexplore.ieee.org/document/1510183. Acesso em: 30 nov. 2022

GANESAN, A. S; CHITHRALEKHA, T. A Survey on Survey of Migration of Legacy Systems. **In Proceedings of the International Conference on Informatics and Analytics (ICIA-16)**. Association for Computing Machinery, New York, NY, USA, n.72, p.1–10. 2016. Disponível em: https://doi.org/10.1145/2980258.2980409. Acesso em: 21 nov. 2022

GARCIA, V. C. **Phoenix:** Uma Abordagem para Reengenharia de Software Orientada a Aspectos. 2005. Dissertação (Mestrado em Engenharia de Software) – Centro de Ciências Exatas e de Tecnologia, Universidade Federal de São Carlos. Disponível em: https://repositorio.ufscar.br/handle/ufscar/621. Acesso em: 19 abr. 2022

GEAR, A. L; BUCKLEY, J. Reengineering Towards Components Using “Reconn-exion”. **In Proceedings of the 10th European software engineering conference held jointly with 13th ACM SIGSOFT international symposium on Foundations of software engineering (ESEC/FSE-13).** Association for Computing Machinery, New York, NY, USA, p.370–373. 2005. Disponível em: https://doi.org/10.1145/1081706.1081765. Acesso em: 22 nov. 2022

IGAWA, R. A**. Uma proposta de Reengenharia com abordagem incremental para sistemas orientado a objetos**. 2013. Trabalho de Conclusão de Curso (Bacharel em Ciência da Computação) – Universidade Estadual de Londrina. Disponível em: http://www.uel.br/cce/dc/wp-content/uploads/TCC-RodrigoIgawa-BCC-UEL-2013.pdf. Acesso em: 04 out. 2022

JUNIOR, S. L. D; SILVA, J. C. A. Processos de Planejamento da Reengenharia de Software apoiados por princípios de Usabilidade. **In Proceedings of the Latin American conference on Human-computer interaction (CLIHC '03)**. Association for Computing Machinery, New York, NY, USA, p.223–226. 2003. Disponível em: https://doi.org/10.1145/944519.944544. Acesso em: 22 nov. 2022

JURIC, M. B; ROZMAN, I; HERICKO, M; DOMAJNKO, T. Integrating legacy systems in distributed object architecture. **SIGSOFT Softw. Eng. Notes 25, 2 (March 2000)**. New York, NY, USA, p.35-39. 2000. Disponível em: https://doi.org/10.1145/346057.346069. Acesso em: 21 nov. 2022

KITCHENHAM, B. CHARTERS, S. Guidelines for Performing Systematic Literature Reviews in Software Engineering. EBSE Technical Report (EBSE-2007-01). Keele University and University of Durham. 2007 *apud* DAMIAN, A. L. Diretivas de Comunicabilidade para Artefatos de Software. 2020. Dissertação – Instituto de Computação, Universidade Federal do Amazonas. Disponível em: https://tede.ufam.edu.br/handle/tede/7974. Acesso em: 31 out. 2022

MEHTA, A; HEINEMAN, G.T. Evolving Legacy Systems Features using Regression Test Case and Components**. In Proceedings of the 4th International Workshop on Principles of Software Evolution (IWPSE '01).** Association for Computing Machinery, New York, NY, USA, p.190–193. 2001. Disponível em: https://doi.org/10.1145/602461.602507. Acesso em: 21 nov. 2022

MÜLLER, H. A. Reverse Engineering Strategies for Software Migration. **In Proceedings of the 19th international conference on Software engineering (ICSE '97)**. Association for Computing Machinery, New York, NY, USA, p.659-660. 1997. Disponível em: https://doi.org/10.1145/253228.253799. Acesso em: 21 nov 2022

NILSSON, S. **Application modernization: Approaches, problems and evaluation.** Dissertation, 2015 *apud* CHERVENSKI, A. S. **Entendimento sobre Sistemas Legados à luz da Teoria Fundamentada em Dados**. 2019. Trabalho de Conclusão de Curso (Bacharelado em Engenharia de Software) - Universidade Federal do Pampa. Disponível em: https://repositorio.unipampa.edu.br/jspui/handle/riu/4835 . Acesso em: 15 jul. 2022

OLIVEIRA, M. C. **Reengenharia de Software:** Reengenharia de um Sistema Legado baseado em padrões. 2003. Trabalho de Conclusão de Curso (Licenciado em Ciência da Computação) – Universidade Presidente Antônio Carlos. Disponivel em: https://ri.unipac.br/repositorio/wp-content/uploads/2019/08/M%C3%8DRIAN-CRISTINA-DE-OLIVEIRA.pdf. Acesso em: 27 out. 2022

PIEKARSKI, A. E. T; QUINÁIA, M. A. **Reengenharia de Software:** o que, por quê e como. 2000. Departamento de Informática – UNICENTRO.

PINTO, H. L. M; BRAGA, J. L. **Sistemas Legados e as Novas Tecnologias:** técnicas de integração e estudo de caso. 2004. Informática Pública vol. 7(1): 47-69, 2005. Disponível em: http://pbh.gov.br/informaticapublica/ANO7_N1_PDF/IP7N1_mendespinto.pdf. Acesso em: 01 out. 2022

SCARPA, D. L; MARANDINO, M. Pesquisa em ensino de ciência: um estudo sobre as perspectivas metodológias. In: Atas do II Encontro Nacional de Pesquisa em Educação em Ciências (ENPEC). Valinhos-SP: ABRAPEC, 1999. Disponível em: http://www.fep.if.usp.br/~profis/arquivos/iienpec/Dados/trabalhos/A07.pdf . Acesso em: 10 jan. 2022

SU, X; YANG, X; LI, J; WU, D. Parallel Iterative Reengineering Model of Legacy Systems. **2009 IEEE International Conference on Systems, Man and Cybernetics**. San Antonio, TX, USA, p. 4054-4058. 2009. Disponível em: https://ieeexplore.ieee.org/document/5346680. Acesso em: 30 nov. 2022

UFVJM