UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CÂMPUS CORNÉLIO PROCÓPIO
DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

JOAO RICARDO MORENO CAMILO

UMA ABORDAGEM CONCEITUAL PARA O GERENCIAMENTO DO PROCESSO DE EVOLUÇÃO DISTRIBUÍDA DE SOFTWARE EM CLIENTES DE SISTEMAS PROPRIETÁRIOS

DISSERTAÇÃO

CORNÉLIO PROCÓPIO

2020

JOAO RICARDO MORENO CAMILO

UMA ABORDAGEM CONCEITUAL PARA O GERENCIAMENTO DO PROCESSO DE EVOLUÇÃO DISTRIBUÍDA DE SOFTWARE EM CLIENTES DE SISTEMAS PROPRIETÁRIOS

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Universidade Tecnológica Federal do Paraná – UTFPR como requisito parcial para a obtenção do título de "Mestre Profissional em Informática".

Orientador: Alexandre L'Erario

CORNÉLIO PROCÓPIO

2020

Dados Internacionais de Catalogação na Publicação

C183 Camilo, João Ricardo Moreno

Uma abordagem conceitual para o gerenciamento do processo de evolução distribuída de software em clientes de sistemas proprietários / João Ricardo Moreno Camilo. - 2020.
77 f. : il. color. ; 31 cm.

Orientador: Alexandre L'Erario.
Dissertação (Mestrado) – Universidade Tecnológica Federal do Paraná. Programa de Pós-Graduação em Informática, Cornélio Procópio, 2020.
Bibliografia: p. 68-73.

1. Software - Desenvolvimento. 2. Software - Manutenção. 3. Engenharia de software. 4. Informática – Dissertações. I. L'Erario, Alexandre, orient. II. Universidade Tecnológica Federal do Paraná. Programa de Pós-Graduação em Informática. III. Título.

CDD (22. ed.) 004

Biblioteca da UTFPR - Câmpus Cornélio Procópio
Bibliotecário/Documentalista responsável:
Romeu Righetti de Araujo – CRB-9/1676

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Cornélio Procópio
Programa de Pós-Graduação em Informática

Título da Dissertação Nº 69:

"UMA ABORDAGEM CONCEITUAL PARA O GERENCIAMENTO DO PROCESSO DE EVOLUÇÃO DISTRIBUÍDA DE SOFTWARE EM CLIENTES DE SISTEMAS PROPRIETÁRIOS".

por

João Ricardo Moreno Camilo

Orientador: **Prof. Dr. Alexandre L'Erario**

Esta dissertação foi apresentada como requisito parcial à obtenção do grau de MESTRE EM INFORMÁTICA – Área de Concentração: Computação Aplicada, pelo Programa de Pós-Graduação em Informática – PPGI – da Universidade Tecnológica Federal do Paraná – UTFPR – Câmpus Cornélio Procópio, às 09h00 do dia 18 de dezembro de 2019. O trabalho foi _____ pela Banca Examinadora, composta pelos professores:

_____
Prof. Dr. Alexandre L'Erario
(Presidente – UTFPR-CP)

_____
Prof. Dr. Cléber Gimenez Corrêa
(UTFPR-CP)

_____
Prof. Dr. Wagner Fontes Godoy
(UTFPR-CP)

_____
Prof. Dr. Rodolfo Miranda de Barros
(UEL)
Participação à distância via _____

Visto da coordenação:

_____
Danilo Sipoli Sanches
Coordenador do Programa de Pós-Graduação em Informática
UTFPR Câmpus Cornélio Procópio

A Folha de Aprovação assinada encontra-se na Coordenação do Programa.

Dedico este trabalho à minha família e amigos.

AGRADECIMENTOS

Agradeço primeiramente à Deus, por me convencer todos os dias desta escolha.

Ao meu orientador Prof. Dr. Alexandre L'Erario, pela sabedoria com que me guiou nesta trajetória, e por todas as motivações que nortearam o rumo desta pesquisa.

Gostaria de deixar registrado também, o meu reconhecimento à minha família e amigos, em especial, minha mãe Eliane, meu pai Eduardo, meu irmão José Eduardo e, principalmente, minha esposa Camila, pois acredito que sem o apoio deles seria impossível concluir esse desafio.

Ao meu pai Eduardo (in memorian), que não está mais entre nós, mas continua sendo minha maior força na vida. Sua lembrança me inspira e me faz persistir.

Aos meus professores.

A Secretaria do PPGI, pela cooperação.

Enfim, a todos os que por algum motivo contribuíram para a realização desta pesquisa.

RESUMO

CAMILO, João Ricardo Moreno. UMA ABORDAGEM CONCEITUAL PARA O GERENCIAMENTO DO PROCESSO DE EVOLUÇÃO DISTRIBUÍDA DE SOFTWARE EM CLIENTES DE SISTEMAS PROPRIETÁRIOS. 77 f. DISSERTAÇÃO – Programa de Pós-graduação em Informática, Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2020.

Contexto: Sistemas corporativos *Enterprise Resource Planning* (ERP) proprietários são softwares comumente incorporados em organizações multinacionais e, portanto, possuem múltiplas fontes de demanda local e global. Este tipo de software está sujeito a constantes evoluções que impactam em seu funcionamento, uma vez que suas mais diversas funcionalidades têm grande correlação e dependência entre elas – produção, finanças, contabilidade, vendas, compras, entre outras. As constantes mudanças são motivadas por adição de novas funcionalidades e melhorias nos recursos, principalmente, por mudanças contínuas e imediatas no contexto legal e econômico do ambiente em que a organização cliente do software corporativa está inserido. A evolução do software pode ser implementada de forma distribuída por diferentes partes interessadas, como o produtor de software, parceiros externos, ou até mesmo pelas próprias organizações clientes dos sistemas corporativos, e assim por diante. A organização cliente desse tipo de software é uma das partes interessadas mais impactadas no processo de evolução, afinal precisa manter constante operação com clientes, governo e o mercado, em geral. Portanto, o gerenciamento do processo de evolução de software em configurações distribuídas globalmente traz desafios significativos para organizações que são clientes deste tipo software. **Objetivo**: Identificar e descrever fatores específicos do processo de evolução de software em configurações distribuídas globalmente sob uma estrutura conceitual. **Método**: Para atingir esse objetivo foi utilizado o método de teoria fundamentada. Como fonte de coleta de dados foram realizadas entrevistas com profissionais da indústria em organizações clientes de um sistema corporativo ERP proprietário. **Resultados**: Foi desenvolvida uma estrutura conceitual para identificar e descrever os desafios e soluções da evolução distribuída de software pela perspectiva de organizações clientes de softwares corporativos proprietários. Como resultado da estrutura conceitual, foram sugeridas melhorias no fluxo do processo de evolução distribuída de software (CAMILO et al., 2018). **Conclusão**: O processo de evolução distribuída de software tem desafios específicos de gerenciamento em relação a processos, pessoas, produtos e tecnologia. Portanto, as empresas que executam a evolução de software em configurações distribuídas devem considerar esses fatores, que parecem estar pouco presentes na literatura de desenvolvimento de software global, embora muitas lições não específicas se apliquem a ambos.

Palavras-chave: Evolução de Software, Desenvolvimento Distribuído de Software, Evolução Distribuída de Software

ABSTRACT

CAMILO, João Ricardo Moreno. AN ANALYSIS OF THE DISTRIBUTED SOFTWARE EVOLUTION PROCESS BY THE PERSPECTIVE OF CUSTOMER ORGANIZATIONS OF PROPRIETARY CORPORATE SYSTEMS. 77 f. DISSERTAÇÃO – Programa de Pós-graduação em Informática, Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2020.

Context: Proprietary Enterprise Resource Planning (ERP) systems are software commonly embedded in multinational organizations and therefore have multiple sources of local and global demand. This type of software is subject to constant evolution that impact its operation. Many functionalities have a great correlation and dependence between them - production, finance, accounting, sales, purchasing, among others. Constant changes are driven by the addition of new features and feature enhancements, primarily by continuous and immediate changes in the legal and economic context of the environment in which the enterprise software customer organization is embedded. Software evolution can be implemented in a distributed way across different stakeholders, such as the software producer, external partners, or even the enterprise systems client organizations themselves, and so on. The customer organization of this type of software is one of the most impacted stakeholders in the evolution process, after all it needs to maintain constant operation with its customers, government and the market in general. Therefore, the process of software evolution in globally distributed configurations poses significant challenges for organizations that are customers of this type of software. **Objective**: To identify specific factors of the software evolution process in globally distributed configurations under a conceptual framework. **Method**: To achieve this objective the grounded theory method was used. As a source of data collection interviews were conducted with industry professionals in client organizations of a proprietary corporate ERP system. **Results**: A conceptual framework has been developed to identify and describe the challenges and solutions of distributed software evolution from the perspective of proprietary enterprise software client organizations. As a result of the conceptual framework, improvements in the flow of the distributed software evolution process have been suggested (CAMILO et al., 2018). **Conclusion**: The distributed software evolution process has specific management challenges regarding processes, people, products, and technology. Therefore, companies that perform software evolution in distributed configurations should consider these factors, which seem to be scarcely present in the global global software development literature, although many non-specific lessons apply to both.

Keywords: Software Evolution, Distributed Software Development, Distributed Software Evolution

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

ERP	Enterprise Resource Planning

SUMÁRIO

1 INTRODUÇÃO

A engenharia de software global tornou-se uma prática comum na área de negócios da tecnologia da informação. Mais de 50 países estão ativamente envolvidos na colaboração de atividades no ciclo de vida do software (GONZALEZ et al., 2006), local ou globalmente. Entre diversas motivações por trás dessa tendência está a econômica, como, por exemplo, a pressão sobre o mercado – por clientes, governos, fornecedores, órgãos regulamentadores, e assim por diante (SENGUPTA et al., 2006).

No mercado global, sistemas corporativos *Enterprise Resource Plannings* (ERP) proprietários são softwares vendidos com frequência. Esse tipo de sistema permite a integração completa do fluxo de informações de todas as áreas funcionais – financeira, produção, contabilidade, vendas, logística, entre outras –, chamadas de módulos, em organizações corporativas por meio de um banco de dados, acessível por uma interface unificada (ADDO-TENKORANG; HELO, 2011). Um sistema ERP é basicamente um software modularizado ou empacotado, com capacidade de personalização ou adição de funcionalidades, módulos e até pacotes por parte de terceiros, não sendo desenvolvido e mantido como outros tipos de projetos de software tradicionais (NG; GABLE, 2010; LOPEZ; SALMERON, 2011).

Um sistema corporativo proprietário é um produto de software desenvolvido por uma empresa privada, e adquirido por organizações clientes no mundo todo. É importante ressaltar que esse tipo de sistema possui interfaces específicas e limitadas de acesso e implementação, e nem sempre o fornecedor disponibiliza todos os recursos de código fonte para que terceiros possam modificar (GODFREY; GERMAN, 2008).

O contexto em que este trabalho está inserido apresenta uma escassez de estudos e pesquisas sobre esse tipo de sistema e, consequentemente, uma lacuna para identificar desafios e propor soluções. Soluções estas que possam existir para outras áreas, ou possam ser propostas por estudos empíricos realizados com profissionais experientes na área, por exemplo.

Como as organizações clientes desses produtos podem ser de qualquer lugar do

mundo, as produtoras de softwares geralmente movem ou distribuem o suporte, manutenção e evolução de seus produtos para os locais mais próximos dos clientes, ou para mercados maiores; ou também disponibilizam parcialmente acesso ao código fonte para alteração por diferentes partes interessadas – organizações clientes, parceiros externos, e assim por diante (NG et al., 2001; YAN, 2004). Em configurações distribuídas, essas operações podem ser realizadas pelas partes interessadas de diferentes formas, local ou globalmente (TIWANA, 2004; CAMILO et al., 2018), como por exemplo:

- As produtoras de softwares controlam e executam suporte, manutenção e evolução por meio de sua própria equipe, por acesso remoto.
- Parceiros externos controlam e executam suporte, manutenção e evolução por meio de sua própria equipe, por acesso remoto.
- A própria organização cliente do software controla e executa suporte, manutenção e evolução por meio de sua própria equipe, por acesso remoto.
- Diferentes partes interessadas acima, em conjunto, controlam e executam suporte, manutenção e evolução por meio de sua próprias equipes, por acesso remoto.

Os termos evolução de software e manutenção de software são frequentemente usados como sinônimos. Existem diferentes definições para evolução de software e manutenção de software. Apesar de compartilharem conceitos semelhantes, alguns autores se referem à evolução quando se adiciona uma nova funcionalidade ou algum outro aspecto do software é melhorado, o que inclui correções maiores, e referem-se à manutenção principalmente a adaptações e correções menores para manter o software em execução (MENS et al., 2010; RAJLICH, 2014). Uma vez que os objetivos da manutenção visam os objetivos reduzidos, os processos geralmente são mais simples e previsíveis do que os processos de evolução (RAJLICH, 2014). Este trabalho adota evolução e manutenção de software por meio desses conceitos.

Muitas organizações não conseguem obter os benefícios esperados da engenharia de software global e a qualidade desejada de serviço e produto (NAKATSU; IACOVOU, 2009). Estudos empíricos confirmaram que a engenharia de software global é uma tarefa complexa e revelou que grandes dificuldades comuns são causadas por: diferenças organizacionais, de fuso horário, culturais, barreiras de coordenação e comunicação e perda de confiança entre as partes interessadas (HERBSLEB; MOITRA, 2001; HERBSLEB; MOCKUS, 2003; DAMIAN; MOITRA, 2006; HUSSEY; HALL, 2007).

Atividades de manutenção e evolução de software normalmente exigem contato extensivo com as partes interessadas, o que pode resultar em atrasos de comunicação, interpretações errôneas dos requisitos e responsabilidades indiretas. Esses desafios são muitas vezes predominantes e ampliados na colaboração em equipes distribuídas (TIWANA, 2004; SEYBOLD; KELLER, 2008). A coordenação deste cenário é considerada crítica para as organizações clientes que utilizam esse tipo de software. Se o sistema corporativo não evoluir, os processos da empresa podem ser temporariamente comprometidos, resultando em severos prejuízos, tanto financeiros como de negócios. Dessa forma, o software se tornará obsoleto (CAMILO et al., 2018).

Para Audy e Prikladnicki (2007), projetos desenvolvidos de maneira distribuída fazem ampliar os problemas inerentes ao desenvolvimento tradicional, gerando dessa forma novos desafios ao adicionar distância física, dispersão temporal e diferenças culturais. Devido à descentralização das empresas e ao aumento da produção de software distribuído, os processos de evolução de software tornam-se mais complexos, exigindo que as organizações busquem soluções que consigam atender as características e necessidades dos cenários dos projetos (ROCHA et al., 2010).

No contexto em que este estudo está inserido, existe uma necessidade de explorar e entender os desafios do gerenciamento do processo de evolução de software em contextos distribuídos – sejam eles locais ou globais –, as fontes desses desafios e as estratégias de mitigação das organizações, caso elas usem alguma (LEHMAN, 1980; BENNETT; RAJLICH, 2000; JABANGWE et al., 2016). Até onde foi identificado pelo autor deste estudo, e por outros autores (ULZIIT et al., 2015), embora existam muitos estudos focados nos desafios da engenharia de software global em geral, não foram identificados estudos abrangentes realizados no contexto de gerenciamento do processo de evolução distribuída de software (CAMILO et al., 2018). É importante destacar que, apesar de compartilharem algumas características, existem distinções claras entre desenvolvimento, manutenção e evolução de software, que serão apresentadas adiante neste estudo, bem como evolução global e local. O foco deste estudo está no contexto global. Ou seja, todos os contextos considerados neste estudo enfrentam distâncias culturais, temporais e/ou geográficas.

Estudos identificaram vários desafios de gerenciamento de projetos de software e estratégias de mitigação e os classificaram sob fatores de pessoas, processos, produtos e tecnologia (ESTEVES; PASTOR, 1999; ULZIIT et al., 2015). No contexto deste estudo, considera-se que o processo de evolução distribuída de software tem desafios específicos de gerenciamento em relação a processos, pessoas, produtos e tecnologia. Portanto, as empresas

que realizam evolução de software em ambientes distribuídos devem considerar esses fatores, que parecem não estar explícitos na literatura de desenvolvimento de software global, embora muitas lições se apliquem a ambos os contextos.

Como contribuição deste trabalho, foi desenvolvida uma estrutura conceitual que identifica e descreve os desafios e soluções da evolução distribuída de software pela perspectiva de organizações clientes de softwares corporativos proprietários. Por fim, a partir das soluções mapeadas, a estrutura conceitual abriu espaço para melhorias no fluxo do gerenciamento do processo de evolução de distribuída de software, inicialmente identificado pelo autor deste trabalho em um artigo publicado anteriormente (CAMILO et al., 2018).

De um ponto de vista prático, estas informações podem ser valiosas de diferentes maneiras. Primeiro, os profissionais podem conduzir o gerenciamento do processo de evolução de software determinando quais desafios eles podem enfrentar ao tornar a evolução distribuída em cenários globais. Segundo, estes profissionais podem encontrar soluções para os desafios que estão enfrentando em cada uma das etapas do fluxo do processo de evolução distribuída de software (CAMILO et al., 2018).

Para alcançar essas contribuições, foram realizadas entrevistas com profissionais da indústria em organizações clientes de sistemas corporativos ERP proprietários. Para análise dos dados foi utilizado o método de teoria fundamentada (STRAUSS; CORBIN, 2008).

1.1 OBJETIVO

Para aprimorar o estado da arte, e preencher essa lacuna encontrada na literatura, o objetivo deste trabalho é identificar e descrever fatores específicos da evolução distribuída de software sob uma estrutura conceitual (WIERINGA; DANEVA, 2015) sob a perspectiva de organizações clientes de softwares corporativos ERP proprietários. Para estabelecer as contribuições deste estudo, são apresentados os seguintes objetivos específicos:

- Identificar o atual fluxo do processo de gerenciamento da evolução distribuída de software pela perspectiva de organizações clientes de sistemas corporativos ERP proprietários (CAMILO et al., 2018);
- Descrever uma estrutura conceitual de desafios e soluções para o gerenciamento do processo de evolução distribuída de software;
- Mapear a estrutura conceitual sobre o atual fluxo de gerenciamento do processo de evolução distribuída de software (CAMILO et al., 2018).

Com a evolução de software acontecendo de maneira distribuída, tanto o desenvolvimento quanto o gerenciamento do projeto tornam-se mais complexos e, consequentemente, acabam herdando características e problemas inerentes à configuração distribuída. Nesse sentido, é importante identificar desafios e soluções desse tipo de cenário e mapeá-los para que a aplicação conjunta desses conceitos seja realizada de forma que não haja prejuízos ao pleno funcionamento do sistema corporativo.

No estudo realizado por Camilo et al. (2018) os autores identificaram o fluxo do processo de evolução distribuída de software e alguns desafios decorrentes do cenário de evolução de software em conjunto com o desenvolvimento distribuído de software, como: ausência de formalização das definições, falta de priorização, falta de padronização da comunicação, falta de coordenação das partes interessadas, heterogeneidade de processos, centralização de conhecimento, alto nível de interferência de diferentes projetos rodando paralelamente, constantes mudanças demandadas por diferentes partes interessadas e o grande número de partes interessadas envolvidas.

Desta forma, considerando os fatores e dados citados que norteiam as motivações deste trabalho, o propósito é desenvolver uma estrutura conceitual que seja base para o processo de evolução distribuída de software. Com a abordagem é possível estabelecer um conjunto organizado de desafios e soluções que contribuem e orientam as organizações clientes de softwares corporativos no gereciamento do processo de evolução distribuída de software. E, consequentemente, é possível estabelecer uma forma de também mapear esta estrutura conceitual sobre as etapas do fluxo do processo de evolução distribuída de software, proporcionando suporte às organizações clientes de softwares corporativos durante todo o gerenciamento deste.

Logo, os resultados alcançaram um diferencial positivo em relação aos estudos existentes, visto que a abordagem abrange duas grandes áreas: Desenvolvimento Global de Software e Evolução de Software, conforme apresentado na Figura 1. Como constructo, o autor deste trabalho utilizou Ulziit et al. (2015), que teve como abordagem a combinação de outras duas grandes áreas: Desenvolvimento Global de Software e Manutenção de Software, conforme apresenta a Figura 1.

1.2 ESTRUTURA DO TRABALHO

Este trabalho está organizado, contando com este capítulo, da seguinte maneira:

I. Capítulo 1 – Introdução: visa a posicionar o leitor em relação ao projeto,

Figura 1: Localização da abordagem perante as áreas de conhecimento

apresentando uma breve explanação sobre o contexto da pesquisa, principais características do ambiente em que o projeto está inserido e ainda apresenta os objetivos e motivação para execução deste trabalho;

II. Capítulo 2 – Fundamentação Teórica: este capítulo visa proporcionar ao leitor um embasamento teórico sobre os temas e principais autores da literatura que serão abordados durante a execução deste projeto;

III. Capítulo 3 – Métodos e Procedimentos de Pesquisa: apresenta o método de pesquisa definido para execução deste trabalho, a configuração e o protocolo de pesquisa;

IV. Capítulo 4 – Análise e Interpretação dos Resultados: apresenta-se o resultado final do trabalho, isto é, todas as características da estrutura identificada e também as melhorias identificadas;

V. Capítulo 5 – Conclusão: descreve a conclusão referente ao trabalho realizado e as contribuições dadas à área de conhecimento e ao estado da arte.

2 FUNDAMENTAÇÃO TEÓRICA

A base teórica necessária para a realização da pesquisa e o entendimento do estudo é apresentada nesse capítulo. O capítulo está organizado da seguinte forma:

2.1 **Desenvolvimento de Software Global** – nesta seção são apresentados conceito e características sobre Desenvolvimento Global de Software.

2.2 **Manutenção de Software** – nesta seção são apresentados conceito e características sobre Manutenção de Software.

2.3 **Evolução de Software** – nesta seção são apresentados conceito e características sobre Evolução de Software.

2.5 **Sistemas ERP** – nesta seção são apresentados conceito e características sobre sistemas ERP.

2.4 **Evolução Distribuída de Software** – nesta seção são apresentados conceito e características sobre Evolução Distribuída de Software.

2.1 DESENVOLVIMENTO DE SOFTWARE GLOBAL

Diferentes modos de colaboração existem para o trabalho colaborativo no desenvolvimento de software distribuído, por exemplo, *onshore outsourcing, onshore insourcing, offshore outsourcing, offshore insourcing* (SMITE et al., 2008; ŠMITE et al., 2010; HOLMSTROM et al., 2006). O foco deste estudo é investigar os locais onde a evolução é realizada em diferentes países (*offshore*), seja com o envolvimento de terceiros (*outsourcing*) ou internamente (*insourcing*).

De acordo com Gonzalez et al. (2005), é possível identificar que as atividades de desenvolvimento de software *offshore* estão em andamento há mais de uma década. No entanto, as empresas de desenvolvimento de software ainda enfrentam muitos problemas. Além disso, diferentes riscos também foram identificados na distribuição do desenvolvimento, manutenção

Tabela 1: Desafios gerais de uma configuração distribuída globalmente (HERBSLEB; MOCKUS, 2003; MOE; ŠMITE, 2007; LANUBILE et al., 2003)

	Distância Geográfica	Distância Cultural	Distância Temporal
Comunicação	Confiança, coesão de equipe e sobrecarga de comunicação	Diversidade cultural	Comunicação síncrona
Coordenação	Distribuição de tarefas	Compartilhamento de conhecimento	Interação dentro de diferentes fusos horários
Controle	Acompanhamento do progresso	Decidir sobre o estilo de gerenciamento a ser usado	Consumo de tempo

e evolução de software, como excesso de custos e tempo, e diferenças culturais. Alguns estudos mostram que esta metodologia de desenvolvimento também apresenta um grande conjunto de desafios (HERBSLEB; MOCKUS, 2003; MOE; ŠMITE, 2007; LANUBILE et al., 2003). Esses desafios estão resumidos na Tabela 1.

Apesar das vantagens oriundas da dispersão física e temporal das equipes, as organizações precisam se atentar aos desafios gerados por meio desta prática relacionados tanto à comunicação, quanto à coordenação, cooperação e controle na execução das tarefas, tais como: os níveis de distância entre membros, diferenças culturais, fusos horários, falta de padronização de processos, incompatibilidade de ferramentas e infraestrutura (SANTOS, 2011).

No desenvolvimento global de software, cinco estudos (VERNER et al., 2014; NIDHRA et al., 2013; IQBAL et al., 2012; NOLL et al., 2010; ULZIIT et al., 2015) têm um objetivo semelhante ao deste trabalho, com a diferença de que eles estão focados no desenvolvimento de software geral ou manutenção de software global, a saber, identificar desafios e estratégias de mitigação. Verner et al. (2014) classificaram os estudos existentes em um estudo terciário, no qual nenhum deles tinha um foco particular na evolução de software. Os estudos tabulam desafios e soluções existentes para problemas no desenvolvimento de software global e manutenção de software global.

Especificamente Ulziit et al. (2015), um dos estudos mais recentes identificados até então, investigou os fatores específicos para gerenciar o processo de manutenção de software em configurações distribuídas globalmente. Os autores identificaram vários desafios de gerenciamento e estratégias de mitigação e os classificou sob fatores de pessoas, processos, produtos e tecnologia. No geral, foi desenvolvida uma estrutura de desafios e soluções, que pode ser usada para entender e classificar desafios globais de manutenção de software. Como resultado, os autores identificaram que o processo de manutenção de software distribuído tem

desafios específicos de gerenciamento em relação a processos, pessoas, produtos e tecnologia.

Trabalhos mais recentes reforçam a existência de desafios no gerenciamento de solicitações de mudança no contexto da engenharia de software global (AKBAR et al., 2019; ANWAR et al., 2019). Akbar et al. 2019, por exemplo, realizaram em seu estudo uma classificação categórica de desafios identificados nesse contexto. Embora o contexto do trabalho desses autores envolva qualquer tipo de solicitação de mudança – não as categorizando como desenvolvimento, manutenção ou evolução –, este serve como constructo para mostrar que desafios ainda permanecem presentes nesse contexto.

2.2 MANUTENÇÃO DE SOFTWARE

Nos dicionários linguísticos populares (MISH, 2004; MCKEAN, 2005), a manutenção é definida como trabalho de preservar as coisas na ordem correta ou em boas condições. Boehm (1981) afirmou que a manutenção é um processo de alterar o software operacional existente sem alterar suas funções fundamentais.

A fase de manutenção é a parte mais longa do ciclo de vida do software e, na maioria dos casos, também é a mais cara (CANFORA; CIMITILE, 2001; AHMED, 2006; HUNT et al., 2008). Nas últimas décadas, o custo da manutenção de software cresceu continuamente. Na década de 1970, os custos estavam em torno de 60% (LIENTZ; SWANSON, 1980a), enquanto nos anos 90 e 2000, os custos relatados aumentaram para cerca de 90% (ERLIKH, 2000). Portanto, as organizações têm visto a mudança do modo de trabalho de manutenção e evolução como uma oportunidade para facilitar o gerenciamento das atividades e redução de custos.

De acordo com a definição de Boehm (1981), os seguintes tipos de atividades são abordados no escopo de manutenção: projeto e correção de partes menores em software existente; redesenho e redesenvolvimento de pequenas partes do software existente; e modicação do código de software existente, partes da documentação ou estrutura do banco de dados. Swanson (1976) classificou as atividades de manutenção em manutenção corretiva, adaptativa e perfectiva. Posteriormente, a taxonomia foi revisada na ISO / IEC 14764 (2006).

Todos estes tipos de manutenção geram a necessidade de modificação no software e a consequente evolução do mesmo. Ainda que seja mais frequente durante a atividade de manutenção, a evolução está presente em todo o ciclo de vida do software. O ciclo de vida do software começa com a necessidade de se ter o software por parte do cliente, passa por diversas fases de desenvolvimento, e termina quando o software deixa de ser utilizado (NOVAIS, 2017).

Em geral, vários desafios são observados na manutenção de software global. Muitos

estudos enfatizaram que um dos desafios mais críticos enfrentados pelos engenheiros de software é lidar com a manutenção de software e gerenciar o controle de mudanças (CHEN; HUANG, 2009; ERDIL et al., 2003; LI et al., 2010; HUNT et al., 2008). Além disso, a manutenção e a evolução de software são as fases menos exploradas e, ao mesmo tempo, as fases mais subestimadas e problemáticas (ERDIL et al., 2003; WOOD et al., 1997). Chen e Huang (2009) notaram que, apesar do alto custo, menos atenção é dada aos problemas encontrados nas atividades de manutenção e evolução do que outras atividades de desenvolvimento.

2.3 EVOLUÇÃO DE SOFTWARE

A evolução de software é o subdomínio da disciplina de engenharia de software que investiga maneiras de adaptar o software aos requisitos de usuário em ambientes de operação em constante mudança (MENS; DEMEYER, 2008; RAJLICH, 2014). Desde o trabalho clássico de Lehman e Belady (1985), a evolução de software foi aceita como um fenômeno que merece ser estudado pelos diversos casos de problemas encontrados em projetos de software. Os problemas são complexos porque envolvem muitas dimensões, afetando, entre outras, todas as fases do processo de software, relações sociais, aspectos gerenciais e econômicos, linguagens e ambientes de programação. Além disso, à medida que avanços de engenharia de software e novas tecnologias e processos são introduzidos, a evolução de software acaba por enfrentar novos problemas e desafios (MENS; DEMEYER, 2008).

As oito leis da evolução de software, também conhecidas por “Leis de Lehman”, começaram a ser formuladas entre as décadas de 70 e 90, com a análise do processo de programação da IBM (LEHMAN, 1996). Nesse artigo o autor mostra que o crescimento evolutivo de um software diminui ao longo do tempo e parte do crescimento vem de adições de novas funcionalidades e suporte a novas arquiteturas, e não de simples correções de erros de versões. A Tabela 2 apresenta as leis.

As leis de Lehman ajudam a compreender como ocorre o processo evolutivo de um software no seu aspecto funcional. Os estudos que levaram às formulações das leis usaram como base processos de desenvolvimento de sistemas baseados em sistemas centralizados e corporativos, com código fechado e de grande porte, com poucos concorrentes no mercado e para uso de grandes corporações (LEHMAN, 1996).

No entanto devido ao avanço das técnicas, processos e práticas de desenvolvimento, manutenção e evolução de software no cenário mundial, verificou-se que as leis da evolução de

Tabela 2: As oito leis de Evolução de Software (LEHMAN, 1996)

Lei	Descrição
Mudança contínua	Um sistema de informação deve ser continuamente adaptado, caso contrário se torna progressivamente menos satisfatório.
Complexidade crescente	À medida que um programa é alterado, sua complexidade cresce a menos que um trabalho seja feito para mantê-la ou diminuí-la.
Auto-regulação	O processo de evolução de software é auto-regulado próximo à distribuição normal com relação às medidas de produtos e atributos de processos.
Conservação da estabilidade organizacional	A taxa de atividade global efetiva média em um sistema em evolução é constante sobre o tempo de vida do produto.
Conservação da familiaridade	Durante a vida produtiva de um programa em evolução, o índice de alterações em versões sucessivas é estatisticamente invariante.
Crescimento contínuo	O conteúdo funcional de um programa deve ser continuamente aumentado para manter a satisfação do usuário durante seu tempo de vida.
Qualidade decrescente	Programas apresentarão qualidade decrescente a menos que sejam rigorosamente mantidos e adaptados às mudanças no ambiente operacional.
Sistema de retorno	Processos de programação de software constituem sistemas de multi-loop, multi-nível e devem ser tratados como tais para serem modificados e melhorados com sucesso.

software necessitam de uma revisão profunda. A princípio imaginavam que todas as leis se aplicassem a evolução de qualquer tipo de software, mas estudos com o softwares de código aberto, por exemplo, mostram que nem sempre isto é verdade (GODFREY; GERMAN, 2008). Estudos têm avaliado as leis de Lehman por perspectivas de sistemas descentralizados e projetos de software de código aberto, e chegam a conclusão de que as leis não são totalmente aplicáveis a todos os tipos de sistema (SCACCHI, 2004; OLIVEIRA et al., 2015; AMANATIDIS; CHATZIGEORGIOU, 2016). No entanto, não é foco deste estudo avaliar as leis de Lehman, mas sim, utilizá-las como base para a motivação do estudo do processo de evolução de software.

A definição de evolução de software muitas vezes se confunde com a manutenção de software, uma vez que uma manutenção de software faz com que o software evolua (NOVAIS, 2017). A atividade de manutenção é, de fato, complexa e impacta significativamente nos custos do software. Essa afirmação foi dada por Lientz e Swanson (1980b) e, mais de 20 anos depois, ela ainda permanecia válida (ERLIKH, 2000; SEACORD et al., 2003; KOSKINEN, 2003; SOUZA et al., 2005).

Também é importante entender a distinção entre evolução de software em contexto distribuído globalmente e em contextos internos tradicionais. Projetos globalmente distribuídos são mais difíceis do que projetos internos para gerenciar (BIANCHI et al., 2003). Geralmente, isso se deve à diversificação e ao isolamento dos membros da equipe do projeto devido à distância (geográfica, temporal e sociocultural), que tem impacto na comunicação, coordenação e controle. (HERBSLEB; MOCKUS, 2003; MOE; ŠMITE, 2007; LANUBILE et al., 2003).

Bhatt et al. (2006a) apresentam que o desenvolvimento, manutenção e evolução de software são processos diferentes (ESTEVES; PASTOR, 1999; EDEN et al., 2014; ULZIIT et al., 2015). E as configurações distribuídas globalmente trazem tanto algumas fontes comuns de desafios, como também grandes diferenças (BHATT et al., 2006a). A Tabela 3 mostra algumas diferenças principais entre desenvolvimento, manutenção e evolução de software. Dessa forma, foi possível identificar características que diferem, igualam ou até combinam os diferentes conceitos.

2.4 EVOLUÇÃO DISTRIBUÍDA DE SOFTWARE

Camilo et al. (2018) realizaram um estudo de caso em uma organização multinacional, cliente de um grande sistema corporativo ERP proprietário. O objetivo era identificar como era

Tabela 3: Principais diferenças entre desenvolvimento, manutenção e evolução de software distribuído.

	Desenvolvimento	Manutenção	Evolução
Pessoal	Atribuir pessoal experiente	Subestimar e designar pessoal novo ou inexperiente	Atribuir pessoal experiente
Habilidades técnicas necessárias	Especialistas individuais para cada habilidade	Análise de requisitos, tecnologias de desenvolvimento, testes, implementação e interação com o cliente	Combinação
Conhecimento de domínio	Conhecimento de unidades de software responsáveis	Conhecimento de todo o produto de software e seu fluxo de negócios	Ambas possibilidades
Fonte de conhecimento	Principalmente código fonte	Código-fonte, documentação e onde encontrar especialistas responsáveis	Código-fonte, documentação e onde encontrar especialistas responsáveis
Grande esforço de trabalho	Principalmente escrevendo código	70% do trabalho é entender o código existente e depurar	Ambas possibilidades
Fonte	(ROBILLARD et al., 2007)	(BHATT et al., 2006b)	(MENS; DEMEYER, 2008)

realizado o gerenciamento do processo de evolução de software em um cenário distribuído.

Os autores identificaram fatores, como: tipos de mudanças que ocorriam no sistema corporativo, partes interessadas envolvidas e o fluxo do processo de evolução distribuída do cenário em questão, além de como ocorre a coordenação dos stakeholders nesse ambiente. O estudo se encontra no apêndice B deste documento (CAMILO et al., 2018).

De acordo com os autores do estudo, as demandas por mudanças no sistema corporativo em questão podem advir de diversas origens internas e externas. No estudo, foram destacadas as principais encontradas (CAMILO et al., 2018):

Mudanças na Legislação. O governo (nacional, estadual, municipal) pode emitir, por meio de suas várias agências e órgãos reguladores, diversos tipos de mudanças, que podem compreender alterações na legislação, podendo impactar diretamente nas áreas de negócio da organização corporativa. Um dos fatores críticos nas mudanças na legislação está no atendimento aos prazos propostos pelo governo.

Mudanças na Tecnologia. As mudanças no processo são consequência das mudanças no contexto da indústria, muitas vezes despertadas por inovações tecnológicas, como inovações no controle da rastreabilidade de produtos, ou na automação de equipamentos fabris ou até na

Tabela 4: Stakeholders identificados por Camilo et al. (2018)

Stakeholder	Descrição
Provedores de Tecnologia	Mercado global que atua na evolução tecnológica, disponibilizando novas soluções de integração, linguagens, software e hardware para melhorar o contexto atual da tecnologia.
Agências Internacionais	Atua na elaboração de normas e metodologias mundiais de operações estruturadas que auxiliam as organizações nas áreas de Qualidade, Segurança e Meio Ambiente. Devido à evolução das normas de segurança, controle, e qualidade, por exemplo, os softwares também devem evoluir.
Governo	Composto por agências que atuam na elaboração de normas, legislações e impostos nacionais. As organizações corporativas estão sujeitas à evolução de seus softwares em atendimento às demandas do governo a nível nacional, estadual e municipal.
Produtora do Software	Atua no desenvolvimento e venda do sistema corporativo, fornecendo melhorias pressionadas pela tecnologia ou customizações demandadas pelos demais stakeholders.
Produtora do Software (subsidiária local)	Subsidiária local da produtora do software que atua na representação local do sistema corporativo no país de operação do sistema, caso a produtora do software detenha uma.
Parceiros Externos	Atuam principalmente no desenvolvimento de soluções para implantação de projetos, sendo responsáveis por auxiliar a organização corporativa no diagnóstico e desenvolvimento de soluções para sistemas corporativos.
Organização Cliente	Cliente do sistema corporativo, composta por departamentos, clientes, fornecedores. Também é composta por funcionários que podem ser: usuários finais, consultores internos, usuários chave, analistas de suporte ou programadores internos, que trabalham diretamente com o sistema corporativo.

tecnologia de servidores, linguagens e banco de dados, que refletem diretamente na evolução do sistema corporativo para se adequar às novas necessidades tecnológicas.

Mudanças nos Padrões Internacionais. Consequência das exigências do mercado e certificações com assuntos como qualidade, segurança e meio ambiente, as entidades e agências internacionais elaboram normas e padrões internacionalmente reconhecidos. A empresa também está sujeita às certificações exigidas pelos atuais e potenciais clientes.

Mudanças na Política Corporativa. Consequência das mudanças na política e modo de trabalho da organização corporativa, muitas vezes o sistema corporativo precisa se adaptar aos novos processos de produção, logística, financeiro, contábil, etc.

A Tabela 4 apresenta os stakeholders identificados pelos autores no processo de evolução distribuída de software (CAMILO et al., 2018).

A Figura 2 apresenta o fluxo de do processo de evolução distribuída de software

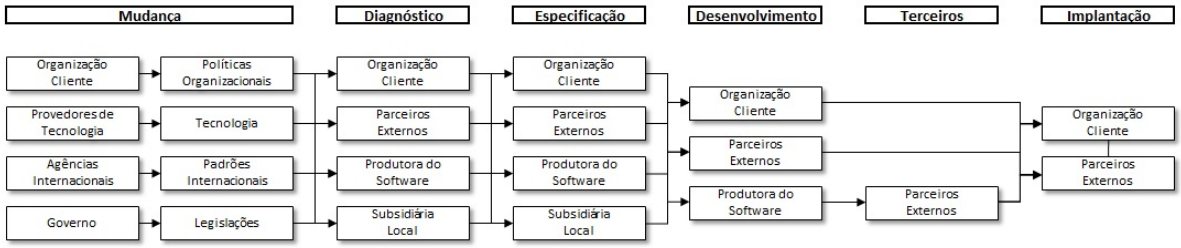

Figura 2: Fluxo genérico de evolução distribuída de um sistema corporativo proprietário, identificado por Camilo et al. (2018)

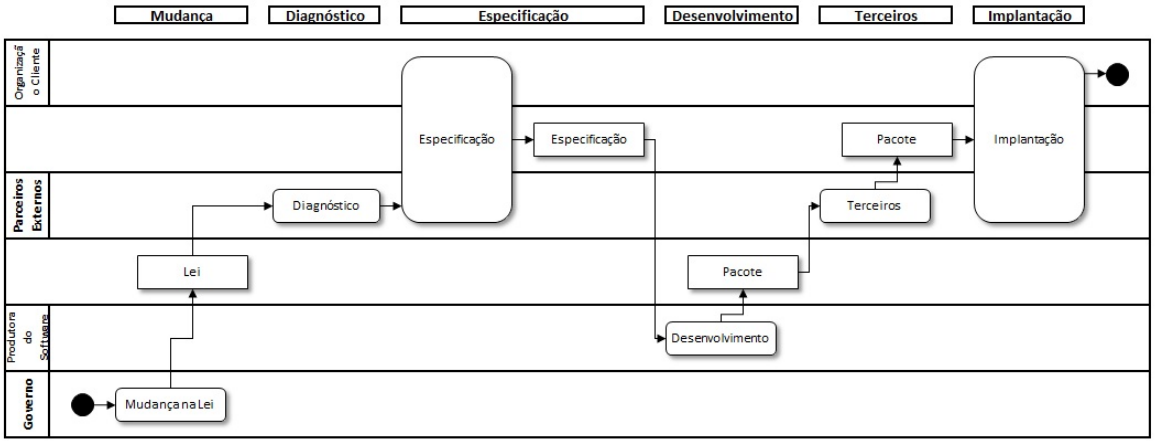

Figura 3: Fluxo de evolução distribuída identificado em uma mudança na legislação nacional identificado por Camilo et al. (2018).

identificado por Camilo et al. (2018). O fluxo é dividido em etapas, que são: Mudança, Diagnóstico, Especificação, Desenvolvimento, Terceiros, Implementação. Fluxo este que pode ocorrer por meio de múltiplos caminhos, dependendo do tipo de demanda e dos stakeholders que participam do processo. As partes interessadas podem estar fisicamente ou temporalmente distantes e podem ou não participar da mesma etapa. As setas representam as partes interessadas que podem ou não participar de cada etapa do fluxo. Portanto, é possível que diferentes partes interessadas participem ou não de um mesmo Diagnóstico, Especificação, Desenvolvimento ou Implantação. A Figura 3 apresenta um exemplo dos múltiplos fluxos possíveis que podem ser executados quando há uma mudança na lei nacional.

2.5 SISTEMAS ERP

Sistemas ERP são definidos como um tipo de sistema corporativo que permitem a integração completa do fluxo de informações de todas as áreas funcionais (financeira, produção, contabilidade, vendas, logística, entre outras) em organizações corporativas por meio de um banco de dados, acessível por meio de uma interface unificada

(ADDO-TENKORANG; HELO, 2011). Um sistema ERP é basicamente um software modularizado ou empacotado (NG; GABLE, 2010), com capacidade de personalização ou adição de funcionalidades e módulos, ou até pacotes por parte de terceiros, não sendo desenvolvido como outros tipos de projetos de software (LOPEZ; SALMERON, 2011).

Projetos em sistemas corporativos ERP diferem de outros tipos de projetos de software, pois cobrem milhares de atividades comerciais, requerem diversas configurações e atividades de modificação para refletir requisitos comerciais imediatos (DANEVA; WIERINGA, 2008), dado o nível de integração, dependência e complexidade de seus diferentes módulos e áreas funcionais. Diferem de projetos de software com fases de personalização, modificação, integração e conversão de dados (OMURAL; DEMIRORS, 2017).

A estrutura do ciclo de vida do sistema corporativo ERP, conforme Figura 4, consiste em seis fases que representam as etapas da vida de um sistema ERP dentro das organizações. Constitui também quatro dimensões, que são vistas como pontos de vista que podem ser usados para analisar essas fases. As fases são a decisão de adoção, aquisição, implementação, uso e manutenção, e evolução e aposentadoria. As dimensões, também vistas como questões importantes dentro da fase de evolução, são: pessoas, processos, produtos (ESTEVES; PASTOR, 1999; ULZIIT et al., 2015) e tecnologia (ULZIIT et al., 2015). Como o foco principal deste estudo é a evolução, este descreverá as quatro dimensões da estrutura do ciclo de vida do ERP em relação à fase de evolução de um sistema ERP proprietário pela perspectiva de organizações clientes desse tipo de sistema corporativo.

Os sistemas ERP precisam ser atualizados regularmente para corrigir problemas, aumentar a segurança, ampliar as funcionalidades ou módulos atuais (HA; AHN, 2014), convocar regulamentações de legislação, atualizar processos e atender à organizações em constante evolução (CALVERT; SEDDON, 2006).

A fase de evolução é sobre a introdução e integração de mais recursos e funcionalidades no sistema ERP (ESTEVES; PASTOR, 1999). Albadri e Abdallah (2009) afirmam que os desafios enfrentados frequentemente durante a fase de evolução são reconhecidos como importantes, mas inadequadamente abordados na pesquisa. Esta fase não tem sido um centro de atenção na literatura anterior sobre sistemas ERP, em geral (KOTB et al., 2011), e nem no contexto da evolução distribuída de software. No contexto deste estudo, foram incluídas apenas investigações abordando a fase de evolução do ciclo de vida do ERP (EDEN et al., 2014).

Os projetos neste tipo de sistema corporativo diferem de outros tipos de projetos de

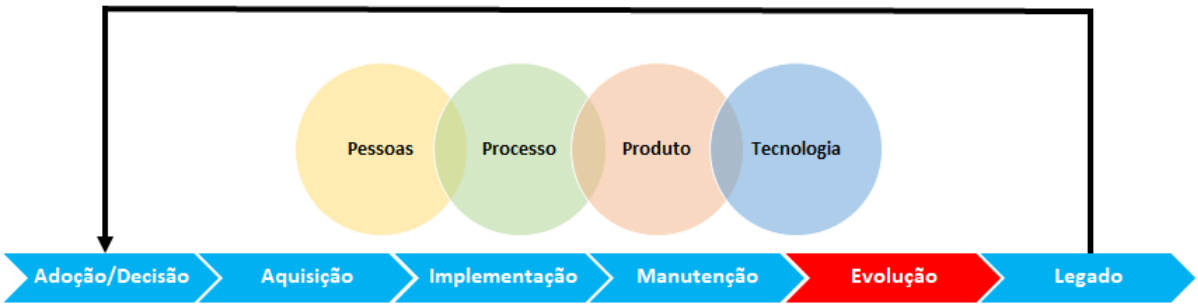

Figura 4: A estrutura do ciclo de vida do ERP (ESTEVES; PASTOR, 1999; EDEN et al., 2014; ULZIIT et al., 2015).

software, pois cobrem milhares de atividades comerciais de diversas áreas funcionais – financeira, produção, contabilidade, e assim por diante –, requerem diversas configurações e atividades de modificação para refletir requisitos comerciais imediatos (DANEVA; WIERINGA, 2008; OMURAL; DEMIRORS, 2017).

Embora exista uma grande quantidade de sistemas de código aberto que podem ser estudados livremente, a grande maioria dos softwares comerciais ainda é criada como fonte proprietária e fechada. Além disso, os estudos de software proprietário geralmente são submetidos às restrições que limitam os detalhes que podem ser publicados. Isso coloca dois problemas fundamentais: primeiro, muitos estudos empíricos não podem ser facilmente encontrados, replicados ou estendidos; e em segundo lugar, muitos detalhes são geralmente deixados fora do estudo (GODFREY; GERMAN, 2008).

3 METÓDOS E PROCEDIMENTOS

O método científico é um conjunto de abordagens, técnicas e processos utilizados pela ciência para formular e solucionar problemas de aquisição objetiva do conhecimento. A metodologia determina qual o modelo de solução empregado sobre um problema que se apresenta sob um determinado âmbito com um escopo definido e uma profundidade estabelecida (L'ERARIO, 2009).

Para gerar conhecimento de maneira sistemática, deve ser definido ao menos um método de pesquisa, considerando o cenário e o propósito. Existem vários métodos de pesquisa, como por exemplo: pesquisa descritiva, não experimental e exploratória; levantamento amostral ou *survey*; experimentação; pesquisa participante; estudo de caso; teoria fundamentada.

O método de teoria fundamentada é um método de geração indutiva de teorias assentada em dados, por meio de análise qualitativa destes e que, agregados ou relacionados a outras teorias, poderão acrescentar ou gerar novos conhecimentos à área de conhecimento do fenômeno estudado. Os procedimentos da teoria fundamentada têm como finalidade identificar, desenvolver e relacionar conceitos, para assim formular uma teoria (STRAUSS; CORBIN, 2008).

Três tipos de teorias são destacadas por Wieringa e Daneva (2015), a saber, estruturas conceituais: fornecendo um conjunto de conceitos descrevendo e explicando um fenômeno; generalizações: para determinar em que grau as descobertas se aplicam a um conjunto maior de empresas; e modelos de inferência: raciocínio que usa dados disponíveis para se chegar a uma conclusão.

O investigador identifica processos que estão ocorrendo em um contexto, e que podem explicar um fenômeno, combinando abordagens indutivas e dedutivas. A teoria está fundamentada nos dados, e não tão somente num corpo existente de teoria, embora também possa compreender um conjunto de diversas outras teorias. Portanto, não se pretende testar ou provar teorias, mas identificar, desenvolver e relacionar conceitos e, consequentemente,

acrescentar novas perspectivas ao entendimento e funcionamento do fenômeno (STRAUSS; CORBIN, 2008).

A entrevista é uma das fontes de coleta de dados qualitativos do método de teoria fundamentada. Uma das características da entrevista refere-se ao fato dela favorecer a relação intersubjetiva entre o pesquisador e o entrevistado e, por meio de trocas verbais e não verbais, permitir uma maior compreensão da opinião de pessoas sobre situações e vivências pessoais (GUERRA, 2010).

A entrevista pode ser do tipo formal ou informal. A entrevista formal segue um plano de ação determinado e é utilizado quando se deseja obter informações com maior profundidade. A entrevista também pode ser estruturada, semiestruturada ou não-estruturada. Na entrevista semiestruturada, o entrevistador faz perguntas predeterminadas, no entanto, com a possibilidade de abertura para novas perguntas, conforme a entrevista é realizada (GUERRA, 2010).

Os dados para análise geralmente constam de transcrições, anotações ou outros tipos de documentos. Após o investigador ter coletado os dados iniciais, realiza as leituras e procede à codificação ou à análise dos dados (STRAUSS; CORBIN, 2008).

A teoria fundamentada começa com codificação aberta e é seguida por codificação axial e seletiva (STRAUSS; CORBIN, 2008; CORBIN; STRAUSS, 2008):

- A codificação aberta é um processo de codificação inicial no qual os códigos são provisórios e melhor fundamentados nos dados;
- Esses códigos são selecionados para ter um significado próximo dos dados brutos, de modo que outros processos de codificação possam abertamente explorá-los;
- Na codificação axial, os códigos abertos estão relacionados e mapeados entre si para construir uma estrutura conceitual que possa explicar os fenômenos de maneira mais precisa e completa;
- Posteriormente, o processo de codificação seletiva aplica-se a integrar os conceitos para construir categorias mais gerais para explicar a teoria sobre um fenômeno e validar as relações de conceito comparando com os dados brutos e refinar a relação, se necessário.

O objetivo de pesquisa deste estudo é conseguir verificar quais são os desafios de gerenciar o proecesso de evolução de software em configurações distribuídas pela perspectiva de organizações clientes de softwares corporativos, e como as organizações clientes de softwares corporativos mitigam esses desafios. Visto que embora existam diversos estudos

focados nos desafios da engenharia de software global em geral (ULZIIT et al., 2015), não foram identificados estudos abrangentes realizados no contexto do gerenciamento da evolução distribuída de software (CAMILO et al., 2018).

Para alcançar o objetivo de pesquisa deste estudo, foram formuladas as seguintes questões de pesquisa:

- Q1: Quais são os desafios de gerenciar o processo de evolução distribuída de software pela perspectiva de organizações clientes de softwares corporativos?
- Q2: Como as organizações clientes de softwares corporativos mitigam esses desafios?

A fim de responder as perguntas de pesquisa, o autor investigou fatores específicos para gerenciar o processo de evolução de software em configurações distribuídas globalmente. Seguindo o método de teoria fundamentada (STRAUSS; CORBIN, 2008), o autor executou entrevistas como técnica de coleta de dados.

O resultado deste estudo é uma estrutura conceitual, descrevendo e explicando conceitos relacionados a desafios e soluções no gerenciamento da evolução distribuída de software pela perspectiva de organizações clientes de softwares corporativos.

A Figura 5 mostra uma visão geral completa da metodologia e execução do estudo, ilustrando como o autor adaptou o método de teoria fundamentada às necessidades específicas deste trabalho. Conforme apresentado no Capítulo 2, estudos anteriores fornecem informações que contribuem para o desenho e execução deste estudo. No desenvolvimento e execução da entrevista, o autor utilizou um guia de entrevista, com base no referencial teórico, e realizou entrevistas com diferentes organizações. Na coleta de dados, o autor integrou os resultados empíricos em um banco de dados e, posteriormente, os processou na análise de dados para formar categorias teóricas. Por fim, utilizou os achados para sugerir melhorias no fluxo do processo de evolução distribuída de software. O processo representado na Figura 5 é evolutivo e afeta o desenho a cada novo estudo publicado.

3.1 DESENVOLVIMENTO E EXECUÇÃO

Os dados de entrada para as entrevistas partiram do processo de evolução distribuída de software (CAMILO et al., 2018), onde todos os entrevistados foram apresentados ao fluxo genérico do processo, conforme Figura 2 (seção 2.4). A intenção do autor era aproveitar a coleta

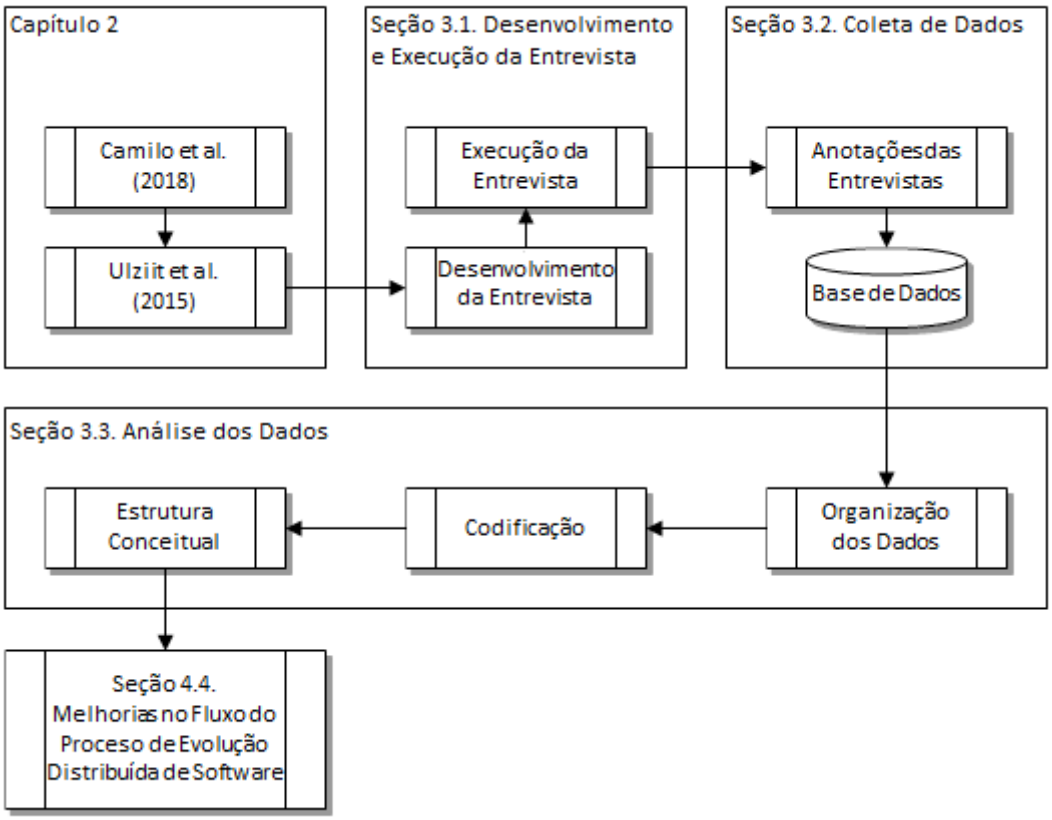

Figura 5: Método de teoria fundamentada (adaptado de Strauss e Corbin 2008)

de dados das entrevistas para classificar os desafios e soluções frente às etapas do processo de evolução distribuída de software.

O curso da entrevista foi elaborado para seguir as diretrizes de Kvale (2009), na qual a coleta de dados da entrevista é encaminhada por meio das etapas de tematização, desenho, entrevista, transcrição, análise, verificação e relato. O objetivo da entrevista foi explorar além do conhecimento do estado da arte.

A entrevista foi projetada de forma qualitativa e semiestruturada, onde as questões fechadas deveriam ser investigadas e questões abertas deveriam ser discutidas. A revisão bibliográfica permitiu a realização de uma entrevista mais focada para agregar conhecimento aos resultados encontrados até então. A utilização de Ulziit et al. (2015) e Camilo et al. (2018) como entrada para entrevistas permitiu obter informações adicionais ao que foi encontrado na revisão de literatura. O guia de entrevista completo é mostrado no Apêndice A.

Os entrevistados foram contactados por e-mail e a pesquisa foi apresentada. Eles receberam informações como o objetivo, o propósito, a estrutura e o resultado esperado do estudo, para se familiarizarem com a pesquisa. No total, foram entrevistadas cinco pessoas de cinco empresas diferentes. Uma entrevista foi realizada face a face, enquanto outras entrevistas foram realizadas por meio de ferramentas de videoconferência. Segundo Kvale e Brinkmann (2009), o entrevistado deve receber instruções sobre o contexto da entrevista antes desta ser realizada. Foi realizada uma breve introdução aos entrevistados, o objetivo do estudo, o formato das perguntas a serem feitas e a permissão do entrevistado para realizar anotações.

Antes da entrevista, por meio de consentimento, os entrevistados foram informados que a entrevista é voluntária, e poderiam interromper a entrevista ou se recusar a responder, e também poderiam recusar que anotações fossem realizadas. Eles também foram informados sobre a intenção de publicar os resultados consolidados. O consentimento informado é um acordo destinado a assegurar que os indivíduos dêem o consentimento para a participação voluntária em uma pesquisa, uma vez que sejam informados das atividades envolvidas e dos riscos associados a elas (DENZIN; LINCOLN, 2011). De modo a cumprir com isso, o autor obteve o consentimento dos participantes após informar as consequências da pesquisa. Em particular, o consentimento ajudou a neutralizar as preocupações dos participantes sobre a sua presença na entrevista e fez com que se sentissem mais seguros e cooperativos.

A estratégia de amostragem intencional (PATTON, 1990) foi utilizada para selecionar os entrevistados. Entrevistados experientes que trabalham em organizações clientes de sistemas corporativos proprietários foram selecionados de empresas de grande porte, onde é provável que fosse encontrado um rico conjunto de desafios e soluções relacionados à evolução distribuída.

Tabela 5: Entrevistados

ID	Empresa	Background
1	Organização 1: Uma grande indústria multinacional, com sede no Reino Unido, onde a principal atividade é o projeto, desenvolvimento, fabricação e venda de veículos para todo o mundo. Tem operações em cerca de 175 países, o que indica sua experiência com engenharia de software global. Conta com cerca de 40 mil funcionários ao redor do mundo.	Entrevistado 1 tem cerca de 19 anos de experiência na atuação com gerenciamento do processo de desenvolvimento global de software. Atua como representante funcional na América Latina. Sua equipe participa colaborativamente da implantação de melhorias e novas funcionalidades, destacando adequações e integrações do ERP às regras de negócio do Brasil.
2	Organização 2: Uma grande indústria multinacional, com sede no Brasil, onde a principal atividade é o industrialização de bebidas para todo o mundo. Tem operações em 3 países. Conta com cerca de 1000 mil funcionários ao redor do mundo.	Entrevistado 2 tem cerca de 3 anos de experiência na atuação com gerenciamento do processo de desenvolvimento global de software. Atua como representante funcional no Brasil. Sua equipe participa colaborativamente da implantação de melhorias e novas funcionalidades, destacando adequações e integrações do ERP às regras de negócio do Brasil.
3	Organização 3: Uma grande indústria multinacional, com sede no Estados Unidos, onde a principal atividade é a distribuição de produtos de tecnologia, focada em automação comercial, soluções corporativas e de segurança física para todo o mundo. Tem operações nos cinco continentes. Cerca de 33 mil clientes. Conta com cerca de 2,5 mil funcionários ao redor do mundo.	Entrevistado 3 tem cerca de 8 anos de experiência na atuação com gerenciamento do processo de desenvolvimento global de software. Atua como representante funcional no Brasil. Sua equipe participa colaborativamente da implantação de melhorias e novas funcionalidades, destacando adequações e integrações do ERP às regras de negócio do Brasil.
4	Organização 4: Uma grande indústria multinacional, com sede no Bélgica, onde a principal atividade é produção de mineração para todo o mundo. Tem operações/clientes em 25 países. Conta com cerca de 6 mil funcionários ao redor do mundo.	Entrevistado 4 tem cerca de 6 anos de experiência na atuação com gerenciamento do processo de desenvolvimento global de software. Atua como representante funcional no Brasil. Sua equipe participa colaborativamente da implantação de melhorias e novas funcionalidades, destacando adequações e integrações do ERP às regras de negócio do Brasil.
5	Organização 5: Uma grande indústria multinacional, com sede na Alemanha, onde a principal atividade é produção de polímeros, que atua nos setores de construção, indústria e como fornecedor automotivo para todo o mundo. Tem operações/clientes em 54 países. Conta com cerca de 22 mil funcionários ao redor do mundo.	Entrevistado 5 tem cerca de 12 anos de experiência na atuação com gerenciamento do processo de desenvolvimento global de software. Atua como representante funcional na Améria do Sul. Sua equipe participa colaborativamente da implantação de melhorias e novas funcionalidades, destacando adequações e integrações do ERP às regras de negócio do Brasil.

A Tabela 5 fornece uma visão geral da experiência dos entrevistados.

3.2 COLETA DE DADOS

Todas as entrevistas foram realizadas pelo autor deste estudo entre janeiro e maio de 2019. O papel do entrevistador foi liderar a entrevista, fazer anotações e fazer perguntas adicionais, caso necessário.

A duração de cada entrevista foi definida para 1 (uma) hora, mas estendida no caso de o entrevistado desejar continuar. Anotações foram utilizadas como método para gravar as entrevistas. Essas entrevistas serviram como base de dados para a análise de dados pelo método de teoria fundamentada.

3.3 ANÁLISE DOS DADOS

Seguindo as etapas do método de teoria fundamentada, os dados foram analisados conforme exemplo:

- Anotação da entrevista 1: "Algumas equipes eram muito boas em comunicação porque o inglês era a primeira língua delas, enquanto outras não."
- Anotação da entrevista 2: "Enfrento dificuldades em entender o que os outros estão tentando dizer. Isso acontece com muita frequência nas conversas por e-mail."

Os trechos de duas das entrevistas discutem os desafios associados à comunicação. O entrevistado 1 discute um desafio na comunicação devido a diferenças na linguagem, então é codificado como "diferença de linguagem". No segundo caso, um entrevistado discute o mesmo desafio, isto é, "diferença de linguagem", mas este também menciona um modo no qual cria um problema, que é o "e-mail". Portanto, isso pode ser codificado como "ambiguidades na correspondência por email". Este código é então adicionado a um código mais generalizado "comunicação escrita". Então, ambos os códigos são agrupados em um conceito "diferença de linguagem". Por fim, esse conceito é adicionado à categoria "Pessoas".

Como etapa seguinte, os dados foram compilados sobre a estrutura conceitual de Ulziit et al. (2015). Portanto, a estrutura conceitual de Ulziit et al. (2015) serviu de base para a construção da estrutura de desafios e soluções deste estudo.

Com o objetivo de identificar as etapas do fluxo do processo de evolução distribuída de software (CAMILO et al., 2018), onde os desafios mapeados acontecem, os entrevistados

Tabela 6: Etapas do processo de evolução distribuída de software (CAMILO et al., 2018)

Etapa	Descrição
1	Solicitação de Mudança
2	Diagnóstico
3	Especificação
4	Desenvolvimento
5	Terceiros
6	Implantação

ajudaram o autor a classificar as etapas do fluxo genérico do processo sobre os desafios e soluções desta estrutura conceitual. Essa classificação foi inserida como sendo a coluna "Etapa", nas tabelas da estrutura conceitual. As etapas são apresentadas na Tabela 6.

3.4 AMEAÇAS À VALIDADE

Ameaças à validade podem afetar a confiabilidade dos resultados. As ameaças relevantes são discutidas nesta seção.

As ameaças relevantes à coleta de dados, em geral, mas reduzidas por meio de ações de controle, foram as seguintes:

- Seleção de entrevistados: Uma ameaça relacionada à seleção do entrevistado foi que alguns dos entrevistados eram colegas do autor, ou até estavam trabalhando juntos em projetos de evolução distribuída do software corporativo. Assim, era mais fácil acessar e obter consentimento por causa do contato próximo. Embora, ao mesmo tempo, os entrevistados preenchessem o critério de amostragem intencional, estavam trabalhando em um contexto altamente relevante para este estudo.

- Privacidade: A coleta de informações limitada devido à privacidade da empresa foi outra ameaça. Para superar esse risco, o autor obteu o consentimento formal sobre a garantia de confidencialidade das informações e não divulgação das empresas envolvidas.

- Generalização: Outra ameaça potencial foi a validade da população entrevistada para generalização. O autor realizou entrevistas de profissionais com até 20 anos de experiência, que se relacionavam com diversos outros profissionais, e que também se relacionavam com pessoas de diferentes localizações geográficas. A ideia era coletar feedback de organizações com diferentes origens, culturas e estruturas

organizacionais. No entanto, o número de entrevistas foi limitado, embora informações profundas e detalhadas tenham sido coletadas. A ameaça é reduzida à medida que as entrevistas são analisadas em conjunto com as informações obtidas na literatura. Isso pode suportar a generalização dos resultados.

As ameaças relevantes às entrevistas, especificamente, foram as seguintes:

- Diferenças: As diferenças nas funções, responsabilidades, cultura organizacional e natureza do contrato entre empregado e empregador podem representar uma ameaça às entrevistas.
- Organizações clientes: Outra ameaça em potencial pode ser o fato de que foram entrevistadas somente organizações clientes de softwares corporativos, mas não foram incluídas as perspectivas da produtora do software, fornecedores do software, parceiros externos, e demais partes interessadas.

A ameaça relevante à análise de dados foi a seguinte:

- Uma grande ameaça à validade durante a análise dos dados diz respeito se as inferências que decorrem dos dados não são influenciadas pelo autor durante a análise. Isso requer métodos de análise que tenham uma abordagem sistemática, ou seja, se outro pesquisador realizar posteriormente o mesmo estudo usando os mesmos dados, os resultados devem ser semelhantes ou complementares. Portanto, o autor utilizou a variante de Strauss do método de teoria fundamentada (STRAUSS; CORBIN, 1997).

4 ANÁLISE E INTERPRETAÇÃO DOS RESULTADOS

Primeiramente, são apresentados e discutidos os desafios e soluções identificados. Depois é fornecida a estrutura de desafios e soluções que são mapeados entre si. Os resultados incluem os achados específicos para gerenciar a evolução distribuída de software, somados da estrutura conceitual de Ulziit et al. (2015) para manutenção global de software. A intenção deste trabalho é agregar teorias de forma que o conhecimento geral seja somado, e não somente destacar isoladamente os achados específicos deste trabalho.

Como será notado a seguir na seção 4.1, há uma grande quantidade de desafios que não são apenas específicos da evolução distribuída de software, mas também podem ser comuns no desenvolvimento e na manutenção de software global. No escopo deste estudo, é discutido apenas desafios e estratégias de mitigação de evolução de software em contextos globalmente distribuídos.

Por meio de análise qualitativa, o autor observou desafios e estratégias na evolução distribuída de software. O autor identificou que estes são influenciados por alguns fatores com padrões comuns. Esses padrões são sintetizados em grandes categorias, considerando as principais entidades de projetos de software: pessoas, processos, produtos e tecnologia (ESTEVES; PASTOR, 1999; ULZIIT et al., 2015), que emergiram da análise dos dados. Assim, o autor se concentrou nesses aspectos, que podem ser específicos ou amplificados no contexto da evolução distribuída de software (CAMILO et al., 2018), mas analisados pela perspectiva de organizações clientes de sistemas corporativos proprietários. As categorias são definidas da seguinte forma:

- Pessoas: Fatores relacionados a humanos, como habilidades profissionais e experiência. Além disso, habilidades pessoais e questões comportamentais também são compreendidas.
- Processo: Fluxo de atividades seguido pela intenção das pessoas de produzir um resultado desejado. Compreende as atividades envolvidas no gerenciamento de projetos e evolução de software.

- Produto: Sistema de software que está sendo evoluído. Resultado de um conjunto de processos. Fatores de produto são características do sistema.
- Tecnologia: Ferramenta de software ou hardware que sustente ou melhore a eficiência de pessoas, processos e produtos.

As tabelas desta seção são apresentadas da seguinte forma:

- a coluna ”Referência Bibliográfica” referencia os achados do referencial teórico da estrutura conceitual de Ulziit et al. (2015);
- a coluna ”Entrevista” referencia os achados das entrevistas realizadas pelo autor, principal alvo da discussão deste trabalho; e
- a coluna ”Etapa” representa a classificação dos desafios sobre as etapas do fluxo do processo de evolução distribuída de software (CAMILO et al., 2018), sendo elas: mudança (1), diagnóstico (2), especificação (3), desenvolvimento (4), terceiros (5), implantação (6).

4.1 DESAFIOS

Baseado nas dimensões pessoas, processo, produto e tecnologia, derivada por meio da aplicação da teoria fundamentada, são apresentados desafios identificados.

4.1.1. *Fator de pessoas*. Cinco (5) categorias de desafios foram identificadas, a saber, atitude pessoal, interesses conflitantes, habilidades interpessoais, diferenças de idioma, pessoal. A Tabela 7 ilustra a estrutura dos desafios relacionados ao fator de pessoas com base nas referências bibliográficas e entrevistas. Com relação às entrevistas, sete (7) dos doze (12) desafios identificados estão mais fortemente relacionados à evolução em particular. Um (1) novo desafio específico relacionado ao fator de pessoas foi identificado nas entrevistas, a saber, indisponibilidade de usuários para testes devido à rotina das áreas:

- *Indisponibilidade de usuários para testes devido à rotina das áreas*: De acordo com as entrevistas, foi possível identificar que o tamanho e complexidade do sistema corporativo em questão demanda testes que envolvem diversos usuários e departamentos da organização cliente, o que, consequentemente, dificulta a disponibilidade de todos os usuários para validar todos os cenários de testes necessários durante um mesmo período. Primeiramente, cada departamento está

Tabela 7: Desafios - Pessoas.

Código Axial	Código Aberto	Referências Bibliográficas	Entrevista	Etapa
Atitude pessoal	Percepção degradada sobre manutenção / evolução	✓	✓	2,3
	Negligenciamento sobre o valor da documentação	✓	✓	3,4
	Falta de vontade de viajar	✓	•	2,3,4,5,6
	Negligenciamento sobre o valor da manutenção / evolução	✓	✓	3,4
Interesses conflitantes	Solicitações de mudança transitória	✓	✓	1,2,3,4,5,6
	Resistência dos clientes para mudança	✓	✓	3,6
Habilidade interpessoal	Baixa moral do time	✓	•	2,3,4,5,6
Diferenças de idioma	Dificuldade em entender o contexto exato dos e-mails	✓	✓	1,2,3,4,5,6
Pessoal	Indisponibilidade de especialista no domínio	✓	•	2,3,4,5,6
	Falta de analistas de testes no local	✓	•	4,5,6
	Indisponibilidade usuários para testes devido à rotina das áreas	•	✓	3,4,5,6
	Envolvimento de muitas equipes diferentes em poucas tarefas	✓	•	4,5

•Não encontrado na Referência Bibliográfica/Entrevista; ✓Encontrado na Referência Bibliográfica/Entrevista;

envolvido em entregas que, geralmente, são obrigações que têm um prazo determinado. Exemplos são o envio de obrigações para o governo ou da folha de pagamento dos funcionários, entre diversas outras obrigações de responsabilidade cada departamento. Também para os entrevistados, muitas vezes, cenários de testes são lembrados já na execução da fase de testes do projeto, o que custa tempo para os usuários e equipes alocados para essa atividade.

- Etapas 3,4,5,6: De acordo com as entrevstas, foi possível identificar que a indisponibilidade de usuários para testes devido à rotina das áreas afeta algumas etapas (3,4,5,6). Na etapa 3, por vezes, a equipe do projeto não consegue avaliar com exatidão a disponibilidade de usuários para a alocação dos testes. Nas etapas seguintes (4,5,6), a consequência da não exatidão fica clara pela indisponibilidade do usuário para realizar o teste de determinado cenário, que demanda a atuação deste. Essa indisponibilidade se dá, muitas vezes, pela própria rotina de trabalho e entregas do departamento.

4.1.2. *Fator de processo*. Seis (6) categorias de desafios foram identificadas, a saber, gerenciamento da mudança, gerenciamento da qualidade, gerenciamento da comunicação, gerenciamento do escopo, gestão do conhecimento, gerenciamento de riscos. A Tabela 8 ilustra a estrutura dos desafios relacionados ao fator processo com base nas referências bibliográficas e entrevistas. Com relação às entrevistas, treze (13) dos trinta e sete (37) desafios

identificados estão mais fortemente relacionados à evolução em particular. A grande maioria dos desafios é relatada apenas na literatura e não é muito enfatizada nas entrevistas. Quatro (4) novos desafios específicos relacionados ao fator processo foram identificados nas entrevistas, a saber, prazos impostos pelos stakeholders, diferentes projetos sendo realizados paralelamente, excesso de cenários de testes, key users não se lembram de todos cenários de testes:

- *Prazos impostos pelos stakeholders*: De acordo com as entrevistas, foi possível identificar que, como os processos de mudança são baseados nos prazos impostos pelas partes interessadas solicitantes, o desafio está na priorização de implementação das solicitações de mudanças dentro do prazo imposto pelos solicitantes. De acordo com os entrevistados, muitas vezes, os prazos coincidem e há necessidade de priorizar uma solicitação de mudança governamental sobre uma solicitação de mudança feita para a melhoria de um processo organizacional. Há casos em que desenvolvimentos de um projeto em execução devem ser desfeitos ou hibernados, para que um outro projeto de maior prioridade seja iniciado, por exemplo.

  - Etapas 1,2,3,4,5,6: De acordo com as entrevistas, foi possível identificar que o prazo imposto pelos stakeholders afeta todas as etapas, justamente pelo fato de que o projeto tem que ser adequado ao prazo proposto. O atraso na conclusão de determinada etapa, pode demandar maior agilidade na execução das etapas posteriores, por exemplo. "É um efeito dominó", avaliou um dos entrevistados.

- *Diferentes projetos sendo realizados paralelamente*: De acordo com as entrevistas, foi possível identificar que a realização de diferentes projetos em paralelo se apresentou como um desafio. O principal fator é que a integração dos diferentes módulos – financeiro, contábil, produção, logística, e assim por diante – dificulta a implementação de determinadas mudanças, pois projetos diferentes podem em algum momento utilizar um mesmo processo como base, o que pode gerar concorrência. Por fim, quanto mais complexo o projeto, maior o impacto em diferentes processos de negócios, e consequente possibilidade de concorrência.

  - Etapas 1,2,3,4,5,6: De acordo com as entrevistas, foi possível identificar que o prazo imposto pelos stakeholders afeta todas as etapas. Nas etapas iniciais (1,2,3) a avaliação de impacto entre projetos pode identificar préviamente a execução de dois ou mais projetos paralelamente, mas essas etapas ainda envolvem indiretamente muita teoria. Nas etapas seguintes (4,5,6) há maior

Tabela 8: Desafios - Processo.

Código Axial	Código Aberto	Referências Bibliográficas	Entrevista	Etapa
Gerenciamento da mudança	Análise de impacto deficiente no pedido de mudança	✓	✓	2,3
	Grande número de solicitações de mudança	✓	✓	1
	Solicitações de mudança transitórias	✓	✓	1,2,3,4,5,6
	Prazos impostos pelos stakeholders	•	✓	1,2,3,4,5,6
	Pedido de alteração pouco claro	✓	✓	1,2,3
	Relato do problema pouco claro	✓	•	2
	Gerenciamento inadequado de solicitações de mudança	✓	•	2,3
	Diferentes projetos sendo realizados paralelamente	•	✓	1,2,3,4,5,6
Gerenciamento da Qualidade	Falta de suítes de testes de regressão	✓	•	4,5
	Falta de procedimentos de garantia de qualidade no processo de manutenção / evolução	✓	•	4,5
	Falta de verificação no hardware	✓	•	2,3
	Problemas de integração	✓	✓	4,5,6
	Testes pouco frequentes antes da implantação	✓	•	4,5
	Muito esforço devido ao duplo teste e inspeção	✓	•	4,5
	Métricas inconsistentes entre as organizações	✓	•	2,3,4,5,6
	Nenhum padrão de documentação	✓	•	3,4,5
	Nenhum padrão de codificação	✓	•	4,5
	Excesso de cenários de testes	•	✓	4,5,6
	Key users não se lembram de todos cenários de testes	•	✓	2,3
Gerenciamento da comunicação	Visão limitada e envolvimento do solicitante durante a mudança	✓	✓	2,3,4,5,6
	Pouco contato com um desenvolvedor original do software	✓	•	3,4,5
Gerenciamento do escopo	Distribuição de tarefas pouco clara	✓	•	4,5,6
	Distribuição errada de tarefas	✓	•	4,5,6
	Falta de controle de tarefas	✓	•	4,5,6
	Sobrecarga no controle de tarefas	✓	•	4,5,6
	Alta pressão de trabalho	✓	•	4,5,6
	Horas de trabalho prolongadas devido a manutenção / evolução de emergência	✓	✓	4,5,6
	Indisponibilidade de pessoal em caso de emergência	✓	✓	4,5,6
Gestão do conhecimento	Falta de documentação para projeto e código fonte	✓	•	3,4,5,6
	Conhecimento tácito de domínio técnico	✓	•	2,3,4,5,6
	Conhecimento tácito do domínio de negócios	✓	•	2,3,4,5,6
	Falta de repositório de conhecimento	✓	•	2,3,4,5,6
	Tempo limitado para aprender	✓	•	2,3,4,5
Gerenciamento de riscos	Falta de análise de impacto	✓	✓	2,3
	Ignorância de acompanhar atualizações de soluções de terceiros	✓	•	1,2
	Manutenção / evolução inesperada de emergência	✓	•	1
	Divulgação de segurança de dados durante a mudança	✓	•	2,3,4,5,6

•Não encontrado na Referência Bibliográfica/Entrevista; ✓Encontrado na Referência Bibliográfica/Entrevista;

possibilidade de surgir problemas, pois, na prática, objetos de desenvolvimento não identificados nas primeiras etapas, podem sofrer concorrência de utilização, e criar discussões sobre qual decisão tomar, qual projeto priorizar, qual impacto avaliar.

- *Excesso de cenários de testes*: De acordo com as entrevistas, foi possível identificar que o tamanho e complexidade do sistema corporativo em questão demandam testes que envolvem diversos cenários da organização cliente, o que, consequentemente, dificulta a realização de todos os cenários necessários dentro do prazo imposto pelas partes interessadas para a conclusão do projeto. Geralmente, por motivos de rotinas da área, os usuários alocados na fase de teste não dispõe do tempo necessário para a realização de todos os cenários de testes.

  - Etapas 4,5,6: De acordo com as entrevistas, foi possível identificar que o excesso de cenários de testes afeta algumas etapas (4,5,6). Na prática, a realização e validação dos cenários pela equipe do projeto envolve muitos cenários de teste, o que demanda grande envolvimento dos dos usuários, prejudicando tanto o andamento do próprio projeto quanto a rotina de trabalho das áreas envolvidas.

- *Key users não se lembram de todos cenários de testes*: De acordo com as entrevistas, foi possível identificar que o tamanho e complexidade do sistema corporativo em questão demanda diversos cenários de testes para garantir que a solicitação de mudança seja atendida. Esses cenários de testes geralmente envolvem processos que são integrados com outros processos, portanto, a quantidade de cenários pode ser relativamente grande. No entanto, os usuários-chave envolvidos no projeto nem sempre conseguem identificar todos os cenários possíveis. A descoberta de que determinado cenário não foi testado se dá durante uma situação real, onde, provavelmente, a organização terá que solucionar o problema sem tempo hábil, podendo trazer prejuízos para esta.

  - Etapas 2,3,4: De acordo com as entrevistas, foi possível identificar que os key users não se lembrarem de todos os cenários de testes afeta algumas etapas (2,3,4). Nas etapas 2 e 3, dificilmente a equipe consegue lembrar cem por cento dos cenários de testes a serem realizados, o que pode ser detectado em etapas posteriores (4), ou após o fim do projeto.

Tabela 9: Desafios - Produto.

Código Axial	Código Aberto	Referências Bibliográficas	Entrevista	Etapa
Independência de terceiros	Uso de componentes de terceiros	✓	•	1,2,3,4,5,6
	Atualizações freqüentes em tecnologias usadas no desenvolvimento	✓	•	1
	Vulnerabilidade da divulgação de informações confidenciais de negócios	✓	•	5
	Mudanças em componentes de terceiros	✓	✓	1,2,3,4,5,6
	Excesso de mudanças em componentes de terceiros	•	✓	1,2,3,4,5,6
Tamanho e complexidade do software	Várias solicitações de mudança	✓	✓	1
	Pedido de alteração complexo	✓	✓	1,2,3
	A necessidade de extensa documentação	✓	✓	1,2,3,4,5,6
	Limitação do acesso ao código fonte	•	✓	3,4,5,6
	Dependência entre os diversos módulos do software	•	✓	2,3,4,5,6

•Não encontrado na Referência Bibliográfica/Entrevista; ✓Encontrado na Referência Bibliográfica/Entrevista;

4.1.3. *Fator de produto*. Duas (2) categorias de desafios foram identificadas, a saber, independência de terceiros e tamanho e complexidade do software. A Tabela 9 ilustra a estrutura dos desafios relacionados ao fator produto com base nas referências bibliográficas e entrevistas. Com relação às entrevistas, sete (7) dos dez (10) desafios identificados estão mais fortemente relacionados à evolução em particular. Três (3) novos desafios específicos relacionados ao fator produto foram identificados nas entrevistas, a saber, excesso de mudanças em componentes de terceiros, limitação do acesso ao código fonte, dependência entre os diversos módulos do software:

- *Excesso de mudanças em componentes de terceiros*: De acordo com as entrevistas, foi possível verificar que os entrevistados identificaram um excesso de mudanças em componentes de terceiros. Essas mudanças somam melhorias nos componentes, realizadas pelos próprios terceiros, como também melhorias solicitadas por outras partes interessadas envolvidas com os terceiros. Essas mudanças podem, ou não, recair sobre as organizações clientes que utilizam esses componentes.

  – Etapas 1,2,3,4,5,6: De acordo com as entrevistas, foi possível identificar que o excesso de mudanças em componentes de terceiros afeta todas as etapas (1,2,3,4,5,6). Quando uma mudança em componentes é implantada, a organização cliente precisa garantir que essa mudança não afetará as funcionalidades existentes no sistema corporativo, o que culmina na validação dos cenários de negócios existentes na organização. Quando componentes de terceiros tem um excesso de mudança de funcionalidades, a organização

cliente fica dependente da validação dos cenários de negócios, prejudicando a rotina de trabalho dos departamentos. O fato se dá quando existe um projeto sendo executado em um componente de terceiro, e o terceiro, por sua vez, realiza uma nova mudança antes mesmo do projeto ser finalizado.

- *Limitação do acesso ao código fonte por terceiros*: De acordo com as entrevistas, foi possível verificar que há limitação na modificação do código fonte do sistema corporativo por partes interessadas que não a produtora do software. Essa limitação faz com que a organização cliente não tenha a autonomia que deseja para implementar todas as solicitações de mudança que deseja no sistema corporativo. Como consequência, a organização cliente acaba recorrendo à sistemas paralelos, parceiros externos ou até a produtora do software para suprir essas necessidades.

  - Etapas 3,4,5,6: De acordo com as entrevistas, foi possível identificar que a limitação excesso de mudanças em componentes de terceiros afeta as etapas (3,4,5,6). Na etapa 3, quando a especificação é feita, a equipe pode conseguir avaliar até onde a atuação é possível dentro do código fonte do sistema corporativo. Na prática, nas etapas 4, 5 e 6, ajustes geralmente são necessários para o sucesso da implementação. Portanto, a descoberta da limitação de abetura do código fonte nessa etapa, pode resultar em prejuízos de cronograma e financeiros, provavelmente, por exigir maior tempo de análise de uma solução de contorno, ou no envolvimento de parceiros externos ou da produtora do software.

- *Dependência entre os diversos módulos do software*: De acordo com as entrevistas, foi possível verificar que há uma grande dependência entre os módulos do sistema corporativo. Isso se deve, pois, de fato, há integração entre os diversos processos de negócio da organização dentro do software. Essa dependência gera desafios para o fato de que as mudanças estão sob a perspectiva de todos os módulos, com relação à impactos e validações.

  - Etapas 2,3,4,5,6: De acordo com as entrevistas, foi possível identificar que a dependência entre os diversos módulos do software afeta as etapas (2,3,4,5,6). Nas etapas 2 e 3, a equipe geralmente avalia impactos de forma um pouco mais genérica. Nas etapas 4, 5 e 6, geralmente surgem problemas relacionados ao impacto de um projeto sobre uma funcionalidade existente de um outro módulo.

Tabela 10: Desafios - Tecnologia.

Código Axial	Código Aberto	Referências Bibliográficas	Entrevista	Etapa
Suporte de hardware	Ambiente de hardware não qualificado	✓	•	2,3,4,5,6
	Falhas de software devido a hardware inadequado / defeituoso	✓	•	4,5,6
Suporte de software	Falta de ferramentas de apoio	✓	•	3,4,5,6
Ambiente de teste	Limitação na reprodução do cenário anômalo	✓	✓	4,5
	Diferença entre ambiente de desenvolvimento e de produção	✓	✓	4,5

•Não encontrado na Referência Bibliográfica/Entrevista; ✓Encontrado na Referência Bibliográfica/Entrevista;

4.1.4. *Fator de tecnologia*. Três (3) categorias de desafios foram identificadas, a saber, suporte de hardware, suporte de software e ambiente de teste. A Tabela 10 ilustra a estrutura dos desafios relacionados ao fator tecnologia com base nas referências bibliográficas e entrevistas. Com relação às entrevistas, dois (2) dos cinco (5) desafios identificados estão mais fortemente relacionados à evolução em particular. Não foram identificados novos desafios específicos relacionados ao fator tecnologia.

4.2 SOLUÇÕES

Semelhante aos desafios, baseado na estrutura (pessoas, processo, produto e tecnologia) derivada por meio da aplicação da teoria fundamentada, são apresentadas as soluções identificadas.

4.2.1. *Fator de pessoas*. Três (3) categorias de soluções foram identificadas, a saber, envolvimento, enriquecimento social e pessoal. A Tabela 11 fornece uma visão geral das soluções identificadas. As soluções foram obtidas principalmente da literatura. Os aspectos sociais foram enfatizados em relação à interação com a equipe, ao enriquecimento social e à facilitação da interação social por meio de ações relacionadas à equipe. Com relação às entrevistas, seis (6) das doze (12) soluções identificadas estão mais fortemente relacionadas à evolução em particular. Três (3) novas soluções específicas relacionadas ao fator pessoas foram identificadas nas entrevistas, a saber, monitorar e priozar as solicitações de mudanças constantemente, realizar ligações e videoconferências com padronização de atas para todas as definições, formalizar com toda a equipe o melhor período para a realização dos testes:

- *Monitorar e priozar as solicitações de mudanças constantemente*: De acordo com as entrevistas, foi possível verificar que há uma necessidade de monitorar e priorizar constantemente solicitações de mudança, de forma que a organização cliente tenha

Tabela 11: Soluções - Pessoas.

Código Axial	Código Aberto	Referências Bibliográficas	Entrevista
Envolvimento	Aumentar o envolvimento do solicitante na manutenção / evolução	✓	✓
	Envolver os solicitantes na especificação do caso de teste para explorar falhas	✓	✓
	Monitorar e priozar as solicitações de mudanças constantemente	•	✓
	Realizar ligações e videoconferências com padronização de atas para todas as definições	•	✓
Enriquecimento Social	Proatividade na comunicação com colegas	✓	•
Pessoal	Construindo uma pequena equipe para ser eficaz	✓	•
	Implantar uma equipe de suporte multifuncional com software e conhecimento de hardware	✓	•
	Ter pelo menos uma pessoa qualificada para ajudar a compartilhar conhecimento	✓	•
	Ter um representante no local com conhecimento de diferentes especialistas técnicos	✓	✓
	Formalizar com toda a equipe o melhor período para a realização dos testes	•	✓
	Envolver a troca de tarefas entre o mantenedor e o desenvolvedor	✓	•
	Atendimento de suporte tanto remoto quanto nativo	✓	•

•Não encontrado na Referência Bibliográfica/Entrevista; ✓Encontrado na Referência Bibliográfica/Entrevista;

tempo hábil para manter um determinado nível de prontidão para analisar, decidir, priorizar e atuar sobre estas. Essa atuação deve estar aberta, por meio de um membro ou equipe responsável, para monitorar todas as partes externas que podem, de uma forma, despertar uma mudança no sistema.

- *Realizar ligações e videoconferências com padronização de atas para todas as definições*: De acordo com as entrevistas, foi possível verificar que há uma necessidade de padronizar a realização de ligações e vídeoconferências para o alinhamento de assuntos que podem causar dificuldade no entendimento do contexto exato dos emails. Como atividade complementar, é sugerida a padronização de atas para todas as definições, sem exceção, de forma que toda e qualquer decisão não seja interpretada de diferentes formas, o que pode impactar fortemente nos prazos e escopo do projeto.

- *Formalizar com toda a equipe o melhor período para a realização de testes*: De acordo com as entrevistas, foi possível verificar que há uma necessidade de padronizar a definição do melhor período para a realização de testes com a equipe do projeto. Com a grande abrangência de envolvidos em projetos de sistemas

corporativos, a necessidade de que todas as partes interessadas validem os cenários de testes, dentro do prazo proposto, é dita como essencial. Essa formalização se deve principalmente pelo tamanho, complexidade e integração dos módulos do sistema corporativo, o que requer grande responsabilidade pela equipe do projeto.

4.2.2. *Fator de processo*. Cinco (5) categorias de soluções foram identificadas, a saber, gerenciamento da qualidade, gerenciameno do escopo, gerenciamento da mudança, gerenciamento da comunicação e gerenciamento do risco. A Tabela 12 fornece uma visão geral das soluções identificadas. Com relação aos processos, foi identificado o maior número de propostas de solução. As soluções foram obtidas principalmente da literatura. Os aspectos de processo foram enfatizados em relação a qualidade, escopo, mudança e comunicação por meio de ações relacionadas à gerenciamento. Com relação às entrevistas, dezesseis (16) das trinta e seis (36) soluções identificadas estão mais fortemente relacionadas à evolução em particular. Quatro (4) novas soluções específicas relacionadas ao fator processos foram identificadas nas entrevistas, a saber, criação de uma base de conhecimento exclusivamente para testes, analisando solicitações de mudança regularmente, reavaliando as priodades dos projetos ativos e, por fim, definindo o escopo e tratar como futura solicitação de mudança:

- *Criação de uma base de conhecimento exclusivamente para testes*: De acordo com as entrevistas, foi possível verificar que há uma necessidade de criação de uma base de conhecimento para o registro de todos os cenários de testes possíveis para servir como referência para toda a equipe do projeto. Essa tratativa é necessária justamente pela dependência e integração existente entre os módulos do sistema corporativo. Dessa forma, a possibilidade de cobrir e validar todos os cenários e seus impactos se torna maior.

- *Reavaliar as priodades dos projetos ativos*: De acordo com as entrevistas, foi possível identificar que há uma necessidade de reavaliar a prioridade de projetos ativos frequentemente, pois o fato de a organização cliente ter diferentes projetos sendo realizados paralelamente pode gerar concorrência de projetos à medida que estes são executados. Portanto, cabe à equipe do projeto a comunicação constante com as equipes de outros projetos sobre o avanço, ou não, de etapas do projeto em questão.

- *Definir o escopo e tratar como futura solicitação de mudança*: De acordo com as entrevistas, foi possível identificar que há uma necessidade de evitar alteração excessiva de escopo. Com solicitações de mudança que podem ser alteradas durante

Tabela 12: Soluções - Processo.

Código Axial	Código Aberto	Referências Bibliográficas	Entrevista
Gerenciamento da Qualidade	Ter compartilhado o mesmo ambiente de teste em todos os sites	✓	✓
	Apresentando as métricas de processo	✓	•
	Automatizando casos de teste	✓	•
	Revisão por pares na inspeção de código	✓	•
	Testes rigorosos de remendos	✓	•
	Uso do teste de aceitação do cliente	✓	•
	Inspeção de teste por diferentes membros	✓	✓
	Desenvolvedores e testadores trabalhando juntos	✓	✓
	Aderir às melhores práticas e padrões	✓	✓
	Concentrar-se no design de manutenção / evolução desde o início	✓	✓
	Definindo sucesso e fracasso de manutenção / evolução na presença de todas as partes interessadas	✓	•
	Medindo a qualidade do processo de manutenção / evolução	✓	•
	Criação de uma base de conhecimento exclusivamente para testes	•	✓
Gerenciamento do escopo	Atribuindo um determinado módulo de software a uma determinada equipe	✓	✓
	Manter tarefas relacionadas em um único site	✓	✓
	Manter um equilíbrio entre a especialização de tarefas e a diversidade de tarefas	✓	•
	Expondo novos funcionários para variedade de tarefas para aprendizagem eficaz	✓	•
	Distribuição de tarefas com base na especialização da equipe	✓	•
	Distribuição de tarefas baseada em padrões	✓	•
	Pessoal designado para versões específicas do produto	✓	•
Gerenciamento da mudança	Tendo gerenciamento central de solicitações de mudança	✓	✓
	Métricas de confiabilidade de software no código-fonte	✓	•
	Processos definidos para enviar uma solicitação de mudança	✓	✓
	Analisando solicitações de mudança regularmente	•	✓
	Reavaliar as priodades dos projetos ativos	•	✓
	Definir o escopo e tratar como futura solicitação de mudança	•	✓
Gerenciamento da comunicação	Introduzindo listas malignas	✓	•
	Construindo comunidade de clientes virtuais	✓	•
	Fornecendo possibilidade de suporte remoto através de um terminal móvel	✓	•
	Minutas da reunião para apoiar a conferência	✓	✓
	Único ponto de comunicação nas equipes	✓	•
	Métodos de comunicação baseados em prioridades	✓	•
Gerenciamento do risco	Considerando datas específicas nacionais no plano	✓	✓
	Planejando o procedimento de manutenção / evolução de emergência mais cedo	✓	✓
	Ter uma lista de verificação de riscos	✓	•
	Aviso prévio antes de desligamento do trabalho	✓	•

•Não encontrado na Referência Bibliográfica/Entrevista; ✓Encontrado na Referência Bibliográfica/Entrevista;

o trajeto do projeto, é necessário definir se cabe ao projeto a conclusão do escopo atual e, posteriormente, a tratativa da alteração de escopo como futura solicitação de mudança; ou se cabe ao projeto a alteração do escopo atual para que a alteração da solicitação de mudança seja implementada ainda neste.

4.2.3. *Fator de produto*. Uma (1) categoria de solução foi identificada, a saber, documentação. A Tabela 13 fornece uma visão geral das soluções identificadas. As soluções foram obtidas principalmente da literatura. Os aspectos de produto foram enfatizados em relação à documentação de diferentes tipos. Com relação às entrevistas, sete (7) das nove (9) soluções identificadas estão mais fortemente relacionadas à evolução em particular. Três (3) novas soluções foram adicionadas por meio de entrevistas, a saber, negociação de periodicidade de implantações, abertura de pontos no código fonte para desenvolvimento por terceiros entre trechos de código fechado, envolver um especialista de cada módulo para análise de impacto em cada projeto:

- *Negociação de periodicidade de implantações*: De acordo com as entrevistas, foi possível verificar que há uma necessidade de avaliar com terceiros a frequência de adição de novas funcionalidades, de forma que a organização cliente evite investir grande parte do seu tempo validando a grande quantidade de cenários de testes que é demandada pela adição de novas funcionalidades. De acordo com os entrevistados, qualquer adição de funcionalidade em componentes de terceiros exige a validação de todos os cenários de testes relacionados, para garantir que a adição desta nova funcionalidade não crie falhas no ambiente do sistema corporativo.

- *Abertura de pontos no código fonte para desenvolvimento por terceiros entre trechos de código fechado*: De acordo com as entrevistas, foi possível verificar que, como há limitação de acesso ao código fonte, há uma necessidade de abertura de trechos do código fonte para alteração por terceiros. Essa abertura possibilita uma maior liberdade da organização cliente para a implementação de funcionalidades sem a total depedência de parceiros externos ou da produtora do software.

- *Envolver um especialista de cada módulo para análise de impacto em cada projeto*: De acordo com as entrevistas, foi possível verificar que, como há dependência entre os diversos módulos do software, há uma necessidade de que cada especialista realize a análise de impacto pela perspectiva do módulo que utiliza. Isso se faz necessário pois, geralmente, o especialista de um módulo não

Tabela 13: Soluções - Produto.

Código Axial	Código Aberto	Referências Bibliográficas	Entrevista
Documentação	Preparando a documentação em diferentes idiomas	✓	✓
	Ter orientações e manuais claros para os processos de manutenção / evolução	✓	✓
	Contrato de acordo de confidencialidade	✓	•
	Documentação atualizada sobre componentes de terceiros usados	✓	✓
	Negociação de periodicidade de implantações	•	✓
	Documentação de todos os projetos e codificação de alterações	✓	✓
	Documento claro do acordo de nível de serviço	✓	•
	Abertura de pontos no código fonte para desenvolvimento por terceiros entre trechos de código fechado	•	✓
	Envolver um especialista de cada módulo para análise de impacto em cada projeto	•	✓

•Não encontrado na Referência Bibliográfica/Entrevista; ✓Encontrado na Referência Bibliográfica/Entrevista;

Tabela 14: Soluções - Tecnologia.

Código Axial	Código Aberto	Referências Bibliográficas	Entrevista
Automação	Automatização de tarefas manuais de rotina	✓	✓
	Detectando tarefas automatizáveis	✓	•
	Desenvolvendo uma ferramenta de atualização automática para economizar custos	✓	•
	Usando tecnologia para automatizar tarefas	✓	•
	Alternância de testes manuais e automatizados	✓	✓

•Não encontrado na Referência Bibliográfica/Entrevista; ✓Encontrado na Referência Bibliográfica/Entrevista;

tem a percepção de que talvez uma mudança possa impactar na funcionalidade de um outro módulo.

4.2.4. *Fator de tecnologia*. Uma (1) categoria de solução foi identificada, a saber, automação. A Tabela 14 fornece uma visão geral das soluções identificadas. As soluções foram obtidas principalmente da literatura. Os aspectos de automação foram enfatizados por meio de ações relacionadas à tarefas e testes. Com relação às entrevistas, duas (2) das cinco (5) soluções identificadas estão mais fortemente relacionadas à evolução em particular. Nenhuma solução foi adicionada através das entrevistas.

4.3 MAPEAMENTO DE DESAFIOS E SOLUÇÕES

Após apresentar a relação de desafios e soluções, foram mapeadas as soluções para os desafios e as descobertas foram realizadas com base na própria interpretação do autor. O autor indicou se a ligação foi realizada com base nas referencias bibliográficas ou nas entrevistas.

Como a ligação na maioria dos casos foi realizada com base na interpretação, os resultados desta seção podem ser interpretados como proposições a serem avaliadas.

5.3.1. *Fator de pessoas*. As soluções em relação aos desafios do fator pessoas foram mapeados e são apresentadas na Tabela 15. Três (3) mapeamentos não foram detectados na literatura relacionada à evolução distribuída de software. Cinco (5) dos onze (11) mapeamentos identificados estão mais fortemente relacionados à evolução em particular. Três (3) novos mapeamentos foram adicionados por meio das entrevistas. Para solicitações de mudança transitória, é sugerido monitorar e priozar solicitações de mudança constantemente. Para a dificuldade em entender o contexto exato dos e-mails, é sugerido realizar ligações e videoconferências com padronização de atas para todas as definições. Para a indisponibilidade de pessoal para testes devido à rotina das áreas, é sugerido formalizar com toda a equipe o melhor período para a realização dos testes.

5.3.2. *Fator de processo*. As soluções em relação aos desafios do fator processo foram mapeados e são apresentadas na Tabela 16. Quatro (4) mapeamentos não foram detectados na literatura relacionada à evolução distribuída de software. Vinte e dois (22) dos trinta e quatro (34) mapeamentos identificados estão mais fortemente relacionados à evolução em particular. Seis (6) novos mapeamentos foram adicionados por meio das entrevistas. Para a dificuldade em entender o contexto exato dos e-mails, é sugerido realizar ligações e videoconferências com padronização de atas para todas as definições. Para a indisponibilidade de pessoal para testes devido à rotina das áreas, é sugerido formalizar com toda a equipe o melhor período para a realização dos testes. Para o grande número de solicitações de mudança, é sugerido monitorar e priozar as solicitações de mudanças constantemente. Para as solicitações de mudança voláteis, é sugerido definir o escopo e tratar como futura solicitação de mudança. Para os prazos impostos pelos stakeholders, é sugerido reavaliar as priodades dos projetos ativos. Para diferentes projetos sendo realizados paralelamente, é sugerido reavaliar as priodades dos projetos ativos.

5.3.3. *Fator de produto*. As soluções em relação aos desafios do fator produto foram mapeados e são apresentadas na Tabela 17. Um (1) mapeamento não foi detectado na literatura relacionada à evolução distribuída de software. Seis (6) dos nove (9) mapeamentos identificados estão mais fortemente relacionados à evolução em particular. Três (3) novos mapeamentos foram adicionados por meio das entrevistas. Para o excesso de mudanças em componentes de terceiros, é sugerida a negociação de periodicidade de releases. Para a limitação do acesso ao código fonte por terceiros, é sugerida a abertura de pontos no código fonte para desenvolvimento por terceiros entre trechos de código fechado. Para várias solicitações de mudança, é sugerido o monitoramento e priorização das solicitações de mudança.

Tabela 15: Desafios x Soluções - Pessoas.

Código Axial	Código Aberto	Referências Bibliográficas	Entrevista	Fase
Atitude pessoal				
Percepção degradada sobre manutenção / evolução	•	•	•	2,3
Negligenciamento sobre o valor da documentação	Ter diretrizes e manuais claros para todos os processos de manutenção / evolução	✓	✓	3,4
Falta de vontade de viajar	Membros da equipe altamente motivados por uma boa liderança	✓	•	2,3,4,5,6
Negligenciando valor de manutenção / evolução	•	•	•	3,4
Interesses conflitantes				
Solicitações de mudança transitória	Definir o escopo e tratar como futura solicitação de mudança	•	✓	1,2,3,4,5,6
Resistência dos clientes para atualizar	•	•	•	3,6
Habilidade interpessoal				
Baixa moral de um time	Membros da equipe altamente motivados por uma boa liderança	✓	•	2,3,4,5,6
Diferenças de idioma				
Dificuldade em entender o contexto exato dos e-mails	Realizar ligações e video conferências com padronização de atas para todas as definições	•	✓	1,2,3,4,5,6
Pessoal				
Indisponibilidade do especialista no domínio	Implantando uma equipe de suporte multifuncional com conhecimento de software e hardware	✓	•	2,3,4,5,6
	Ter pelo menos uma pessoa qualificada para ajudar a compartilhar conhecimento	✓	•	2,3,4,5,6
	Plataforma colaborativa para gerenciar e compartilhar conhecimento de domínio	✓	•	2,3,4,5,6
Falta de pessoal de testes no local	Ter um representante no local com conhecimento de diferentes especialistas técnicos	✓	✓	4,5,6
Indisponibilidade de pessoal para testes devido à rotina das áreas	Formalizar com toda a equipe o melhor período para a realização dos testes	•	✓	4,5,6
Envolvimento de muitas equipes diferentes em poucas tarefas	Distribuição de tarefas com base na especialização da equipe	✓	✓	4,5
	Distribuição de tarefas baseada em padrões	✓	•	4,5

•Não encontrado na Referência Bibliográfica/Entrevista; ✓Encontrado na Referência Bibliográfica/Entrevista;

Tabela 16: Desafios x Soluções - Processo.

Código Axial	Código Aberto	Referências Bibliográficas	Entrevista	Fase
Gerenciamento de mudança				
Análise de impacto deficiente do pedido de alteração	Métricas de estabilidade de software no código fonte tocadas	✓	•	2,3
Grande número de solicitações de mudança	Monitorar e priozar as solicitações de mudanças constantemente	•	✓	1
Solicitações de mudança voláteis	Definir o escopo e tratar como futura solicitação de mudança	•	✓	1,2,3,4,5,6
Prazos impostos pelos stakeholders	Reavaliar as priodades dos projetos ativos	•	✓	1,2,3,4,5,6
Pedido de alteração pouco claro	Processos definidos para enviar uma solicitação de mudança	✓	✓	1,2,3
Relatório de emissão pouco claro	Processos definidos para enviar uma solicitação de mudança	✓	✓	2
Gerenciamento inadequado de solicitações de mudança	Tendo gerenciamento central de solicitações de mudança	✓	✓	2,3
Diferentes projetos sendo realizados paralelamente	Reavaliar as priodades dos projetos ativos	•	✓	1,2,3,4,5,6
Gestão da Qualidade				
Falta de suítes de testes de regressão	Uso do teste de aceitação do cliente	✓	•	4,5
Falta de procedimentos de garantia de qualidade no processo de manutenção / evolução	Apresentando a revisão de gate e as métricas de processo	✓	•	4,5
Falta de verificação no hardware	•	•	•	2,3
Problemas de integração	Desenvolvedores e testadores trabalhando juntos	✓	✓	4,5,6
Testes pouco frequentes antes de implantar patches	Desenvolvedores e testadores trabalhando juntos	✓	✓	4,5
Muito esforço devido ao duplo teste / trabalho de inspeção	Automatizando casos de teste	✓	•	4,5
	Automatização de tarefas manuais de rotina	✓	•	4,5
	Usando tecnologia para automatizar tarefas	✓	•	4,5
Métricas inconsistentes entre as organizações	Definindo sucesso e fracasso de manutenção / evolução na presença de todas as partes interessadas	✓	•	2,3,4,5,6
	Medindo a qualidade do processo de manutenção / evolução	✓	•	2,3,4,5,6
	Medida realista do progresso, dando visibilidade imparcial e imparcial sobre seu progresso	✓	•	2,3,4,5,6

•Não encontrado na Referência Bibliográfica/Entrevista; ✓Encontrado na Referência Bibliográfica/Entrevista;

Código Axial	Código Aberto	Referências Bibliográficas	Entrevista	Fase
Nenhum padrão de documentação	Documento claro do acordo de nível de serviço	✓	•	3,4,5
	Focando na padronização	✓	•	3,4,5
Nenhum padrão de codificação	Aderir às melhores práticas e padrões	✓	✓	4,5
	Usando processos e ferramentas consistentes em todos os sites	✓	•	4,5
Excesso de cenários de testes	Criação de uma base de conhecimento exclusivamente para testes	•	✓	4,5,6
Key users não se lembram de todos os cenários de testes	Criação de uma base de conhecimento exclusivamente para testes	•	✓	4,5,6
Nenhum padrão de design uniforme	Focando na padronização	✓	•	3,4,5
	Concentrar-se no design de manutenção / evolução desde o início	✓	•	3,4,5
Gestão de comunicação				
Visão limitada e envolvimento do cliente durante a mudança de software	Aumentando o envolvimento do cliente na manutenção / evolução	✓	✓	2,3,4,5,6
Nenhum contato com um desenvolvedor original	Envolvendo os clientes na especificação de casos de teste para explorar falhas	✓	✓	2,3,5
Gerenciamento do escopo				
Distribuição de tarefas pouco clara	Atribuindo um determinado módulo de software a uma determinada equipe para qualquer alteração	✓	✓	4,5,6
	Usando o software de gerenciamento de fluxo de trabalho baseado na web	✓	✓	4,5,6
Distribuição errada de tarefas	Manter um equilíbrio entre a especialização de tarefas e a diversidade de tarefas	✓	•	4,5,6
	Uso de software de gerenciamento de fluxo de trabalho baseado na web	✓	•	4,5,6
Falta de controle de tarefas	Uso de software de gerenciamento de fluxo de trabalho baseado na Web	✓	•	4,5,6
Sobrecarga no controle de tarefas	Uso de software de gerenciamento de fluxo de trabalho baseado na Web	✓	•	4,5,6

•Não encontrado na Referência Bibliográfica/Entrevista; ✓Encontrado na Referência Bibliográfica/Entrevista;

Código Axial	Código Aberto	Referências Bibliográficas	Entrevista	Fase
Alta pressão de trabalho	Tendo o gerenciamento de problemas	✓	•	4,5,6
	Planejando o procedimento de manutenção / evolução de emergência mais cedo	✓	✓	4,5,6
Horas de trabalho prolongadas devido a manutenção / evolução de emergência	•	•	•	4,5,6
Indisponibilidade de pessoal em caso de emergência	Planejando o procedimento de manutenção / evolução de emergência mais cedo	✓	✓	4,5,6
Gestão do conhecimento				
Falta de documentação para projeto e código fonte	Ter diretrizes e manuais claros para todos os processos de manutenção / evolução	✓	✓	3,4,5,6
Conhecimento tácito de domínio técnico e empresarial	Treinamento formal ou atividades informais de troca de conhecimento	✓	✓	2,3,4,5,6
	Visita no local regular	✓	•	2,3,4,5,6
Falta de repositório de conhecimento	Plataforma colaborativa para gerenciar e compartilhar conhecimento de domínio	✓	•	2,3,4,5,6
Tempo limitado para aprender	Treinamento formal ou atividades informais de troca de conhecimento	✓	•	2,3,4,5
	Sistema de tutoria para entrada de pessoal em um projeto	✓	•	2,3,4,5
Gerenciamento de riscos				
Falta de análise de impacto	Concentrar-se no design de manutenção / evolução desde o início	✓	✓	2,3
Ignorância de acompanhar atualizações de soluções de terceiros	Documentação atualizada sobre componentes de terceiros usados	✓	✓	1,2
Manutenção / evolução inesperada de emergência	Fornecendo possibilidade de suporte remoto através de um terminal móvel ou agente	✓	•	1,2
	Planejando o procedimento de manutenção / evolução de emergência mais cedo	✓	•	1,2
Divulgação de segurança de dados durante a mudança	•	•	•	2,3,4,5,6

•Não encontrado na Referência Bibliográfica/Entrevista; ✓Encontrado na Referência Bibliográfica/Entrevista;

Tabela 17: Desafios x Soluções - Produto.

Código Axial	Código Aberto	Referências Bibliográficas	Entrevista	Fase
Independência de terceiros				
Uso de componentes de terceiros	Documentação atualizada sobre componentes de terceiros utilizados	✓	✓	1,2,3,4,5,6
Atualizações frequentes em tecnologias usadas no desenvolvimento	Documentação atualizada sobre componentes de terceiros usados	✓	•	1
Vulnerabilidade da divulgação de informações confidenciais de negócios	Contrato por acordo de não divulgação	✓	✓	5
Mudanças em componentes de terceiros	Documentação atualizada sobre componentes de terceiros usados	✓	✓	1,2,3,4,5,6
Excesso de mudanças em componentes de terceiros	Negociação de periodicidade de releases	•	✓	2,3,4,5,6
Limitação do acesso ao código fonte por terceiros	Abertura de pontos no código fonte para desenvolvimentopor terceiros entre trechos de código fechado	•	✓	2,3,4,5,6
Tamanho e complexidade				
Várias solicitações de mudança	Monitoramento e priorização das solicitações de mudança	•	✓	1
Pedido de alteração complexo	•	•	•	1,2,3
A necessidade de extensa documentação	Automatização de tarefas manuais de rotina	✓	•	1,2,3,4,5,6
Dependência entre os diversos módulos do software	Envolver um especialista de cada módulo para análise de impacto em cada projeto	✓	•	1,2,3,4,5,6

Tabela 18: Desafios x Soluções - Tecnologia.

Código Axial	Código Aberto	Referências Bibliográficas	Entrevista	Fase
Suporte de hardware				
Ambiente de hardware não qualificado	Focando na padronização	✓	•	2,3,4,5,6
Falhas de software devido a hardware inadequado / defeituoso	•	•	•	4,5,6
Suporte de software				
Falta de ferramentas de apoio	Usando tecnologia para automatizar tarefas	✓	•	3,4,5,6
Ambiente de teste				
Limitação na reprodução do cenário anômalo	Métricas de estabilidade de software no código fonte tocadas	✓	•	4,5
Diferença entre desenvolvimento e ambiente de produção	Ter compartilhado ou mesmo ambiente de teste em todos os sites	✓	✓	4,5

5.3.4. *Fator de tecnologia*. As soluções em relação aos desafios do fator tecnologia foram mapeados e são apresentadas na Tabela 17. Um (1) mapeamento não foi detectado na literatura relacionada à evolução distribuída de software. Um (1) dos quatro (4) mapeamentos identificados estão mais fortemente relacionados à evolução em particular. Nenhum mapeamento foi adicionado por meio das entrevistas.

Q1: Quais são os desafios de gerenciar o processo de evolução distribuída de software pela perspectiva de organizações clientes de softwares corporativos?

Tendo como base a estrutura conceitual mapeada, o autor classificou os desafios do gerenciamento do processo de evolução distribuída de software em fatores de pessoas, processos, produtos e tecnologia. Para pessoas, os fatores específicos estão relacionados à atitude pessoal, interesses conflitantes, habilidade interpessoal, diferenças de idioma e pessoal. Para processos, os fatores específicos estão relacionados ao gerenciamento de mudanças, gerenciamento da qualidade, gerenciamento da comunicação, gerenciamento do escopo, gestão do conhecimento e gerencimaneto de riscos. Para produto, os fatores específicos estão relacionados à independência de terceiros e tamanho e complexidade do software. Para tecnologia, os fatores específicos estão relacionados à suporte de hardware, suporte de software e ambiente de teste. Como os estudos apresentados nas referências bibliográficas foram realizados em diferentes contextos e complementados por entrevistas, é possível definir de que a estrutura é estável. Assim, descobertas futuras podem ser facilmente adicionadas à estrutura existente.

Q2: Como as organizações de software mitigam esses desafios?

Embora o fator pessoas não tenha muitos desafios específicos para a evolução distribuída, três (3) novas soluções sugeridas foram específicas para esse contexto. Da perspectiva de pessoal, foi identificada a importância de monitorar e priozar as solicitações de mudanças constantemente, realizar ligações e videoconferências com padronização de atas para todas as definições e, por fim, formalizar com toda a equipe o melhor período para a realização dos testes. O maior número de soluções foi identificado para o fator do processo. Quatro (4) novas soluções sugeridas foram específicas para esse contexto. Da perspectiva de processo, foi identificada a importância de monitorar e priorizar as solicitações de mudança constantemente, definir o escopo e tratar como futura solicitação de mudanças, reavaliar as prioridades dos projetos ativos e criação de uma base de conhecimento exclusivamente para testes. Da perspectiva do produto, três (3) novas soluções sugeridas específicas foram propostas no contexto. Foi identificada a importância de negociação de periodicidade de releases de componentes de terceiros, abertura de pontos no código fonte para desenvolvimento por terceiros em entre trechos de códigos fechado e, por fim, monitoramento e priorização das solicitações de mudança. Da perspectiva da tecnologia, nenhuma nova solução sugerida específica de evolução foi proposta no contexto. Depois de estruturar as soluções, foram mapeados os desafios e as soluções entre si. Em alguns casos, a ligação entre desafios e soluções pode ser deduzida, mas foi explicitada muito raramente nas próprias investigações. Assim, o vínculo é baseado na interpretação e pode servir como insumo para a formulação de proposições e teorias. Um número alto de proposições pode ser derivado das tabelas. Por exemplo, a partir do mapeamento na Tabela 15, várias soluções foram apresentadas para lidar com a indisponibilidade de especialistas em domínio, o que desperta uma possibilidade futura de avaliar qual das soluções é melhor aplicada para determinado contexto. Além disso, desperta uma possibilidade futura de avaliar quais dos desafios são mais evidentes e presentes na rotina dos projetos, possibilitando maior planejamento e prioridade sobre a atuação da equipe frente aos desafios de maior prioridade.

4.4 MELHORIAS NO FLUXO DO PROCESSO DE EVOLUÇÃO DISTRIBUÍDA

A análise e interpretação dos resultados realizada na seção anterior abre espaço para algumas melhorias no fluxo de gerenciamento do processo de evolução distribuída de software. Para isso, levou-se em conta as soluções sugeridas pelos entrevistados para os desafios encontrados neste processo. O fluxo do processo pode ser melhorado para que desafios, como o "grande número de solicitações de mudança" possam ser mitigados por soluções, como

"monitorar e priozar constantemente as solicitações de mudanças", seguindo as sugestões compartilhadas pela perspectiva dos entrevistados e pela interpretação do autor.

As principais melhorias identificadas são destacadas nesta subseção. É importante destacar que os fluxos abaixo não apresentam todas as atividades que podem ser envolvidas nas etapas do processo de evolução distribuída de software. Eles destacam apenas melhorias sugeridas por meio das novas soluções identificadas, dentro do fluxo do processo apresentado na Figura 2 (seção 2.4):

Monitor de Mudanças: A inserção de uma pré etapa (Figura 6) ao fluxo do processo de evolução distribuída, exclusivamente para monitorar possíveis solicitações de mudanças de curto a médio prazo, é resultado de uma das novas soluções sugeridas para mitigar alguns dos desafios identificados na estrutura conceitual. Conforme apresentando na Figura 7, o monitor de mudanças é baseado em um processo contínuo onde, principalmente, a organização cliente do software corporativo, por meio do consultor de cada módulo funcional ou uma equipe, monitora possíveis solicitações de mudanças futuras no ambiente onde a organização está inserida. A intenção é que discussões informais e "boatos" sobre mudanças futuras sejam acompanhadas e avaliadas para que, quando formalizadas, a organização já tenha em mãos informações o suficiente sobre os impactos das novas solicitações frente aos projetos futuros e, principalmente, aos projetos que estão em andamento. Dessa forma, desafios como: solicitações de mudança transitória, grande número de solicitações de mudança, solicitações de mudança voláteis, prazos impostos pelos stakeholders, pedido de alteração pouco claro, diferentes projetos sendo realizados paralelamente, ignorância de acompanhar atualizações de soluções de terceiros, mudanças em componentes de terceiros; podem possivelmente serem tratados com soluções, como: monitorar e priozar as solicitações de mudanças constantemente, estabelecer priorização sobre solicitações de mudança, definir o escopo e tratar como futura solicitação de mudança, reavaliar as priodades dos projetos ativos, processos definidos para enviar uma solicitação de mudança, por exemplo.

Feedback (priorização): Apesar de ter um fluxo semelhante ao do monitor de mudanças, o ciclo de Feedback (priorização) (Figura 8) sugere, também pela perspectiva da organização cliente do software corporativo, a avaliação da prioridade da solicitação de mudança durante a realização de um ou mais projetos. A priorização de implementação de mudanças se dá por políticas de priorização que a própria organização cliente do software corporativo definir. Há definições que são comumente tratadas como sendo de maior prioridade que outras. Por exemplo, o tipo de stakeholder que solicita uma mudança pode estar diretamente relacionado à prioridade de uma mudança, ou seja, uma solicitação de adição de

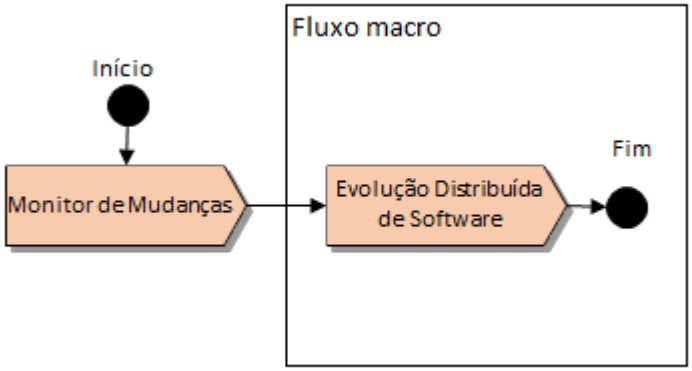

Figura 6: Fluxo macro do processo de evolução distribuída

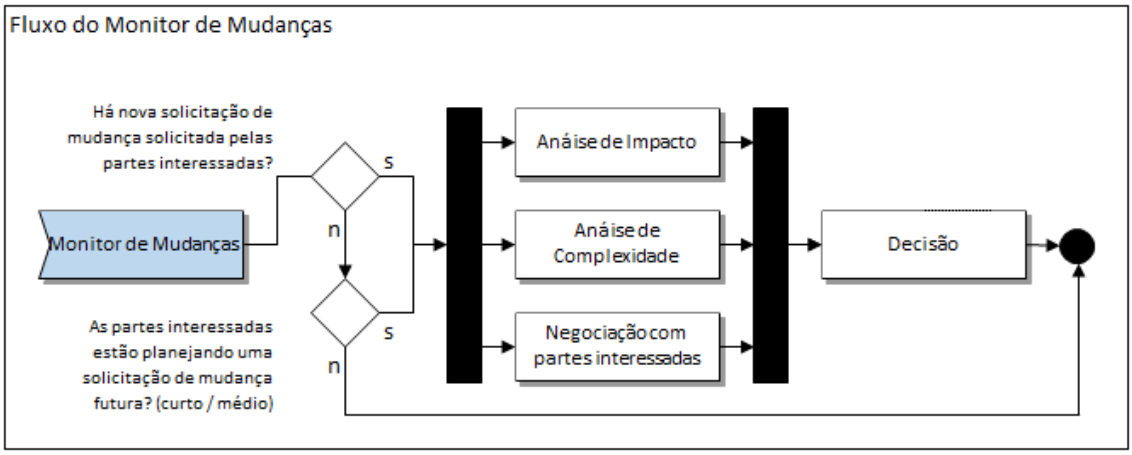

Figura 7: Fluxo do monitor de mudanças

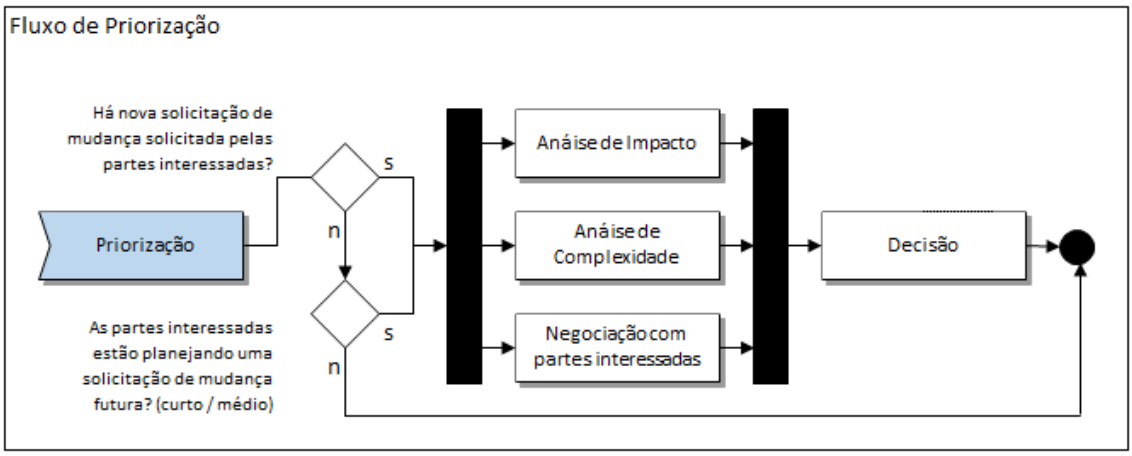

Figura 8: Fluxo do priorização

funcionalidade realizada internamente por um departamento da organização provavelmente terá menor prioridade que uma solicitação de adição de nova funcionalidade realizada por um órgão regulamentador, pois as penalidades de não cumprimento são maiores. No entanto, cabe à organização cliente do software corporativo o alinhamento e decisão por qual política de priorização optar, uma vez que, em meio às entrevistas deste estudo, por exemplo, foi verificado que cada organização trabalha com um segmento e, por isso, cada organização demonstrou optar por priorizar mudanças por meio de sua própria política.

Marcos de priorização: Os marcos de priorização são sugeridos para inserir o ciclo de Feedback (priorização) nas diversas etapas do fluxo de evolução distribuída de software, conforme apresentado na Figura 9. É de extrema importância o envolvimento e aceite de todos os stakeholders nas principais definições do projeto para que desafios, como "pedido de alteração pouco claro" não se torne uma "solicitação de mudança transitória". Isto é, se um pedido de alteração pouco claro for incorretamente diagnosticado e especificado ("pedido de alteração pouco claro"), por exemplo, poderá haver solicitação de mudanças de requisitos em meio ao andamento do projeto. Nesse momento, existe a oportunidade de avaliar se uma solicitação de mudança transitória deverá ser considerada como alteração do escopo do projeto, ou se deverá ser considerado como alteração futura. O processo de priorização deverá estar presente em todas as etapas do fluxo. Portanto, é necessário que a equipe avalie constantemente a necessidade de priorização de etapas. Caso a priorização permaneça apenas no início do projeto, seria necessário aguardar o fim do projeto iniciar um novo ciclo. Nessa etapa, mudanças constantes nas solicitações de mudança podem atrasar o processo.

As melhorias acima são resultados das principais soluções extraídas das entrevistas de profissionais com anos de experiência na engenharia global de software, somados às

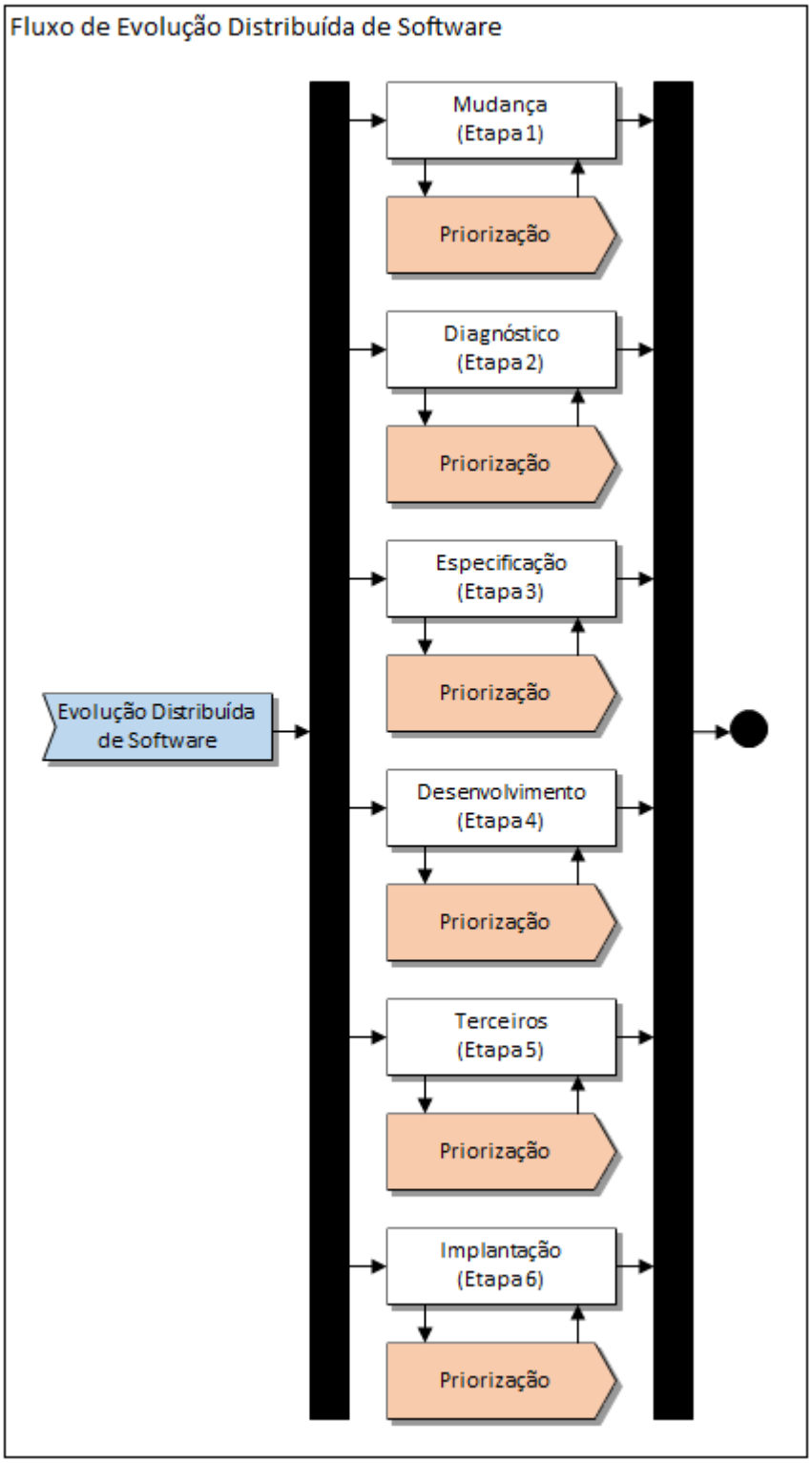

Figura 9: Fluxo de evolução distribuída de software

interpretações do autor. Levando em consideração as referências bibliográficas deste estudo, a perspectiva das organizações clientes de softwares corporativos parece nem sempre ser levada em conta como uma parte interessada ativa, mas sim passiva. Isso se deve justamente pelo fato de que muitas mudanças ocorrem no mercado global sem que a perspectiva da organização cliente seja devidamente colocada como uma das partes interessadas mais afetadas com a mudança. Um exemplo comum é o governo de um país ou órgão regulamentador surgir com uma necessidade específica de mudança, e as organizações clientes apenas descobrem que há uma mudança quando o esta é publicada já com um prazo definido para implantação. Portanto, o resultado que deste estudo fomenta o estado da arte, destacando uma maior atenção para as organizações que são clientes de softwares corporativos, e uma das partes interessadas mais afetadas neste processo. A aplicação deste estudo em outras organizações pode resultar em descobertas tanto de novos desafios quanto novas soluções para a evolução distribuída de software.

4.5 LIMITAÇÕES DO TRABALHO

Esse trabalho está limitado às perspectivas das organizações clientes de sistemas corporativos, não levando em conta as perspectivas de outras partes interessadas, pois o objetivo é identificar como essas partes interessadas, especificamente, gerenciam o processo de evolução distribuída de software. Foram incluídas apenas investigações sobre a fase de evolução do ciclo de vida do sistema corporativo ERP (EDEN et al., 2014).

O tipo de sistema corporativo analisado é o proprierário, uma vez que o contexto deste estudo está inserido neste cenário. Além de que muitos estudos empíricos em sistemas corporativos proprietários não são facilmente encontrados, replicados ou estendidos; e em segundo lugar, muitos detalhes são geralmente negligenciados do estudo (GODFREY; GERMAN, 2008).

O foco deste estudo é investigar os locais onde a evolução é realizada em diferentes países.

Este trabalho envolve apenas atividades de melhoria e adição de novas funcionalidades, que são atividades onde, geralmente, os cenários são menos previsíveis e mais complexos. Portanto, não envolve pequenas correções para fazer o software permanecer em pleno funcionamento. Apesar de compartilhar algumas características em comum com desenvolvimento e manutenção, no escopo deste trabalho, é discutido apenas desafios e estratégias de mitigação da evolução distribuída de software.

5 CONCLUSÃO

A engenharia de software global é uma área de conhecimento que disponibiliza diversas oportunidades de investigação científica, pois ainda é considerada uma área em constante crescimento, principalmente, pelas motivações por trás dessa tendência, como: custos mais baixos para o trabalho; desenvolvimento ininterrupto; pressão sobre o mercado (governo, clientes, fornecedores); e falta de mão-de-obra lucrativa (SENGUPTA et al., 2006).

Inicialmente, o autor realizou uma pesquisa da literatura e baseou o desenho da entrevista nos achados; em particular, novos desafios e soluções foram obtidos. As Tabelas 15-18 apresentam quais desafios e soluções foram identificados, somados dos achados do referencial teórico. Assim, acredita-se na utilidade em combinar pesquisas da literatura com outras metodologias, para construir estudos baseados em seus resultados.

A estrutura foi derivada usando o método de teoria fundamentada. O método de teoria fundamentada ajudou o autor a conduzir uma análise abrangente e estruturada dos dados qualitativos. A desvantagem é que a codificação pode ser influenciada pela interpretação e pontos de vista do indivíduo que faz a codificação. Com relação a este estudo, foi proposta uma abordagem conceitual com conceitos relacionados a desafios e soluções, e uma consequente melhoria do fluxo do processo de evolução distribuída de software. Em trabalhos futuros, a base de conhecimento poderá ser ampliada ainda mais com base em estudos e entrevistas adicionais.

A ausência e limitações de referências na literatura nortearam a necessidade do desenvolvimento desta pesquisa, o trabalho relatou a lacuna encontrada no gerencimamento do processo de evolução distribuída de software pela perspectiva de organizações clientes, agravada por limitações de referências na literatura para auxílio nesse processo. Acredita-se que tal lacuna interfere diretamente na realização de projetos de evolução distribuída. Supõe-se que os benefícios obtidos durante o diagnóstico, especificação e priorização de uma mudança devam recompensar de modo significativo em um resultado satisfatório para a organização cliente, dado que essas atividades são consideras um fator crítico de sucesso ou

fracasso para o desenvolvimento de um projeto.

Com o mapeamento foi possível desenvolver uma estrutura conceitual de desafios e soluções para que profissionais possam conduzir o gerenciamento do processo de evolução distribuída de software, determinando quais desafios eles podem enfrentar ao tornar a evolução distribuída. Dessa forma, estes profissionais podem encontrar soluções para os desafios que estão enfrentando. Por fim, a estrutura conceitual abriu espaço para melhorias no fluxo do processo de gerencimaneto da evolução distribuída de software, possibilitando um fluxo mais formal e padronizado de execução deste processo, visando obter o produto final desejado.

O mapeamento dos desafios e soluções e o fluxo de evolução distribuída de software proposto poderão contribuir tanto para a adição de novas funcionalidades para sistemas corporativos ERP, quanto para outros tipos de sistemas corporativos, visto que a abordagem pode ser adaptada para cada cenário específico.

Constatam-se, com base nos objetivos do trabalho, que as afirmações contidas neste são referentes à literatura disponível estudada, os métodos de pesquisa realizados, bem como as publicações realizadas durante o desenvolvimento do estudo. Este trabalho fornece um referencial teórico refinado sobre o contexto de evolução distribuída de software, que tende a viabilizar a construção de conhecimento. Pelo todo exposto, é esperado com esta pesquisa contribuir para uma visão mais ampla e uma maior compreensão da importância das estratégias de gerenciamento do processo evolução distribuída de software.

5.1 CONTRIBUIÇÕES DO TRABALHO

Para que fosse possível atingir o objetivo da pesquisa, como citado, foi necessário o desenvolvimento de entrevistas, analisadas pelo método de teoria fundamentada. Este processo teve como finalidade identificar e descrever quais são os desafios e soluções do gerenciamento da evolução de software em configurações distribuídas globalmente e como as organizações de software mitigam esses desafios.

Conforme relatado no referencial teórico, há pouca pesquisa sobre o processo de evolução distribuída de software, dado que estudos têm focado no desenvolvimento de software global e, mais recentemente, na manutenção de software global (ULZIIT et al., 2015). Assim sendo, a pesquisa desenvolvida resultou em uma abordagem baseada nos conceitos dessas duas áreas, e em características específicas, para auxiliar no gerenciamento do processo de evolução distribuída de software.

As contribuições resultantes da abordagem conceitual deste trabalho são:

- A necessidade de monitoramento constante: seja por meio de um ou mais membros responsáveis, ou até por meio de uma equipe, a organização cliente precisa de alguma forma ser notificada de que uma mudança está sendo planejada e será solicitada por uma das partes interessadas, seja a curto ou médio prazo.

- A necessidade de priorização: cada etapa do processo de evolução se demonstra ser autônoma, pois, no início de cada uma delas, é esperado que haja uma atividade de planejamento e priorização. Essa atividade envolve avaliar se o que foi definido para aquela etapa deverá prosseguir conforme o planejado, se os desenvolvimentos deverão ser desfeitos, ou se um outro projeto deverá estar à frente, por exemplo.

- A necessidade do teste de sistema em campo: com colaboradores multifuncionais e multitarefas, a organização cliente geralmente não disponibiliza de tempo o suficiente para a execução dos cenários de testes. Estes, apesar de serem planejados nas etapas iniciais do projeto, são negligenciados por diversos motivos, sendo, por vezes, executados *on-the-fly* (KANER et al., 1999). Portanto, deve existir um consenso entre a necessidade de conclusão dos testes de um projeto com as rotinas de obrigações e entregas dos departamentos da organização cliente.

Todas essas contribuições geram, por sua vez, possibilidades futuras específicas de exploração de cada uma delas.

Tendo em vista os resultados obtidos na execução do método, é possível afirmar que a abordagem auxilia no processo de evolução distribuída de software, uma vez que compila diversas referências bibliográficas e entrevistas de profissionais com anos de experiência na área.

Pelo exposto, a pesquisa contribui para uma visão mais ampla na compreensão do processo de evolução distribuída de software, visto que parte deste trabalho já possui artigo aceito e publicado (Apêndice B).

5.2 TRABALHOS FUTUROS

Identifica-se um potencial de crescimento nesta linha de pesquisa. A solução para alguns desafios ainda permenace escassa na literatura até o momento. Ainda há a possibilidade de novos desafios serem descobertos conforme novos estudos forem sendo realizados.

Com a aplicação do método de teoria fundamentada, algumas possibilidades de pesquisas futuras foram constatadas, são elas:

- Aplicação da abordagem em um novo contexto produtivo;
- Refinar a abordagem de acordo com os novos dados obtidos;
- Para desafios que apresentam múltiplas soluções, há uma possibilidade de avaliar qual das soluções é melhor aplicada para determinado contexto;
- Identificar como criar uma base de testes e como priorizar os cenários desta;
- Avaliar quais dos desafios são mais evidentes e presentes na rotina dos projetos, possibilitando maior planejamento e prioridade sobre a atuação das equipes frente aos desafios;;
- Identificar como partes do código fonte podem ser abertas;
- Identificar como se dá a transição de serviço dos etapa pré e da etapa pós evolução;
- Identificar como lidar com a concorrência de projetos de evolução por meio de portfólio de projetos.

Outras abordagens podem ser utilizadas no contexto, como por exemplo: entre desafios que apresentam mais de uma solução possível, há uma preferida em relação à outra, ou elas devem ser combinadas? Para cada um dos mapeamentos, também pode-se questionar se existe realmente um vínculo explícito entre um desafio e uma solução, e como essa solução pode ser melhor detalhada para os profissionais, para cada cenário que cada organização está envolvida. Portanto, essas abordagens abrem espaço para diversas discussões específicas sobre cada desafio e sua respectiva solução para determinado contexto.

REFERÊNCIAS

ADDO-TENKORANG, R.; HELO, P. Enterprise resource planning (erp): A review literature report. In: **Proceedings of the World Congress on Engineering and Computer Science**. [S.l.: s.n.], 2011. v. 2, p. 19–21.

AHMED, R. E. Software maintenance outsourcing: Issues and strategies. **Computers & Electrical Engineering**, Elsevier, v. 32, n. 6, p. 449–453, 2006.

AKBAR, M. A. et al. Investigation of the requirements change management challenges in the domain of global software development. **Journal of Software: Evolution and Process**, Wiley Online Library, p. e2207, 2019.

ALBADRI, F. A.; ABDALLAH, S. Erp training and evaluation: Erp life-cycle approach to end-users' characterization and competency building in the context of an oil & gas company. **Ibima business review**, v. 3, n. 2, 2009.

AMANATIDIS, T.; CHATZIGEORGIOU, A. Studying the evolution of php web applications. **Information and Software Technology**, Elsevier, v. 72, p. 48–67, 2016.

ANWAR, R. et al. Systematic literature review of knowledge sharing barriers and facilitators in global software development organizations using concept maps. **IEEE Access**, IEEE, v. 7, p. 24231–24247, 2019.

AUDY, J. L. N.; PRIKLADNICKI, R. **Desenvolvimento distribuído de software**. [S.l.]: Elsevier, 2007.

BENNETT, K. H.; RAJLICH, V. T. Software maintenance and evolution: a roadmap. In: ACM. **Proceedings of the Conference on the Future of Software Engineering**. [S.l.], 2000. p. 73–87.

BHATT, P. et al. An influence model for factors in outsourced software maintenance. **Journal of Software Maintenance and Evolution: Research and Practice**, Wiley Online Library, v. 18, n. 6, p. 385–423, 2006.

BHATT, P. et al. An empirical study of factors and their relationships in outsourced software maintenance. In: IEEE. **2006 13th Asia Pacific Software Engineering Conference (APSEC'06)**. [S.l.], 2006. p. 301–308.

BIANCHI, A. et al. Defect detection in a distributed software maintenance project. In: **GSD'03 The International Workshop on Global Software Development**. [S.l.: s.n.], 2003. p. 48.

BOEHM, B. W. **Software Engineering Economics**. 1st. ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1981. ISBN 0138221227.

CALVERT, C.; SEDDON, P. B. The importance of ongoing erp training and support. **ACIS 2006 Proceedings**, p. 91, 2006.

CAMILO, J. R. M. et al. A process for distributed software evolution: a proprietary software case study. In: ACM. **Proceedings of the 13th Conference on Global Software Engineering**. [S.l.], 2018. p. 44–53.

CANFORA, G.; CIMITILE, A. Software maintenance. In: **Handbook of Software Engineering and Knowledge Engineering: Volume I: Fundamentals**. [S.l.]: World Scientific, 2001. p. 91–120.

CHEN, J.-C.; HUANG, S.-J. An empirical analysis of the impact of software development problem factors on software maintainability. **Journal of Systems and Software**, Elsevier, v. 82, n. 6, p. 981–992, 2009.

CORBIN, J.; STRAUSS, A. Basics of qualitative research: Techniques and procedures for developing grounded theory (3rd ed.). thousand oaks, ca: Sage. **Organizational Research Methods - ORGAN RES METHODS**, v. 12, p. 614–617, 07 2008.

DAMIAN, D.; MOITRA, D. Guest editors' introduction: Global software development: How far have we come? **IEEE software**, IEEE, v. 23, n. 5, p. 17–19, 2006.

DANEVA, M.; WIERINGA, R. Cost estimation for cross-organizational erp projects: Research perspectives. **Software Quality Journal**, Kluwer Academic Publishers, Hingham, MA, USA, v. 16, n. 3, p. 459–481, set. 2008. ISSN 0963-9314. Disponível em: <http://dx.doi.org/10.1007/s11219-008-9045-8>.

DENZIN, N. K.; LINCOLN, Y. S. **The Sage handbook of qualitative research**. [S.l.]: Sage, 2011.

EDEN, R.; SEDERA, D.; TAN, F. Sustaining the momentum: archival analysis of enterprise resource planning systems (2006–2012). **Communications of the Association for Information Systems**, v. 35, n. 1, p. 3, 2014.

ERDIL, K. et al. Software maintenance as part of the software life cycle. **Comp180: Software Engineering Project**, p. 1–49, 2003.

ERLIKH, L. Leveraging legacy system dollars for e-business. **IT professional**, IEEE, v. 2, n. 3, p. 17–23, 2000.

ESTEVES, J.; PASTOR, J. An erp lifecycle-based research agenda. In: **1st International Workshop in Enterprise Management & Resource Planning**. [S.l.: s.n.], 1999.

GODFREY, M. W.; GERMAN, D. M. The past, present, and future of software evolution. In: **2008 Frontiers of Software Maintenance**. [S.l.: s.n.], 2008. p. 129–138.

GONZALEZ, R.; GASCO, J.; LLOPIS, J. Information systems outsourcing risks: a study of large firms. **Industrial management & Data systems**, Emerald Group Publishing Limited, v. 105, n. 1, p. 45–62, 2005.

GONZALEZ, R.; GASCO, J.; LLOPIS, J. Information systems outsourcing: A literature analysis. **Information & management**, Elsevier, v. 43, n. 7, p. 821–834, 2006.

GUERRA, J. H. L. Proposta de um protocolo para o estudo de caso em pesquisas qualitativas. **Encontro Nacional de Engenharia de Produção**, v. 30, p. 1–13, 2010.

HA, Y. M.; AHN, H. J. Factors affecting the performance of enterprise resource planning (erp) systems in the post-implementation stage. **Behaviour & Information Technology**, Taylor & Francis, v. 33, n. 10, p. 1065–1081, 2014.

HERBSLEB, J. D.; MOCKUS, A. An empirical study of speed and communication in globally distributed software development. **IEEE Transactions on software engineering**, IEEE, v. 29, n. 6, p. 481–494, 2003.

HERBSLEB, J. D.; MOITRA, D. Global software development. **IEEE software**, IEEE, v. 18, n. 2, p. 16–20, 2001.

HOLMSTROM, H. et al. Global software development challenges: A case study on temporal, geographical and socio-cultural distance. In: IEEE. **2006 IEEE International Conference on Global Software Engineering (ICGSE'06)**. [S.l.], 2006. p. 3–11.

HUNT, B.; TURNER, B.; MCRITCHIE, K. Software maintenance implications on cost and schedule. In: IEEE. **2008 IEEE Aerospace Conference**. [S.l.], 2008. p. 1–6.

HUSSEY, J. M.; HALL, S. E. **Managing global development risk**. [S.l.]: Auerbach Publications, 2007.

IQBAL, A.; GENCEL, C.; ABBAS, S. **Communication risks and best practices in global software development**. [S.l.]: Ajmal Iqbal, 2012.

ISO/IEC. International standard-iso/iec 14764 ieee std 14764-2006 software engineering; software life cycle processes &; maintenance. 2006.

JABANGWE, R.; ŠMITE, D.; HESSBO, E. Distributed software development in an offshore outsourcing project: A case study of source code evolution and quality. **Information and Software Technology**, Elsevier, v. 72, p. 125–136, 2016.

KANER, C.; FALK, J.; NGUYEN, H. Q. **Testing Computer Software, 1999**. [S.l.]: John Wiley & Sons, Inc. New York, NY, USA, 1999.

KOSKINEN, J. Software maintenance costs. **Information Technology Research Institute, ELTIS-Project University of Jyväskylä**, p. 16, 2003.

KOTB, M. T.; HADDARA, M.; KOTB, Y. T. Back-propagation artificial neural network for erp adoption cost estimation. In: SPRINGER. **International Conference on ENTERprise Information Systems**. [S.l.], 2011. p. 180–187.

KVALE, S.; BRINKMANN, S. **Interviews: Learning the craft of qualitative research interviewing**. [S.l.]: Sage, 2009.

LANUBILE, F.; DAMIAN, D.; OPPENHEIMER, H. L. Global software development: technical, organizational, and social challenges. **ACM SIGSOFT Software Engineering Notes**, ACM, v. 28, n. 6, p. 2–2, 2003.

LEHMAN, M. M. Programs, life cycles, and laws of software evolution. **Proceedings of the IEEE**, IEEE, v. 68, n. 9, p. 1060–1076, 1980.

LEHMAN, M. M. Laws of software evolution revisited. In: SPRINGER. **European Workshop on Software Process Technology**. [S.l.], 1996. p. 108–124.

LEHMAN, M. M.; BELADY, L. A. (Ed.). **Program Evolution: Processes of Software Change**. San Diego, CA, USA: Academic Press Professional, Inc., 1985. ISBN 0-12-442440-6.

LI, T.; YONGGANG, M.; JIANJIE, D. Metric-based tracking management in software maintenance. In: IEEE. **2010 Second International Workshop on Education Technology and Computer Science**. [S.l.], 2010. v. 1, p. 675–678.

LIENTZ, B. P.; SWANSON, E. B. **Software maintenance management: A study of the maintenance of computer application software in 487 data processing organizations**. [S.l.]: Addison-Wesley, 1980.

LIENTZ, B. P.; SWANSON, E. B. **Software maintenance management: a study of the maintenance of computer applications software in 487 data processing organizations**. [S.l.]: Addison-Wesley, Reading, MA, 1980.

LOPEZ, C.; SALMERON, J. L. A framework for classifying risks in erp maintenance projects. In: **Proceedings of the International Conference on e-Business**. [S.l.: s.n.], 2011. p. 1–4.

MCKEAN, E. **The new oxford American dictionary**. [S.l.]: Oxford University Press New York, 2005.

MENS, T.; DEMEYER, S. **Software Evolution**. 1. ed. [S.l.]: Springer Publishing Company, Incorporated, 2008. ISBN 3540764399, 9783540764397.

MENS, T. et al. Guest editors' introduction: Software evolution. **IEEE Software**, v. 27, n. 4, p. 22–25, July 2010. ISSN 0740-7459.

MISH, F. C. **Merriam-Webster's collegiate dictionary**. [S.l.]: Merriam-Webster, 2004.

MOE, N. B.; ŠMITE, D. Understanding lacking trust in global software teams: A multi-case study. In: MÜNCH, J.; ABRAHAMSSON, P. (Ed.). **Product-Focused Software Process Improvement**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. p. 20–34. ISBN 978-3-540-73460-4.

NAKATSU, R. T.; IACOVOU, C. L. A comparative study of important risk factors involved in offshore and domestic outsourcing of software development projects: A two-panel delphi study. **Information & Management**, Elsevier, v. 46, n. 1, p. 57–68, 2009.

NG, C. S. P.; CHAN, T.; GABLE, G. G. A client-benefits oriented taxonomy of erp maintenance. In: IEEE COMPUTER SOCIETY. **Proceedings of the IEEE International Conference on Software Maintenance (ICSM'01)**. [S.l.], 2001. p. 528.

NG, C. S.-P.; GABLE, G. G. Maintaining erp packaged software – a revelatory case study. **Journal of Information Technology**, v. 25, n. 1, p. 65–90, Mar 2010. ISSN 1466-4437. Disponível em: <https://doi.org/10.1057/jit.2009.8>.

NIDHRA, S. et al. Knowledge transfer challenges and mitigation strategies in global software development—a systematic literature review and industrial validation. **International journal of information management**, Elsevier, v. 33, n. 2, p. 333–355, 2013.

NOLL, J.; BEECHAM, S.; RICHARDSON, I. Global software development and collaboration: barriers and solutions. **ACM inroads**, ACM, v. 1, n. 3, p. 66–78, 2010.

NOVAIS, R. L. Visualizando evolução de software em detalhes. Instituto de Matemática, 2017.

OLIVEIRA, R. P. de; ALMEIDA, E. S. de; GOMES, G. S. da S. Evaluating lehman's laws of software evolution within software product lines: A preliminary empirical study. In: SPRINGER. **International Conference on Software Reuse**. [S.l.], 2015. p. 42–57.

OMURAL, N. K.; DEMIRORS, O. Effort estimation for erp projects: A systematic review. In: **2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)**. [S.l.: s.n.], 2017. p. 96–103.

PATTON, M. Q. **Qualitative evaluation and research methods**. [S.l.]: SAGE Publications, inc, 1990.

RAJLICH, V. Software evolution and maintenance. In: **Proceedings of the on Future of Software Engineering**. New York, NY, USA: ACM, 2014. (FOSE 2014), p. 133–144. ISBN 978-1-4503-2865-4. Disponível em: <http://doi.acm.org/10.1145/2593882.2593893>.

ROBILLARD, P. N. et al. Outsourcing software maintenance: Processes, standards & critical practices. In: IEEE. **2007 Canadian Conference on Electrical and Computer Engineering**. [S.l.], 2007. p. 682–685.

ROCHA, R. et al. Modelos de colaboração no desenvolvimento distribuído de software: uma revisão sistemática da literatura. In: **IV Workshop de Desenvolvimento Distribuído de Software, Salvador, BA**. [S.l.: s.n.], 2010.

SANTOS, A. Cristinne Corrêa dos. Comunicação em projetos de desenvolvimento distribuído de software: Um estudo terciário. Universidade Federal de Pernambuco, 2011.

SCACCHI, W. Opportunities and challenges for modeling and simulating free/open source software processes. **Institute for Software Research University of California**, 2004.

SEACORD, R. C.; PLAKOSH, D.; LEWIS, G. A. **Modernizing legacy systems: software technologies, engineering processes, and business practices**. [S.l.]: Addison-Wesley Professional, 2003.

SENGUPTA, B.; CHANDRA, S.; SINHA, V. A research agenda for distributed software development. In: ACM. **Proceedings of the 28th international conference on Software engineering**. [S.l.], 2006. p. 731–740.

SEYBOLD, C.; KELLER, R. K. Aligning software maintenance to the offshore reality. In: IEEE. **2008 12th European Conference on Software Maintenance and Reengineering**. [S.l.], 2008. p. 33–42.

SMITE, D. et al. Reporting empirical research in global software engineering: A classification scheme. In: IEEE. **2008 IEEE International Conference on Global Software Engineering**. [S.l.], 2008. p. 173–181.

ŠMITE, D. et al. Empirical evidence in global software engineering: a systematic review. **Empirical software engineering**, Springer, v. 15, n. 1, p. 91–118, 2010.

SOUZA, S. C. B. de; ANQUETIL, N.; OLIVEIRA, K. M. de. A study of the documentation essential to software maintenance. In: ACM. **Proceedings of the 23rd annual international conference on Design of communication: documenting & designing for pervasive information**. [S.l.], 2005. p. 68–75.

STRAUSS, A.; CORBIN, J. Grounded theory in practice sage. Thousand Oaks CA, 1997.

STRAUSS, A.; CORBIN, J. **Basics of qualifying research: Techniques and procedures for developing ground theory**. [S.l.]: Thousand Oaks, CA: Sage Publications, Inc, 2008.

SWANSON, E. B. The dimensions of maintenance. In: IEEE COMPUTER SOCIETY PRESS. **Proceedings of the 2nd international conference on Software engineering**. [S.l.], 1976. p. 492–497.

TIWANA, A. Beyond the black box: knowledge overlaps in software outsourcing. **IEEE software**, IEEE, v. 21, n. 5, p. 51–58, 2004.

ULZIIT, B. et al. A conceptual framework of challenges and solutions for managing global software maintenance. **Journal of Software: Evolution and Process**, Wiley Online Library, v. 27, n. 10, p. 763–792, 2015.

VERNER, J. M. et al. Risks and risk mitigation in global software development: A tertiary study. **Information and Software Technology**, Elsevier, v. 56, n. 1, p. 54–78, 2014.

WIERINGA, R.; DANEVA, M. Six strategies for generalizing software engineering theories. **Science of computer programming**, Elsevier, v. 101, p. 136–152, 2015.

WOOD, M. et al. Comparing and combining software defect detection techniques: a replicated empirical study. In: **Software Engineering—ESEC/FSE'97**. [S.l.]: Springer, 1997. p. 262–277.

YAN, Z. g. Efficient maintenance support in offshore software development: a case study on a global e-commerce project. IET, 2004.

APÊNDICE A – GUIA DE ENTREVISTA

Guia de Entrevista
A entrevista começou com o histórico do entrevistado e de sua empresa.
• Há quanto tempo você trabalha em uma organização de software?
• Há quanto tempo você está envolvido na evolução de software em ambiente distribuído?
• Você poderia me contar sobre um breve histórico da empresa ou projeto?
• Introdução do produto mantido?
Questões relacionadas a pessoas: Essas perguntas devem revelar dificuldades devido aos fatores do pessoal de evolução e suas práticas de mitigação.
• Quem está envolvido no trabalho de evolução? Por favor, descreva seus papéis e influências no projeto.
• Você concorda ou discorda de pessoas que dizem que o trabalho de evolução é muito difícil, chato ou menos inovador? Se sim, porque? Se não, porque não?
• Você já teve disputas ou mal-entendidos com outras partes interessadas (idioma ou cultura)? Se sim, como você resolveu o conflito? Se não, como você gerencia o envolvimento de diferentes partes interessadas?
• Quão difícil é manter um ritmo normal de trabalho quando um novo membro se junta ao projeto ou um membro antigo deixa o trabalho? Se muito, como seu impacto pode ser mitigado? Se não muito, como você acompanha um ritmo normal nesse caso?
• Com que frequência você precisa trabalhar por horas extras? Se muitas vezes ou mais, como você se sente? Por que isso acontece, regularmente ou inesperadamente? Se raro, qual técnica você usa para evitar sobrecarga de trabalho?
Questões relacionadas ao produto: Essas questões são para explorar os desafios causados pelos fatores específicos do produto de software que está sendo evoluído e suas práticas de mitigação.
• Existe alguma solução de terceiros (componente COTS) incluída no software mantido? Se sim, com que frequência você precisa lidar com mudanças relacionadas a eles? Quanto esforço é gasto?
• Que tipo de documentação você atualiza como alterações de software? Como a documentação ajuda nas atividades diárias?
• Na sua opinião, quais são as coisas importantes para manter a comunicação da equipe distribuída viva e eficaz? Por quê?
Questões relacionadas ao processo: Essas questões são para explorar os desafios causados por fatores relacionados ao processo na atividade de evolução e suas práticas de mitigação.
• Quais são os métodos que você usa para se comunicar com a equipe remota? Por quê?
• Com que frequência você tem uma reunião cara a cara?
• Quais são as dificuldades que você encontra na comunicação?
• Você se sente desafiado a lidar com a solicitação de mudança? Se sim, quais são eles? Como você os supera? Se não, como você resolveu isso? Quais mecanismos?
• Quem está envolvido na avaliação de risco e impacto da mudança de software?
• Quem está envolvido no processo de teste? Como?
• Como você mantém e acompanha as alterações do software? Qual mecanismo você usa?
• Quais são as dificuldades quando você implanta a evolução de software? Como eles podem ser resolvidos ou mitigados?
• Você mede o nível de sucesso ou falha de evolução? Se sim, quais atributos você inclui? Por quê? Se não, como você garante se o processo de evolução é aceitável? Por quê?
• Quais são as dificuldades encontradas durante a evolução de emergência? Como você supera?
• Você acha que sua distribuição de tarefas de evolução atual funciona bem em seu projeto? Se sim, como a tarefa é acionada e o progresso é rastreado? Se não, como poderia ser melhorado ou reorganizado?
• Você mantém algum repositório de documentação do sistema, guia do usuário ou outros documentos? Se sim, como funciona? Por que isso é necessário? Se não, como você gerencia os artefatos importantes do projeto?
• Você faz algum treinamento regular para compartilhar conhecimento entre os membros? Se sim, como você o propõe? Como isso ajuda nas atividades de evolução? Se não, como você compartilha ou troca habilidades e experiência dentro da equipe? Por quê?
• Você já se preocupou com a segurança da informação durante o treinamento ou o compartilhamento de conhecimento em outras partes interessadas? Se sim, como você lida com a preocupação com a segurança?
Questões relacionadas à tecnologia: Essas questões são para explorar os desafios trazidos pelo suporte tecnológico na atividade de evolução e suas práticas de mitigação.
• Como a falta ou o baixo suporte de hardware e software afetam as atividades de evolução? Como você resolveu eles?
• Todos os membros da equipe usam as mesmas ferramentas de software durante a evolução? Se sim, como isso ajuda em suas atividades de evolução?
• Com que frequência você enfrenta falha de software devido a falha de hardware? Se frequentemente, quais são as principais causas? Como você os resolve? Se menos, como você suporta a evolução de hardware?
• Se você quiser adicionar problemas que não são solicitados, sinta-se à vontade.

APÊNDICE B – A PROCESS FOR DISTRIBUTED SOFTWARE EVOLUTION: A PROPRIETARY SOFTWARE CASE STUDY

CAMILO, J. R. M. et al. A process for distributed software evolution: a proprietary softwarecase study. In: ACM.Proceedings of the 13th Conference on Global Software Engineering.[S.l.], 2018. p. 44–53.

Proprietary enterprise software is commonly embedded in multinational organizations and therefore has multiple sources of global or local demand. This type of software is subject to constant evolutions motivated by improvements in the features or by changes in the legal and economic context of its environment. Problems arising from the demand for requirements are associated with the suitability of a particular local requirement to its global context, for example: tax rules are characteristic of a specific country while a business rule can achieve every global structure. In addition, software customization can be implemented by the software producer, external partners, programmers allocated within the customer company, and so on. The coordination of this scenario is considered critical for the productive sector company that uses proprietary enterprise software. If the software does not evolve, the company processes can be temporarily compromised, and the software becomes obsolete. We report a case study of a large proprietary ERP system in a multinational company located in Brazil that is among the three largest exporters in its market segment. As a contribution, we present the current structure of the distributed evolution process of the software in question and how the stakeholder coordination of this scenario occurs.

Keywords: Distributed Software Development, Software Evolution, Distributed Software Evolution