PUCRS

ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

ANIELLE SEVERO LISBOA

ENSINANDO ENGENHARIA DE SOFTWARE ONLINE PARA PESSOAS EM SITUAÇÃO DE VULNERABILIDADE SOCIAL

Porto Alegre
2021

PÓS-GRADUAÇÃO - *STRICTO SENSU*

Pontifícia Universidade Católica
do Rio Grande do Sul

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ENSINANDO ENGENHARIA DE SOFTWARE ONLINE PARA PESSOAS EM SITUAÇÃO DE VULNERABILIDADE SOCIAL

ANIELLE SEVERO LISBOA

Dissertação apresentada como requisito parcial à obtenção do grau de Mestra em Ciência da Computação na Pontifícia Universidade Católica do Rio Grande do Sul.

Orientador: Prof. Dr. Rafael Prikladnicki

Porto Alegre
2021

Ficha Catalográfica

L769e Lisboa, Anielle Severo

Ensinando Engenharia de Software online para pessoas em situação de vulnerabilidade social / Anielle Severo Lisboa. – 2021.

164.

Dissertação (Mestrado) – Programa de Pós-Graduação em Ciência da Computação, PUCRS.

Orientador: Prof. Dr. Rafael Prikladnicki.

1. Diversidade social. 2. Engenharia de Software. 3. Ensino de Engenharia de Software. 4. Pandemia. 5. Vulnerabilidade Social. I. Prikladnicki, Rafael. II. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da PUCRS com os dados fornecidos pelo(a) autor(a).

Bibliotecária responsável: Clarissa Jesinska Selbach CRB-10/2051

ANIELLE SEVERO LISBOA

ENSINANDO ENGENHARIA DE SOFTWARE ONLINE PARA PESSOAS EM SITUAÇÃO DE VULNERABILIDADE SOCIAL

Dissertação apresentada como requisito parcial para obtenção do grau de Mestra em Ciência da Computação do Programa de Pós-Graduação em Ciência da Computação, Escola Politécnica da Pontifícia Universidade Católica do Rio Grande do Sul.

Aprovado(a) em 26 de Fevereiro de 2021.

BANCA EXAMINADORA:

Profª. Drª. Tayana Uchôa Conte (PPGI/UFAM)

Prof. Dr. Afonso Henrique Corrêa de Sales (PPGCC/PUCRS)

Prof. Dr. Rafael Prikladnicki (PPGCC/PUCRS - Orientador)

DEDICATÓRIA

Dedico este trabalho a minha mãe Vera Lúcia, aos meus irmãos Dion Maier, Francieli, Paola e Izadora, aos meu sobrinhos Pietro, Lara e Antoni, a minha cunhada Aline, meu noivo Nailton e a minha amada vó Eulália (*in memorian*).

"Todos os nossos sonhos podem-se realizar se tivermos a coragem de persegui-los."
(Walt Disney)

AGRADECIMENTOS

Primeiramente eu gostaria de agradecer à Deus por toda força, determinação e suporte até aqui. Por colocar pessoas maravilhosas na minha vida no período acadêmico, por terem me dado a força de vontade, as habilidades e as ferramentas que me são necessárias para cumprir minha missão nesta vida.

Agradeço a minha mãe Vera Lúcia, minha avó Eulália (*in memorian*) e minha tia Eva Maria, por me darem todo o apoio que eu precisei para poder realizar este trabalho. Muito obrigada por entenderem e aceitarem a minha ausência em momentos com a família, e por sempre me incentivarem a persistir. Vocês são tudo pra mim.

Ao meu orientador, Prof. Dr. Rafael Prikladnicki, que me ofereceu um trabalho norteado, me apontado um mundo de possibilidades e apoiando minhas decisões. Nossas reuniões, curtas, mas objetivas, me abriram portas e caminhos valiosos. Agradeço a generosidade ao compartilhar seus conhecimentos e discutir ideias e soluções comigo. Muito obrigada por confiar em mim!

Às minhas irmãs Francieli, Paola e Izadora, meu irmão Dion Maier, minha cunhada Aline e meus Sobrinhos, Pietro, Lara e Antoni. Eu amo vocês!

À Prof.ª. Drª. Sabrina Marczak, que me recebeu de braços abertos no grupo e me apresentou o mundo da pesquisa científica e ao professor Rafael, meu orientador.

Aos amigos que conquistei no mestrado pra vida toda: Daniela Kuinchtner, Karina Kohl, Michelle Miranda, Cássio Trindade, Olimar Borges, Mateus Colissi, Daniel Reyes e Flavielle Blanco. Muito obrigada por todos os momentos que passamos juntos, nos risos, nos almoços, no café, nas lágrimas e grupos de estudos para as disciplinas obrigatórias. Vocês foram essenciais, meu muito obrigada!

E por último e não menos importante, meu namorado que sempre me apoiou e me incentivou durante toda essa jornada. Eu te amo, Nailton Andrade Júnior. Muito obrigada por tudo! Muito obrigada também, minha sogra, Francisca, meu Sogro Nailton e minha cunhada Nathalia.

ENSINANDO ENGENHARIA DE SOFTWARE ONLINE PARA PESSOAS EM SITUAÇÃO DE VULNERABILIDADE SOCIAL

RESUMO

O software é essencial nos ambientes de mercado heterogêneos e nos mais variados domínios de conhecimento. O desenvolvimento de software como produto tem como base a Engenharia de Software. Ela faz parte da grade curricular de um curso de graduação na área de Ciência da Computação e em alguns casos é o próprio curso. O ensino de Engenharia de Software é necessário para que uma futura engenheira de software possa desenvolver da melhor maneira possível as atividades envolvidas neste processo: as especificações de requisitos, adoção de padrões e metodologias, o projeto, o desenvolvimento, testes e manutenção do software. A diversidade social é caracterizada por pessoas diferentes em relação a raças, etnias, crenças religiosas, status socioeconômico, idioma, origem geográfica, gênero e / ou orientação sexual. A vulnerabilidade social relaciona-se com situação do status socioeconômico de grupos de pessoas com poucos recursos financeiros, educação, de moradia e ou acesso a oportunidades para seu desenvolvimento enquanto cidadão. Pessoas em situação de vulnerabilidade social possuem acesso limitado à educação, cursos e aperfeiçoamentos necessários para construir uma profissão. A pandemia da COVID-19 ampliou este cenário e grande parte das instituições de ensino e empresas optaram pela continuidade do trabalho, com atividades remotas. O objetivo desta pesquisa é, dentro de um contexto de estudos de diversidade, identificar as formas de ensino de engenharia de software considerando pessoas em situação de vulnerabilidade social. Foram executados um mapeamento sistemático da literatura para estudar diversidade em engenharia de software (estudo secundário) e uma revisão sistemática sobre Ensino de Engenharia de Software (estudo terciário). Também foi executado um estudo qualitativo, utilizando as técnicas de entrevistas semiestruturadas para coleta de dados e técnicas de análise de dados da *Grounded Theory*. Como resultado principal a contribuição deste tra-

balho é um conjunto de práticas de ensino de engenharia de Software online, considerando pessoas em situação de vulnerabilidade social.

Palavras-Chave: Diversidade Social, Engenharia de Software, ensino de Engenharia de Software, Pandemia, Vulnerabilidade Social.

TEACHING SOFTWARE ENGINEERING ONLINE TO PEOPLE IN SOCIALLY VULNERABLE SITUATIONS

ABSTRACT

Software is essential in heterogeneous market environments and the most diverse fields of knowledge. Software development is based on Software Engineering, which is part of the curriculum of the undergraduate course. Software Engineering disciplines cover all aspects of software production. Qualified teaching of Software Engineering is essential for future software engineers to follow software development techniques and processes satisfactorily, such as user specifications, standards, methodologies, validation, and maintenance. Social diversity is characterized by people of different races, ethnicities, religious beliefs, socioeconomic status, language, geographical origin, gender, and sexual orientation. Social vulnerability is related to the socioeconomic status of groups of people with few financial resources, education, housing, and/or access to opportunities for their development as a citizen. People in social vulnerability situations have limited access to education and courses to build a profession. In 2020, the COVID-19 Pandemic is affecting the progress of on-site work, classes, and activities. As a result, most educational institutions and companies choose to continue working with remote activities. In this research, we understand how Software Engineering teaching is conducted to people in social vulnerability conditions during the Pandemic. We executed two systematic mappings of the literature: the first one to study diversity in Software Engineering (secondary study) and the second one for teaching in Software Engineering (tertiary study). In the qualitative study, we use semi-structured interview techniques to collect and analyze Grounded Theory methodologies. The main contribution of our research is a set of teaching practices in software engineering in times of Pandemic for people in social vulnerability situations.

Keywords: Social Diversity in Software Engineering, Pandemic, Social Vulnerability, Teaching in Software Engineering.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

AER - Análise e Especificação de Requisitos

COVID-19 - *COrona virus Disease*

DS - Designer de Software

EBSE - *Evidence-based software engineering*

ES – Engenharia de Software

FC - Fundamentos da Computação

FME - Fundamentos da Matemática e Engenharia

FLOSS - *Free/Libre and Open Source Software*

GBL - *Game-based learning*

GSD - *Global Software Development*

GT – *Grounded Theory*

ID - Identificador

LGBTQIA+ - Lésbicas, Gays, Bissexuais, Transexuais, Queer, Intersexo, Assexual

MASO - Modelagem e Análise de Software

MOOCs - *Massive Open Online Course*

MSL – Mapeamento Sistemático da Literatura

OMS - Organização Mundial da Saúde

PBL - *Project-based learning*

PO – *Product Owner*

PRP - Prática Profissional

PS - Processo de Software

PUCRS – Pontifícia Universidade Católica do Rio Grande do Sul

QA - Analista de Qualidade

QP - Questão de Pesquisa

QS - Qualidade de Software

RSL – Revisão Sistemática da Literatura

SE 2014 - As Diretrizes Curriculares para Programas de Graduação em Engenharia de Software

SEEK - Conhecimento em Educação em Engenharia de Software

SG - Segurança

SGs - *Serius Games*

SWEBOK - *Guide to the Software Engineering Body of Knowledge*

TCLE - Termo de Consentimento Livre e Esclarecido

TDD - *Test-driven development*

TI - Tecnologia da Informação

UX - Designer de Experiência do Usuário

VVS - Verificação e Validação de Software

XP - *Extreme Programming*

SUMÁRIO

1. INTRODUÇÃO

O software é essencial nos ambientes de mercado heterogêneos e nos mais variados domínios de conhecimento. O desenvolvimento de software como produto tem como base a Engenharia de Software (ES) e, segundo Pressman [80] é conceituada como sendo a aplicação de uma abordagem sistemática, disciplinada e quantificável. Ela faz parte da grade curricular de um curso de graduação (em alguns casos é o próprio curso) e segundo Sommerville [93], são disciplinas que englobam todos os aspectos da produção de um software, desde os estágios iniciais como a especificação até a manutenção desse sistema.

As atividades da Engenharia de Software são colaborativas, realizadas por pessoas diversas e suas diferentes características. Idade, raça, gênero, cultural, social, etnia e entre outros [86]. Segundo Fleury [35] a diversidade é resultado da interação entre indivíduos com diferentes identidades e que convivem no mesmo sistema social. Os autores Taylor et al. [97] mencionam que na computação, a diversidade em uma equipe de desenvolvimento de software é importante para o desenvolvimento de produtos mais robustos para o mercado, em que as inclusões de pessoas diversas beneficiam as fases de desenvolvimento e a questão da representatividade na sociedade. Pieterse et al. [75] afirmam que as pesquisas sobre diversidade na ES mencionam que o assunto é importante, pois criam equipes melhores, com eficiência e agilidade.

De acordo com Santana [88] a diversidade social é caracterizada por pessoas de raças, etnias, crenças religiosas, status socioeconômico, idioma, origem geográfica, gênero e / ou orientação sexual distintas. Monteiro et al. [62] relaciona o termo vulnerabilidade social à situação do status socioeconômico de grupos de pessoas com poucos recursos financeiros, educação, de moradia e ou acesso a oportunidades para seu desenvolvimento enquanto cidadão. Pessoas em situação de vulnerabilidade social possuem acesso limitado à educação, capacitação e aperfeiçoamentos necessários para construir uma profissão [84].

A capacitação e a formação qualificada de profissionais diversos são cada vez mais necessárias na indústria de desenvolvimento de software. Beckman et al. [12] citam que especificamente no ensino de Engenharia de Software, a qualidade dos profissionais está diretamente relacionada à qualidade da educação, embora existam outros fatores que contribuem para isto. O ensino de Engenharia de Software pode beneficiar e fomentar a vida profissional e financeira de pessoas em situação de vulnerabilidade social. E isso pode impulsionar o processo de inserção, formando novos profissionais da engenharia de software para a indústria de desenvolvimento de software. Entretanto, devido à pandemia do coronavírus, as ações tomadas pelos governos foram principalmente aconselhar a população a ficarem em suas casas para impedir a propagação do vírus, e isso teve um impacto direto na vida das pessoas modificando o ensino para um formato virtual, de acordo com Stokes et al. [96].

O formato online impacta na maneira de ensinar e aprender devendo ser adaptadas às demandas da sociedade. Isso implica no uso de metodologias ativas de ensino e na introdução de tecnologias da informação e comunicação nos ambientes de aprendizagem [96]. Segundo Romero et al. [87] em situações específicas, como as vivenciadas durante a pandemia da COVID-19, a educação online e os dispositivos de comunicação tornaram-se muito importantes para a realização do processo de ensino-aprendizagem.

Nesse contexto e com a pesquisa na literatura sobre o assunto, justifica-se a realização da pesquisa devido a necessidade de contribuir com práticas de ensino online de Engenharia de Software para as pessoas em situação de vulnerabilidade social. Assim, a principal contribuição deste trabalho é a elaboração de um conjunto de práticas de ensino de Engenharia de Software online considerando as pessoas em situação de vulnerabilidade social.

A questão de pesquisa principal que norteou este estudo foi:

Como a Engenharia de Software é ensinada online para pessoas em situação de vulnerabilidade social?

Para responder à questão de pesquisa mencionada anteriormente, foram desenvolvidos o objetivo geral e específicos deste estudo, apresentada nas subseções a seguir.

1.1 Objetivos

O objetivo desta pesquisa é identificar as formas de Ensino online de Engenharia de software, considerando pessoas em situação de vulnerabilidade social. De forma a complementar o objetivo geral proposto, os seguintes objetivos específicos foram definidos:

- Identificar os conceitos relacionados a Engenharia de Software;
- Identificar os conceitos relacionados a Diversidade;
- Executar um estudo secundário de Mapeamento Sistemático da Literatura (MSL);
- Identificar os conceitos relacionados a Ensino de Engenharia de Software;
- Executar um estudo terciário de Mapeamento Sistemático da Literatura (MSL);
- Planejar e executar um estudo qualitativo usando entrevistas semiestruturadas para coletar dados;
- Executar as técnicas da Teoria Fundamentada nos dados (*Grounded Theory*) para analisar os dados;
- Propor um conjunto de práticas para ensinar Engenharia de Software online considerando pessoas em situação de vulnerabilidade social;

1.2 Motivação

Ainda que a diversidade já faça parte dos estudos da Antropologia há mais de 150 anos [78], o assunto só começou a ser abordado na literatura das organizações e da Administração nos últimos 30 anos [99].

Figura 1.1 – Quebra-cabeça da Diversidade
Fonte: ESTD (2020) https://www.esdt.com.au/diversity.html

A Figura 1.1 apresenta um quebra cabeça com algumas categorias da diversidade. Segundo Torres et al. [99] nesse período, a diversidade tem sido definida de diferentes formas, além de tomada como sinônimo para tantos outros conceitos (cota, ação afirmativa, inclusão).

1.2.1 Por que estudar sobre diversidade em Engenharia de Software?

Page [68] diz que as pessoas não podem classificar a diversidade como boa ou má, entretanto é necessário entender o que é diversidade. Ele menciona que diversidade cognitiva se refere aos estilos diferentes que as pessoas possuem para solucionar os problemas e podem oferecer perspectivas únicas porque pensam de maneira diferente e como elas são interpretadas, identificadas e resolvidas. A diversidade de identidade é determinada pela afiliação a um grupo social como gênero, cultura, etnia, religião, orientação sexual etc. [68]. A diversidade cognitiva está ligada nos melhores resultados. E estudos mostram

que a diversidade cognitiva é influenciada pela diversidade de identidade (por exemplo, gênero, raça, status socioeconômico etc.). A diversidade social, em contexto de vulnerabilidade social é caracterizada pela condição de grupos de pessoas que vivem em condições desfavoráveis como: condições precárias de moradia, ausência de um ambiente familiar e/ou educacional de qualidade.

A diversidade em Engenharia de Software apresenta múltiplas dimensões em relação a idade, etnia, gênero, pessoas com deficiência, pessoas em situação de vulnerabilidade social [107]. Isso acontece sobre a perspectiva de como as pessoas se percebem e como essas pessoas percebem os outros. Essas percepções afetam as interações entre os profissionais de uma equipe de desenvolvimento de software [70].

Estudar diversidade em Engenharia de Software é relacionar colaboração, compartilhamento de conhecimento, aspectos humanos em Engenharia de Software, indústria de desenvolvimento de software com desenvolvimento de produtos mais robustos para o mercado e representatividade na área. O assunto é importante, pois criam equipes melhores, devido suas experiências e vivencias.

1.2.2 Contexto de estudo

A autora deste estudo é mentora de um projeto de capacitação em engenharia de software [95]. Esse projeto de capacitação tem como principais características: ambiente de aprendizado, o desenvolvimento de profissionais iniciantes com as habilidades de engenharia de software, conforme a Figura 1.2. E, a inclusão da diversidade, com pessoas da diversidade de gênero, diversidade social (em contexto de vulnerabilidade social), diversidade étnico / racial, entre outros. Esse projeto é uma parceria entre uma instituição de ensino, uma empresa especializada em consultoria e desenvolvimento ágil de software, uma cooperativa e uma empresa de comunicação situadas no parque tecnológico desta instituição.

A Figura 1.2 apresenta as pessoas estudantes do projeto em uma cerimônia ágil de retrospectiva, facilitada por uma pessoa mentora. A Figura 1.3 apresenta as estudantes em uma sessão teórica e prática de aprendizado em Engenharia de Software, por um engenheiro de software vinculado à uma das empresas parceiras do projeto.

Esse projeto visa acelerar durante um período de 18 semanas o processo de aprendizagem com as habilidades da engenharia de software em estudantes de graduação, estudantes de curso técnico e de ensino regular [95]. O objetivo principal do projeto é ensinar e capacitar as estudantes possibilitando o acesso à indústria de desenvolvimento de software de maneira rápida e eficiente. Esse projeto utiliza uma técnica chamada de software Kaizen [33]. É um método de treinamento com a imersão em um ambiente real

Figura 1.2 – Estudantes de Engenharia de Software em uma Cerimônia ágil
Fonte: a autora (2019)

de alto desempenho, baseado nas metodologias ágeis e com apoio total de mentores e profissionais capacitados na área.

O processo de recrutamento dos estudantes para composição da equipe visa à inclusão e diversidade dos candidatos. Então, o processo considera a inclusão de equipes mais diversas: pessoas em situação de vulnerabilidade social, mulheres e homens cisgêneros, mulheres e homens transgêneros, travestis, LGBTQIA +, étnico / Racial, entre outras. O processo de seleção ocorre duas vezes ao ano.

1.2.3 Mudança de escopo

A proposta deste estudo era investigar como as pessoas em situação de vulnerabilidade social aprendiam as habilidades da Engenharia de Software. Esse estudo seria feito em estudantes de um curso de capacitação em Engenharia de Software presencial. No ano de 2020, no projeto mencionado na subseção 1.2.2, o desenvolvimento das atividades de ensino de Engenharia de Software foram 100% aplicadas de forma online. Devido à pandemia do coronavírus (COVID-19) foram necessários adequar a forma de trabalho, substituindo as atividades presenciais para as atividades online, durante o período de isolamento social.

A Figura 1.4 apresenta uma atividade de retrospectiva totalmente online. O primeiro semestre de 2021 está planejado para ocorrer da mesma forma. Neste caso, foram

Figura 1.3 – Estudantes de Engenharia de Software em uma Sessão de ensino
Fonte: a autora (2019)

necessárias adequações no escopo do plano de estudo e pesquisa apresentado em janeiro de 2020 pela autora.

1.3 Organização do volume

O presente volume está organizado da seguinte maneira: o Capítulo 1 apresenta uma contextualização sobre o tema, os objetivo do estudo, os motivos e a justificativa que fazem dessa pesquisa um tópico relevante para a área de Ciência da Computação, além do contexto e a adequação do escopo do estudo devido a pandemia do coronavírus (COVID-19); no Capítulo 2 são explanados uma visão geral sobre a teoria relacionada à Engenharia de Software, ensino de Engenharia de Software e ao tema diversidade em Engenharia de Software, pandemia do Coronavírus (COVID-19) e os trabalhos similares que de alguma forma relacionam-se com este estudo; o Capítulo 3 apresenta o método de pesquisa do estudo; já o Capítulo 4 apresenta o desenvolvimento de dois Mapeamentos Sistemáticos da Literatura (um estudo secundário e outro estudo terciário) - que serviram como base para a construção deste estudo; a compreensão do estudo qualitativo, planejamento e resultados finais do estudo são apresentados no Capítulo 5; já no Capítulo 6, apresenta a proposta desta pesquisa, em que são apresentados uma proposta com práticas de ensino de Engenharia de Software online; a conclusão, as contribuições da pesquisa e os trabalhos

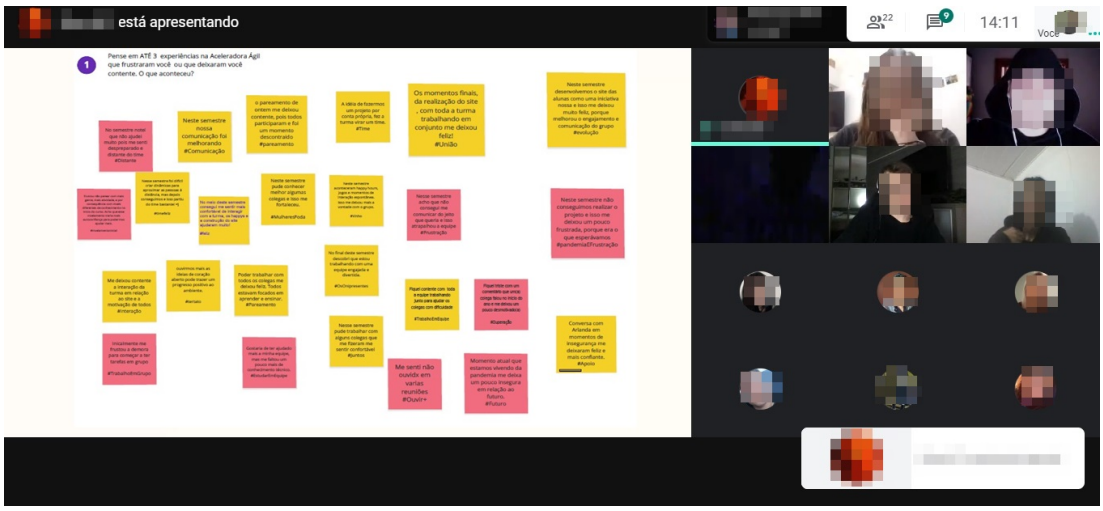

Figura 1.4 – Pessoas estudantes em uma atividade de Retrospectiva online
Fonte: a autora (2020)

futuros encontram-se no Capítulo 7. Por último, as Referências bibliográficas utilizadas e os Apêndices.

2. REFERENCIAL TEÓRICO

Este capítulo apresenta o embasamento teórico sobre os principais conceitos relacionados à essa pesquisa: Engenharia de Software (seção 2.1), ensino de Engenharia de Software (seção 2.2); Diversidade em Engenharia de Software (seção 2.3), Contexto da Pandemia do Coronavirus (seção 2.4) e Trabalhos relacionados (seção 2.5)

2.1 Engenharia de Software

Para Sommerville [93] a Engenharia de Software engloba todos os aspectos da produção de software, como processos técnicos do desenvolvimento, atividades de gerenciamento de projeto de software, desenvolvimento de ferramentas, métodos e teorias de apoio a produção de software.

Já Pressman [79] elaborou a seguinte definição para Engenharia de Software:

> A Engenharia de Software é uma tecnologia em camadas. Como ilustra a figura 2.1, qualquer abordagem de Engenharia (inclusive a Engenharia de software) deve estar fundamentada em um comprometimento organizacional com a qualidade. A Engenharia de software engloba um processo, métodos de gerenciamento e desenvolvimento de software, bem como ferramentas.

A Engenharia de Software tem como base a camada de processos. Esse processo é a conexão que mantém as camadas de tecnologia coerentes, facilitando o desenvolvimento de software de forma racional e dentro do prazo estabelecido, entre o desenvolvedor de software e o usuário. O processo define uma metodologia que deve ser determinada para a entrega efetiva de tecnologia. Ela é constituída com a base de controle no gerenciamento de projetos e determina em qual contexto aplicar os métodos técnicos, artefatos (modelos, documentos, dados, relatórios, formulários etc.), marcos, qualidade como garantia e mudanças conduzidas de forma apropriada [79].

Figura 2.1 – Camadas da Engenharia de Software
Fonte: [79], página 16

Na Engenharia de Software, os métodos, conforme Figura 2.1 fornecem as informações técnicas para desenvolver softwares. Nele está contido as tarefas (comunicação, análise de requisitos, modelagem de projeto, a construção do software, testes e suporte). Os métodos são baseados em uma gama de princípios básicos que lideram cada área da tecnologia e incluem as atividades de modelagem e outras técnicas [79].

Já as ferramentas em Engenharia de Software, fornecem um apoio automatizado ou semiautomatizado para o processo e para os métodos. Quando é realizada a integração das ferramentas, de forma que as informações sejam criadas, organizadas para que uma outra ferramenta possa ser utilizada, é estabelecido um sistema para o apoio ao desenvolvimento de software, denominado como Engenharia de Software com o auxílio do computador [79].

Um processo de software reúne uma gama de atividades e resultados ligados à produção de software [93]. Pode também ser referenciado como ciclo de vida do software, pois descreve a "vida" do produto de software desde a concepção até a implementação, entrega e a manutenção [80].

A expressão e a descrição de um processo são realizadas por meio de modelos. Cada modelo possui suas características únicas, sendo em alguns casos, semelhante na teoria, mas diferente na prática [80]. Os modelos são compostos pelas atividades do processo, dos papéis, métodos da Engenharia de Software e ferramentas utilizadas no apoio.

De acordo com Sommerville [93] existem atividades que são comuns a todos os modelos de processo de software. São elas:

- Especificação de software: clientes e a equipe definem o produto a ser desenvolvido e as restrições existentes;
- Desenvolvimento de software: projeção e desenvolvimento do software;
- Validação de software: onde é feita a validação do que foi produzido com a conformidade do que foi solicitado;
- Evolução do software: adaptação às mudanças dos requisitos do cliente e de mercado.

A escolha de um determinado modelo de processo para desenvolvimento de software deve ser realizada mediante a algumas condições que devem ser consideradas, tais como: a natureza do projeto e da aplicação, os métodos e as ferramentas a serem utilizados, os controles e os produtos a serem entregues [79]. Embora não exista um modelo "ideal", cabe ressaltar que existe a possibilidade de os modelos serem aprimorados no contexto de diferentes organizações [93].

2.2 Ensino de Engenharia de Software

A Engenharia de Software é uma disciplina preocupada com a efetividade da teoria, conhecimento e prática aplicadas no desenvolvimento efetivo e eficiente de sistemas de software que atendem satisfatoriamente os requisitos dos usuários [53]. De modo geral, Engenharia de Software é ensinada no ensino formal no nível de graduação e/ou pós-graduação dos cursos de computação, ou por meio de treinamentos profissionais de curta duração [81].

Segundo Prikladnicki et al. [81], para ensinar engenharia de software é importante considerar as técnicas existentes de ensino e aprendizado. As metodologias mais comuns para ensinar Engenharia de Software englobam as aulas expositivas, aulas de laboratório, entre outros. Contudo, metodologias variadas podem contribuir com o aprendizado efetivo das alunas, como por exemplo:

- A substituição de aulas expositivas por discussão de casos práticos [43];
- Dinâmicas de grupo e o uso de jogos [103];
- *Capstone projects* (um esforço em grupo para que as alunas executam um projeto do início ao fim) [44].

O desenvolvimento de software é uma atividade altamente técnica e requer engenheiros de software com habilidades, conhecimento, experiência em diversas metodologias, ferramentas e técnicas [58]. Segundo Matturro [58], as empresas de software, quando contratam e formam as equipes de projetos costumam focar no conhecimento técnico, nas habilidades em potencial dos engenheiros de software.

Para Bollin et al. [13], o ensino da ES tem como intuito capacitar habilidades no profissional para desenvolver sistemas computacionais, por meio das técnicas de análise, projeto e programação, e habilidades de avaliação do produto, mediante a técnicas de revisão e estratégias de teste. Com a evolução dos sistemas computacionais, a complexidade no desenvolvimento e manutenção de software também aumenta. Com base na preocupação em produzir sistemas computacionais de qualidade e que atendam aos requisitos solicitados pelos clientes, os profissionais da ES devem assumir a responsabilidade de assegurar que os artefatos desenvolvidos possuam um elevado padrão de qualidade em termos de confiança e confiabilidade [106].

2.2.1 Desafios no ensino de Engenharia de Software

Os profissionais da área da ES exercem um papel fundamental em equipes de desenvolvimento de projetos de software, pois podem participar em várias fases do ciclo de vida do produto [1]. Eles podem atuar nas análises, levantamento de requisitos, arquitetura, desenvolvimento, testes e manutenção. Contudo, para que este profissional desempenhe suas funções dentro dos padrões desejáveis para a empresa, é recomendável que possua uma base sólida na formação acadêmica [1], seja por meio de uma graduação, pós-graduação ou curso de capacitação/aperfeiçoamento.

Segundo Zambon et al. [108], considerando a complexidade dos assuntos relacionados a ES, cabe as pessoas professoras buscarem alternativas para auxiliar suas alunas na compreensão deste conteúdo. Uma das alternativas utilizadas para o aprendizado em sala de aula é a inserção de jogos educacionais, digitais e não-digitais, em geral, direcionados a práticas simuladas.

2.2.2 Diretrizes curriculares de Engenharia de Software

As diretrizes curriculares para programas de graduação em Engenharia de Software (SE 2014) [7] fornecem uma orientação para instituições de ensino sobre como constituir um curso de graduação em Engenharia de Software. As diretrizes[1] definem um corpo de conhecimento com sugestões para todos os graduados em engenharia de software: o conhecimento de educação em Engenharia de Software (SEEK).

O SE 2014 foi construído em torno de três grandes iniciativas que envolveram o grande número de voluntários, bem como todos os membros do Comitê Diretivo. Segundo Moreno et al. [63], a primeira dessas iniciativas foi o desenvolvimento de um conjunto de resultados curriculares e uma declaração do que todo graduado em Engenharia de Software deve saber. A segunda iniciativa envolveu determinar e especificar o conhecimento a ser incluído em um programa de graduação em Engenharia de Software. A terceira iniciativa foi a construção de um conjunto de recomendações curriculares, descrevendo como um currículo de engenharia de software, incorporando como SEEK.

2.2.2.1 Conhecimento em educação em Engenharia de Software (SEEK)

O SEEK representa um corpo de conhecimento central de conteúdos essenciais para que os profissionais que ensinam Engenharia de Software, que serve como base de ensino necessários para qualquer pessoa obter um diploma de graduação nessa área.

[1] Disponivel em: https://www.acm.org/binaries/content/assets/education/se2014.pdf

Portanto, embora SEEK não representa um currículo completo, ele fornece a base para o projeto, implementação e entrega de unidades educacionais centrais que constituem um currículo de engenharia de software [63].

O corpo do SEEK é organizado hierarquicamente em três níveis. As áreas de conhecimento representam subdisciplinas específicas de Engenharia de Software, geralmente reconhecidas como partes significativas do corpo de conhecimento de ES que uma estudante de graduação deve aprender. Cada área é dividida em divisões menores chamadas unidades, que representam módulos temáticos individuais dentro de uma área. Finalmente, cada unidade é subdividida em um conjunto de tópicos, que é o nível mais baixo da hierarquia [7]. Os conteúdos podem ser agrupados em nomes de cursos diferentes, gerando currículos específicos que cobrem o mesmo núcleo de conhecimento de Engenharia de Software.

O SEEK está organizado em dez “áreas de conhecimento” [7], representado em subdisciplinas particulares da Engenharia de Software, onde geralmente são reconhecidas como uma parte significativa do conhecimento em Engenharia de Software - recém graduado deve conhecer. A Figura 2.2 ilustra as 10 áreas de conhecimento sugeridas pelas diretrizes curriculares para programas de graduação em Engenharia de Software, por meio do SEEK:

Sigla	Área	Descrição
FC	Fundamentos da Computação	Os Fundamentos da computação incluem as bases da ciência da computação que dão suporte ao design e à construção de produtos de software. Esta área também inclui o conhecimento sobre a transformação de um projeto em implementação, bem como as técnicas e ferramentas usado durante este processo
FME	Fundamentos da Matemática e Engenharia	Os fundamentos matemáticos e de engenharia da Engenharia de Software fornecem fundamentos teóricos e científicos para a construção de produtos de software com os atributos desejados. Esses fundamentos apoiam a descrição precisa dos produtos da Engenharia de Software. Eles fornecem a base matemática para modelar e facilitar o raciocínio sobre esses produtos e suas inter-relações e formando a base para uma previsível processo de design.
PRP	Prática Profissional	A prática profissional preocupa-se com o conhecimento, as habilidades e as atitudes que os engenheiros de software devem possuir para praticar a Engenharia de Software com profissionalismo, responsabilidade e ética.
MASO	Modelagem e Análise de Software	Modelagem e análise podem ser consideradas conceitos centrais em qualquer disciplina de engenharia porque são essenciais para documentar e avaliar decisões e alternativas de projeto.
AER	Análise e Especificação de Requisitos	A construção de requisitos inclui a elicitação e análise das necessidades das partes interessadas e a criação de uma descrição apropriada do comportamento e qualidades desejados do sistema, juntamente com restrições e suposições relevantes.

Figura 2.2 – Áreas de conhecimento em educação em Engenharia de Software (Parte 1)
Fonte: Adaptado de [7]

Sigla	Área	Descrição
DS	Design de Software	O design de software preocupa-se com questões, técnicas, estratégias, representações e padrões usados para determinar como implementar um componente ou sistema.
VVS	Verificação e Validação de Software	A verificação e validação de software usa uma variedade de técnicas para garantir que um componente ou sistema de software satisfaça seus requisitos e atenda às expectativas das partes interessadas.
PS	Processo de Software	O processo de software preocupa-se em fornecer estruturas apropriadas e eficazes para as práticas de engenharia de software usadas para desenvolver e manter componentes e sistemas de software nos níveis individuais, de equipe e empresas.
QS	Qualidade de Software	A qualidade do software é uma preocupação transversal, identificada como uma entidade separada para reconhecer sua importância e fornecer um contexto para alcançar e garantir a qualidade em todos os aspectos da prática e do processo de engenharia de software.
SG	Segurança	A segurança do software tem dois componentes distintos e relacionados. Como área de conhecimento distinta trata da proteção de informações, sistemas e redes. Como uma preocupação transversal, ele se concentra em como a segurança deve ser incorporada a todas as partes do ciclo de vida de desenvolvimento de software.

Figura 2.3 – Áreas de conhecimento em educação em Engenharia de Software (Parte 2)
Fonte: Adaptado de [7]

As Figuras 2.2 e 2.3 apresentam a sigla, a área e uma breve descrição das 10 áreas de conhecimento em Engenharia de Software (SEEK): Fundamentos da Computação (FC), Fundamentos da Matemática e Engenharia (FME), Prática Profissional (PRP), Modelagem e Análise de Software (MASO), Análise e Especificação de Requisitos (AER), Design de Software (DS), Verificação e Validação de Software (VVS), Processo de Software (PS), Qualidade de Software (QS) e Segurança (SG).

2.3 Diversidade em Engenharia de Software

A diversidade está sendo abordada intensamente por diferentes áreas do conhecimento e da sociedade. Esse assunto também está sendo discutido pela Engenharia de software, uma vez que o desenvolvimento de software é uma tarefa colaborativa e equipes de desenvolvimento de software são formadas por pessoas diferentes [92].

Silveira et al. [92] comentam que a representatividade de pessoas da diversidade de etnias e de gênero ainda é pequena na engenharia de software. As empresas de tecnologia devem apoiar a diversidade nas equipes de desenvolvimento de software e esse desafio está sendo adotado por várias empresas, uma vez que a trajetória para a inclusão das classes sub-representadas ao desenvolvimento de software tem um valor e necessidade inegável. Estudos dizem que a diversidade cria melhores equipes e oferece melhores resultados, entre outros benefícios.

Menezes et al. [60] citam que a diversidade é contemplada pelas oportunidades e desafios que impactam ao redor do mundo. As pessoas precisam adotar a tolerância,

respeito, empatia e compreensão das diferenças. O valor da diversidade no ambiente de trabalho pode aumentar a força de trabalho, como a produtividade e competitividade organizacional, bem como promover a publicidade de inclusão da empresa [65]. Para Vasilescu et al. [102] a diversidade social é uma importante fonte de criatividade e adaptabilidade nas equipes. Equipes com diversidade social podem aproveitar informações mais amplas, ideias variadas, habilidades aprimoradas de resolução de problemas, tornando-se mais eficazes.

Equipes culturalmente diversas são necessárias nos ambientes de trabalho, e seus efeitos podem afetar o desempenho, processos, resultados organizacionais e até avanços tecnológicos [107]. Uma outra contribuição da diversidade de funcionários é tornar as empresas mais adequadas para atender e entender os clientes externos e requisitos, respectivamente [70]. Equipes diversas permitem gerar potencial e obtenção de vantagem competitiva entre eles [70], como aumentar o estágio de produção [75], aumento na eficácia [107] e produtos finais mais robustos [97].

2.3.1 Diversidade Social

Segundo Alves et al. [5], as diferenças entre as pessoas são consistentemente observadas quanto ao lugar que eles ocupam na hierarquia social. Essas diferenças estão associadas às oportunidades educacionais, às trajetórias ocupacionais, questões financeiras e de moradia, educação, acesso aos bens e serviços, ao comportamento político e social etc. O estudo dessas diferenças, seja como um fenômeno a ser explicado ou sua associação a outros fenômenos sociais, constitui uma área de grande importância nas pesquisas de forma geral [5]. A Diversidade social, são caracterizadas por diferenças e valores compartilhados por todos os humanos nas relações sociais [88]. Essas diferenças são expressas através da língua, cultura, etnia, condições sociais, entre outros [88].

As condições sociais, em relação a vulnerabilidade social é caracterizado pela condição de grupos de pessoas que vivem em condições desfavoráveis como: condições precárias de moradia, ausência de um ambiente familiar, financeiro e/ou educacional de qualidade [62]. Essas pessoas possuem acesso limitado à educação, cursos e aperfeiçoamentos necessários para construir uma profissão [84].

Carrara et al. [19] caracterizam a vulnerabilidade social entre os meios de pesquisas sociais, educacionais e psicológicos, ou famílias em situação de risco, famílias pobres, famílias de baixa renda, famílias de camadas populares entre outros para denotar o mesmo sentido. Já Prati et al. [77] definem vulnerabilidade social como famílias que apresentam vulneráveis por estarem fragilizadas e suscetíveis a fatores de risco. A vulnerabilidade pode ser caracterizada também pela impossibilidade de modificar a condição atual em que se encontram, muitas em condições precárias no que se refere à alimentação, higiene, educação e ou saúde [19].

2.4 Pandemia do Coronavirus (COVID-19)

A Organização Mundial da Saúde (OMS) anunciou que o Coronavírus tornou-se uma emergência mundial da saúde pública no dia 30/01/2020 [96] e como uma Pandemia em 11/03/2020 [9]. A capacidade de transmissão do vírus é mais forte que uma gripe convencional - que geralmente acontece uma 1 vez por ano. A Organização Mundial da saúde recomendou o distanciamento social em lugares públicos e privados, pois uma pessoa infectada pelo vírus pode contaminar de 2 a 5 pessoas (essa média varia da localização geográfica, faixa etária e tempo) [21].

O atual cenário devido à pandemia cria uma grande demanda em adaptações de convivência, atividades educacionais, trabalho e/ou financeira [98]. De acordo com Tietze et al. [98] os esforços advindos das alterações de rotina, formas de convívio e funções geram ações instantâneas das organizações e instituições nas atividades, devido ao isolamento social.

2.4.1 Ensino remoto em tempos de pandemia

A educação a distância é uma alternativa ao ensino presencial tradicional [49]. O ensino à distância é uma metodologia que envolve o uso de dispositivos móveis para realização do processo de ensino-aprendizagem, essa facilidade com o uso da tecnologia da informação fornece conteúdo de ensino e conduz as avaliações necessárias. Este conceito mitiga a necessidade de estudantes e professoras estarem fisicamente presentes em um local [49].

Em situações excepcionais como a vivida durante à pandemia da COVID-19 no mundo, os métodos de educação virtual assumem grande importância, sendo o principal caminho para a educação das alunas [87]. No entanto, os benefícios desse conceito foram amplamente percebidos nas novas circunstâncias causadas pela pandemia de COVID-19. Essas mudanças como infraestrutura, internet e na aquisição de dispositivos móveis impactam diretamente na vida das pessoas [87].

A adequação dos cursos existentes para serem ministrados na modalidade online e o acesso dos alunos à tecnologia foram grandes desafios que podem restringir o sucesso da modalidade de ensino presencial [9]. A experiência em ministrar cursos existentes projetados para entrega em sala de aula na modalidade de educação à distância ensinou lições importantes a muitas pessoas professoras, inclusive nos resultados apresentados no estudo da autora desse trabalho [95]. Algumas lições serão apresentadas no capítulo 6 como resultados da pesquisa durante a pandemia.

2.5 Trabalhos relacionados

Nesta seção tem-se um apanhado sobre os trabalhos que de alguma forma se relacionam com esse estudo. Nesta seção foram destacados somente estudos relevantes relacionados ao ensino de Engenharia de Software.

2.5.1 Um jogo de cartas experimental para ensinar processos de Engenharia de Software

Segundo Baker et al. [10], é fundamental a base inicial teórica do ensino em habilidades da Engenharia de Software. O estudo apresenta um jogo de cartas educacional para pessoas alunas do curso de computação, que simula o processo de Engenharia de Software, desde a especificação de requisitos até a entrega do produto - essa prática fornece as alunas uma experiência útil na criação de resultados, enriquecendo o currículo, que geralmente não são abordados na maioria dos cursos. Desse modo, o autor descreve como o jogo foi projetado, como a mecânica foi construída e os resultados do experimento realizado com pessoas alunas.

Baker et al. [10] desenvolveu um jogo de cartas educacional com a possibilidade de simulação do processo de Engenharia de Software - projetado para ensinar os problemas desse processo, que não são abordados na maioria dos cursos. O jogo é organizado como um jogo competitivo, no qual as pessoas assumem um papel de líderes de projetos na mesma empresa. Ambos recebem o mesmo projeto e recebem instruções para concluir o mais rápido possível. A pessoa jogadora que concluir o projeto primeiro será a vencedora. No entanto, as jogadoras devem equilibrar várias preocupações recorrentes enquanto trabalham, incluindo seu orçamento e as demandas do cliente em relação à confiabilidade do software produzido. Elas devem esforçar-se para seguir com as práticas adequadas da Engenharia de Software para evitar quaisquer consequências adversas - que podem afetar o seu andamento, diminuindo seu ritmo e possibilitando que seu oponente na corrida, conclua o projeto primeiro.

Ao concluir seu projeto, as pessoas jogadoras jogam cartas com base no modelo de ciclo de vida em cascata, conforme mostrado na Figura 2.4. As pessoas alunas estão mais acostumadas com o modelo Cascata, apresentado na Figura 2.4, além disso, o modelo em cascata refletido no jogo não é simplesmente linear - os jogadores podem retornar às fases anteriores de desenvolvimento (com uma penalidade), bem como pular fases, se desejado.

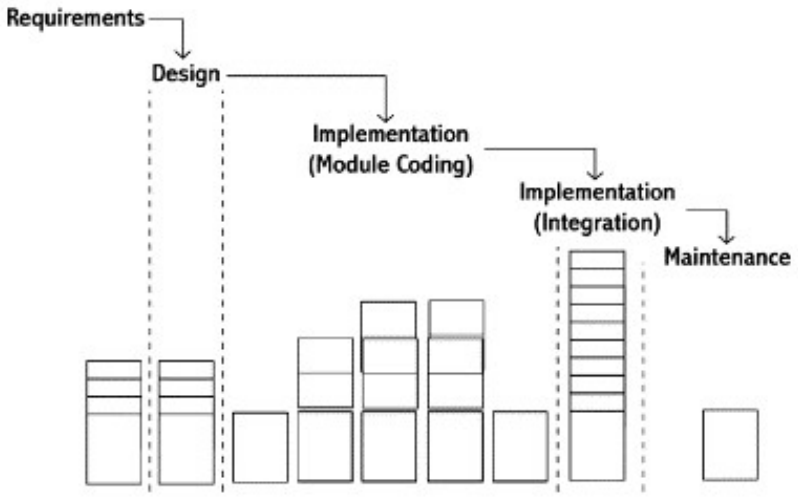

Figura 2.4 – Fases correspondentes as áreas do jogo
Fonte: [10], página 6.

2.5.2 Ensino de Engenharia de Software através do design de jogos

Para tornar o ensino de Engenharia de Software de forma realista com prática e diversão, Claypool et al. [23] realizaram um estudo para aumentar o interesse das alunas na disciplina de Engenharia de Software, através do uso de projetos baseados e centrados em jogos de computador. A intenção era que as alunas participassem ativamente das diferentes fases do ciclo de vida do software, desde a elicitação de requisitos, teste e manutenção. Além disso, o intuito era expor as pessoas alunas a problemas reais no projeto e gerenciamento de equipes, apresentando os diferentes aspectos do design de jogos de computador.

Cada atividade da Engenharia de Software foi projetada em módulos com base em jogos de computador. No total foram desenvolvidos 10 módulos: módulo 1 (Introdução); módulo 2 (Ciclos de vida do desenvolvimento de software); módulo 3 (Gerenciamento de projetos e equipes); módulo 4 (Elicitação de requisitos); módulo 5 (Introdução ao design de jogos); módulo 6 (Análise de requisitos); módulo 7 (Projeto de sistema); módulo 8 (Design de objetos); módulo 9 (Implementação) e módulo 10 (Teste).

2.5.3 Programa Aceleradora Ágil: a colaboração entre a indústria e a academia até o treinamento eficaz com métodos ágeis

De acordo com Steglich et al. [95] O projeto da aceleradora ágil, acontece em um parque tecnológico com a parceria entre uma universidade e três renomadas empresas, uma consultoria e desenvolvimento ágil, uma cooperativa e uma em comunicação. Essa

parceria já dura 8 anos e tem como principal objetivo, preparar estudantes de graduação e curso técnico para atuar em equipes ágeis de alto desempenho. Esta parceria criou um ambiente culturalmente rico para aprendizagem dos estudantes, influenciando outras empresas a seguir a mesma iniciativa dentro do parque tecnológico. Este estudo conduziu um estudo de caso com o objetivo de caracterizar esta parceria (explicando como funciona), apresentando os benefícios para as estudantes do projeto. Os resultados apontam a importância dessa parceria - fornecendo um ambiente de aprendizagem, envolvendo as estudantes em um ambiente real de indústria de desenvolvimento de software - com projetos reais e verdadeiras partes interessadas. Conforme alguns relatos de estudantes, o projeto é muito importante para ingressarem no mercado de trabalho.

O estudo de Steglich et al. [95] apresenta uma parceria com a indústria, em que, as estudantes aprendem imersos no programa de 18 semanas, em um ambiente semelhante ao da indústria, onde a prática é promovida, orientada e apoiada por pessoas profissionais da indústria e instrutores acadêmicos, orientados pela teoria. A teoria é revisitada organicamente, toda vez que uma nova atividade é introduzida, com o interesse de reforçar conteúdos teóricos e praticar exemplos vivenciados na realidade da indústria de desenvolvimento de software.

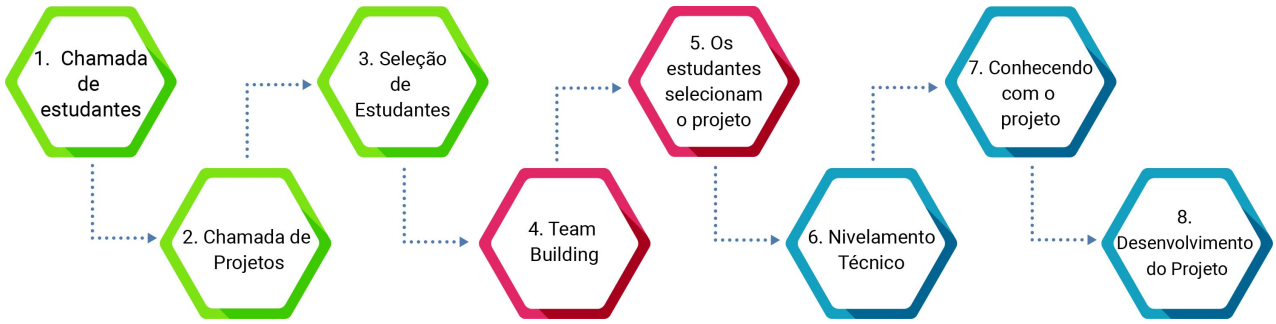

Figura 2.5 – Fases da Aceleradora Ágil
Fonte: [95], página 24.

As fases do projeto da Aceleradora Ágil são apresentadas na Figura 2.5, as fases de 1 a 3 (destacadas em verde) são originárias do processo de seleção para chamada de estudantes candidatas, chamada para seleção de projetos que será desenvolvido pelas alunas na fase posterior (fase 8) e a seleção das estudantes para serem alunas do projeto da Aceleradora Ágil. As fases 4 e 5 (destacadas em rosa) apresentam o processo de *Team Building* - consolidação de equipe, como as atividades de integração e a seleção do projeto que será desenvolvido pelas estudantes. As fases de 6 a 8 (destacadas em azul), referem-se ao processo de nivelamento técnico das estudantes, com exercícios de programação, a fase 7 apresenta o processo e familiarização com o projeto escolhido pelas estudantes e desenvolvimento do projeto com apoio de profissionais da área de desenvolvimento de software, vinculadas ao projeto da Aceleradora Ágil.

O artigo mencionado anteriormente, foi desenvolvido e publicado pelas pessoas autoras deste estudo, no XXXIV Simpósio Brasileiro de Engenharia de Software - SBES 2020.

2.5.4 O Status socioeconômico e uso do computador: projetando software para usuários de Vulnerabilidade social

Hill [45] afirma que status socioeconômico baixo afeta profundamente a maneira como as crianças e os adultos se comportam e se desenvolvem ao longo de sua vida formando hábitos e estratégias diferentes em comparação as pessoas de status socioeconômico alto. O autor formula 4 hipóteses sobre como essas diferenças podem impactar suas estratégias de solução de problemas durante o uso do software. As hipóteses formuladas pelo autor [45] são apresentadas a seguir:

- Hipótese 1: indivíduos com status socioeconômico baixo terão menos probabilidade do que seus pares com status socioeconômico alto em usar recursos de software para corrigir problemas encontrados no software?
- Hipótese 2: indivíduos com status socioeconômico baixo podem ter baixa autoeficácia na computação, levando à uma sensação de falha inevitável em suas tarefas de computação?
- Hipótese 3: indivíduos com status socioeconômico baixo podem utilizar dicas de ajuda de software narrativo com mais facilidade do que instruções passo a passo?
- Hipótese 4: indivíduos com status socioeconômico baixo podem reagir negativamente e desistir de tarefas de computação quando o sistema tira o controle?

Para conclusão do estudo, o autor apresenta um método *GenderMag* [15] - um método de inspeção de software desenvolvido para ajudar os profissionais de software a identificar problemas de inclusão de gênero no desenvolvimento do projeto. O método *GenderMag* baseia-se em pesquisas sobre as diferenças nos padrões de uso de software, agrupados por gênero e caracterizado por 5 aspectos: motivação, estilo de processamento de informações, autoeficácia do computador, aversão ao risco e ajustes.

2.5.4.1 Conclusão sobre o trabalhos relacionados

Com os estudos mencionados anteriormente, foram identificados que o ensino de Engenharia de Software pode ser realizado de formas distintas, ou seja, com diversão, interação, de forma *gamificada* ou a configuração de um possível ambiente real de indústria de

desenvolvimento de software. De várias maneiras o conhecimento pode ser captado entre os estudantes. Entretanto, o estudo de Hill [45] apresentou as diferenças no aprendizado e nas formas de agir, entre pessoas de status socioeconômico baixo e alto, em atividades de Engenharia de Software.

Estes estudos serviram como base para o embasamento teórico desse trabalho e também na construção do protocolo de pesquisa qualitativa para coletar dados em entrevistas semiestruturadas e técnicas de análise de dados de *Grounded Theory*, apresentado no Apêndice D.

3. MÉTODO DE PESQUISA

O método de pesquisa desenvolvido nesse trabalho compreende um estudo empírico e exploratório com dados qualitativos. Mais detalhes sobre o método de pesquisa qualitativo serão apresentados no capítulo 5. O desenho de pesquisa e suas fases são apresentados na seção 3.1. A seção 3.2 descreve os aspectos metodológicos e a base metodológica é apresenta na seção 3.3.

3.1 Desenho de Pesquisa

O desenho de pesquisa desse estudo está dividido em 4 fases: Fase 1 - Base Teórica; Fase 2 - Mapeamento Sistemático da Literatura; Fase 3 - Coleta de dados em entrevistas semiestruturadas e o uso de técnicas da Teoria fundamentada nos dados (*Grounded Theory*) para a análise de dados; Fase 4: Proposta final. A Figura 3.1 apresenta as fases e suas atividades.

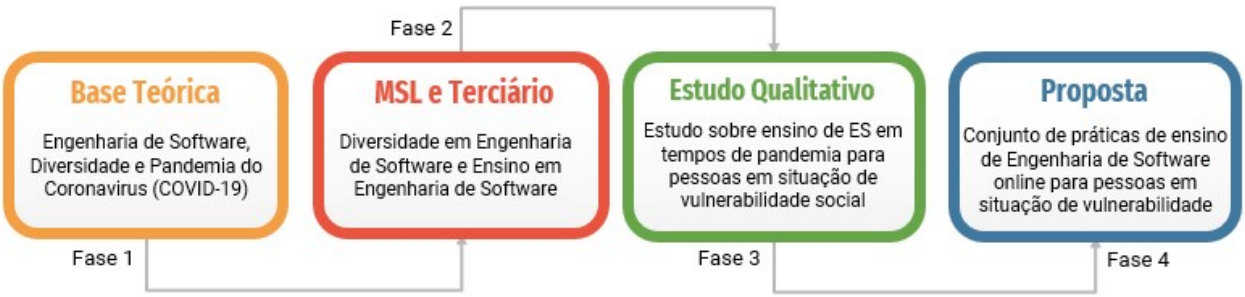

Figura 3.1 – Desenho de Pesquisa
Fonte: a autora (2020)

3.1.1 Fase 1: Base Teórica

Esta fase é exploratória, e possui apenas uma atividade. Nesta fase buscou-se estudar o referencial teórico, envolvendo inicialmente os conteúdos de Engenharia de Software, Diversidade e Pandemia do Coronavírus (COVID-19). Durante essa fase foram estudados conceitos relacionados a Engenharia de Software. Todo o estudo foi realizado através de uma busca na literatura por trabalhos publicados em eventos, conferências na área e em livros de autores que são referências na área de Engenharia de Software, como Pressman [80] e Sommerville [94]. Para se ter uma visão geral e de forma mais aprofundada, na fase 2 optou-se por fazer um mapeamento sistemático da literatura para entender

o contexto de diversidade em Engenharia de Software. Ainda na fase 1, devido à adequação do escopo do estudo, apresentada na seção 1.2.3, foi necessária uma nova busca na literatura sobre a pandemia do Coronavírus (COVID-19), para a autora desse estudo ter como base teórica os acontecimentos e adequações das vivências e nas formas de ensino de Engenharia de Software devido à pandemia. Esses estudos ainda são limitados devido à pandemia ainda estar presente atualmente.

3.1.2 Fase 2: MSL e Estudo Terciário

A segunda fase consistia no planejamento e execução de um estudo secundário, mapeamento sistemático da literatura (MSL). O objetivo do mapeamento sistemático foi de aprofundar o entendimento sobre como a Engenharia de Software está estudando e tratando o assunto diversidade em Engenharia de Software. Durante o processo de leitura dos artigos científicos, foram identificados que alguns tipos de diversidade ainda são pouco abordados em Engenharia de Software. O MSL permitiu a identificação dos trabalhos já existentes sobre o tema, serviu como base teórica para o referencial teórico dessa pesquisa e a extração de resultados e lições aprendidas a respeito de diversidade em Engenharia de Software. Devido à adequação do escopo, mencionada na seção 1.2.3 foi realizado um estudo terciário, para identificar estudos sobre ensino de Engenharia de Software. Esse estudo foi planejado e executado em revisões e mapeamentos sistemáticos da literatura publicados nos principais periódicos e conferências da área. O estudo terciário resultou em mais embasamento teórico para o referencial teórico desse estudo, além de direcionar um caminho para as perguntas que foram planejadas e realizadas nas entrevistas semiestruturadas, na Fase 3 do desenho de pesquisa.

3.1.3 Fase 3: Estudo Qualitativo

Nesta fase, planejou-se e executou-se um protocolo de pesquisa para coleta de dados qualitativos com a técnica de entrevistas semiestruturadas e análise de dados usando técnicas de *Grounded Theory* com pessoas professoras e alunas do curso de capacitação em Engenharia de Software, mencionado na seção 1.2.2, capítulo 1. Após as 5 primeiras entrevistas coletadas e analisadas - por meio do processo de codificação, o protocolo foi ajustado, devido aos resultados gerados. O método foi iterativo (progresso através de refinamentos) e incremental (com a ideia inicial, executa-se o que pode possibilidade a evolução). De acordo com as evoluções dos resultados, foi realizada uma segunda rodada com 5 entrevistas, nesse caso, o protocolo novamente foi incrementado com perguntas

mais abertas. Em uma terceira rodada de 14 novas entrevistas, os resultados obtidos foram saturados. Os resultados em detalhes serão apresentados no capítulo 6.

3.1.4 Fase 4: Proposta

Após o desenvolvimento da base teórica (fase 1), o planejamento e execução de um estudo secundário de mapeamento sistemática da literatura, um estudo e terciário e a execução do estudo qualitativo com coleta de dados em entrevistas semiestruturadas e análise de dados com técnicas de *Grounded Theory*, a última fase do desenho de pesquisa consistiu em propor um conjunto de práticas de ensino de Engenharia de Software online considerando pessoas em situação de vulnerabilidade social. Como contribuição, essas práticas podem ser ensinadas de forma 100% online. Ainda como resultado, foram identificadas as lições aprendidas durante todo o processo deste estudo.

3.2 Aspectos Metodológicos

De forma a garantir e ampliar a validade deste estudo, um rigoroso processo de pesquisa foi planejado. Nesta pesquisa, a definição e utilização de protocolos para desenvolvimento e formalização dos estudos tiveram por objetivo sistematizar as tarefas de coleta e análise, aumentando a confiabilidade da pesquisa. O mapeamento sistemático da literatura foi planejado utilizando um protocolo de acordo com Petersen [72]. Já o protocolo de pesquisa para coleta de dados qualitativos por meio de entrevistas semiestruturadas, e análise com técnicas de *Grounded Theory* foram planejados de acordo com Glaser et al. [40].

Quanto à generalização dos resultados, Patton [71] menciona que a teoria fundamentada nos dados, ao contrário dos métodos quantitativos, a generalização estatística de uma amostra para a população não é um objetivo. A pesquisa qualitativa procura casos ricos em informações que permitem aprofundar o pesquisador na compreensão. No início do processo, cada caso foi tratado como único e estudado detalhadamente. As análises dos casos, foram cruzadas, seguiram posteriormente e são baseadas em casos individuais bem compreendidos.

O viés de todo o processo de pesquisa foi minimizado com a avaliação das etapas de pesquisa com um pesquisador experiente (o professor orientador). No mapeamento sistemático, a avaliação ocorreu principalmente no protocolo, na ficha de leitura dos artigos e na análise dos resultados. Nas entrevistas semiestruturadas, o protocolo da coleta dos dados e a análise dos resultados também contaram com esta avaliação.

3.3 Base Metodológica da *Grounded Theory* (Teoria fundamentada nos dados)

A fim de coletar dados, Przyborski et al. [82], comenta que qualquer pesquisador empírico preocupado com ações sociais deve: comunicar-se diretamente com os sujeitos ou observar sua interação. Assim, o pesquisador enfrenta a mesma tarefa de compreender e interpretar o comportamento do outro. Para este estudo, após o Mapeamento Sistemático da Literatura, optou-se por utilizar a coleta de dados e entrevistas semiestruturadas e técnicas de análise de dados da Teoria fundamentada nos dados. Segundo Glaser [40] a *Grounded Theory* (GT) é uma geração sistemática da teoria partir de dados analisados por um rigoroso método de pesquisa. Ela foi desenvolvida por sociólogos como Glaser e Strauss [40] com resultados de suas pesquisas colaborativas nos hospitais em doentes terminais. Eles publicaram "A descoberta da Teoria Fundamentada"[40] que lançou a fundamentação da GT. Glaser et al. [41] definem como:

> A abordagem da teoria fundamentada é uma metodologia geral de análise vinculada a coleta de dados, que usa um conjunto de métodos aplicados sistematicamente, para gerar uma teoria indutiva sobre uma área.

A coleta e análise de dados foi desenvolvida com base em um estudo terciário de mapeamento sistemático da literatura sobre ensino de Engenharia de Software. O objetivo desse estudo em específico foi entender as práticas de ensino de Engenharia de Software em estudos secundários. Buscou-se um entendimento sobre quais práticas são estudadas na literatura. Como resultado, o maior número de estudos encontrados menciona as Diretrizes Curriculares da ACM/IEEE [7], especificamente o corpo de Conhecimento e Educação em Engenharia de Software (SEEK).

3.3.1 Seleção do projeto de ensino e unidade de análise

A unidade de análise do estudo foi definida como sendo o projeto de ensino de Engenharia de Software que a autora desse estudo trabalha. É um projeto rico em dados. Desta forma, foram escolhidos profissionais da indústria que participam do projeto como pessoas mentoras / professoras e estudantes do projeto. Os profissionais pertenciam a 3 organizações diferentes e todos colaboraram no processo de coleta de dados por meio das entrevistas. No capítulo 6 apresentam-se detalhadamente os resultados encontrados em cada um dos casos estudados nesta etapa.

3.3.2 Fonte dos dados e seleção dos participantes

A coleta de dados foi constituída por fontes primárias. As fontes primárias foram constituídas de entrevistas. Foram realizadas 24 entrevistas semiestruturadas individuais em profundidade. Partiu-se de um roteiro básico com questões formuladas aos entrevistados e adequadas conforme seu desenvolvimento.

O critério inicial para a definição das pessoas entrevistados centrou-se em pessoas mentoras / professoras que participam do projeto desde o início ou a mais tempo. Neste sentido, a população envolvida constituía-se de profissionais que possuíam o perfil de desenvolvedores de software, gerentes de projeto e designer UX. O instrumento de coleta de dados consistiu em um roteiro para entrevista semiestruturada (Apêndice D). As entrevistas foram organizadas para identificar práticas de ensino de Engenharia de Software online para pessoas em situação de vulnerabilidade social.

3.3.3 Análise dos Dados

A técnica utilizada para a análise de dados foi o processo de codificação (codificação aberta, codificação seletiva e codificação teórica) e comparação constante indicado por Glaser e Strauss [40] e Glaser [41]. Desta forma, a cada entrevista transcrita, o processo de análise era realizado posteriormente. Após cada transcrição, os dados foram preparados, uma leitura cuidadosa foi realizada, de modo a buscar a familiarização da pesquisadora com os dados antes de iniciar a codificação. Para cada pergunta da entrevista foram identificadas categorias de acordo com os dados codificados. Este processo foi conduzido pela pesquisadora e depois consolidado com o orientador, avaliando o conjunto de categorias a serem considerados.

À medida que novas entrevistas são coletadas e analisadas, em detalhes, alguns deles poderão ser semelhantes aos já analisados (uns se enquadram no mesmo conceito), mas ao mesmo tempo são diferentes de alguma forma. Capturar essas variações, os conceitos devem ser continuamente revisados, por exemplo, pela introdução de novas propriedades, o que, por sua vez, exige que a pesquisadora revisite exemplares já analisados para ver se o conceito alterado ainda se ajusta a ele ou qual seu valor específico em relação a um recém-introduzido chamado de comparação constante.

3.3.4 Fases e operacionalização das entrevistas semiestruturadas

A pesquisa foi desenvolvida com 11 profissionais de 3 organizações, uma de desenvolvimento e consultoria de software, uma cooperativa e uma instituição de ensino, vinculada ao curso de capacitação em Engenharia de Software e com 13 alunas - 6 alunas são pessoas em situação de vulnerabilidade social [84]. Após entrar em contato com cada uma das participantes, todos dispuseram cerca de uma hora de tempo para as entrevistas que ocorreu à distância por meio da plataforma *(Google Meet)*[1], devido à questão da pandemia [9].

Um roteiro semiestruturado foi utilizado como instrumento de coleta. Este roteiro foi desenvolvido tomando-se por base um roteiro inicial de questões, a partir dos resultados do MSL e estudo terciário executado e representado pelo protocolo de pesquisa desenvolvido para coletar dados em entrevistas semiestruturadas e técnicas análise de dados da *Grounded Theory* (Apêndice D).

A validação da estrutura e conteúdo do protocolo de pesquisa foi realizada por uma profissional da indústria com experiência desde a primeira turma do curso de capacitação em Engenharia de Software. Após sua aplicação foi possível identificar o que poderia ser melhorar e adaptar as perguntas do protocolo para se a adequar ao objetivo da pesquisa. Foram realizadas 3 iterações sucessivas até gerar uma versão estável do roteiro.

A análise das entrevistas seguiu várias etapas, que iniciaram pela definição do universo estudado, delimitando o que estaria envolvido. Em seguida, iniciou-se a categorização, através da codificação, representando tópicos significativos em função das quais o conteúdo foi classificado. Os resultados das transcrições foram categorizados e, por fim, os resultados foram desenvolvidos.

[1] Disponível: https://meet.google.com/

4. ESTUDOS SECUNDÁRIO E TERCIÁRIO

Neste capítulo, é apresentado o desenvolvimento do mapeamento sistemático da literatura (MSL) - um estudo secundário e um estudo terciário. Como etapa inicial desta pesquisa - para ter um entendimento sobre o assunto diversidade em Engenharia de Software foi planejado e executado um estudo secundário de mapeamento sistemático da literatura, apresentada na seção 4.2. Após a adequação da proposta, apresentada no capítulo 1 (seção 1.2.3) devido à pandemia do coronavírus (COVID-19), desenvolveu um estudo terciário de revisões sistemáticas com foco no ensino de Engenharia de Software, apresentada na seção 4.3.

4.1 O que é um Mapeamento Sistemático da Literatura?

De acordo com Petersen [72], um mapeamento sistemático da literatura (MSL) é uma revisão ampla dos estudos primários e secundários existentes em um tópico de pesquisa específico que visa identificar a evidência disponível nesse tópico. São estudos que fornecem dados quantitativos, através de relatórios categorizados, fornecendo um mapa visual dos resultados coletados. Assim, um MSL é um estudo secundário que tem como objetivo identificar e classificar a pesquisa relacionada a um tópico amplo de pesquisa.

Os resultados de um MSL ajudam a identificar os assuntos que estão sendo falados no momento e lacunas na área, capazes de sugerir pesquisas futuras. Ela serve como base para posicionar adequadamente novas atividades de pesquisa, segundo Petersen [73]. Assim, MSLs visam prover uma visão geral de um tópico e identificar se há subtópicos nos quais mais estudos primários ou secundários são necessários [73]. Um mapeamento sistemático da literatura é um tipo de método de pesquisa muito utilizado na área de Engenharia de Software. O processo seguido para elaboração da MSL foi descrito por Bailey et al. [50], conforme ilustrado na Figura 4.1 e criado nesse estudo para melhor entendimento do passo a passo do processo.

O processo do mapeamento sistemático da literatura possui etapas essenciais para o seu desenvolvimento: definição de questões de pesquisa, revisão do escopo, conduzir a pesquisa, realizar a busca de artigos, triagem dos estudos, selecionar artigos relevantes, palavras-chave de resumos, montar os esquemas de classificação, extração dos dados e processos de mapeamento, conforme Figura 4.1. Cada etapa do processo tem um resultado e o resultado do processo é o mapeamento sistemático da literatura.

A análise dos resultados obtidos pelo mapeamento sistemático é concentrada na apresentação das frequências das publicações para cada categoria. Isso possibilita ver quais categorias foram enfatizadas em pesquisas anteriores e, assim, identificar as lacunas

Figura 4.1 – Fluxo do mapeamento sistemático da literatura
Fonte: Adaptado de [72], página 2.

e possibilidades para pesquisas futuras. Os dados coletados são analisados qualitativamente e quantitativamente, onde o material a ser estudado é analisado passo a passo com um controle metodológico rígido. É um procedimento mais intuitivo, maleável e adaptável, visto que depende do entendimento do pesquisador sobre o que está sendo lido para que se possa chegar nas conclusões.

O ciclo de vida da MSL divide-se em três fases:

- Planejamento: etapa onde definem-se as questões de pesquisa e o protocolo de pesquisa é desenvolvido e avaliado.
- Execução: compreende a identificação de artigo de controle, seleção e avaliação inicial dos estudos, e extração e sumarização dos resultados.
- Documentação: fase em que é feita a redação e a verificação do relatório desenvolvido, com base nos resultados encontrados nos artigos estudados.

O protocolo do MSL foi desenvolvido e aplicado em cima dos motores de busca da Scopus. Essa base foi escolhida por ser a maior base de dados de resumos e citações científicas, compilando mais de 50 milhões de registros, 21 milhões de títulos e 5.000 editores, dentre os quais ACM, Elsevier, IEEE, Springer etc.

4.2 MSL: Diversidade em Engenharia de Software

Esta seção, apresenta a execução e os resultados de um estudo secundário de mapeamento sistemático da literatura sobre diversidade em Engenharia de Software.

4.2.1 Questões de Pesquisa

Para atingir ao objetivo foram definidas duas questões de pesquisa, descritas a seguir:

- (RQ1) Quais são os tipos de diversidade encontradas na literatura científica da Engenharia de Software?
- (RQ2) Quais são os principais tópicos investigados dentro dos tipos identificados no presente estudo?

4.2.2 Palavras-chave

De acordo com as questões de pesquisa citadas anteriormente, foram definidas palavras-chave que deveriam ser encontradas nos títulos, palavras-chave, resumos ou texto dos estudos encontrados. São palavras que servem como referência para o andamento do estudo. As palavras-chave são listadas a seguir:

- *Software Engineer, Software Developer, Software Development, Diversity, Gender, Woman, Transgender, LGBT, ethnicity, Racial, ethnic, social, age, socieconomic status, disabilities, geography, sex characteristics*

4.2.3 *String* de busca

Após algumas simulações foi definida a seguinte *string* de busca:

("Software Engineer"OR "Software Developer"OR "Software Development") AND (diversity OR gender OR woman OR transgender OR LGBT OR ethnicity OR racial OR ethnic OR social OR age OR "socioeconomic status"OR disabilities OR geography OR “sex characteristics”)

4.2.4 Validação do protocolo de pesquisa

A validação do protocolo foi realizada a partir dos artigos de controle que tratam sobre os assuntos da área de interesse e indicados por pesquisadores da área. Esses artigos são detalhados a seguir.

4.2.4.1 Identificação dos artigos de controle

Os artigos de controle selecionados para esse estudo serviram como uma lista de validação para garantir a confiabilidade e a relevância das pesquisas. Esses trabalhos foram previamente indicados pelos pesquisadores da área como referência, conforme a seguir:

- *From Diversity by Numbers to Diversity as Process: Supporting Inclusiveness in Software Development Teams with Brainstorming [34]* ;
- *Perceptions of Diversity on Git Hub: A User Survey [100]*;
- *Gender and Tenure Diversity in GitHub Teams [101]*.

4.2.5 Critérios de inclusão

Para dar continuidade ao mapeamento, foram definidos os seguintes critérios de inclusão:

- Artigos de periódicos acadêmicos, conferências e *workshops*;
- Trabalhos escritos em inglês;

4.2.5.1 Critérios de exclusão

Na sequência, são apresentados os seguintes critérios de exclusão:

- Palavras-chave e Resumos que não estão concentradas na Engenharia de Software ou áreas relacionadas;
- Palavras-chave e Resumos em que a palavra Engenharia de Software, desenvolvimento de software ou desenvolvedor de software não esteja relacionada a diversidade ou raça, gênero ou mulher ou transgênero ou étnica ou social ou deficiência ou idade ou LGBT ou etnia ou características sexuais ou geografia ou status socioeconômico;

- Artigos que não estão concentrados na Engenharia de Software ou áreas relacionadas;
- Artigos que não estão concentrados na Engenharia de Software, desenvolvimento de software ou desenvolvedor de software e diversidade ou raça, gênero ou transgêneros ou étnica ou social ou deficiência ou idade ou lgbt ou etnia ou características sexuais ou geografia ou status socioeconômico;
- Anais de conferência; Cursos; *Standards*; Painéis;
- O formato do estudo não está em pdf ou não está disponível.

4.2.6 Mapear estudos que falam sobre diversidade na Engenharia de Software

Sabe-se que existe um desequilíbrio conhecido de gênero e estabelecido nas estruturas de engenharia de software. É de conhecimento também, a falta de representatividade de alguns outros grupos diversos, como, por exemplo, raça/etnia, deficiências e pessoas em situação de vulnerabilidade social. As discussões sobre diversidade de gênero na Engenharia de Software estão aumentando. Mas neste caso, ainda existem pessoas de outras diversidades que não fazem parte das inclusões na engenharia de software.

O objetivo do estudo secundário de mapeamento sistemático foi mapear e identificar uma visão geral da área da pesquisa, por meio da classificação e contagem de contribuições relacionadas ao assunto. Os resultados serão apresentados através de uma lista de artigos sobre como a diversidade está sendo discutida na Engenharia de Software. Nele foi listado os locais de publicações, a frequência de publicações ao longo dos anos, e quais tópicos estão sendo abordados em cada diversidade. Essas informações serão descritas e ilustradas nas próximas subseções.

O presente estudo foi realizado por três pesquisadoras da área, incluindo a autora deste trabalho. Neste caso, identificou-se que as *strings* de busca deveriam ser amplas em relação aos tipos de diversidade. As diversidades estudadas por cada pesquisadora são, diversidade de gênero e diversidade de étnico/racial, e a diversidade social, estudada pela autora e considerada como base para este estudo.

4.2.7 Resultados

Nessa subseção são apresentados os resultados obtidos na execução do mapeamento sistemático da literatura. Os dados foram coletados a partir da análise, leitura e classificação dos artigos selecionados durante o processo.

4.2.7.1 Sumarização dos resultados

A seleção de resultados começou a partir de 5.794 documentos encontrados através das bases de dados (ACM, IEEE e SCOPUS). Em seguida, foram aplicados os critérios de inclusão (4.2.5) e exclusão (4.2.5.1) através das palavras-chaves e os títulos encontrados. Neste caso, o número de artigos diminuiu para 347. Na segunda filtragem foram realizadas as leituras completas dos resumos para identificar se mais artigos poderiam ser removidos, resultando um total de 331 artigos. Para terceira filtragem foram realizadas a eliminação de documentos duplicados da mesma base de dados ou de bases diferentes, totalizando 237 artigos. A leitura completa dos estudos ocorreu na última fase (filtro final), totalizando os 201 artigos para esse mapeamento - a lista final dos artigos encontra-se no Apêndice A. Os resultados detalhados são listados na Tabela 4.1 a seguir:

Tabela 4.1 – Critérios de inclusão/exclusão

Base de Dados	Estudos encontrados	1º Filtro	2º Filtro	Filtro Final
ACM	2604	68	68	42
IEEE	1519	110	94	82
SCOPUS	1671	169	169	77
	5794	347	331	201

4.2.7.2 Estudos publicados por ano

A frequência das publicações ao longo dos anos é apresentada na Figura 4.2, com o número de artigos para cada ano estudado, no período de 2002 a meados de maio de 2020. Não foi aplicada nenhuma restrição temporal à pesquisa, uma vez que o interesse dos autores era em entender a partir de quando a diversidade é um assunto para a pesquisa em Engenharia de Software. O primeiro estudo encontrado é de 2002 e novos estudos foram relatados, em sequência, até o ano atual de 2020. Neste caso, os estudos de 2020 são proporcionais, uma vez que a pesquisa foi realizada em maio de 2020.

4.2.7.3 Locais de publicação

Quanto aos locais de publicação foram encontrados 94 artigos publicados em conferências e 19 artigos publicados em *Workshops*. Na Figura 4.3 são listados o número de artigos em locais de conferências e *Workshops*, com a a maior frequência de publicações. Foram considerados apenas locais com mais de 2 estudos.

Os quatro principais periódicos com mais de uma publicação sobre diversidade na Engenharia de Software são exibidos na Figura 4.4. *IEEE Software* é uma revista e um periódico científico revisados por pares, publicados pela *IEEE Computer Society*, cobrindo

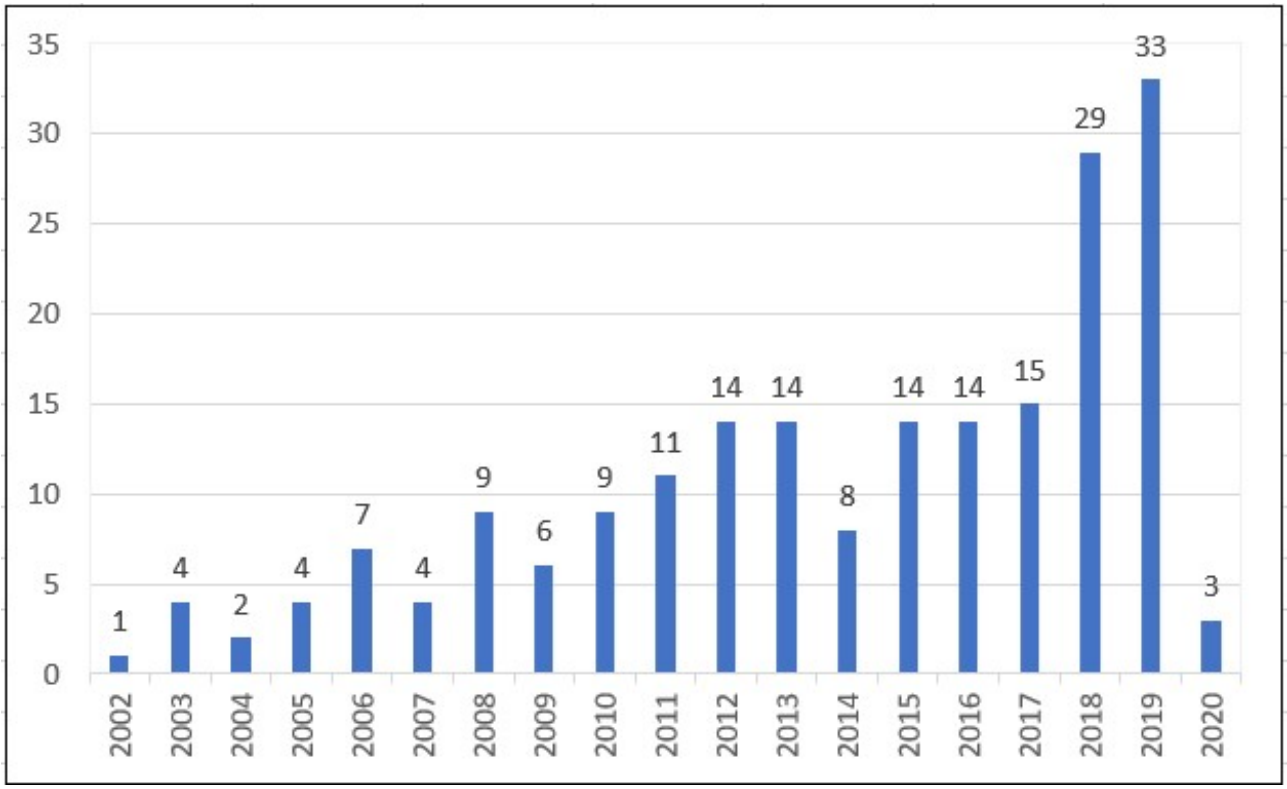

Figura 4.2 – Número de estudos por ano
Fonte: a autora (2020)

todos os aspectos da engenharia, processos e práticas de software. Neste caso, o periódico indica que os estudos são considerados valiosas contribuições científicas, uma vez que são publicados em fóruns de alta qualidade e aparece no topo da lista de periódicos com quatro publicações sobre o assunto. O *ACM Inroads* (ACM) publicou três estudos. O *Journal of Systems and Software* (Elsevier) e *Information and Software Technology* (Elsevier) publicaram dois artigos cada. Outras 11 revistas publicaram um artigo sobre diversidade na Engenharia de Software.

4.2.8 Estudos publicados por tipos de diversidades

A Figura 4.5 ilustra as classificações por tipos de diversidades na Engenharia de software. Conforme a *string* de busca descrita na subseção 4.2.1, as diversidades consideradas foram: diversidade de gênero, diversidade social, diversidade de etnias, diversidade de deficiências, diversidade LGBTQIA+, diversidade de idade, diversidade geográfica e diversidade cultural. A diversidade de gênero teve o maior número de publicações encontradas.

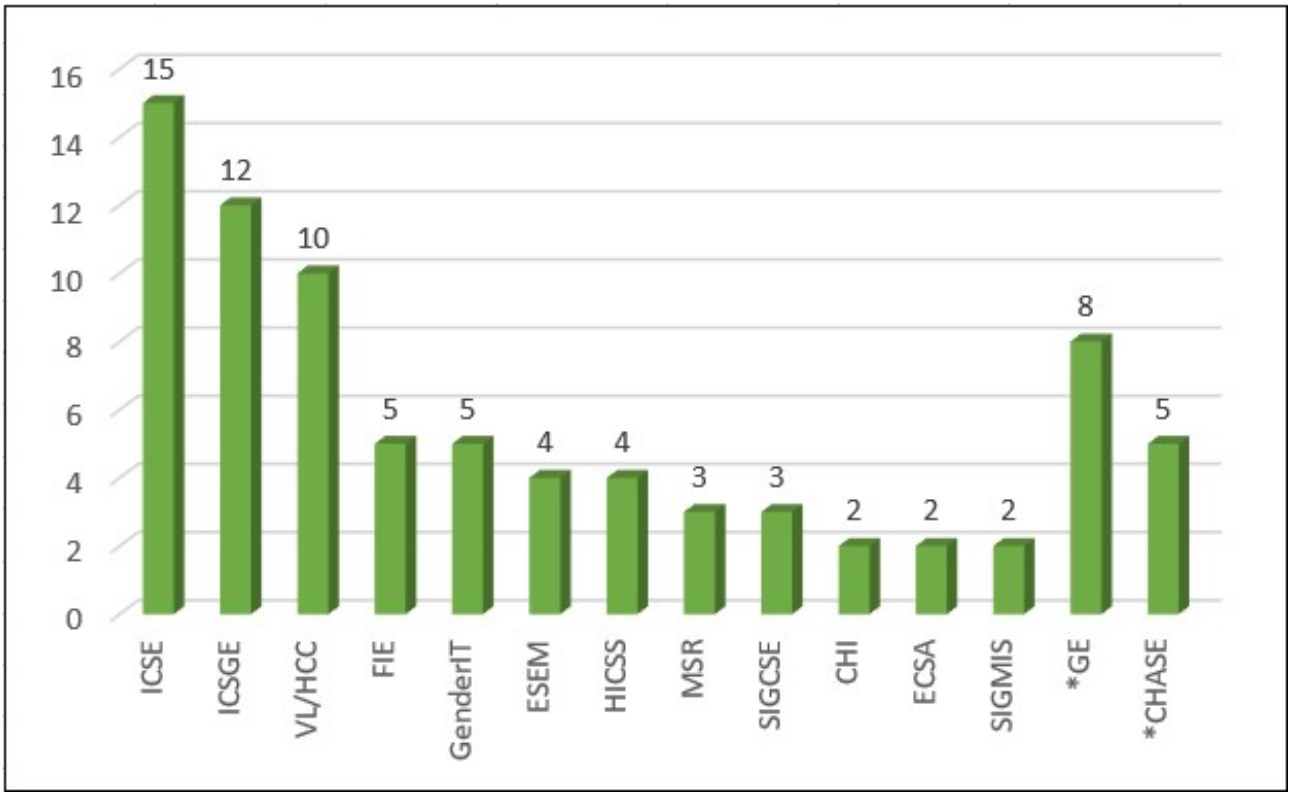

Figura 4.3 – Número de artigos por conferência e workshop*
Fonte: a autora (2020) (*Número de artigos por workshop.)

4.2.9 Estudos sobre Diversidade Social

A execução do mapeamento sistemático da literatura resultou em 6 estudos sobre diversidade social. Esses artigos foram importantes para o andamento desse estudo. As discussões sobre cada estudo serão apresentadas no subseção 4.2.10.

4.2.10 Discussão

Foram classificados 6 artigos no MSL classificados como diversidade social, conforme a Figura 4.5 apresentadas por tipos na subseção 4.2.8. Os estudos foram identificados como: [A54], [A66], [A113], [A147], [A171] e [A194].

4.2.10.1 Diversidade Social

Anuradha et al. [27] definem a diversidade social como o conjunto de diferenças e valores de todos os seres humanos compartilhados nas relações sociais, a coexistência de diferentes grupos sociais dentro de um determinado cenário geopolítico ou, em termos mais simples, a diferenciação da sociedade em grupos. Essas diferenças são expressas através

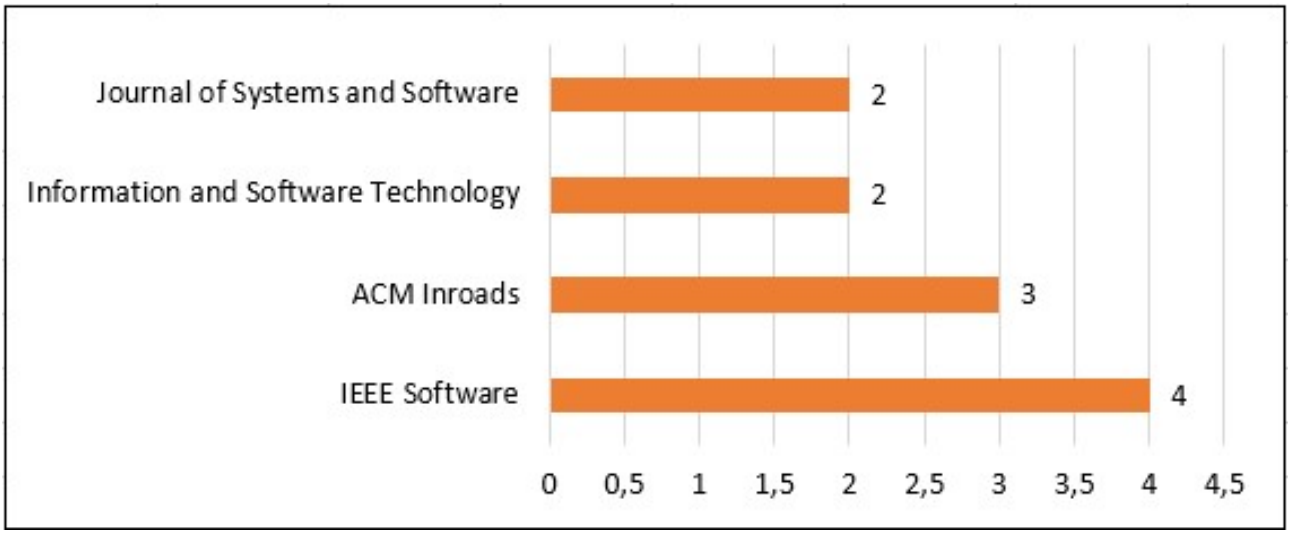

Figura 4.4 – Número de artigos por *journals*
Fonte: a autora (2020)

do idioma, cultura, etnia, rituais, entre outros. Além disso, considera-se outros termos como "pluralidade", "multiculturalismo", "diferenciação social"como uma definição de diversidade social.

4.2.10.2 Ler histórias digitais: dar voz as pessoas socialmente excluídas em vários contextos

O Moutafidou et al. [64] [A54] intensificam sobre a importância da demanda relacionada a inclusão social, em um momento em que a tecnologia se transformou parte integrante da sociedade moderna e das mudanças sociais. A amplitude do uso das tecnologias e seus grandes avanços resultaram em oportunidades para as pessoas explorarem novas formas de comunicação, interação, expressão e participação na sociedade. Nem todas as pessoas têm acesso à essas possibilidades. A inclusão social é de extrema importância para o acesso de todos.

4.2.10.3 Fatores sociais em Engenharia de Software: as divergências, a comunidade e as práticas técnicas

O artigo de Sharp et al. [90] [A194] estudaram sobre o impacto e os problemas dos fatores humanos em Engenharia de Software. Eles mencionam o impacto de um determinado indivíduo; a importância da comunidade e a natureza social das práticas técnicas. Esse fator é decorrente do trabalho realizado para entender o desenvolvimento ágil e, em particular, a construção do software. O artigo está focado em fatores sociais, isto é, fatores que surgem das várias interações que precisam ocorrer para desenvolver o software. Ele

Categoria	Nº de estudos	Estudos
Gênero	65	A3, A8, A11, A15, A19, A27, A28, A31, A33, A34, A38, A51, A55, A56, A60, A67, A69, A71, A72, A73, A74, A77, A78, A79, A80, A81, A82, A83, A84, A85, A86, A87, A88, A90, A91, A92, A94, A97, A101, A103, A105, A106, A108, A114, A119, A123, A128, A129, A131, A141, A143, A145, A149, A157, A160, A166, A178, A180, A181, A183, A184, A185, A189, A190, A191
Cultura	56	A2, A7, A12, A16, A17, A21,A23, A30, A35, A39, A41, A42, A43, A44, A45, A46, A47, A48, A49, A50, A58, A59, A62, A93, A102, A104, A107, A109, A118, A121, A134, A136, A139, A140, A142, A144, A148, A151, A155, A159, A162, A164, A168, A169, A173, A175, A176, A182, A186, A187, A192, A193, A196, A198, A199, A200
Identidades	43	A1, A4, A5, A6, A10, A12, A14, A20, A22, A25, A26, A29, A32, A36, A37, A40, A52, A57, A68, A70, A75, A76, A89, A96, A98, A117, A122, A126, A130, A132, A135, A138, A146, A153, A156, A158, A165, A167, A170, A172, A174, A188, A201
Cognitivo	15	A9, A24, A53, A61, A99, A115, A116, A120, A133, A150, A152, A161, A163, A167, A197
Idade	12	A18, A63, A95, A110, A111, A112, A125, A127, A137, A177, A179, A195
Social	6	A54, A66, A113, A147, A171, A194
Raça/Etnia	2	A65, A124
Deficiência	1	A154
LGBT	1	A100

Figura 4.5 – Relação dos tipos de diversidade x quantidade de estudos
Fonte: a autora (2020)

tem como base três estudos diferentes realizados nos últimos dez anos para identificar três fatores específicos e seu impacto no desenvolvimento de software.

4.2.10.4 Enriquecendo o aprendizado de trabalho por sua diversidade: combinar o aprendizado de serviço universitário e a responsabilidade social corporativa para ajudar as ONGs a adaptar a tecnologia às suas necessidades

Chang et al. [20] [A66] trouxe a importância da responsabilidade social corporativa e o uso do serviço comunitário de aprendizado para melhorar o engajamento dos estudantes de comunidades marginalizadas. O documento fornece reflexões baseadas em campo do envolvimento sustentado. Ele sugere maneiras pelas quais as experiências de aprendizado de serviço podem criar espaço para inovação e crescimento no setor sem fins lucrativos, nas comunidades locais, nas empresas e nos próprios estudantes. O estudo também relata sobre o aprimoramento das habilidades de comunicação e pensamento crítico dos alunos. Era responsabilidade dos estudantes e da Organização Não Governamental (ONG) identificar, definir, esclarecer e, assim, priorizar os problemas existentes. Os alunos experimentaram um senso de social, solidariedade e vínculos em uma sociedade que une as pessoas.

4.2.10.5 O seu software tem valor?

Whittle [104] [A113] relata que geralmente na construção de software são ignorados os valores humanos - deveriam ser considerados os valores humanos mais amplos, como compaixão, responsabilidade social e justiça durante o desenvolvimento de software.

A maneira como o software é projetado influencia fundamentalmente a sociedade, mas os valores humanos - sobre os quais todos devem se preocupar, têm sido uma preocupação secundária na engenharia de software.

4.2.10.6 O status socioeconômico e uso do computador: projetando software para usuários de vulnerabilidade social

O estudo de Hill [45] [A147] foi mencionado no capítulo 2, seção 2.5, subseção 2.5.4, como um dos trabalhos relacionados. Ela apresenta as diferenças entre indivíduos com Status Socioeconômico baixo e alto, formulando hipóteses sobre como essas diferenças podem impactar suas estratégias de solução de problemas durante o uso do software. Este estudo está focado no ensino da Engenharia de Software impactando em pessoas com status socioeconômico baixo.

4.2.10.7 Programação para iniciantes

Segundo Sute Lei [54][A171], as tarefas de desenvolvimento de software são tradicionalmente realizadas por desenvolvedores de sistemas altamente qualificados e bem treinados. Muitas vezes, é necessária uma sólida formação educacional para entender as linguagens de programação usadas no desenvolvimento. Existem certas dificuldades em relação as pessoas com desvantagens educacionais, nas tarefas de desenvolvimento de software. O autor apresenta uma possibilidade natural de atender a essa necessidade, fornecendo uma interface de usuário fácil de entender, com a capacidade de ilustrar a lógica de programação de maneira simples e objetiva.

A tabela 4.2 mostra os tópicos da Engenharia de Software abordados nos estudos, os números de estudos, as identificações e o percentual de cada tópico.

A figura 4.6 apresenta uma combinação entre os tópicos da Engenharia de Software, relacionadas aos tipos de diversidades, mapeadas nesse estudo. Os tópicos sobre Engenharia de Software com maior número de estudos são: educação de Engenharia de Software, construção de software e desempenho e produtividade de equipe, conforme demonstrados na Tabela 4.2. Os tipos de diversidade, com maior número de estudos, de acordo com a Figura 4.5 na subseção 4.2.8, são: gênero, cultura e identidades.

De modo geral, os estudos sobre educação em Engenharia de Software abordam novos modelos e métricas do ensino e da prática de engenharia de software direcionadas aos estudantes e grupos de capacitação profissional de diversidade (étnico, idade e gênero). No tópico Construção de Software, observa-se os estudos que abordam sobre o desenvolvimento de software em comunidades virtuais, destacando a participação em comunidade *Open Source* e analisando as contribuições da diversidade em projetos em Engenharia de Software. Em relação ao tópico desempenho e produtividade de equipes, os

Tabela 4.2 – Estudos classificados por tópicos de Engenharia de Software

Tópicos de Engenharia de Software	Nº	Estudos	%
Educação em ES	42	A5, A8, A12, A15, A17, A25, A28, A29, A33, A62, A63, A64, A66, A67, A71, A77, A78, A83, A86, A95, A104, A111, A112, A115, A123, A124, A135, A136, A143, A145, A147, A149, A150, A156, A157, A170, A171, A178, A181, A183, A186, A190	20,90%
Construção de Software	36	A27, A34, A36, A40, A59, A74, A79, A80, A81, A82, A87, A88, A90, A91, A94, A96, A108, A110, A113, A116, A119, A127, A128, A129, A130, A144, A146, A159, A160, A184, A185, A188, A189, A191, A194, A201	17,91%
Desempenho / Produtividade	32	A9, A13, A16, A18, A20, A22, A24, A37, A38, A39, A48, A53, A60, A61, A65, A70, A76, A102, A103, A125, A134, A138, A153, A155, A163, A166, A169, A174, A177, A179, A182, A196	15,92%
Times Globais	31	A2, A21, A23, A31, A35, A42, A44, A45, A47, A49, A50, A58, A69, A93, A100, A107, A118, A121, A133, A142, A151, A162, A168, A173, A175, A176, A187, A192, A198, A199, A200	15,42%
Composição de Equipe	29	A4, A7, A10, A11, A19, A26, A52, A55, A57, A72, A73, A75, A85, A89, A92, A99, A105, A106, A114, A122, A131, A132, A137, A141, A152, A158, A165, A172, A180	14,43%
Prática profissional em ES	14	A3, A30, A43, A51, A84, A120, A126, A139 A148, A161, A167, A193, A195, A197	6,97%
Métodos ágeis	7	A1, A14, A41, A56, A98, A101, A117	3,48%
Requisitos de Software	4	A68, A109, A140, A164	1,99%
Adaptação de processos	2	A97, A154	1,00%
Fundamentos da Computação	1	A54	0,50%
Fundamentos Matemáticos	1	A6	0,50%
Gerenciamento de ES	1	A46	0,50%
Teste de Software	1	A32	0,50%

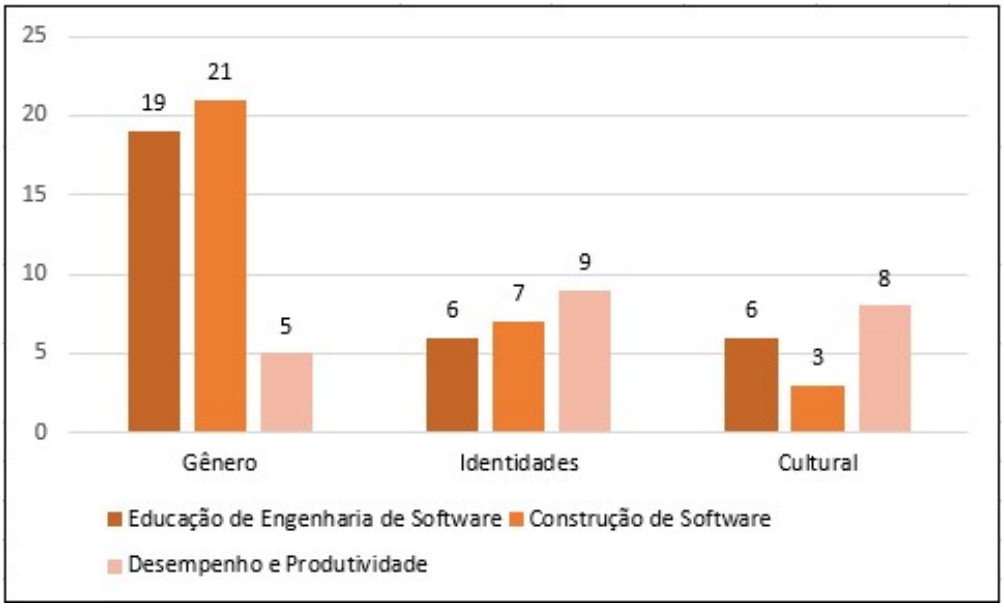

Figura 4.6 – Tópicos de Engenharia de Software x diversidades
Fonte: a autora (2020)

estudos classificados, buscam identificar índices para analisar o desempenho de equipes compostas por diversidade, como forma de justificar a adoção ou não de práticas inclusivas.

4.2.11 Conclusão

Neste estudo foram apresentados os resultados de um estudo secundário de mapeamento sistemático da literatura sobre diversidade em Engenharia de Software. Os resultados foram apresentados sobre todos os tipos de diversidade em Engenharia de Software encontrados no MSL. A discussão está com foco em diversidade social, tema chave para o andamento desse estudo e a Síntese da Discussão com foco em diversidade em tópicos da Engenharia de Software.

4.3 Estudo terciário: Ensino de Engenharia de Software

Esta seção apresenta a execução e os resultados de um estudo terciário de revisão sistemática sobre ensino de Engenharia de Software em estudos secundários de mapeamentos e revisões sistemáticas da literatura.

4.3.1 Questões de pesquisa

Para atingir o objetivo da pesquisa foram definidas duas questões de pesquisa, descritas a seguir:

- (RQ1) Como a Engenharia de Software é ensinada em ambientes de ensino?
- (RQ2) Quais modelos de currículo (corpos de conhecimento) foram usados para planejar as estratégias de ensino de Engenharia de Software?

4.3.2 Palavras-chave

De acordo com as questões de pesquisa citadas anteriormente, foram definidas palavras-chave que deveriam ser encontradas nos títulos, palavras-chave, resumos ou texto dos estudos encontrados. São palavras que servem como referência para o andamento do estudo. As palavras-chave são listadas a seguir:

- *Software Engineer, Education, Learn, Teach, Methodology, Practices, Techniques, Skills, Systematic literature mapping, Systematic literature review*

4.3.3 *String* de busca

Após algumas simulações, foi definida a seguinte *string* de busca:

“Software engineering education” AND (“Systematic literature mapping” OR “Systematic literature review”)

4.3.4 Validação do protocolo de pesquisa

A validação do protocolo foi realizada a partir dos artigos de controle que tratam sobre os assuntos da área de interesse e indicados por pesquisadores da área. Esses artigos são detalhados na próxima subseção.

4.3.4.1 Identificação dos artigos de controle

Os artigos de controle selecionados para esse estudo serviram como uma lista de validação para garantir a confiabilidade e a relevância das pesquisas. Esses trabalhos foram previamente indicados pelos pesquisadores da área como referência, conforme a seguir:

- *A systematic mapping study on practical approaches to teaching software engineering [56]*
- *What Software Engineering "Best Practices" are we Teaching Students-a Systematic Literature Review [55]*

4.3.5 Critérios de inclusão

Para dar continuidade ao estudo, foram definidos os seguintes critérios de inclusão:

- Estudos que falam sobre educação em engenharia de software ou áreas afins;
- Estudos que falam sobre abordagens, técnicas e ou metodologias de ensino de Engenharia de software;
- Palavras-chaves, títulos ou resumos que falam sobre educação em engenharia de software;
- Palavras-chaves, títulos ou resumos que falam sobre abordagens, técnicas e ou metodologias de ensino de Engenharia de software;
- Artigos de periódicos acadêmicos, conferências e *workshops*;
- Trabalhos escritos em inglês e português (pt-br);

4.3.5.1 Critérios de exclusão

Na sequência, são apresentados os seguintes critérios de exclusão:

- Estudos duplicados;
- Palavras-chaves, títulos ou resumos que não falam sobre educação em engenharia de software;
- Palavras-chaves, títulos ou resumos que não falam sobre abordagens, técnicas e ou metodologia de ensino de Engenharia de software;

- Estudos que não falam sobre educação em engenharia de software;
- Estudos que não falam sobre abordagens, técnicas e ou metodologias de ensino de Engenharia de software;
- Anais de conferência; Cursos; *Standards*; Painéis;
- O formato do estudo não está em pdf ou não está disponível.

4.3.6 Mapear estudos que falam sobre ensino de Engenharia de Software

Para Ouhbi et al. [67] o ensino de Engenharia de Software visa reunir a teoria e prática, para que a estudante consiga desenvolver uma compreensão aprofundada dos conceitos e princípios fundamentais, juntamente com as habilidades e competências para resolução de problemas do mundo real. Entretanto, não é apenas uma preocupação acadêmica com o ensino de tópicos relevantes, mas também a responsabilidade de moldar profissionais adequados à demanda da exigência da indústria de desenvolvimento de software. A indústria de tecnologia da informação está em crescente desenvolvimento e de forma transnacional [4], vários desafios estão enfrentando a qualificação de engenheiros de software, capazes de desenvolver produtos de acordo com os padrões industriais internacionais em mercados externos.

Em 2018, a Engenharia de Software completou 50 anos. Ela é uma disciplina jovem e promissora que ainda está em desenvolvimento e evolução - isso ocorre também no ensino de Engenharia de software. Segundo Pombo et al. [76] nos últimos anos, várias abordagens pedagógicas foram implementadas com o intuito de melhorar a motivação e o envolvimento das pessoas estudantes. A adoção de tecnologias em salas de aula, inibiu modelos tradicionais em termos de interação entre professoras e alunas, nas dinâmicas em grupo, entregas e experiências de aprendizagem. A lógica por trás dessas abordagens tem como intuito produzir soluções inovadoras, permitindo um local remoto ou como base de aprendizagem, em que o palestrante pode providenciar as atividades de aprendizagem para promover o pensamento das alunas, melhorando os níveis cognitivos e habilidades na resolução de problemas deles.

O objetivo da revisão foi identificar em estudos secundário de mapeamentos e revisões sistemáticas, uma visão geral da área da pesquisa, por meio da classificação e contagem de contribuições relacionadas ao assunto. Os resultados serão apresentados através de uma lista de estudos sobre como a Engenharia de Software é ensinada. Nele foi listado os locais de publicações, a frequência de publicações ao longo dos anos, e quais tópicos estão sendo abordados em Engenharia de Software. Essas informações serão descritas e ilustradas nas próximas subseções.

O presente estudo foi realizado pelo professor Orientador e a autora desta dissertação. Neste caso, identificou-se que as *strings* de busca deveriam ser amplas em relação as formas de ensino de Engenharia de Software, em estudos secundários de revisões e mapeamentos. Foram considerados estudos de mapeamentos e revisões sistemáticas da literatura publicados em conferências e *Workshops* da área de Engenharia de Software.

4.3.7 Resultados

Nessa subseção são apresentados os resultados obtidos na execução do estudo terciário. Os dados foram coletados a partir da análise, leitura e classificação dos artigos selecionados durante o processo.

4.3.8 Sumarização dos resultados

A seleção de resultados começou a partir de 155 documentos encontrados através das bases de dados (ACM, IEEE, *ScienceDirect* e SCOPUS). Em seguida, foram aplicados os critérios de inclusão (subseção 4.3.5) e exclusão (subseção 4.3.5.1) através das palavras-chaves e os títulos encontrados. Neste caso, o número de artigos diminuiu para 58. Na segunda filtragem foram realizadas as leituras completas dos resumos para identificar se mais artigos poderiam ser removidos, resultando um total de 57 artigos. Para terceira filtragem foram realizadas a eliminação de documentos duplicados da mesma base de dados ou de bases diferentes, totalizando 8 artigos duplicados. A leitura completa dos estudos ocorreu na última fase (filtro final), totalizando os 20 artigos para esse estudo. Uma lista final com todos os estudos encontrados é listada no Apêndice B. Os resultados detalhados são listados na Tabela 4.3 a seguir:

Tabela 4.3 – Critérios de inclusão/exclusão

Base de Dados	Estudos encontrados	1º Filtro	2º Filtro	Duplicados	Filtro Final
ACM	48	15	15	2	0
IEEE	50	22	21	2	9
ScienceDirect	18	12	12	1	6
SCOPUS	39	9	9	3	5
	155	58	57	8	20

4.3.9 Estudos publicados por ano

A frequência das publicações ao longo dos anos é apresentada na Figura 4.7, com o número de artigos para cada ano publicados. Não foi aplicada nenhuma restrição temporal à pesquisa, uma vez que o interesse das pessoas autoras era em entender a partir de quando o ensino era assunto para a pesquisa em Engenharia de Software. O primeiro estudo encontrado é de 2010 e novos estudos foram relatados, em sequência, até o ano atual de 2020. Neste caso, os estudos de 2020 são proporcionais, uma vez que a pesquisa foi realizada entre agosto a setembro de 2020.

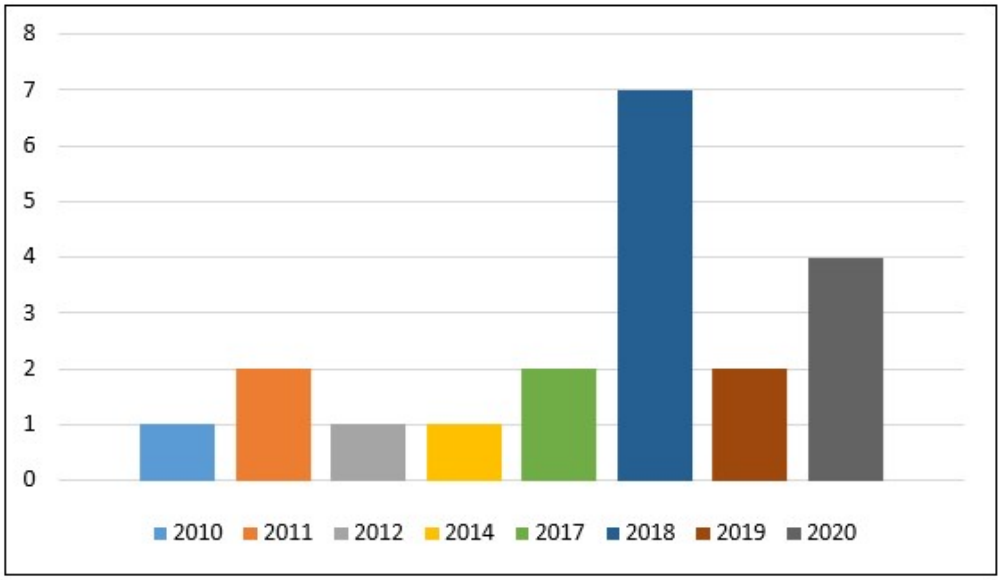

Figura 4.7 – Número de estudos por ano
Fonte: a autora (2020)

4.3.10 Locais de publicação

Quanto aos locais de publicação foram encontrados 9 artigos publicados em conferências. A Figura 4.8 lista o número de artigos em locais de conferências. A conferência com a maior frequência de publicações foi a *Frontiers in Education Conference (FIE)* com 3 estudos. Os demais têm apenas um estudo cada.

Os sete principais periódicos com uma publicação ou mais sobre ensino de Engenharia de Software são exibidos na Figura 4.9. O *Journal of Systems and Software* (Elsevier) publica artigos que cobrem todos os aspectos da Engenharia de Software - foram encontrados 6 estudos da *Elsevier*. O *Computer Applications in Engineering Education, Computer Standards Interfaces, Computers Education, IEEE Software, IEEE Transacti-*

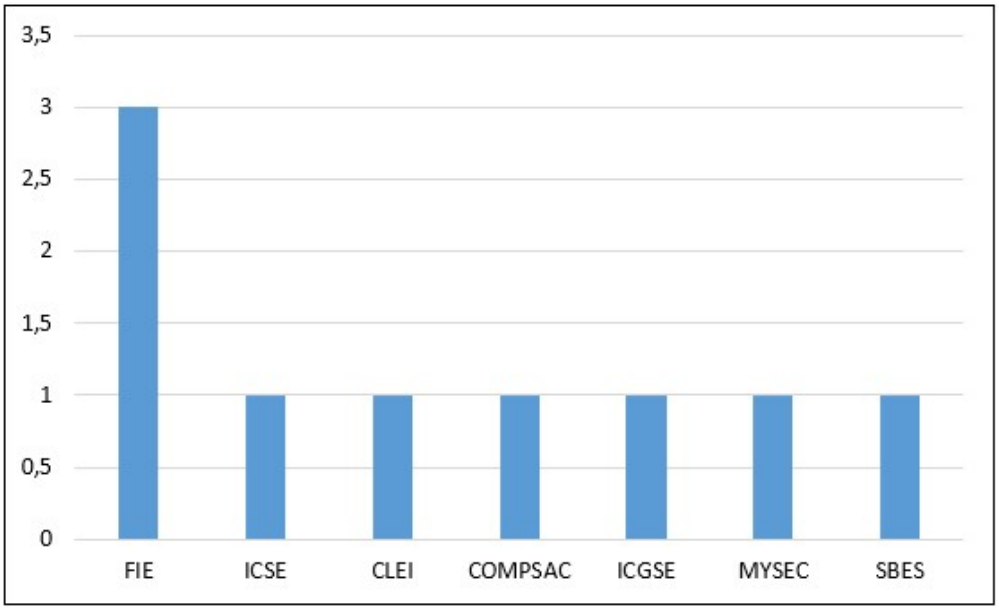

Figura 4.8 – Número de artigos por conferência
Fonte: a autora (2020)

ons on Software Engineering e *Information and Software Technology* publicaram um estudo cada.

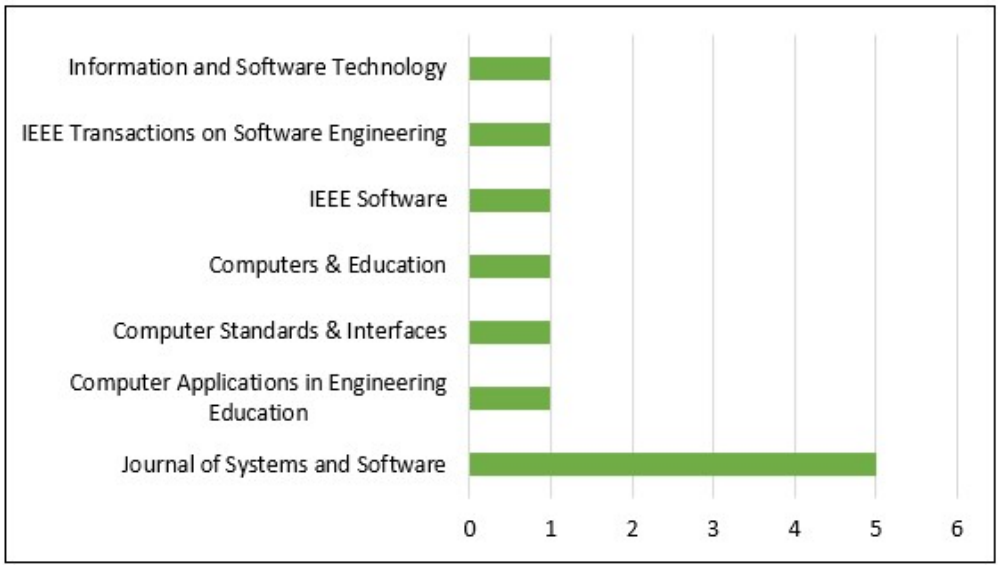

Figura 4.9 – Número de artigos por *journals*
Fonte: a autora (2020)

4.3.11 Estudos sobre ensino de Engenharia de Software

A Tabela 4.4 apresenta os estudos relacionadas por formas, técnicas ou abordagens de ensino de Engenharia de Software. Os estudos em detalhes serão discutidos na subseção 4.3.12.

Tabela 4.4 – Estudos por tópicos de ensino de Engenharia de Software

Tópicos de Ensino	Quantidade	Identificação
Gamificação, Jogos Educacionais, Aprendizagem baseada em jogos (GBL)	5	[A1], [A2], [A10], [A11], [A18]
Abordagens Práticas	3	[A3], [A6], [A20]
Alinhamento com as necessidades industriais	2	[A4], [A5]
Aprendizagem baseada em projetos (PBL)	2	[A17], [A8]
Currículo de Engenharia de Software	1	[A15]
Desenvolvimento de Software Global	1	[A14]
Ensino em Teste de Software	1	[A16]
Engenharia de Software Baseada em Evidências	1	[A19]
FLOSS	1	[A9]
MOOCs	1	[A13]
Programação em pares	1	[A7]
Práticas Lúdicas	1	[A12]
	20	

Conforme apresentado na Tabela 4.4, o maior número de estudos encontrados e classificados foi na área de Jogos: *Gamificação*[1], Jogos educacionais e Aprendizagem baseada em jogos (GBL), com 5 estudos no total. Foram classificados 3 estudos como Abordagens práticas em geral. Os 2 estudos [A4][A5] são estudos focados no ensino com base nas necessidades da indústria de desenvolvimento de software. A aprendizagem baseada em projetos com 2 estudos. Os demais: currículo de Engenharia de Software, desenvolvimento de software global, ensino em teste de software, Engenharia de Software baseada em evidências, FLOSS, MOOCs, programação em pares e práticas lúdicas foram classificados apenas um estudo cada

4.3.12 Discussão

Nesta subseção serão apresentados a discussão dos estudos de mapeamentos e revisões sistemáticas da literatura, encontrados e classificados, sobre ensino de Engenharia de Software. A Tabela 4.4 apresentada na subseção 4.3.11 foi organizada por tópicos

[1] De acordo com Deterding et al. [29], gamificação é um termo relativamente novo que tem sido usado para denotar o uso de elementos de jogos e técnicas de design de jogos em contextos não relacionados a jogos

de ensino, quantidade e identificação para cada estudo do mapeado. Os tópicos mais numerosos serão abordados nas próximas Subseções.

4.3.12.1 Gamificação, Jogos Educacionais e Aprendizagem Baseada em Jogos (GBL)

O estudo de Calderon et al. [16] [A1] foi conduzido por meio uma revisão da literatura multi vocal sobre *Serious Games* para ensino de padrões de processo de Software. Os resultados da revisão revelaram que o *Serious Games* tem potencial como ferramentas de suporte para ensino de padrões de processo de software, mas ainda é necessário mais pesquisas e resultados experimentais para observar o potencial dos *Serious Games* - (SGs) como recursos de aprendizagem. O intuito do estudo era entender quais SGs para ensino sobre padrões de software estavam disponíveis, como eles eram avaliados e quais os benefícios eram percebidos com o uso de SGs no contexto do ensino de padrões do processo de software. Foram encontrados 3 estudos no total (pós remoção dos filtros iniciais e duplicados), com um total de 6 SGs diferentes:

- *Prodec* - é um SGs baseado em simulação para ensinar motivar os desenvolvedores de software na aprendizagem e prática dos princípios de gerenciamento de projetos de software;
- *Floors* - é um SGs que propõe uma experiência de aprendizagem interativa para introduzir ISO/IEC 12207:1995, por meio da criação de diferentes andares de um ambiente virtual onde vários processos da norma são discutidos e implementados;
- *Go for it!* - é um jogo educacional não tecnológico para contribuir com o ensino dos elementos da norma ISO / IEC 29110 onde os jogadores são incentivados a compreender os detalhes básicos do processo de gerenciamento de projetos;
- *Problems and programmers (PnP)* - é um jogo de cartas educacional que simula o processo de engenharia de software e é projetado para ensinar as questões de processo que não podem ser suficientemente tratadas por palestras e projetos educacionais;
- *DesignMPS* - é um jogo de computador projetado para apoiar o ensino de modelagem de processos de software, reforçando conceitos relevantes e fornecendo exercícios de modelagem de processos de software;
- *SimSE* - é um jogo de computador educacional, interativo e totalmente gráfico que simula processos de engenharia de software, e é projetado especificamente para treinar alunos em situações que requerem um entendimento e tratamento de problemas de processo de software.

Mauricio et al. [59][A2] realizaram um estudo de mapeamento sistemático sobre métodos relacionados a jogos para de ensino de Engenharia de Software. Segundo os autores, existem algumas áreas de conhecimento onde o uso de jogos ainda pode ser mais

explorado. A *gamificação* é uma nova tendência e as pesquisas existentes na área são bastante preliminares. Os autores notaram também, uma falta de padronização tanto na definição dos objetivos de aprendizagem quanto na classificação dos métodos relacionados ao jogo. O objetivo do estudo foi identificar os métodos relacionados a jogos para o ensino de Engenharia de Software, entender como esses métodos relacionados aos jogos suportam as áreas de conhecimento da engenharia de software do SE 2014 e compreender as características específicas de cada método relacionado ao jogo.

O estudo de Alhammad et al. [2] [A10] utilizou como método de pesquisa um mapeamento sistemático da literatura para estudar a *Gamificação* para ensinar Engenharia de Software. Eles descobriram que o objetivo de aplicar a *Gamificação* no campo da Engenharia de Software está diretamente relacionado a melhorar o envolvimento do aluno e, em menor grau, a melhorar o conhecimento do aluno, embora outros alvos sejam a aplicação das melhores práticas da Engenharia de Software e socialização. Foram mencionadas questões perspicazes sobre o custo de implementação da *Gamificação*, padrões nos elementos de *Gamificação* usados com mais frequência, os processos de ES e as atividades de ensino abordadas.

Petri et al. [74] [A11] realizaram um estudo para avaliar os jogos educacionais para ensinar computação por meio de uma revisão sistemática da literatura. Os Jogos educacionais são consideradas uma estratégia de ensino para ensinar computação. A revisão sistemática buscou 3617 artigos, dos quais 112 artigos relevantes foram identificados, descrevendo 117 estudos sobre avaliação de jogos para educação em computação. Com base nesses estudos, foram analisados como as avaliações são definidas (os fatores de análise avaliados, projetos de pesquisa, modelos / métodos de avaliação usados, tipos de instrumentos de coleta de dados etc.), como foram executadas (tamanho da amostra e replicações) e analisados (dados métodos de análise utilizados). Como resultado, os autores confirmaram que a maioria das avaliações usa um desenho de pesquisa simples no qual, normalmente, o jogo é usado e, posteriormente, o *feedback* subjetivo é coletado por meio de questionários das pessoas estudantes. A maioria das avaliações é executada com pequenas amostras, sem replicação, usando principalmente métodos qualitativos para análise de dados.

Já Garcia et al. [37] [A18] realizaram uma revisão sistemática da literatura para identificar os efeitos da aprendizagem baseada em jogos na aquisição de "habilidades pessoais"em cursos de graduação em engenharia de software. A aprendizagem baseada em jogos (GBL) combina a aprendizagem com diferentes recursos conhecidos, como jogos, para apoiar e melhorar o processo de ensino / aprendizagem e / ou avaliação do aluno por meio da aprendizagem ativa. O estudo buscou o uso do GBL para o ensino de Engenharia de Software na graduação, de 2001 a 2020, abordando quatro questões de pesquisa:

- Que tipos de jogos foram desenvolvidos para o ensino de engenharia de software?

- Quais áreas de engenharia de software foram abordadas por esses jogos?
- Quais habilidades pessoais foram promovidas com o uso desses jogos?
- Como essas habilidades foram avaliadas?

O estudo encontrou 96 estudos para responder as quatro perguntas, mencionadas anteriormente. As descobertas forneceram evidências sobre o desenvolvimento de jogos digitais com foco no ensino dos fundamentos da engenharia de software definidos pelo corpo de conhecimento da Engenharia de Software. Além disso, esses jogos foram capazes de promover a aquisição de mais de uma habilidade benéfica para alunos de graduação. As descobertas forneceram evidências sobre o desenvolvimento de jogos digitais com foco no ensino dos fundamentos da Engenharia de Software definidos pelo corpo de conhecimento de ES.

4.3.12.2 Abordagens práticas

Foram encontrados 3 estudos classificados como abordagens práticas como abordagens de ensino de Engenharia de Software. A Engenharia de Software baseada na prática parece ser a melhor abordagem instrucional para lidar com a educação modernas em Engenharia de Software [56].

O estudo de Marques et al. [56] [A3] desenvolveu um estudo de mapeamento sistemático sobre abordagens práticas para ensinar Engenharia de Software. Os autores mencionam que atualmente não existe uma abordagem predominante, nem uma forma certa ou errada de conduzir essas experiências práticas, mas certamente existem abordagens que são mais eficazes do que outras, dependendo do contexto de ensino.

Foram selecionados e classificados 173 estudos. O mapeamento sistemático mapeou a abordagem "aprender fazendo"é a mais utilizada, com 93 estudos (54%), seguida da abordagem PBL / OBL com 21 estudos (12%), estudos de caso com 15 estudos (9%), jogos com 12 estudos (7%), e depois simulação (5%), tradicional (4%), Open source (3%), aprendizagem de serviço (2%) e sala de aula invertida com apenas um estudo.

Aprender fazendo e também PBL / OBL são estratégias flexíveis que podem ser ajustadas para aproveitar as capacidades e interesses das pessoas instrutoras e alunas. Essas abordagens não impõem restrições importantes ou requerem recursos especiais das pessoas instrutoras. Essa pode ser a razão pela qual essas alternativas parecem ser frequentemente usadas para apoiar essas experiências práticas.

Angelo e Beer [6][A6] desenvolveram um estudo na literatura para introduzir as abordagens de arquitetura de software em projetos de ensino com metodologias ágeis. Os autores mencionam que as atividades de arquitetura de software não são muito discutidas em métodos ágeis no desenvolvimento de software. Inicialmente foi realizada um *Snowballing* onde foram encontrados 55 estudos sobre a interação entre métodos ágeis e arquitetura

de software. As autoras são alunas de um curso, por meio dos grupos de alunas, desenvolvem projetos reais de software usando Scrum e uma das habilidades necessárias para o desenvolvimento é demonstrar o design do sistema. Foi introduzido um arquiteto de software como parte da equipe e um *Product Owner*[2] mais realista como um único ponto de comunicação entre os arquitetos. Outra abordagem inserida foi revelar para as pessoas alunas que eles podem enfrentar cenários na prática em forma de aula teórica.

Quais são as melhores práticas em Engenharia de Software para ensinar? Por meio de uma revisão sistemática da literatura, Marques et al. [55] [A20] apresentam que é um desafio muito grande ensinar Engenharia de Software, visto que a indústria de software cresce rapidamente e que novas tecnologias de desenvolvimento são lançadas constantemente. Os autores mapearam 17 estudos no total. Algumas metodologias usadas na indústria de desenvolvimento de software são chamadas de "melhores práticas". As diretrizes curriculares de Engenharia de Software foram consideradas como base na comparação das "melhores práticas"de ensino de Engenharia de Software. As 10 áreas do conhecimento foram consideradas nesse estudo. Foram constatadas 57 práticas recomendadas para ensinar Engenharia de Software. Todas as áreas de conhecimento são contempladas com algumas “melhores práticas”; mas em comparação, foi realizado no nível de unidade de conhecimento x “melhores práticas”, mostrou que 67% das unidades de conhecimento estão cobertas por “melhores práticas”. As melhores práticas de x diretrizes curriculares são listadas a seguir:

- Dinâmicas em grupo, psicologia, habilidades de comunicação e profissionalismo - práticas reflexivas, mentores, programação em pares, propriedade coletiva, semana de quarenta horas, revisão em pares, líder de time, redistribuição de atividades, grupos auto-gerenciáveis, contatos metafóricos, reciprocidade, *feedback*, respeito;
- Fundamentos, elicitação, especificação, documentação e validação de requisitos - requisitos;
- Design de Software - design Simples, design padrão, refatoração, análise e design orientado a objetos;
- Verificação e validação de software - Inspeções, prototipação, revisão em pares, análise estatística de teste, testes unitários, testes integrados;
- Processo de Software - PSP, métodos formais, realização de análise de causa e efeito, implementação de projeto de planejamento de integração contínua, gerenciamento de configurações, controle de versões, construção contínua desenvolvimento iterativo e incremental, manutenção;

[2]O Product Owner é responsável por desenvolver e manter o *product backlog*, a lista de *user stories* que definem os requisitos para o projeto. Alguns, *Product Owner* mencionam que nem sempre é necessário o conhecimento deles sobre as melhores práticas em engenharia de requisitos [52].

- Qualidade de software - código padrão, gerenciamento de qualidade e conformidade com as métricas de satisfação do cliente, garantia na qualidade, evolução;
- Fundamentos de segurança - Segurança;

Os demais estudos mencionam outros tópicos de ensino: 2 artigos estudam o Alinhamento com as necessidades de indústria de desenvolvimento de software [39] [A4] e [38] [A5]; Os estudos de Cico et al. [22][A8] e Miyashita et al. [61] [A17] descrevem uma revisão na literatura sobre aprendizagem baseada em projetos (PBL); O estudo de Qadir et al. [83] [A15] relata um estudo de mapeamento sistemático para sintetizar e agregar os esforços relatados relacionados ao currículo de Engenharia de Software e fornecer uma visão geral ampla da área.

Foram encontrados 20 trabalhos sobre mapeamentos e revisões sistemáticas da literatura que estudam as formas de ensino de Engenharia de Software. A Figura 4.10 apresenta uma ilustração com cada percentual de estudos encontrados que mencionam diretrizes curriculares, SWEBOK ou que não mencionam ambos.

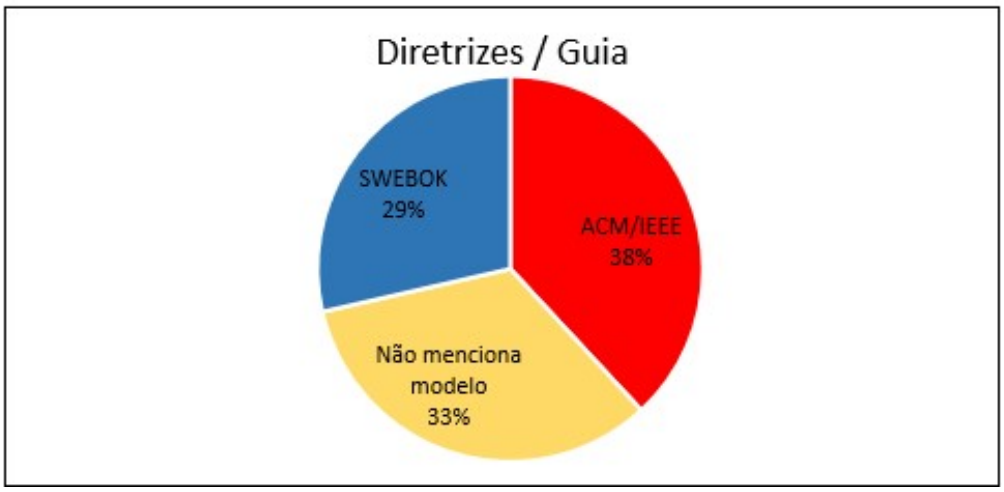

Figura 4.10 – Diretrizes da ACM / IEEE, SWEBOK de ensino de ES
Fonte: a autora (2020)

As diretrizes curriculares para programas de graduação em Engenharia de Software - ACM e IEEE[3] são mencionadas em 7 estudos [A2], [A10], [A11], [A12], [A16], [A19] e [A20] do mapeamento sistemático da literatura. O Guia de conhecimento em Engenharia de Software - SWEBOK[4] são mencionados como base em 5 estudos do MSL [A1], [A4], [A5], [A8] e [A18]. O estudo [A15] possui como base as diretrizes da ACM/IEEE e SWEBOK. O restante dos estudos encontrados (7 no total) não menciona nenhum guia ou diretrizes sobre ensino de Engenharia de Software [A3], [A6], [A7], [A9], [A13], [A14] e [A17]. Os estudos [A7], [A12] e [A17] não mencionam as necessidades da indústria de desenvolvimento de software como foco para o ensino de Engenharia de Software.

[3]Disponivel em: https://www.acm.org/binaries/content/assets/education/se2014.pdf
[4]Disponível em: https://www.computer.org/education/bodies-of-knowledge/software-engineering

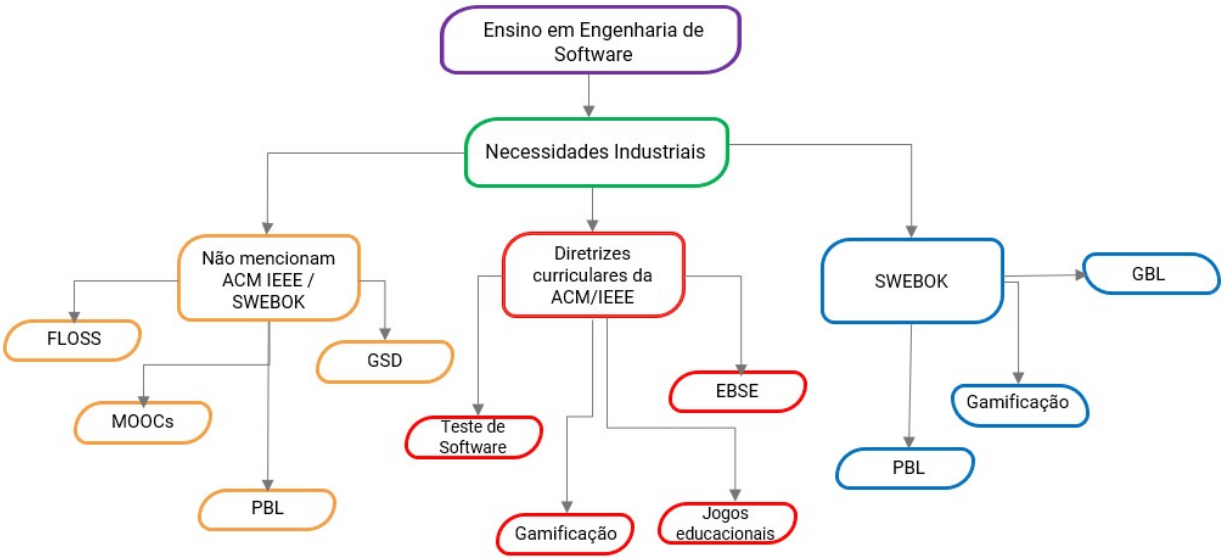

Figura 4.11 – Ensino de ES com foco nas necessidades da indústria
Fonte: a autora (2020)

Um total de 17 estudos apresentam as necessidades da indústria de desenvolvimento de software como foco para o desenvolvimento de formas de ensino de Engenharia de Software. A Figura 4.11 ilustra os tópicos abordados em cada um dos estudos encontrados.

4.3.13 Conclusão

Com esse estudo terciário sobre ensino de Engenharia de Software, foram identificadas as diretrizes curriculares de Engenharia de Software da ACM / IEEE estão presente em 8 estudos, ou seja, 38% dos estudos mapeados (maior parte dos estudos encontrados). As diretrizes serviram como base para o roteiro de perguntas das entrevistas semiestruturadas e nas técnicas de análise de dados da Teoria fundamentada nos dados - (*Grounded Theory*), para desenvolver a proposta deste estudo. As perguntas foram coletadas por meio de entrevista semiestruturadas em pessoas professoras e alunas de um curso de capacitação em Engenharia de Software [95].

5. ESTUDO QUALITATIVO

Neste capítulo são apresentados o embasamento teórico do estudo qualitativo, o planejamento e resultados do estudo.

A pesquisa qualitativa é usada para entender como as pessoas experimentam o mundo. De acordo com Hoepfl et al. [47], embora existam muitas abordagens para a pesquisa qualitativa, ela pode ser flexível e concentrada em reter um significado rico na interpretação dos dados. As abordagens mais comuns incluem a teoria fundamentada em dados, etnografia, pesquisa-ação, pesquisa fenomenológica e pesquisa narrativa. Elas compartilham algumas semelhanças, mas enfatizam objetivos e perspectivas diferentes [47].

Segundo Przyborski et al. [82] as abordagens quantitativas procuram eliminar qualquer interpretação por parte da pessoa pesquisadora. Isso é alcançado por meio de instrumentos de pesquisa padronizados, que devem produzir objetivos, dados confiáveis e válidos, por exemplo, a confirmação de que todos os sujeitos entendam as perguntas da pesquisa, exatamente da mesma forma. As abordagens qualitativas têm como principal objetivo enfatizar a reconstrução do significado subjetivamente dos seres humanos observados por meio de suas ações e artefatos [82]. Esses métodos, às vezes, são chamados de reconstrutivos. Assim, em pesquisas qualitativas, “a pesquisadora é o instrumento” [71] que permite uma reconstrução adequada para coletar o máximo de ações ou comunicações possíveis na coleta de dados [82].

A comunicação é um aspecto central dos métodos qualitativos nas ciências sociais. É organizada em duas formas [109]: primeira - as ações sociais sempre envolvem a comunicação, ou seja, a compreensão e a interpretação do que o outro (cujo envolvimento torna a ação social) quer dizer ou fazer. Comunicação é, portanto, parte integrante do fenômeno social em estudo, seja diretamente na comunicação, ou como parte de outra ação social, como o desenvolvimento de software; segunda - a comunicação da pessoa pesquisadora com os sujeitos tem como objetivo coletar dados, seja diretamente ou indiretamente, por exemplo, em entrevistas ou por meio de observação. O ponto chave é entender o que os sujeitos pretendem dizer ou fazer da perspectiva de uma pessoa pesquisadora, não necessariamente diferente da compreensão do senso comum na conversa do dia a dia.

5.1 Características do método de pesquisa qualitativa

Patton [71] menciona que existem muitos métodos de pesquisa qualitativa, ou ao menos, 16 perspectivas - como etnografia, etnometodologia, hermenêutica, análise narrativa, e outros. Alguns deram origem a vários métodos de pesquisa qualitativa. Duas razões para esta diversidade são: as raízes em diferentes disciplinas (como antropologia, filosofia,

sociologia ou psicologia) e a tendência de escolher (ou, historicamente, desenvolver) um método de pesquisa adequado ao tópico de pesquisa [36].

Os métodos compartilham várias características essenciais, apesar de sua variedade. Patton [71] e Flick et al. [36] oferecem listas semelhantes de 12 dessas características em seus livros didáticos. Zieris et al. [109] fornece uma visão geral consolidada com caraterísticas da pesquisa qualitativa em termos de pesquisa, perspectiva e tópicos, desenho geral e amostragem, e sobre a coleta e análise de dados, com base nos estudos de Patton [71] e Flick et al. [36], apresentada a seguir:

- **Investigação naturalística em situações cotidianas** - a pesquisa qualitativa, muitas vezes, foca em situações cotidianas e/ou conhecimento cotidiano ou senso comum. Consequentemente, os métodos qualitativos estudam situações do mundo real à medida que se desdobram naturalmente. Em contraste com os ambientes de laboratório, a pessoa pesquisadora não tenta controlar a situação, ali é um curso de ação predeterminado e as pessoas são observadas em um ambiente familiar para elas;

- **Projeto de Pesquisa Emergente** - um método de pesquisa qualitativa é escolhido com base no tópico de interesse, e não o contrário. Os métodos não prescrevem práticas de observação rígidas, mas exigem abertura na coleta de dados, análise e flexibilidade quanto ao desenho da pesquisa e também na compreensão em aprofundar o tema;

- **Amostragem proposital** - ao contrário dos métodos quantitativos, a generalização estatística de uma amostra para a população não é um objetivo. Em vez disso, os estudos qualitativos procuram casos ricos em informações que permitem aprofundar em detalhes a compreensão da pessoa pesquisadora. No início do processo, cada caso é tratado como único e estudado detalhadamente. A comparação das análises de cada caso, seguem posteriormente e são baseadas em casos individuais bem compreendidos;

- **Dados ricos em detalhes, diversos e densos** - os métodos qualitativos visam dados ricos em detalhes, típicos e complementares. As formas de coleta de dados são entrevistas em profundidade e abertas a observação direta;

- **A pessoa pesquisadora** - as pessoas pesquisadoras qualitativas se aproximam dos fenômenos de interesse para captar o máximo de detalhes possíveis. Em geral, as próprias ações e observações da pessoa pesquisadora no campo, seus próprios pensamentos e percepções, e suas reflexões sobre isso são importantes para os métodos qualitativos. A influência ou o viés da pessoa pesquisadora deve ser desconsiderado na coleta e análise dos dados;

- **Neutralidade** - como a realidade social é construída pelos seres humanos, ela precisa ser reconstruída pela pessoa investigadora. Para isso, as perspectivas dos sujeitos precisam ser compreendidas sem julgá-las.
- **A importância do contexto** - os dados devem ser coletados em seu "ambiente natural", conforme o contexto social, histórico e temporal dos indivíduos que atuam socialmente. É importante para interpretar e analisar suas ações;
- **Perspectiva abrangente** - os fenômenos sociais são complexos, e isso refere-se a possibilidade de serem totalmente compreendidos, separados e estudados em detalhes, considerando as relações simples de causa e efeito. Métodos qualitativos visam a compreensão abrangente dos fenômenos, incluindo seus relacionamentos complexos;
- **Criação de teoria** - é caracterizado pelo processo geral da pesquisa qualitativa, a "análise indutiva". É um processo ascendente que começa com observações concretas e detalhadas, a partir de quais temas "emergem", e servem como uma forma de hipótese a ser examinada à luz de novos dados coletados. O modo como essa emergência funciona, precisamente, depende do método de pesquisa específico. O procedimento comumente usados em métodos qualitativos é a abdução. Isso significa que quando a pessoa pesquisadora percebe uma combinação de características em seus dados interpretados, quando existem, não é uma explicação, e sim, é "inventada"uma teoria.

5.2 Grounded Theory

A *Grounded Theory* (GT) é uma geração sistemática de teoria a partir de dados analisados por um método de pesquisa rigoroso [40]. A GT foi desenvolvida pelos sociólogos Glaser e Strauss [40] como resultado de sua pesquisa colaborativa sobre hospitais com pacientes em estados terminais. Eles publicaram o "*The Discovery of Grounded Theory*"[40], estabelecendo o começo da *Grounded Theory*. Glaser define a GT, como:

> "A abordagem da teoria fundamentada em dados é uma metodologia geral de análise relacionada com coleta de dados, usando um conjunto de métodos aplicados sistematicamente para gerar uma teoria indutiva sobre uma área substantiva". [41]

O objetivo da GT é gerar uma teoria como um conjunto, relacionando as hipóteses, por meio de comparação constante de dados em níveis crescentes de abstração [41]. Ao gerar uma teoria, a pessoa pesquisadora da GT descobre a principal preocupação dos participantes da pesquisa e como eles procedem para resolvê-la. A diferença característica do método GT é a ausência de um problema de pesquisa ou hipótese, inicial, em vez disso,

a pesquisadora tenta descobrir o problema de pesquisa como a principal preocupação dos participantes do processo [3, 69].

As diferenças entre os dois criadores da *Grounded Theory* (Teoria fundamentada nos dados) resultaram no surgimento de duas versões do método da teoria fundamentada nos dados: a versão de Glaser da GT, frequentemente referido como GT clássico e a versão de Strauss, denominada GT Straussian [42]. Nesta pesquisa, optou-se por escolher as técnicas de análise de dados da Teoria fundamentada nos dados Clássica, principalmente pelo maior número de recursos disponíveis [42].

Para este estudo, após um estudo terciário de revisão sistemática sobre ensino de Engenharia de Software, optou-se por escolher as entrevistas semiestruturadas para coletar dados e o uso de técnicas de análise de dados da GT, como o método de pesquisa por vários motivos. Em primeiro lugar, o ensino de métodos ágeis tem foco em pessoas e interações e a GT, usada como método de pesquisa qualitativa, permite estudar as interações sociais e o comportamento [69]. Em segundo lugar, a GT é mais adequada para áreas de pesquisa que não foram exploradas em grande detalhe anteriormente, e a pesquisa sobre a ensino + pandemia do coronavírus é limitada. Em terceiro lugar, a GT é um dos poucos métodos de pesquisa que concentrada na geração de teoria, ao invés de estender ou verificar teorias já existentes. E finalmente a *Grounded Theory* permite gerar uma teoria nova, e relacionar Engenharia de Software, diversidade social e tempos de pandemia.

5.2.1 *Grounded Theory* em Engenharia de Software

O estudo de Hoda et al. [46] menciona que pessoas pesquisadoras exploraram vários aspectos da Engenharia de Software usando *Grounded Theory* como método de pesquisa. Por exemplo, Coleman e O'Connor [24] conduziram uma pesquisa da Teoria Fundamentada nos Dados sobre o uso de melhoria de processos de software na indústria de software irlandesa. Eles coletaram dados em fases através de entrevistas com gerentes seniores em 21 empresas de desenvolvimento de software. As principais descobertas do estudo sugerem que o custo do processo é a principal motivação por trás do software empresas que não usam modelos de "melhores práticas"e que as empresas adaptam o padrão de processos de software para seus próprios contextos. O maior custo de processo percebido pelas organizações participantes foi o custo da documentação.

Hoda et al. [46] menciona o trabalho de Dagenais et al. [26]. Eles usaram a Teoria fundamentada nos dados para estudar as características relevantes do projeto, os obstáculos de familiarização, enfrentados pelos novos membros da equipe, e o apoio que podem auxiliar os novos membros a se moverem para novos projetos. O estudo foi baseado em dados qualitativos coletados em 18 participantes localizados em 8 países, a maioria dos participantes eram desenvolvedores com menos de um ano de experiência. O estudo

concluiu que os recém-chegados muitas vezes enfrentam projetos desconhecidos e com designs hostis, enquanto os membros mais experientes ajudam os mais novos na ambientação. A presença de um guia / mentor, em comparação com os guias documentais, foi considerada muito útil.

A *Grounded Theory* está sendo cada vez mais usada para estudar os aspectos humanos e sociais em equipes de desenvolvimento ágil de software. A natureza social das equipes ágeis foi explorada por meio de um estudo de *Grounded Theory* de Whitworth et al. [105]. Os resultados destacam a importância das práticas sociais e de interação, como reuniões diárias, e o uso de meios de informação e comunicação nas organizações, para tratar a responsabilidade social e a conscientização. Os resultados enfatizam a importância da auto-organização e das habilidades das equipes ágeis, embora tenha sido identificado um número muito pequeno de pesquisas sobre o tema.

Martin et al. [57] usaram a Teoria fundamentada nos dados para estudar o papel da equipe local, com clientes em projetos de *Extreme Programming* (XP). A pesquisa foi realizada utilizando como método de pesquisa a *Grounded Theory* clássica por meio de entrevistas semi estruturadas informais com equipes XP em diferentes organizações. A pesquisa descobriu que, embora os clientes tenham percebido como é gratificante estar presente, também foi visto como uma sobrecarga inerentemente e insustentável. Eles concluíram que o cliente no local em projetos XP precisa ser desempenhado por uma equipe de pessoas, e não só por uma pessoa como inicialmente assumido na literatura. Esta equipe informal de clientes XP consiste em papéis diferentes. Dessas diferentes funções, a função do analista de negócios é a mais próxima da função clássica do representante do cliente, interagindo diretamente com o desenvolvimento da equipe. Eles também descrevem certas práticas do cliente como *Customer Boot Camp* e a personalização de pares. Essas práticas, assim identificadas, podem ser combinadas com as funções do cliente, e isso pode ajudar a reduzir a carga colocada na função do cliente no local da equipe XP.

Com os estudos mencionados anteriormente, acredita-se que a as técnicas de análise de dados da teoria fundamentada nos dados são adequadas para explorar como a Engenharia de Software é ensinada por pessoas para as pessoas. Este estudo, concentra-se na aplicação de técnicas de análise de dados da Teoria Fundamentada nos Dados para identificar como a Engenharia de Software é ensinada online para pessoas em situação de vulnerabilidade social. A Figura 5.1 apresenta o fluxo e ordem das atividades por meio da coleta de dados em entrevistas semiestruturadas e a utilização de técnicas para analisar os de *Grounded Theory*, com base no estudo de Hoda et al. [46]. As atividades em detalhes - como os dados foram coletados e analisados (as atividades na cor cinza representam as técnicas de análise de GT) pelos autores deste estudo, são apresentadas nas Seções a seguir.

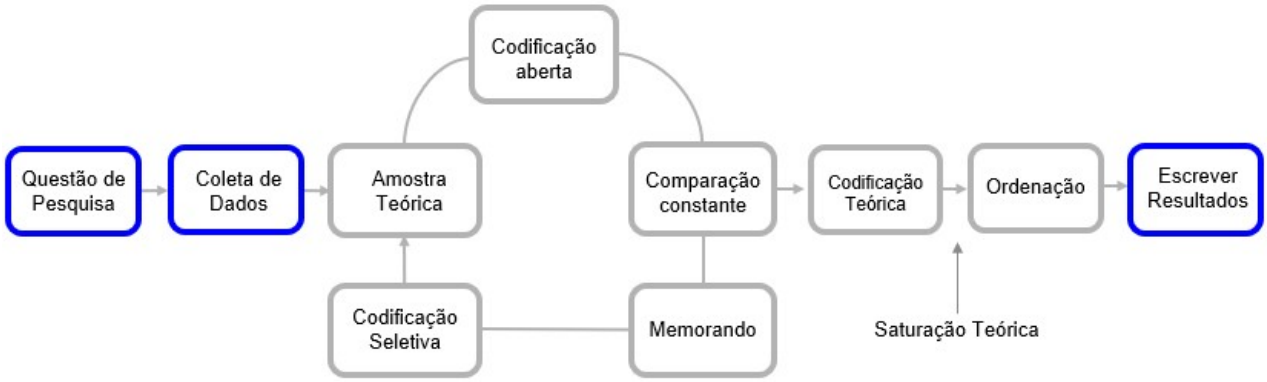

Figura 5.1 – Atividades para coleta de dados e técnicas de análise de dados - *Grounded Theory*

Fonte: adaptado de [46].

5.3 Área de Interesse e a Questão de Pesquisa

Glaser [40] afirma que para descobrir e estudar os principais problemas dos participantes, recomenda-se que a pessoa pesquisadora se abstenha de formular um problema ou pergunta de pesquisa com antecedência. O problema de pesquisa deve ser o problema dos participantes do estudo e não deve ser preconcebido ou forçado, mas deve ser permitido surgir [40].

Embora a pessoa pesquisadora não seja aconselhada a formular uma questão de pesquisa, ela deverá escolher uma área de interesse geral. O primeiro passo da pessoa autora deste estudo, foi realizar um mapeamento sistemático na literatura (MSL) - os resultados foram apresentados no capítulo 4, para entender o que a Engenharia de Software estava estudando sobre o assunto Diversidade. Além disso, foi realizado um estudo terciário de revisão sistemática sobre o ensino de Engenharia de Software. O estudo sobre ensino de Engenharia de Software possibilitou a construção do protocolo de pesquisa para a pesquisa qualitativa com coleta de dados em entrevistas semiestruturadas e técnicas de análise dos dados de *Grounded Theory*. Esse protocolo é apresentado no Apêndice C.

De acordo com os resultados obtidos dos mapeamentos sistemáticos da literatura sobre diversidade em Engenharia de Software e ensino de Engenharia de Software, a autora deste estudo desenvolveu um fluxograma para o entendimento da área de interesse para formular a questão de pesquisa, conforme apresentado na Figura 5.2.

Conforme apresentado na Figura 5.2 a questão de pesquisa (QP) formulada para coletar e analisar dados de pessoas mentoras / professoras e pessoas alunas foi a seguinte:

QP. Como a Engenharia de Software é ensinada em tempos de pandemia para pessoas em vulnerabilidade social?

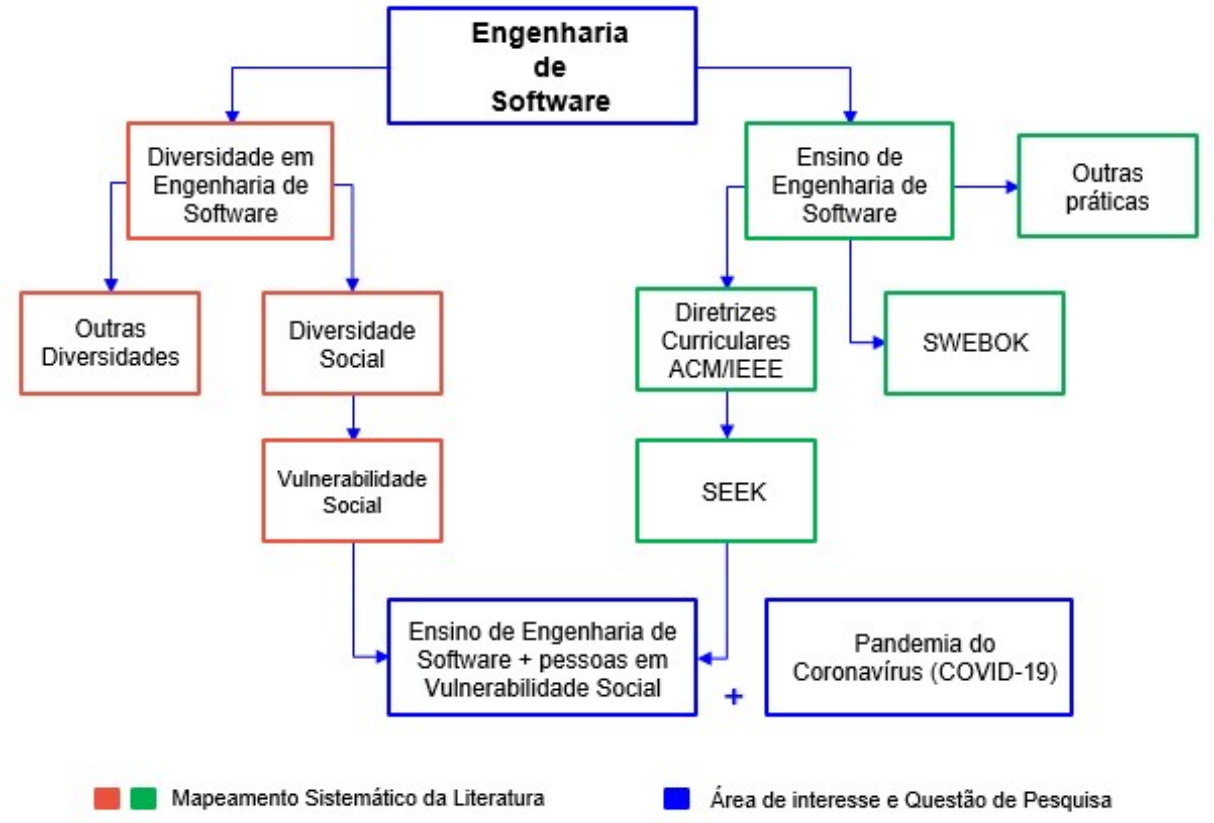

Figura 5.2 – Esquema gerado para a área de interesse
Fonte: a autora (2020)

5.4 Coleta de Dados

A técnica selecionada para a coleta de dados foi o da entrevista semiestruturada, que permite, ao mesmo tempo, a liberdade de expressão da pessoa entrevistada e manutenção do foco pela pessoa entrevistadora [11]. Ela é executada por um roteiro de perguntas previamente estabelecidas (Apêndice D).

Nas próximas subseções serão apresentados a seleção de participantes dos estudos e o planejamento para a realização das entrevistas.

5.4.1 Seleção de Participantes

Foram realizadas 24 entrevistas entre pessoas mentoras / professoras e alunas de um ambiente de ensino de engenharia de software. Esses cursos de capacitação ocorreram de forma online, no ano de 2020, devido à pandemia da COVID-19. A duração das capacitações dura em média 18 semanas. As pessoas mentoras / professoras e alunas foram entrevistadas no período de novembro a dezembro de 2020.

Para todos os casos, o participante iniciou sua participação no estudo somente após ser informado e concordado com as condições da atividade conforme declarado no Termo de Consentimento Livre e Esclarecido (TCLE) - pode ser consultado no Apêndice E, o qual foi discutido com o participante antes da atividade de coleta de dados (entrevistas) iniciarem. O participante ficou livre para se retirar da atividade a qualquer momento, sem haver necessidade de explicitar razões para isso.

5.4.2 Entrevistas

As entrevistas semiestruturadas foram realizadas com duração mínima de 25 minutos e no máximo de 1h, por meio da Plataforma do *Google Meet*[1]. Foram entrevistados, pessoas mentoras / professoras e alunas que participam do curso de capacitação desde a forma presencial, antes da Pandemia do Coronavírus (COVID-19) (ver capítulo 1, seção 1.2.2, Figura 1.2) e durante a pandemia. E pessoas mentoras / professoras e alunas que participaram somente do projeto durante a pandemia (ver capítulo 1, seção 1.2.2, Figura 1.3). O foco foi as experiências vivenciadas e práticas de ensino de Engenharia de Software, de acordo com conhecimento de ensino de Engenharia de Software (SEEK) [14], durante o curso de capacitação em Engenharia de Software, em particular, durante a pandemia do Coronavírus (COVID-19). A Figura 5.3 apresenta os dados coletados nas entrevistas das pessoas participantes.

[1] Disponível em: https://meet.google.com/

Nº	ID	Empresa	Função	Semestres	Área	Tempo médio	Obs.	Vulne. Social	Infra.	Rodada
1	R1	E1	Coach	18	Scrum Master	50min		Não		
2	R11	E2	Mentora	2	ES (Full Stack)	30 min		Não		
3	R7	E3	Mentora	2	Designer de Produto, UX	25 min		Não		1
4	R5	E2	Mentora	6	ES (Full Stack)	20 min	Ex estudante	Não		
5	R2	E1	Mentora	2	Psicóloga	60 min		Não		
6	R10	E3	Mentora	2	ES (Front-end)	30 min	Ex estudante	Não		
7	R8	E2	Mentora	2	ES (Full Stack)	30 min	Ex estudante	Não		
8	R3	E1	Ass. de Ensino	2	Designer de Produto, UX	25 min	Ex estudante	Sim	SIM	2
9	R27	E1	Estudante	2	ES (Front-end)	30 min		Não		
10	R26	E1	Estudante	2	ES (Front-end)	30 min		Não		
11	R24	E1	Estudante	2	ES (Full Stack)	40 min		Não		
12	R22	E1	Estudante	2	ES (Full Stack), QA	40 min		Não		
13	R9	E3	Mentora	2	ES (Full Stack)	30 min	Ex estudante	Não		
14	R4	E1	Mentora	1	ES (Front-end)	35 min	Ex estudante	Não		
15	R20	E1	Estudante	2	ES (Back-end)	25 min	Ex estudante	Sim	SIM	
16	R19	E1	Estudante	3	ES (Full Stack)	40 min	Ex estudante	Sim	SIM	
17	R29	E1	Estudante	2	ES (Front-end)	30 min		Não		3
18	R23	E1	Estudante	2	ES (Front-end)	40 min		Não		
19	R18	E1	Estudante	4	ES (Full Stack)	30 min	Ex estudante	Sim	SIM	
20	R28	E1	Estudante	2	ES (Back-end)	45min		Não		
21	R21	E1	Estudante	2	ES (Front-end)	45min	Ex estudante	Sim	SIM	
22	R25	E1	Estudante	2	ES (Back-end)	25min		Sim	SIM	
23	R12	E2	Mentora	2	QA	30 min		Não		
24	R13	E2	Mentora	2	QA	30 min		Não		

Figura 5.3 – Visão geral das pessoas participantes das entrevistas semiestruturadas
Fonte: a autora (2020)

A Figura 5.3 apresenta as pessoas participantes das entrevistas semiestruturadas e identificadas por um identificador (ID). As empresas participantes foram identificadas pelas siglas E1, E2 e E3. As funções de cada pessoa participante foram especificadas de acordo com o estudo já publicado pelas pessoas autoras desse trabalho [95], relacionadas a seguir:

- ***Coach***: responsável pela pré-avaliação das pessoas estudantes, bem como conduzir o planejamento, revisão e retrospectiva de cada iteração. Esta função também apoia o assistente de ensino e realiza a avaliação final do projeto;
- **Mentora**: é responsável por apoiar a equipe de alunas durante todo o projeto, ensino e introdução de novas práticas de desenvolvimento, e nas instruções dos pares (programação em pares);
- **Assistente de ensino**: é uma pessoa estudante que já participou de turmas anteriores, e tem a responsabilidade de ajudar as pessoas estudantes a preparar o ambiente de desenvolvimento, resolvendo impedimentos, auxiliando a equipe nas dúvidas técnicas e colaborando na atividades, com apoio do *Coach*;
- **Estudante**: são pessoas estudantes responsáveis pelo desenvolvimento do projeto, incluindo requisitos, interface de usuário, arquitetura, atividades de codificação e garantia de qualidade para se comunicar com o Proprietário do produto (POs)[2].

[2]De acordo com Steglich et al. [95], POs, são os responsáveis por listar os requisitos, priorizando-os e avaliando os requisitos entregues ao final de cada iteração.

Na coluna "área", foram listadas as áreas que cada pessoa mentora e estudantes são vinculadas. As siglas serão apresentadas a seguir:

- ES: pessoa Engenharia de software;
- UX: pessoa Designer de Experiência do Usuário;
- QA: Pessoa analista de qualidade.

A coluna observações destina-se para explicar as observações identificadas nas entrevistas em relação às pessoas participantes. A informação "Ex estudante"refere-se aos participantes que foram estudantes do curso de capacitação em Engenharia de Software, e agora são profissionais da indústria de desenvolvimento de software - e atuam como pessoas mentoras.

O protocolo foi iterativo e incremental, ou seja, foi ajustado de acordo com as necessidades identificadas na análise de cada dado coletado. O protocolo de pesquisa e roteiro de entrevistas (Apêndice D) foi ajustado 3 vezes. A primeira rodada foi realizada, com 5 pessoas mentoras. A segunda rodada foi realizada com 2 pessoas mentoras, 1 assistente de ensino e 2 estudantes. A terceira e última rodada foi realizada com 14 pessoas, em que, 10 pessoas eram estudantes, 4 pessoas mentoras. Na rodada 2 e 3, um total 6 estudantes declaram-se pessoas em situação de vulnerabilidade social[3]. Esses 6 estudantes necessitam de apoio / doação para aquisição de equipamentos como computadores, periféricos de entrada, saída e conexão com internet para realizar o curso de engenharia de software.

5.5 Análise dos dados

Nesta fase do estudo qualitativo, foram utilizadas as técnicas de análise de dados da *Grounded Theory* nos dados obtidos nas entrevistas. As subseções a seguir apresentam os mecanismos de codificação, como código aberto, codificação seletiva, teórica, método de comparação constante e memorandos.

5.5.1 Codificação Aberta

A codificação aberta é o primeiro passo da análise de dados em uma *Grounded Theory*. Corbin e Strauss [25] conceituam a codificação aberta como o processo analítico

[3]Segundo Raoport et al. [84], Vulnerabilidade Social é caracterizado pela condição de grupos de pessoas que vivem em condições desfavoráveis como: condições precárias de moradia, ausência de um ambiente familiar e/ou educacional de qualidade.

pelos quais os conceitos são identificados e desenvolvidos em relação às suas propriedades e suas dimensões. Esse processo envolve as atividades de quebrar, examinar, comparar, conceituar e categorizar os dados que serão sumarizados em uma lista de códigos e categorias originadas dos rótulos atribuídos livremente a cada frase, linha ou parágrafo.

À medida que novos casos são estudados em detalhes, alguns deles serão semelhantes aos já analisados (se enquadram no mesmo conceito), mas ao mesmo tempo são diferentes de alguma forma. A cada captura dessas variações, os conceitos são continuamente revisados, por exemplo, pela introdução de novas propriedades, o que, por sua vez, exige que a pessoa pesquisadora revisite exemplares já analisados, para ver se o conceito alterado ainda se ajusta a ele ou qual é seu valor específico em relação a uma propriedade recém-introduzida [25].

5.5.2 Comparação constante

Corbin e Strauss [25] mencionam que os segmentos particulares dos dados não são analisados e codificados uma vez, mas são comparados repetidamente com novos dados. Isso é feito no nível de conceito, procurando sempre diferenças e semelhanças relevantes. Os códigos decorrentes de cada entrevista foram constantemente comparados com os códigos da mesma entrevista, e de outras entrevistas. Isto é Método de comparação constante de *Grounded Theory* [40, 41], onde foram agrupados para produzir um nível mais alto de abstração, chamando de conceito em teoria fundamentada nos dados (*Grounded Theory*).

5.5.3 Memorando

Glaser [40] classifica como um processo contínuo de escrever memorandos teóricos em toda a teoria fundamentada em dados. Memorandos são notas teóricas sobre os dados e as conexões conceituais entre as categorias escritas à medida que atingem a pessoa pesquisadora. São considerados como o estágio central ou a base da geração da teoria. Memorandos podem ser ideias fluentes da pessoa pesquisadora sobre os códigos e seus relacionamentos e não devem ser misturados com dados. Os memorandos são escritos a partir das ideias sobre os códigos emergentes e seus relacionamentos. Como recomendado por Glaser [40], muitas vezes a codificação e outras atividades para capturar as ideias por meio do memorando, eram interrompidas, à medida que chegavam até as pesquisadoras.

5.5.4 Codificação Seletiva

Uma vez que a categoria principal é estabelecida, a pesquisadora pausa a codificação aberta e passa para a codificação seletiva, um processo que envolve a codificação seletiva para a variável central, delimitando a codificação para apenas as variáveis que se relacionam com a variável central de maneiras suficientemente significativas para produzir uma teoria parcimoniosa [40]. A categoria principal orienta a coleta de dados, análise e amostragem teórica [40]. Quanto mais coletar e analisar dados, em uma categoria particular leva a um ponto de resultados decrescentes, nesse caso, a categoria teria atingido a Saturação Teórica [41]. O pesquisador pode parar de coletar dados e codificar para aquela categoria.

5.5.5 Codificação Teórica

Nesta etapa ocorre a fase final da codificação [41], conhecida também como Codificação Teórica (códigos teóricos). A codificação teórica é definida como a propriedade da codificação e análise comparativa constante. Ela é responsável pela produção da relação conceitual entre as categorias e suas propriedades a cada surgimento [41].

Glaser lista várias estruturas comuns de teorias conhecidas como grupos teóricos [41, 42]. Alguns deles incluem: Os seis C's (causas, contextos, contingências, consequências, covariâncias e condições); Processo (etapas, fases, passagens etc.); Graus por grupo (limite, alcance, intensidade etc.); Grupo por dimensão (dimensões, elementos, divisões etc.) Tipos de Grupo (tipo, forma, crianças, estilos, classes, gênero) e muito mais.

De acordo com as recomendações de Glaser [41], a codificação teórica foi usada na última fase de análise dos dados. Esta codificação foi a solução adequada para descrever a descoberta de práticas de ensino de Engenharia de Software.

O desafio da pessoa pesquisadora de Engenharia de Software é se acostumar a pensar de forma teórica - os resultados anteriores, não podem tornar isso visível. A fim de superar este problema, deve-se considerar que os códigos teóricos são baseados na classificação de memorandos e não em dados [42]. A possibilidade de deixar em aberto os conceitos e categorias emergentes - ler sobre diferentes teóricos códigos, tornam as pessoas pesquisadoras sensíveis ao ver os códigos teóricos.

5.5.6 Ordenação

Quando a pessoa pesquisadora estava quase terminando a coleta de dados com todos os entrevistados, iniciou-se o processo de classificação conceitualmente dos memorandos teóricos. A classificação dos memorandos constitui um esboço teórico. A classificação é uma etapa essencial e não pode ser desconsiderada [40]. A vantagem desta classificação é a organização e união dos dados fragmentados, novamente [40]. A pesquisadora deve ter em mente que os dados não precisam ser classificados, e sim as ideias. Deve-se evitar a classificação dos memorandos em ordem cronológica. A classificação deve ser feita em um nível conceitual, resultando em um esboço da teoria - como as diferentes categorias relacionam-se com a categoria principal.

Foram coletados os detalhes de todos os memorandos. Foram classificados por tópicos para que eles fossem ordenados um após o outro. Logo, foi gerado um esboço da teoria usando os nomes dos tópicos na mesma ordem. Este esboço serviu como base para a proposta deste estudo.

O maior desafio envolvido na classificação dos memorandos é que, embora seja muito mais fácil agrupar memorandos relacionados, a ordem dos memorandos talvez não seja nitidamente óbvia. Foi necessário embaralhar alguns memorandos e pensar nos relacionamentos entre os diferentes tópicos de memorando para encontrar uma ordem que fizesse mais sentido. As pessoas autoras deste estudo resolveram traçar as relações entre as diferentes categorias de forma manual. Uma vez que as relações foram estabelecidas em um diagrama (usando linhas para conectar as categorias), foi mais fácil identificar como os memorandos (cobrindo diferentes categorias e conceitos) foram relacionados.

5.5.7 Escrever os Resultados

A etapa final da pesquisa qualitativa é a redação dos resultados, que segue o esboço teórico gerado como resultado de classificação e codificação teórica. Os resultados deste trabalho giram em torno do ensino de Engenharia de software online para pessoas em situação de vulnerabilidade social.

5.6 Resultados

Esta seção apresenta o resultado da coleta de dados por meio de entrevistas semiestruturadas, analisadas pelo processo de codificação da teoria fundamentada nos dados. O processo de entrevista semiestruturada permitiu a compreensão da realidade investigada,

possibilitando a identificação dos pontos essenciais e de como eles devem ser introduzidos para ensinar Engenharia de Software.

A seguir serão apresentados os resultados das entrevistas semiestruturadas. A seção 5.6.1 apresenta os resultados consolidados das respostas e perfis dos participantes das entrevistas. A seção 5.6.2 apresenta os resultados consolidados sobre as práticas de ensino de Engenharia de Software durante a pandemia.

5.6.1 Capacitação em Engenharia de Software

Os dados coletados das entrevistas semiestruturadas foram categorizados e analisados por meio do processo de codificação dos dados da Teoria fundamentada nos dados. O *Microsoft Excel*[4] foi a plataforma utilizada para auxiliar no processo de codificação dos dados coletados, neste estudo.

A Tabela 5.1 a seguir apresenta a pergunta realizada pela pessoa entrevistadora e as respectivas respostas dos respondentes identificados com "ID".

Tabela 5.1 – Referências das entrevistas sobre a capacitação em ES

Como você define o curso de capacitação em Engenharia de Software que você é mentora/estudante?	
ID	Respostas dos entrevistados
R1	*"Hoje o foco principal do curso é a inclusão e capacitação de alunas no mercado de trabalho. Possibilitar um ambiente de aprendizado em que essa estudante possa se engajar e motivar a buscar mais o aprendizado pela experiência positiva"*
R11	*"Selecionar e dar capacidade para pessoas da Diversidade"*
R2	*"Porta de entrada, um projeto mais avançado na parte técnica, uma noção básica de lógica de programação. Conceitos do contexto de métodos ágeis, pair programming, consiste em trabalhar com práticas de métodos ágeis"*
R5	*"Formar pessoas iniciantes na área para o mercado de trabalho com as vivencias da indústria de desenvolvimento de software"*
R10	*"O foco é o desenvolvimento das estudantes em conhecimento de mercado, com prática profissional, preparando para um mercado real, não somente na teoria"*

As categorias são resultadas do processo de Codificação aberta dos resultados obtidos da Tabela 5.1. Cada entrevista foi codificada e comparada com os dados coletados. As categorias "Ambiente de Aprendizado, Inclusão e Vivenciar a indústria de desenvolvimento de software"são resultados da coleta de dados. A Tabela 5.2 apresenta o conceito das categorias e condições causais, fenômeno, estratégias, condições intervenientes e consequências de cada categoria identificada no processo de codificação dos dados.

[4]O Microsoft Excel é um editor de planilhas produzido pela Microsoft para computadores que utilizam o sistema operacional Microsoft Windows. Fonte: https://www.microsoft.com/pt-br/microsoft-365/excel

Tabela 5.2 – Categorias geradas das entrevistas

Categorias	**Conceitos**	**Fenômeno, estratégias, condições causais, condições intervenientes e consequências**
Ambiente de aprendizado	Interação síncronas, espaço para aprender, oportunidade de errar e acertar	que proporciona ensinar e aprender com teoria e prática vivenciadas na indústria
Capacitação	capacitar profissionais no início da carreira, ensino prático	É a contraposição de uma abordagem conteudista de cursos normais, para pessoas em vulnerabilidade social com possibilidade de inserção no mercado de trabalho
Diversidade	Seleção de pessoas	de pessoas possibilitando um ambiente inclusivo onde é permitido errar e acertar como aprendizado
Inclusão	Oportunidade para os menos favorecidos	de diversidade de gênero, social, LGBTQIA+ e étnico /racial para ter espaço na TI
Vivenciar a Indústria de Software	Ambiente de engajamento e motivação	de forma objetiva e prática com exemplos reais de indústria
Prática de Métodos Ágeis	Ensino colaborativo e com exemplos reais	de forma iterativa, sendo ajustada de acordo com a necessidade da turma
Prática Profissional	Relação teoria e prática	e pessoal, vivenciando exemplos reais de projeto, com apoio de pessoas da indústria

5.6.1.1 Caracterização dos respondentes

Foram coletadas 24 entrevistas semiestruturadas entre pessoas mentoras e estudantes (11 pessoas mentoras e 13 estudantes). A média de idade entre as pessoas mentoras participantes é de 27 anos e estudantes, 25 anos. A seguir serão descritas as perguntas realizadas em cada entrevista (Obs.: as perguntas eram realizadas pela pesquisadora e ao mesmo tempo informado que as respostas não eram obrigatórias se existisse algum desconforto em respondê-las).

Qual seu nível de escolaridade? Um total de 2 pessoas mentoras responderam seu nível de escolaridade como pós-graduação (Lato Sensu) completo; já 4 pessoas mentoras responderam, superior completo e 5, superior incompleto. O número de 5 estudantes respondeu como ensino médio completo e 8 como superior incompleto.

Qual gênero[5] **você se identifica?** Na entrevista, 6 pessoas mentoras responderam como seu gênero: feminino cisgênero e masculino cisgênero, um total de 5 respondentes. 10 estudantes responderam feminino cisgênero e 3 masculinos cisgênero.

Em relação a sua cor[6]**, como você autodeclara-se?** No total de 13 mentoras, 8 responderam sua cor como branca; 3 responderam como preta. Onze estudantes declararam serem brancas e duas, pretas.

Sobre seu status socioeconômico, como você se classifica sua Classe Social?[7]

Oito mentoras classificaram sua classe social como "C", duas como "B"é uma pessoa como "A". Sete estudantes classificaram como "D"e seis estudantes como "E".

5.6.1.2 Contexto da pandemia do coronavírus (COVID-19)

Durante o ano de 2020 o curso de capacitação em Engenharia de Software foi realizado de forma online, devido à pandemia do coronavírus (COVID-19). Diante do contexto da pandemia, a Tabela 5.3 apresenta a pergunta realizada pela entrevistadora e as respectivas respostas de cada respondente, identificados de acordo com o "ID".

Tabela 5.3 – Referências das entrevistas sobre contexto pandemia

Como foi sua experiência no curso durante a pandemia?	
ID	Respostas dos entrevistados
R10	*"Falta contato físico, as vezes com câmera desligada e não consegue identificar a reação da pessoa. No presencial tu consegues identificar o sentimento deles. E no remoto não"*
R8	*"Aos poucos foi melhorando a comunicação. A questão do aprendizado técnico, é diferente de aluna para a aluna, de turma para a turma, ou até mesmo as tecnologias. Presencialmente teria um DOJO na turma, com todo mundo. Hoje tem a possibilidade de fazer o Live Coding"*
R20	*"No início, foi um desafio (nunca tinha estudado e nem trabalhado de forma remota), entender de como vai funcionar as coisas, o ritmo das coisas. Depois aos poucos foi se adaptando, dá para aprender bem e fazer as coisas bem"*
R2	*"Está sendo bem desafiador, bastante frustrante para as mentoras, não ter mais espaço físico onde se encontra possibilidades e alternativas"*
R7	*"Eu considero como um desafio para dinâmica em grupos, tem ferramentas que precisam ser utilizadas, mas são pagas. Contato através da câmera (desligada), sem saber a reação deles. Presencialmente é mais próxima"*

A dificuldade sem o contato humano para compartilhar conhecimento foi mencionado pelo respondente R7, pois as *Webcams* desligadas, ou a falta de interação das es-

[5] Gênero refere-se a formas de se identificar e ser identificada como homem ou como mulher [28]

[6] Autodeclaração de cor de acordo com o IBGE. Disponível em: https://educa.ibge.gov.br/jovens/conheca-o-brasil/populacao/18319-cor-ou-raca.html

[7] Tabela de classes sociais no Brasil de acordo com o IBGE, disponível em: https://www.ibge.gov.br/estatisticas/sociais/populacao/9221-sintese-de-indicadores-sociais

tudantes, dificultou a identificação das dificuldades ou facilidades das pessoas estudantes nas atividades de ensino. A Tabela 5.4 apresenta os códigos gerados da Tabela 5.3 sobre o contexto da pandemia durante o andamento do curso de capacitação em Engenharia de Software.

Tabela 5.4 – Contexto da pandemia do coronavírus

Categorias	**Conceitos**	**Fenômeno, estratégias, condições causais, condições intervenientes e consequências**
Aplicar Dinâmica em Grupo	integralização, quebra gelo, criatividade	para integralizar as pessoas que nunca tiveram contato presencial. Outras dinâmicas foram aplicadas para ensinar cerimônias ágeis
Aprendizado Técnico	exemplos reais, práticas de indústria de desenvolvimento de software	adquirir aprendizado técnico para me tornar um profissional de qualidade
Comunicação	meios de comunicação, dinâmicas das duplas, team building	facilitar a comunicação entre o time de alunas
Desafio	sem contato físico, reação, interação	de executar um projeto real durante a pandemia e 100% online, sem conseguir acompanhar de forma mais próxima o rendimento de cada aluna
Interação		dificuldade de interação com alunos que não ligavam suas câmeras durante as sessões
Live Coding	programação ao vivo, compartilhamento do problema e solução	ensinar a programar através do Live Coding para alunas sem contato presencial

A categoria *Live Coding* gerada dos resultados da entrevista com a respondente R8 surgiu como uma inovação durante as aulas online do curso de capacitação. Essa abordagem ensina a programar ao vivo, com compartilhamento de tela, construindo uma solução em conjunto com a turma de estudantes e pessoas mentoras. Isso possibilitou ensinar programação e exercitar a lógica de programação, por meio de linguagens de desenvolvimento para as pessoas estudantes do curso. Aplicar as dinâmicas em grupo foi um desafio mencionado pelas pessoas mentoras R2 e R7, uma vez que, muitas atividades de

quebra gelo são necessárias para o contato físico. Foram necessárias realizar uma busca com dinâmicas criativas que podem ser realizadas de forma online ou adaptadas para o formato.

A Tabela 5.5 apresenta os resultados das entrevistas.

Tabela 5.5 – Referências das entrevistas sobre facilidades na pandemia

Como foi sua experiência no curso durante a pandemia? Facilidades	
ID	Respostas dos entrevistados
R20	*"Questão de transporte público, e da locomoção da PUCRS até a escola que ele estuda noite. Economia de dinheiro em passagem e tempo a se locomover"*
R17	*"Mais perto da filha, deslocamento para o trabalho. Gerenciamento de família e tempo para se organizar"*
R28	*"Deslocamento até o local do curso"*
R19	*"Conversar com o mentor técnico de outro estado. Cultura experiências diferentes"*
R4	*"Questões de horário, as reuniões podem ser uma atrás da outra, sem precisar se locomover de um lugar para outro. Sessões com pessoas mentoras de fora da cidade"*
R29	*"Não encontrou"*
R21	*"Confortável em estar estudando e comendo lanche na frente ao computador, eu fiz amizades remotas. Eu tive a oportunidade de ser eu mesma. Não existia coisas de aparência, sem julgar a sexualidade da pessoa, as vezes com foto sem estar com vídeo, ao vivo. Aí perdia a vergonha de perguntar. Voltar várias vezes do assunto. O fato é descobrir que é quase a mesma coisa que pessoalmente"*
R25	*"Ambiente acolhedor, incentivo dos colegas e pessoas mentoras, poder ter essa chance de aprender, saber que é um ambiente virtual de aprendizado, trabalhar com um projeto real e possibilidade de comunicação assíncrona. Esforço da equipe para o projeto dar certo"*
R1	*"A questão do ambiente virtual, Canvas colaborativo, miro ajudou a tentar aproximar o que é uma experiência física, na Inception, apresentação para as pessoas"*
R11	*"É muito mais fácil de entrar em uma sala para ajudar as alunas do que ir presencialmente (comportamento entre os pares). É uma facilidade"*

As categorias geradas para as Facilidades de acordo com a Tabela 5.5 encontradas no ensino de Engenharia de Software online durante a pandemia:

- Os entrevistados R25 e R1 mencionaram o **Ambiente virtual** (para compartilhar conhecimento, pedir ajuda): possibilitar um ambiente virtual de aprendizado para pessoas diferentes;
- **Apoio de mentoras de outros lugares** (profissionais da indústria de outras cidades e estados): ter mentoria de profissionais das empresas parceiras que moram em outros estados e cidades foi mencionado pelos entrevistados R19 e R4;

- A **Autonomia** foi mencionada por R17 (procurar alternativas de aprendizado fora do curso): na organização das tarefas do curso, possibilitando autonomia na procura de soluções com mentoras ou na internet;
- **Canvas Colaborativo**[8] (apoio de mentoras, POs e alunas para manusear uma ferramenta para o projeto): um método visual para planejar o projeto real, de acordo com R1;
- **Comodidade** (ficar em casa, sem precisar sair na rua): no conforto da casa foi descrito pelos respondentes R20, R17 e R28;
- **Comportamento entre os pares** (acompanhar entrando em cada sala virtual): em salas individuais através das plataformas de videoconferência, R11;
- **Comunicação assíncrona** (tudo organizado e disponível no Google Drive[9], Trello[10], Google Calendar[11]): possibilitando acesso aos conteúdos abordados em sessões, fora de horário foi mencionado por R25;
- R20, R17 e R28 citam o **Deslocamento** (sem precisar sair de casa para o deslocamento, economia de transporte): de casa para o local onde funcionava a curso de forma presencial;

As dificuldades encontradas durante o percurso no curso de capacitação de Engenharia de Software, durante a pandemia foi respondido pelos participantes das entrevistas. A Tabela 5.6 apresenta os resultados das dificuldades encontradas:

- O respondente R22 descreveu a **Autonomia** (familiares não entendem que ela está em horário de aprendizado. Atividades de casa, como limpeza e alimentação): devido à timidez para pedir ajuda. Dificuldades de organização para determinadas tarefas;
- **Concentração** (em atividades de aprendizado ou compartilhamento de conhecimento): a família não entendia que ela estava em horário de trabalho. Chamava para outras atividades ou fazia barulho sempre foram mencionadas pelos entrevistados R24, R19 e R29;
- **Contato Pessoal** (a falta de contato pessoal): dificuldades de interagir de forma online com os colegas, de acordo com R24, R22 e R9;
- **Dificuldades de conexão** (com a internet): a região tinha queda de internet constante, ou temporais foi apresentada por R24;

[8]Canva. Disponível em: https://www.canva.com/pt$_br$/
[9]Google Drive. Disponível em: https://www.google.com.br/drive/apps.html
[10]Trello. Disponível em: https://trello.com/pt-BR
[11]Google Calendar. Disponível em: https://calendar.google.com/calendar/u/0/r

Tabela 5.6 – Referências das entrevistas sobre dificuldades na pandemia

Como foi sua experiência no curso durante a pandemia? Dificuldades	
ID	Respostas
R24	*"Concentração, é muito fácil estar em casa aí o ambiente desmotiva. Conexão com internet, processo mais demorado. Falta do contato humano"*
R22	*"Sim, depois da superação tu acabas esquecendo. Tive dificuldades de acessar, organizar na agenda da turma, dificuldades no código e na relação com os colegas no início, trocar conhecimento"*
R9	*"Acesso ao sentimento dos alunos, contato físico, acompanhamento individual. Infraestrutura dos alunos, sem acesso, ou equipamentos"*
R20	*"Acostumar-se a mexer nas tecnologias, trabalho remoto, conciliar os horários com as atividades da casa. Problema no computador, sem resolver na hora, demora e atrapalha. No presencial seria mais rápido e dinâmico"*
R19	*"Aprendizado a distância, concentração, tentar expressar a opinião diante da turma, sem olhar para as pessoas. Tem gente que assume a frente e não dá oportunidade de falar, com ponto de vista bom. Falta de confiança em ele mesmo para expressar sua opinião"*
R29	*"Dificuldade de comunicação, aprender remoto, distração durante o trabalho"*
R7	*"Dificuldade comunicação, ferramentas de apoio, ou dos computadores que não estão bons, dificuldade de conexão, protótipos, dinâmicas criativas"*

- As **Dinâmicas criativas** (inovação de ideias): sem a necessidade de contato com as pessoas e **Ferramentas de apoio** (novas): no início foi difícil encontrar ferramentas para trabalhar de forma online foi uma dificuldade encontrada por R7;
- R22 e R19 mencionam a **Interação** (com as alunas e mentoras, sem o contato pessoal presencial): foi difícil interagir com pessoas diferentes, compartilhar ou perguntar;
- O **Planejamento** (no começo da pandemia, no início da adaptação): organizar as tarefas do curso separada das tarefas diárias de casa, foi mencionado por R20;
- **Reação das estudantes** (devido às *Webcams* desligadas ou áudio no mudo): R22 menciona que ver a reação das estudantes a cada conteúdo ensinado, não foi possível.

A categoria **autonomia** resultou tanto nas facilidades e dificuldades encontradas por pessoas professoras e estudantes. A categoria **Interação** foi a mais mencionada nas respostas, devido à necessidade do contato presencial, antes da pandemia.

5.6.1.3 Infraestrutura

Durante a pandemia, o uso de equipamentos como computadores, periféricos[12] de entrada e saída (teclados, *Webcams, mouses*, *Headset* e conexão com a internet, entre outros foram essenciais para as atividades profissionais e educacionais. Antes de iniciar as atividades de ensino de Engenharia de Software, as mentoras realizaram uma consulta com todas as estudantes para entender a possibilidade de disponibilidade de infraestrutura para as aulas em ES. Um total de 6 estudantes informaram que não tinham computadores e periféricos de entrada e saída - informaram que tinham apenas um *Smartfone* para as atividades de ensino. As 6 estudantes (R3, R18, R19, R20, R21 e R25), classificaram sua classe social como "E"e informaram que eram pessoas em situação de vulnerabilidade social [84], durante as entrevistas.

As mentoras do curso de capacitação em Engenharia de Software realizaram uma consulta com suas empresas para identificar a possibilidade de empréstimos ou doações desses equipamentos. Após esse levantamento, 4 estudantes conseguiram o empréstimo dos equipamentos e 2 estudantes, por meio de doações. Toda a logística e transporte foi realizada por pessoas mentoras do curso. Com isso, as atividades de ensino de Engenharia de Software começaram a ser executadas.

5.6.2 Ensino de Engenharia de Software

Para entender como a Engenharia de Software foi ensinada para as estudantes do curso de capacitação, foram consideradas as diretrizes curriculares para programas de graduação em Engenharia de Software ACM / IEEE como base para as perguntas sobre ensino de Engenharia de Software, durante as entrevistas. Elas foram organizadas de acordo com as 10 áreas do conhecimento em educação em Engenharia de Software - SEEK.

Nas subseções a seguir, as 10 áreas do conhecimento em educação em Engenharia de Software foram dispostas de acordo com o desenvolvimento de cada resposta das pessoas respondentes.

5.6.2.1 Fundamentos da Computação + Fundamentos da Matemática e Engenharia

De acordo com [7] a área de Fundamentos da Computação está focada nas bases da Ciência da Computação que dão suporte ao design e a construção de produtos de software. A área de fundamentos da Matemática e Engenharia fornecem fundamentos

[12]Os periféricos podem ser usados para fornecer entrada de dados para um computador e, também, para obter a saída desejada [17].

teóricos e científicos para a construção de produtos de software com os atributos desejados [7].

Durante o processo de coleta de dados, foram identificadas a necessidade de agrupar essas duas áreas do conhecimento de educação em Engenharia de Software (Fundamentos da Computação + Fundamentos da Matemática em Engenharia), pois as respostas foram comparadas igualmente como uma só área de conhecimento pelos respondentes das entrevistas, além das categorias (coincidentes) resultantes do processo de codificação dos dados.

Para entender como fundamentos da computação e fundamentos da matemática e engenharia eram ensinadas para as estudantes do curso de capacitação em Engenharia de Software, são apresentados os resultados as respostas de cada respondente na Tabela 5.7. Os processos de codificação de cada entrevista foram iniciados gerando as categorias apresentadas mais a diante.

As categorias são ilustradas na Figura 5.4 foram geradas no processo de codificação e descritas nos tópicos a seguir:

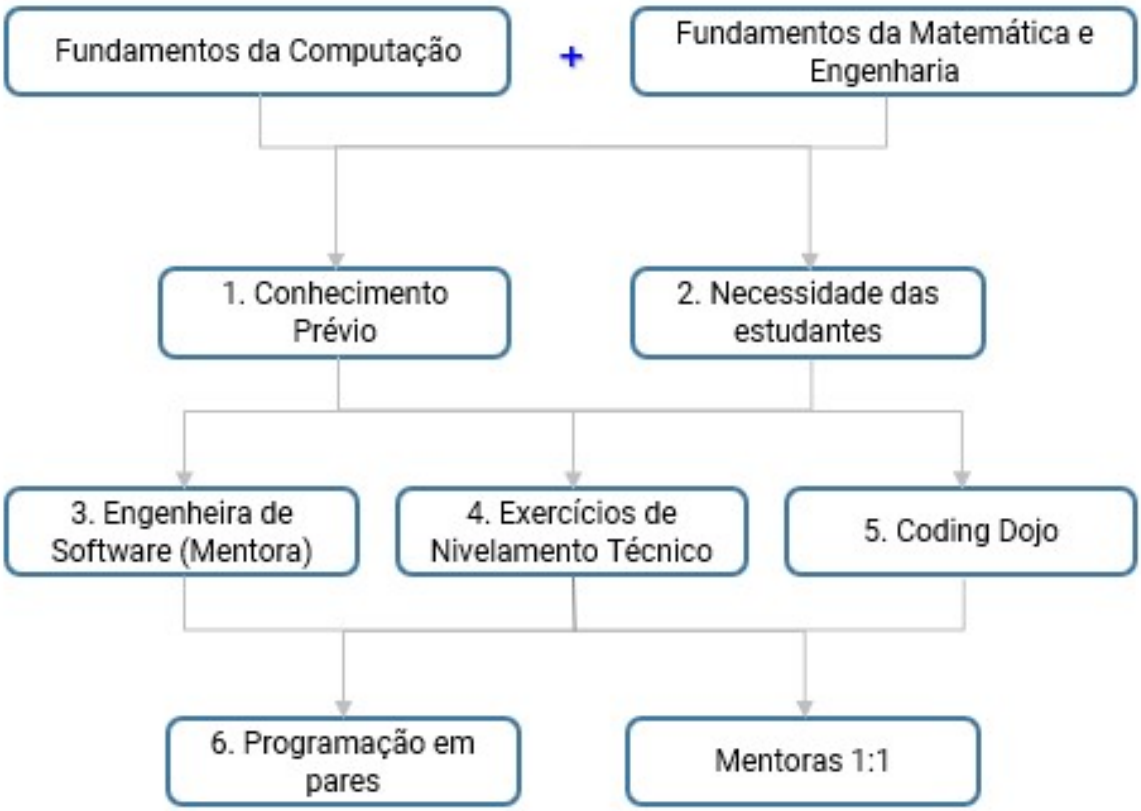

Figura 5.4 – Categorização: Fundamentos da Computação, Matemática e Engenharia
Fonte: a autora (2021)

Tabela 5.7 – Referências das entrevistas sobre fundamentos ES

Como a área de Fundamentos da Computação foi ensinada? + Como a área de Fundamentos da matemática e Engenharia foi ensinada?	
ID	Respostas das entrevistas
R1	*"Nivelamento técnico através de exercícios. Destravar, orientado a algo. Fragmento de processos práticas, orientação objetos. Sessões exercícios, review de código, não tem checklist de grade curricular"*
R11	*"Já na parte Aceleradora Ágil é com exercícios de nivelamento técnico, código sessão de coding Dojos, diretamente no código. Entender o que é uma lógica"*
R5	*"Tudo gira em torno de sessões, com nivelamento técnico junto com mentoria, com exemplos reais, ajudando nas dúvidas"*
R10	*"Quando eu era aluna, eu sentia que não conseguia ver a lógica sendo iniciada. Mas agora como mentora o conteúdo vai sendo adaptado de acordo com as necessidades das estudantes"*
R8	*"As sessões iniciais, de lógica e orientação objetos. Repositório de exercícios de lógica, com exercícios de nivelamento técnico"*
R3	*"Logo no começo, é realizado o nivelamento técnico pra ver o que cada aluno já conhece, e o que precisa aprender. Sobre programação orientada objetos acontece após o nivelamento, através de coding DOJOs, exercícios com a turma e acompanhamento de mentores engenheiros de software"*
R27	*"Foi bem complicado, foi início de pandemia, apoio de mentores engenheiros de software com sessões de java, dojos, ajuda de um colega, sessões práticas e dojos técnicos"*
R26	*"Dojos técnicos, sessões de atividades práticas, pareamento, exercícios de nivelamento"*
R24	*"Nivelamento técnico, com exercícios ou em duplas ou em trios, várias sessões com mentores 1:1, praticando junto, pareando, dojo. Muita coisa prática"*
R25	*"Foram feitas no início sessões com mentores engenheiros de software, com exemplos e exercícios práticos. A turma se juntou para fazer junto, com apoio de mentores. Pedíamos mais uma sessão caso fosse necessário"*
R21	*"Bom, através dos mentores engenheiros de software. Eles pegam um exemplo e mostram. Eles propõem um exercício e deixam a gente a vontade para ir fazendo sozinho com dicas e apoio. Rotacionando as duplas que compartilham conhecimento entre si"*

O **1. Conhecimento prévio** foi mencionado pela pessoa respondente R1, no qual indica-se que as pessoas estudantes tenham experiência com lógica de programação, linguagens de programação ou vivência na área. Essa parte é fundamental para que os fundamentos da computação sejam estudados de forma mais revisada, sem detalhes aprofundados. Com esse conhecimento prévio, as pessoas mentoras conseguem visualizar a necessidade de novos conhecimentos para serem introduzidas nas estudantes do curso.

A **2. Necessidade das estudantes** foi percebida pela pessoa respondente R10, em que, após a identificação do conhecimento prévio das estudantes, é realizado um levantamento por meio das pessoas engenheiras de software nas estudantes do curso. A ideia

é entender por meio de exercícios básicos de programação qual é o nível de conhecimento dos estudantes para introduzir os conhecimentos novos de forma correta e objetiva.

A pessoa **3. Engenheira de Software (Mentora)** foi descrita por R3, R21, R25 e R27, como: a pessoa engenheira de software é uma pessoa mentora, vinculada ao curso de capacitação por meio de uma empresa parceira deste projeto. É uma pessoa que vive diariamente a Engenharia de software na indústria de desenvolvimento de software. As mentoras têm um papel fundamental no desenvolvimento das estudantes, sendo responsáveis pelas atividades de ensino de Engenharia de Software, assuntos atuais da área de Tecnologia da Informação - (TI) e acompanhamento individual por meio de reuniões de 1:1.

Os **4. Exercícios de nivelamento técnico**: São exercícios criados por estudantes de turmas anteriores do curso de capacitação em Engenharia de Software, com apoio e supervisão de pessoas mentoras. A ideia desses exercícios é praticar os fundamentos da computação, lógica de programação e matemática com as novas estudantes do curso. Durante a pandemia, esses exercícios foram criados e desenvolvidos por meio do *Coding Dojo* e *live coding*. Com essa prática é possível entender o que os estudantes sabem, o que precisa ser ensinado e o que não é necessário ensinar novamente. Isso é calculado pelo tempo das entregas e acompanhamento de pessoas mentoras nessas atividades. Essa categoria foi criada de acordo com as respostas das respondentes R3, R5, R8 e R11.

5. *Coding Dojo*: É um método de compartilhamento de conhecimento para o desenvolvimento de exercícios de programação de computadores. É a construção de uma solução (código) para um determinado problema por meio do *live coding* criando uma solução em conjunto com as pessoas mentoras em conjunto e as estudantes do curso de capacitação em Engenharia de Software. O *Coding Dojo* pode ser realizado quantas vezes forem necessárias e a qualquer momento que for solicitada pelas estudantes. Isso ajuda os estudantes a destravar uma atividade que está demorando muito tempo para ser resolvida por elas. Pode ser feito também entre todas as estudantes ou com em conjunto com as mentoras.

A **6. Programação em pares**: É uma prática do *Extreme Programming* - (XP). Essa prática propõe que o desenvolvimento de código seja implementado por uma dupla de pessoas, trabalhando de forma colaborativa, ao mesmo tempo, no mesmo computador, de forma alternada de quem está codificando [91]. Essa prática é introduzida para os estudantes, por meio de sessões de compartilhamento de conhecimento das pessoas mentoras para as estudantes. É um momento de juntar esforços e de compartilhar conhecimento. Os exercícios de nivelamento podem ser realizados por meio da programação em pares ou por meio do *Coding Dojo*, de acordo com R24 e R26.

a **7. Mentoria 1:1**: A mentoria 1:1 são reuniões semanais e ou quinzenais realizadas por uma pessoa mentora e uma estudante. A ideia desse momento é entender as necessidades, plano de carreira e realizar esse acompanhamento em cada reunião 1:1 e foi mencionada por R24. Esse ponto foi mencionado especialmente por pessoas em situa-

ção de vulnerabilidade social, classificadas como Classe Social "E". Essas reuniões tinham como intuito auxiliar no desenvolvimento de *Soft Skills*[13], resolução de conflitos, por meio de um processo sistematizado, em que a estudante fosse capaz de desenvolver-se. Pode ser realizado um questionário com pontos que a pessoa sente conforto e desconforto ou que a estudante tem interesse em trabalhar, em uma escala de 1 a 10.

5.6.2.2 Prática Profissional

A área de prática profissional para as estudantes é necessária para o desenvolvimento de conhecimento, habilidades e as atitudes de uma pessoa engenheira de software que deve possuir para praticar a Engenharia de Software com profissionalismo, responsabilidade e ética [7].

No curso de capacitação em Engenharia de Software, existe o apoio de pessoas mentoras que trabalham em indústria de desenvolvimento de software, como *Coach* de métodos ágeis, analista de negócios (BA), analista de qualidade de software (QA) e designer experiência do usuário - (UX). Além destas pessoas especialistas da área de Engenharia de Software, existe um apoio fundamental de uma profissional de psicologia focada no desenvolvimento pessoal das estudantes do curso. A seguir, serão apresentadas as respostas das entrevistas na Tabela 5.8.

Neste contexto, as categorias apresentadas a seguir são resultados do processo de codificação aberta, seletiva e teórica. A Figura 5.5 ilustra o processo de relação das categorias sobre a prática profissional em Engenharia de Software.

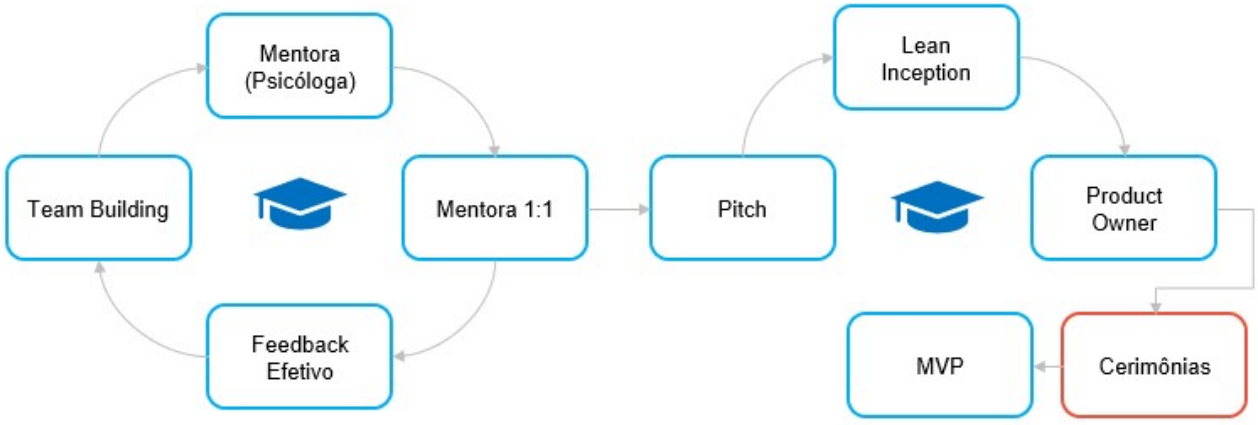

Figura 5.5 – Categorização: Prática profissional em Engenharia de Software
Fonte: a autora (2021)

Cada categoria será apresentada a seguir:

[13]Segundo Dixon et al. [30] *Soft Skills* é uma combinação de habilidades interpessoais e sociais, por outro lado, incluem os procedimentos técnicos ou administrativos que podem ser quantificados e mensurados.

Tabela 5.8 – Referências das entrevistas sobre Prática Profissional

Em relação a Engenharia de Software, como a Prática Profissional é ensinada?	
ID	Repostas das entrevistas
R1	*"É mais rico que um curso de graduação, alinhamento de um projeto, como conduzir essas questões, feedbacks, questões de melhoria de processo, refinamento progressivo de escopo de projeto. Questões práticas profissionais, desde o início. Essa é uma das partes ricas do curso. Avanço do projeto que propicia novos conhecimentos (depende avançar o projeto). Ter projetos que rodem por mais de um semestre. Talvez mais práticas poderiam ser vistas. Inception inicial do zero, mas não tem oportunidade de ver outras coisas"*
R2	*"Buscar um projeto em que seja compatível e possível desenvolver no decorrer do curso. Pré Inception, Inception, Retrospectivas, dailys, inserção de termos e cerimônias. Ela propôs uma dinâmica de pareamento (não técnico) voltado de team building (aprimoramento de relação interpessoal dentro de um time). Esse momento foi proporcionar para as alunas um momento que elas se conhecerem, formando duplas, duplas não aleatórias, através da observação. Mistura de pessoas diferentes, aumentando esse vínculo com pessoas de perfis diferentes, depois no terceiro momento foi feito um sorteio aleatório entre as alunas, foco no direcionamento através de entrevistas, como questões da aceleradora e da vida pessoal. Aprofundamento das perguntas com enfoque do aspecto da criatividade e imaginação, trocas bem profundas através das perguntas"*
R5	*"Além das sessões, vem muito do fato de simular um projeto real, com entregas, contatos com clientes, sessões de ágil, fazer com que a turma entenda e faça essas práticas"*
R10	*"Tentar trazer um pouco da experiência de mentoria de como é feito na indústria, sempre dando autonomia, como dailys, retrospectivas, trazendo métodos ágeis, com adaptações nas necessidades reais do projeto"*
R8	*"Foram apresentadas sessões sobre o manifesto ágil, daylis, retrospectivas, sessões teóricas, práticas com acompanhamento de profissionais"*
R21	*"Através da mentora psicóloga trouxe conteúdo propondo uma aluna para ser parceria para montar a facilitar um momento. Cada um tem a oportunidade e experiência vivenciadas"*
R25	*"Sessões sobre o assunto, ou a gente tinha uma conversa com os mentores de pós com dinâmicas com a mentora psicóloga sobre isso, na prática também. Depois fazemos juntos com todos ao vivo"*
R18	*"Sessões teóricas com exemplos práticos, dailys, retrospectivas e showcases com apoio de mentores, depois fica por conta do aluno, com dicas e ajuda da mentoria 1:1. Dinâmicas de team building também ajudou"*
R20	*"Através de sessões sobre assunto, sessões teóricas e com exemplos práticos, tivemos o Pitch que foi um momento muito massa. E com isso a efetivação prática. Toda semana tem sessão com as POS para alinhar as questões do projeto"*
R19	*"Lá no início aprendemos o que era Lean Inception. Tivemos Sessões teóricas e práticas, e cada encontro vai aprendendo com a parte interessada, como facilitar, como fazer. Tudo de forma prática, visão de projeto na aceleradora. Apoio da mentoria 1:1 e feedbacks"*

A dinâmica de ***Team Building***: é uma formação de equipes para aprimorar relações sociais e definir papéis [31]. Elas acontecem por meio de dinâmicas em duplas ou em grupo para "quebrar gelo"e realizar a integração entre as estudantes. Essa formação é realizada por uma mentora psicóloga e outras mentoras de Engenharia de Software. Geralmente é a primeira atividade que deve ser realizada quando uma nova turma inicia o curso de capacitação em Engenharia de Software. Elas podem ser feitas a cada 15 dias, alternando as duplas, grupos e ideias de dinâmicas e assuntos abordados. Todas as pessoas estudantes entrevistadas mencionaram que o *Team Building* foi fundamental para suas formações nesse período. Essa categoria foi gerada de acordo com as respostas das pessoas respondentes R2 e R18.

A **Mentora Psicóloga**: responsável por auxiliar no desenvolvimento pessoal das estudantes do curso de capacitação em ES. As entrevistadas ressaltaram a importância do papel dela nesse período da Pandemia do Coronavírus, devido ao isolamento social e a necessidade de contato físico entre a turma. Esta categoria foi gerada de acordo com as respostas de R21 e R25. A formação *Team Building* foi essencial nesse período, além da mentoria 1:1 - também realizada pela mentora psicóloga;

O ***Feedback* Efetivo**: prática de mentora para estudante, estudante para estudante, estudante para mentora ou mentora para mentora. Possibilidade de trazer pontos positivos e construtivos durante as observações no comportamento, nas práticas de ensino e aprendizado para o crescimento profissional e pessoal da estudante ou mentora. *Feedback Box, Feedback 360, One-on-one*. Essa categoria foi gerada de acordo com R1 e R19.

As categorias apresentadas anteriormente são um ciclo que acontece do início ao fim do curso de capacitação em ES.

O respondente R20 menciona que o ***Pitch*** é importante. O curso de capacitação em ES, disponibiliza todo o início de uma turma nova de estudantes, a chamada de projetos com interesse social para serem desenvolvidas pelos estudantes. Esse momento acontece por meio do *Pitch* - momento de defesa de ideia de projeto, apresentado por *Product Owner* - (POs). As estudantes acompanham os *Pitchs* já pré selecionados por mentoras (seleção feita com base nas condições do projeto em ser código livre e a entrega ser produto mínimo viável). Após as defesas dos POs, as estudantes escolhem de dois a três projetos para conhecer melhor e realizar a pré *Inception*.

Os respondentes R19 e R10 citam o **Lean Inception**. Elas são práticas colaborativas para o alinhamento de um grupo de pessoas sobre o produto mínimo viável - (MVP) [18]. Essas práticas (Pré *Inception*) são realizadas após a seleção de dois ou três projetos do *Pitch*, com a pessoa mentora *Coach*, POs de um dos projetos selecionados e estudantes do curso. São realizadas dinâmicas de jornada de usuário, comparação com ideias já existentes no mercado de tecnologia, definição de *personas*, ideias de protótipos entre outros.

Ao final dessas práticas, as estudantes escolhem um projeto para a *Inception*. Todas essas práticas são baseadas em referências nos estudos de Paulo Caroli [18].

O ***Product Owner* - (POs)**: São pessoas que utilizam o *Scrum* (ou alguma técnica similar) para definir estórias e priorizar o *backlog*[14] do projeto que as estudantes do curso de capacitação em ES escolhem para desenvolver. Os POs devem comparecer no mínimo uma vez por semana no *Showcase* para acompanhar e definir as entregas e melhorias do projeto, respectivamente. Além disso, devem ter disponibilidade nos canais de comunicação do curso para responder às dúvidas das pessoas estudantes e mentoras durante a semana sobre o projeto. Essa categoria foi gerada de acordo com R5 e R20.

As **Cerimônias Ágeis** foi uma categoria gerada pelas respostas de R2, R10 e R8: É uma parte fundamental de métodos ágeis, são essenciais para desenvolver um projeto de qualidade com acompanhamento dos POs, mentoras e estudantes. Elas são divididas em quatro tipos, conforme ilustrado na Figura 5.6, a seguir:

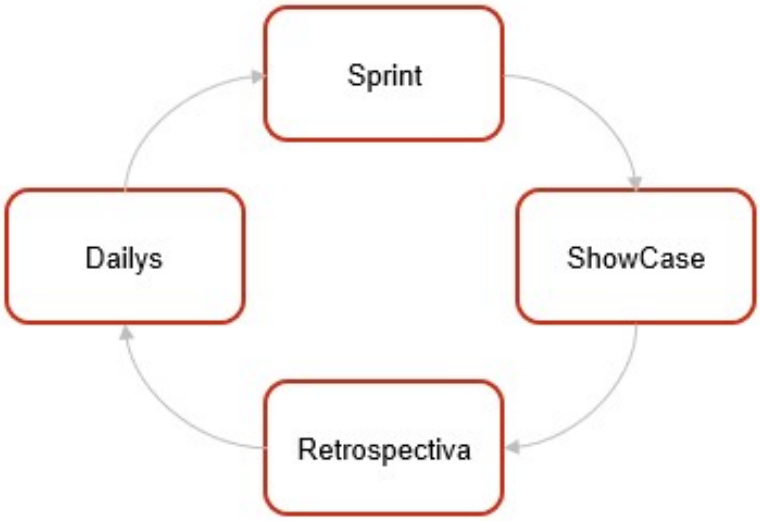

Figura 5.6 – Categorização: Cerimônias ágeis
Fonte: a autora (2021)

Dailys: são reuniões diárias de 15 minutos, geralmente na primeira hora do dia, onde os estudantes informam o que está acontecendo, o que já foi feito e o que precisa ser feito no desenvolvimento do projeto. Nesse momento as estudantes alinham e organizam os pontos no *Kanban*. No início, as mentoras acompanham, controlam o tempo e alinham os pontos que devem ser tratados. Logo depois, fica por conta de as estudantes realizarem

[14]Um *product backlog* (ou simplesmente *backlog*) é uma lista de itens de trabalho (por exemplo, histórias de usuários, *bugs* pendentes e várias tarefas) usados por equipes de software para coordenar o trabalho que deve ser feito [89].

de forma autônoma, essas tarefas. Essas *Dailys* acontecem com o apoio da plataforma *Miro*[15].

Sprint: são encontros semanais para planejar e preparar as atividades com êxito das estudantes. Ao participar da reunião, os POs do projeto terão uma lista de pendências de produto que deve ser priorizada, em que são discutidas cada item com as estudantes, mentoras, estimando o esforço envolvido. Esses encontros ocorrem no mesmo dia que o *ShowCase*.

ShowCase: é um momento para mostrar parte do projeto desenvolvido pelas estudantes para os POs, naquela semana. Esse momento é fundamental para receber *feedbacks* dos trabalhos desenvolvidos até então. Para esse momento, o uso da plataforma *Miro* foi fundamental.

Retrospectiva: As retrospectivas são momentos que acontecem a cada 15 dias e no final do curso de capacitação em ES. As estudantes, mentoras e POs, podem trazer pontos que estão funcionando, o que precisa ser melhorado e ações para construir soluções dos problemas encontrados. Geralmente quem facilita esse momento é uma pessoa mentora e uma estudante - essa, para desenvolver as habilidades das estudantes em facilitação e comunicação. As retrospectivas são apoiadas pela plataforma *FunRetro*[16].

O R5 mencionada o **Produto Mínimo Viável - MVP**: é a construção da versão mais simples e enxuta de um produto, empregando o mínimo possível de recursos para entregar a principal proposta de valor da ideia [85]. Assim, é possível validar o produto antes de seu lançamento. Essa construção é realizada com as pessoas estudantes do curso de capacitação em ES, com suporte das pessoas mentoras e POs. As tecnologias para o desenvolvimento do projeto são definidas no início do desenvolvimento do projeto, com foco em tecnologias que estão aquecidas no mercado de TI.

A categoria **Mentoras 1:1** também foi mencionada por R21 e R25, como um ponto muito forte no ensino de prática profissional em Engenharia de Software. As estudantes em situação de vulnerabilidade social, mencionaram um **questionário de mentoria sistematizada** desenvolvido por pessoas mentoras 1:1 para as mesmas. No primeiro dia de mentoria 1:1, as pessoas estudantes poderiam responder as questões do questionário a seguir, nivelando de 1 a 10 (1: nada confortável e 10: muito confortável).

1. Trabalhar de forma 100% online;
2. Comunicar-se em grupo;
3. Programar sozinha;
4. Programar com um par;

[15] *Miro - Remote Collaboration* é uma plataforma de quadro branco colaborativa, essencial para trabalhos online de equipes de desenvolvimento de software. Disponível em: https://miro.com/.

[16] Plaforma para criar retrospectivas online. Disponível em: https://www.funretrospectives.com/.

5. Errar;
6. Receber *feedback* positivo;
7. Receber *feedback* construtivo;
8. Facilitar Sessões de conhecimento em grupo;
9. Interagir com os POs;
10. Expor sua opinião técnica;

Após o preenchimento deste questionário, a mentora 1:1 realiza encontros semanais e ou quinzenais, trabalhando os pontos nivelados com a numeração mais baixa, com a estudante. Após o final do curso, esses pontos eram revistos e nivelados novamente para a estudante visualizar sua evolução no curso.

5.6.2.3 Modelagem e Análise de Software + Análise e Especificação de Requisitos

A modelagem e análise de sistemas podem ser consideradas pontos centrais em qualquer disciplina de engenharia de software, pois são essenciais para documentar e avaliar as decisões alternativas do projeto [7]. A análise e especificação de requisitos inclui a elicitação e análise das necessidades das partes interessadas e a criação de uma descrição apropriada do comportamento e qualidade desejados do software, juntamente com as restrições e suposições relevantes [7].

Durante o processo de coleta de dados foram identificadas a necessidade de agrupar essas duas áreas do conhecimento de educação em Engenharia de Software (Modelagem e Análise de Software + Análise e Especificação de Requisitos), pois as respostas foram comparadas igualmente como uma só área de conhecimento de acordo com os respondentes. A seguir são apresentadas as respostas das entrevistas identificadas com "ID"colorido para mostrar as relações das categorias, conforme Tabela 5.9.

As categorias geradas também coincidiram nas duas áreas. A seguir são apresentadas as categorias geradas do processo de codificação, conforme a Figura 5.7.

O ***Coach* (mentora)**: é responsável por ensinar métodos ágeis, conduzir o planejamento, revisão e retrospectiva de cada iteração. O *Coach* também apoia a assistente de ensino, ensinando as atividades mencionadas anteriormente e realiza a avaliação final do projeto. Essa pessoa mentora também participa, organiza e direciona os encontros semanais de *ShowCase* e *Sprint* com os POs e estudantes do curso de capacitação, além de ajudar na modelagem de software. A pessoa *Coach* fica disponível para ajudar as pessoas estudantes e a assistente de ensino nas dúvidas que ocorrem na semana, por meio de uma plataforma de comunicação utilizada no curso. Os conteúdos de modelagem de sistemas

Tabela 5.9 – Referências das entrevistas sobre Modelagem + Requisitos

Como a área de Modelagem e Análise de Software foi ensinada? + Como a área de Análise e Especificação de Requisitos foi ensinada?	
ID	Respostas das entrevistas
R11	*"Foi realizado uma Inception com diferentes projetos, para depois para fazer uma Inception no projeto escolhido. Personas, e algo viável para a produção. Nas ágeis sessões de teoria, mostrando na prática como funciona"*
R24	*"O Software foi apoiado com os POs, mentores"*
R22	*"Foi coletado informações, dos POs, com apoio do mentor coach, toda semana encontro com as POs, apresentando o que foi desenvolvido e identificando coisas novas"*
R9	*"Em conjunto com a mentoria, POS e alunos, com apoio de pessoas em experiência. Depois nivelar, deixando os alunos por conta própria realizar com apoio da mentoria. Sessões técnicas, sobre o assunto, dado exemplos com apoio da parte interessada, com projetos reais diversas dinâmicas"*
R4	*"Através de sessões, onde eles são divididos em grupo, onde eles pegam os exemplos, com simulação e quebra de história. Escrevendo a história como se fosse um problema real"*
R20	*"Sessões com o Coach fazendo e mostrando como exemplos na prática"*
R19	*"Skets básicas, através de protótipos, mini cards e histórias de como vai ser feito. Informações importantes, quem vai utilizar o sistema, através do apoio do Kanban"*
R29	*"O mentor Coach e os analistas de negócios, ensinaram fazendo em exemplos práticas, quebrando alguma história, pareando. Tivemos sessões de UML também"*
R18	*"Aprendemos a UML. Depois os mentores BA ajudam a quebrar as histórias, critérios de aceitação, tasks, conceito de história e consegue desenvolver com ajuda de mentores"*

e métodos ágeis, são abordados por meio de sessões de compartilhamento de conhecimento, com os assuntos e abordagens mais comuns na indústria de desenvolvimento de software. Conteúdos teóricos são introduzidos, logo depois com exemplos práticos. Essa categoria foi gerada de acordo com as respostas de R20, R22 e R29.

A categoria **Analista de Negócios (BA)** foi gerada das respostas de R18 e R29. O BA é uma pessoa mentora vinculada ao curso por uma empresa de desenvolvimento de software, parceira do curso. Essa pessoa ajuda na análise e especificação de requisitos do software que as pessoas estudantes estão desenvolvendo. Esses conteúdos são abordados por meio de sessões de compartilhamento de conhecimento, com os assuntos e abordagens mais comuns na indústria de desenvolvimento de software. Conteúdos teóricos são introduzidos, logo depois com exemplos práticos.

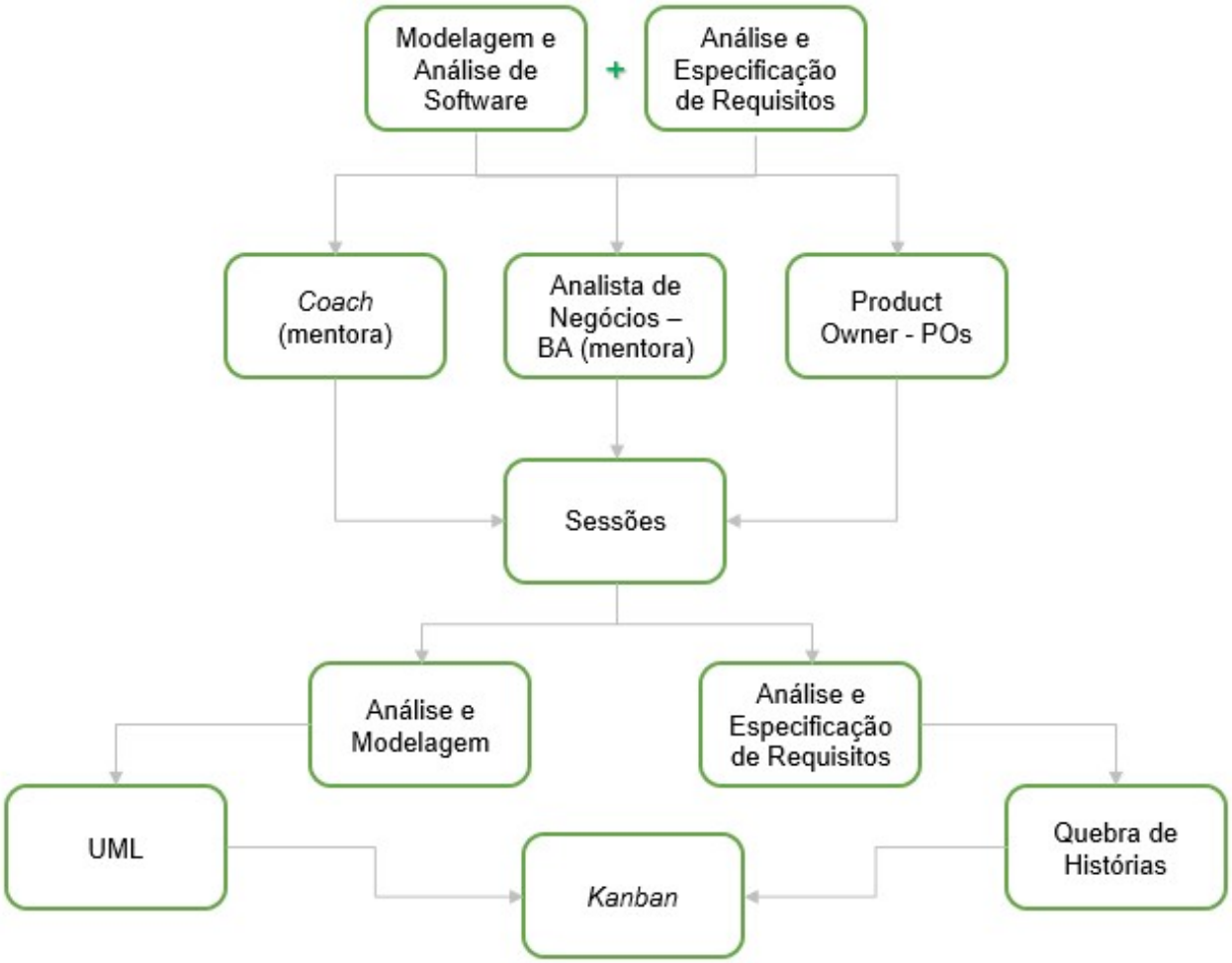

Figura 5.7 – Categorização: Modelagem e Análise de Software + Análise e Especificação de Requisitos

Fonte: a autora (2021)

A categoria ***Product Owner*** gerada de R9 e R24, também apoiam as BAs e a pessoa *Coach*, nas sessões de compartilhamento de conhecimento - R11 e encontros semanais de *ShowCase* e *Sprint*.

As **Sessões** foram categorias geradas das entrevistas de R4, R9, R11, R20 e R29 : são momentos com duração mínima de 30 min à 1h, para compartilhar conhecimento de Engenharia de Software. Essas sessões são organizadas e desenvolvidas por pessoas mentoras, com apoio ou não das estudantes. A ideia é trazer o contexto do conteúdo abordando a teoria e prática, com exemplos reais vivenciados na indústria de desenvolvimento de software. As sessões acontecem na ordem dos tópicos comuns abordados em todas as turmas do curso de capacitação, ou de acordo com novos conhecimentos inseridos e solicitados pelas estudantes. Todos os momentos de encontros e sessões, entre estudantes, POs e mentoras, acontecem por meio da plataforma de videoconferência *Google Meet*.

A modelagem e análise de sistemas é ensinada por meio de Sessões e desenvolvida por meio da linguagem de modelagem Unificada - **UML**, categoria gerada de R29. Para

analisar e especificar os requisitos, a pessoa BA introduz o conteúdo por meio de sessões as **Quebras de História** diretamente do projeto que será desenvolvido pelas estudantes.

O ***Kanban*** - R19: é um *framework* da gestão ágil que funciona para diversos tipos de equipe, já que ele é muito visual e facilmente adaptável à sua realidade [51]. Ele é construído com as mentoras, POs em conjunto com as estudantes do curso. Devido às atividades serem online, o *Kanban* foi criado por meio da plataforma *Trello*[17].

5.6.2.4 Design de Software

O design de software foca nos problemas, técnicas, estratégias, representações e padrões usados para determinar como implementar um componente ou sistema [7]. No curso de capacitação em Engenharia de Software, as práticas de design de software são ensinadas por meio de **Sessões** de compartilhamento de conhecimento. Essas sessões são facilitadas por pessoas mentoras na área de designer de produto e UX (experiência do usuário) - profissionais da indústria de desenvolvimento de software, vinculada ao curso de capacitação em ES. As categorias apresentadas foram resultadas do processo de codificação (análise dos dados) e geradas das entrevistas, conforme apresentado na Tabela 5.10 a seguir.

A Figura 5.8 ilustra as categorias geradas da Tabela 5.10.

O **Design de Produto**: é uma pessoa mentora que trabalha na indústria de desenvolvimento de software com o concebimento de produtos, sempre preocupada com as necessidades do usuário, considerando as possibilidades e sempre zelando pela segurança do uso do produto (software) pelo usuário. Esse profissional tem a função de compartilhar conhecimento com as estudantes do curso de capacitação em ES, por meio de **Sessões** dos seguintes tópicos (categorias), de acordo com R7:

- **Interface humano-computador** por meio de **Sessões** de teoria e exemplos práticos, são abordadas assunto como usabilidade, heurísticas de Nielsen[18] e escala de cores.
- **Identidade visual**, categoria gerada pela entrevista de R8: a identidade visual do projeto desenvolvido pelas estudantes são construídas com as mentoras (UX e Design de produto), categoria gerada de R7, R8 e R10 utilizando a plataforma ***Figma***[19] e sendo validadas com as POs.
- De acordo com R8, **Prototipação**: os tipos de protótipos existentes são ensinados por meio de **Sessões** por uma mentora especialista na área de **Experiência de Usuário** (UX). Após a introdução da teoria, os protótipos do projeto desenvolvido pelas estudantes são criados na plataforma ***Figma*** e validadas com os POs.

[17] Trello: uma ferramenta baseada em nuvem que usa o método *Kanban* de gerenciamento de projetos [48]

[18] São 10 heurísticas que devem ser levadas em consideração no desenvolvimento de qualquer produto de software [66].

[19] Figma: uma plataforma de edição gráfica de vetor e prototipagem. Disponível em: https://www.figma.com/.

Tabela 5.10 – Referências das entrevistas sobre Design de Software

Como a área de Design de Software foi ensinada?	
ID	Respostas das entrevistas
R7	*"Eu como mentora preparei algumas sessões, com introdução de UX, Design de produto, IHC em slides, fui verificando a necessidade de conhecimento de acordo com o desenvolvimento do projeto. Mostrar o básico de ferramentas que auxiliam nesse processo, como FIGMA. Fizemos exercícios como prática. Dinâmicas em grupos, práticas com exemplos reais. Geração de alternativas"*
R10	*"Com apoio de mentores UX, com base de conhecimento de como é feito no mercado, sessões de padrões de mercado, sem fugir do mercado, com pessoas que já trabalham na área. E com isso eles conseguem identificar esse ponto (estudantes)"*
R8	*"Na Aceleradora, com apoio de pessoas especialistas de UX, a prototipagem, identidade visual é realizada com sessões e exemplos práticos. O desenvolvimento e desenho é feito com pessoas especialistas, de acordo com conhecimento e parte técnica das alunas"*
R3	*"Atualmente é realizada algumas sessões, como cores, padrões, tipografia, sessões teóricas e práticas com apoio de mentores especializados. Conhecer o público-alvo"*
R24	*"Teve apoio de mentores especialistas na área mostrando através de sessões práticas e exemplos. Além da minha mentora 1:1 ser da área, eu sempre tirava minhas dúvidas em nossas conversas porque eu sou uma pessoa tímida"*
R21	*"Fizemos um percurso com o design de UX de história com a opinião de clientes, como funciona um site bom. Depois levamos essa ideia para o FIGMA (prototipação do software), fizeram alguns exercícios com a gente para entender a plataforma coletando feedbacks com as POs. Lapidavam mais um pouco até tornar-se lapidadas por ela"*

Outra categoria gerada pelas menção da entrevistada R8 foi a **mentora 1:1** nessa área de conhecimento. As estudantes trouxeram que esse apoio das mentoras foi essencial para as dúvidas que surgiram nas atividades de sessões - mas não tinham conforto para questionar na frente de alguns colegas de curso. Essas dúvidas eram sanadas nos encontros 1:1.

5.6.2.5 Verificação e Validação de Software + Qualidade de Software

A verificação e validação de software usa uma variedade de técnicas para garantir que um componente ou sistema de software satisfaça seus requisitos e atenda as expectativas das partes interessadas [7]. A Qualidade de Software é uma preocupação transversal, identificada como uma entidade separada para reconhecer sua importância e fornecer um contexto de garantia a qualidade em todos os aspectos da prática e do processo de Engenharia de Software [7].

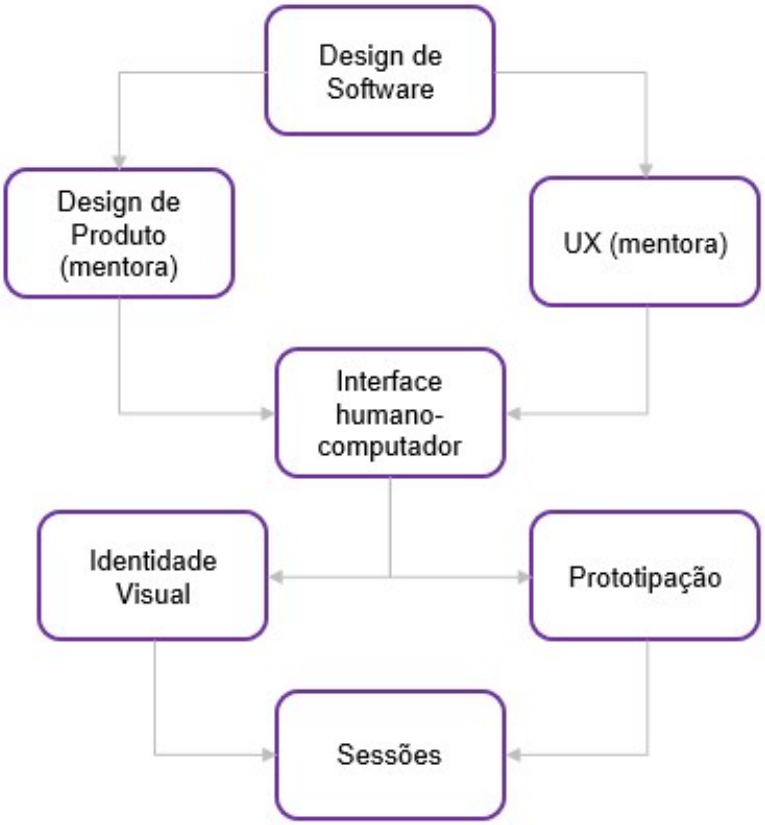

Figura 5.8 – Categorização: Design de Software
Fonte: a autora (2021)

Durante o processo de coleta de dados foram identificadas a necessidade de agrupar essas duas áreas do conhecimento de educação em Engenharia de Software (Verificação e Validação de Software + Qualidade de Software), pois as respostas foram comparadas igualmente como uma só área de conhecimento de acordo com os respondentes. A Tabela 5.11 apresenta as respostas das entrevistas e posteriormente a Figura 5.9 apresenta as categorias geradas das entrevistas.

As categorias geradas também coincidiram nas duas áreas. A seguir são apresentadas as categorias geradas do processo de codificação aberta, conforme a Figura 5.9.

A pessoa **Engenheira de Software** foi uma categoria gerada das entrevistas de R20 e R29: a pessoa engenheira de software (mentora) é um profissional responsável por toda a arquitetura do software, com foco na área de desenvolvimento *Front-end e ou Back-end*, vinculado a uma empresa de desenvolvimento de software, parceira do projeto do curso. Sua responsabilidade é compartilhar conhecimento, por meio de **Sessões** - R11 e R19, para as estudantes do curso de capacitação em ES. O conteúdo da categoria gerada

Tabela 5.11 – Referências das entrevistas sobre Validação, Verificação e Qualidade de Sof.

Como a área de Verificação e Validação de Software foram ensinados? + Como a área de Qualidade de Software foi ensinada?	
ID	Respostas das entrevistas
R11	*"Sessões de Teste de Software, importância do teste unitário. Na medida do possível com práticas e suporte"*
R10	*"A documentação é bem mais pautada na clareza de um código, acredita-se muito mais em pontos detalhados, questões de como criar um projeto. Os testes são ensinados e motivados a aplicar, e isso com apoio de pessoas mentoras na área de QA"*
R8	*"A ideia de teste Unitário, com apoio de TDD (com sessões por uma QA), setup, testes como exemplo para seguir como base na prática do projeto. Sobre a validação com a POs, o contato e eles saberem o potencial do time, é bem conversado em cada encontro"*
R24	*"Na parte de QA, foi mostrado através de uma mentora de QA"*
R22	*"Condução de testes no código, através de sessões de uma mentora QA"*
R9	*"Sessões de testes automatizados, aplicados exercícios sem contexto do projeto e depois com o contexto do projeto"*
R20	*"Sessão de QA, Circle CI, testes e qualidade com mentores engenheiros de software"*
R19	*"Sessões de Teste e qualidade de software com mentores especialistas. Teste do projeto, tudo através de alunas interessadas na área para]implementar no projeto"*
R29	*"Essa parte a gente aprendeu sobre CodeReview com os engenheiros de software mentores"*

da entrevista R20, o ***Circle CI***[20] ***e Code Review***[21] - R29 são abordados de forma teórica, com exemplos práticos no projeto de desenvolvido pelas estudantes.

A categoria **Analista de Qualidade de Software - (QA)** foi gerada das entrevistas de R10, R22 e R24, define a Mentora na área de QA, é responsável por criar sessões de ensino com teoria e exemplos práticos do projeto de software, além de apoiar na parte de qualidade de software do projeto (analisar as estratégias de teste, olhando para a qualidade de código, teste de performance e escalabilidade) do projeto de software. Outra papel é **compartilhar conhecimento** sobre desenvolvimento guiado a testes - **TDD**[22], categoria criada de acordo com a R8 e R11.

Circle CI, Code Review **e TDD** práticas ensinadas, por meio de sessões na **Implementação**, categoria criada de R19 do projeto de software desenvolvido pelas estudantes do curso de capacitação em Engenharia de Software. Após muita prática, elas são recomendadas para as estudantes terem autonomia em fazerem sem apoio das mentoras.

[20] *Circle CI* é uma ferramenta de integração contínua para integrar os códigos das duplas ou grupo das estudantes desenvolvedoras com frequência. Disponível em: https://circleci.com/.

[21] A revisão de código é uma atividade de garantia de qualidade de software na qual uma ou várias pessoas verificam um programa principalmente partes do código-fonte. Elas podem ser realizadas pela pessoa engenheira de software ou a analista de qualidade [32].

[22] De acordo com Astels et al. [8] TDD são testes escritos antes da produção do código do projeto.

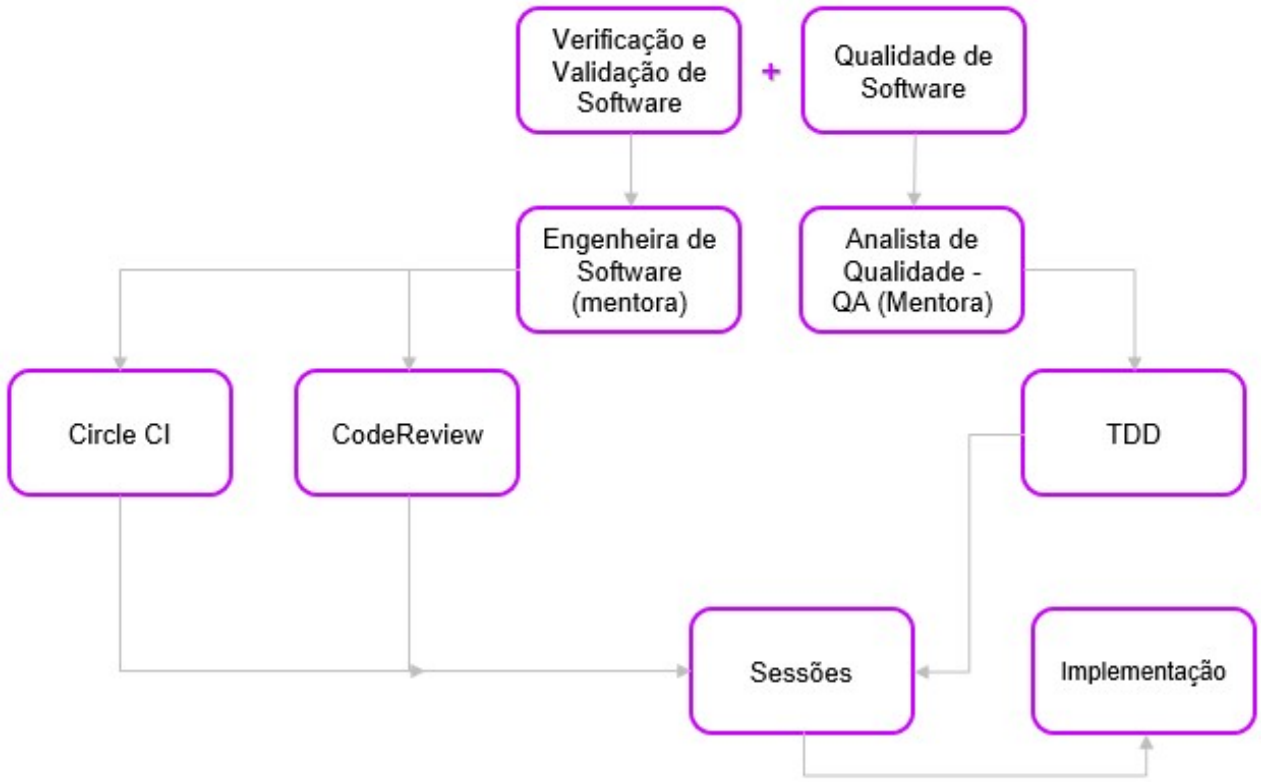

Figura 5.9 – Categorização: Verificação e validação de software + Qualidade de software
Fonte: a autora (2021)

Qualquer dúvida pode ser consultada nos canais de comunicação do curso com as mentoras de QA e Engenheiras de Software ou em reuniões 1:1.

5.6.2.6 Processo de Software

O processo de software preocupa-se em fornecer estruturas apropriadas e eficazes para as práticas de Engenharia de Software serem usadas para desenvolver e manter componentes e sistemas de software nos níveis individuais de equipe e empresas [7]. Esta área de conhecimento cobre vários modelos de processo e oferece suporte às experiências individuais e de equipe com um ou mais processos de desenvolvimento de software, incluindo planejamento, execução, rastreamento e gerenciamento de configurações.

Na área de conhecimento de Processo de Software em Engenharia de Software, apenas cinco categorias foram criadas: **Ciclo de Vida de Software**, **Gerenciamento de Requisitos**, **Gerente de Projetos**, **Sessões** e **Métodos ágeis**, de acordo com as respostas das pessoas entrevistadas, conforme apresentado na Tabela 5.12 a seguir:

O **Ciclo de Vida de Software** é apresentado por meio de **Sessões** de compartilhamento de conhecimento teóricas, sendo apresentadas os principais ciclos: a análise e a definição de requisitos; o planejamento do projeto de desenvolvimento; a implementação das funcionalidades no código-fonte; a execução dos testes de segurança e rastreamento

Tabela 5.12 – Referências das entrevistas sobre Processos de Software

Como a área de processos de software foram ensinadas?	
ID	Respostas das entrevistas
R26	*"Foi feita sessões explicando com exemplos sobre o ciclo de vida de software. Essa sessão foi apresentada por um gerente de projetos (mentor)"*
R28	*"Criamos um backlog com as tarefas que precisávamos fazer com apoio de um mentor da área de gerencia de projetos. Antes de tudo sempre tínhamos uma sessão sobre o assunto"*
R21	*"Eu aprendi muito sobre métodos ágeis no curso. Sempre lia sobre a teoria, mas a prática é totalmente diferente"*
R19	*"A gente sempre tinha sessão de compartilhamento de conhecimento antes de cada conteúdo. Aprendi sobre ciclo de vida de software"*
R23	*"Conseguíamos controlar os requisitos pelo backlog, sempre acompanhado do mentor gerente de projetos"*

de *bugs*; a integração da aplicação no ambiente de trabalho do usuário. Essas sessões são apresentadas por uma pessoa mentora na área de gerência de projetos. Essa categoria foi gerada de acordo com as respostas de R19 e R26.

O **Gerenciamento de Requisitos** foi uma categoria gerada das respostas de R23 e R28. Ela apresenta o *Backlog* para gerenciar as tarefas, com base nas prioridades definidas entre os POs, com estimativa de tempo que será necessário para completar as várias funcionalidades. O controle do *Backlog* é de responsabilidade da pessoa mentora *Coach* e estudantes do curso.

A pessoa **Gerente de Projetos** é um profissional na área de Engenharia de Software, vinculada a uma indústria de desenvolvimento de software, parceira do curso. Essa pessoa é responsável por compartilhar conhecimento sobre a área de **Métodos ágeis**, gerenciamento de projetos, em **Sessões** e acompanhar o desenvolvimento do projeto das estudantes, com suporte e consultoria na área. Essa categoria foi gerada de acordo com R23, R26 e R28.

O curso de capacitação em Engenharia de Software é totalmente focado em **Métodos ágeis**, de acordo com R21. O conteúdo da área são introduzidos desde o início do curso, com **Sessões** e exemplos práticos no desenvolvimento do projeto. Uma **Sessão** muito importante e sempre recomendada é um encontro virtual com pessoas profissionais de cada área da Engenharia de Software: Gerente de Projetos, Analista de Negócios, Desenvolvedora *Back-end*, Desenvolvedora *Front-end*, *Tech Lead*, Analista de qualidade de teste, Analista de infraestrutura, DevOps, desenvolvedora *Mobile* e *Product owner*. Essas pessoas compartilham suas funções, tarefas e os meios de acesso às oportunidades de cada área. Isso possibilita a motivação das estudantes, em entender e identificar qual área faz mais sentido seguir, nas suas futuras carreiras na área de tecnologia da informação. As categorias **Sessões** foi criada de acordo com R19, R26 e R28.

5.6.2.7 Segurança

A segurança do software tem dois componentes distintos, mas relacionados. Como um conhecimento autônomo nesta área, trata da proteção de informações, sistemas e redes. Como um corte transversal compartilha a preocupação, e fornece um foco em como a segurança deve ser incorporada em todas as partes do ciclo de vida de desenvolvimento de software. Para preparar engenheiros de software que possam desenvolver softwares, a segurança deve ser integrada com as práticas e processos associados outras áreas de conhecimento [7].

A Tabela 5.13 apresenta os resultados das entrevistas sobre a área de segurança de software no curso de capacitação em Engenharia de Software.

Tabela 5.13 – Referências das entrevistas sobre Segurança de Software

Como a área de Segurança de Software foi ensinada?	
ID	Respostas das entrevistas
R23	*"Sessão com teoria e exemplos práticos, código mais seguro e ambiente mais seguro"*
R18	*“Sessão de segurança, criptografia, dicas de segurança”*
R28	*“Fizemos uma sessão facilitada por alunos e mentores”*
R24	*“Teve uma sessão para mostrar dicas sobre segurança de sistemas”*
R25	*“Uma sessão sobre segurança, mas não colocamos em pratica nada”*

Nesse período do curso de forma 100% online, os respondentes informaram que a área de conhecimento Segurança em Engenharia de Software teve apenas um ponto ensinado, por meio da **Sessão** de compartilhamento de conhecimento, sobre **Segurança de Sistemas**. Foram apresentadas para as estudantes princípios, exemplos e características de Segurança de sistemas de software de forma teórica.

5.6.3 Lições gerais aprendidas

Nesta seção serão apresentadas as lições gerais aprendidas durante o desenvolvimento dessa pesquisa.

Lição 1: Diferença no Ensino online para o ensino presencial

Todos os 24 entrevistados responderam que existe uma diferença do ensino presencial para o ensino online. Diferenças nas abordagens de ensino, devido à falta de contato físico. Em vários momentos de sessões, as estudantes não interagiam por áudio ou com suas câmeras ligadas, ou seja, não tinha como acompanhar o entendimento das estudantes naquele momento e isso foi mencionado também pelas estudantes do curso. Outro ponto foi o avanço do projeto, ele avançou de forma mais lenta no ensino online.

Lição 2: Plataformas de comunicação e videoconferência

A possibilidade de acesso às plataformas de comunicação e videoconferência. No primeiro dia, algumas estudantes não conseguiram entrar na sala virtual para as apresentações no *Team Building*, devido às dificuldades de acesso ou por nunca terem utilizado aquele meio de comunicação. Aos poucos, foram aprendendo as formas de conectar e interagir pelas plataformas.

Lição 3: Sessões

Todas as 10 áreas de conhecimento em Engenharia de Software foram abordadas por meio de sessões de compartilhamento de conhecimento, facilitadas por pessoas mentoras (profissionais da área). Elas poderiam ser pareadas com as estudantes. Cada Sessão tem um conteúdo teórico e com exemplos práticos vivenciados no dia a dia da indústria de desenvolvimento de software.

Lição 4: Pessoas mentoras

Sugere-se que as pessoas mentoras sejam profissionais que trabalham em indústrias de desenvolvimento de software ou pesquisadores de pós-graduação. Essas pessoas vivenciam diariamente o mercado aquecido de tecnologia da informação, estão por dentro das tendências, boas práticas e boas maneiras de trabalho, tecnologias e metodologias de desenvolvimento de software. É necessário que essa profissional tenha interesse em compartilhar conhecimento com pessoas que estão entrando na área.

Lição 5: Desenvolvimento de um projeto real

Ter o contato com *Product Owner*, profissionais da área de engenharia de software, *Pitch, Lean Inception* e Métodos ágeis, possibilitou as estudantes terem uma visão de negócio com olhar mais crítico, descobrindo as tendências para o desenvolvimento de um projeto com um intuito social. É uma imersão temporária na indústria com foco na aceleração de aprendizado prático das estudantes. Todas as estudantes mencionaram que os pontos citados anteriormente foram seus primeiros contatos com essas práticas.

Lição 6: Conhecimento prévio

É necessário que as estudantes tenham conhecimento prévio ou alguma vivência de programação ou lógica de programação para começar no curso de capacitação em engenharia de software. Essa noção agrega muito nas sessões iniciais de fundamentos da computação, matemática e engenharia.

Lição 7: Necessidade da turma

O nivelamento técnico das estudantes ocorre por meio de exercícios práticos de nivelamento técnico de programação e lógica de programação. Com esse nivelamento consegue-se entender qual a real necessidade das estudantes para os avanços dos conteúdos abordados. Dependendo do projeto que vai ser desenvolvido, os conteúdos são iterativos de acordo com a necessidade das estudantes.

5.6.4 Lições aprendidas para pessoas em situação de vulnerabilidade social

Nesta seção serão apresentadas as lições aprendidas relacionadas às pessoas em situação de vulnerabilidade social, durante o desenvolvimento dessa pesquisa.

Lição 1: Mentora 1:1

Todas as pessoas estudantes que se classificaram como vulnerabilidade social mencionaram que a mentoria 1:1 foi essencial para possibilitar momentos de compartilhamento de conhecimento, identificar os erros e acertos no trabalho em equipe, de forma individual entre mentora e estudante. Esse espaço possibilitou as estudantes sentirem-se mais confortáveis para questionar e entender, quando comparado em momentos com toda a turma. Outro ponto em destaque foi o questionário aplicado no primeiro 1:1 pela pessoa mentora com a estudante. Isso ajudou elas trabalharem os pontos nivelados com uma numeração baixa durante os demais encontros e no final, nivelar novamente os mesmos pontos para identificar o quanto a estudante evoluiu no período de aprendizado no curso de capacitação em Engenharia de Software.

Lição 2: Mentora Psicóloga

A mentora na área de psicologia foi responsável pelas dinâmicas para prática profissional e de integração das estudantes. Todas as estudantes mencionaram a importância dessa profissional, durante a pandemia devido à necessidade de contato físico para integrar com facilidade, com as demais estudantes da turma e também nas questões do isolamento social. Momentos como a dinâmica das duplas, um tipo de dinâmica para *Team Building*, executada e planejada a cada 2 semanas, ajudou na união da turma para os pareamentos e no trabalho em equipe.

Lição 3: Programação em pares

Um dos pontos fortes mencionado por todas as pessoas entrevistadas, é a programação em pares. Isso possibilita a troca de conhecimento, possibilidade de errar e aprender com a dupla. O pareamento não é somente na programação, pode ser feito em sessões facilitadas de forma pareada entre uma pessoa mentora e uma estudante, ou duas estudantes, ou duas mentoras. Isso agrega no aprendizado das estudantes.

5.6.5 Limitação da pesquisa

Uma das principais limitações dessa pesquisa refere-se ao número de estudantes entrevistadas - 13 no total e 11 pessoas mentoras dos ciclos (em semestres) que o curso de capacitação em Engenharia de Software foi executado durante a pandemia - 2 no total. Deve-se, entretanto, destacar que especificamente os resultados, principalmente os

da categorização das entrevistas, foram sustentados nos estudos secundário e terciário, o que permite um bom grau de segurança nas conclusões obtidas. Isto também é típico do tipo de pesquisa desenvolvida, exploratória e de base qualitativa, possibilitando o uso de inferências nas conclusões obtidas

6. UM CONJUNTO DE PRÁTICAS DE ENSINO DE ENGENHARIA DE SOFTWARE ONLINE

Neste capítulo apresenta-se um conjunto de práticas para ensinar Engenharia de Software online, proposto. Na seção 6.1 descreve-se o conjunto de práticas de ensino e as suas características identificadas para pessoas em geral e pessoas em situação de vulnerabilidade social. A seção 6.2 apresenta as limitações do conjunto proposto.

Este conjunto tem por objetivo servir de apoio para os cursos de capacitação em Engenharia de Software ensinar ES, de forma online para pessoas em situação de vulnerabilidade social ou não. A indústria de desenvolvimento de software será beneficiada por essa pesquisa, por constatar que a diversidade é necessária em equipes de desenvolvimento de software. O conjunto de práticas foi dimensionado em duas categorias: consequência de utilização das práticas e o indicador de melhoria.

6.1 Conjunto de práticas de ensino de Engenharia de Software online

O conjunto de práticas de ensino de Engenharia de Software foi criado com base nas entrevistas realizadas em pessoas professoras e estudantes do curso de capacitação em Engenharia de Software. Ele é recomendado para atuar como base para os cursos de capacitação em ES. A organização do conjunto foi alicerçada em três estratégias:

1. A partir dos resultados do estudo terciário do MSL sobre ensino de Engenharia de Software, foram identificados que grande parte dos estudos mencionam como base as Diretrizes Curriculares para programas de Graduação em Engenharia de Software ACM / IEEE, em especial as 10 áreas do Conhecimento em Educação em Engenharia de Software - SEEK. As 10 áreas serviram como base para as entrevistas semiestruturadas com professores e estudantes do curso de capacitação em ES e como referência para o conjunto de ensino de Engenharia de Software;

2. Listar as práticas de ensino a partir das técnicas de análise de dados da teoria fundamentada nos dados;

3. Gerar um conjunto de práticas de ensino de Engenharia de Software online para pessoas em situação de vulnerabilidade social.

As Tabela 6.1 apresenta um conjunto de práticas de ensino de Engenharia de Software online. As práticas (IDs) destacadas em vermelho são práticas para pessoas em situação de vulnerabilidade social neste estudo. A tabela foi organizada de acordo com as 10 áreas de conhecimento em educação em Engenharia de Software - SEEK. A seguir são

apresentadas, em detalhes, as práticas obtidas a partir das três estratégias. Elas serão organizadas por áreas do conhecimento em Educação Engenharia de Software: (Fundamentos da Computação - FC; Fundamentos da Matemática e Engenharia - FME; Prática Profissional - PRP; Modelagem e Análise de Software - MASO; Análise e Especificação de Requisitos - AER; Design de Software - DS; Verificação e Validação de Software - VVS; Processo de Software - PS; Qualidade de Software - QS; Segurança - SG). As práticas de ensino recomendadas são identificadas pela letra P + número da prática, a descrição de cada uma e a consequência das práticas organizadas a seguir:

Tabela 6.1 – Conjunto de Práticas de ensino de Engenharia de Software

Área SEEK	ID	Prática	Consequência
PRP	**P1**	Team Building	Possibilita a integração com dinâmicas e a aproximação entre as estudantes e professoras
PRP	**P2**	Inserir Reuniões 1:1	São encontros semanais ou quinzenais com uma professora do curso na área de Engenharia de Software (escolhida pelas estudantes), para Feedbacks e acompanhamento individual das estudantes até o final do curso.
FC	P3	Algoritmos e Estrutura de Dados	Possibilita que as estudantes entendam conceitos relacionados a fundamentos da computação para iniciar programação de computadores em exercícios
FME	P4	Nivelar as estudantes tecnicamente	Identifica o nível de conhecimento das estudantes na área de fundamentos matemático. Sugere-se que os exercícios sejam para pessoas iniciantes.
FC + FME	**P5**	*Coding Dojo*	Compartilhamento de conhecimento para o desenvolvimento de exercícios de lógica de programação
PRP	P6	*Lean Inception*	Apresenta o mundo mais próximo da realidade da Indústria de Desenvolvimento de Software para as estudantes de um curso de capacitação em ES.
PRP	P7	Trabalhar com o Product Owner	O contato das estudantes com o dono do produto possibilita criar e validar requisitos do software
PRP	P8	Cerimônias ágeis	são essenciais para as estudantes desenvolverem um projeto de qualidade com acompanhamento dos POs e professoras.

Os indicadores de melhoria de cada prática serão apresentados na sequência.

P1 (*Team Building*): ajuda no processo de relacionamento entre pares ou grupo. Indica-se que as professoras participem desse momento junto com as estudantes. Podem ser feitas a cada 15 dias, rotacionando os grupos ou duplas;

Área SEEK	ID	Prática	Consequência
PRP	**P9**	Programação em pares	Permite os estudantes compartilhar conhecimento entre a sua dupla. Auxilia na geração de ideias e soluções para um determinado problema
MASO	P10	UML	São necessárias para que as estudantes consigam modelar um software. Essa prática deve ser realizada em suporte total das professoras.
AER	P11	*User Stories*	Elas possibilitam o ensino de narrativas curtas, objetivas e com foco na perspectiva do usuário que solicitou a funcionalidade
MASO + AER	**P12**	*Inserir o Kanban*	Auxilia e controla os fluxos de produção (Feito, fazendo e será feito) do projeto em que as estudantes estão realizando no curso
DS	P13	*Prototipar*	Permite que as estudantes consigam avaliar e validar os requisitos com os POs
DS	P14	Identidade Visual do projeto	Auxilia na comunicação ao público, na criação de ideia, valores, propósito e a missão do projeto de software.
DS	P15	Introdução de IHC	Possibilita aos estudantes conhecimentos básicos sobre a interação humano-computador
VVS	P16	Circle CI	Permite as estudantes integrar continuamente os códigos gerados pelas duplas, com o auxilio dos professores
VVS + QS	P17	*Code Review*	A revisão do código permite que as estudantes garantam a qualidade do código, com apoio de professoras da área, compartilhando as melhores práticas de qualidade do código
QS	P18	TDD	Os desenvolvimento guiado a testes possibilita as estudantes escrever os testes antes de escrever os códigos
PS	P19	Ciclo de Vida do Software	Sessões com exemplos práticos de conhecimento teórico sobre os principais ciclos de vida do software
PS	P20	Gerenciamento de Requisitos	Inserção do Backlog no desenvolvimento do projeto
PS	P21	Métodos Ágeis	Desenvolvimento do projeto com métodos ágeis (práticas ágeis e cerimônias ágeis). Sessões teóricas para introdução do assunto com exemplos reais e prática no projeto desenvolvido pelas estudantes
SG	P22	Segurança de Sistemas	Sessões com a introdução sobre segurança de sistemas e melhores práticas

P2 (Inserir reuniões 1:1) : possibilita a evolução da estudante para trabalhar em par ou grupo. Indica-se aplicar o questionário de mentoria sistematizada;

P3 (Algoritmos e estrutura de dados): ajuda no aprendizado da prática e no estabelecimento de padrões de código que ajudará na colaboração entre a turma de estudantes;

P4 (Nivelar as estudantes tecnicamente): possibilita nivelar o conhecimento das estudantes. A partir desse processo pode-se iniciar por exercícios mais avançados em conhecimento e na criação de sessões mais avançadas ou não;

P5 (*Coding Dojo*): instiga o aprendizado em grupo, através do compartilhamento da tela do exercício do professor. A solução do problema é construída em conjunto;

P6 (*Lean Inception*): vivenciar a Seleção de um projeto que será desenvolvido, participar da concepção do produto e entender as necessidades, ajudará no pensamento crítico das estudantes;

P7 (*Product Owner*): vivenciar a seleção de um projeto que será desenvolvido, participar da concepção do produto e entender as necessidades, ajudará no pensamento crítico das estudantes;

P8 (Cerimônias ágeis): sugere-se que as cerimônias sejam facilitadas por professores do curso (na primeira semana). Depois, deve ser pareado com uma estudante até que duas estudantes consigam realizar sozinhas (rotacionando) as reuniões *Dailys, Sprints, ShowCase* de Retrospectivas. Para isso, podem ser utilizadas as plataformas *FunRetro* ou *Miro*;

P9 (Programação em pares): possibilita o aprendizado em trabalho em equipe, na escuta ativa e no respeito com os colegas;

P10 (UML): possibilita o aprendizado do estudante para modelar, ilustrar e documentar e escrever uma ideia software real;

P11 (*User Stories*): sugere-se uma sessão com teoria e prática. Após cada semana, a professora faz um processo de aprendizado prático com uma estudante (rotacionando por semana);

P12 (Inserir *Kanban*): sugere-se a utilização do *Trello* para criar o *Kanban*. A cada *Daily*, uma dupla de estudantes organizam as tarefas. A organização do *Kanban* proporciona autonomia e aprendizado das estudantes;

P13 (Prototipar): para os protótipos pode-se utilizar a plataforma *Figma*. Esse processo instiga o aprendizado, interesse crítico e *feedback* das estudantes no processo de construção do protótipo;

P14 (Identidade visual do projeto): indica-se utilizar a plataforma *Figma* nesse processo de construção. Sessões na plataforma podem ser introduzidas para ideia inicial de fazer, acompanhada de uma professora da área de Design de software ou UX;

P15 (Introdução de IHC): permitem que os estudantes entendam os conceitos relacionados a comunicação dos usuários com computador ou ao contrário. Sugere-se sessões sobre conceitos iniciais, usabilidade, heurísticas de Nielsen [66];

P16 (*Circle CI*): a integração contínua é o aprendizado prático de mesclar as cópias de trabalho de todas as estudantes em uma linha principal compartilhada várias vezes ao dia;

P17 (*Code Review*): aperfeiçoa o conhecimento das estudantes na garantia da qualidade do software, com revisões diárias apoiadas pelas mentoras / professoras engenheiras de software ou Analista de qualidade de software;

P18 (TDD): sugere-se que o processo de TDD deve ser realizado de maneira prática com exemplos reais, apoiadas de uma professora de analista de qualidade de Software;

P19 (Ciclo de vida do software): permite as estudantes, analisar os requisitos, planejar o desenvolvimento do software, implementar as funcionalidades, executar os testes e integrar no ambiente do usuário;

P20 (Gerenciamento de requisitos): a estudante aprende a gerenciar as tarefas com base nas prioridades definidas pelos POs;

P21 (Métodos ágeis): as estudantes vivenciam e aprendem métodos ágeis no desenvolvimento de um produto real de software com mentoria de professoras especialistas na área ou profissionais da indústria de desenvolvimento de software;

P22 (Segurança de sistemas): as pessoas estudantes entendem o conceito sobre a segurança de sistemas, o que deve e o que não deve ser feito.

Para estudantes em situação de vulnerabilidade social, as práticas P1, P2, P5, P9 e P12 são muito importantes para o entrosamento e organização nas atividades de ensino. Além disso, é muito importante identificar se todas as estudantes têm acesso a computador, periféricos de entrada e saída e acesso à internet. Caso contrário, é preciso buscar meios de ajuda ou possibilidade de aquisição deles, para essas estudantes (empréstimos ou doações) para conseguirem realizar o curso de capacitação em Engenharia de Software. Outro ponto importante é o acompanhamento individual com pessoas professoras / mentoras para reuniões 1:1 com as estudantes. Já no primeiro dia de mentoria 1:1, sugere-se que cada mentor tenha de 2 a 3 estudantes para fazer esse acompanhamento. O ideal é que as pessoas estudantes respondam o questionário de mentoria sistematizada (mencionado na prática P2 no conjunto de práticas), nivelando de 1 a 10 (1: nada confortável e 10: muito confortável).

1. Trabalhar de forma 100% online;
2. Comunicar-se em grupo;
3. Programar sozinha;

4. Programar com um par;
5. Errar;
6. Receber *feedback* positivo;
7. Receber *feedback* construtivo;
8. Facilitar sessões de compartilhamento de conhecimento em grupo;
9. Interagir com os POs;
10. Expor sua opinião técnica;

Após o preenchimento desse questionário, a mentoria 1:1 realiza encontros semanais e ou quinzenais, trabalhando os pontos nivelados com a numeração mais baixa, com a estudante. No final do curso, esses pontos devem ser revistos e nivelados novamente para a estudante visualizar sua evolução no curso.

É importante destacar que antes de aplicar as práticas de ensino de Engenharia de Software, sugere-se começar com sessões de conhecimento teórico e exemplos práticos - com 30min a 60min de duração para compartilhar conhecimento para turma de estudantes. Essas sessões devem ser introduzidas sempre antes de cada conteúdo ser abordado de forma prática e facilitada por professoras e/ou estudantes. Esse ponto foi muito mencionado pelas pessoas entrevistadas

Nas entrevistas, um tópico mencionado foi sobre pessoas professoras. Sugere-se que sejam profissionais da indústria de desenvolvimento de software ou pesquisadores acadêmicos que tenham interesse em compartilhar conhecimento de tendências da área, boas práticas e boas maneiras de Engenharia de Software. Podem ser: Gerente de projetos, Analista de qualidade, desenvolvedores, Analista de negócios, Analista de infraestrutura e acadêmicos de pós-graduação em Ciência da Computação entre outros.

As professoras e estudantes mencionaram o quão valioso é o desenvolvimento de um projeto real - permitindo que as estudantes vivenciem todas as fases do desenvolvimento de um projeto real até a entrega ao cliente. Elas aprendem tópicos essenciais em todas as 10 áreas de conhecimento em educação em Engenharia de Software.

Sobre facilitação, as estudantes entrevistadas introduziram que a sugestão de facilitar permite que elas desenvolvam a prática de comunicação, agregando conhecimento com essa prática, para compartilhar o conhecimento por meio de sessões, pareando com professoras ou outras estudantes.

Fornecer e pedir *feedbacks* positivos e construtivos para as estudantes e professoras após sessões, atividades de desenvolvimento do projeto, interações, atitudes e facilitações, ajudam as estudantes e professoras a identificarem pontos que devem ser melhorados ou o que devem continuar fazendo. A cultura de *feedback* efetivo é essencial,

se oferecida adequadamente, tem a capacidade de crescer e desenvolver as pessoas do curso, melhorar os níveis de confiança e comunicação e fortalecer os vínculos entre estudantes e professoras.

Para finalizar, todos os encontros diários de compartilhamento de conhecimento - Sessões, pareamento, Mentoria 1:1, cerimônias ágeis, entre outros, devem ser realizadas em plataformas de videoconferência e compartilhamento de tela, de acesso gratuito para todas as pessoas, incluindo estudantes, professores e POs.

6.2 Limitações do Conjunto Proposto de práticas de ensino de Engenharia de Software Online

O conjunto de práticas de ensino de Engenharia de Software proposto trata-se de uma proposta inicial, portanto não foi possível o aprofundamento na análise empírica devido à pandemia do coronavírus e a restrição de tempo para desenvolver um estudo de mestrado.

7. CONCLUSÃO

Com o avanço da pandemia do coronavírus (COVID-19) [9], pesquisadores da área de Engenharia de Software estão se defrontando com desafios e com a necessidade de propor novas soluções para ensinar Engenharia de Software, devido à quantidade pequena de estudos até então. A educação à distância é uma alternativa ao ensino presencial tradicional [49], ela usa as facilidades da tecnologia da informação para fornecer conteúdo de ensino e conduzir as avaliações necessárias. Este conceito mitiga a necessidade de estudantes e professores estarem fisicamente presentes em um local. No entanto, os benefícios desse conceito foram amplamente percebidos nas novas circunstâncias causadas pela pandemia de COVID-19 [49].

Durante a pandemia, foram identificadas uma grande diferença na realidade de pessoas em situação de vulnerabilidade social, pois inclui fatores de estrutura familiar, moradia, educação e ou financeira, em relação às pessoas que não se enquadram nesses fatores. Com isso, cursos e aperfeiçoamentos são necessários para construir uma profissão [84]. Possibilitar o acesso dessas pessoas em cursos de capacitação, com o acompanhamento necessário e com práticas de ensino que direcionam ao um ambiente de indústria real de desenvolvimento de software, impulsiona a inclusão de diversidade na tecnologia da informação e as vidas dessas pessoas.

O conjunto de práticas para ensinar Engenharia de Software online para pessoas em situação de vulnerabilidade social é uma tentativa de contribuir com educação, diversidade e aspectos humanos em Engenharia de Software. Os resultados encontrados apresentam evidências de que a área de ES necessita de mais pesquisas voltadas para estes novos desafios que o ambiente de ensino de ES, em tempos de pandemia está trazendo. Do ponto de vista científico, a legitimação do conjunto de práticas de ensino é decorrente do processo de pesquisa como um todo, representado no desenho de pesquisa e desdobrado nas diversas fases deste estudo, conforme apresentado no capítulo 3.

7.1 Contribuições da Pesquisa

A principal contribuição desta pesquisa é a proposta de um conjunto de práticas para ensinar Engenharia de Software online para pessoas iniciantes na área e 5 específicas para pessoas em situação de vulnerabilidade social (apresentado no capítulo 7), que contempla evidências da literatura e a experiência de profissionais da indústria de desenvolvimento de software, acadêmicos de pós-graduação em Ciência da Computação e estudantes do curso de capacitação em Engenharia de Software. O conjunto de práticas de ensino soma-se aos resultados empíricos encontrados nas MSLs e no estudo qualitativo. O

objetivo desta pesquisa é que o conjunto proposto contribua como uma base para os cursos de capacitação que ensinam ou desejam ensinar Engenharia de Software.

A questão de pesquisa deste estudo foi respondida ao longo do processo de formulação do conjunto de práticas de ensino proposto, descreveram-se o estado da arte do estudo terciário sobre ensino de Engenharia de Software e do estudo secundário sobre diversidade em Engenharia de Software, apresentadas a MSL executada no capítulo 4. Adicionalmente, os resultados da MSL serviram como base para as entrevistas semiestruturadas. O conjunto de práticas de ensino foi criado de acordo com as experiências dos profissionais da indústria de desenvolvimento de software, acadêmicos de pós-graduação em Ciência da Computação e estudantes do curso de capacitação em Engenharia de Software obtidas nas 24 entrevistas (capítulo 6).

7.2 Trabalhos Futuros

Identifica-se um grande potencial de crescimento nesta linha de pesquisa, onde os pontos fortes envolvem uma parceria estável entre a academia e a indústria, criando condições de experimentação e aprendizagem únicas, decorrentes de uma sinergia positiva entre os parceiros.

Como recomendações para dar continuidade aos trabalhos nesta área de estudos, sugere-se os seguintes tópicos:

1. Fazer revisão sistemática da literatura específica, tendo como base os resultados do MSL sobre diversidade social;
2. Ampliar a pesquisa para outros cursos de capacitação em Engenharia de Software com foco em métodos ágeis;
3. Ampliar a pesquisa para outros cursos de capacitação em Engenharia de Software com foco em métodos ágeis presencial;
4. Avaliação da extensão proposta, por meio de estudo de caso, aplicando-a em outros cursos de capacitação em ES online;
5. Avaliação da extensão proposta, por meio de estudo de caso, aplicando-a em outros cursos de capacitação em ES presencial;
6. Escrita de artigos científicos reportando os resultados da pesquisa.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Alarifi, A.; Zarour, M.; Alomar, N.; Alshaikh, Z.; Alsaleh, M. “Secdep: Software engineering curricula development and evaluation process using swebok”, *Information and Software Technology*, vol. 74, Jun, 2016, pp. 114–126.

[2] Alhammad, M. M.; Moreno, A. M. “Gamification in software engineering education: A systematic mapping”, *Journal of Systems and Software*, vol. 141, Jul, 2018, pp. 131–150.

[3] Allan, G. “A critique of using grounded theory as a research method”, *Electronic Journal of Business Research Methods*, vol. 2–1, Jan, 2003, pp. 1–10.

[4] Altbach, P. G.; Knight, J. “The internationalization of higher education: Motivations and realities”, *Journal of Studies in International Education*, vol. 11–3-4, Set, 2007, pp. 290–305.

[5] Alves, M. T. G.; Soares, J. F. “Medidas de nível socioeconômico em pesquisas sociais: uma aplicação aos dados de uma pesquisa educacional”, *Opinião Pública*, vol. 15–1, Jun, 2009, pp. 1–30.

[6] Angelov, S.; de Beer, P. “Designing and applying an approach to software architecting in agile projects in education”, *Journal of Systems and Software*, vol. 127, Jun, 2017, pp. 78–90.

[7] Ardis, M.; Budgen, D.; Hislop, G. W.; Offutt, J.; Sebern, M.; Visser, W. “Se 2014: Curriculum guidelines for undergraduate degree programs in software engineering”, *Computer*, vol. 48–11, Nov, 2015, pp. 106–109.

[8] Astels, D. “Test driven development: A practical guide”. Nova Jersey, EUA: Prentice Hall Professional Technical Reference, 2003.

[9] Bahasoan, A. N.; Ayuandiani, W.; Mukhram, M.; Rahmat, A. “Effectiveness of online learning in pandemic covid-19”, *International Journal of Science, Technology & Management*, vol. 1–2, Jul, 2020, pp. 100–106.

[10] Baker, A.; Navarro, E. O.; Van Der Hoek, A. “An experimental card game for teaching software engineering”. In: Proceedings of the 16th Conference on Software Engineering Education and Training, 2003, pp. 216–223.

[11] Batista, E. C.; de Matos, L. A. L.; Nascimento, A. B. “A entrevista como técnica de investigação na pesquisa qualitativa”, *Revista Interdisciplinar Científica Aplicada*, vol. 11–3, Jan, 2017, pp. 23–38.

[12] Beckman, K.; Coulter, N.; Khajenoori, S.; Mead, N. R. “Collaborations: closing the industry-academia gap”, *IEEE Software*, vol. 14–6, Nov/Dez, 1997, pp. 49–57.

[13] Bollin, A.; Hochmuller, E.; Samuelis, L. “Teaching software project management using simulations-the ameise environment: from concepts to class room experience”. In: Proceedings of the 25th Conference on Software Engineering Education and Training, 2012, pp. 85–86.

[14] Bourque, P.; Lavoie, J.-M.; Lee, A.; Trudel, S.; Lethbridge, T. C.; et al.. “Guide to the software engineering body of knowledge (swebok) and the software engineering education knowledge (seek)-a preliminary mapping”. In: Proceedings of the 10th International Workshop on Software Technology and Engineering Practice, 2002, pp. 8–8.

[15] Burnett, M.; Stumpf, S.; Macbeth, J.; Makri, S.; Beckwith, L.; Kwan, I.; Peters, A.; Jernigan, W. “Gendermag: A method for evaluating software’s gender inclusiveness”, *Interacting with Computers*, vol. 28–6, Nov, 2016, pp. 760–787.

[16] Calderón, A.; Ruiz, M.; O’Connor, R. V. “A multivocal literature review on serious games for software process standards education”, *Computer Standards & Interfaces*, vol. 57, Mar, 2018, pp. 36–48.

[17] Canal, D. C. G. “Administração em Sistemas de Informação”. São Paulo, BR: Saraiva Educação S.A, 1999.

[18] Caroli, P. “Lean inception”. São Paulo, BR: Caroli.org, 2017.

[19] Carrara, M. “Dificuldade de aprendizagem e vulnerabilidade social sob a percepção da comunidade escolar”, *Universidade do Sul de Santa Catarina. Pós graduação em Educação e Direitos Humanos*, vol. 1, Dez, 2016, pp. 28.

[20] Chang, Y.-J.; Chen, Y.-R.; Wang, F. T.-Y.; Chen, S.-F.; Liao, R.-H. “Enriching service learning by its diversity: Combining university service learning and corporate social responsibility to help the ngos adapt technology to their needs”, *Systemic Practice and Action Research*, vol. 27–2, Abr, 2014, pp. 185–193.

[21] Childs, M. “Towards a patent pool for hiv medicines: the background”, *The Open AIDS Journal*, vol. 4, Jan, 2010, pp. 33.

[22] Cico, O.; Jaccheri, L.; Nguyen-Duc, A.; Zhang, H. “Exploring the intersection between software industry and software engineering education-a systematic mapping of software engineering trends”, *Journal of Systems and Software*, vol. 172, Dez, 2020, pp. 110736.

[23] Claypool, K.; Claypool, M. “Teaching software engineering through game design”. In: Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education, 2005, pp. 123–127.

[24] Coleman, G.; O’Connor, R. “Using grounded theory to understand software process improvement: A study of irish software product companies”, *Information and Software Technology*, vol. 49–6, Jun, 2007, pp. 654–667.

[25] Corbin, J. M. “Basics of qualitative research: Grounded theory procedures and techniques”. Blacksburg, EUA: Sage, 1990.

[26] Dagenais, B.; Ossher, H.; Bellamy, R. K.; Robillard, M. P.; De Vries, J. P. “Moving into a new software project landscape”. In: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering, 2010, pp. 275–284.

[27] De, A.; Drèze, J.; Kumar, A. S. “Public report on basic education in India”. Otawwa, CN: Oxford University Press New Delhi, 1999.

[28] De Jesus, J. G. “Orientações sobre identidade de gênero: conceitos e termos”, *Guia técnico sobre pessoas transexuais, travestis e demais transgêneros, para formadores de opinião*, vol. 2, Dez, 2012, pp. 42.

[29] Deterding, S.; Sicart, M.; Nacke, L.; O’Hara, K.; Dixon, D. “Gamification. using game-design elements in non-gaming contexts”. In: *Extended Abstracts on Human Factors in Computing Systems*, 2 ed., Association for Computing Machinery, 2011, vol. 23, pp. 2425–2428.

[30] Dixon, J.; Belnap, C.; Albrecht, C.; Lee, K. “The importance of soft skills”, *Corporate Finance Review*, vol. 14–6, Jun, 2010, pp. 35.

[31] Dyer Jr, W. G.; Dyer, J. H.; Dyer, W. G. “Team building: Proven strategies for improving team performance”. San Francisco, EUA: John Wiley & Sons, 2013.

[32] Edmundson, A.; Holtkamp, B.; Rivera, E.; Finifter, M.; Mettler, A.; Wagner, D. “An empirical study on the effectiveness of security code review”. In: Proceedings of the 5th International Symposium on Engineering Secure Software and Systems, 2013, pp. 197–212.

[33] Estácio, B.; Prikladnicki, R.; Morá, M.; Notari, G.; Caroli, P.; Olchik, A. “Software kaizen: Using agile to form high-performance software development teams”. In: Proceedings of the Agile Conference, 2014, pp. 1–10.

[34] Filippova, A.; Trainer, E.; Herbsleb, J. D. “From diversity by numbers to diversity as process: supporting inclusiveness in software development teams with

brainstorming". In: Proceedings of the 39th International Conference on Software Engineering, 2017, pp. 152–163.

[35] Fleury, M. T. L. "Gerenciando a diversidade cultural: experiências de empresas brasileiras", *Revista de Administração de Empresas*, vol. 40–3, Abr, 2000, pp. 18–25.

[36] Flick, U. "Introdução à pesquisa qualitativa-3". Porto Alegre, BR: Artmed editora, 2008.

[37] Garcia, I.; Pacheco, C.; Méndez, F.; Calvo-Manzano, J. A. "The effects of game-based learning in the acquisition of "soft skills" on undergraduate software engineering courses: A systematic literature review", *Computer Applications in Engineering Education*, vol. 28–5, Jul, 2020, pp. 1327–1354.

[38] Garousi, V.; Giray, G.; Tuzun, E.; Catal, C.; Felderer, M. "Closing the gap between software engineering education and industrial needs", *IEEE Software*, vol. 37–2, Mar, 2019, pp. 68–77.

[39] Garousi, V.; Giray, G.; Tuzun, E.; Catal, C.; Felderer, M. "Aligning software engineering education with industrial needs: A meta-analysis", *Journal of Systems and Software*, vol. 156, Out, 2019, pp. 65–83.

[40] Glaser, B. "Theoretical sensitivity", *Advances in the Methodology of Grounded Theory*, vol. 2, Jun, 1978, pp. 164.

[41] Glaser, B. G. "Basics of grounded theory analysis: Emergence vs forcing". Nova York, EUA: Sociology Press, 1992.

[42] Glaser, B. G.; Holton, J. "Remodeling grounded theory". In: Proceedings of the Qualitative Social Research, 2004, pp. 27.

[43] Gnatz, M.; Kof, L.; Prilmeier, F.; Seifert, T. "A practical approach of teaching software engineering". In: Proceedings of the 16th Conference on Software Engineering Education and Training, 2003, pp. 120–128.

[44] Goold, A.; Horan, P. "Foundation software engineering practices for capstone projects and beyond". In: Proceedings of the 15th Conference on Software Engineering Education and Training, 2002, pp. 140–146.

[45] Hill, C. "Socio-economic status and computer use: Designing software that supports low-income users". In: Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing, 2016, pp. 266–267.

[46] Hoda, R.; Noble, J.; Marshall, S. “Developing a grounded theory to explain the practices of self-organizing agile teams”, *Empirical Software Engineering*, vol. 17–6, Abr, 2012, pp. 609–639.

[47] Hoepfl, M. C.; et al.. “Choosing qualitative research: A primer for technology education researchers”, *Journal of Technology Education*, vol. 9, Jan, 1997, pp. 47–63.

[48] Johnson, H. A. “Trello”, *Journal of the Medical Library Association*, vol. 105–2, Abr, 2017, pp. 209.

[49] Kanij, T.; Grundy, J. “Adapting teaching of a software engineering service course due to covid-19”. In: Proceedings of the 32nd Conference on Software Engineering Education and Training, 2020, pp. 1–6.

[50] Kitchenham, B.; Brereton, O. P.; Budgen, D.; Turner, M.; Bailey, J.; Linkman, S. “Systematic literature reviews in software engineering–a systematic literature review”, *Information and Software Technology*, vol. 51–1, Jan, 2009, pp. 7–15.

[51] Kniberg, H.; Skarin, M. “Kanban and Scrum-making the most of both”. Nova York, EUA: Info Q, 2010.

[52] Ktata, O.; Lévesque, G. “Agile development: Issues and avenues requiring a substantial enhancement of the business perspective in large projects”. In: Proceedings of the 2nd Canadian Conference on Computer Science and Software Engineering, 2009, pp. 59–66.

[53] LeBlanc, R. J.; Sobel, A.; Diaz-Herrera, J. L.; Hilburn, T. B.; et al.. “Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering”. Keele University, UK: IEEE Computer Society, 2006.

[54] Lei, S. “Towards programming for the non-technical”. In: Proceedings of the Symposium on Human Centric Computing Languages and Environments, 2003, pp. 291–292.

[55] Marques, M.; Robledo, J. “What software engineering “best practices” are we teaching students-a systematic literature review”. In: Proceedings of the IEEE Frontiers in Education Conference, 2018, pp. 1–8.

[56] Marques, M. R.; Quispe, A.; Ochoa, S. F. “A systematic mapping study on practical approaches to teaching software engineering”. In: Proceedings of the IEEE Frontiers in Education Conference, 2014, pp. 1–8.

[57] Martin, A.; Biddle, R.; Noble, J. “Xp customer practices: A grounded theory”. In: Proceedings of the Agile Conference, 2009, pp. 33–40.

[58] Matturro, G. “Soft skills in software engineering: A study of its demand by software companies in uruguay”. In: Proceedings of the 6th international workshop on cooperative and human aspects of software engineering, 2013, pp. 133–136.

[59] Maurício, R. d. A.; Veado, L.; Moreira, R. T.; Figueiredo, E.; Costa, H. “A systematic mapping study on game-related methods for software engineering education”, *Information and Software Technology*, vol. 95, Mar, 2018, pp. 201–218.

[60] Menezes, Á.; Prikladnicki, R. “Diversity in software engineering”. In: Proceedings of the 11th International Workshop on Cooperative and Human Aspects of Software Engineering, 2018, pp. 45–48.

[61] Miyashita, Y.; Tanaka, T.; Hazeyama, A. “Systematic literature review regarding communication support in project-based learning of software development”. In: Proceedings of the 42nd Annual Computer Software and Applications Conference, 2018, pp. 781–782.

[62] Monteiro, S. R. d. R. P. “O marco conceitual da vulnerabilidade social”, *Sociedade em Debate*, vol. 17–2, Abr, 2011, pp. 29–40.

[63] Moreno, A. M.; Sanchez-Segura, M.-I.; Medina-Dominguez, F.; Carvajal, L. “Balancing software engineering education and industrial needs”, *Journal of Systems and Software*, vol. 85–7, Jul, 2012, pp. 1607–1620.

[64] Moutafidou, A.; Bratitsis, T. “Digital storytelling: giving voice to socially excluded people in various contexts”. In: Proceedings of the 8th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-exclusion, 2018, pp. 219–226.

[65] Mujtaba, B. “Workforce diversity management: Challenges, competencies and strategies”. Florida, EUA: Llumina Press, 2007.

[66] Nielsen, J.; Molich, R. “Heuristic evaluation of user interfaces”. In: Proceedings of the SIGCHI conference on Human factors in computing systems, 1990, pp. 249–256.

[67] Ouhbi, S.; Pombo, N. “Software engineering education: Challenges and perspectives”. In: Proceedings of the IEEE Global Engineering Education Conference, 2020, pp. 202–209.

[68] Page, S. E. “The difference: How the power of diversity creates better groups, firms, schools, and societies-new edition”. Cambridge, UK: Princeton University Press, 2008.

[69] Parry, K. W. “Grounded theory and social process: A new direction for leadership research”, *The Leadership Quarterly*, vol. 9–1, Mar, 1998, pp. 85–105.

[70] Patrick, H. A.; Kumar, V. R. “Managing workplace diversity: Issues and challenges”, *Sage Open*, vol. 2–2, Abr, 2012, pp. 33–47.

[71] Patton, M. Q. “Qualitative research & evaluation methods: Integrating theory and practice”, *American Journal of Evaluation*, vol. 4, Jan, 2015, pp. 603–605.

[72] Petersen, K.; Feldt, R.; Mujtaba, S.; Mattsson, M. “Systematic mapping studies in software engineering”. In: Proceedings of the 12th international Conference on Evaluation and Assessment in Software Engineering, Jun, 2008, pp. 68–77.

[73] Petersen, K.; Vakkalanka, S.; Kuzniarz, L. “Guidelines for conducting systematic mapping studies in software engineering: An update”, *Information and Software Technology*, vol. 64, Ago, 2015, pp. 1–18.

[74] Petri, G.; von Wangenheim, C. G. “How games for computing education are evaluated? a systematic literature review”, *Computers & Education*, vol. 107, Abr, 2017, pp. 68–90.

[75] Pieterse, V.; Kourie, D. G.; Sonnekus, I. P. “Software engineering team diversity and performance”. In: Proceedings of the Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on IT Research in Developing Countries, 2006, pp. 180–186.

[76] Pombo, N.; Garcia, N.; Alves, P. “How to get a badge? unlock your mind: Motivation through student empowerment”. In: Proceedings of the IEEE Global Engineering Education Conference, 2019, pp. 115–119.

[77] Prati, L. E.; Couto, M. C. P.; Koller, S. H. “Famílias em vulnerabilidade social: rastreamento de termos utilizados por terapeutas de família”, *Psicologia: Teoria e Pesquisa*, vol. 25–3, Set, 2009, pp. 403–408.

[78] Presotti, L. “Gerenciar a diversidade cultural nas organizações: caminhos para a inclusão”, *Universidade de Brasilia*, vol. 1, Dez, 2011, pp. 1–152.

[79] Pressman, R.; Maxim, B. “Engenharia de Software-8ª Edição”. São Paulo, BR: McGraw Hill Brasil, 2016.

[80] Pressman, R. S. “Software engineering: a practitioner’s approach”. São Paulo, BR: Palgrave Macmillan, 2005.

[81] Prikladnicki, R.; Albuquerque, A. B.; von Wangenheim, C. G.; Cabral, R. “Ensino de engenharia de software: desafios, estratégias de ensino e lições aprendidas”, *Fórum de Educação em Engenharia de Software*, vol. 1, Out, 2009, pp. 1–8.

[82] Przyborski, A.; Slunecko, T. “Against reification praxeological methodology and its benefits”. In: *Dynamic Process Methodology in the Social and Developmental Sciences*, 1 ed., Springer USA, 2009, vol. 1, pp. 141–170.

[83] Qadir, Muhammad Manan e Usman, M. “Software engineering curriculum: A systematic mapping study”. In: Proceedings of the 5th Malaysian Conference in Software Engineering, 2011, pp. 269–274.

[84] Raoport, A.; da Silva, S. B. “Desempenho escolar de crianças em situação de vulnerabilidade social”, *Revista a Educação em Rede: Formação e Prática Docente*, vol. 2, Jun, 2013, pp. 2.

[85] Ries, E. “Minimum viable product: a guide”, *Startup Lessons Learned*, vol. 3, Ago, 2009, pp. 1.

[86] Robbins, S.; Judge, T.; Sobral, F. “Comportamento organizacional: teoria e prática no contexto brasileiro”. São Paulo, BR: Pearson Prentice Hall, 2010.

[87] Romero-Rodríguez, J.-M.; Aznar-Díaz, I.; Hinojo-Lucena, F.-J.; Gómez-García, G. “Mobile learning in higher education: Structural equation model for good teaching practices”, *IEEE Access*, vol. 8, Abr, 2020, pp. 91761–91769.

[88] Santana, D. “What makes a latino, hispanic or latinx?”, *Embracing Diversity*, vol. 7, Jul, 2017, pp. 33.

[89] Sedano, T.; Ralph, P.; Péraire, C. “The product backlog”. In: Proceedings of the 41st International Conference on Software Engineering, 2019, pp. 200–211.

[90] Sharp, H.; Robinson, H. “Some social factors of software engineering: The maverick, community and technical practices”. In: Proceedings Workshop on Human and Social Factors of Software Engineering, 2005, pp. 1–6.

[91] Silveira, F. E. d. “Relato de experiência do uso de programação em pares no desenvolvimento de sistemas da UFRGS”. In: Proceedings of the Information and Communication Technology Workshop of Federal Institutions of Higher Education, 2016, pp. 4.

[92] Silveira, K. K.; Prikladnicki, R. “A systematic mapping study of diversity in software engineering: a perspective from the agile methodologies”. In: Proceedings of the 12th International Workshop on Cooperative and Human Aspects of Software Engineering, 2019, pp. 7–10.

[93] Sommerville, I. “Software engineering”. Nova York, EUA: Addison-Wesley/Pearson, 2011.

[94] Sommerville, I. “Engenharia de software, 9a”, *São Paulo, BR*, vol. 137035152, Jul, 2011, pp. 18.

[95] Steglich, C.; Lisboa, A.; Prikladnicki, R.; Marczak, S.; da Costa Móra, M.; Olchik, A.; Heck, N.; Rachid, Y.; Ghidorsi, G. “Agile accelerator program: From industry-academia collaboration to effective agile training”. In: Proceedings of the 34th Brasilian Symposium on Software Engineering, 2020, pp. 21–30.

[96] Stokes, E. K.; Zambrano, L. D.; Anderson, K. N.; Marder, E. P.; Raz, K. M.; Felix, S. E. B.; Tie, Y.; Fullerton, K. E. “Coronavirus disease 2019 case surveillance—united states”, *Morbidity and Mortality Weekly Report*, vol. 69–24, Jun, 2020, pp. 759.

[97] Taylor, V.; Ladner, R. “Data trends on minorities and people with disabilities in computing”, *Communications of the ACM*, vol. 54–12, Dez, 2011, pp. 34–37.

[98] Tietze, F.; Vimalnath, P.; Aristodemou, L.; Molloy, J. “Crisis-critical intellectual property: Findings from the covid-19 pandemic”, *Available at SSRN*, vol. 1, Jun, 2020, pp. 18.

[99] Torres, C. V.; Pérez-Nebra, A. “Diversidade e inclusão nas organizações”, *Psicologia, Organizações e Trabalho no Brasil*, vol. 2, Jan, 2014, pp. 526–546.

[100] Vasilescu, B.; Filkov, V.; Serebrenik, A. “Perceptions of diversity on git hub: A user survey”. In: Proceedings of the 8th International Workshop on Cooperative and Human Aspects of Software Engineering, 2015, pp. 50–56.

[101] Vasilescu, B.; Posnett, D.; Ray, B.; van den Brand, M. G.; Serebrenik, A.; Devanbu, P.; Filkov, V. “Gender and tenure diversity in github teams”. In: Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, 2015, pp. 3789–3798.

[102] Vasilescu, B.; Serebrenik, A.; Filkov, V. “A data set for social diversity studies of github teams”. In: Proceedings of the 12th Working Conference on Mining Software Repositories, 2015, pp. 514–517.

[103] Von Wangenheim, C. G.; Shull, F. “To game or not to game?”, *IEEE Software*, vol. 26–2, Fev, 2009, pp. 92–94.

[104] Whittle, J. “Is your software valueless?”, *IEEE Software*, vol. 36–3, Jun, 2019, pp. 112–115.

[105] Whitworth, E.; Biddle, R. “The social nature of agile teams”. In: Proceedings of the Agile Conference, 2007, pp. 26–36.

[106] Wolf, M. “Embedded software in crisis”, *Computer*, vol. 49–1, Jan, 2016, pp. 88–90.

[107] Woszczynski, A.; Beise, C.; Myers, M.; Moody, J. “Diversity and the information technology workforce: an examination of student perceptions”. In: Proceedings of the 3th Conference on Computer Personnel Research: Freedom in Philadelphia–Leveraging Differences and Diversity in the IT Workforce, 2003, pp. 117–122.

[108] Zambon, C.; Thiry, M. “Ludic practices to support the development of software engineering educational games: A systematic review”. In: Proceedings of the XLIV Latin American Computer Conference, 2018, pp. 794–802.

[109] Zieris, F. “Qualitative analysis of knowledge transfer in pair programming”. In: Proceedings of the 37th IEEE International Conference on Software Engineering, 2015, pp. 855–858.

APÊNDICE A – REFERÊNCIAS DO MSL SOBRE DIVERSIDADE EM ENGENHARIA DE SOFTWARE

A1	Choi, K. S. (2015). A comparative analysis of different gender pair combinations in pair programming. Behaviour & Information Technology, 34(8), 825-837.
A2	Hamdan, K., Belkhouche, B., & Smith, P. (2018, June). A Culture-Based Profile Model of Software Evaluators. In International Conference on Information Science and Applications (pp. 559-570). Springer, Singapore.
A3	Vasilescu, B., Serebrenik, A., & Filkov, V. (2015, May). A data set for social diversity studies of GitHub teams. In 2015 IEEE/ACM 12th working conference on mining software repositories (pp. 514-517). IEEE.
A4	Allhutter, D. (2010, July). A Deconstructivist Methodology for Software Engineering. In ENASE (pp. 207-213).
A5	Hang, K., Ostrander, T., Orr, A., Archer, J., & Uland, S. (2020, February). A Five -Year Retrospective on an Applied Baccalaureate Program in Software Development. In Proceedings of the 51st ACM Technical Symposium on Computer Science Education (pp. 894-898).
A6	Rajagopalan, S., & Rajamani, L. (2013, December). A Fuzzy Logic Rule Based Forecasting Model: Work-Life Balance in IT among Software vs. Services Industry on the View of Women Software Engineer. In 2013 International Conference on Machine Intelligence and Research Advancement (pp. 241-246). IEEE.
A7	Kanij, T., Merkel, R., & Grundy, J. (2011, September). A preliminary study on factors affecting software testing team performance. In 2011 International Symposium on Empirical Software Engineering and Measurement (pp. 359-362). IEEE.
A8	Janzen, D. S., Bahrami, S., da Silva, B. C., & Falessi, D. (2018, October). A reflection on diversity and inclusivity efforts in a software engineering program. In 2018 IEEE Frontiers in Education Conference (FIE) (pp. 1-9). IEEE.
A9	AYDEMIR, Fatma Basak; DALPIAZ, Fabiano. A roadmap for ethics-aware software engineering. In: 2018 IEEE/ACM International Workshop on Software Fairness (FairWare). IEEE, 2018. p. 15-21.
A10	Gilal, A. R., Jaafar, J., Omar, M., Basri, S., & Waqas, A. (2016). A rule-based model for software development team composition: Team leader role with personality types and gender classification. Information and Software Technology, 74, 105-113.
A11	Gilal, A. R., Jaafar, J., Omar, M., Basri, S., & Aziz, I. D. A. (2019). A set of rules for constructing gender-based personality types' composition for software programmer. In Proceedings of the International Conference on Data Engineering 2015 (DaEng-2015) (pp. 363-374). Springer, Singapore.
A12	Mead, N. R., Drommi, A., Shoemaker, D., & Ingalsbe, J. (2009, July). A study of the impact on students understanding cross cultural differences in software engineering work. In 2009 33rd Annual IEEE International Computer Software and Applications Conference (Vol. 1, pp. 644-645). IEEE.

A13	Narasimhan, V. L. (2011, June). A subjective perspective on genderization issues in software development life cycle. In 2011 International Conference on Recent Trends in Information Technology (ICRTIT) (pp. 1335-1340). IEEE.
A14	Silveira, K. K., & Prikladnicki, R. (2019, May). A systematic mapping study of diversity in software engineering: a perspective from the agile methodologies. In 2019 IEEE/ACM 12th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE) (pp. 7-10). IEEE.
A15	Garcia-Holgado, A., Vázquez-Ingelmo, A., Verdugo-Castro, S., González, C., Gómez, M. C. S., & Garcia-Peñalvo, F. J. (2019, April). Actions to promote diversity in engineering studies: a case study in a Computer Science Degree. In 2019 IEEE Global Engineering Education Conference (EDUCON) (pp. 793-800). IEEE.
A16	Ozawa, H., & Zhang, L. (2013, August). Adapting agile methodology to overcome social differences in project members. In 2013 Agile Conference (pp. 82-87). IEEE.
A17	Leung, W. S. (2017, May). African ethics for enhancing soft skills in young IT professionals in Southern Africa. In 2017 IST-Africa Week Conference (IST-Africa) (pp. 1-9). IEEE.
A18	Schloegel, U., Stegmann, S., Maedche, A., & Van Dick, R. (2018). Age stereotypes in agile software development–an empirical study of performance expectations. Information Technology & People.
A19	Judy, K. H. (2012, January). Agile values, innovation and the shortage of women software developers. In 2012 45th Hawaii International Conference on System Sciences (pp. 5279-5288). IEEE.
A20	Altiner, S., & Ayhan, M. B. (2018). An approach for the determination and correlation of diversity and efficiency of software development teams. South African Journal of Science, 114(3-4), 1-9.
A21	Damian, D. E., & Zowghi, D. (2003, January). An insight into the interplay between culture, conflict and distance in globally distributed requirements negotiations. In Proceedings of the 36th Annual Hawaii International Conference on System Sciences, HICSS 2003. HICSS 2003.
A22	Fernandez-Sanz, L., & Misra, S. (2012). Analysis of cultural and gender influences on teamwork performance for software requirements analysis in multinational environments. IET software, 6(3), 167-175.
A23	Czekster, R. M., Fernandes, P., Sales, A., & Webber, T. (2010, August). Analytical modeling of software development teams in globally distributed projects. In 2010 5th IEEE International Conference on Global Software Engineering (pp. 287-296). IEEE.
A24	MAZNI, Omar; SYED-ABDULLAH, Sharifah-Lailee; HUSSIN, Naimah Mohd. Analyzing personality types to predict team performance. In: 2010 International Conference on Science and Social Research (CSSR 2010). IEEE, 2010. p. 624-628.

A25	Brinkman, B., & Diekman, A. (2016, February). Applying the communal goal congruity perspective to enhance diversity and inclusion in undergraduate computing degrees. In Proceedings of the 47th ACM technical symposium on computing science education (pp. 102-107).
A26	Dattero, R., & Galup, S. D. (2005). Assessing gender differences in software developers using the human capital model. Information Resources Management Journal (IRMJ), 18(3), 68-87.
A27	Dias Canedo, E., Acco Tives, H., Bogo Marioti, M., Fagundes, F., & Siqueira de Cerqueira, J. A. (2019). Barriers Faced by Women in Software Development Projects. Information, 10(10), 309.
A28	Borsotti, V. (2018, May). SIGSOFT Distinguished Paper-Barriers to Gender Diversity in Software Development Education: Actionable Insights from a Danish Case Study. In 2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET) (pp. 146-152). IEEE.
A29	Weng, J., & Murphy, C. (2018, February). Bridging the Diversity Gap in Computer Science with a Course on Open Source Software. In 2018 Research on Equity and Sustained Participation in Engineering, Computing, and Technology (RESPECT) (pp. 1-4). IEEE.
A30	Quinones, P. A., Fussell, S. R., Soibelman, L., & Akinci, B. (2009, February). Bridging the gap: discovering mental models in globally collaborative contexts. In Proceedings of the 2009 international workshop on Intercultural collaboration (pp. 101-110).
A31	Hazzan, O., & Dubinsky, Y. (2006, May). Can diversity in global software development be enhanced by agile software development?. In Proceedings of the 2006 international workshop on Global software development for the practitioner (pp. 58-61).
A32	Grigoreanu, V., Cao, J., Kulesza, T., Bogart, C., Rector, K., Burnett, M., & Wiedenbeck, S. (2008, September). Can feature design reduce the gender gap in end-user software development environments?. In 2008 IEEE Symposium on Visual Languages and Human-Centric Computing (pp. 149-156). IEEE.
A33	Grigoreanu, V., Cao, J., Kulesza, T., Bogart, C., Rector, K., Burnett, M., & Wiedenbeck, S. (2008, September). Can feature design reduce the gender gap in end-user software development environments?. In 2008 IEEE Symposium on Visual Languages and Human-Centric Computing (pp. 149-156). IEEE.
A34	Wurzelová, P., Palomba, F., & Bacchelli, A. (2019, May). Characterizing women (not) contributing to open-source. In 2019 IEEE/ACM 2nd International Workshop on Gender Equality in Software Engineering (GE) (pp. 5-8). IEEE.
A35	Jaakkola, H., Henno, J., Thalheim, B., & Mäkelä, J. (2015, May). Collaboration, distribution and culture-challenges for communication. In 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO) (pp. 657-664). IEEE.
A36	Wang, Z., Wang, Y., & Redmiles, D. (2018, May). Competence-confidence gap: A threat to female developers' contribution on github. In 2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS) (pp. 81-90). IEEE.

A37	Taylor, W. A. (2004). Computer-mediated knowledge sharing and individual user differences: an exploratory study. European journal of information systems, 13(1), 52-64.
A38	Silveira, K. K., Musse, S., Manssour, I. H., Vieira, R., & Prikladnicki, R. (2019, May). Confidence in programming skills: gender insights from StackOverflow developers survey. In 2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion) (pp. 234-235). IEEE.
A39	Teka, D., Dittrich, Y., & Kifle, M. (2017, September). Contextualizing user centered design with agile methods in Ethiopia. In 2017 IEEE AFRICON (pp. 911-916). IEEE.
A40	Kofink, A. (2015, October). Contributions of the under-appreciated: Gender bias in an open-source ecology. In Companion Proceedings of the 2015 ACM SIGPLAN International Conference on Systems, Programming, Languages and Applications: Software for Humanity (pp. 83-84).
A41	Szabó, D. M., & Steghöfer, J. P. (2019, May). Coping strategies for temporal, geographical and sociocultural distances in agile GSD: a case study. In 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP) (pp. 161-170). IEEE.
A42	Avritzer, A. (2006, October). Coping with cultural diversity in GSE environments. In 2006 IEEE International Conference on Global Software Engineering (ICGSE'06) (pp. 199-199). IEEE.
A43	Wang, Y., & Zhang, M. (2019, May). Country stereotypes, initial trust, and cooperation in global software development teams. In 2019 ACM/IEEE 14th International Conference on Global Software Engineering (ICGSE) (pp. 152-161). IEEE.
A44	Huang, H., & Trauth, E. M. (2008). Cultural diversity challenges: issues for managing globally distributed knowledge workers in software development. In Global Information Technologies: Concepts, Methodologies, Tools, and Applications (pp. 2677-2683). IGI Global.
A45	Wong, B., & Hasan, S. (2008, May). Cultural influences and differences in software process improvement programs. In Proceedings of the 6th international workshop on Software quality (pp. 3-10).
A46	Nonoyama, T., Wen, L., Rout, T., & Tuffley, D. (2017, October). Cultural issues and impacts of software process in very small entities (VSEs). In International Conference on Software Process Improvement and Capability Determination (pp. 70-81). Springer, Cham.
A47	Hsieh, Y. (2006, October). Culture and shared understanding in distributed requirements engineering. In 2006 IEEE International Conference on Global Software Engineering (ICGSE'06) (pp. 101-108). IEEE.
A48	Hodgson, A., Hubbard, E. M., & Siemieniuch, C. E. (2011, June). Culture and the performance of teams in complex systems. In 2011 6th International Conference on System of Systems Engineering (pp. 95-100). IEEE.

A49	Casado-Lumbreras, C., Colomo-Palacios, R., Soto-Acosta, P., & Misra, S. (2011). Culture dimensions in software development industry: The effects of mentoring.
A50	Deshpande, S., Richardson, I., Casey, V., & Beecham, S. (2010, August). Culture in global software development-a weakness or strength?. In 2010 5th IEEE International Conference on Global Software Engineering (pp. 67-76). IEEE.
A51	Allhutter, D., & Hofmann, R. (2012). Deconstructive design as an approach for opening trading zones. In Computer Engineering: Concepts, Methodologies, Tools and Applications (pp. 394-411). IGI Global.
A52	Ruiz Ben, E. (2007). Defining expertise in software development while doing gender. Gender, Work & Organization, 14(4), 312-332.
A53	Developing diversity awareness of software engineers: A Diversity Framework and its application in an academic and life-long learning context
A54	Moutafidou, A., & Bratitsis, T. (2018, June). Digital storytelling: giving voice to socially excluded people in various contexts. In Proceedings of the 8th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-exclusion (pp. 219-226).
A55	Robson, N. (2018, October). Diversity and decorum in open source communities. In Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (pp. 986-987).
A56	Barke, H. (2018, May). How many story points for diversity? estimation as a chance for diversity reflexion. In Proceedings of the 4th Conference on Gender & IT (pp. 221-223).
A57	Menezes, Á., & Prikladnicki, R. (2018, maio). Diversidade em engenharia de software. In Proceedings of the 11th International Workshop on Cooperative and Human Aspects of Software Engineering (pp. 45-48).
A58	Wickramasinghe, V., & Nandula, S. (2015). Diversity in team composition, relationship conflict and team leader support on globally distributed virtual software development team performance. Strategic Outsourcing: An International Journal.
A59	Chidambaram, L. (2005, January). Diversity: Is there more than meets the eye? A longitudinal study of the impact of technology support on teams with differing diversity. In Proceedings of the 38th Annual Hawaii International Conference on System Sciences (pp. 48a-48a). IEEE.
A60	Bosch, N. V., Ramos, A. M. G., & Samaranch, E. A. (2014, September). Doing and undoing genders and information and communication technologies. In Proceedings of the XV International Conference on Human Computer Interaction (pp. 1-2).

A61	Bosch, N. V., Ramos, A. M. G., & Samaranch, E. A. (2014, September). Doing and undoing genders and information and communication technologies. In Proceedings of the XV International Conference on Human Computer Interaction (pp. 1-2).
A62	Colomo-Palacios, R., Mishra, A., Casado-Lumbreras, C., & Soto-Acosta, P. (2013). E-Mentoring in Global Software Development Teams: Success Factors to Develop a Common Culture. In Multiculturalism in Technology-Based Education: Case Studies on ICT-Supported Approaches (pp. 160-175). IGI Global.
A63	Kross, S., & Guo, P. J. (2019, October). End-user programmers repurposing end-user programming tools to foster diversity in adult end-user programming education. In 2019 IEEE Symposium on Visual Languages and Human-Centric]Computing (VL/HCC) (pp. 65-74). IEEE.
A64	Fowler, A., & Schreiber, I. (2017, February). Engaging under-represented minorities in STEM through game jams. In Proceedings of the second international conference on game jams, hackathons, and game creation events (pp. 1-5).
A65	Congalton, J. (2014). A mixed methods investigation of ethnic diversity and productivity in software development teams: a thesis presented in partial fulfilment of the requirements for the degree of Doctorate of Philosophy in Information Systems at Massey University, Wellington, New Zealand (Doctoral dissertation, Massey University).
A66	Chang, Y. J., Chen, Y. R., Wang, F. T. Y., Chen, S. F., & Liao, R. H. (2014). Enriching service learning by its diversity: Combining university service learning and corporate social responsibility to help the NGOs adapt technology to their needs. Systemic Practice and Action Research, 27(2), 185-193.
A67	Khaled, R. (2011, September). Equality= inequality: probing equality-centric design and development methodologies. In IFIP Conference on Human-Computer Interaction (pp. 405-421). Springer, Berlin, Heidelberg.
A68	Al Hinai, M. A., & Chitchyan, R. (2019). Equality Requirements for Software Systems: A Survey. In ICT4S.
A69	El-Bahey, R., & Zeid, A. (2013, July). Establishing a globally distributed software development system in academic settings: An ergonomic perspective. In IEEE International Professonal Communication 2013 Conference (pp. 1-8). IEEE.
A70	Paul, R., Bosu, A., & Sultana, K. Z. (2019, February). Expressions of sentiments during code reviews: Male vs. female. In 2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER) (pp. 26-37). IEEE.
A71	Seibel, S., & Veilleux, N. (2019). Factors influencing women entering the software development field through coding bootcamps vs. computer science bachelor's degrees. Journal of Computing Sciences in Colleges, 34(6), 84-96.
A72	Nguyen-Duc, A., Khodambashi, S., Gulla, J. A., Krogstie, J., & Abrahamsson, P. (2018). Female Leadership in Software Projects—A Preliminary Result on Leadership Style and Project Context Factors. In Towards a Synergistic Combination of Research and Practice in Software Engineering (pp. 149-163). Springer, Cham.

A73	Razavian, M., & Lago, P. (2015). Feminine expertise in architecting teams. IEEE Software, 33(4), 64-71.
A74	Lee, A., & Carver, J. C. (2019, May). FLOSS participants' perceptions about gender and inclusiveness: a survey. In 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE) (pp. 677-687). IEEE.
A75	Filippova, A., Trainer, E., & Herbsleb, J. D. (2017, May). From diversity by numbers to diversity as process: supporting inclusiveness in software development teams with brainstorming. In 2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE) (pp. 152-163). IEEE.
A76	Tassabehji, R., Harding, N., Lee, H., & Dominguez-Pery, C. (2020). From female computers to male comput rs: Or why there are so few women writing algorithms and developing software. Human Relations, 0018726720914723.
A77	Ihsen, S., Jeschke, S., Wilke, M., & Buschmeyer, A. (2007). GENDER AND DIVERSITY IN EDUCATIONAL PROGRAMS FOR COMPUTER SCIENTISTS AND ENGINEERS.
A78	Chinn, D., & VanDeGrift, T. (2008, September). Gender and diversity in hiring software professionals: what do students say?. In Proceedings of the Fourth International Workshop on Computing Education Research (pp. 39-50).
A79	Vasilescu, B., Posnett, D., Ray, B., van den Brand, M. G., Serebrenik, A., Devanbu, P., & Filkov, V. (2015, April). Gender and tenure diversity in GitHub teams. In Proceedings of the 33rd annual ACM conference on human factors in computing systems (pp. 3789-3798).
A80	Leavy, S. (2018, May). Gender bias in artificial intelligence: The need for diversity and gender theory in machine learning. In Proceedings of the 1st international workshop on gender equality in software engineering (pp. 14-16).
A81	Terrell, J., Kofink, A., Middleton, J., Rainear, C., Murphy-Hill, E., Parnin, C., & Stallings, J. (2017). Gender differences and bias in open source: Pull request acceptance of women versus men. PeerJ Computer Science, 3, e111.
A82	Kuechler, V., Gilbertson, C., & Jensen, C. (2012, September). Gender differences in early free and open source software joining process. In IFIP International Conference on Open Source Systems (pp. 78-93). Springer, Berlin, Heidelberg.
A83	Bastarrica, M. C., & Simmonds, J. (2019, May). Gender differences in self and peer assessment in a software engineering capstone course. In 2019 IEEE/ACM 2nd International Workshop on Gender Equality in Software Engineering (GE) (pp. 29-32). IEEE.
A84	Catolino, G., Palomba, F., Tamburri, D. A., Serebrenik, A., & Ferrucci, F. (2019). Gender diversity and community smells: insights from the trenches. IEEE Software, 37(1), 10-16.

A85	Catolino, G., Palomba, F., Tamburri, D. A., Serebrenik, A., & Ferrucci, F. (2019, May). Gender diversity and women in software teams: How do they affect community smells?. In 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS) (pp. 11-20). IEEE.
A86	Hamilton, M., Luxton-Reilly, A., Augar, N., Chiprianov, V., Gutierrez, E. C., Duarte, E. V., ... & Wong, E. (2016). Gender equity in computing: International faculty perceptions and current practices. In Proceedings of the 2016 ITiCSE Working Group Reports (pp. 81-102).
A87	Widder, D. G. (2019, October). Gender in Open Source Communities: Different Migration Patterns and Forms of Work. In 2019 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC) (pp. 241-242). IEEE.
A88	Mendez, C., Sarma, A., & Burnett, M. (2018, May). Gender in open source software: what the tools tell. In Proceedings of the 1st International Workshop on Gender Equality in Software Engineering (pp. 21-24).
A89	Schelhowe, H. (2005, June). Gender questions and computing science. In Proceedings of the international symposium on Women and ICT: creating global transformation (pp. 10-es).
A90	Vedres, B., & Vasarhelyi, O. (2019). Gendered behavior as a disadvantage in open source software development. EPJ Data Science, 8(1), 25.
A91	Lin, Y. W., & den Besten, M. (2019). Gendered work culture in free/libre open source software development. Gender, Work & Organization, 26(7), 1017-1031.
A92	Hill, C., Ernst, S., Oleson, A., Horvath, A., & Burnett, M. (2016, September). GenderMag experiences in the field: The whole, the parts, and the workload. In 2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC) (pp. 199-207). IEEE.
A93	Garrison, G., Wakefield, R. L., Xu, X., & Kim, S. H. (2010). Globally distributed teams: The effect of diversity on trust, cohesion and individual performance. ACM SIGMIS Database: the database for Advances in Information Systems, 41(3), 27-48.
A94	Qiu, H. S., Nolte, A., Brown, A., Serebrenik, A., & Vasilescu, B. (2019, May). Going farther together: The impact of social capital on sustained participation in open source. In 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE) (pp. 688-699). IEEE.
A95	Kopeć, W., Nielek, R., & Wierzbicki, A. (2018, May). Guidelines towards better participation of older adults in software development processes using a new SPIRAL method and participatory approach. In Proceedings of the 11th International Workshop on Cooperative and Human Aspects of Software Engineering (pp. 49-56).
A96	Irrgang, M. (2018, May). Hands-on participatory and interdisciplinary design in computer science: an example. In Proceedings of the 4th Conference on Gender & IT (pp. 29-33).

A97	Tapia, A. H. (2003, April). Hostile_Work_Environment. com. In Proceedings of the 2003 SIGMIS conference on Computer personnel research: Freedom in Philadelphia–leveraging differences and diversity in the IT workforce (pp. 64-67).
A98	Morgan, S. (2017, October). How are programming questions from women received on stack overflow? a case study of peer parity. In Proceedings Companion of the 2017 ACM SIGPLAN International Conference on Systems, Programming, Languages, and Applications: Software for Humanity (pp. 39-41).
A99	Pieterse, V., Leeu, M., & van Eekelen, M. (2018, March). How personality diversity influences team performance in student software engineering teams. In 2018 Conference on Information Communications Technology and Society (ICTAS) (pp. 1-6). IEEE.
A100	Ford, D., Milewicz, R., & Serebrenik, A. (2019, May). How remote work can foster a more inclusive environment for transgender developers. In 2019 IEEE/ACM 2nd International Workshop on Gender Equality in Software Engineering (GE) (pp. 9-12). IEEE.
A101	Sudbery, C. (2016, May). How XP Can Improve the Experiences of Female Software Developers. In International Conference on Agile Software Development (pp. 261-269). Springer, Cham.
A102	Levina, N., & Kane, A. A. (2009, February). Immigrant managers as boundary spanners on offshored software development projects: partners or bosses?. In Proceedings of the 2009 international workshop on Intercultural collaboration (pp. 61-70).
A103	Gila, A. R., Jaafa, J., Omar, M., & Tunio, M. Z. (2014, September). Impact of personality and gender diversity on software development teams' performance. In 2014 International Conference on Computer, Communications, and Control Technology (I4CT) (pp. 261-265). IEEE.
A104	Casey, V. (2011). Imparting the importance of culture to global software development. ACM inroads, 1(3), 51-57.
A105	Wang, Y., & Redmiles, D. (2019, May). Implicit gender biases in professional software development: An empirical study. In 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS) (pp. 1-10). IEEE.
A106	Clarke, L., Pollock, L., Stout, J., Ellis, C., Camp, T., Bizot, B., & McKinley, K. S. (2018, May). Improving diversity in computing research: An overview of CRA-W activities. In 2018 IEEE/ACM 1st International Workshop on Gender Equality in Software Engineering (GE) (pp. 41-44). IEEE.
A107	Al-Ani, B., & Redmiles, D. (2009, July). In strangers we trust? Findings of an empirical study of distributed teams. In 2009 Fourth IEEE International Conference on Global Software Engineering (pp. 121-130). IEEE.
A108	Imtiaz, N., Middleton, J., Chakraborty, J., Robson, N., Bai, G., & Murphy-Hill, E. (2019, May). Investigating the effects of gender bias on GitHub. In 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE) (pp. 700-711). IEEE.

A109	Humayun, M., & Gang, C. (2012, December). Investigating the role of organizational structure in developing shared understanding of requirements within GSD. In 2012 15th International Multitopic Conference (INMIC) (pp. 433-438). IEEE.
A110	Davidson, J. L. (2013, September). Involving older adults in the design and development of free/open source software. In 2013 IEEE Symposium on Visual Languages and Human Centric Computing (pp. 177-178). IEEE.
A111	Davidson, J. L. (2013, September). Involving older adults in the design and development of free/open source software. In 2013 IEEE Symposium on Visual Languages and Human Centric Computing (pp. 177-178). IEEE.
A112	Morrison, P., & Murphy-Hill, E. (2013, May). Is programming knowledge related to age? an exploration of stack overflow. In 2013 10th Working Conference on Mining Software Repositories (MSR) (pp. 69-72). IEEE.
A113	Whittle, J. (2019). Is your software valueless?. IEEE Software, 36(3), 112-115.
A114	Bennaceur, A., Cano, A., Georgieva, L., Kiran, M., Salama, M., & Yadav, P. (2018, May). Issues in Gender Diversity and Equality in the UK. In Proceedings of the 1st International Workshop on Gender Equality in Software Engineering (pp. 5-9).
A115	Samuelsen, T., Colomo-Palacios, R., & Kristiansen, M. (2016, November). Learning software project management in teams with diverse backgrounds. In Proceedings of the Fourth International Conference on Technological Ecosystems for Enhancing Multiculturality (pp. 127-131).
A116	Hersh, M. A., & Stapleton, L. (2006). LearnMaths: A case study of the development of learning software to support social inclusion. IFAC Proceedings Volumes, 39(23), 57-62.
A117	Weilemann, E., & Brune, P. (2015, September). Less distress with a scrum mistress? On the impact of females in agile software development teams. In Proceedings of the ASWEC 2015 24th Australasian Software Engineering Conference (pp. 3-7).
A118	Casey, V. (2009, July). Leveraging or exploiting cultural difference?. In 2009 Fourth IEEE International Conference on Global Software Engineering (pp. 8-17). IEEE.
A119	Draude, C., & Maaß, S. (2018, May). Making IT work: integrating gender research in computing through a process model. In Proceedings of the 4th Conference on Gender & IT (pp. 43-50).
A120	Jablokow, K., & Myers, M. (2010, August). Managing cognitive and cultural diversity in global IT teams. In 2010 5th IEEE International Conference on Global Software Engineering (pp. 77-86). IEEE.
A121	Bosnić, I., Ciccozzi, F., Crnković, I., Čavrak, I., Nitto, E. D., Mirandola, R., & Žagar, M. (2019). Managing diversity in distributed software development education—a longitudinal case study. ACM Transactions on Computing Education (TOCE), 19(2), 1-23.
A122	Diegmann, P. (2017). Measuring the effect of team diversity and collective intelligence in agile teams on software development efficiency.

A123	Eidelman, L., Hazzan, O., Lapidot, T., Matias, Y., Raijman, D., & Segalov, M. (2011). Mind the (gender) gap: can a two-hour visit to a hi-tech company change perceptions about computer science?. ACM Inroads, 2(3), 64-70.
A124	Leonard, S. E., Pearcy, B. M., Shehab, R. L., & Walden, S. E. (2013, October). Minority student informed retention strategies. In 2013 IEEE Frontiers in Education Conference (FIE) (pp. 567-573). IEEE.
A125	Davidson, J. L., Mannan, U. A., Naik, R., Dua, I., & Jensen, C. (2014, August). Older adults and free/open source software: A diary study of first-time contributors. In Proceedings of The International Symposium on Open Collaboration (pp. 1-10).
A126	Gren, L. (2018, May). On gender, ethnicity, and culture in empirical software engineering research. In 2018 IEEE/ACM 11th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE) (pp. 77-78). IEEE.A
A127	Davidson, J. L., Naik, R., Mannan, U. A., Azarbakht, A., & Jensen, C. (2014, July). On older adults in free/open source software: reflections of contributors and community leaders. In 2014 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC) (pp. 93-100). IEEE.
A128	Izquierdo, D., Huesman, N., Serebrenik, A., & Robles, G. (2018). Openstack gender diversity report. IEEE Software, 36(1), 28-33.
A129	Nafus, D. (2012). 'Patches don't have gender': What is not open in open source software. New Media & Society, 14(4), 669-683.
A130	Vasilescu, B., Filkov, V., & Serebrenik, A. (2015, May). Perceptions of diversity on git hub: A user survey. In 2015 IEEE/ACM 8th International Workshop on Cooperative and Human Aspects of Software Engineering (pp. 50-56). IEEE.
A131	Blincoe, K., Springer, O., & Wrobel, M. R. (2019). Perceptions of Gender Diversity's impact on mood in software development teams. IEEE Software, 36(5), 51-56.
A132	Kohl, K., & Prikladnicki, R. (2018, May). Perceptions on diversity in Brazilian agile software development teams: A survey. In 2018 IEEE/ACM 1st International Workshop on Gender Equality in Software Engineering (GE) (pp. 37-40). IEEE.
A133	Licorish, S. A., & MacDonell, S. G. (2014, May). Personality profiles of global software developers. In Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering (pp. 1-10).
A134	Ranganathan, C., & Alfaro, I. (2011). Project Performance in Global Software Development Teams: Do Prior Work Ties and Nationality Diversity Matter?.
A135	Bonhomme-Biais, A., & Romano, R. (2013). Pursuing professional changes. Computer, 46(3), 66-68.
A136	Gotel, O., Kulkarni, V., Say, M., Scharff, C., & Sunetnanta, T. (2012). Quality indicators on global software development projects: does 'getting to know you'really matter?. Journal of Software: Evolution and Process, 24(2), 169-184.

A137	Schloegel, U., Stegmann, S., Maedche, A., & Van Dick, R. (2016). Reducing age stereotypes in software development: The effects of awareness-and cooperation-based diversity interventions. Journal of Systems and Software, 121, 1-15.
A138	Kohl, K., Musse, S. R., Manssur, I., Vieira, R., & Prikladinick, R. (2019). Reinforcing Diversity Company Policies: Insights from StackOverflow Developers Survey. In Proceedings of the 21st International Conference on Enterprise Information Systems,{ICEIS} 2019., 2019, Grécia..
A139	Rastogi, A., Nagappan, N., Gousios, G., & van der Hoek, A. (2018, October). Relationship between geographical location and evaluation of developer contributions in github. In Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (pp. 1-8).
A140	Cunha, A. B., & Canen, A. G. (2008, July). Requirements gathering in information technology: a Cross-cultural perspective. In 2008 IEEE International Professional Communication Conference (pp. 1-8). IEEE.
A141	Spichkova, M., Schmidt, H., & Trubiani, C. (2017, September). Role of women in software architecture: an attempt at a systematic literature review. In Proceedings of the 11th European Conference on Software Architecture: Companion Proceedings (pp. 31-34).
A142	Chilito, P., Viveros, D., Pardo, C., & Pino, F. J. (2018, May). Scrum+: An agile guide for the global software development (GSD) multi-model project management. In 2018 IEEE Colombian Conference on Communications and Computing (COLCOM) (pp. 1-6). IEEE.
A143	Alev, A. T. E. S. (2011). NOTE FOR EDITOR: Self-Efficacy Beliefs, Achievement Motivation And Gender As Related To Educational Software Development. Turkish Online Journal of Distance Education, 12(3), 11-22.
A144	Patwardhan, A. (2017, October). Sentiment identification for collaborative, geographically dispersed, cross-functional software development teams. In 2017 IEEE 3rd International Conference on Collaboration and Internet Computing (CIC) (pp. 20-26). IEEE.
A145	Borsotti, V. (2018, May). SIGSOFT Distinguished Paper-Barriers to Gender Diversity in Software Development Education: Actionable Insights from a Danish Case Study. In 2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET) (pp. 146-152). IEEE.
A146	Aué, J., Haisma, M., Tómasdóttir, K. F., & Bacchelli, A. (2016, September). Social diversity and growth levels of open source software projects on github. In Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (pp. 1-6).
A147	Hill, C. (2016, September). Socio-economic status and computer use: Designing software that supports low-income users. In 2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC) (pp. 266-267). IEEE.
A148	Ahmed, F., Capretz, L. F., Bouktif, S., & Campbell, P. (2012). Soft skills requirements in software development jobs: A cross-cultural empirical study. Journal of systems and information technology.
A149	Marques, M. (2015, October). Software engineering education—Does gender matter in project results?—A Chilean case study. In 2015 IEEE Frontiers in Education Conference (FIE) (pp. 1-8). IEEE.

A150	Pieterse, V., Kourie, D. G., & Sonnekus, I. P. (2006, October). Software engineering team diversity and performance. In Proceedings of the 2006 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries (pp. 180-186).
A151	Alfaro, I., & Chandrasekaran, R. (2015). Software quality and development speed in global software development teams. Business & Information Systems Engineering, 57(2), 91-102.
A152	Liang, T. P., Jiang, J., Klein, G. S., & Liu, J. Y. C. (2009). Software quality as influenced by informational diversity, task conflict, and learning in project teams. IEEE Transactions on Engineering Management, 57(3), 477-487.
A153	Ford, D., Harkins, A., & Parnin, C. (2017, October). Someone like me: How does peer parity influence participation of women on stack overflow?. In 2017 IEEE symposium on visual languages and human-centric computing (VL/HCC) (pp. 239-243). IEEE.
A154	Burgstahler, S., Ladner, R. E., & Bellman, S. (2012). Strategies for increasing the participation in computing of students with disabilities. ACM Inroads, 3(4), 42-48.
A155	Avritzer, A., Beecham, S., Kroll, J., Menasché, D. S., Noll, J., & Paasivaara, M. (2014, August). Survivability models for global software engineering. In 2014 IEEE 9th International Conference on Global Software Engineering (pp. 100-109). IEEE.
A156	Lingard, R., & Berry, E. (2002, November). Teaching teamwork skills in software engineering based on an understanding of factors affecting group performance. In 32nd Annual frontiers in Education (Vol. 3, pp. S3G-S3G). IEEE.
A157	Castro, L. M. (2018, September). Teaching the next generation of software architects: a gender-focused survey on worldwide curricula. In Proceedings of the 12th European Conference on Software Architecture: Companion Proceedings (pp. 1-4).
A158	He, J., Butler, B. S., & King, W. R. (2007). Team cognition: Development and evolution in software project teams. Journal of Management Information Systems, 24(2), 261-292.
A159	Windiarti, I. S., Ferris, T. L., & Berryman, M. J. (2011, December). Technology and knowledge sharing strategy in systems engineering practice performed by Indonesian expatriate engineers. In 2011 IEEE International Conference on Industrial Engineering and Engineering Management (pp. 509-513). IEEE.
A160	Choi, N., & Pruett, J. A. (2015). The characteristics and motivations of library open source software developers: An empirical study. Library & Information Science Research, 37(2), 109-117.
A161	Gibson, W. (2011). The difference engine. Spectra Books.
A162	Daniel, S., Agarwal, R., & Stewart, K. J. (2013). The effects of diversity in global, distributed collectives: A study of open source project success. Information Systems Research, 24(2), 312-333.

A163	Chen, D. N., Shie, Y. J., & Liang, T. P. (2009, August). The impact of knowledge diversity on software project team's performance. In Proceedings of the 11th international conference on electronic commerce (pp. 222-230).
A164	Damian, D. E., & Zowghi, D. (2002, September). The impact of stakeholders' geographical distribution on managing requirements in a multi-site organization. In Proceedings IEEE Joint International Conference on Requirements Engineering (pp. 319-328). IEEE.
A165	Liang, T. P., Wu, J. C. H., Jiang, J. J., & Klein, G. (2012). The impact of value diversity on information system development projects. International Journal of Project Management, 30(6), 731-739.
A166	Grier, D. A. (2006). The Print of the Hand. Computer, 39(8), 8-10.
A167	Tarzimoghadam, S., & Nasle Seraji, J. (2013). The relationship between cognitive ability and demographic charactristics in Tehran computer software engineers. Iran Occupational Health, 10(5), 56-62.
A168	Gheni, A. Y., Jusoh, Y. Y., Jabar, M. A., Ali, N. M., Abdullah, R. H., Abdullah, S., & Khalefa, M. S. (2015, August). The virtual teams: E-leaders challenges. In 2015 IEEE Conference on e-Learning, e-Management and e-Services (IC3e) (pp. 38-42). IEEE.
A169	Hodgson, A., Hubbard, E. M., & Siemieniuch, C. E. (2012). Toward an understanding of culture and the performance of teams in complex systems. IEEE Systems Journal, 7(4), 606-615.
A170	Fendler, J., & Winschiers-Theophilus, H. (2010, May). Towards contextualised software engineering education: an African perspective. In Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1 (pp. 599-607).
A171	Lei, S. (2003, October). Towards programming for the non-technical. In IEEE Symposium on Human Centric Computing Languages and Environments, 2003. Proceedings. 2003 (pp. 291-292). IEEE.
A172	Chinn, D., & VanDeGrift, T. (2008). Uncovering student values for hiring in the software industry. Journal on Educational Resources in Computing (JERIC), 7(4), 1-25.
A173	Saleem, N., Mathrani, S., & Taskin, N. (2019, May). Understanding the different levels of challenges in global software development. In 2019 ACM/IEEE 14th International Conference on Global Software Engineering (ICGSE) (pp. 76-77). IEEE.
A174	Wang, Y. (2018, December). Understanding the Reputation Differences between Women and Men on Stack Overflow. In 2018 25th Asia-Pacific Software Engineering Conference (APSEC) (pp. 436-444). IEEE.
A175	Zeid, A. (2015, October). Using simulation games to teach global software engineering courses. In 2015 IEEE Frontiers in Education Conference (FIE) (pp. 1-9). IEEE.
A176	Ghanbari, H., Similä, J., & Markkula, J. (2015). Utilizing online serious games to facilitate distributed requirements elicitation. Journal of Systems and Software, 109, 32-49.

A177	Morrison, P., Pandita, R., Murphy-Hill, E., & McLaughlin, A. (2016, September). Veteran developers' contributions and motivations: An open source perspective. In 2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC) (pp. 171-179). IEEE.
A178	Berenson, S. B., Slaten, K. M., Williams, L., & Ho, C. W. (2004). Voices of women in a software engineering course: Reflections on collaboration. Journal on Educational Resources in Computing (JERIC), 4(1), 3-es.
A179	Murakami, Y., Tsunoda, M., & Uwano, H. (2017, October). WAP: Does Reviewer Age Affect Code Review Performance?. In 2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE) (pp. 164-169). IEEE.
A180	James, T., Galster, M., Blincoe, K., & Miller, G. (2017, May). What is the perception of female and male software professionals on performance, team dynamics and job satisfaction? Insights from the trenches. In 2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP) (pp. 13-22). IEEE.
A181	Chinn, D., & VanDeGrift, T. (2008). What students say about gender in hiring software professionals. ACM SIGCSE Bulletin, 40(3), 344-344.
A182	Diamant, E. I., Fussell, S. R., & Lo, F. L. (2008, November). Where did we turn wrong? Unpacking the effect of culture and technology on attributions of team performance. In Proceedings of the 2008 ACM conference on Computer supported cooperative work (pp. 383-392).
A183	Lyon, L. A., & Green, E. (2020). Women in coding boot camps: an alternative pathway to computing jobs. Computer Science Education, 30(1), 102-123.
A184	Robles, G., Reina, L. A., González-Barahona, J. M., & Domínguez, S. D. (2016, May). Women in free/libre/open source software: The situation in the 2010s. In IFIP International Conference on Open Source Systems (pp. 163-173). Springer, Cham.
A185	Mahmod, M., & Dahalin, Z. M. (2012). Women in open source software innovation process: Where are they?. Journal of Information & Communication Technology, 11.
A186	Duran, A., De, R., Garcia, I., Giraldo, E., & Castro, M. (2008, October). Work in progress-Agent Based Social Simulations by cross-cultural student teams. In 2008 38th Annual Frontiers in Education Conference (pp. S4E-23). IEEE.
A187	Prikladnicki, R. (2009, July). Exploring propinquity in global software engineering. In 2009 Fourth IEEE International Conference on Global Software Engineering (pp. 133-142). IEEE.
A188	Jorgensen, M., & Grimstad, S. (2012). Software development estimation biases: The role of interdependence. IEEE Transactions on Software Engineering, 38(3), 677-693.
A189	Ford, D., Behroozi, M., Serebrenik, A., & Parnin, C. (2019, May). Beyond the code itself: how programmers really look at pull requests. In 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS) (pp. 51-60). IEEE.
A190	Zweben, S. H., & Bizot, E. B. (2016). Representation of women in postsecondary computing: Disciplinary, institutional, and individual characteristics. Computing in Science & Engineering, 18(2), 40-56.

A191	Moon, E. (2013, January). Gendered patterns of politeness in free/libre open source software development. In 2013 46th Hawaii International Conference on System Sciences (pp. 3168-3177). IEEE.
A192	Iftikhar, A., Alam, M., Musa, S., & Su'ud, M. M. (2017, August). Trust Development in virtual teams to implement global software development (GSD): A structured approach to overcome communication barriers. In 2017 IEEE 3rd International Conference on Engineering Technologies and Social Sciences (ICETSS) (pp. 1-5). IEEE.
A193	Rastogi, A., Nagappan, N., Gousios, G., & van der Hoek, A. (2018, October). Relationship between geographical location and evaluation of developer contributions in github. In Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (pp. 1-8).
A194	Sharp, H., & Robinson, H. (2005, May). Some social factors of software engineering: the maverick, community and technical practices. In Proceedings of the 2005 workshop on Human and social factors of software engineering (pp. 1-6).
A195	Schloegel, U., Stegmann, S., Van Dick, R., & Maedche, A. (2018). Age stereotypes in distributed software development: The impact of culture on age-related performance expectations. Information and Software Technology, 97, 146-162.
A196	Hayashi, J., Higo, Y., Matsumoto, S., & Kusumoto, S. (2019, May). Impacts of daylight saving time on software development. In 2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR) (pp. 502-506). IEEE.
A197	Guzman, E. (2013, September). Visualizing emotions in software development projects. In 2013 First IEEE Working Conference on Software Visualization (VISSOFT) (pp. 1-4). IEEE.
A198	Guzmán, J. G., Ramos, J. S., Seco, A. A., & Esteban, A. S. (2011). Success factors for the management of global virtual teams for software development. International Journal of Human Capital and Information Technology Professionals (IJHCITP), 2(2), 48-59.
A199	Seerat, B., Samad, M., & Abbas, M. (2013, October). Software project management in virtual teams. In 2013 Science and Information Conference (pp. 139-143). IEEE.
A200	Storey, M. A., Zagalsky, A., Figueira Filho, F., Singer, L., & German, D. M. (2016). How social and communication channels shape and challenge a participatory culture in software development. IEEE Transactions on Software Engineering, 43(2), 185-204.
A201	Vasilescu, B. (2014, May). Human aspects, gamification, and social media in collaborative software engineering. In Companion Proceedings of the 36th International Conference on Software Engineering (pp. 646-649).

APÊNDICE B – REFERÊNCIAS DO MSL SOBRE ENSINO DE ENGENHARIA DE SOFTWARE

A1	Calderón, A., Ruiz, M., & O'Connor, R. V. (2018). A multivocal literature review on serious games for software process standards education. Computer Standards & Interfaces, 57, 36-48.
A2	Maurício, R. D. A., Veado, L., Moreira, R. T., Figueiredo, E., & Costa, H. (2018). A systematic mapping study on game-related methods for software engineering education. Information and software technology, 95, 201-218.
A3	Marques, M. R., Quispe, A., & Ochoa, S. F. (2014, October). A systematic mapping study on practical approaches to teaching software engineering. In 2014 IEEE Frontiers in Education Conference (FIE) Proceedings (pp. 1-8). IEEE.
A4	Garousi, V., Giray, G., Tüzün, E., Catal, C., & Felderer, M. (2019). Aligning software engineering education with industrial needs: a meta-analysis. Journal of Systems and Software, 156, 65-83.
A5	Garousi, V., Giray, G., Tuzun, E., Catal, C., & Felderer, M. (2019). Closing the gap between software engineering education and industrial needs. IEEE Software, 37(2), 68-77.
A6	Angelov, S., & de Beer, P. (2017). Designing and applying an approach to software architecting in agile projects in education. Journal of Systems and Software, 127, 78-90.
A7	Salleh, N., Mendes, E., & Grundy, J. (2010). Empirical studies of pair programming for CS/SE teaching in higher education: A systematic literature review. IEEE Transactions on Software Engineering, 37(4), 509-525.
A8	Cico, O., Jaccheri, L., Nguyen-Duc, A., & Zhang, H. (2021). Exploring the intersection between software industry and Software Engineering education-A systematic mapping of Software Engineering Trends. Journal of Systems and Software, 172, 110736.
A9	Brito, M. S., Silva, F. G., G. Chavez, C. V. F., Nascimento, D. C., & Bittencourt, R. A. (2018, September). FLOSS in software engineering education: an update of a systematic mapping study. In Proceedings of the XXXII Brazilian Symposium on Software Engineering (pp. 250-259).
A10	Alhammad, M. M., & Moreno, A. M. (2018). Gamification in software engineering, education: A systematic mapping. Journal of Systems and Software, 141, 131-150.
A11	Petri, G., & von Wangenheim, C. G. (2017). How games for computing education are evaluated? A systematic literature review. Computers & education, 107, 68-90.
A12	Zambon, C., & Thiry, M. (2018, October). Ludic Practices to Support the Development of Software Engineering Educational Games: A Systematic Review. In 2018 XLIV Latin American Computer Conference (CLEI) (pp. 794-802). IEEE.
A13	Prates, J. M., Garcia, R. E., & Maldonado, J. C. (2018, October). MOOCs on the Context of Software Engineering Teaching and Training: Trends and Challenges. In 2018 IEEE Frontiers in Education Conference (FIE) (pp. 1-9). Ieee.

A14	Monasor, M. J., Vizcaíno, A., Piattini, M., & Caballero, I. (2010, August). Preparing students and engineers for global software development: a systematic review. In 2010 5th IEEE International Conference on Global Software Engineering (pp. 177-186). IEEE.
A15	Qadir, M. M., & Usman, M. (2011, December). Software engineering curriculum: A systematic mapping study. In 2011 Malaysian Conference in Software Engineering (pp. 269-274). IEEE.
A16	Garousi, V., Rainer, A., Lauvås Jr, P., & Arcuri, A. (2020). Software-testing education: A systematic literature mapping. Journal of Systems and Software, 165, 110570.
A17	Miyashita, Y., Tanaka, T., & Hazeyama, A. (2018, July). Systematic Literature Review Regarding Communication Support in Project-Based Learning of Software Development. In 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC) (Vol. 1, pp. 781-782). IEEE.
A18	Garcia, I., Pacheco, C., Méndez, F., & Calvo-Manzano, J. A. (2020). The effects of game-based learning in the acquisition of “soft skills” on undergraduate software engineering courses: A systematic literature review. Computer Applications in Engineering Education, 28(5), 1327-1354.
A19	Budgen, D., Drummond, S., Brereton, P., & Holland, N. (2012, June). What scope is there for adopting evidence-informed teaching in se?. In 2012 34th International Conference on Software Engineering (ICSE) (pp. 1205-1214). IEEE.
A20	Marques, M., & Robledo, J. (2018, October). What software engineering “best practices” are we teaching students-a systematic literature review. In 2018 IEEE Frontiers in Education Conference (FIE) (pp. 1-8). IEEE.

APÊNDICE C – PROTOCOLO DE PESQUISA PARA O USO DE TÉCNICAS DE ANÁLISE DE DADOS DE GROUNDED THEORY

Título: Ensinando Engenharia de Software online para pessoas em situação de vulnerabilidade social
Aluna/Pesquisadora: Anielle Severo Lisboa
Professor Orientador: Prof. Rafael Prikladnicki (PUCRS)

C.1 Objetivo

Este protocolo apresenta um planejamento para o uso de técnicas de *Grounded Theory* (Teoria Fundamentada nos dados) para análise de dados qualitativos, por meio de entrevistas semiestruturadas em pessoas professoras e estudantes de um curso de capacitação em Engenharia de Software. Após as primeiras coletas de dados e analisados, o protocolo pode ser ajustado, se necessário, pois o método será iterativo (progresso através de refinamentos) e incremental (com a ideia inicial, executa-se o que pode possibilidade a evolução). O foco é entender como a Engenharia de software é ensinada para pessoas da diversidade social em contexto de vulnerabilidade social durante a pandemia

C.2 Referencial Teórico

Para este protocolo foram utilizados como base teórica os seguintes estudos:

1. *Developing a grounded theory to explain the practices of self-organizing Agile teams* (Hoda, 2012);
2. *Using grounded theory to study the experience of software development (Adolph, 2011)*;
3. Mapeamento sistemático da literatura desenvolvido pela autora deste estudo, sobre Diversidade na Engenharia de Software (Lisboa, 2020);
4. Mapeamento sistemático da literatura em revisões e mapeamentos sistemáticos da literatura desenvolvido pela autora deste estudo, sobre Ensino em Engenharia de Software (Lisboa, 2020);

C.3 Método de Pesquisa

Para este estudo foi realizado um mapeamento sistemático da literatura sobre Diversidade em Engenharia de Software e outro sobre Ensino em Engenharia de Software. As fases de 1 a 7, já estão finalizadas, conforme demonstradas no esquema da Figura C.1. Este protocolo de estudo foi planejado para a fase 8 e 9 (fase final) de mestrado da autora desse estudo.

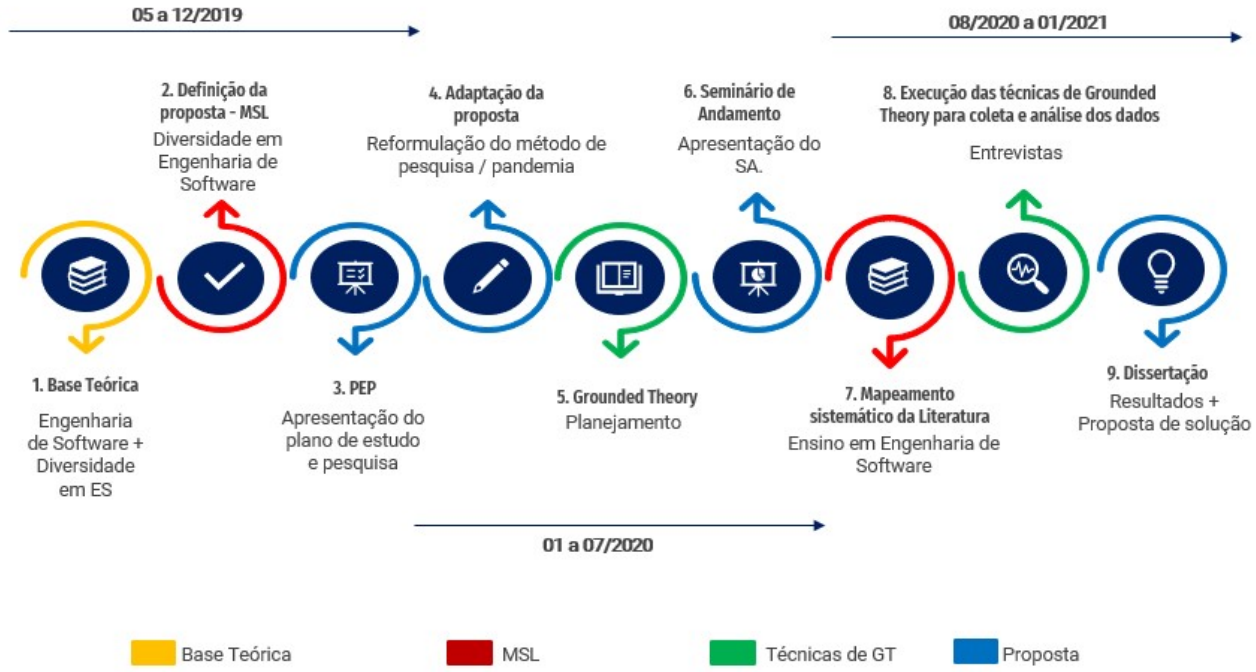

Figura C.1 – Cronograma de Execução do estudo
Fonte: a autora (2020)

A Figura C.2 apresenta o fluxo e ordem das atividades por meio da coleta de dados em entrevistas semiestruturadas e a utilização de técnicas para analisar os de *Grounded Theory*, com base no estudo de Hoda et al. [46]. As atividades em detalhes - como os dados foram coletados e analisados (as atividades na cor cinza representam as técnicas de análise de GT) pelos autores deste estudo, são apresentadas nas Seções à seguir.

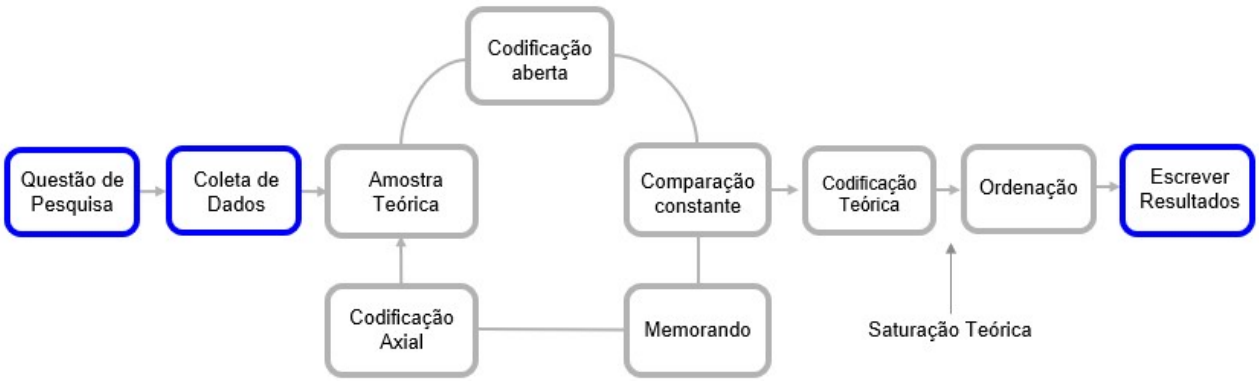

Figura C.2 – Fases das técnicas de *Grounded Theory* para coleta e análise de dados
Fonte: a autora (2020)

C.3.1 Área de Interesse e a Questão de pesquisa

Para descobrir e estudar os principais problemas dos participantes, recomenda-se que a pessoa pesquisadora se abstenha de formular um problema ou pergunta de pesquisa com antecedência [40]. O problema de pesquisa deve ser o problema dos participantes do estudo e não deve ser preconcebido ou forçado, mas deve ser permitido surgir [40]. Embora a pessoa pesquisadora não seja aconselhado a formular uma questão de pesquisa, ele deverá escolher uma área de interesse geral. A autora deste estudo desenvolveu o seguinte esquema da área de interesse, com base nos estudos realizados até agora, por meio de dois mapeamentos sistemáticos da literatura, conforme a Figura C.3:

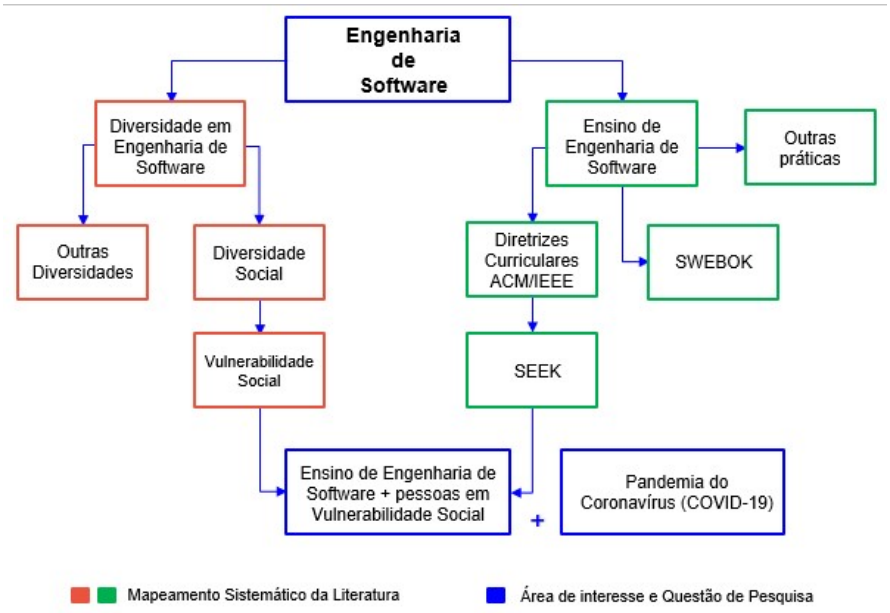

Figura C.3 – Fluxograma para a área de interesse e questão de pesquisa
Fonte: a autora (2020)

C.3.2 Questão de pesquisa

Como a Engenharia de Software é ensinada online para pessoas em vulnerabilidade social?

C.4 Coleta de Dados

A técnica selecionada para a coleta de dados foi o da entrevista semiestruturada, que permite, ao mesmo tempo, a liberdade de expressão da pessoa entrevistada e manutenção do foco pela pessoa entrevistadora [11]. Ela é executada por um roteiro de perguntas previamente estabelecidas (Apêndice D).

Na próxima Subseções serão apresentados a seleção de participantes dos estudos e o planejamento para a realização das entrevistas.

C.4.1 Seleção dos participantes

Serão realizadas em média 30 entrevistas entre pessoas professoras e estudantes de um ambientes de aprendizagem em Engenharia de Software. Esses cursos de capacitação estão ocorrendo de forma online, devido a pandemia da COVID-19. A duração das capacitações dura em média 18 semanas. Os professores e alunos serão entrevistados nesse período.

Para todos os casos, o participante inicia sua participação no estudo somente após ser informado e concordar com as condições da atividade conforme declarado no Termo de Consentimento Livre e Esclarecido - TCLE (Ver Apêndice E), o qual será discutido com o participante antes da atividade de coleta de dados (entrevistas) iniciarem, o participante é livre para se retirar da atividade a qualquer momento, sem haver necessidade de explicitar razões para isso.

C.4.2 Entrevistas semiestruturadas

Serão realizadas com duração mínima de 30 minutos e no máximo de 1h, com foco nas experiências vivenciadas no percurso dos participantes nos cursos de capacitação em Engenharia de Software, em particular, nas formas de ensino, durante a pandemia do Coro-

navírus. Para esta coleta de dados, foi desenvolvido um roteiro de perguntas, apresentado no Apêndice D.

Esse protocolo é iterativo e incremental, ou seja, será ajustado de acordo com a necessidade identificadas nas coletas e análises de dados iniciais. Pretende-se coletar percepções sobre o ensino e os desafios encontrados dos participantes, durante as atividades de ensino e aprendizagem.

C.5 Análise dos dados

Nesta fase do estudo qualitativo, foram utilizadas as técnicas de análise de dados da *Grounded Theory* nos dados obtidos nas entrevistas. As subseções a seguir apresentam os mecanismos de codificação, como código aberto, codificação seletiva, teórica, método de comparação constante e memorandos.

C.5.1 Codificação Aberta

A codificação aberta é o primeiro passo da análise de dados em uma *Grounded Theory*. Corbin e Strauss [25] conceituam a codificação aberta como o processo analítico pelos quais os conceitos são identificados e desenvolvidos em relação às suas propriedades e suas dimensões. Esse processo envolve as atividades de quebrar, examinar, comparar, conceituar e categorizar os dados que serão sumarizados em uma lista de códigos e categorias originadas dos rótulos atribuídos livremente a cada frase, linha ou parágrafo.

À medida que novos casos são estudados em detalhes, alguns deles serão semelhantes aos já analisados (se enquadram no mesmo conceito), mas ao mesmo tempo são diferentes de alguma forma. A cada captura dessas variações, os conceitos são continuamente revisados, por exemplo, pela introdução de novas propriedades, o que, por sua vez, exige que a pessoa pesquisadora revisite exemplares já analisados, para ver se o conceito alterado ainda se ajusta a ele ou qual é seu valor específico em relação a uma propriedade recém-introduzida [25].

C.5.2 Comparação constante

Corbin e Strauss [25] mencionam que os segmentos particulares dos dados não são analisados e codificados uma vez, mas são comparados repetidamente com novos dados. O mesmo é feito no nível de conceito, procurando sempre diferenças e semelhanças

relevantes. Os códigos decorrentes de cada entrevista foram constantemente comparados com os códigos da mesma entrevista, e de outras entrevistas. Isto é Método de comparação constante de *Grounded Theory* [40, 41], onde foram agrupados para produzir um nível mais alto de abstração, chamando de conceito em teoria fundamentada nos dados (*Grounded Theory*).

C.5.3 Memorando

Glaser [40] classifica como um processo contínuo de escrever memorandos teóricos em toda a teoria fundamentada em dados. Memorandos são notas teóricas sobre os dados e as conexões conceituais entre as categorias escritas à medida que atingem a pessoa pesquisadora. São considerados como o estágio central ou a base da geração da teoria. Memorandos podem ser ideias fluentes da pessoa pesquisadora sobre os códigos e seus relacionamentos e não devem ser misturados com dados. Os memorandos são escritos a partir das ideias sobre os códigos emergentes e seus relacionamentos. Como recomendado por Glaser [40], muitas vezes a codificação e outras atividades para capturar as ideias por meio do memorando, eram interrompidas, à medida que chegavam até as pesquisadoras.

C.5.4 Codificação Seletiva

Uma vez que a categoria principal é estabelecida, a pesquisadora pausa a codificação aberta e passa para a codificação seletiva, um processo que envolve a codificação seletiva para o variável central, delimitando a codificação para apenas as variáveis que se relacionam com o variável central de maneiras suficientemente significativas para produzir uma teoria parcimoniosa [40]. A categoria principal orienta a coleta de dados, análise e amostragem teórica [40]. Quanto mais coletar e analisar dados, em uma categoria particular leva a um ponto de resultados decrescentes, nesse caso, a categoria teria atingido a Saturação Teórica [41]. O pesquisador pode parar de coletar dados e codificar para aquela categoria.

C.5.5 Codificação Teórica

Nesta etapa ocorre a fase final da codificação [41], conhecida também como Codificação Teórica (códigos teóricos). A codificação teórica é definida como a propriedade da

codificação e análise comparativa constante. Ela é responsável pela produção da relação conceitual entre as categorias e suas propriedades a cada surgimento [41].

Glaser lista várias estruturas comuns de teorias conhecidas como grupos teóricos [41, 42]. Alguns deles incluem: Os seis C's (causas, contextos, contingências, consequências, covariâncias e condições); Processo (etapas, fases, passagens, etc); Graus por grupo (limite, alcance, intensidade, etc.) Grupo por dimensão (dimensões, elementos, divisões, etc.) Tipos de Grupo (tipo, forma, crianças, estilos, classes, gênero) e muito mais.

De acordo com as recomendações de Glaser [41], a codificação teórica foi usada na última fase de análise dos dados. Esta codificação foi a solução adequada para descrever a descoberta de práticas de ensino de Engenharia de Software.

O desafio da pessoa pesquisadora de Engenharia de Software é se acostumar a pensar de forma teórica - os resultados anteriores, não podem tornar isso visível. A fim de superar este problema, deve-se considerar que os códigos teóricos são baseados na classificação de memorandos e não em dados [42]. A possibilidade de deixar em aberto os conceitos e categorias emergentes - ler sobre diferentes teóricos códigos, tornam as pessoas pesquisadoras sensíveis ao ver os códigos teóricos.

C.5.6 Ordenação

Quando a pessoa pesquisadora estava quase terminando a coleta de dados com todos os entrevistados, iniciou-se o processo de classificação conceitualmente dos memorandos teóricos. A classificação dos memorandos constitui um esboço teórico. A classificação é uma etapa essencial e não pode ser desconsiderada [40]. A vantagem desta classificação é a organização e união dos dados fragmentados, novamente [40]. A pesquisadora deve ter em mente que os dados não precisam ser classificados, e sim as ideias. Deve-se evitar a classificação dos memorandos em ordem cronológica. A classificação deve ser feita em um nível conceitual, resultando em um esboço da teoria - como as diferentes categorias relacionam-se com a categoria principal.

Foram coletados os detalhes de todos os memorandos. Foram classificados por tópicos para que os mesmos fossem ordenados um após o outro. Logo, foi gerado um esboço da teoria usando os nomes dos tópicos na mesma ordem. Este esboço serviu como base para a proposta deste estudo.

O maior desafio envolvido na classificação dos memorandos é que, embora seja muito mais fácil agrupar memorandos relacionados, a ordem dos memorandos talvez não seja nitidamente óbvia. Foi necessário embaralhar alguns memorandos e pensar nos relacionamentos entre os diferentes tópicos de memorando para encontrar uma ordem que fizesse mais sentido. As pessoas autoras deste estudo resolveram traçar as relações entre

as diferentes categorias de forma manual. Uma vez que as relações foram estabelecidas em um diagrama (usando linhas para conectar as categorias), foi mais fácil identificar como os memorandos (cobrindo diferentes categorias e conceitos) foram relacionados.

C.5.7 Escrever os Resultados

A etapa final da pesquisa qualitativa é a redação dos resultados, que segue o esboço teórico gerado como resultado de classificação e codificação teórica. Os resultados deste trabalho giram em torno do ensino de Engenharia de software online para pessoas em situação de vulnerabilidade social.

APÊNDICE D – INSTRUMENTOS PARA A COLETA DE DADOS: ENTREVISTAS SEMIESTRUTURADAS

Esse Roteiro foi adaptado 3x em encontro com os resultados analisados após as entrevistas. Foram realizadas 24 entrevistas no total. A Figura D.1 apresenta os instrumentos de coleta de dados.

Instrumento	Roteiro Semiestruturado
Formato	• Roteiro semiestruturado que será usado para: entrevista remota, via vídeo conferência • Com duração prevista de 30-60 minutos, incluindo a leitura do TCLE e introdução do estudo que está sendo realizado.
Objetivo	• Como a Engenharia de Software é ensinada em tempos de pandemia para pessoas em vulnerabilidade social?
Participantes	• Serão recrutados em torno de 24 voluntários para participarem das entrevistas semiestruturadas; • Os participantes desse estudo são alunos e professores dos cursos de capacitação em Engenharia de Software;
Identificação	Questões para mapear o perfil do participante e do curso 1. Nome 2. Idade 3. Nível de escolaridade 4. Gênero 5. Raça/Etnia (resposta opcional) 6. Status Socioeconômico (resposta opcional)

Figura D.1 – Instrumento de coleta de dados: Entrevista Semiestruturadas
Fonte: a autora (2020)

A seguir são apresentados a quantidade de entrevistas para cada roteiro e suas incrementações.

D.1 Roteiro de Entrevistas - RODADA 1

O roteiro inicial (RODADA 1), validado com o Orientador, foram realizadas 5 entrevistas. A Figura D.2 apresenta o roteiro inicial de entrevistas.

1ª RODADA	
Participante	Identificação
	Idade
	Grau de Instrução
	Gênero, Raça/Etnia, Status Socioeconômico
Curso	Sobre
	Experiência
Pandemia	Adaptação
	Como?
	Facilidades
	Dificuldades
	Diferença entre o presencial e o remoto?
	Plataformas de videoconferencia
	Equipamentos
	Conexão com internet
	Comunicação
Como é ensinado? A Engenharia de Software?	Fundamentos da Computação
	Fundamentos de matemática e Engenharia
	Prática Profissional
	Modelagem e Análise de Software
	Análise e Especificação de Requisitos
	Design de Software
	Verificação e Validação de Software
	Qualidade de Software
	Processo de Software
	Segurança

Figura D.2 – Roteiro de Entrevistas semiestruturadas (RODADA 1)
Fonte: a autora (2020)

D.2 Roteiro de Entrevistas - RODADA 2

Com o roteiro da rodada 2, foram realizadas 5 entrevistas. A Figura D.3 apresenta o roteiro de entrevistas.

2ª RODADA	
Participante	Identificação
	Idade
	Grau de Instrução
	Gênero, Raça/Etnia, Status Socioeconômico
Curso	Sobre
	Papel
	Tempo
	Área
	Empresa
Pandemia	Adaptação
	Como?
	Facilidades
	Dificuldades
	Diferença entre o presencial e o remoto?
	Plataformas de videoconferencia
	Equipamentos
	Conexão com internet
	Comunicação
Como é ensinado? A Engenharia de Software?	Fundamentos da Computação + Fundamentos de matemática e Engenharia
	Prática Profissional
	Modelagem e Análise de Software + Análise e Especificação de Requisitos
	Design de Software
	Verificação e Validação de Software + Qualidade de Software
	Processo de Software
	Segurança
	Outros

Figura D.3 – Roteiro de Entrevistas semiestruturadas (RODADA 2)
Fonte: a autora (2020)

D.3 Roteiro de Entrevistas - RODADA 3

Com o roteiro da rodada 3, foram realizadas 14 entrevistas. A Figura D.4 apresenta o roteiro final das entrevistas.

3ª RODADA	
Participante	Identificação
	Idade
	Grau de Instrução
	Gênero, Raça/Etnia, Status Socioeconômico
Curso	Sobre
	Papel
	Tempo
	Área
	Empresa
Pandemia	Como?
	Facilidades
	Dificuldades
	Diferença entre o presencial e o remoto?
	Aquisição de Equipamentos ou conexão?
Como é ensinado a seguinte área de conhecimento em	Fundamentos da Computação + Fundamentos de matemática e Engenharia
	Prática Profissional
	Modelagem e Análise de Software + Análise e Especificação de Requisitos
	Design de Software
	Verificação e Validação de Software + Qualidade de Software
	Processo de Software
	Segurança
	Outros informações

Figura D.4 – Roteiro de Entrevistas semiestruturadas (RODADA 3)
Fonte: a autora (2020)

APÊNDICE E – TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO (TCLE)

Nós, Anielle Severo Lisboa (aluna de mestrado) e Rafael Prikladnicki (professor orientador), responsáveis pela pesquisa “Ensinando Engenharia de Software online para pessoas em situação de vulnerabilidade social”, estamos fazendo um convite para você participar como voluntário nesse estudo. Esta pesquisa pretende observar como a Engenharia de Software é ensinada online para pessoas em situação de vulnerabilidade social. Acreditamos que ela seja importante pois tem como objetivo prover uma avaliação qualitativa de dados coletados em campo, criando uma base de conhecimento com informações analisadas através de estratégias com rigor metodológico.

Para sua realização será feito o seguinte: através do uso técnicas de teoria fundamentada em dados *(Grounded Theory)*, serão estudados dados e experiências nos ambientes de aprendizagem em engenharia de software. A coleta de dados será iterativa e incremental.

Sua participação constará em entrevistas semiestruturadas e observações. Lembrando que o objetivo deste estudo não é avaliar o participante, mas, sim, analisar as práticas de ensino relacionados ao tópico da pesquisa. Os benefícios que esperamos como estudo são através de dados coletados em campo, criar uma base de conhecimento com informações analisadas através de estratégias com rigor metodológico; um protocolo detalhado do estudo que pode ser usado para a replicação do estudo em ambientes de aprendizagem em engenharia de software, possibilitando assim, o enriquecimento do estudo e a possibilidade de contribuir com a comunidade de engenharia de software e de futuros profissionais.

Durante todo o período da pesquisa você tem o direito de esclarecer qualquer dúvida ou pedir qualquer outro esclarecimento, bastando para isso entrar em contato, com: Rafael Prikladnicki no e-mail: Rafael.prikladnicki@pucrs.br ou Anielle Severo Lisboa no telefone (55) 99638 8112, e-mail: (anielle.lisboa@edu.pucrs.br) a qualquer hora.

Você tem garantido o seu direito de não aceitar participar ou de retirar sua permissão, a qualquer momento, sem nenhum tipo de prejuízo ou retaliação, pela sua decisão.

As informações desta pesquisa serão confidenciais, e serão divulgadas apenas em eventos ou publicações científicas, não havendo identificação dos participantes, a não ser entre os responsáveis pelo estudo, sendo assegurado o sigilo sobre sua participação

Ao assinar este termo de consentimento, você não abre mão de nenhum direito legal que teria de outra forma.

Não assine este termo de consentimento a menos que tenha tido a oportunidade de fazer perguntas e tenha recebido respostas satisfatórias para todas as suas dúvidas. Se

você concordar em participar deste estudo, você rubricará todas as páginas e assinará e datará duas vias originais deste termo de consentimento. Você receberá uma das vias para seus registros e a outra será arquivada pelo responsável pelo estudo.

Eu, _____, após a leitura (ou a escuta da leitura) deste documento e de ter tido a oportunidade de conversar com o pesquisador responsável, para esclarecer todas as minhas dúvidas, acredito estar suficientemente informado, ficando claro que minha participação é voluntária e que posso retirar este consentimento a qualquer momento sem penalidades ou perda de qualquer benefício. Estou ciente também dos objetivos da pesquisa, dos procedimentos aos quais serei submetido, dos possíveis danos ou riscos deles provenientes e da garantia de confidencialidade e esclarecimentos sempre que desejar.

Diante do exposto expresso minha concordância de espontânea vontade em participar deste estudo.

Assinatura do participante da pesquisa ou de seu representante legal

---

Assinatura de uma testemunha

DECLARAÇÃO DO PARTICIPANTE QUE OBTEVE O CONSENTIMENTO

Expliquei integralmente este estudo ao participante. Na minha opinião e na opinião do participante , houve acesso suficiente às informações, incluindo riscos e benefícios, para que uma decisão consciente seja tomada.

Data: _____

Assinatura do Investigador

Pontifícia Universidade Católica do Rio Grande do Sul
Pró-Reitoria de Graduação
Av. Ipiranga, 6681 - Prédio 1 - 3º. andar
Porto Alegre - RS - Brasil
Fone: (51) 3320-3500 - Fax: (51) 3339-1564
E-mail: prograd@pucrs.br
Site: www.pucrs.br