UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

CATÁLOGO DE PADRÕES DE SOFTWARE NO DESENVOLVIMENTO DE PROJETOS ACADÊMICOS EM PARCERIA COM OUTRAS INSTITUIÇÕES

Santa Maria, RS, Brasil

2022

Caroline Guterres Silva

CATÁLOGO DE PADRÕES DE SOFTWARE NO DESENVOLVIMENTO DE PROJETOS ACADÊMICOS EM PARCERIA COM OUTRAS INSTITUIÇÕES

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação (PPGCC) da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Mestre em Ciência da Computação.**

ORIENTADORA: Prof. ª. Dra.. Lisandra Manzoni Fontoura

Santa Maria, RS, Brasil

2022

Silva Guterres, Caroline
CATÁLOGO DE PADRÕES DE SOFTWARE NO DESENVOLVIMENTO DE PROJETOS ACADÊMICOS EM PARCERIA COM OUTRAS INSTITUIÇÕES / Caroline Silva Guterres.- 2022.
58 p.; 30 cm

Orientador: Lisandra Manzoni Fontoura
Dissertação (mestrado) - Universidade Federal de Santa Maria, Centro de Educação Física e desportos, Programa de Pós-Graduação em Ciência da Computação , RS, 2022

1. Processo de Software 2. Tríplice-Hélice 3. Padrão de Software I. Manzoni Fontoura, Lisandra II. Título.

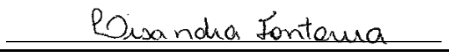
Sistema de geração automática de ficha catalográfica da UFSM. Dados fornecidos pelo autor(a). Sob supervisão da Direção da Divisão de Processos Técnicos da Biblioteca Central. Bibliotecária responsável Paula Schoenfeldt Patta CRB 10/1728.

End. Eletr.: carolguterres.silva@gmail.com

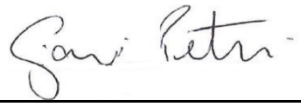
CATÁLOGO DE PADRÕES DE SOFTWARE NO DESENVOLVIMENTO DE PROJETOS ACADÊMICOS EM PARCERIA COM OUTRAS INSTITUIÇÕES

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação (PPGCC) da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Mestre em Ciênciada Computação**

Aprovado em 11 de janeiro de 2022:



Lisandra Manzoni Fontoura, Dra. (UFSM)
(Presidente/Orientador)



Giani Petri, Dr. (UFSM)

Thais Andrea Baldissera, Dra. (IFFAR- JC)

Santa Maria, RS2022

DEDICATÓRIA

À minha família.

AGRADECIMENTOS

Agradeço, primeiramente, a toda minha família, a qual sempre apoiou minha jornada de estudo, nunca medindo esforços para auxiliar no meu crescimento, e pela compreensão em todos os momentos da jornada acadêmica. Sou grata por transmitirem todo amor, carinho e união presente em nossa família. Amo cada um de vocês!

Agradeço à minha orientadora Lisandra Manzoni Fontoura por ter aceitado o convite e porestar sempre disponível em compartilhar todo seu conhecimento, bem como todo seu estímulo, paciência e compreensão durante o desenvolvimento deste estudo.

Agradeço ao Programa de Pós Graduação em Ciência da Computação da Universidade Federal de Santa Maria pela oportunidade e conhecimento adquirido ao longo desses anos, que com certeza contribuíram muito para o meu crescimento pessoal e profissional

Agradeço especialmente a todos os meus colegas de laboratório/grupo de pesquisa, com sugestões e esclarecimento de dúvidas, assim como, grata aos participantes que contribuíram para a validação do estudo, agradeço pela disponibilidade e atenção com o desenvolvimento dapesquisa.

Às demais pessoas, cuja colaboração direta ou indireta permitiu a realização deste trabalho.

RESUMO

CATÁLOGO DE PADRÕES DE SOFTWARE NO DESENVOLVIMENTO DE PROJETOS ACADÊMICOS EM PARCERIA COM OUTRAS INSTITUIÇÕES

AUTORA: CAROLINE GUTERRES SILVA
ORIENTADOR: LISANDRA MANZONI FONTOURA

Universidades têm estabelecido parcerias com empresas e/ou governo visando o desenvolvimento de projetos de inovação tecnológica para gerar soluções a partir de problemas apresentados por instituições governamentais e empresas. Neste contexto, o modelo "tríplice hélice" tem sido bastante utilizado por focar nas relações entre universidade-empresa-governo como um meio de incentivar a dinâmica de inovação. O desenvolvimento de software é uma das áreas em que essas parcerias são comumente estabelecidas. Para que esses projetos sejam bem-sucedidos, é necessário definir processos de software que consigam lidar com as características das instituições envolvidas e do problema. A partir da realização de uma revisão sistemática de literatura, identificou-se uma carência de processos de software adequados a projetos desenvolvidos na academia. Este estudo tem como objetivo propor um catálogo de padrões, documentando práticas e técnicas, recomendadas na literatura e que têm sido adotadas na elaboração de processos de software acadêmicos envolvendo parcerias externas. Os padrões consideram como contexto de uso ambientes acadêmicos que desenvolvem projetos em parceria com outras instituições. Os padrões descrevem soluções para problemas e desafios identificados no ambiente universitário no qual o desenvolvimento de software será realizado. Após a elaboração do catálogo, de acordo com os problemas identificados no ambiente da universidade, serão recomendadas práticas de desenvolvimento a serem usadas no processo específico para este projeto em parceria com outras instituições. Como resultado deste estudo, definiu-se um catálogo de recomendação de práticas de desenvolvimento de software adequadas para projetos acadêmicos realizados em parceria com empresas e/ou governo de acordo com as características do projeto. A validação por meio do questionário mostrou que grande parte dos padrões de software apresentados podem solucionam o problema identificado, de acordo com o feedback positivo dos participantes, a partir da vivência prática de desenvolvimento de software. O catálogo proposto pode auxiliar a equipe de desenvolvimento de software na tomada de decisões com relação às práticas de software, bem como pode ser estendido a outros contextos situacionais.

Palavras-chave: Tríplice Hélice. Processo de Software. Padrões.

ABSTRACT

CATALOG OF SOFTWARE PATTERNS IN THE DEVELOPMENT OF ACADEMIC PROJECTS IN PARTNERSHIP WITH OTHER INSTITUTIONS

AUTHOR: CAROLINE GUTERRES SILVA
ADVISOR: LISANDRA MANZONI FONTOURA

Universities have established partnerships with companies and/or government aiming at the development of technological innovation projects to generate solutions based on problems presented by government institutions and companies. In this context, the “triple helix” model has been widely used for focusing on university-company-government relations as a means of encouraging the dynamics of innovation. Software development is one of the areas where these partnerships are commonly established. For these projects to be successful, it is necessary to define software processes that can deal with the characteristics of the institutions involved and the problem. Based on a systematic literature review, a lack of appropriate software processes for projects developed in academia was identified. This study aims to propose a catalog of patterns, documenting practices and techniques recommended in the literature and which have been adopted in the development of academic software processes involving external partnerships. The standards consider academic environments that develop projects in partnership with other institutions as a context for use. Patterns describe solutions to problems and challenges identified in the university environment in which software development will take place. After preparing the catalog, according to the problems identified in the university environment, development practices will be recommended to be used in the specific process for this project in partnership with other institutions. As a result of this study, a catalog of recommendations for software development practices suitable for academic projects carried out in partnership with companies and/or government was defined, according to the characteristics of the project. The validation through the questionnaire showed that most of the software standards presented can solve the identified problem, according to the positive feedback of the participants, based on the practical experience of software development. The proposed catalog can assist the software development team in making decisions regarding software practices, as well as being extended to other situational contexts

Keywords: Triple Helix. Software Process. Patterns.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE GRÁFICOS

SUMÁRIO

1 INTRODUÇÃO

Devido à crescente relevância do conhecimento e da pesquisa para o desenvolvimento econômico, o papel da universidade pode ser visto, além do contexto de atividades de ensino, pesquisa e extensão, ela também possui a missão de auxiliar no desenvolvimento econômico. Para isso, é necessário aproximar a universidade de outros setores da economia, estabelecer parcerias com empresas e/ou governo, visando propor soluções inovadoras para os problemas encontrados nesses ambientes (DAMOC, 2017).

Porém, é necessário observar que existem diferenças entre essas instituições em relação a cultura, metas e objetivos. Em uma visão simplista, o governo é orientado ao desenvolvimento econômico, as universidades voltadas para o conhecimento, e as empresas direcionadas para o lucro, representando três ambientes culturais diferentes (MINEIRO et al., 2019).

Diante desse contexto, verifica-se que existe um desafio, a partir do âmbito de colaboração entre as instituições, o qual torna-se necessário identificar uma abordagem de desenvolvimento de software que beneficie ambas as partes envolvidas (WOHLIN et al., 2020).

Segundo Fuggetta e Di Nitto (2014), ano após ano, o software tem se tornado um componente cada vez mais essencial e vital para a sociedade, afetando os diferentes setores da sociedade e a vida diária das pessoas. Portanto, o desenvolvimento de software é uma atividade crítica que precisa ser estudada, entendida e melhorada.

Para que projetos de software sejam bem-sucedidos, é necessário definir processos adequados que levem em consideração as características de cada projeto (KRUCHTEN 2013)(WOHLIN; RUNESON, 2021). Kruchten (2012) descreve que os fatores organizacionais influenciam diretamente nos fatores relacionados ao projeto, orientando sobre o processo e as práticas que devem ser adotadas.

Conforme Ginsberg e Quinn (1995), a adaptação de processo de software pode ser definida como uma prática de customizar um processo padrão de desenvolvimento, com a intenção de atender as necessidades de uma organização e/ou projeto. Esse processo pode envolver a seleção e adaptação de processos existentes, assim como a elaboração de novos processos.

Com base na revisão sistemática de literatura realizada (SILVA; FILHO; FONTOURA, 2020), observou-se uma carência de processos de software voltados a projetos de pesquisa desenvolvidos na academia em parcerias com indústria e/ou governo, que possuem como

resulta o desenvolvimento de um produto de software. Além disso, esses projetos podem ter características diversas de acordo com as instituições envolvidas e o software a ser desenvolvido, tais aspectos podem ser considerados como desafios diante do contexto de desenvolvimento de software aplicado ao ambiente acadêmico.

Com intuito de resolver esse problema, este trabalho propõe um catálogo elaborado a partir da literatura de processos de software e considera as duas principais abordagens de desenvolvimento de software: ágeis e planejados, bem como a combinação dessas práticas em abordagens híbridas. Boehm e Turner (2003) descrevem que por meio da adaptação de processos é possível equilibrar o uso de práticas planejadas e/ou ágeis, originando os processos híbridos.

Dessa forma, os padrões propostos visam relacionar as adversidades presentes no ambiente acadêmico com práticas de processo e organizacionais, descritas na literatura, e, assim, facilitar a elaboração de processos de software adaptados a projeto de software específico.

1.1 CONTEXTUALIZAÇÃO DO PROBLEMA

Analisa-se que projetos de inovação desenvolvidos em ambiente acadêmico em parceria com empresa e/ou governo necessitam de processos de software que atendam as diversidades desse contexto.

No entanto, conforme identificado na literatura, existe uma carência de metodologias que abordam os desafios presentes no ambiente acadêmico. Sendo assim, é essencial sugerir processos mais adequados para o desenvolvimento de um projeto, que considere os objetivos do projeto, a equipe de desenvolvimento e o domínio de negócios de software diante de um ambiente de parceria entre as instituições.

1.2 OBJETIVOS

Os objetivos deste trabalho foram divididos em geral e específicos, descritos logo abaixo.

1.2.1 Objetivo Geral

Propor um catálogo com recomendação de práticas de desenvolvimento de software, descritas como padrões de processo, para uso em projetos desenvolvidos por universidades em parceria com empresas e/ou governo.

1.2.2 Objetivos Específicos

Os objetivos específicos do presente trabalho são:

- Pesquisar sobre estudos que relatam o contexto de desenvolvimento de software em ambientes acadêmicos;
- Identificar problemas relacionados ao ambiente acadêmico em projetos de software em parceria com outras instituições;
- Analisar e relacionar práticas de software com os problemas identificados;
- Descrever os padrões de software e sugerir as soluções para os problemas;
- Validar e comparar os padrões de software identificados na literatura com a experiência prática de profissionais com projetos em parceria com outras instituições.

1.3 ORGANIZAÇÃO DO TEXTO

O texto está estruturado da seguinte forma: o Capítulo 1 detalha o problema de pesquisa e os objetivos do trabalho; o Capítulo 2 descreve os principais conceitos necessários ao entendimento da pesquisa; o Capítulo 3 relata a revisão sistemática realizada como base para desenvolvimento da pesquisa; o Capítulo 4 apresenta os trabalhos relacionados; o Capítulo 5 descreve os procedimentos metodológicos, caracterizando a pesquisa desenvolvida por meio de etapas definidas para execução do estudo; o Capítulo 6 detalha o desenvolvimento do catálogo de práticas, a partir das etapas apresentadas na metodologia; no Capítulo 7 a validação da pesquisa realizada por meio de um questionário online; o Capítulo 8 finaliza o trabalho, apresentando as considerações finais e os trabalhos futuros a partir da realização desta pesquisa.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os conceitos fundamentais e necessários para o entendimento deste trabalho, o qual contextualiza a visão de diferentes autores em uma abordagem conceitual a respeito de cooperação universidade-empresa-governo e processo de software.

2.1 COOPERAÇÃO UNIVERSIDADE – EMPRESA – GOVERNO

O termo “Tríplice Hélice” foi criado por Henry Etzkowitz nos anos 90 com o objetivo de descrever o modelo de inovação com base na relação Governo-Universidade-Empresa (ETZKOWITZ, 1994). Os autores Etzkowitz e Zhou (2017) definem que a Tríplice Hélice é um modelo de inovação em que a universidade/academia, a indústria/empresa e o governo interagem com intuito de promover o desenvolvimento por meio da inovação e do empreendedorismo. Essa teoria é embasada na perspectiva de que a universidade gera relações com as empresas e o governo, com intuito de produzir novos conhecimentos, inovação tecnológica e desenvolvimento econômico (ETZKOWITZ; LEYDESDORFF, 2000).

Nessa representação, busca-se a produção de novos conhecimentos, a inovação tecnológica e o desenvolvimento econômico por meio de processos dinâmicos de experiências nas relações entre ciência, tecnologia, pesquisa e desenvolvimento, em uma espiral de transições sem fim (MIKOSZ, 2017). As interações universidade-indústria-governo são pontos-chaves para o crescimento econômico e o desenvolvimento social baseados no conhecimento (ETZKOWITZ; ZHOU, 2017).

A importância dada à inovação tecnológica tem sido observada cada vez mais no cenário atual (SARPONG et al., 2016). Gomes, Coelho e Gonçalo (2014) afirmam que o processo de cooperação pode trazer inúmeros benefícios para os agentes envolvidos. Em processos de software, a cooperação entre os *stakeholders* é fundamental para o sucesso do projeto (PMI, 2017), e um processo de software adequado, que considere as características e objetivos dos *stakeholders*, pode promover o engajamento destes.

2.2 PROCESSO DE DESENVOLVIMENTO DE SOFTWARE

Um processo de software pode ser definido como o conjunto coerente de políticas, estruturas organizacionais, tecnologias, procedimentos e artefatos necessários para conceber, desenvolver, implantar e manter um produto de software (FUGGETTA, 2000). Portanto, um modelo de processo descreve práticas e/ou atividades que devem ser executadas no

desenvolvimento, a ordem de execução, a relação entre elas, assim como as características de cada atividade (XU; RAMESH, 2008). Na literatura existem diferentes abordagens para elaboração de processos de desenvolvimento de software, sendo as duas principais: planejados ou prescritivos e ágeis.

Processos prescritivos ou planejados impõem um processo disciplinado para desenvolvimento de software com o objetivo de torná-lo previsível e eficiente, por meio de um processo detalhado (BOEHM; THUNER,2003) (FOWLER, 2002). Os métodos planejados concentram-se no planejamento e no trabalho inicia coletando e documentando um conjunto completo de requisitos, seguido pelo desenvolvimento e inspeção de projetos arquitetônicos de alto nível (BOEHM; THUNER, 2003).

Métodos ágeis buscam balancear nenhum processo e muito processo, provendo apenas um processo suficiente para obter o resultado esperado (FOWLER, 2002). Métodos ágeis podem ser caracterizados por: desenvolvimento iterativo, avaliação e melhoria do processo continuamente, desenvolvido por equipe pequenas e altamente motivadas (PRESSMAN, 2011) (BECK et al., 2001). Segundo Koskella (2003), os métodos ágeis são relacionados ao desenvolvimento rápido de software com foco em menos tempo em análise e projeto, emprega simplicidade no desenvolvimento. Nesse desenvolvimento, a comunicação entre desenvolvedores e clientes é priorizado, dando preferência à entrega sobre a análise do projeto (BECK; ANDRES, 2004).

Vários estudos propõem a combinação de métodos ágeis e planejados, dando origem a métodos híbridos (CONFORTO et al., 2015) (BARLOW et al., 2011)(BOEHM; TURNER, 2003)(KHURMANN et al. 2019)(VIJAYASARATHY; BUTLER, 2016). Abordagens híbridas consideram as características de duas ou mais abordagens, práticas de duas ou mais abordagens e papéis de duas ou mais abordagens. Observa-se que a metodologia híbrida não está associada apenas a um método, mas representa uma variedade de combinação entre práticas tradicionais e ágeis. Boehm (2002) concluí que abordagens híbridas, que combinam métodos planejados e ágeis, são viáveis e necessárias para projetos que combinam características desses dois métodos.

Boehm e Turner (2003) descrevem que métodos ágeis, planejados ou híbridos abrangem um conjunto de condições sob as quais eles têm maior probabilidade de sucesso. A partir da análise dessas condições, os autores definiram cinco fatores críticos que descrevem um ambiente de projeto e ajudam a determinar o equilíbrio do método, que são: tamanho da equipe, criticidade, dinamismo, equipe e cultura. O modelo Octopus Model, proposto por Krutchen (2013), propõe oito fatores contextuais para a adaptação de processos de software,

que são: tamanho, arquitetura estável, modelo de negócios, distribuição da equipe, taxa de mudança, idade do sistema, criticidade e governança.

Xu e Ramesh (2008) realizaram um estudo para auxiliar gerentes de projetos a determinar como adaptar um processo de software que atenda os desafios particulares de cada projeto e definiram um questionário para avaliar o ambiente de projeto e identificar desafios atuais e futuros.

Todo projeto é único, portanto, todos os aspectos precisam ser considerados na adaptação de processos (KLUENDER et al., 2017). Dessa forma, considera-se que a adaptação é uma tarefa complexa e trabalhosa, envolvendo a seleção, elaboração e combinação de técnicas e processos, com base em todo contexto do projeto.

2.3 PADRÕES DE SOFTWARE

A origem dos padrões de projeto deu-se com o trabalho feito pelo arquiteto Christopher Alexander no final dos anos 70, contendo uma fundamentação básica para engenharia de software. Padrões são maneiras de descrever melhores práticas, bons projetos, e capturar experiências de uma forma que seja possível para outros reusarem essa experiência.

Padrões de processo podem ser utilizados para adpatção e melhoria no processo de desenvolvimento de software (HUANG; ZHANG, 2003). O objetivo dos padrões para a comunidade de software é documentar experiências para ajudar desenvolvedores de software a resolver problemas recorrentes encontrados durante o desenvolvimento de software (WAHONO, 2008).

Considerando o contexto dos padrões organizacionais, Coplien (1998) foi um dos primeiros pesquisadores a estudar padrões de processo e organizacionais. Ele propôs uma linguagem de padrões para descrever e construir novas organizações. Segundo Coplien (1998), as organizações de software bem-sucedidas apresentam os mesmos padrões organizacionais e de processo. Esses padrões não são encontrados em organizações menos produtivas ou não tão bem-sucedidas. Os padrões de organizações bem- sucedidas podem ser capturados e usados para estabelecer estruturas organizacionais e práticas que podem melhorar a probabilidade de sucesso em novas organizações.

Ambler (1998) considera que padrões de processo descrevem uma coleção de técnicas, ações e/ou tarefas (atividades) gerais para desenvolver software orientado a objetos. São considerados blocos de processo reutilizáveis, que podem ser usados para adaptar o processo de software para encontrar as necessidades específicas da organização. Quando aplicados

juntos de uma maneira organizada, padrões de processo podem ser usados para construir processos de software para a organização.

De modo geral, conforme os autores Coplien (1995) um padrão visa descrever uma solução para um problema que ocorre com frequência durante o desenvolvimento de software. Dessa forma, considera-se que catálogo de padrões possibilita agrupar os padrões, facilitando seu reuso.

3 REVISÃO SISTEMÁTICA DE LITERATURA

Em 2019 foi realizada uma revisão sistemática de literatura (RSL) com objetivo de identificar as características e limitações de metodologias ágeis e orientadas a planos e de relacionar as características organizacionais das instituições que compõem a tríplice hélice (universidade- governo-empresa) e as metodologias adotadas no processo de desenvolvimento de software (SILVA; FILHO; FONTOURA, 2020).

A revisão sistemática teve como objetivo responder às questões de pesquisa descritas abaixo e que são brevemente descritas neste trabalho. Para uma descrição mais abrangente, o artigo publicado pode ser consultado (SILVA; FILHO; FONTOURA, 2020)

RQ1: Quais são as características e limitações da metodologia ágil e orientada a planos?

As principais características dos métodos ágeis mencionados nos trabalhos foram: comunicação e colaboração com o cliente (AWAD, 2015), requisitos de software evoluem conforme o software é desenvolvido, adaptado às mudanças, desenvolvido em incrementos, entregas frequentes dos principais requisitos do sistema (TURK; FRANCE; RUMPE, 2014)(FITRIANI; RAHAYU; SENSUSE, 2016)(DOS SANTOS; ALVES; CANEDO, 2014). Por outro lado, métodos ágeis não são adequados em: ambientes de desenvolvimento distribuído, desenvolvimento envolvendo grandes equipes ou larga escala, software crítico ou complexo (TURKE et al., 2014)(AWAD, 2015). Além disso, métodos ágeis requerem uma equipe altamente qualificada e disciplinada (DOS SANTOS; ALVES; CANEDO, 2014).

Em relação a processos planejados, os trabalhos descrevem como características principais: documentação formal, uso em projetos grandes ou críticos, foco no controle, verificação e validação. A metodologia orientada a planos não é adequada em: projeto isolado de seu ambiente, restrições de tempo e incerteza no estabelecimento de metas (SPUNDAK, 2014).

RQ2: Quais são as características de cada organização (universidade, governo e indústria)?

Em relação à universidade, as características mais comuns relatadas foram:

- ✓ As instituições acadêmicas se concentram no aprendizado, pesquisa e inovação e têm como objetivo publicar artigos (DIAS; KODIKARA; JAYAWARDENA, 2013). Portanto, as reuniões ocorrem com menor frequência, porque as pessoas trabalham em período parcial nos projetos (CERECI; KARAKAYA, 2018);

- ✓ As equipes são compostas por trabalhadores com diferentes níveis de habilidade, e há uma alta rotatividade da equipe. Eles ficam na universidade enquanto fazem seus cursos (BRONDANI; MELLO; FONTOURA, 2019);
- ✓ Essas equipes precisam investigar soluções de pesquisa altamente complexas (DIAS; KODIKARA; JAYAWARDENA, 2013);
- ✓ A equipe não está familiarizada com o domínio comercial, dificultando adefinição dos requisitos de software (BRONDANI; MELLO; FONTOURA, 2019). Além disso, nem sempre os usuários finais realizam testes de software (CERECI; KARAKAYA, 2018).

As características da indústria são:

- ✓ Geralmente, o trabalho é realizado em período integral, o desenvolvimento do software é lucrativo e a entrega do produto é limitada pelo orçamento (CERECI; KARAKAYA, 2018);
- ✓ A política da empresa deve ser rigorosamente seguida e isso geralmente limita a escolha do processo mais apropriado (DIAS; KODIKARA; JAYAWARDENA, 2013) (KUHRMANN et al., 2019).

Em relação ao governo, os trabalhos destacam:

- ✓ Existência de leis ou padrões a serem seguidos (DIAS; KODIKARA; JAYAWARDENA, 2013);
- ✓ Necessidade de transparência nos dados e controle dos órgãos internos relacionados à instituição (DOS SANTOS et al., 2014);
- ✓ Alguns trabalhos destacam a alta qualidade, criticidade, confiabilidade e requisitos complexos no desenvolvimento de software para o exército (BRONDANI; MELLO; FONTOURA, 2019).

RQ3: Quais metodologias são usadas em projetos desenvolvidos em universidades, governo e indústria?

Na universidade, o processo de prototipagem e desenvolvimento de software é baseado em experiências de outros projetos, são usados quadros brancos para desenhar diagramas e mapas conceituais para ilustrar ideias, padrões não são usados com frequência (CERECI; KARAKAYA 2018). Em um projeto para o Exército, Brondani, Mello e Fontoura (2019) descrevem uma metodologia híbrida, na qual práticas relacionadas a metodologias orientadas a planos foram utilizadas para as atividades de análise, design e verificação, e métodos ágeis foram usados para gerenciar atividades de implementação e teste.

Não há consenso sobre as metodologias utilizadas na indústria. Alguns trabalhos mencionam métodos ágeis, principalmente Scrum, outros trabalhos mencionam metodologias orientadas a planos. Metodologias híbridas também são citadas (DIAS; KODIKARA; JAYAWARDENA, 2013)(KUHRMANN et al., 2019)(VIJAYASARATHY; BUTLER 2016)(KLUNDER et al., 2017)(MARINHO et al., 2019).

No contexto governamental, os trabalhos mencionam o uso de métodos ágeis, enfatizando características como a visibilidade do projeto para a equipe, a resposta a mudanças, a entrega das principais funcionalidades do sistema, o aumento da colaboração do cliente no processo de desenvolvimento e a possibilidade de estimativa do esforço (DOS SANTOS; ALVES; CANEDO, 20 014).

RQ4: Quais trabalhos descrevem o uso de metodologias de desenvolvimento híbridas?

Vários trabalhos argumentam que o uso de metodologias híbridas traz benefícios ao projeto e identificam alguns critérios para sua adoção. Os critérios são:

Os métodos ágeis enfatizam a comunicação, o compartilhamento de conhecimento e a visibilidade do projeto; por outro lado, as metodologias orientadas a planos propõem planejamento, controle e estimativa detalhados (THEOCHARIS et al., 2015);

Os processos de desenvolvimento orientados a planos são descritos por atributos como previsibilidade e estabilidade, enquanto os métodos ágeis funcionam bem em projetos com muitas alterações de requisitos e quando o cliente está envolvido no processo de desenvolvimento (RIESENER; DÖLLE; AYS, 2019).

É válido ressaltar que os resultados obtidos evidenciam a escassez de modelos de processos definidos específicamente para projetos desenvolvidos em universidades. Verificou-se que não há uma metodologia que possa cobrir as necessidades de todos os projetos de software, pois cada projeto e organização possui suas características. Além disso, fica claro a falta de processos específicos para projetos universitários e as diferenças entre esse ambiente e a indústria, o que exige processos customizados. Em contrapartida, existe um grande número de projetos utilizando práticas provenientes de mais de um método, explorando e relacionando as melhores práticas da metodologia ágil e orientada a planos, definidas como metodologias híbridas.

4 TRABALHOS RELACIONADOS

Nesta seção são apresentados alguns trabalhos que descrevem processos usados em projetos que envolvem universidade, indústria e governo.

Brondani, Mello e Fontoura (2019) descreveram um estudo de caso, no qual foi definido um processo de software para atender as necessidades específicas de um projeto desenvolvido em uma universidade em parceria com o Exército Brasileiro. Como resultado da experiência dos autores, foi elaborado um conjunto de lições aprendidas que podem ser usadas na definição de novos projetos. O processo proposto por Brondani, Mello e Fontoura (2019) se caracteriza por ser uma abordagem híbrida, na qual são utilizadas práticas como: definição de planos a partir de metas estabelecidas em contrato, requisitos estáveis e controle de mudanças, provenientes de métodos orientados a planos. Porém, práticas ágeis também são utilizadas, principalmente em relação ao gerenciamento da equipe, com reuniões semanais e entregas incrementais.

Os autores Garousi et al. (2019) apresentam um estudo, no qual buscam explorar e caracterizar projetos de colaboração entre indústria e universidade, com intuito de contribuir com a área de pesquisa e auxiliar pesquisadores e profissionais da área. Este estudo resultou em uma análise sobre as necessidades, o impacto e os desafios encontrados nos projetos que envolvem tais instituições. Os autores descrevem desafios relacionados a recursos humanos, organizacionais e comunicação, a incompatibilidade de aspectos culturais, metas e objetivos definidos por cada instituição, bem como a falta ou queda de interesse com relação ao projeto. Por outro lado, os autores identificam sugestões que visam contribuir para o sucesso do projeto, tais como: encontrar e manter um objetivo em comum com relação ao desenvolvimento, justamente pelo motivo da diversidade de características entre as instituições; trabalho em equipe e garantia de um gerenciamento eficaz.

Cereci e Karakaya (2018) descrevem sobre a necessidade de uma nova metodologia de desenvolvimento de software para projetos de pesquisa realizados em universidades. Os autores comentam que a maior parte das metodologias atende apenas às necessidades da indústria/empresa, não se adequando ao contexto de pesquisa nas universidades. Os autores entrevistaram acadêmicos que participaram de seis diferentes projetos na academia. Os autores concluíram que o fato dos membros da equipe terem dedicação parcial afeta negativamente o projeto, o nível de conhecimento dos estudantes é comparativamente mais baixo que de engenheiros da indústria, projetos de pesquisa tendem a ter mudanças de

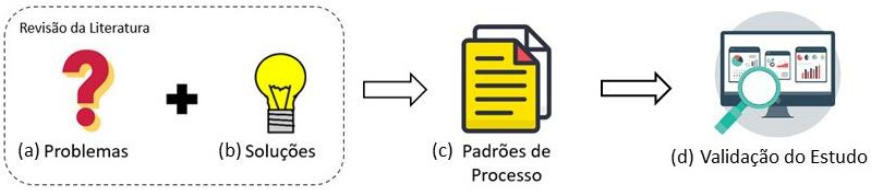
requisitos, alta rotatividade entre os membros da equipe, e burocracia administrativa associada a projetos financiados.

A partir da análise dos trabalhos relacionados, é possível identificar a necessidade de definição de práticas de desenvolvimento de software adequadas a projetos desenvolvidos em colaboração da universidade com outras instituições. Os trabalhos analisados descrevem lições aprendidas e práticas usadas em um único projeto. Consolidar as práticas descritas na literatura e organizá-las por meio de um catálogo de padrões, associando problemas comuns em projetos de software desenvolvidos em universidades com soluções adotadas, facilita o reuso e a elaboração de processos de software adotado no ambiente acadêmicos. Além disso, a sistematização facilita a melhoria das soluções descritas com base em experiências de uso.

5 METODOLOGIA

O objetivo deste trabalho é propor um catálogo de padrões de processo que documentam problemas comumente encontrados em projetos acadêmicos e possíveis soluções descritas na literatura especializada. Os padrões visam auxiliar a equipe de desenvolvimento na tomada de decisões em relação à seleção das melhores práticas de software, permitindo um melhor gerenciamento de recursos, atividades e artefatos que envolvem projetos de software desenvolvidos em ambiente acadêmico em parceria com outras instituições. Para a elaboração do catálogo de padrões, foram seguidas quatro atividades, identificadas na Figura 1, que são:

Figura 1 — Descrição da metodologia do estudo



Fonte: Autoria própria.

a) Identificação de problemas relacionados a projetos de software desenvolvidos em universidades em parceria com outras instituições;

b) Associação de práticas de software adotadas com sucesso em projetos reais para descrever os problemas identificados;

c) Documentação do padrão de processo associando os problemas encontrados às soluções descritas na literatura como práticas de software;

d) Validação do trabalho por meio de formulário online de questões com perquisadores/profissionais que tiveram experiência/vivência de desenvolvimento de software em projetos realizados em parceria com outras instituições.

6 IDENTIFICAÇÃO DE PROBLEMAS NO AMBIENTE UNIVERSITÁRIO

Os problemas foram identificados com base na literatura especializada, considerando projetos de desenvolvimento com parcerias externas. Foram realizadas buscas nas bases de dados: IEEE Xplore, ACM, Scopus, e Google Acadêmico.

Esses trabalhos foram analisados, e treze problemas foram elencados. No Quadro 1, é possível visualizar os problemas apontados por cada autor. A seguir, uma breve descrição do problema segundo a visão dos autores dos trabalhos.

- **Alta rotatividade da equipe:**

Brondani, Mello e Fontoura (2019) relatam que devido às equipes de software serem formadas por alunos de graduação ou pós-graduação, estes permanecem no projeto em média de dois anos, o qual resulta em uma alta rotatividade na equipe.

Cereci e Karakaya (2018) apontam a rotatividade entre estudantes da graduação como um problema crítico que ocasiona a perda de conhecimento e algumas divergências do plano de projeto.

- **Disponibilidade em tempo parcial:**

Em relação à disponibilidade de tempo, Brondani, Mello e Fontoura (2019), Dias Kodikara, Ekanayaka (2013), Cereci e Karakaya (2018), Andrade et al. (2017) relatam que devido à equipe de software ser formada por alunos de graduação e pós-graduação, eles necessitam dividir seu tempo com o curso e suas pesquisas acadêmicas, resultando em um tempo limitado para disponibilizar ao projeto, afetando negativamente o progresso do projeto de software.

- **Problemas de comunicação:**

A disponibilidade de tempo parcial impacta na redução da frequência das reuniões. Cereci e Karakaya (2018) comentam que quanto menos frequente são as reuniões dos projetos, menos informações os pesquisadores têm sobre o status do projeto e o progresso de outros pesquisadores.

O sistema de comunicação hierárquico existente em algumas empresas ou ambiente governamental dificulta a comunicação entre a equipe e as partes interessadas, ocorrendo ausência ou atraso nas respostas às dúvidas levantadas pela equipe e/ou cliente do projeto (BRONDANI; MELLO; FONTOURA, 2019)(DIAS et al., 2014)(ANDRADE et al, 2017).

- **Divisão de responsabilidade:**

Cereci e Karakaya (2018) destacam que as equipes universitárias são dinâmicas, e, assim, as funções com relação a cada atividade não são bem estabelecidas, bem como o nível de experiência da equipe é diferente de equipes da indústria.

- **Falta de familiaridade com domínio do projeto:**

Os autores Brondani, Mello e Fontoura (2019) e Andrade et al. (2017) descrevem dificuldades relacionadas ao domínio de negócio e às atividades realizadas pelas empresas parceiras, resultando em uma demanda de tempo adicional para compreender os processos organizacionais, bem como um envolvimento da equipe para obter conhecimento sobre estes e para desenvolvimento do projeto.

- **Trabalhadores com diferentes habilidades:**

Uma equipe acadêmica é formada por pessoas com diferentes habilidades. De acordo com Cereci e Karakaya (2018), os alunos de graduação possuem um nível de experiência e conhecimento menor comparado a profissionais no contexto de uma empresa.

- **Complexidade das soluções:**

Diante do desenvolvimento de software realizado pela universidade em parceria com outras instituições, identifica-se que as instituições buscam as universidades para desenvolvimento de soluções complexas e inovadoras que demandam pesquisas para resolver problemas (Brondani, Mello e Fontoura 2019).

Crawrford (2002), Monteiro e Alencar (2007) apontam que à medida que essa complexidade aumenta, torna-se indispensável que a comunicação seja eficaz para o sucesso do projeto.

- **Instabilidade nos requisitos:**

O desenvolvimento de software inovador implica dificuldades de se definir um conjunto de requisitos de software. A tendência é que os requisitos evoluem ao longo do projeto, a partir dos resultados das pesquisas desenvolvidas. Esse desenvolvimento iterativo implica mudanças nas especificações (BRONDANI; MELLO; FONTOURA, 2019)(DIAS, KODIKARA; EKANAYAKA, 2013)(CERECI; KARAKAYA, 2018)(ANDRADE et al. 2017).

- **Pouco contato com clientes/usuários:**

Cereci e Karakaya (2018) descrevem que em alguns projetos acadêmicos os usuários finais não fazem parte do projeto, em outros eles se envolvem na obtenção de requisitos e fornecem feedback apenas ao final do progresso.

- **Dificuldade de comercialização de produtos:**

De acordo com Andrade et al. (2017), a comercialização do produto é um dos desafios identificados em projetos realizados em parceria com outras instituições, pelo motivo de que a instituição parceira possui interesse em obter exclusivo direito comercial do produto. Justifica- se esse problema devido ao fato de que as instituições em parceria com a universidade possuem objetivos/interesses diferentes com relação à finalização do projeto (CERECI; KARAKAYA, 2018)(ANDRADE et al., 2017)

- **Dificuldade de divulgar os resultados da pesquisa devido a *non-disclosure agreements:***

Andrade et al. (2017) salientam que em projetos com outras instituições, na maioria das vezes, são firmados acordos de não divulgação (NDAs) para proteger o conhecimento de hardware e software disponibilizado durante a execução do projeto, assim como informações obtidas em visitas ou reuniões.

Esses acordos podem limitar a publicação dos resultados obtidos nas pesquisas, que são indicadores de sucesso importantes para projetos na academia.

- **Visões/objetivos divergentes:**

Dias, Kodikara, Ekanayaka (2013), Cereci e Karakaya (2018) e Andrade et al. (2017) relatam que, em muitos projetos, a universidade e a indústria têm objetivos divergentes em relação ao projeto desenvolvido. Andrade et al. (2017) relatam que, aos olhos da indústria, a academia possui apenas conhecimento teórico, enquanto, para a academia, a indústria possui apenas conhecimento prático. Essas visões diferentes podem ocasionar conflitos que precisam ser resolvidos para que os projetos atinjam seus objetivos.

- **Demora de feedback pelos stakeholders das entregas:**

Andrade et al. (2017) descrevem que projetos realizados no ambiente acadêmico em parceria com outras instituições, com relação às partes interessadas no projeto, carecem de um feedback adequado e no tempo correto, ocasionando, então, um atraso no desenvolvimento do software.

Quadro 1 — Problemas identificados no ambiente da universidade

Problemas Encontrados	T1	T2	T3	T4
Alta rotatividade da equipe	X		X	
Disponibilidade em tempo parcial	X	X	X	X
Problemas de comunicação	X	X	X	X
Divisão de responsabilidades			X	
Falta de familiaridade com domínio do projeto	X			X
Trabalhadores com diferentes habilidades			X	
Complexidade das soluções	X			
Instabilidade nos requisitos (devido a pesquisas ou não)	X	X	X	X
Pouco contato com clientes/usuários			X	
Dificuldade de comercialização de produtos				X
Dificuldade de divulgar os resultados da pesquisa devido a non-disclosure agreements				X
Visões/objetivos divergentes			X	X
Demora de feedback pelos stakeholders das entregas				X
Referências: T1 (BRONDANI; MELLO; FONTOURA, 2019); T2 (DIAS; KODIKARA; EKANAYAKA, 2013); T3 (CERECI; KARAKAYA, 2018); T4 (ANDRADE et al. 2017); T5 (MATHIES et al., 2019)*.				

Fonte: Autoria própria.

6.1 ASSOCIAÇÃO DE PRÁTICAS AOS PROBLEMAS

Após o mapeamento de problemas encontrados em projetos de software no contexto da universidade, identificou-se práticas de desenvolvimento de software utilizadas com sucesso para reduzir o impacto do problema, essas práticas foram elencadas com base na literatura utilizada no desenvolvimento da pesquisa. Portanto, para cada problema foram associadas práticas, como pode ser visto no Quadro 2. Uma prática pode estar associada a mais de um problema, e algumas práticas são descritas na literatura sem especificar o problema a que se referem. Essas são descritas na última linha do quadro.

Quadro 2 — Definição padrão de software x problema identificado

Padrão / Problema	Diagrama de processo de negócio	Processo iterativo e incremental	Definição de pesquisas individuais aos pesquisadores	Revisão constante do código-fonte	Definir responsáveis pelo fluxo de comunicação, lado cliente e equipe	Trabalhadores contratados tempo integral	Trabalho Colaborativo	Revisão de relatórios e artigos antes da publicação	Reuniões periódicas	Definição Plano de Trabalho
Alta rotatividade da equipe						T1	T1			
Disponibilidade em tempo parcial						T1;T3; T4				
Problemas de comunicação	T1				T1;T4;T5				T4;T 3	T4
Divisão de responsabilidades										T4
Desconhecimento do domínio da aplicação	T1						T1			
Trabalhadores com diferentes habilidades				T3			T1; T3			
Complexidade das soluções			T1							
Instabilidade nos requisitos		T1;T4								
Pouco contato com clientes/usuários					T4					
Dificuldade de comercialização de produtos										T4
Dificuldade de divulgar os resultados da pesquisa devido a non-disclosure agreements								T4		
Visões/objetivos divergentes								T4		
Demora de feedback pelos stakeholders das entregas										T4
Lições aprendidas sem definição de problema		T5		T5			T5			

Fonte: Autoria própria.

6.2 DOCUMENTAÇÃO DE PADRÕES DE PROCESSO

O próximo passo da pesquisa foi a documentação de padrões de processo, associando os problemas encontrados às soluções descritas na literatura como práticas de software. Para descrição de cada padrão de software, utilizou-se a base de estudos literária adotada no desenvolvimento do trabalho.

Segundo a sugestão de Schumacher et al. (2006), cada padrão é descrito por meio das seguintes propriedades: propósito, descrição do problema, solução e, para alguns casos, é descrita a dinâmica proposta para o padrão. Válido ressaltar que tais informações foram elaboradas, conforme identificação na literatura.

Cada padrão é identificado da seguinte forma: [P], conforme sequência.

a) Padrão: Elaborar diagramas de processo de negócio [P1]:

- **Propósito:**

Compreender o domínio de negócio relacionado ao software que será desenvolvido. A criação de um diagrama de processo de negócios facilita a compreensão dos processos de negócio de uma organização e ajuda a entender, especificar e priorizar os requisitos de software.

- **Problema:**

A equipe de software precisa compreender o domínio de negócio, pois é a partir desse domínio que serão extraídos os requisitos do software, assim como entender a dinâmica de negócio da instituição parceira.

Compreender o domínio de negócio é uma atividade complexa para os desenvolvedores. Em muitas situações, existem manuais que apresentam detalhes técnicos, mas são bastante extensos, tornando difícil para a equipe entender e aprender os processos organizacionais que precisam ser contemplados no novo software a partir destes.

Além dos processos de negócios, existem termos que são específicos deste domínio, e o não-conhecimento de seu significado impacta na comunicação entre a equipe de desenvolvimento e o cliente. Brondani, Mello e Fontoura (2019) citam em seu trabalho a dificuldade de compreender as doutrinas militares para desenvolvimento de um simulador virtual tático para treinamento de militares.

- **Solução:**

A partir da modelagem de diagramas de negócio, é possível compreender o domínio do projeto a ser desenvolvido (GUDWIN, 2015). Dessa forma, são coletadas informações sobre o domínio do problema que devem ser conhecidas e compreendidas pela equipe, visando facilitar a elicitação de requisitos para o sistema, a especificação, a implementação e a validação (BROY, 2013)(PRIETO-DÍAZ, 1990).

A elaboração dos diagramas de processo de negócio faz com que a equipe e o cliente tenham um entendimento comum sobre as tarefas que são executadas no domínio de negócio em questão, facilitando a comunicação entre eles e a elicitação e especificação dos requisitos do software.

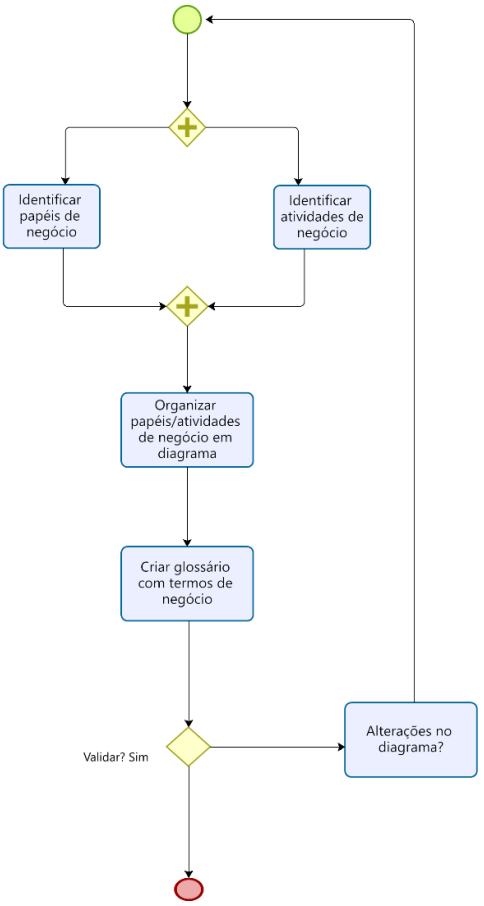
Brondani, Mello e Fontoura (2019) descrevem que a adoção de diagramas de negócios, além de auxiliar a equipe a compreender o domínio do projeto, melhorou o fluxo de comunicação entre a equipe e as partes envolvidas.

Amaral et al. (2008) e Forsell (2002) identificam as necessidades de melhorias nos processos de negócio: nivelar os trabalhos a serem realizados, desenvolver uma visão estratégica, gerenciar a organização e controlar as mudanças.

- **Dinâmica:**

Sugere-se as atividades descritas na Figura 2 para a elaboração de processos de negócio. O diagrama deve ser criado nas fases iniciais do projeto, antes da definição dos requisitos.

Figura 2 — Dinâmico do processo de negócio



Fonte: Autoria própria.

b) Padrão: Definir um processo iterativo e incremental [P2]:

- **Propósito:**

Esse padrão propõe um ciclo de vida para o processo de software que utiliza pequenos ciclos de desenvolvimento, que são chamados de iterações. No início da iteração, são priorizados os requisitos que agregam mais valor ao negócio do cliente. Ao final da iteração, é entregue uma versão operacional ao cliente e a equipe realiza uma avaliação sobre o processo de software visando melhorá-lo continuamente.

- **Problema:**

Definir um conjunto de requisitos no início do desenvolvimento e esperar que esses requisitos não mudem durante o desenvolvimento é inviável. Provavelmente, ocorrerão muitas mudanças em requisitos durante o desenvolvimento de software. O início de um projeto de software apresenta muitas incertezas em relação aos requisitos, o que ocasiona mudanças nesses

requisitos durante todo o projeto. As mudanças em requisitos causam impacto em termos de custos e tempos adicionais. Porém, mudanças em requisitos são inevitáveis e os projetos precisam identificar como lidar com elas.

- **Solução:**

A adoção da abordagem iterativa e incremental (Figura 3) permite aprimorar cenários em que mudanças são inevitáveis e, assim, controlar riscos provenientes (SCHWABER; SUTHERLAND, 2020).

No desenvolvimento iterativo e incremental, são entregues várias versões ao cliente, e os requisitos são refinados ao longo do tempo. Por isso, esse ciclo de vida responde rápido às mudanças de requisitos, proporciona o feedback constante do cliente que está interagindo com a equipe para definição de requisitos e avaliando as versões entregues do software, melhorando o fluxo de comunicação entre as pessoas envolvidas (AGILE GUIDE, 2017).

Beck (2000) e Fowler (2004) descrevem que desenvolvimento deve ser realizado em ciclos curtos e iterativos (1 - 4 semanas), no qual o resultado da próxima iteração é um incremento de trabalho aprimorado, e isso é repetido até que o produto de software atenda aos requisitos definidos.

- **Dinâmica**

Diante de um processo incremental, são realizadas várias iterações durante o projeto, sendo que a cada iteração são definidos os requisitos do software e é entregue uma nova versão ao cliente (Figura 3).

Figura 3 — Dinâmica da definição processo iterativo e incremental

Fonte: Autoria própria.

c) Padrão: Definição de pesquisas individuais aos pesquisadores [P3]:

- **Propósito:**

Trabalhos de pesquisa individuais devem ser definidos a partir dos objetivos do projeto, visando delimitar o contexto de pesquisa de cada membro do projeto. Esses projetos individuais podem dar origem a trabalhos de conclusão de cursos, dissertações de mestrado e teses de doutorado, dependendo de sua complexidade. Um projeto de pesquisa individual deve identificar claramente seu propósito, problema a ser pesquisado e as atividades a serem executadas.

- **Problema:**

Conforme Brondani, Mello e Fontoura (2019), soluções complexas e inovadoras requerem a solução de desafios técnicos para seu desenvolvimento. Quanto mais inovador o projeto, mais complexas são as soluções, em muitos casos sendo necessário testar diferentes alternativas para a solução.

- **Solução:**

Devido à complexidade e às diferentes alternativas possíveis para a solução do problema, tópicos de pesquisa, com objetivos bem definidos, podem ser isolados de forma que possam gerar diferentes projetos de pesquisa a serem desenvolvidos por diferentes pesquisadores orientadores e estudantes de graduação e pós-graduação.

Brondani, Mello e Fontoura (2019) descrevem em seu estudo como solução deste problema a identificação de problemas de pesquisa que podem ser explorados em trabalhos de conclusão de curso e dissertações de mestrado realizados sob a supervisão de um pesquisador da área. Sugere-se que os resultados dessas pesquisas sejam avaliados e, caso sejam bem-sucedidos e atinjam os objetivos desejados, sejam incorporados ao software desenvolvido no projeto de pesquisa. Salienta-se que mais de uma pesquisa pode estar sendo realizada ao mesmo tempo visando um único objetivo. Após a finalização, identifica-se a que apresenta o melhor resultado para incorporar no projeto.

d) Padrão: Revisão constante do código fonte [P4]:

- **Propósito:**

Identificar possíveis problemas/bugs no código fonte, melhorando a manutenibilidade do código e, dessa forma, facilitando as alterações futuras.

- **Problema:**

Conforme identificado na RSL e nos trabalhos relacionados, equipes acadêmicas são compostas por pessoas com diferentes níveis de habilidades, e seus membros permanecem no projeto durante um curto período de tempo (alta rotatividade). Portanto, são constituídas por pessoas em formação, o que impacta na produção de um software de fácil alteração futura.

- **Solução**

Os autores Bernhart e Grechenig (2013) definem o método de revisão em duas situações distintas: forma contínua, de modo que torne escalável e forneça um feedback antecipado; e sempre que mudanças ocorrerem, compare as alterações no momento de aceitação da mudança.

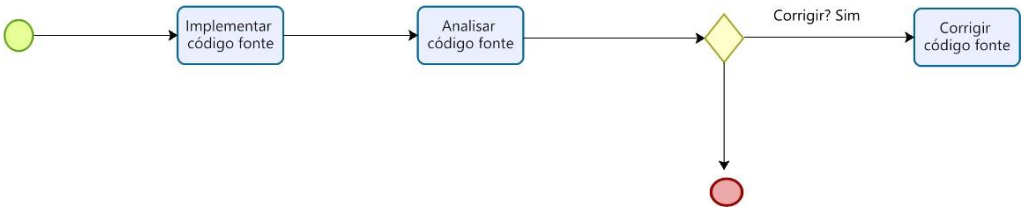
Sugere-se que esse processo, conforme Ram et al. (2018), Rahman e Roy (2017), possa ser realizado com auxílio de ferramentas automatizadas. Já Brondani, Mello e Fontoura (2019) descrevem que antes das entregas formais, contenha períodos de refatoração do código, visando melhorar a legibilidade e a documentação, além de remover linhas de códigos desnecessárias.

Sendo assim, define-se um processo (Figura 4) para realizar atividades relacionadas a esse padrão, em um primeiro momento realiza-se a implementação do código, em seguida a análise e, se necessário, a correção do código fonte.

- **Dinâmica**

Na Figura 4, apresenta do método para realizar a revisão do código, essas tarefas devem ser realizadas durante todo o processo de desenvolvimento de software que envolva a implementação de código fonte.

Figura 4 — Dinâmica da revisão constante do código



Fonte: Autoria própria.

e) Padrão: Definir responsáveis pelo fluxo de comunicação, lado cliente e equipe [P5]:

- **Propósito:**

Definir um membro responsável da equipe e cliente, com objetivo de estabelecer um fluxo de informação diante da necessidade de dúvidas/respostas com relação ao desenvolvimento de software, de modo que as informações fluam sem conflitos.

- **Problema:**

Brondani, Mello e Fontoura (2019) descrevem que a comunicação hierarquizada pode ocasionar um atraso no desenvolvimento de software, uma vez que uma dúvida e sua resposta precisam passar por vários níveis nessa estrutura de comunicação.

Verifica-se, também, que a maior parte de projetos de software não levam em consideração o envolvimento do usuário/cliente no processo, na maioria das vezes, isso ocorre

apenas ao final do projeto (DIAS; KODIKARA; EKANAYAKA 2014)(CERECI; KARAKAYA, 2018)(ANDRADE et al., 2017).

- **Solução:**

Diante do problema identificado de comunicação, considera-se necessário estabelecer um responsável para sanar as dúvidas abordadas tanto pela equipe quanto pelo cliente, centrando todas as informações a essa pessoa (BRONDANI; MELLO; FONTOURA, 2019). Dessa forma, torna o gerenciamento das informações de modo bem definido e organizado, permitindo eficácia na transmissão da comunicação (MONTEIRO; ALENCAR, 2007), e, assim, melhorando o fluxo de informação.

f) Padrão: Trabalhadores contratados em tempo integral [P6]:

- **Propósito:**

Definir vínculos de profissionais em tempo integral para gerenciar as atividades do desenvolvimento de software. Esses profissionais são responsáveis por manter o histórico do projeto e repassar o conhecimento aos membros da equipe, minimizando os efeitos da rotatividade entre os membros da equipe.

- **Problema:**

Equipes acadêmicas possuem muitos membros que são estudantes da universidade e permanecem no projeto durante a realização de um curso. Portanto, esses alunos conciliam as atividades acadêmicas do curso com as atividades do projeto.

Além disso, segundo Andrade et al. (2017), frequentemente, membros do projeto participam de pesquisas para gerar publicações de artigos científicos, participando de congressos e conferências em busca de soluções para os desafios propostos. Assim como, trabalhos (CERECI; KARAKAYA, 2018)(BRONDANI; MELLO; FONTOURA, 2019) exploram a alta rotatividade de pessoal e a participação de grande parte da equipe em tempo parcial.

- **Solução:**

Com intuito de minimizar os riscos de alta rotatividade e o grande número de trabalhadores em tempo parcial e inexperientes, Brondani, Mello e Fontoura (2019), Andrade et al. (2017) sugerem a contratação de profissionais experientes, em regime de trabalho integral, visando o gerenciamento dos membros da equipe e a continuidade no projeto.

g) Padrão: Trabalho colaborativo [P7]:

- **Propósito:**

Permitir um ambiente de desenvolvimento de software colaborativo com intuito de complementar as capacidades e conhecimentos individuais, minimizando falhas e, consequentemente, produzindo melhores resultados com relação ao processo de desenvolvimento de um produto de software.

- **Problema:**

A alta rotatividade dos membros da equipe pode ocasionar a perda de conhecimento em relação a avanços realizados em problemas de pesquisa e sobre o conhecimento do domínio de negócio e requisitos de software.

A baixa frequência de realização de reuniões e a dedicação em tempo parcial de grande parte da equipe podem gerar problemas de comunicação entre os membros do projeto e destes com o cliente.

Equipes acadêmicas são formadas por muitos estudantes de cursos de graduação e pós-graduação, o que resulta em níveis de habilidade divergente entre os membros da equipe e alta rotatividade, pois estes permanecem no projeto enquanto estão fazendo seus cursos (BRONDANI; MELLO; FONTOURA, 2019)(CERECI; KARAKAYA, 2018).

- **Solução:**

Segundo Brondani, Mello e Fontoura (2019), os membros da equipe podem aprender uns com os outros, compartilhando informações sobre o domínio de negócio e requisitos e, dessa forma, disseminando o conhecimento sobre o sistema desenvolvido.

O trabalho colaborativo pode ser realizado por meio de sistemas de controle de versão, ferramentas de modelagem e gerenciamento colaborativo. Dessa forma, sugere-se promover a integração entre a equipe, explorando, também, mecanismos de comunicação, tornando a relação mais dinâmica.

h) Padrão: Revisão de relatórios e artigos antes da publicação [P8]:

- **Propósito:**

Estabelecer acordos com relação aos objetivos e visões das diferentes instituições, e, assim, possibilitar que a instituição parceira tenha conhecimento e esteja a par das informações a serem publicadas, permitindo, então, que sejam “autorizadas” a serem disponibilizadas de modo público.

- **Problema:**

Considerando o ambiente de desenvolvimento de projeto em parceria com a academia, Andrade et al. (2017) identificam que existem diferentes interesses e atitudes entre a academia e a indústria em relação à publicação dos resultados do projeto. A indústria não demonstra interesse com relação a publicação de artigos científicos, devido à confiabilidade de informações estratégicas. Já as universidades precisam das publicações para melhorar seus indicadores de produtividade (ANDRADE et al. 2017).

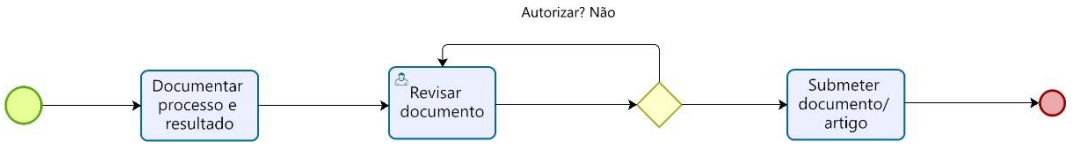
- **Solução:**

Andrade et al. (2017) sugerem como solução deste problema a definição de um acordo informal, no qual a academia deve enviar previamente os trabalhos e artigos produzidos no âmbito do projeto para revisão.

- **Dinâmica**

Conforme a solução do problema, define-se a necessidade de revisão do documento e deste ser autorizado pela instituição parceira antes da submissão para algum evento/revista/conferência/banca. Na Figura 5 pode ser visualizado o fluxo de execução das atividades propostas.

Figura 5 — Dinâmica da revisão de relatórios e artigos.



Fonte: Autoria própria..

i) Padrão: Reuniões periódicas [P9]:

- **Propósito:**

Adotar reuniões de acompanhamento periódicas, visando melhorar a comunicação entre a equipe, auxiliar na definição de metas e objetivos e o acompanhamento destes.

- **Problema:**

Verifica-se que à medida que a equipe se reúne com menor frequência, os colaboradores possuem menos informações sobre o status do projeto e o andamento das atividades, afetando a comunicação entre os envolvidos no projeto e o controle do projeto (CERECI; KARAKAYA, 2018).

- **Solução:**

A adoção de reuniões periódicas entre os membros da equipe do projeto permite ter uma visão em relação ao progresso do projeto, o feedback contínuo e o reporte de impedimentos que estão impactando a realização das atividades pelos membros da equipe (ANDRADE et al. 2017). A periodicidade irá depender da disponibilidade da equipe, mas o ideal é que sejam em intervalos bem curtos.

j) Padrão: Definir um plano de trabalho [P10]:

- **Propósito:**

Especificar os objetivos específicos do projeto, além de estabelecer responsabilidades, prazos e orçamentos, permitir estabelecer os termos de garantia com relação ao produto de software.

- **Problema:**

As mudanças com relação ao plano de trabalho, durante a execução do projeto, resultam em atraso no desenvolvimento do software e no cronograma, resultando em custos adicionais (ANDRADE et al. 2017), assim como problemas relacionados à ausência ou atrasos com relação a respostas sobre o que foi/deve ser desenvolvido, resultando em atrasos no desenvolvimento de software, assim como afetam a motivação da equipe.

Além da necessidade de cláusulas com relação a confidencialidade e consequentemente publicação dos dados e resultados, visto que cada instituição possui uma visão diferente com relação aos resultados obtidos sobre o produto de software (ANDRADE et al. 2017).

- **Solução:**

O documento de plano de trabalho deve conter as diversas funções no projeto e seus respectivos responsáveis e o alinhamento sobre entregas periódicas do produto de desenvolvimento. Após seu desenvolvimento, deve ser aprovado por todas as partes (ANDRADE et al. 2017). Verifica-se que, por meio de um plano de trabalho, é possível estabelecer o planejamento do projeto, evitando possíveis mal-entendidos e auxiliando na comunicação dos envolvidos no projeto.

Como forma de solucionar o problema da confidencialidade, o plano do trabalho pode prever os direitos de cada instituição, especificando os autores envolvidos na elaboração do software, assim, mantendo a propriedade intelectual que é de interesse do contexto acadêmico e transferindo os direitos comerciais de uso.

Além disso, a documentação pode auxiliar a equipe com relação à tomada de decisão, considerando as diretrizes institucionais que envolvem as instituições, bem como auxiliar no processo de comunicação, uma vez que possui direitos e deveres de cada instituição bem-definidos.

Considera-se que essa documentação dos padrões de software, por meio da literatura, permitirá verificar a validade de tais padrões, aplicados no contexto prático de desenvolvimento de software, a partir da experiência de profissionais com vivência em projetos realizados em parceria com outras instituições. Essa validação é possível identificar no próximo capítulo.

7 VALIDAÇÃO DO ESTUDO

Para validação deste trabalho, definiu-se como método de validação a realização de um estudo de caso com uma investigação do tipo empírica, a fim de realizar uma busca de dados por meio da experiência, ou vivência do participante.

O estudo de caso, segundo Yin (2005), buscou verificar particularidade e complexidade de um caso, resultando na compreensão de suas atividades dentro de suas circunstâncias. Pode-se definir que, na área de Engenharia de Software, o método empírico de estudo de caso é uma importante avaliação de métodos, auxiliando o pesquisador na análise para avaliação e validação dos resultados (SJOBERG; DYBA; JORGENSEN 2007).

Dessa forma, conforme Kitchenham e Pfleeger (2002), define-se que o questionário é um método de coleta de dados quantitativo e qualitativo. Partindo dessa perspectiva, define-se que a validação do estudo foi baseada na disponibilização de um questionário para os profissionais que possuem experiência com projetos de desenvolvimento de software em parceria com outras instituições.

Com base na literatura, Wohlin, Höst, Henningsson (2003), Runeson e M. Höst (2009), foram definidas as seguintes etapas para a realização do estudo de caso:

1 Concepção e projeto: definição dos objetivos e projeto do estudo de caso;
2 Preparação para coleta de dados: definição dos procedimentos para coleta de dados;
3 Coleta: execução da abordagem com o apoio da ferramenta e coleta de dados;
4 Análise e relato dos dados coletados: análise dos dados coletados e reporte dos resultados.

1) Concepção e projeto:

Nesta fase, definiu-se o projeto em que o estudo de caso foi realizado, juntamente com os objetivos, as hipóteses de pesquisa e como essas hipóteses seriam avaliadas e os resultados obtidos.

O estudo de caso dessa pesquisa visa verificar a validade dos padrões de software elaborados a partir da literatura, relacionando com a vivência/experiência prática de desenvolvimento de software, considerando o ambiente de colaboração da universidade com empresa e/ou governo.

Considerando o objetivo dessa pesquisa, assim como relacionando o problema do estudo, o qual identifica-se a carência de metodologias voltadas ao desenvolvimento de software em um ambiente universitário em cooperação com outras instituições que atenda aos problemas evidenciados nesse contexto.

Diante dessa perspectiva, define-se as hipóteses que irão nortear a validação deste estudo, dessa forma, foi elaborado tais hipóteses com base na documentação dos padrões documentados nessa pesquisa. Para cada problema identificado associado ao padrão descrito, relacionou-se uma circunstância, abordando o contexto prático de processo de software, conforme Quadro 3.

A identificação de cada item logo abaixo refere-se aos seguintes itens: padrão [P1], problema [PR1] e hipótese [H1], sucessivamente.

Quadro 3 — Definição de hipóteses com relação aos padrões de software

Padrão	Problema	Hipóteses
[P1] Elaborar diagramas de processo de negócio	[PR1] Problemas de comunicação	[H1] Adotar diagramas de processo de negócio facilita a comunicação da equipe com os stakeholders.

	[PR2] Desconhecimento do domínio da aplicação	[H2] A elaboração de diagramas de negócio auxilia na compreensão do domínio da aplicação.
[P2] Definir um processo iterativo e incremental	[PR3] Instabilidade nos requisitos	[H3] O uso de um processo iterativo e incremental pode minimizar o número de solicitações de mudanças de requisitos.
[P3] Definição de pesquisas individuais aos pesquisadores	[PR4] Complexidade das soluções	[H4] Os trabalhos de pesquisas individuais podem contribuir com soluções para sistemas complexos.
[P4] Revisão constante do código fonte	[PR5] Retrabalho	[H5] A revisão constante do código pode melhorar a legibilidade e documentação do código e, dessa forma, minimizar o retrabalho.
[P5] Definir responsáveis pelo fluxo de comunicação, lado cliente e equipe	[PR6] Problemas de comunicação	[H6] A designação responsável pelo fluxo de informação pode resolver problemas de comunicação.
	[PR7] Pouco contato com clientes/usuários	[H7] Os responsáveis pelo fluxo de comunicação

		auxiliam a sanar dúvidas sempre que necessário, aumentando o contato do cliente com a equipe, vice- versa.
[P6]Trabalhadores contratados em	[PR8] Alta rotatividade da equipe	[H8] Trabalhadores em tempo integral auxiliam no gerenciamento dos membros da equipe e do histórico e compartilhamento das informações, permitindo continuidade no projeto.

tempo integral	[P9] Disponibilidade em tempo parcial e inexperiência	[H9] Gerenciamento de trabalhadores em tempo integral permite que colaboradores com mais experiência compartilhem sobre o domínio do projeto.
[P7] Trabalho colaborativo	[P10] Alta rotatividade da equipe	[H10] Trabalho colaborativo permite que todos tenham informação com relação ao projeto de software.
	[P11] Desconhecimento do domínio da aplicação	[H11] A partir do trabalho colaborativo, é possível o compartilhamento sobre informações do domínio da aplicação, mantendo a equipe com o mesmo nível de conhecimento em relação ao desenvolvimento do projeto.
	[P12] Trabalhadores com diferentes habilidades	[H12] Trabalho colaborativo permite que profissionais mais experientes compartilhem seus conhecimentos de modo que capacite os colaboradores e melhore o nível de habilidade.

[P8] Revisão de relatórios e artigos antes da publicação	[PR13] Dificuldade de divulgar os resultados da pesquisa devido a non-disclosure agreements	[H13] A revisão de relatórios e artigos antes da publicação permite análise e autorização das informações publicadas.
	[PR14] Visões/objetivos divergentes	[H14] O acordo com relação à publicação das informações permite estabelecer diretrizes sobre os resultados do produto de software, considerando cada instituição.
[P9] Reuniões periódicas	[PR15] Problemas de comunicação	[H15] As reuniões com maior frequência ajudam na comunicação no projeto.
	[PR16] Diretrizes contratuais	[H16] Reuniões periódicas facilitam sobre questões de diretrizes contratuais.
[P10] Definir um plano de trabalho	[PR17] Problemas de comunicação	[H17] Realizar a definição do processo de trabalho de software permite obter uma comunicação eficaz no projeto.
	[PR18] Divisão de responsabilidades	[H18] O problema com relação à divisão de responsabilidades pode ser resolvido por meio da definição de um plano de trabalho.
	[PR19] Complexidade das soluções	[H19] Plano de trabalho auxilia no contexto da complexidade das soluções de software.
	[P20] Dificuldade de comercialização de produtos	[H20] No plano de trabalho, é possível definir questões referentes à comercialização de produtos.
	[P21] Diretrizes contratuais	[H21] As diretrizes contratuais podem ser definidas por meio de um plano de trabalho.

Fonte: Autoria própria.

2) Preparação para coleta de dados:

Nessa etapa definiu-se o método para realização de coleta dos dados, diante do planejamento do estudo de caso. Dado esse cenário, define-se como método de coleta dos dados a partir do questionário por meio do Google Forms, permitindo, então, melhor compreensão sobre situação- problema diante da experiência/vivência do participante.

A formulação dos itens do questionário baseou-se no estudo bibliográfico realizado nesta pesquisa. Dessa forma, no questionário, irá conter a descrição de cada padrão, explicando o(s) problema(s) relacionado(s), a solução como sugestão e, quando necessário, a visualização da dinâmica do padrão de software, e, por conseguinte, a apresentação da hipótese.

A fim de proteger a privacidade dos entrevistados, a informação dos dados coletados será usada apenas para propósito de validação da pesquisa. Assim, garantindo que a identidade de cada participante não será divulgada.

Diante da definição de diferentes cenários de análise relacionados às hipóteses estabelecidas na validação da pesquisa, define-se que, com intuito de classificar a veracidade de cada padrão conforme o contexto prático, será utilizada a Escala de Likert, ou seja, varia de: discordo totalmente a concordo totalmente (LIKERT, 1932). Válido registrar, também, ao final do questionário, que o participante terá a liberdade por meio do seu conhecimento e experiência profissional de descrever exemplos que facilitem e relacione com suas respostas.

O público alvo classifica-se como profissionais da área de desenvolvimento de software, considerando os seguintes requisitos, o participante deve: trabalhar ou ter trabalhado em uma das instituições que constituem a tríplice hélice (universidade/empresa/governo); ter experiência de uma equipe de desenvolvimento de software; ter a vivência de desenvolvimento em projetos realizados em parceria universidade e empresa ou universidade e governo.

A seleção dos participantes realizou-se por meio de uma amostragem por conveniênica, o qual selecionou os participantes a partir da busca de projetos relacionados ao tema da pesquisa, assim como identificou-se projetos de conhecimento da autora do trabalho. A partir desses critérios de seleção, realizou-se um convite via e-mail para o grupo de pessoas, solicitando a participação na validação da pesquisa, o envio foi realizado 10 colaboradores e teve retorno de 8 respondentes.

Finalizado a etapa de planejamento para coleta de dados, a próxima fase refere-se à organização de todo material coletado, utilizando ferramentas de apoio para auxiliar esse processo.

3) Coleta:

Após a descrição da preparação para coleta de dados, nesse tópico será apresentado como será a execução dessa coleta de informações.

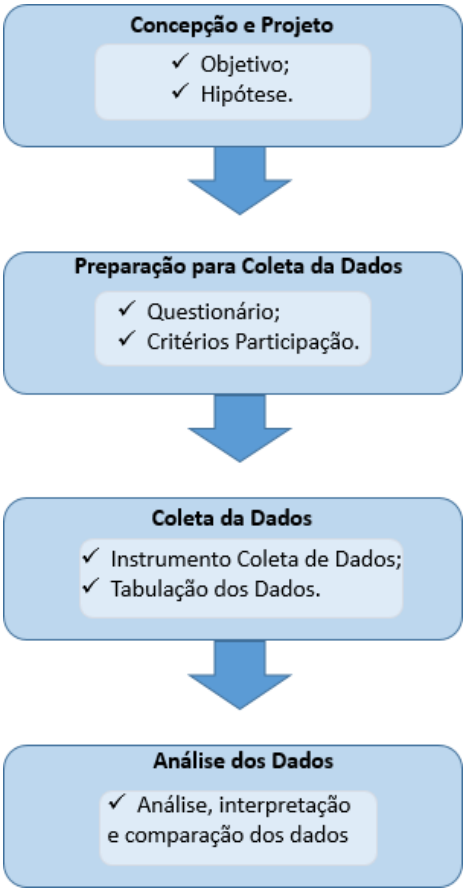
Dessa forma, o questionário será composto por perguntas fechadas estabelecidas por meio de uma escala e uma questão aberta descritiva. Após a coleta dos dados, os mesmos serão reportados em uma tabela, relacionando a resposta de cada participante com cada cenário descrito, utilizando o recurso de planilha do Google Forms. Ao finalizar a coleta de dados por meio do questionário, será realizado o processo de análise dos dados.

4) Análise:

A partir da etapa de coleta das informações, a próxima etapa corresponde à tarefa de análise dos dados. Dessa forma, utilizando como objeto de estudo as respostas do questionário, será realizado a análise do contexto, permitindo a identificação de dados e de informações relevantes à pesquisa. Portanto, essa fase relaciona-se com a estratégia de cruzamento das informações coletadas com o referencial desenvolvido nesse estudo, e, assim, permitindo realizar uma comparação/validação dos padrões de software identificados na base literária com relação à prática de desenvolvimento de software em cooperação, descrito pelos entrevistados.

A descrição dessas atividades pode ser verificada na Figura 6:

Figura 6 — Descrição das atividades de validação



Fonte: Autoria própria.

7.1 RESULTADOS E DISCUSSÕES

De acordo com o questionário aplicado, foi possível coletar e observar alguns resultados, tal descrição dos dados e análise são descritos nas próximas seções.

7.1.1 Público alvo

A partir das questões adotadas na primeira sessão do questionário, foi possível coletar informações pessoais dos participantes. O questionário teve participação de 8 respondentes, o qual nas questões iniciais foi possível traçar o perfil do público alvo, com intuito de compreender o sobre cada sujeito e relacionar com o contexto da pesquisa.

Dessa forma, realizou-se três questionamentos, sendo uma do tipo resposta única e duas como descritivas, tais questões são: (Q1) Qual a instituição você representa?; (Q2) Qual seu cargo/função na instituição? (Q3) Qual seu nível de experiência em anos de trabalho?. Por meio dessas questões, foi possível coletar informações e direcionar melhor a validação da pesquisa, na qual se identificou que a metade dos respondentes representa a universidade (50%) (Gráfico 1), assim como cargo/função define-se como: analista de sistema, desenvolvedor de software e pesquisadores e professor/pesquisador, e, por fim, observa-se que os anos de experiência varia entre 3 a 12 anos.

O Quadro 4 representa as informações coletadas no questionário.

Quadro 4 — Perfil dos participantes

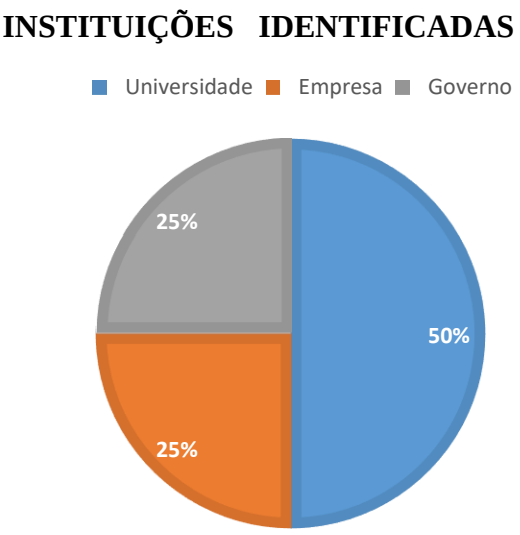
Participante	Instituição	Cargo/Função	Anos de Experiência
Participante 1 [PT1]	Universidade	Analista de Sistema/Pesquisador	4
Participante 2 [PT2]	Governo	Militar	4
Participante 3 [PT3]	Governo	Analista/Militar	6
Participante 4 [PT4]	Empresa	Desenvolvedor de Software	3
Participante 5 [PT5]	Universidade	Desenvolvedor de Software	5

48

Participante 6 [PT6]	Universidade	Pesquisador/Bolsista	3
Participante 7 [PT7]	Empresa	Coordenador de Qualidade/Scrum Master	6
Participante 8 [PT8]	Universidade	Professor/Pesquisador	12

Fonte: Autoria própria.

Gráfico 1 — Análise de instituições identificadas



Fonte: Autoria própria.

Todos os participantes já tiveram experiência em desenvolvimento de software, relacionando projetos em parceria com outras instituições, assim permitindo descrever sobre sua experiência no desenvolvimento de software de acordo com seu cargo/função.

7.1.2 Casos

No questionário disponibilizado, avaliou-se individualmente cada padrão identificado na literatura, com a seguinte estrutura: descrição do propósito do padrão, problema relacionado, sugestão da solução e quando disponível a dinâmica sobre como executar a atividade referente ao padrão de software. Partindo desse contexto, em seguida, buscou-se apresentar as hipóteses definidas para cada problema identificado.

Como forma de representar cada afirmação, houve a utilização da Escala de Likert, em que se atribuiu para cada uma um valor, como: concordo completamente (5 pontos); concordo parcialmente (4 pontos); não concordo nem discordo (3 pontos); discordo parcialmente (2 pontos); e discordo totalmente (1 ponto) (LIKERT, 1932). Cada valor atribuído auxilia na validação do padrão associado à experiência prática do sujeito.

A Figura 7 representa os dados do questionário tabulados conforme valor definido por cada participante, considerando cada caso disponibilizado. Na última coluna da tabela, como forma de auxiliar na análise dos dados, realizou-se como parâmetro estatístico a mediana das respostas de cada caso, observa-se que os valores variam entre 4.5 e 5.

O Gráfico 2, representa a distribuição de frequência das respostas do participante, considerando os valores atribuídos.

Verifica-se que a maior parte das hipóteses receberam valores de grau de relevância positivo, ou seja, concordo (4) e concordo totalmente (5). Dessa forma, para a maioria dos participantes, define-se que o padrão de software associado ao contexto descrito é válido ao contexto prático de desenvolvimento de software em projetos acadêmicos em parceria com outras instituições. Esse contexto, pode ser enfatizado nos primeiros casos [H1] [H2], o qual todos os participantes concordaram totalmente com o cenário apresentado.

Além disso, identificou-se que os participantes [PT3], [PT5] e [PT6], diferentes dos outros respondentes, atribuíram apenas valores entre concordo (4) e concordo totalmente (5), relacionando com exatidão os padrões de software e a prática de desenvolvimento de software.

Ainda sobre esse cenário, observa-se que [PT4] discorda totalmente de que no plano de trabalho é possível definir questões referente à comercialização de produtos, associando, então, sobre questões de direitos comerciais e a divergência entre as visões/objetivos de cada instituição.

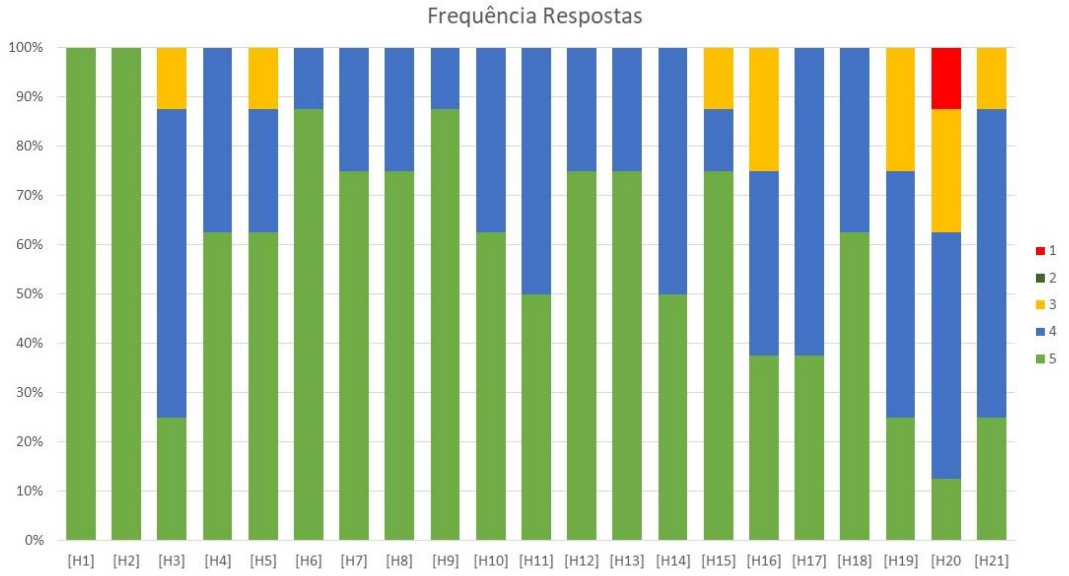
Partindo desse contexto, sugere-se como complementação ao padrão de software, conforme Andrade et al. (2017), a adoção de contrato com cessão dos direitos de cada instituição, especificando os autores envolvidos na elaboração do software, assim, mantendo a propriedade intelectual que é de interesse do contexto acadêmico e transferindo os direitos comerciais de uso para a instituição parceira. Válido lembrar que a Lei nº 9.609, dispõe da proteção da propriedade intelectual sobre software e sua comercialização no país (BRASIL, 1998).

Figura 7 — Tabulação dos dados a partir do questionário

Padrão	Problema	Hipótese	[PT1]	[PT2]	[PT3]	[PT4]	[PT5]	[PT6]	[PT7]	[PT8]	Mediana
[P1]	[PR1]	[H1]	5	5	5	5	5	5	5	5	5
	[PR2]	[H2]	5	5	5	5	5	5	5	5	5
[P2]	[PR3]	[H3]	4	4	4	3	5	4	4	5	4
[P3]	[PR4]	[H4]	5	4	5	4	4	5	5	5	5
[P4]	[PR5]	[H5]	5	3	4	4	5	5	5	5	5
[P5]	[PR6]	[H6]	5	5	5	5	5	5	4	5	5
	[PR7]	[H7]	5	5	4	4	5	5	5	5	5
[P6]	[PR8]	[H8]	5	4	5	4	5	5	5	5	5
	[PR9]	[H9]	5	5	5	4	5	5	5	5	5
[P7]	[PR10]	[H10]	4	5	4	4	5	5	5	5	5
	[PR11]	[H11]	4	4	5	4	5	5	4	5	4,5
	[PR12]	[H12]	5	4	5	4	5	5	5	5	5
[P8]	[PR13]	[H13]	5	4	5	5	5	5	5	5	5
	[PR14]	[H14]	5	4	4	4	4	5	5	5	4,5
[P9]	[PR15]	[H15]	5	5	3	4	5	5	5	5	5
	[PR16]	[H16]	5	4	4	3	4	5	3	5	4
[P10]	[PR17]	[H17]	4	4	4	4	5	5	4	5	4
	[PR18]	[H18]	4	5	5	4	5	5	4	5	5
	[PR19]	[H19]	3	4	4	3	4	5	4	5	4
	[PR20]	[H20]	3	4	3	1	4	5	4	4	4
	[PR21]	[H21]	5	4	3	4	4	5	4	4	4

Fonte: Autoria própria.

Gráfico 2 — Análise de frequência de respostas dos participantes.



Fonte: Autoria própria.

Ao final do questionário, disponibilizou-se uma seção possibilitando uma descrição sobre a experiência relatada, o [PT2] aponta a seguinte observação: a comunicação no projeto é imprescindível, permitindo a eficiência do fluxo de informações, evitando então o retrabalho e permitindo corresponder às expectativas dos patrocinadores. Ainda, ressalta que em desenvolvimento de software também é importante uma estrutura ou um responsável em gerenciar questões entre a instituição parceira e a equipe de projeto.

O [PT7] descreve o seguinte apontamento: o diagrama de negócio é essencial para a compreensão, clareza e fluidez dos processos, além do processo iterativo e incremental, que são bases das metodologias ágeis e o sucesso nos projetos.

Diante desses comentários, enfatiza-se a necessidade de definir um responsável pelo fluxo de comunicação, buscando gerenciar as informações do projeto de software e, assim, minimizar efeitos dos problemas relacionados a esse padrão de software. Além da importância de utilizar técnicas referentes à compreensão do domínio de negócio e de adotar uma metodologia que permite entregar um produto de software com valor agregado a cada novo incremento e, por meio de iteração, proporcionar uma melhora contínua ao desenvolvimento.

Após validar os padrões de software, coletar e analisar as respostas dos participantes, foi possível verificar que no geral os padrões de software, se encaixam com o contexto prático de desenvolvimento de software, permitindo, então, obter uma validação positiva com relação ao estudo realizado a partir da literatura, assim, sugerindo uma abordagem de processo de software para ser adotada em projetos acadêmicos em parceria com outras instituições.

8 CONSIDERAÇÕES FINAIS

Em síntese, verifica-se que realizar um estudo e sugerir práticas de software são temas de suma importância, visto que na literatura existe uma escassez de processos de software que considerem a universidade como ambiente de desenvolvimento em parceria com outras instituições.

Existe uma necessidade de uma metodologia de software aplicada a esse contexto, além do fato de que cada projeto possui suas características e particularidades, bem como envolve diferentes instituições com objetivos de software divergentes.

A partir deste cenário, realizou-se uma revisão sistemática de literatura, o qual buscou-se discutir a importância da adoção de métodos de processo de software adequados, conforme as características de cada projeto, bem como a possibilidade de combinação de diferentes abordagens.

Como resultado dessa pesquisa, evidenciou-se então a carência de modelos de processos definidos específicos para projetos que seguem o modelo “tríplice hélice”. Verificou-se também, que não há metodologia específica que possa cobrir as necessidades de todos os projetos de software, pois cada projeto e organização possui suas características específicas.

Além do fato, que a falta de processos específicos para projetos universitários é evidente, e as diferenças entre esse ambiente e as outras instituições exigem processos personalizados. Em contrapartida, existe um grande número de projetos utilizando práticas provenientes de mais de um método, explorando e relacionando as melhores práticas da metodologia ágil e orientada à planos, definidas como metodologias híbridas.

Salienta-se que com intuito de contribuir e expandir com o tema da pesquisa, a revisão foi publicada na International Conference on Enterprise Information Systems (ICEIS) (SILVA; FILHO; FONTOURA, 2020).

Por conseguinte, como desenvolvimento da pesquisa realizou-se uma análise de estudos, com intuito de identificar problemas presentes na universidade, no contexto desenvolvimento de software em parceria com outras instituições. Diante disso, após essa etapa, investigou-se e relaciou-se soluções para os determinados problemas encontrados, permitindo posteriormente a documentação dos padrões de software. Essa documentação foi realizada com base na literatura e assim proposto uma solução a ser adotado pela equipe de projeto.

Dessa forma, a partir deste processo metodológico, foi possível por meio da validação do estudo evidenciar as práticas de software aplicadas ao contexto prático de desenvolvimento de software em ambiente de cooperação. Considera-se que, após a aplicação do questionário e a análise das questões respondidas pelos participantes, comparou-se de forma positiva a aplicação das práticas descritas no catálogo com a experiência dos respondentes da pesquisa.

Os resultados deste estudo servirão para auxiliar a equipe de software na tomada de decisões com relação à seleção das práticas de software, além de permitir melhor gerenciamento dos recursos, atividades e artefatos que envolvem o desenvolvimento, bem como fornece orientação para projetos de software desenvolvidos em ambiente acadêmico em parceria com outras instituições.

Embora as práticas tenham sido propostas conforme contexto da pesquisa, os padrões de software podem ser utilizados em outros contextos de desenvolvimento, visto que apresentem os desafios/problemas identificados.

Como trabalhos futuros, propõem-se validar os padrões de software voltados para aplicação prática de desenvolvimento de software, buscando analisar e acrescentar novos itens e, assim, auxiliar a reduzir falhas em projetos de software.

REFERÊNCIAS

AMARAL, F. P. et al. O papel das ferramentas para sistematização de processos denegócios (BPMS). In: **ENCONTRO NACIONAL DE ENGENHARIA DE PRODUÇÃO**, 28., 2008, Rio de Janeiro.

AMBLER, S. W. **An introduction to process patterns**. Cambridge UniversityPress, 1998.

ANDRADE, R. M. C.; Lelli V.; Castro,R. N. S.; Santos, I. S. ***Fifteen years of industry and academia partnership: lessons learned from a brazilian researchgroup***. IEEE/ACM 4th International Workshop on Software Engineering Research and Industrial Practice (SER&IP), 2017, pp. 10-16, doi: 10.1109/SER-IP.2017..2.

AWAD, M. A. A comparison between agile and traditional software development methodologies. In **2nd National Conference on Applied Research**. Computer Science and Information Technology, 2015.

BARLOW, J.; GIBONEY M. J.; KEITH, D. W.; SHUETZLER, R. M. **Overview and guidance on agile development in large organizations**, Commun. Assoc.Inf. Syst., Vol. 29, n. 2, 2011.

BECK, K. **Extreme Programming Explained**.—Embrace Change. Addison-Wesley, 2000.

BECK, K., et al. Manifesto for Agile Software Development.Disponível em: https://agilemanifesto.org/history.html. Acesso em: Julho, 2021.

BECK, K, ANDRES, C. **Extreme Programming Explained**. Publisher: Addison-Wesley Professional; 2 edition, 2004.

BERNHART, M.; e GRECHENIG, T. On the understanding of programs withcontinuous code reviews. **IEEE International Conference on Program Comprehension**. 192-198. 10.1109/ICPC.2013.6613847, 2013.

BOEHM, B. **Get ready for agile methods, with care. computer**. 35. 64-69. 10.1109/2.976920, 2002.

BOEHM, B. E TURNER, R.. Balancing agility and discipline: a guide for theperplexed. Boston, MA: Addison Wesley, 2003.

BRASIL. Decreto N° 9.609, DE 19/02/1998. Regulamentada a Lei Brasileira sobre os Propriedade Intelectual de Programa de Computador, sua comercialização no País. **Diário Oficial da União,** Poder Executivo, Brasília, DF, 19 de fevereiro de 1998.

BRONDANI, C.; MELLO, O.; FONTOURA, L. A case study of a software development process model for SIS-ASTROS". 600-605. 10.18293/SEKE2019-098. 2019.

BROY M. Modelagem de Domínio e Engenharia de Domínio: Tarefas Chave em Engenharia de Requisitos. Em: Münch J., Schmid K. (eds) Perspectives on the Future of Software Engineering. Springer, Berlim, Heidelberg, 2013, https://doi.org/10.1007/978-3-642-37395-4_2

CERECI, I.; KARAKAYA, Z. Need for a software development methodology for research-based software projects. In **3rd International Conference on Computer Science and Engineering (UBMK)**, Sarajevo: pp. 648-651, 2018.

CONFORTO, E. C.; AMARAL, D. C.; SILVA, F. B.; REBENTISCH, E. Modelos Híbridos: Unindo Complexidade, Agilidade e Inovação, Revista MundoPM, 2015.

Coplien, J.; Schmidt, D. (eds.) **Pattern Languages of Program Design**, Reading-MA, Addison-Wesley, 1995.

COPLIEN, J. O. A generative development—process pattern language. Thepatterns handbooks: techniques, strategies, and applications. Cambridge University Press, USA, 243–300, 1998.

CRAWRFORD, L. Profiling the competent project manager. In: **Slevine, Cleland& Pinto (Edts), the frontiers of project management research** (pp. 151-176). Newton Square, PA: Project management institute, 2002.

DIAS, D. M. P.; KODIKARA, N. D.; JAYAWARDENA, M. The need for novel development methodologies for software projects in universities: a Sri Lanka case study. 2013. In: **International Journal of Future Computer and Communication** (January): 494–98,

DOS SANTOS, V.; ALVES, S.; CANEDO, E. D. Development methodologycase study: brazilian electoral justice. 2014. In **Iberian Conference on Information Systems and Technologies**, CISTI: 1–6.

ETZKOWITZ, H**. Academic-industry relations: a sociological paradigm for economic developmen**t. In: Leydersdorff, L.; Van Den Besslaar, P., Evolutionary economics and chaos theory: new directions in technology studies. London: Pinter Publishers, 1994.

ETZKOWITZ, H.; LEYDESDORFF, L. **The dynamics of innovation: from National Systems and ‘‘Mode 2’’ to a Triple Helix of university–industry–government relations**. Research Policy, 2000.

ETZKOWITZ, H.; ZHOU, C. **Hélice tríplice: inovação e empreende-dorismo universidade- indústria-governo**, Estudos Avançados, 31(90), 23-48. https://dx.doi.org/10.1590/s0103-40142017.3190003. 2017.

FITRIANI, W R.; RAHAYU, P.; SENSUSE, D. I. Challenges in agile software development: a systematic literature review. In **International Conference on Advanced Computer Science and Information Systems**, ICACSIS 2016: 155–64. 2017.

FORSELL, M. **Adicionando Análise de Domínio ao Método de Desenvolvimento de Software**. In: Harindranath G. et al. (eds) New Perspectiveson Information Systems Development. Springer, Boston, MA. https://doi.org/10.1007/978-1-4615-0595-2_15, 2002.

FOWLER, M. **Using agile software process com offshore development**. 2004. Disponível em: https://martinfowler.com/articles/agileOffshore.html.Acesso em: Julho, 2021.

FOWLER, M. The new methodology. Wuhan Univ. J. of Nat. Sci. 6, 12–24. 2005.

FUGGETTA, A. Software process: a roadmap. In **Proc. of The Future ofSoftware Engineering**, ICSE'2000, Ireland. 2000.

FUGGETTA, A.; Di NITTO, E. Software process; future of software engineering.FOSE 2014 - Proceedings. 10.1145/2593882.2593883, 2014.

GAROUSI, V et. al. Characterizing industry-academia collaborations in software engineering: evidence from 101 projects. **Empirical Software Engineering**. 24. 10.1007/s10664-019-09711-y, 2019.

GOMES, M.; COELHO, T.; GONÇALO, C. Tríplice hélice: a relação universidade-empresa em busca da inovação. **Revista Eletrônica de GestãoOrganizacional.** ISSN 1679-1827. 2014.

GUDWIN, R. R. **Engenharia de Software: uma visão prática**. 2° Edição. DCA-FEEC-UNICAMP, 2015.

KITCHENHAM, B.; PFLEEGER, S. Principles of survey research part 2:designing a survey. **SIGSOFT Softw. Eng**. Notes, 18–20. DOI:https://doi.org/10.1145/566493.566495, 2002.

KLUNDER, J.; HOHL, P.; FAZAL-BAQAIE, M.; KRUSCHE, S.; KÜPPER, S.; LINSSEN, O.; PRAUSE, C. HELENA Study: Reasons for Combining Agile and Traditional Software Development Approaches. 2017.

KRUTCHEN, P. **Contextualizing Agile Software Development**. Vancouver,BC, Canada Proc, 2013.

KUHRMANN, M. et al. Hybrid software development approaches in practice: a european perspective. In **IEEE Software** 36(4): 20–31, 2019.

LIKERT, R. **A technique for the measurement of attitudes**. Archives of Psychology, 22(140), 1-55. 1932.

MARINHO, M.; NOLL, J.; RICHARDSON I.; BEECHAM, S. Plan-Driven Approaches Are Alive and Kicking in Agile Global Software Development. In:**To appear, Proceedings of the International Symposium on Empirical Software Engineering and Measurement (ESEM)**, 2019.

MATTHIES, C., HUEGLE. J., DÜRSCHMID, T. E TEUSNER, R. (2019). "Attitudes, Beliefs, and Development Data Concerning Agile Software Development Practices." **IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)**, Montreal, QC, Canada, doi: 10.1109/ICSE-SEET.2019.00025.

MIKOSZ, V. M. **A relação universidade-empresa-governo no contexto do sistema nacional de ciência, tecnologia e inovação brasileiro: um estudo de caso dos**

mecanismos de cooperação e seus intervenientes em uma universidade pública. Dissertação (Mestrado em Planejamento e Governança Pública) – Programa de Pós-Graduação em Planejamento e Governança Pública –Universidade Tecnológica Federal do Paraná. Curitiba, 2017.

MINEIRO, A., SOUZA, D. L. & VIEIRA, K. C., CASTRO, C. E BRITO, M. J. (2019). "Da Hélice Tríplice A Quíntupla: Uma Revisão Sistemática". **Revista Economia & Gestão**. 18. 77-93. 10.5752/P.1984-6606.2018v18n51p77-93.

MONTEIRO, A.; ALENCAR, L. Análise dos problemas de comunicação emprojetos de desenvolvimento de software. **XXVII Encontro Nacional de Engenharia de Produção (ENEGEP)**, Foz do Iguaçu-PR, 2007.

PRESSMAN, R. S. Engenharia de Software: uma abordagem profissional. 7º Ed. Porto Alegre: AMGH. 2011.

PRIETO-DÍAZ, R. Análise de domínio: uma introdução. **SIGSOFT Softw. Eng**. Notes 15, 2 (abril de 1990), 47–54. DOI: https: //doi.org/10.1145/382296.382703.1990.

PROJECT MANAGEMENT INSTITUTE – **PMI. Guia PMBOK®: um guia para o conjunto de conhecimentos em gerenciamento de projetos**, Sextaedição, Pennsylvania: PMI. 2017.

RAHMAN, M. M; ROY, C. K. Impact of Continuous Integration on Code Reviews. **IEEE/ACM 14th International Conference on Mining SoftwareRepositories (MSR)**, pp. 499-502, doi: 10.1109/MSR.2017.39. 2017.

RAM, A.; SAWANT, A. A; CASTELLUCCIO, M.; BACCHELLI, A. What makes a code change easier to review? an empirical investigation on code change reviewability. In **Proc. ESEC/FSE**, 2018.

RIESENER, M.; DÖLLE,C.; AYS, J.; AYS. J. L.. Hybridization of development projects through process-related combination of agile and plan-driven approaches". **In IEEE International Conference on Industrial Engineering and Engineering Management**, 2019.

RUNESON, P.; HÖST, M. Guidelines for conducting and reporting case study research in software engineering, **Empir. Softw. Eng**., vol. 14, no. 2, pp. 131–164, 2009, doi: 10.1007/s10664-008-9102-8., 2009.

SARPONG, D.; ABD RAZAK, A., ALEXANDER, E.; MEISSNER, D. Organizing practices of university, industry and government that facilitate (or impede) the transition to a hybrid triple helix model of innovation. Technological Forecasting and Social Change. 123. 10.1016/j.techfore.2015.11.032. 2016.

SCHUMACHER, M.; FERNANDEZ-BUGLIONI, E.; HIBERTSON, D.; BUSCHAMANN, F.; SOMMERLAD, P**. Security Patterns: IntegratingSecurity and Systems Engineering**. ISBN: 978-0-470-85884-4. 2006.

SCHWABER, K.; SUTHERLAND, J**. O guia do scrum: o guia definitivo parao Scrum as regras do jogo**. 2020. Disponível em: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-PortugueseBR.pdf. Acesso em: 12 nov. 21.

SILVA, C.; FILHO, E.; FONTOURA, L. Software processes used in university, government, and industry: a systematic review. In **Proceedings of the 22nd International Conference on Enterprise Information Systems - (ICEIS)** - Volume 2: ICEIS, ISBN 978-989-758-423-7, pages 314-321. DOI: 10.5220/0009419203140321, 2020.

SJOBERG, D. I. K.; DYBA, T.; JORGENSEN, M. The future of empirical methods in software engineering research, **In Future of Software Engineering (FOSE '07)**. IEEE Computer Society, Washington, DC, USA, pp. 358-378, 2007.

SPUNDAK, M. Mixed Agile/Traditional Project Management Methodology – Reality or Illusion?. **In Procedia - Social and Behavioral Sciences** 119: 939–48. http://dx.doi.org/10.1016/j.sbspro.2014.03.105. 2014.

THEOCHARIS, G.; KUHRMANN, M.; MÜNCH, J.: DIEBOLD, P**. Is Water-Scrum-Fall Reality? On the Use of Agile and Traditional Development Practices. Lecture Notes in Computer Science**. 9459. 149-166. 10.1007/978-3-319-26844-6_11. 2015.

TURK, D.; FRANCE, R.; RUMPE, B. Limitations of agile software processes". in third international conference on extreme programming and flexible processes.In **Software Engineering**, XP2002, Italy. 2014.

VIJAYASARATHY, L. R.; BUTLER, C. W. Choice of software development methodologies: do organizational, project, and team characteristics matter?" In**IEEE Software** 33(5): 86–94. 2016.

WAHONO, R. On the requirements pattern of software engineering. **Proceeding Temu Ilmiah**. XI. ISSN 0918-7685.2008.

WOHLIN C.; HÖST M.; HENNINGSSON K. **Empirical research methods in software engineering**. Springer. https://doi.org/10.1007/978-3-540-45143-3_2. 2003.

WOHLIN C.; MENDES E.; FELIZARDO, K.R., Kalinowski, K.R. Challengesand recommendations to publishing and using credible evidence in software engineering. Inf. **Softw. Technol**. 127 106366. 2020.

XU, P.; RAMESH, B. Using process tailoring to manage software development challenges. In **IT Professional**. doi: 10.1109/MITP.2008.81.2008.

YIN. R. K. **Estudo de caso: planejamento e métodos**. 3 ed., Porto Alegre:Bookman. 2005.